
Math Book

Lê Quốc Dũng

18/08/2025

Contents

1	Giới thiệu	1
1.1	Những lời nói đầu	1
2	Toán khó quá người ơi	13
2.1	Đại số	13
2.2	Lý thuyết nhóm	35
2.3	Hình học	77
2.4	Giải tích	131
2.5	Xác suất thống kê	157
2.6	Đại số tuyến tính	176
2.7	Số học	189
2.8	Toán rời rạc	207
3	Mật mã học cũng khó	251
3.1	Đại số Bool và mật mã học	251
3.2	Phá mã	312
3.3	Mật mã học	353
4	Lời giải cho những vấn đề	419
4.1	Lời giải các bài tập trong sách tham khảo	419
4.2	Các cuộc thi năm 2020	459
4.3	Các cuộc thi năm 2021	462
4.4	Các cuộc thi năm 2022	473
4.5	Các cuộc thi năm 2023	479
4.6	Các cuộc thi năm 2024	519
4.7	Các cuộc thi năm 2025	589
5	Bên lề sách của Fermat	609
5.1	Ngoài lề	609
5.2	Nước Nga du kí	618
5.3	Sưu tầm những câu chuyện về các nhà toán học	622
5.4	Men of Mathematics	622
6	Tài liệu tham khảo	631
	Bibliography	633

Giới thiệu

Notebook về toán ...

Các bạn có thể tải phiên bản PDF của notebook này ở [mathbook.pdf](#).

1.1 Những lời nói đầu

1.1.1 Toán học là gì với mình?

Mình không giỏi toán. Tuy nhiên toán lại là môn mình dành nhiều thời gian nhất trong suốt 10 năm (2015 tới 2025). Đây là mối lương duyên không tầm thường, nhưng cũng không hề lãng mạn tí nào.

Những ngày đầu học toán

Mình bắt đầu học toán nhiều từ cấp 2 nhưng cũng không quá đặc sắc. Mình chỉ đơn giản là làm nhiều bài tập hơn, nhiều dạng hơn thôi. Mình may mắn được học với thầy cô tâm huyết, cũng như các bạn học cùng chí hướng. Nhờ các cô dạy toán và bạn bè mà mình tiến bộ, biết thêm nhiều kiến thức mới (đặc biệt từ chỗ mấy bạn học thêm ở trung tâm 218 ^^).

Năm 2015 mình vào lớp 10 chuyên toán. Sau ba năm thì mình nhận ra mình học toán rất ... tệ, nhưng mình cũng nhận ra đam mê mạnh mẽ của mình cho môn toán. Mình rất biết ơn các thầy, cô trong trường, luôn giúp đỡ và động viên mình, cũng như cho mình thấy được nhiều lối tư duy khác nhau. Đặc biệt, chuyên toán học toán theo cách khác với những lớp khác mà tư duy đó mình vẫn sử dụng tới tận hiện nay.

Thầy chủ nhiệm lớp mình suốt 3 năm là giáo viên toán. Mình kính trọng thầy không chỉ vì thầy giỏi, mà thầy dạy mình cách học toán sao cho đúng. Thông thường học sinh chúng mình bám theo các tính chất để giải bài toán. Ví dụ, đường phân giác của một góc thường được hiểu là đường chia đôi góc đó. Tuy nhiên thầy nói đó là tính chất, không phải định nghĩa. Đường phân giác của góc là **tập hợp các điểm cách đều hai cạnh của góc**.

Càng học nhiều toán thì mình càng nhận thấy định nghĩa là nơi bắt đầu mọi thứ, là điểm quan trọng nhất của các vấn đề toán học. Các tính chất giúp chúng ta tìm lời giải nhanh hơn, nhưng khi bế tắc thì định nghĩa là nơi để chúng ta bám vào và tìm hướng giải. Trong trường hợp hình học, mỗi đối tượng hình học đều là **một tập hợp điểm**. Do đó chúng ta có thể nói **điểm** là đơn vị cơ bản nhất. Các định nghĩa đóng vai trò xây dựng nền tảng toán học mà về sau mình mới hiểu được ý nghĩa của chúng và hậu quả nếu chúng "có vấn đề". Phần sau mình sẽ nói rõ hơn về điều này.

Một điều quan trọng mình được thầy nhắc là **không nên bám vào các phương pháp giải toán**. Thông thường các sách tham khảo, khóa học ở cấp 3 sẽ theo dạng Bài 1. Chủ đề A. Phương pháp 1 rồi lại Bài 2. Chủ đề A. Phương pháp 2. Theo mình, các phương pháp rất hữu ích khi gấp bài đúng dạng, chỉ cần

"áp dụng công thức" là xong. Tuy nhiên việc học theo phương pháp cũng là con dao hai lưỡi. Nếu chúng ta gặp dạng chưa học thì khả năng cao là chịu chết. Việc đọc các phương pháp giải bài là cần thiết để có nhiều phương án giải quyết khác nhau, nhưng quan trọng là khả năng tư duy để kết nối những cái mới gắp với những điều đã biết. Một bài hình học gồm dữ kiện và học sinh cần giải quyết bốn câu hỏi nhỏ (a), (b), (c), (d). Đây thường là cấu trúc của câu hình học trong đề tuyển sinh lớp 10 ở thời mình. Tất nhiên là dù đề chung hay đề thi chuyên toán luôn có những bạn làm trọn vẹn bốn câu. Vấn đề là, khi cắt bỏ ba câu hỏi (a), (b), (c) và chỉ bảo giải câu (d) thì bài toán trở thành tầm quốc gia, quốc tế (VMO, IMO). Lý do rất đơn giản, những câu (a), (b), (c), (d) được tăng dần theo độ khó và câu trước thường là tiền đề để giải hoặc làm gợi ý cho câu sau. Khi chúng ta mất gợi ý thì chúng ta sẽ tiếp cận ra sao để giải quyết câu (d)? Đây chính là thực tế của toán học nói chung. Nhiều công thức, định lí được phát biểu đơn giản, ngắn gọn như câu (d) của đề tuyển sinh, nhưng để chứng minh chúng thì mất mấy trăm năm nỗ lực của con người chứ không phải một lè sách là đủ.

Hai điều thầy nói có thể thấy rõ trong các cuộc thi olympiad toán, và cũng là thực tế trong toán học. Câu khó thường có dạng:

Số A được gọi là số như này nếu nó thỏa mãn các điều kiện như kia. Hãy chứng minh nếu A là số như này thì nó sẽ có các tính chất như nọ.

Đây là một dạng toán rất khó nhẫn vì chúng ta đụng phải một khái niệm, định nghĩa mới lạ. Lúc này chúng ta phải bám sát định nghĩa vì đó là thứ duy nhất để cho chúng ta để chứng minh tính chất. Vấn đề là chỉ với mỗi định nghĩa không giúp chúng ta tìm phương pháp giải. Việc dựa trên các phương pháp đã biết còn tùy vào chúng ta liên kết được định nghĩa đó với kiến thức, kinh nghiệm và trải nghiệm của bản thân tới mức nào. Điều thú vị là kiến thức, kinh nghiệm và trải nghiệm có thể được nâng cao nhờ sự chăm chỉ và rèn luyện "trực giác". Đối với người không giỏi toán như mình thì việc nâng khả năng của trực giác rất có lợi, cho phép mình "phán đoán" cách tiếp cận sẽ đưa tới lời giải. Làm càng nhiều, sự thấu hiểu và cảm nhận của mình với những bài toán càng nhạy.

Một giáo viên dạy toán khác của lớp mình cũng đã để lại nhiều bài học quý giá. Một lần nữa, câu chữ trong toán học rất quan trọng và chúng ta phải cẩn thận khi xử lý. Các bạn có thể thấy hai cách viết $\mathbb{R} \setminus \{0\}$ và $(-\infty, 0) \cup (0, +\infty)$ giống nhau vì đều chỉ việc bỏ số 0 khỏi tập số thực \mathbb{R} . Tuy nhiên khi xét đến **các khoảng (đoạn) xác định của hàm số** thì đây là vấn đề rất quan trọng. Khi xét tính đồng biến, nghịch biến bằng đạo hàm, giả sử hàm số $f(x) = \frac{1}{x}$ với đạo hàm $f'(x) = -\frac{1}{x^2} < 0$ với mọi $x \in \mathbb{R} \setminus \{0\}$. Khi đó chúng ta có thể nói rằng **hàm số $f(x)$ xác định trên tập $\mathbb{R} \setminus \{0\}$** . Tuy nhiên, nếu chúng ta kết luận rằng $f(x)$ nghịch biến trên $\mathbb{R} \setminus \{0\}$ thì đây là kết luận **sai hoàn toàn**. Kết luận đúng phải dựa trên các khoảng xác định, nghĩa là **hàm số nghịch biến trên hai khoảng $(-\infty, 0)$ và $(0, +\infty)$** . Ở [1] (trang 220, định lí 2) ghi rằng:

Giả sử hàm số f có đạo hàm trên khoảng \mathbb{I} . Khi đó

- Nếu $f'(x) > 0$ với mọi $x \in \mathbb{I}$ thì f đồng biến trên \mathbb{I} .
- Nếu $f'(x) < 0$ với mọi $x \in \mathbb{I}$ thì f nghịch biến trên \mathbb{I} .
- Nếu $f'(x) = 0$ với mọi $x \in \mathbb{I}$ thì f có giá trị không đổi trên \mathbb{I} .

Ở đây, sách giáo khoa ghi rõ ràng khoảng xác định mà trên đó hàm số có đạo hàm, nghĩa là không được nói $f(x)$ ở trên nghịch biến trên một khoảng "hut" ở giữa như $(-\infty, 0) \cup (0, +\infty)$ hay $\mathbb{R} \setminus \{0\}$, mà phải là nghịch biến trên hai khoảng xác định: $(-\infty, 0)$ và $(0, +\infty)$.

Một vấn đề quan trọng dễ bị sai sót là các mệnh đề "**Nếu** như này **thì** như kia". Theo ngôn ngữ logic thì đây là phép kéo theo.

Phép kéo theo "Nếu P thì Q " hay " $P \Rightarrow Q$ ", trong đó P và Q là hai mệnh đề, sai khi P đúng và Q sai, và đúng trong các trường hợp còn lại.

Thông thường có hai điểm dễ bị hiểu nhầm hoặc cố tình hiểu nhầm (khi đánh tráo khái niệm, bẻ cong logic, ...).

1. Đây là phép kéo theo, nghĩa là phải hội tụ đủ tất cả điều kiện P thì chúng ta mới có Q .
2. Khi phủ định hai mệnh đề ta không thu được mệnh đề cùng tính đúng sai với mệnh đề ban đầu, tức là khi có \overline{P} chưa chắc có \overline{Q} . Minh thấy việc này xảy ra gần như trong tất cả lĩnh vực không riêng toán, không biết do vô tình hay cố ý. Ví dụ mình có mệnh đề "Nếu trời mưa thì đường ướt". Như vậy có thể suy ra "Nếu trời không mưa thì đường không ướt"? Câu trả lời là **chưa chắc**, đường vẫn có thể ướt vì lý do khác, ví dụ như nước bị rò rỉ ở đường ống gần đó.

Về mặt logic thì phép kéo theo $P \Rightarrow Q$ sẽ **hoàn toàn tương đương** phép kéo theo $\overline{Q} \Rightarrow \overline{P}$, trong đó \overline{P} và \overline{Q} là mệnh đề phủ định của P và Q . Với ví dụ trên, chúng ta suy ra được "Nếu đường không ướt thì trời không mưa". Mệnh đề này có cùng giá trị logic với mệnh đề ban đầu "Nếu trời mưa thì đường ướt": đúng thì đúng chung, sai thì sai chung. Đây là vấn đề khi lên đại học và đổi chiêu sách vở mình rút ra được. Ở cấp 3 mình chỉ được nhắc rằng việc phủ định chưa chắc đúng còn lý do đằng sau (như mình vừa trình bày) thuộc khuôn khổ nội dung về logic và được giảm tải ở thời mình. Khi lên đại học thì môn toán rời rạc thường sẽ có phần cơ sở logic chính là nền tảng của phép kéo theo.

Đại học, tình yêu không thành với toán

Khi được tiếp cận các kiến thức toán cao hơn ở đại học thì mình nhận ra một vấn đề thú vị của bản thân: mình học toán rời rạc (lý thuyết số, hàm Boolean, lý thuyết nhóm) tốt hơn toán liên tục (các loại giải tích). Hệ quả là mình dở tệ môn vật lý vì nhiều kiến thức về giải tích được áp dụng để giải quyết các bài toán vật lý. Artificial Intelligence (AI) và Machine Learning (ML) là những từ khóa "hot trend" nhất lúc mình học đại học, nhưng vì khả năng toán liên tục yếu kém mà mình sớm đuối sức và không đi sâu nữa. Tuy nhiên mình vẫn có hy vọng sẽ hiểu một ngày nào đó ^-^. Do đó thay vì AI/ML mình đã chọn theo mật mã học (cryptography), cụ thể là mã hóa khóa đối xứng (symmetric key cryptography) dựa trên các hàm Boolean. Minh thấy mình theo lĩnh vực này bớt thảm hơn AI/ML nên là mình theo tới tận bây giờ (2025).

Khi lên đại học, các kiến thức trở nên phức tạp hơn cả về bề rộng lẫn chiều sâu. Chúng ta thường sẽ học nhiều kiến thức hơn (mở rộng) nhưng đồng thời học sâu bản chất (chiều sâu). Minh thích ham vui nên mình học rất nhiều loại toán: giải tích, xác suất thống kê, hình học giải tích, hình học affine, hình học phi Euclid, lý thuyết số, đại số Boolean, đại số tuyến tính, lý thuyết đồ thị, lý thuyết tập hợp, ...

Đa phần trong đó là ... cưỡi ngựa xem hoa. :) Các bạn có thể thấy những môn về toán rời rạc mình học rất nhiều và viết rất kỹ ở notebook này, mỗi tội đa phần vẫn là cơ bản. :) Các kiến thức về toán liên tục ở notebook này vào thời điểm hiện tại còn ít.

Quan niệm của mình về toán học

Mình thích toán vì sự logic và chặt chẽ của nó. Khi học những môn trừu tượng như lý thuyết nhóm, số phức, ... thì mình còn thích hơn vì khi đó có một nền tảng chung, trừu tượng hơn cho những gì chi tiết. Ví dụ, hình học Euclid được xây dựng dựa trên các tiên đề của Euclid từ **quan sát trực quan**. Về sau, các nhà toán học xây dựng một nền tảng cao hơn mà với một điều kiện nhất định thì Euclid đúng, nhưng với một điều kiện khác thì Euclid nhường chỗ cho người khác. Lúc này trực quan không còn đúng nữa mà phụ thuộc vào trí tưởng tượng, sáng tạo của con người. Cách tiếp cận tổng quát hóa này giúp làm đơn giản vấn đề nhưng cũng là cơ sở cho những "hạt giống" khác phát triển ở nhiều lĩnh vực khác.

Năm 2024 mình có xem một series trên Youtube của kênh [Nhận thức mới](#) về định lí bất toàn của Godel do giáo sư Phạm Việt Hưng trình bày. Minh thấy series rất thú vị và mình có nhiều quan điểm giống giáo sư, nhưng cũng có nhiều quan điểm trái ngược.

Mặc dù toán học logic và chặt chẽ, nhưng định lí bất toàn đã chứng minh được rằng toán học không hoàn hảo. Quan trọng hơn là tính đúng đắn của toán học nằm ngoài toán học và có nhiều điều, kể cả trong toán học, chúng ta không thể biết là đúng hay sai. Điều này mình đồng ý với Godel lão giáo sư. Tuy nhiên một vấn đề là nếu toán học sai thì sẽ hậu quả sẽ rất khủng khiếp. Ví dụ, trước khi khái niệm "giới hạn" ra đời thì người xưa có những lập luận đúng về mặt logic nhưng sai trên thực tế như nghịch lý nổi tiếng của Zeno. Ở thời của Zeno thì các khái niệm trừu tượng như "vô hạn", "vô cùng lớn", "vô cùng bé" chưa xuất hiện. Do đó khi làm việc với những đại lượng vô hạn đã gây ra những hiểu lầm mà về mặt trực quan chúng ta lại thấy "có vẻ đúng". Euclid cho rằng "Một phần thì nhỏ hơn toàn bộ". Tuy nhiên lý thuyết tập hợp của

Cantor đã chứng minh được một phần (của tập vô hạn) cũng bằng toàn bộ, thậm chí có nhiều kiểu vô hạn với "kích thước" lớn nhỏ khác nhau.

Trước thế kỉ 18, toán học được sử dụng làm công cụ chính để nghiên cứu các lĩnh vực khác, nhất là vật lí. Điều này phù hợp khi giải thích các hiện tượng **quan sát được** vào thời đó như mặt phẳng Euclid. Vấn đề là vật lí dựa trên nền tảng toán học, nhưng toán học lại dựa trên các quan sát hiện tượng vật lí. Đây là một vòng luẩn quẩn và chỉ cần sai sót nhỏ trong toán sẽ gây ra ảnh hưởng rất nghiêm trọng. Từ đó toán học cần phải đúng đắn từ nội tại.

Các nhà toán học thế kỉ 18, 19 như Cauchy, Hilbert, ... đã xây dựng lại toán học không dựa trên quan sát vật lí mà dựa trên các định nghĩa chặt chẽ. Trong series về định lí bất toàn, giáo sư Phạm Việt Hưng có nói các nhà toán học thời kì này tôn sùng và thần thánh hóa toán học quá mức. Điều này minh chứng đồng ý vừa không đồng ý với giáo sư. Rõ ràng việc các nhà toán học làm lúc đó là rất cần thiết để đảm bảo nội tại toán học thống nhất, không mâu thuẫn, và chặt chẽ. Việc họ tôn sùng quá mức cho thấy họ hiểu sự quan trọng của việc họ làm và rất nhiều những điều bị bác bỏ trước đó đã được đưa trở lại toán học. Từ đây mở ra rất nhiều khả năng phát triển, mở rộng tới những nơi chúng ta chưa thể thấy, sửa chữa những sai lầm của người đi trước. Bằng việc để trí tưởng tượng bay xa, kết hợp với logic chặt chẽ, đã giúp nhiều lĩnh vực đoán trước sự tồn tại của nhiều đối tượng trước cả khi phát hiện ra chúng.

Tuy nhiên quan niệm của mình về các nhà toán học thời kì đó có chút khác. Mình công nhận là những đóng góp của họ đã đưa tới sự phát triển đáng kinh ngạc trong nhiều lĩnh vực, nhưng bản thân mình thấy đó là công việc ... rất nhảm chán. Mục tiêu quan trọng nhất của họ là làm chặt chẽ và đảm bảo tính đúng đắn của toán học từ những khái niệm có thể gọi là cơ bản nhất làm nền tảng. Những khái niệm như điểm, đường, ... đã được định nghĩa từ rất lâu ở bộ sách huyền thoại Elements của Euclid [2], nhưng trong sách lại định nghĩa chung chung kiểu:

- mặt là thứ có bề dài và có bề rộng;
- đường là thứ chỉ có bề dài mà không có bề rộng;
- điểm là thứ không có bề dài lẫn bề rộng.

Ở đây, "thứ" là gì? Nhiều hình thù kì quặc đã được xây dựng, vừa có thể được xem là đường mà cũng vừa có thể được xem là mặt theo định nghĩa của Euclid, ví dụ như tấm thảm Sierpinski. Rõ ràng những nhà toán học hiện đại đã làm công việc chán ngắt là định hình lại toán học nhân loại trong 2000 năm. Cộng 1 respect cho các nhà toán học chứ mình thấy là oái rồi. :>>>

Đối với mình, toán học là bộ môn của đam mê, logic, và cả toxic. =))) Các nhà toán học không quan tâm kết quả của mình có ứng dụng thực tiễn gì. Họ quan tâm tính đúng đắn của từng mệnh đề, từng chứng minh. Họ cố gắng bảo vệ niềm tin của mình đến nỗi dám đương đầu mọi thứ, thậm chí tử thần.

Ở thời Trung Cổ nơi những tòa án dị giáo đòn áp dã man các nhà khoa học ủng hộ thuyết nhật tâm của Copernicus, các nhà khoa học (đa phần là toán và thiên văn) thậm chí còn bác bỏ những điểm chưa đúng của Copernicus - cho rằng Trái Đất quay xung quanh Mặt Trời - và bổ sung rằng bản thân Mặt Trời cũng quay quanh cái gì đó khác nữa. Cauchy tin tưởng cách xây dựng giới hạn dựa trên ngôn ngữ $\delta - \varepsilon$ tới nỗi năm lần bảy lượt chơi trò "mèo vờn chuột" với định chế khoa học cao nhất nước Pháp là Viện Hàn lâm và Đại học Bách khoa (Ecole Polytechnique) [3]. Tất nhiên Cauchy đã có cuộc đời không dễ dàng gì và nhiều lần rơi vào tình cảnh khó khăn. Cantor tin tưởng lý thuyết tập hợp của mình là vững như đá tảng, nếu mũi tên nào bắn vào thì mũi tên sẽ bật ngược lại người bắn [3]. Tuy nhiên phe đối lập với Cantor có các nhà toán học vĩ đại Poincare và Kronecker với quan điểm bảo thủ luôn cố gắng vì sự ổn định của ông. Về sau, những người ủng hộ ông như Hilbert, Dedekind, ... đã thắng thế, và lý thuyết tập hợp của Cantor đã được truyền bá rộng rãi. Đáng tiếc thay, Cantor đã ra đi mãi mãi trước đó do đột quỵ từ trầm cảm bởi sức ép từ phe Kronecker và nỗi đau mất người thân.

Kết luận

Các bạn thấy đó, bản thân toán học không hoàn hảo và thậm chí các nhà toán học nhiều lần xung đột với nhau về logic. Về mặt logic thì ai cũng đúng, nhưng éo le là người khác không chấp nhận bạn đúng và bác bỏ nó. :'(Lúc này, hệ thống định nghĩa là điều cực kì quan trọng và cần được thống nhất trên toàn thế giới, giữa các cộng đồng khoa học. Trong các công trình của mình thì hệ thống định nghĩa phải rõ ràng, thống nhất và không mâu thuẫn. Khi đó những kiến thức được xây dựng trên đó sẽ hợp lý về logic.

Toán học đã, đang và sẽ luôn là niềm đam mê của mình bất chấp khả năng có hạn. ^) Mình hy vọng notebook này sẽ lưu trữ đam mê tuổi trẻ của mình, và nếu trong khả năng có hạn, tiếp thêm sức mạnh cho những người theo đuổi toán học.

Moscow, ngày 23 tháng 02 năm 2025.

1.1.2 Cơ sở xây dựng notebook

Mục tiêu của notebook

Notebook này được xây dựng từ kiến thức của bản thân mình, những điều thú vị đối với mình trên con đường học toán và mật mã học. Lượng kiến thức và công bố khoa học ngày nay quá nhiều nên mình chỉ ghi lại những điều mình thích và mình nghĩ là quan trọng với mình.

Notebook này được mình xây dựng cho bản thân trong tương lai. Tuy vậy mình cũng hy vọng một ngày đẹp trời nào đó notebook này trở thành nơi giúp cộng đồng khoa học Việt Nam phát triển mạnh.

Công cụ phát triển notebook

Ban đầu, notebook được viết bởi LaTeX PDF. Phần cũ mình đã lưu ở [đây](#), các bạn có thể xem nếu hứng thú.

Cá nhân mình rất thích font chữ trên LaTeX PDF (CMU Serif) nên mình bắt đầu viết vào tháng 3 năm 2023. Tuy nhiên sau nhiều suy nghĩ và tham khảo ý kiến thì mình đã quyết định chuyển hết sang dạng web vào tháng 10 năm 2024.

Việc viết notebook ở dạng web có một số ưu điểm so với PDF:

- cỡ chữ phù hợp với việc đọc trên điện thoại: mặc dù cùng cỡ chữ 12pt nhưng trên điện thoại đọc PDF khó khăn hơn đọc web;
- khả năng đưa code (Python, C++) vào: việc bỏ code vào web không bị giới hạn bởi lề web và những đoạn code quá dài không cần phải suy nghĩ nên xuống dòng như thế nào như trên PDF;
- một tính năng thú vị mà mình hay dùng của package `sphinx` (Python) là các admonition. Mình sử dụng hai class là `danger` và `dropdown` cho các phần chứng minh và sau này mình dùng `dropdown` cho các phần code. Lý do cho việc này mình sẽ giải thích rõ ở phần sau;
- mình sử dụng package `sphinx-proof` để đánh số thứ tự cho định nghĩa, định lí, nhận xét, tương tự như việc khai báo `newenvironment` hay `newtheorem` trong LaTeX. Hiện tại nhược điểm của package này là chưa hỗ trợ tiếng Việt và mình cũng chưa biết đóng góp hay chỉnh sửa ra sao.

Tuy nhiên một số nhược điểm ở dạng web cũng khá khó chịu:

- các công thức toán "inline", tức là được đặt trong cặp dollar $\$ \dots \$$ không tự động xuống dòng nên dễ bị tràn ra bên phải. Thực ra chúng ta có thể kéo màn hình qua và đọc được nhưng không "mượt";
- font chữ: mình sử dụng theme `Furo` giống với trang tài liệu của SageMath (phiên bản 2025) và nhà thiết kế theme nói rằng font chữ được lựa chọn cẩn thận để phù hợp với việc đọc trên web. Mình đồng ý với họ vì nhìn các font LaTeX (CMU Serif) trên web lại không thuận mắt như trên PDF. Cơ mà nếu được thì mình vẫn thích các font LaTeX hơn.

Các hình vẽ chất lượng cao được vẽ từ TikZ và sau đó mình chuyển đổi PDF thành JPEG để đưa vào web nên chất lượng bị giảm.

UPDATE ngày 07/04/2025: ban đầu mình viết notebook bằng markdown (sử dụng Jupyter Book) nhưng vì Jupyter Book dựa trên Sphinx mà Sphinx có nhiều tính năng hay ho hơn nên mình chuyển sang viết bằng reStructuredText (rST) thay vì markdown. Dù vậy nếu đọc giả vẫn có thể đóng góp bằng markdown (như bên dưới mình có trình bày) và mình sẽ tự sửa thành rST, tất nhiên là kèm credit.

Nguyên lí xây dựng notebook

Chứng minh là bắt buộc, nhưng dễ gây chán

Mình nhận thấy nhiều quyển sách bị đánh giá thấp vì lỗi viết định lí rồi sau đó chứng minh, rồi lại tiếp tục như vậy. Đây thường là sách giáo trình được viết theo khuôn mẫu của nhà xuất bản nên điều này dễ hiểu. Tuy nhiên mình muốn ghi lại nhiều nội dung và việc ghi chứng minh sẽ khiến bài viết dài lê thê.

Chứng minh là bắt buộc nên mình vẫn viết nhưng sử dụng admonition của sphinx với class dropdown nhằm ẩn phần chứng minh đi. Các bạn có thể bấm vào chữ "Chứng minh" để xem đầy đủ. Template thông thường của chứng minh sẽ có dạng

```
.. admonition:: Chứng minh  
    :class: danger, dropdown
```

Ở đây viết chứng minh.

Kết quả sẽ như sau:

ⓘ Chứng minh

Ở đây viết chứng minh.

Class danger để màu đỏ cho đẹp. :)))

Nếu có thể ẩn phần code thì mình sẽ ẩn

Notebook này có rất nhiều demo phá mã với code, cũng như writeup CTF. Mình nhận thấy cần có code để dễ theo dõi (cả đẽ lắn lời giải). Tuy nhiên đôi khi code khá dài và việc này cũng khiến bài viết nhìn có vẻ dài. Do đó mình cũng sử dụng class dropdown để ẩn phần code đi hoặc chuyển thành đường dẫn tải xuống, và nếu các bạn muốn đọc chi tiết thì chỉ cần ấn vào để xem. Template lúc này có dạng

```
.. admonition:: main.py  
    :class: dropdown
```



```
... code-block:: python  
  
    print("Hello, World")
```

Kết quả sẽ là:

ⓘ main.py

```
print("Hello, World")
```

Tiếng Việt là nền tảng

Mình rất thích tiếng Việt và mình cũng không nghĩ là nên làm khó bản thân trong tương lai, thậm chí đọc giả, bằng việc viết ngôn ngữ khác. Tiếng Việt rất giàu đẹp, phong phú, và sẽ luôn là xương sống cho notebook này.

Một vấn đề khá đau đầu là rất nhiều từ vựng chuyên ngành hiện tại chưa có thuật ngữ tương đương trong tiếng Việt. Các thuật ngữ mình dùng đa phần là lấy từ wikipedia hoặc đọc từ các sách tiếng Việt, còn nếu không có thì mình sẽ giữ nguyên tiếng Anh. Đôi khi các thuật ngữ lại nghe khá ... chuối và các tay to nước nhà đã cải thiện dịch thuật. Ví dụ như "Dynamic programming" trong bảng dịch quyển "Introduction to Algorithm" của MIT năm 2006 được gọi là "Lập trình động", rất sát từ (mà giờ nhiều người gọi là dịch kiểu word-by-word). Lúc mình học cấp 3 (2015-2018) thì được là "Quy hoạch động" nghe vui hơn :))

Một vấn đề đau đầu tương tự là sự trùng lặp thuật ngữ tiếng Việt. Hiện tại hai từ **encode** và **encrypt** đều được dịch là **mã hóa**, nhưng về bản chất của hai hành động là khác nhau hoàn toàn. Encode sẽ chuyển đổi giữa hai bảng chữ cái, ví dụ chuyển các ký tự trên bàn phím thành các số nhị phân gọi là bảng mã ASCII, hoặc trong lý thuyết coding là $\mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ với m và n không nhất thiết giống nhau. Trong khi đó encrypt sẽ biến đổi trong cùng bảng chữ cái, ví dụ RSA là $\mathbb{Z}_n \rightarrow \mathbb{Z}_n$ bằng phép modulo, mã khôi nói chung có dạng $\mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. Ở notebook này mình sẽ dùng mã hóa để chỉ **encrypt**, còn code thì mình sẽ giữ nguyên thuật ngữ tiếng Anh (lý thuyết coding, encode, decode, coder, decoder).

Mọi khái niệm, định nghĩa mình sẽ cố gắng viết bằng cả tiếng Việt, tiếng Anh và tiếng Nga. Quan trọng nhất là các bài viết trên notebook này sẽ chỉ được viết bằng tiếng Việt.

Tổng kết

Việc xây dựng notebook này giống như tài liệu học tập cho bản thân mình. Tuy nhiên nếu may mắn mà notebook này giúp bạn nào đó học tốt hơn thì cũng là điều tốt.

Mọi đóng góp của độc giả để giúp notebook tốt hơn, thậm chí đóng góp bài viết (ở dạng markdown) càng được hoan nghênh. Mình viết trên reStructuredText nhưng có thể không thân thiện với đa số người dùng nên các bạn gửi markdown về và mình sẽ tự sửa về rST. Các bạn có thể mở issue trên repository nếu có vấn đề với nội dung mình viết. Nếu các bạn muốn đóng góp bài viết (contribute) cho repository này, các bạn có thể contribute vào repository trên github hoặc gửi file markdown qua mail cho mình (dunqlq@yandex.ru). Các bạn nhớ để lại tên ở cuối bài viết hoặc giới thiệu trong mail vì có khả năng mình sẽ biên tập lại bài viết của các bạn (về kí hiệu, bổ sung diễn giải, ...) nhưng mình giữ tên tác giả ở đầu bài viết. Như mình đã nói ở trên, các bài viết trên notebook này chỉ được viết bằng tiếng Việt nên mình xin phép chỉ nhận các bài đóng góp bằng tiếng Việt.

Cám ơn các bạn đã đọc những dòng ghi chú về những cơ sở và động lực mình xây dựng notebook này!!!

Moscow, ngày 23 tháng 02 năm 2025.

1.1.3 Những lời răn

There is no royal road to geometry.

Không có con đường hoàng gia đến hình học.

---Euclid

Học, học nữa, học mãi.

---Vladimir Ilich, Lenin (1870 - 1924)

We must know. We will know.

Chúng ta phải biết. Chúng ta sẽ biết.

---David, Hilbert (1862 - 1943)

Mình tin rằng khả năng của con người là có hạn. Tuy nhiên giới hạn của mỗi người trong mỗi lĩnh vực mỗi khác. Có người có năng lực về mảng này nhưng lại kém mảng kia, nhưng cũng có người giỏi mảng kia và kém mảng này. Việc chắc chắn một điều không thể thực hiện khi chỉ mới nhìn là không hợp lý. Bằng chứng là loài người có khả năng kế thừa tri thức rất mạnh, và những điều được cho là bế tắc của hiện tại, ai biết được 500 năm sau có còn bế tắc nữa không.

Trong series về định lí bất toàn của Godel trên Youtube, giáo sư Phạm Việt Hưng nói rằng có những điều mà chúng ta không bao giờ biết được, không bao giờ hiểu được. Minh không đồng ý với giáo sư ở điểm này vì định lí cuối cùng của Fermat từng bị cho là không thể giải trong hơn 300 nhưng vẫn khuất phục trước trí tuệ của loài người. Hiện tại có thể là chúng ta không biết, nhưng không có gì đảm bảo chúng ta mãi mãi không biết.

Không thể là một nhà toán học mà không có tâm hồn thi sĩ.

---Sofia, Kovalevskaya (1850 - 1891)

Đường đi ngàn dặm, khởi đầu bởi một bước chân.

---Lão Tử

Mình rất thích câu nói này. Nếu mình làm một việc gì đó, có thể mình sẽ thành công, cũng có thể mình sẽ thất bại. Tuy nhiên chắc chắn mình sẽ học hỏi được gì đó trong quá trình làm việc. Lúc trước mình hay sợ thất bại, sợ uổng công mà không được gì, nhưng về sau mình nhận ra nếu không làm thì mình mãi sẽ không làm được gì cả.

Lịch sử loài người đi qua nhiều lần thử sai. Đôi khi cái giá phải trả cho tri thức, cho chân lý rất đắt. Không ít nhà khoa học đã bỏ mạng khi thực hiện các thí nghiệm hoặc chế tạo các thiết bị giúp đời sống con người tốt hơn. Ở thời Trung Cổ, các nhà khoa học thậm chí còn dám chống lại Tòa án Dị giáo vì sự đúng đắn của khoa học, và kết cục luôn rất thảm khốc. Những bài toán từng được mệnh danh là không thể giải, ví dụ như định lí cuối của Fermat, đã được giải sau 300 năm cố gắng của loài người. Trên con đường giải quyết định lí cuối cùng của Fermat, rất nhiều lý thuyết mới đã ra đời, thúc đẩy toán học phát triển mạnh mẽ.

Trong phim *Mr. Robot* (phần 1), Mr. Robot đã nói với Elliot rằng:

Khi xem xét đến tận cùng của mọi vấn đề thì đều quy về 0 hoặc 1. Nếu cậu làm, cậu là 1. Nếu cậu không làm, cậu là 0.

Câu nói của Mr. Robot cũng giống với niềm tin của nhà toán học David Hilbert. Ông tin rằng trên đời chỉ có hai loại người: những người làm việc và tạo ra kết quả, và những người không làm gì. Minh xin phép bổ sung cho ý này, kết quả không nhất thiết phải theo hướng tốt. Một thí nghiệm có thể thành công, có thể thất bại. Nếu thành công tức là lý thuyết có triển triển. Nếu thất bại, chúng ta xem xét nguyên nhân, rút kinh nghiệm, thay đổi cách tiếp cận, ... Trong cả hai trường hợp, chúng ta đều đang tiến lên, không ì ạch một chỗ.

Không có việc gì khó.

Chỉ sợ lòng không bền.

Đào núi và lấp biển.

Quyết chí ắt làm nên.

---Hồ Chí Minh (1890 - 1969)

Luôn yêu đẻ sống. Luôn sống đẻ học toán. Luôn học toán đẻ yêu.

---Diễn đàn toán học Việt Nam

Những thứ đích thực có giá trị không sinh ra từ tham vọng hoặc ý thức trách nhiệm đơn thuần, mà đến từ tình yêu và sự hiến dâng cho nhân loại và những điều khách quan.

---Albert Einstein

Tôi tư duy nên tôi tồn tại.

---René Descartes

1.1.4 Kí hiệu và công thức toán trong mật mã

Khi tham khảo các tài liệu về mật mã thì mình gặp chút khó khăn vì mỗi tác giả kí hiệu mỗi kiểu cho cùng khái niệm toán học. Hơn nữa, trong cộng đồng toxic CH mình cũng thấy được nhiều người kí hiệu mà không hiểu rõ nó là gì. Trong bài viết này mình sẽ nói về các kí hiệu toán học mà mình sẽ dùng trong các bài viết của mình, cũng như quan điểm về chúng.

Có nên sử dụng kí hiệu mọi lúc, mọi nơi?

Trong toán học, kí hiệu được sử dụng nhằm mô tả những định nghĩa, định lí. Thông qua kí hiệu, các nhà toán học của nhiều quốc gia khác nhau có thể "thấu hiểu" nhau bất chấp rào cản ngôn ngữ. Tuy nhiên trong các bài viết của mình thì mình sẽ không sử dụng kí hiệu mọi lúc, mọi nơi.

Ví dụ, định nghĩa giới hạn hàm số theo kiểu $\delta - \varepsilon$ được ghi như sau

$$\forall \varepsilon > 0, \exists \delta > 0 : \forall |x - x_0| < \delta \Rightarrow |f(x) - L| < \varepsilon.$$

Đây là định nghĩa giới hạn hữu hạn của hàm số, nói rằng hàm số $f(x)$ tiến tới L khi x tiến tới x_0 . Mình thấy việc kí hiệu đôi khi khiến mình bị rối khi theo dõi các phần (có lẽ do mình không giỏi toán). =(((

Do đó định nghĩa giới hạn hàm số ở trên được viết lại theo tiếng Việt như sau:

- với mọi $\varepsilon > 0$
- tồn tại $\delta > 0$ sao cho:
 - với mọi x mà $|x - x_0| < \delta$
 - ta sẽ có $|f(x) - L| < \varepsilon$.

Khi này, việc chứng minh giới hạn hàm số theo định nghĩa sẽ "thông" hơn. Mình sẽ theo từng câu chữ ở trên:

- lấy $\varepsilon > 0$ bất kì (tương ứng lượng từ *với mọi*)
- tìm $\delta > 0$ (chứng minh tồn tại, thường sẽ liên hệ với ε ở trên) thỏa mãn:
 - nếu với mọi x thỏa mãn $|x - x_0| < \delta$
 - mình sẽ suy ra được $|f(x) - L| < \varepsilon$.

Kết luận: việc viết ra đầy đủ giúp mình biết được các bước cần thực hiện theo định nghĩa, định lí nào đó.

\mathbb{Z}_n hay $\mathbb{Z}/n\mathbb{Z}$?

Thông thường, việc kí hiệu \mathbb{Z}_n được hiểu là tập hợp các số dư có thể có khi chia một số nguyên bất kì cho n , nói cách khác

$$\mathbb{Z}_n = \{0, 1, 2, \dots, n - 1\}.$$

Đối với $\mathbb{Z}/n\mathbb{Z}$, chúng ta có thể dùng hai cách lí giải: bằng lý thuyết nhóm và bằng quan hệ tương đương (quan hệ hai ngôi).

Nếu sử dụng lý thuyết nhóm, nếu ta nhân mỗi phần tử trong \mathbb{Z} với n thì

$$n\mathbb{Z} = \{\dots, -2n, -n, 0, n, 2n, \dots\}.$$

Ta lần lượt cộng i cho các phần tử của $n\mathbb{Z}$ với $i = 0, 1, \dots, n - 1$. Khi đó ta sẽ có

$$\begin{aligned} 0 + n\mathbb{Z} &= \{\dots, -2n, -n, \mathbf{0}, n, 2n, \dots\}, \\ 1 + n\mathbb{Z} &= \{\dots, -2n + 1, -n + 1, \mathbf{1}, n + 1, 2n + 1 \dots\}, \\ &\vdots \\ (n - 1) + n\mathbb{Z} &= \{\dots, -n + 1, -1, \mathbf{n - 1}, 2n - 1, 3n - 1, \dots\}. \end{aligned}$$

Các tập $0 + n\mathbb{Z}$, $1 + n\mathbb{Z}$, ..., $(n - 1) + n\mathbb{Z}$ rời nhau nên chúng ta có đúng n coset. Hơn nữa phép cộng số nguyên có tính giao hoán nên $i + n\mathbb{Z} = n\mathbb{Z} + i$. Như vậy tập $n\mathbb{Z}$ là **nhóm con chuẩn tắc** (hay **normal subgroup**) của \mathbb{Z} . Khi đó $\mathbb{Z}/n\mathbb{Z}$ được gọi là **nhóm thương** (hay **quotient group**) và kí hiệu

$$\mathbb{Z}/n\mathbb{Z} = \{0 + n\mathbb{Z}, 1 + n\mathbb{Z}, \dots, (n - 1) + n\mathbb{Z}\}.$$

Đối với cách giải thích sử dụng quan hệ tương đương, chúng ta kí hiệu

$$\bar{x} = \{y \in \mathbb{Z} : x \equiv y \pmod{n}\}.$$

Nói cách khác, x và y có quan hệ nếu x và y có cùng số dư khi chia cho n .

Dễ thấy

$$\begin{aligned} \bar{0} &= \{\dots, -2n, -n, \mathbf{0}, n, 2n, \dots\}, \\ \bar{1} &= \{\dots, -2n + 1, -n + 1, \mathbf{1}, n + 1, 2n + 1 \dots\}, \\ &\vdots \\ \overline{n - 1} &= \{\dots, -n + 1, -1, \mathbf{n - 1}, 2n - 1, 3n - 1, \dots\}. \end{aligned}$$

Đây là ví dụ hoặc bài tập phổ biến trong các bài giảng về quan hệ tương đương. Ở đây, quan hệ tương đương chia (phân hoạch) tập \mathbb{Z} thành n tập con không giao nhau, gọi là các **lớp tương đương**. Chúng ta lấy số nguyên không âm nhỏ nhất của mỗi lớp làm đại diện cho lớp đó, tức là $0, 1, \dots, n - 1$ như trên.

Khi đó, **tập thương** là tập hợp các lớp tương đương

$$\mathbb{Z}/n\mathbb{Z} = \{\bar{0}, \bar{1}, \dots, \overline{n - 1}\}.$$

Như vậy, dù giải thích theo lý thuyết nhóm hay quan hệ tương đương đều chỉ ra rằng $n\mathbb{Z}$ chia tập \mathbb{Z} thành n tập con không giao nhau, và chúng ta lấy số nguyên không âm nhỏ nhất của mỗi tập làm đại diện cho tập đó. Lúc này, các phép cộng, trừ và nhân (không có chia) trên tập $\mathbb{Z}/n\mathbb{Z}$ sẽ cho các phần tử vẫn thuộc $\mathbb{Z}/n\mathbb{Z}$.

Tuy nhiên phép tính trên \mathbb{Z}_n phải chỉ định phép modulo n , chẳng hạn là $a + b \pmod{n}$ với $a, b \in \mathbb{Z}_n$.

Lý do hai tập này có thể dùng như nhau là tính đẳng cấu, kí hiệu là $\mathbb{Z}_n \cong \mathbb{Z}/n\mathbb{Z}$. Trong nhiều tài liệu, tập \mathbb{Z}_n được định nghĩa là

$$\mathbb{Z}_n = \{\bar{0}, \bar{1}, \dots, \overline{n - 1}\}$$

chỉ vành các số dư (residue ring, кольцо вычетов). Ý nghĩa vẫn giống $\mathbb{Z}/n\mathbb{Z}$, ta xét tất cả phần tử của \mathbb{Z} dưới n lớp rời nhau. Do đó mình nghĩ rằng cần hiểu rõ ý nghĩa của $\mathbb{Z}/n\mathbb{Z}$ trước khi phán \mathbb{Z}_n ở bất cứ đâu.

\mathbb{Z}_p hay \mathbb{F}_p ?

Khi p là số nguyên tố thì chúng ta có thể thực hiện phép chia khác 0 (nhân nghịch đảo) trong modulo p . Khi đó tập \mathbb{Z}_p trở thành trường và chúng ta có thể dùng \mathbb{F}_p để thể hiện rõ ý này ($F = \text{Field}$). Cách kí hiệu \mathbb{Z}_p không sai nhưng mình nghĩ sẽ khó theo dõi xem đâu là trường, đâu không phải là trường (mới chỉ là vành).

$\text{GF}(2^8)$ hay $\text{GF}(256)$?

Rõ ràng $2^8 = 256$, và hai cách kí hiệu là một. Tuy nhiên mình chọn viết cách đầu.

Đầu tiên, việc kí hiệu 2^8 sẽ dễ liên hệ tới phần tử của trường, tức là các đa thức bậc 8 và có dạng

$$a_0 + a_1x + \cdots + a_7x^7$$

với $a_i \in \text{GF}(2)$.

Nếu viết $\text{GF}(256)$, chúng ta phải nhớ xem 256 phân tích thành thừa số nguyên tố ra sao. Điều này đơn giản nếu chúng ta làm việc với các số quen thuộc như $2^8 = 256$. Tuy nhiên nếu chúng ta nghiên cứu trên số nguyên tố khác, chẳng hạn

$$\text{GF}(6561), \text{GF}(625), \dots$$

thì không phải ai cũng nhớ. Chúng ta phải mất công phân tích số 6561 thành 3^8 , hay 625 thành 5^4 , rồi mới làm tiếp.

Một ví dụ khác là

$$\text{GF}(340282366920938463463374607431768211456),$$

thì cũng là $\text{GF}(2^{128})$ thôi, được dùng khi tính message authentication code (MAC) của thuật toán mã hóa đối xứng. Ở đây chắc chắn mình sẽ chọn viết $\text{GF}(2^{128})$ thay vì con số dài ngoằng kia.

Như vậy, việc viết $\text{GF}(256)$ không sai, nhưng mình nghĩ viết $\text{GF}(2^8)$ có nhiều ưu điểm hơn và sẽ giúp tạo thói quen tốt.

\mathbb{F}_{2^8} hay \mathbb{F}_2^8 ?

Hai tập hợp này có cùng số phần tử là $2^8 = 256$. Tuy nhiên ý nghĩa của chúng khác nhau hoàn toàn.

Đầu tiên, \mathbb{F}_{2^8} chỉ trường các đa thức

$$a_0 + a_1x + \cdots + a_7x^7,$$

với $a_i \in \mathbb{F}_2$. Các phép tính cộng, trừ, nhân, chia hai đa thức được thực hiện trong modulo $m(x)$ - là một đa thức tối giản bậc 8 với hệ số trong \mathbb{F}_2 .

Trong khi đó, \mathbb{F}_2^8 chỉ các vector

$$\mathbf{a} = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7) \in \mathbb{F}_2^8$$

với $a_i \in \mathbb{F}_2$. Ta xem \mathbb{F}_2^8 là một không gian vector. Khi đó chúng ta chỉ có hai phép tính trên tập \mathbb{F}_2^8 là cộng hai vector và nhân vector với một phần tử thuộc \mathbb{F}_2 . Nếu các bạn mở rộng lên không gian Euclid hay gì đó thì vẫn không giống \mathbb{F}_{2^8} .

GF(p^n) hay \mathbb{F}_{p^n} ?

Hai cách kí hiệu đều có ý nghĩa như nhau.

Khi $n = 1$ thì mình thấy dùng GF(p) hay \mathbb{F}_p đều được.

Khi $n \geq 2$ thì mỗi cách kí hiệu đều có ưu điểm và nhược điểm riêng. Cả hai cách đều chỉ trường số với các phần tử là đa thức

$$a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1},$$

với $a_i \in \mathbb{F}_p$, hay cũng có thể viết $a_i \in \text{GF}(p)$.

Việc viết \mathbb{F}_{p^n} có nhược điểm là làm đại lượng p^n hơi nhỏ, khó nhìn nên sử dụng GF(p^n) sẽ tốt hơn. Tuy nhiên một ưu điểm của \mathbb{F}_{p^n} là có thể chỉ tập các vector mà mỗi vị trí là một phần tử trong \mathbb{F}_{p^n} . Ví dụ

$$\mathbf{f} = (f_1(x), f_2(x), \dots, f_m(x)) \in \mathbb{F}_{p^n}^m$$

với $f_i(x)$ là các phần tử thuộc \mathbb{F}_{p^n} . Nếu sử dụng GF(p^n) thì phải viết $\text{GF}(p^n)^m$ khá rối. Khi chúng ta xét tới ma trận có phần tử trong \mathbb{F}_{p^n} thì còn rối hơn nữa. Thay vào đó ta có thể viết

$$\mathbf{A} = \begin{pmatrix} a_{1,1}(x) & a_{1,2}(x) & \cdots & a_{1,d}(x) \\ a_{2,1}(x) & a_{2,2}(x) & \cdots & a_{2,d}(x) \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1}(x) & a_{m,2}(x) & \cdots & a_{m,d}(x) \end{pmatrix} \in \mathbb{F}_{p^n}^{m \times d}$$

với $a_{i,j}(x)$ là phần tử thuộc \mathbb{F}_{p^n} .

Từ các lý do trên có thể thấy \mathbb{F}_{p^n} đa dụng hơn nên mình sẽ dùng cách kí hiệu này.

2.1 Đại số

2.1.1 Đại cương về tập hợp

Tập hợp là khái niệm nền tảng, có mặt trong hầu khắp các ngành của toán học. Mình có dịp đọc quyển *Toán học qua các câu chuyện về tập hợp* của Tủ sách Sputnik [4], dịch từ quyển *Рассказы о множествах* của Виленкин Н.Я. [5] và thấy những câu chuyện rất thú vị. Nếu hứng thú các bạn có thể tìm đọc.

Tập hợp

Mở đầu về tập hợp

Một **tập hợp** (set) bao gồm các phần tử khác nhau. Tập hợp là khái niệm cơ sở cho nhiều vấn đề của toán học. Tuy nhiên chúng ta lại không có một định nghĩa chặt chẽ về tập hợp mà chỉ có thể biểu diễn nó. Để biểu diễn tập hợp ta có hai cách.

- Liệt kê. Ví dụ $A = \{1, 2, 3, 4\}$, $B = \{a, b, c\}$.
- Sử dụng tính chất đặc trưng. Ví dụ $A = \{a \in \mathbb{N}^* : a < 5\}$.

Ở đây hai cách biểu diễn tập hợp A là giống nhau.

❶ Definition 1.1 (Tập hợp rỗng)

Tập hợp rỗng không chứa phần tử nào, kí hiệu là \emptyset .

❶ Definition 1.2 (Tập hợp con)

Xét tập hợp A . Tập hợp B được gọi là **tập hợp con** của tập A nếu mọi phần tử của B đều nằm trong A . Nói cách khác với mọi $b \in B$ thì $b \in A$. Ta kí hiệu $B \subset A$.

❶ Remark 1.1

Tập hợp rỗng là con của mọi tập hợp.

Để thấy rằng mọi tập hợp là tập hợp con của chính nó. Do đó tập con này được gọi là tập con tầm thường (trivial subset). Để kí hiệu một tập con có thể bằng tập chứa nó ta viết $B \subseteq A$. Trong trường hợp B là tập con của A nhưng không bằng A ta có thể viết $B \subsetneq A$.

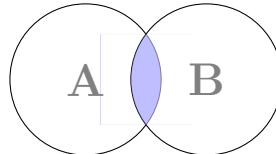
Toán tử trên tập hợp

Chúng ta xem xét ba toán tử cơ bản trên tập hợp là **giao**, **hợp** và **hiệu** của hai tập hợp. Để biểu diễn các toán tử này ta có thể dùng biểu đồ Venn.

❶ Definition 1.3 (Giao của hai tập hợp)

Giao của hai tập hợp A và B là tập hợp các phần tử thuộc cả A và B .

$$A \cap B = \{x : x \in A \text{ và } x \in B\}.$$



Hình 2.1: Phép giao hai tập hợp

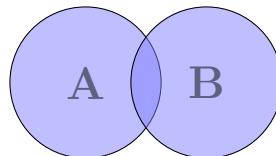
Hình 2.1 là biểu đồ Venn tương ứng của phép giao hai tập hợp. Khi giao của hai tập hợp A và B là rỗng thì ta nói hai tập rời nhau. Kí hiệu $A \cap B = \emptyset$.

❷ Definition 1.4 (Hợp của hai tập hợp)

Hợp của hai tập hợp A và B là tập hợp các phần tử thuộc A hoặc B .

$$A \cup B = \{x : x \in A \text{ hoặc } x \in B\}.$$

Hình 2.2 là biểu đồ Venn tương ứng của phép hợp hai tập hợp.



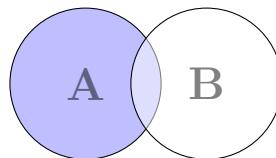
Hình 2.2: Phép hợp hai tập hợp

❸ Definition 1.5 (Hiệu của hai tập hợp)

Hiệu (hay phần bù) của tập hợp A đối với tập hợp B là tập hợp các phần tử thuộc A nhưng không thuộc B .

$$A \setminus B = \{x : x \in A \text{ và } x \notin B\}.$$

Hình 2.3 là biểu đồ Venn tương ứng của hiệu hai tập hợp.



Hình 2.3: Phép hiệu hai tập hợp

Lực lượng của tập hợp

Để chỉ số lượng phần tử của một tập hợp ta dùng khái niệm lực lượng của tập hợp.

Kí hiệu lực lượng của tập hợp A là $|A|$ hoặc $\#A$.

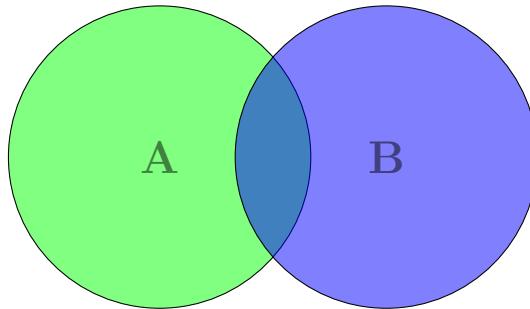
Khi một tập hợp có vô số phần tử, ta gọi đó là tập vô hạn. Ngược lại ta gọi là tập hữu hạn.

Example 1.1

Các tập hợp số thông dụng \mathbb{N} , \mathbb{Z} , \mathbb{Q} , \mathbb{R} là các tập vô hạn.

Tập hợp $A = \{1, 2, 3, 4, 5\}$ là tập hữu hạn có 5 phần tử. Kí hiệu $|A| = 5$.

Từ biểu đồ Venn chúng ta cũng có thể tìm được công thức tính lực lượng của tập $A \cup B$.



Hình 2.4: Nguyên lý bù trừ cho hai tập hợp

Dựa vào hình ta có thể suy ra công thức sau:

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

Ánh xạ

[TODO] Viết lại ánh xạ dựa trên một giáo trình chuẩn.

Ánh xạ

Cho hai tập hợp X và Y .

Nói đơn giản, ánh xạ f biến một phần tử $x \in X$ thành một và chỉ một phần tử $y \in Y$.

❶ Definition (Ánh xạ)

Một ánh xạ f từ tập X đến tập Y là một quy tắc đặt tương ứng mỗi phần tử x của X với một (và chỉ một) phần tử của Y . Phần tử này được gọi là **ánh** của x qua ánh xạ f và được kí hiệu là $f(x)$.

Tập hợp X được gọi là **tập xác định** của f . Tập hợp Y được gọi là **tập giá trị** của f .

Ánh xạ f từ X đến Y được kí hiệu là $f : X \rightarrow Y$ hoặc $f(x) = y$.

Cho $a \in X$ và $y \in Y$. Nếu $f(a) = y$ thì ta nói y là **ánh** của a và a là **nghịch ánh** của y qua ánh xạ f .

❷ Chú ý

1. Mỗi phần tử a của X chỉ có một ánh duy nhất (là phần tử $f(a)$).
2. Mỗi phần tử y của Y có thể có nhiều nghịch ánh hoặc không có nghịch ánh nào.

Tập

$$f(X) = \{y \in Y : \exists x \in X, y = f(x)\}$$

được gọi là **tập ảnh** của f .

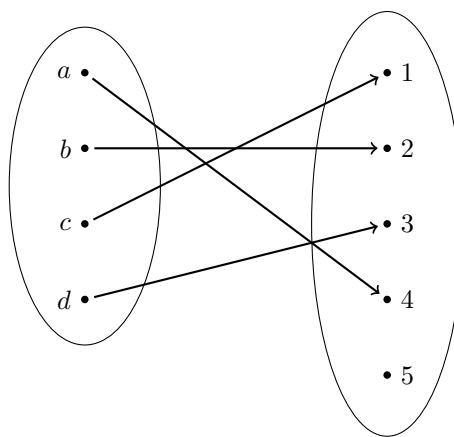
Như vậy, tập ảnh $f(X)$ là tập tất cả phần tử của Y có nghịch ánh.

Ánh xạ có ba loại:

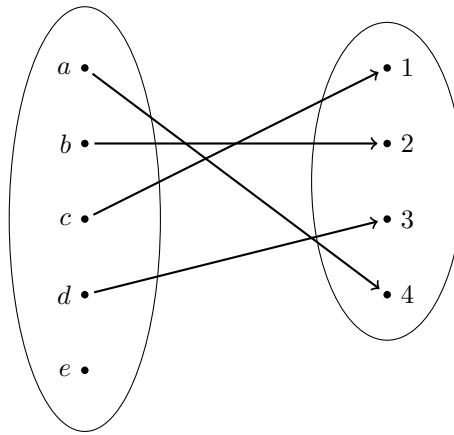
1. **Đơn ánh** (hay **Injection**): Hai phần tử khác nhau của tập nguồn cho hai ảnh khác nhau, tức là với mọi $x_1, x_2 \in X$ mà $x_1 \neq x_2$, thì $f(x_1) \neq f(x_2)$.
2. **Toàn ánh** (hay **Surjection**): Mọi phần tử $y \in Y$ đều có ít nhất một phần tử $x \in X$ mà $f(x) = y$. Nói cách khác với mỗi phần tử trong Y ta đều tìm được phần tử thuộc X biến thành nó.
3. **Song ánh** (hay **Bijection**): Nếu ánh xạ đó vừa là đơn ánh, vừa là toàn ánh.

Dựa vào định nghĩa và hình vẽ, ta có thể rút ra kết luận như sau

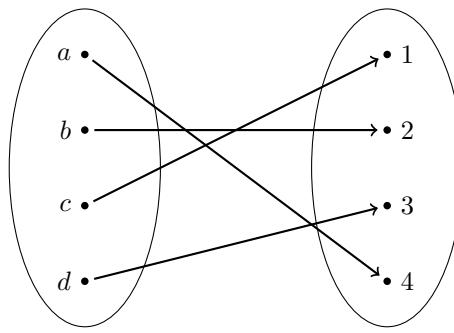
1. Đối với đơn ánh, do mọi phần tử của X đều có ảnh ở Y , tuy nhiên có thể có phần tử ở Y không do phần tử nào của X biến thành (trong hình là 5). Do đó $|X| \leq |Y|$.
2. Đối với toàn ánh, mọi phần tử của Y đều có nguồn gốc xuất xứ, tuy nhiên có thể có phần tử của X không biến thành y nào của Y (trong hình là e). Do đó $|X| \geq |Y|$.
3. Đối với song ánh, do là kết hợp giữa đơn ánh và toàn ánh, khi đó dấu đẳng thức xảy ra, $|X| = |Y|$.



Hình 2.5: Đơn ánh



Hình 2.6: Toàn ánh



Hình 2.7: Song ánh

Cho song ánh $f : X \rightarrow Y$. Khi đó với mỗi $y \in Y$ tồn tại duy nhất một phần tử $x \in X$ mà $f(x) = y$.

Phần tử duy nhất $x \in X$ này được gọi là ảnh của phần tử $y \in Y$ qua **ánh xạ ngược** của f .

i **Definition (Ánh xạ ngược của song ánh)**

Ánh xạ ngược của $f : X \rightarrow Y$, kí hiệu là f^{-1} là ánh xạ từ Y tới X biến phần tử $y \in Y$ thành phần tử $x \in X$ duy nhất, như vậy

$$f^{-1}(y) = x \iff f(x) = y.$$

Như vậy, nếu f không phải song ánh thì chúng ta không thể xác định ánh xạ ngược.

Example

Xét hàm số $f : \mathbb{R} \rightarrow \mathbb{R}$, $x \rightarrow y = f(x) = x^3$.

Lúc này, f là song ánh và mình có thể biểu diễn x theo y là $x = f^{-1}(y) = \sqrt[3]{y}$.

Definition (Ánh xạ hợp)

Xét hai ánh xạ $f : X \rightarrow Y$, $f(x) = y$ và $g : Y \rightarrow Z$, $z = g(y)$. Ánh xạ hợp của g và f được kí hiệu là

$$g \circ f : X \rightarrow Z, \quad z = g(f(x)) = g(f(x)).$$

Definition (Tích Descartes)

Tích Descartes của hai tập hợp $A = \{a_1, a_2, \dots, a_n\}$ và $B = \{b_1, b_2, \dots, b_m\}$ là tập hợp

$$A \times B = \{(a_i, b_j) : a_i \in A, b_j \in B\}.$$

Example

Với $A = \{1, 2, 3\}$ và $B = \{4, 5\}$ thì tích Descartes là

$$S = A \times B = \{(1, 4), (1, 5), (2, 4), (2, 5), (3, 4), (3, 5)\}.$$

Với nhiều tập hợp ta định nghĩa tích Descartes tương tự.

Example

Xét ba tập nguồn X, Y, Z , và tập đích là T , ánh xạ $\phi : X \times Y \times Z \rightarrow T$, với $\phi(x, y, z) \rightarrow t$ là ánh xạ ba biến, tập nguồn của ánh xạ khi này là tích Descartes $X \times Y \times Z$.

Hàm số

Hàm số

Khi hai tập nguồn và đích của ánh xạ là hai tập hợp số, ta có hàm số.

❶ Example

Hàm số $f : \mathbb{R} \rightarrow \mathbb{R}$ với $y = f(x) = x^3 + x + 1$. Ở đây $f : X \rightarrow Y$ với $X \equiv \mathbb{R}$ và $Y \equiv \mathbb{R}$.

Lưu ý rằng tập nguồn và đích không nhất thiết là tập hợp số cơ bản (\mathbb{Q}, \mathbb{R}) mà cũng có thể là tích Descartes của chúng.

❶ Example

Hàm số $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ với $z = f(x, y) = x + y + xy$. Ở đây $f : X \times Y \rightarrow Z$ với $X \equiv \mathbb{R}$, $Y \equiv \mathbb{R}$ và $Z \equiv \mathbb{R}$.

❶ Example

Hàm số $f : \mathbb{R} \rightarrow \mathbb{R}$ cho bởi $y = f(x) = x^3$ là song ánh.

❶ Chứng minh

Ta thấy nếu $f(x_1) = f(x_2)$, tương đương $x_1^3 = x_2^3$ nên $x_1 = x_2$. Do đó f là đơn ánh.

Với mọi $y = x^3 \in \mathbb{R}$, do căn bậc ba luôn tồn tại nên ta có $x = \sqrt[3]{y}$, nghĩa là luôn tồn tại x để $f(x) = y$ với mọi $y \in \mathbb{R}$. Do đó f là toàn ánh.

Kết luận f là song ánh.

Đồng biến và nghịch biến

❶ Definition (Hàm số đồng biến)

Xét hàm số $f(x)$ xác định trên khoảng $(a; b) \subset \mathbb{R}$. Ta nói $f(x)$ **đồng biến (tăng)** trên $(a; b)$ nếu với mọi $x_1, x_2 \in (a; b)$ mà $x_1 < x_2$ ta có $f(x_1) < f(x_2)$.

Tương tự $f(x)$ **nghịch biến (giảm)** trên $(a; b)$ nếu với mọi $x_1, x_2 \in (a; b)$ mà $x_1 < x_2$ ta có $f(x_1) > f(x_2)$.

Lưu ý ở các so sánh trên dấu bằng có thể xảy ra. Khi đó hàm số được gọi là tăng **không nghiêm ngặt** (hoặc giảm **không nghiêm ngặt**).

Nếu hàm số đồng biến (hoặc nghịch biến) trên khoảng xác định nào đó thì ta nói hàm số đơn điệu trên khoảng đó.

Đồ thị của hàm số khi đồng biến sẽ đi lên (theo chiều từ trái sang phải), và đi xuống nếu nghịch biến.

❶ Example

Khảo sát sự biến thiên của hàm số $f(x) = x^2 + 3$.

Để khảo sát sự biến thiên, một cách làm đơn giản theo định nghĩa là ta xét $x_1 < x_2$ và so sánh $f(x_1)$ với $f(x_2)$.

Ta có

$$f(x_1) - f(x_2) = x_1^2 + 3 - x_2^2 - 3 = (x_1 - x_2)(x_1 + x_2).$$

Do $x_1 < x_2$, nên với $x_1, x_2 > 0$ thì $x_1 + x_2 > 0$ và $x_1 - x_2 < 0$. Ta suy ra $f(x_1) - f(x_2) < 0$ và từ đó $f(x_1) < f(x_2)$. Như vậy $f(x)$ đồng biến trên $(0; +\infty)$.

Tương tự, khi $x_1, x_2 < 0$ thì $x_1 + x_2 < 0$. Khi đó $f(x_1) > f(x_2)$ nên $f(x)$ nghịch biến trên $(-\infty; 0)$.

Để thể hiện sự biến thiên của hàm số ta sử dụng bảng biến thiên.

Đối với hàm số $y = x^2 + 3$ ở trên bảng biến thiên có dạng:

x	$-\infty$	0	$+\infty$
$f(x) = x^2 + 3$	$+\infty$		$+\infty$
		3	

Hình 2.8: Bảng biến thiên hàm số $y = x^2 + 3$

Ta đã chứng minh được hàm số nghịch biến trên $(-\infty; 0)$ và đồng biến trên $(0; +\infty)$, giá trị $f(0) = 3$ nên bảng biến thiên thể hiện sự tăng giảm trên các khoảng. Dựa vào bảng biến thiên ta có thể hình dung ra dạng của đồ thị hàm số.

Đồ thị hàm số

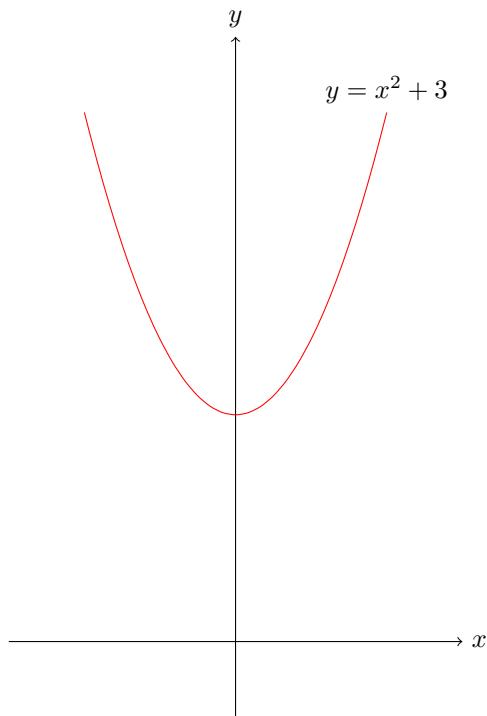
Để biểu diễn sự phụ thuộc của biến y theo biến x , hay nói cách khác là biểu diễn hàm số $y = f(x)$, ta có thể dùng đồ thị.

Đồ thị được vẽ trên hệ tọa độ Descartes Oxy . Bảng biến thiên cho ta thấy tính đơn điệu trên các khoảng xác định, và đồ thị sẽ cho ta thấy rõ hơn độ "cong" của những đường cong.

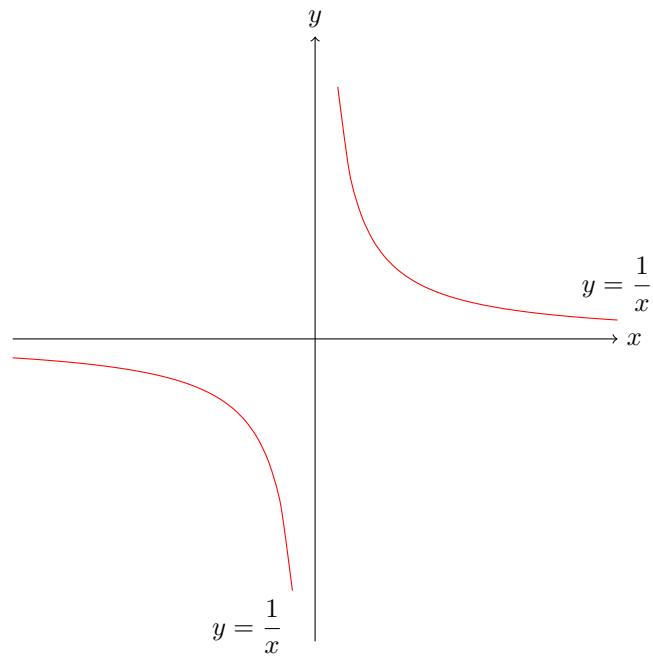
Example

Với hàm số $y = x^2 + 3$ ở trên. Đồ thị hàm số có dạng như hình 2.9.

Với hàm số $y = \frac{1}{x}$. Ta thấy rằng hàm số không xác định tại $x = 0$. Khảo sát sự biến thiên như bên trên ta thấy hàm số nghịch biến ở hai khoảng xác định là $(-\infty; 0)$ và $(0; +\infty)$. Đồ thị hàm số có dạng như hình 2.10.



Hình 2.9: Đồ thị hàm số $y = x^2 + 3$



Hình 2.10: Đồ thị hàm số $y = \frac{1}{x}$

Từ đồ thị của hai hàm số trên ta thấy rằng mặc dù cùng là nghịch biến trên $(-\infty; 0)$ nhưng nghịch biến của $y = x^2 + 3$ nhìn "nhẹ nhàng" hơn. Trong khi đồ thị $y = \frac{1}{x}$ thì ban đầu "nhẹ nhàng", sau thì như "roi tự

do".

Một số loại hàm số

Một số hàm số có tính chất đặc biệt giúp chúng ta tiết kiệm công sức trong chứng minh, tính toán.

Hàm chẵn và hàm lẻ

Xét hàm số $y = f(x)$ xác định trên miền D có tính đối xứng, nghĩa là với mỗi phần tử dương $x \in D$ thì có phần tử đối $-x \in D$ hoặc ngược lại. Khi đó

❶ Definition (Hàm số chẵn)

Hàm số $y = f(x)$ được gọi là **hàm số chẵn** nếu với mọi $x \in D$ ta có $f(-x) = f(x)$.

Ví dụ như hàm số $y = x^2 + 3$ ở trên là một hàm chẵn vì với mọi $x \in \mathbb{R}$ ta có

$$f(x) = x^2 + 3 = (-x)^2 + 3 = f(-x).$$

Dễ thấy rằng đồ thị của hàm chẵn đối xứng qua trục tung. Dựa vào tính chất này, trong lúc khảo sát hoặc tính toán đôi khi ta chỉ cần quan tâm một bên trục tung, bên kia tương tự.

❶ Definition (Hàm số lẻ)

Hàm số $y = f(x)$ được gọi là **hàm số lẻ** nếu với mọi $x \in D$ ta có $f(-x) = -f(x)$.

Ví dụ như hàm số $y = \frac{1}{x}$ ở trên là một hàm lẻ vì với mọi $x \in (-\infty; 0) \cup (0; +\infty)$ ta có

$$f(-x) = \frac{1}{-x} = -\frac{1}{x} = -f(x).$$

Dễ thấy rằng hàm lẻ đối xứng qua gốc tọa độ $O(0, 0)$.

Hàm cộng tính

Xét hàm số $y = f(x)$ xác định trên miền D .

❶ Definition (Hàm cộng tính)

Hàm số $y = f(x)$ được gọi là **cộng tính** nếu với mọi $x, y \in D$ mà $x + y \in D$, ta có $f(x+y) = f(x) + f(y)$.

❶ Example

Hàm số $y = 2x$ trên \mathbb{R} là hàm cộng tính vì với mọi $x, y \in \mathbb{R}$, ta có

$$f(x+y) = 2(x+y) = 2x + 2y = f(x) + f(y).$$

Hàm nhân tính

Tương tự hàm cộng tính, ta định nghĩa hàm nhân tính.

❶ Definition (Hàm nhân tính)

Hàm số $y = f(x)$ được gọi là **nhân tính** nếu với mọi $x, y \in D$ ta có $f(xy) = f(x) \cdot f(y)$.

Hàm nhân tính quan trọng được sử dụng trong số học là hàm φ Euler về số lượng các số nguyên tố cùng nhau với số nguyên dương n . Nếu một hàm số học là nhân tính thì chúng ta chỉ cần quan tâm giá trị của hàm số đó tại các số nguyên tố là đủ.

Hàm tuần hoàn

Xét hàm số $y = f(x)$ xác định trên miền D .

❶ Definition (Hàm tuần hoàn)

Hàm số $y = f(x)$ được gọi là **tuần hoàn** nếu tồn tại số T sao cho $f(x + T) = f(x)$ với mọi $x \in D$.

Nói cách khác, hàm số sẽ lặp lại sau một đoạn nhất định.

Số T nhỏ nhất thỏa mãn $f(x + T) = f(x)$ được gọi là **chu kỳ** của hàm tuần hoàn.

Vì sao số T cần là nhỏ nhất?

Ta thấy rằng, nếu $f(x + T) = f(x)$ với mọi $x \in D$, ta thay x bởi $x + T$ thì thu được $f(x + T + T) = f(x + T)$, hay $f(x + 2T) = f(x + T)$. Như vậy ta suy ra $f(x + 2T) = f(x + T) = f(x)$. Tiếp tục như vậy, sau $2T$ hàm số cũng lặp lại đúng trạng thái đó với $3T, 4T, \dots$. Do đó số T nhỏ nhất thỏa mãn đẳng thức $f(x + T) = f(x)$ sẽ là chu kỳ.

❶ Example

Hàm số $y = \sin(x)$ là hàm tuần hoàn với chu kỳ $T = 2\pi$. Do đó chúng ta chỉ cần khảo sát hàm số trong khoảng $(-\pi; \pi)$ thôi là đủ.

Các nghịch lý về tập vô hạn

Tiếp theo chúng ta sẽ xem hết những bài toán hết sức thú vị cùng những lập luận cũng thú vị không kém để thấy rằng có nhiều điều bất ngờ sẽ xảy ra nếu vận dụng những lý luận chặt chẽ.

Nghịch lý Zeno

Zeno là nhà triết học cổ Hy Lạp nổi tiếng với bài toán Achilles và rùa (Achilles là anh hùng trong thần thoại Hy Lạp). Bài toán được phát biểu đơn giản như sau:

Nếu Achilles chạy đua và xuất phát sau con rùa thì Achilles sẽ không bao giờ bắt kịp con rùa.

Bài toán nghe thật mực cười nhưng dưới lập luận của Zeno thì bài toán sẽ trở nên "có lý".

Zeno lập luận như sau: gọi d_1 là khoảng cách ban đầu giữa Achilles và con rùa. Achilles sẽ mất một khoảng thời gian t_1 để đi tới vị trí con rùa. Tuy nhiên trong khoảng thời gian t_1 đó con rùa cũng đã đi một đoạn d_2 nào đó rồi. Dĩ nhiên d_2 sẽ ngắn hơn d_1 . Nhưng nếu quá trình này lặp đi lặp lại, d_n sẽ trở nên càng ngày càng nhỏ, tuy nhiên không bao giờ bằng 0. Nói cách khác, Achilles không bao giờ bắt kịp con rùa.

Dưới góc nhìn của toán học hiện đại, điều này chưa hẳn đúng. Vì thời Zeno chưa có nhiều khái niệm lân công cụ về vô hạn, nên người ta đã công nhận tổng vô hạn sẽ là vô hạn. Học sinh lớp 11 hiện nay khi học tới cấp số nhân lùi vô hạn sẽ biết cách tính tổng

$$\frac{1}{10} + \frac{1}{100} + \cdots + \frac{1}{10^n} = \frac{1}{9}$$

là hữu hạn.

So sánh \mathbb{N} và \mathbb{Z}

Hai tập hợp \mathbb{N} và \mathbb{Z} là các tập vô hạn, như vậy lực lượng của tập hợp nào lớn hơn?

Câu hỏi tưởng chừng như vô vị vì nhìn vào mọi người đều thấy rằng \mathbb{Z} "bao trọn" \mathbb{N} (số nguyên kéo dài vô hạn về bên trái lẫn phải trong khi số tự nhiên chỉ kéo dài vô hạn về bên phải). Tuy nhiên, nhà toán học Cantor đã tìm ra một lý luận đầy *tính thuyết phục* để chứng minh rằng lực lượng của hai tập là bằng nhau.

Ta xét ánh xạ $f : \mathbb{Z} \rightarrow \mathbb{N}$ như sau:

- $f(0) = 0;$
- các số âm của \mathbb{Z} biến thành các số lẻ của \mathbb{N} ;
- các số dương của \mathbb{Z} thì biến thành các số chẵn của \mathbb{N} .

Ví dụ $f(-1) = 1$, $f(-2) = 3$, $f(-3) = 5$ và cứ như vậy tăng lên.

Tương tự với số dương $f(1) = 2$, $f(2) = 4$.

Ta có công thức

$$z = f(n) = \begin{cases} 2n, & \text{nếu } n \geq 0 \\ -1 - 2n, & \text{nếu } n < 0. \end{cases}$$

Như vậy f là đơn ánh vì hai phần tử khác nhau của \mathbb{Z} sẽ cho ra hai phần tử khác nhau thuộc \mathbb{N} . Tương tự f cũng là toàn ánh vì mọi phần tử thuộc \mathbb{N} đều có một phần tử từ \mathbb{Z} biến thành. Như vậy f là song ánh. Vậy lực lượng \mathbb{N} và \mathbb{Z} bằng nhau.

Bằng lập luận tương tự cũng có thể chứng minh số phần tử của \mathbb{Q} bằng số phần tử của \mathbb{N} . Những lập luận này đã gây ra tiếng vang lớn vào thời đó.

Ở [hình 2.11](#) cho thấy một cách xây dựng song ánh từ \mathbb{N} tới \mathbb{Z}^2 , trong đó:

- điểm $(0, 0)$ tương ứng với 1;
- điểm $(1, 0)$ tương ứng với 2;
- điểm $(1, 1)$ tương ứng với 3;
- điểm $(0, 1)$ tương ứng với 4;
- cứ tiếp tục như vậy theo hình xoắn vuông.

Vietsub cho [hình 2.11](#): Không có chuyện \mathbb{N} và \mathbb{Z}^2 có cùng số phần tử. Ở đây thuật ngữ "số phần tử" không thực sự chính xác mà nên gọi là "lực lượng" vì khi nói đến các tập vô hạn (tức tập có vô hạn phần tử) thì vô hạn không thể so sánh với vô hạn. Hai tập hợp vô hạn chỉ có thể có cùng lực lượng.



Hình 2.11: Song ánh giữa \mathbb{N} và \mathbb{Z}^2 . Nguồn: https://vk.com/wall-91031095_82482.

Từ đây tập hợp vô hạn có thể chia ra **đếm được** (countable) và **không đếm được** (uncountable). Tiếp theo ta định nghĩa hai dạng tập hợp này.

1. Tập hợp được gọi là **đếm được** khi tồn tại song ánh từ nó tới \mathbb{N} .
2. Tập hợp được gọi là **không đếm được** khi nó không phải là tập đếm được.

Định lý về \mathbb{R}

➊ Theorem 1.1

Tập hợp số thực \mathbb{R} là tập không đếm được.

Chúng ta cần một nhận xét sau:

Khoảng $(0; 1)$ là tương đương với tập \mathbb{R} .

Chúng ta có thể xây dựng một song ánh từ \mathbb{R} tới $(0, 1)$, ví dụ $f(x) = \frac{e^x}{e^x + 1}$.

Khi đó, thay vì chứng minh \mathbb{R} không đếm được, ta chỉ cần chứng minh đoạn $(0; 1)$ không đếm được.

Chứng minh

Cantor đưa ra hai phương pháp chứng minh và cả hai đều độc đáo.

Phương án 1: Phương pháp chéo hóa (diagonalization).

Xét ánh xạ

$$\begin{aligned} 0 &\rightarrow 0, a_{0,0}a_{0,1}a_{0,2}\dots \\ 1 &\rightarrow 0, a_{1,0}a_{1,1}a_{1,2}\dots \\ 2 &\rightarrow 0, a_{2,0}a_{2,1}a_{2,2}\dots \\ &\dots \end{aligned}$$

Ta chứng minh ánh xạ này không phải toàn ánh.

Xét số $y = 0, b_0b_1b_2\dots$ với $b_i \neq a_{i,i}$ với mọi i , tức là trên đường chéo của các số trên ta chọn số b_i khác với số trên đường chéo. Như vậy số y này có chữ số ở vị trí 0 khác $f(0)$, chữ số ở vị trí 1 khác $f(1)$, vân vân và mây mây, nên không tìm được số n nào mà $f(n) = y$. Ta suy ra f không phải toàn ánh và từ đó không phải song ánh.

Phương án 2. Phương pháp dãy các đoạn thẳng đóng bị chặn lồng vào nhau (sequence of closed bounded nested).

Giả sử đoạn $(0; 1)$ đếm được. Khi đó ta có thể liệt kê các phần tử của đoạn là $I = \{x_1, x_2, \dots\}$.

Từ tập I ta lấy ra một đoạn con I_1 sao cho $x_1 \notin I_1$.

Tiếp theo, từ tập I_1 ta lấy ra một đoạn con I_2 sao cho $x_2 \notin I_2$.

Tiếp tục như vậy, ta lấy ra các đoạn con

$$\dots \subset I_n \subset \dots \subset I_2 \subset I_1 \subset I$$

với $x_n \notin I_n$ với mọi $n \in \mathbb{N}$.

Theo định lý về các đoạn thẳng đóng bị chặn lồng vào nhau thì giao của chúng không rỗng, tức là tồn tại số x thuộc giao giao của các tập I_1, \dots, I_n . Phần tử $x \in I_n$ với mọi n . Do $x_n \notin I_n$ và $x \in I_n$ nên $x \neq x_n$ với mọi n , tức là không nằm trong tập I . Điều này mâu thuẫn với giả sử đoạn $(0; 1)$ đếm được, suy ra đoạn $(0; 1)$ là tập không đếm được.

2.1.2 Đa thức

Một số vấn đề về đa thức.

Giới thiệu về đa thức

Definition 2.1 (Đa thức một biến)

Đa thức theo một biến x là hàm số có dạng

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0.$$

Các số a_i được gọi là **hệ số** (hay **coefficient**, **коэффициент**).

Hệ số bậc cao nhất là a_n . Hệ số tự do là a_0 .

Biểu thức a_kx^k được gọi là **hạng tử bậc** k . Hạng tử bậc cao nhất là a_nx^n .

1. Nếu $a_i \in \mathbb{R}$ thì ta nói $P(x)$ là đa thức với hệ số thực.
2. Nếu $a_i \in \mathbb{Q}$ thì ta nói $P(x)$ là đa thức với hệ số hữu tỷ.
3. Nếu $a_i \in \mathbb{Z}$ thì ta nói $P(x)$ là đa thức với hệ số nguyên.

i Definition 2.2 (Bậc của đa thức)

Nếu $a_n \neq 0$ thì số tự nhiên n được gọi là **bậc của đa thức** (hay **degree**, **степень**) và ta kí hiệu $\deg P = n$.

So sánh, cộng, trừ và nhân hai đa thức

Hai đa thức

$$P(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0$$

và

$$Q(x) = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_1 x + b_0$$

bằng nhau khi và chỉ khi $m = n$, và $a_k = b_k$ với mọi $k = 0, 1, \dots, m$.

Khi cộng và trừ hai đa thức $P(x)$ và $Q(x)$ ta thực hiện theo từng hệ số của x^k , nghĩa là

$$P(x) \pm Q(x) = \sum_{k=0}^{\max(m,n)} (a_k \pm b_k) x^k.$$

i Example

Xét hai đa thức

$$\begin{aligned} P(x) &= x^3 - 4x^2 - 5x + 3, \\ Q(x) &= x^4 - 3x^3 + 5x^2 - x - 1. \end{aligned}$$

Lúc này hệ số của hai đa thức $P(x)$ và $Q(x)$ là

$$\begin{aligned} a_4 &= 0, a_3 = 1, a_2 = -4, a_1 = -5, a_0 = 3, \\ b_4 &= 1, b_3 = -3, b_2 = 5, b_1 = -1, b_0 = -1. \end{aligned}$$

Như vậy ta có tổng và hiệu

$$\begin{aligned} P(x) + Q(x) &= (0+1) \cdot x^4 + [1+(-3)] \cdot x^3 + (-4+5) \cdot x^2 + [-5+(-1)] \cdot x + [3+(-1)] \\ &= x^4 - 2x^3 + x^2 - 6x + 2. \\ P(x) - Q(x) &= (0-1) \cdot x^4 + [1-(-3)] \cdot x^3 + (-4-5) \cdot x^2 + [-5-(-1)] \cdot x + [3-(-1)] \\ &= -x^4 + 4x^3 - 9x^2 - 4x + 4. \end{aligned}$$

Khi nhân hai đa thức $P(x)$ và $Q(x)$ ta nhận được đa thức bậc $m + n$ là

$$R(x) = \sum_{k=0}^{m+n} c_k x^k$$

với hệ số c_k được xác định bởi

$$c_k = \sum_{i=0}^k a_i b_{k-i}.$$

Remark

Nếu đa thức $P(x)$ nhận mọi giá trị $x \in \mathbb{R}$ làm nghiệm thì $P(x) \equiv 0$.

Theorem (Bậc của tổng, hiệu và tích của các đa thức)

Cho $P(x)$ và $Q(x)$ là các đa thức bậc m và n tương ứng. Khi đó

1. $\deg(P \pm Q) \leq \max(m, n)$, trong đó
 - Nếu $\deg P \neq \deg Q$ thì dấu bằng xảy ra.
 - Nếu $\deg P = \deg Q$, hay $m = n$, thì $\deg(P \pm Q)$ có thể nhận bất kì giá trị nào nhỏ hơn hoặc bằng m .
2. $\deg(P \cdot Q) = m + n$.

Example

Xét đa thức $P(x) = -x + 1$ và $Q(x) = x + 1$. Khi đó

$$\deg P = 1, \quad \deg Q = 1$$

và

$$P(x) + Q(x) = 2, \quad P(x) - Q(x) = -2x.$$

Như vậy

$$\begin{aligned} \deg(P + Q) &= 0 < \max(\deg P, \deg Q), \\ \deg(P - Q) &= 1 = \max(\deg P, \deg Q). \end{aligned}$$

Đa thức nội suy Lagrange

Trong đại số, công thức nội suy Lagrange cho phép chúng ta tìm được một đa thức $f(x)$ trên trường \mathbb{F} bất kì khi biết được một số cặp $(x_i, f(x_i))$ nhất định với $x_i, f(x_i) \in \mathbb{F}$.

Để tìm đa thức $f(x)$ có bậc n ta cần ít nhất $n + 1$ cặp $(x_i, f(x_i))$ với $1 \leq i \leq n + 1$ và $x_i \neq x_j$ với mọi $i \neq j$.

Khi đó, ta có **công thức nội suy Lagrange** như sau:

$$f(x) = \sum_{i=1}^{n+1} \left(y_i \cdot \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \right).$$

❶ Example 2.3

Giả sử chúng ta có hàm $f(x) = x^2 + x + 1$. Khi đó $f(1) = 3$, $f(-1) = 1$, $f(0) = 1$.

Từ ba cặp $(x_i, f(x_i))$ trên mình sẽ tìm ngược lại $f(x)$ ban đầu.

Theo công thức thì

$$\begin{aligned} f(x) &= y_1 \cdot \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} + y_2 \cdot \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} \\ &\quad + y_3 \cdot \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} \end{aligned}$$

Thay số vào thì ta có

$$f(x) = 3 \cdot \frac{(x - (-1))(x - 0)}{(1 - (-1))(1 - 0)} + 1 \cdot \frac{(x - 1)(x - 0)}{(-1 - 1)(-1 - 0)} + 1 \cdot \frac{(x - 1)(x - (-1))}{(0 - 1)(0 - (-1))}$$

Thu gọn lại ta có $f(x) = x^2 + x + 1$ (đúng với hàm cần tìm).

2.1.3 Phương pháp chứng minh toán học

Chứng minh trực tiếp

Giả sử chúng ta có điều kiện ban đầu là P và ta cần chứng minh mệnh đề Q .

Đối với chứng minh trực tiếp, từ P chúng ta suy ra P_1 nào đó, rồi lại suy ra P_2 từ P_1 . Chúng ta làm vây cho đến khi nhận được mệnh đề Q .

Chứng minh trực tiếp hữu dụng đối với những lời giải tuần tự từng bước.

❶ Example 3.1

Cho tam giác ABC với G, H, O lần lượt là trọng tâm, trực tâm và tâm đường tròn ngoại tiếp tam giác ABC . Chứng minh rằng ba điểm G, H và O thẳng hàng.

Ở đây, với O là tâm đường tròn ngoại tiếp tam giác, ta vẽ đường tròn đó trước (Hình 2.12).

Tiếp theo, ta vẽ đường kính AD .

Lúc này, vì góc \widehat{ACD} chắn nửa đường tròn (AD là đường kính) nên \widehat{ACD} là **góc vuông**, hay CD vuông góc AC .

Tiếp theo, vì H là trực tâm nên BH vuông góc cạnh đối diện AC .

Từ hai kết quả trên suy ra $BH//CD$ vì cùng vuông góc AC .

Tương tự ta cũng có $CH//BD$.

Tứ giác $BHCD$ có hai cặp cạnh song song là $BH//CD$ và $CH//BD$ nên $BHCD$ là hình bình hành.

Giao điểm hai đường chéo của hình bình hành là trung điểm mỗi đường chéo. Gọi I_1 là trung điểm BC thì I_1 cũng là trung điểm HD .

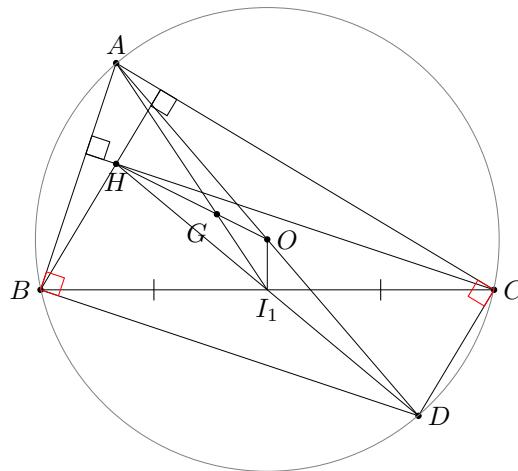
Vì O là trung điểm AD , I_1 là trung điểm HD nên OI_1 là đường trung bình tam giác DHA , hay $\overrightarrow{OI_1} = \frac{1}{2}\overrightarrow{AH}$.

Do G là trọng tâm $\triangle ABC$ và AI_1 là trung tuyến (I_1 là trung điểm BC) nên G chia đoạn thẳng AI_1 theo tỉ lệ $2 : 1$, nghĩa là $\overrightarrow{AG} = 2\overrightarrow{GI_1}$.

Tiếp theo chúng ta biến đổi

$$\begin{aligned}\overrightarrow{AH} &= 2\overrightarrow{OI_1} \\ \overrightarrow{AG} + \overrightarrow{GH} &= 2(\overrightarrow{OG} + \overrightarrow{GI_1}) = 2\overrightarrow{OG} + 2\overrightarrow{GI_1} \\ \overrightarrow{GH} &= 2\overrightarrow{OG}.\end{aligned}$$

Biểu thức cuối cùng chứng tỏ G, H và O thẳng hàng, ngoài ra G chia đoạn thẳng HO theo tỉ lệ $2 : 1$.



Hình 2.12: Đường thẳng Euler

Quy nạp toán học (cơ bản)

Giả sử ta muốn chứng minh một mệnh đề P đúng với mọi $n \geq 1$. Phép quy nạp toán học hoạt động theo ba bước như sau:

1. Chứng minh mệnh đề P đúng với $n = 1$. Đây gọi là *bước cơ sở*.
2. Giả sử mệnh đề P đúng với $n = k \geq 1$. Đây gọi là *giả thiết quy nạp*.
3. Chứng minh mệnh đề P đúng với $n = k + 1$ từ giả thiết quy nạp ở bước 2.

Như vậy phép **quy nạp toán học** (hay **mathematical induction**, **математическая индукция**) hoạt động theo bậc thang. Từ giả thiết quy nạp mệnh đề P đúng với $n = 1$, theo chứng minh ở bước 3 thì mệnh đề P cũng đúng ở bước $n = 1 + 1 = 2$. Do P đúng với $n = 2$ nên cũng đúng ở $n = 3$. Cứ tiếp tục như vậy P sẽ đúng với mọi $n \geq 1$. Đây là sự hiệu quả đáng kinh ngạc của phép quy nạp toán học.

Example 3.2

Chứng minh công thức tổng quát cho tổng $1 + 2 + \dots + n$ là $\frac{n(n+1)}{2}$.

Với $n = 1$ thì $1 = \frac{1(1+1)}{2}$. Như vậy công thức đúng cho $n = 1$. Đây là bước cơ sở.

Giả sử mệnh đề đúng với $n = k \geq 1$. Nghĩa là $1 + 2 + \dots + k = \frac{k(k+1)}{2}$. Đây là giả thiết quy nạp.

Bây giờ ta cần chứng minh mệnh đề đúng với $n = k + 1$, nghĩa là ta cần chứng minh

$$1 + 2 + \dots + k + (k+1) = \frac{(k+1)(k+2)}{2}.$$

Từ giả thiết quy nạp ta suy ra

$$\begin{aligned} 1 + 2 + \dots + k + (k+1) &= \frac{k(k+1)}{2} + (k+1) \\ &= \frac{k(k+1) + 2(k+1)}{2} \\ &= \frac{(k+1)(k+2)}{2}. \end{aligned}$$

Vậy là ta đã có điều cần chứng minh, và công thức đã được chứng minh đúng với mọi $n \geq 1$.

Remark 3.1

Tùy thuộc bài toán, bước cơ sở có thể không phải bắt đầu từ 1 mà là một số nguyên dương nào đó khác.

Quy nạp toán học (mạnh)

Quy nạp toán học mạnh (strong induction) là một phiên bản mạnh hơn của phép quy nạp toán học ở trên.

Trong phép quy nạp toán học mạnh, *giả thiết quy nạp* sẽ được thay bằng: Giả sử mệnh đề P ĐÚNG TỐI $n = k \geq 1$.

Điểm khác biệt của quy nạp mạnh với quy nạp ban đầu là việc giả thiết quy nạp đúng với mọi n nhỏ hơn hoặc bằng k và chúng ta sẽ chứng minh mệnh đề đúng với $n = k + 1$. Trong khi đó ở quy nạp ban đầu thì giả thiết quy nạp chỉ đúng với $n = k$ thôi.

Tính đúng đắn của quy nạp mạnh vẫn giống như quy nạp thông thường, nghĩa là vẫn hoạt động theo bậc thang. Khi mệnh đề đúng với $n = 1$ (bước cơ sở) thì chứng minh ở bước 3 cho kết quả mệnh đề P đúng với $n = 2$. Do mệnh đề P đúng với $n = 1, 2$ nên sẽ đúng với $n = 3$. Cứ tiếp tục như vậy thì P sẽ đúng với mọi $n \geq 1$.

Tại sao chúng ta cần dùng quy nạp toán học mạnh khi bản chất vẫn giống quy nạp thông thường?

Lý do là đôi khi chúng ta chứng minh $n = k + 1$ không dựa trên $n = k$, mà là một điểm nào đó nhỏ hơn, nghĩa là trong khoảng $[1, k]$.

Example 3.3

Dãy số Fibonacci định nghĩa bởi $F_1 = F_2 = 1$ và $F_{n+2} = F_{n+1} + F_n$ với mọi $n \geq 1$. Chứng minh rằng

công thức tổng quát của dãy Fibonacci là

$$F_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right].$$

Khi $n = 1$ thì ta có $F_1 = 1$, đúng với điều kiện ban đầu.

Khi $n = 2$ thì ta có $F_2 = 1$, đúng với điều kiện ban đầu.

Giả thiết quy nạp: giả sử với mọi $n = k \geq 1$ ta đều có $F_k = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^k - \left(\frac{1-\sqrt{5}}{2} \right)^k \right]$.

Khi đó, với $n = k + 1$, ta có

$$\begin{aligned} F_{k+1} &= F_k + F_{k-1} \\ &= \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^k - \left(\frac{1-\sqrt{5}}{2} \right)^k \right] + \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^{k-1} - \left(\frac{1-\sqrt{5}}{2} \right)^{k-1} \right] \\ &= \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^k + \left(\frac{1+\sqrt{5}}{2} \right)^{k-1} \right] - \frac{1}{\sqrt{5}} \left[\left(\frac{1-\sqrt{5}}{2} \right)^k + \left(\frac{1-\sqrt{5}}{2} \right)^{k-1} \right] \\ &= \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^{k-1} \left(\frac{1+\sqrt{5}}{2} + 1 \right) - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^{k-1} \left(\frac{1-\sqrt{5}}{2} + 1 \right). \end{aligned}$$

Để ý rằng

$$\begin{aligned} \frac{1+\sqrt{5}}{2} + 1 &= \frac{3+\sqrt{5}}{2} = \frac{6+2\sqrt{5}}{4} = \frac{1+2\sqrt{5}+(\sqrt{5})^2}{4} \\ &= \frac{(1+\sqrt{5})^2}{2^2} = \left(\frac{1+\sqrt{5}}{2} \right)^2, \end{aligned}$$

tương tự ta cũng có $\frac{1-\sqrt{5}}{2} + 1 = \left(\frac{1-\sqrt{5}}{2} \right)^2$, nên ở trên suy ra

$$\begin{aligned} F_{k+1} &= \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^{k-1} \left(\frac{1+\sqrt{5}}{2} \right)^2 + \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^{k-1} \left(\frac{1-\sqrt{5}}{2} \right)^2 \\ &= \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^{k+1} - \left(\frac{1-\sqrt{5}}{2} \right)^{k+1} \right]. \end{aligned}$$

Như vậy mệnh đề đúng với $n = k + 1$ và ta có điều phải chứng minh.

Ở đây quy nạp mạnh thể hiện ở việc ta cần giả thiết đúng với $n = k$ và $n = k - 1$ để chứng minh cho $n = k + 1$.

Chứng minh bằng phản chứng

Giả sử chúng ta có điều kiện P và cần chứng minh kết quả Q . Điều này tương đương với mệnh đề logic $P \Rightarrow Q$. Chứng minh bằng phản chứng dựa trên sự tương đương của các mệnh đề logic, nghĩa là

$$(P \Rightarrow Q) \iff (\bar{Q} \Rightarrow \bar{P}).$$

Khi đó, từ kết quả Q cần chứng minh, chúng ta giả sử rằng đang có \bar{Q} , tức là phủ định của mệnh đề cần chứng minh. Bằng các lập luận logic chúng ta sẽ suy ra được điều trái với điều kiện ban đầu, tức là \bar{P} . Đây là cơ sở của phép chứng minh bằng phản chứng.

❶ Example 3.4

Chứng minh rằng với mọi số tự nhiên n , nếu n^3 chia hết cho 3 thì n chia hết cho 3.

Ở đây:

1. Điều kiện, tức mệnh đề P , là " n^3 chia hết cho 3".
2. Mệnh đề cần chứng minh Q là " n chia hết cho 3".

Ta suy ra:

1. Phủ định của mệnh đề P là " n^3 không chia hết cho 3", tức mệnh đề \bar{P} .
2. Phủ định của mệnh đề Q là " n không chia hết cho 3", tức mệnh đề \bar{Q} .

Như vậy phép phản chứng đưa ta tới việc chứng minh: nếu số tự nhiên n không chia hết cho 3 thì n^3 không chia hết cho 3.

Nếu n không chia hết cho 3 thì n có dạng $3k + 1$ hoặc $3k + 2$ với $k \in \mathbb{Z}$.

- nếu $n = 3k + 1$ thì $n^3 = 27k^3 + 27k^2 + 9k + 1$, khi chia 3 sẽ dư 1. Khi đó n^3 không chia hết cho 3;
- nếu $n = 3k + 2$ thì $n^3 = 27k^3 + 54k^2 + 36k + 8$, khi chia 3 sẽ dư 2 (vì 8 chia 3 dư 2). Khi đó n^3 cũng không chia hết cho 3.

Như vậy khi n không chia hết cho 3 (mệnh đề \bar{Q}) thì n^3 cũng không chia hết cho 3 (mệnh đề \bar{P}). Theo phản chứng ta có, nếu n^3 chia hết cho 3 (mệnh đề P) thì n chia hết cho 3 (mệnh đề Q). Đây là điều phải chứng minh.

2.1.4 Bảng thuật ngữ đại số

Tiếng Việt	Tiếng Anh	Tiếng Nga
tập hợp	set	множество
tập hợp rỗng	empty set	пустое множество
lực lượng (của tập hợp)	cardinality	мощность
phép giao tập hợp	intersection of sets	пересечение множеств
phép hợp tập hợp	union of sets	объединение множеств
phép hiệu hai tập hợp	set difference	разность двух множеств
ánh xạ	map	отображение
đơn ánh	injection	инъекция
toàn ánh	one-to-one map	инъективное отображение
	surjection	сюръекция
	onto map	сюръективное отображение
song ánh	bijection	биекция
	one-to-one and onto map	биективное отображение
		взаимно однозначное отображение
hàm số	function	функция
hàm đơn điệu	monotonic function	монотонная функция
(hàm số) đồng biến	increasing	возрастающая
(hàm số) tăng nghiêm ngặt	strictly increasing	строго возрастающая
(hàm số) nghịch biến	decreasing	убывающая
(hàm số) giảm nghiêm ngặt	strictly decreasing	строго убывающая
hàm số chẵn	even function	четная функция
hàm số lẻ	odd function	нечетная функция
hàm cộng tính	additive function	аддитивная функция
hàm nhân tính	multiplicative function	мультипликативная функция
hàm tuần hoàn	periodic function	периодическая функция
đa thức	polynomial	многочлен
bậc (đa thức)	degree	степень
hệ số (đa thức)	coefficient	коэффициент
quy nạp (toán học)	(mathematical) induction	(математическая) индукция

2.2 Lý thuyết nhóm

2.2.1 Lý thuyết nhóm



Hình 2.13: Évariste Galois (1811-1832)

Nhóm

Nhóm và nhóm con

Definition 1.16 (Nhóm)

Một tập hợp G và toán tử hai ngôi \star trên G tạo thành một **nhóm** (hay **group**, **группа**) nếu:

1. Tồn tại phần tử $e \in G$ sao cho với mọi $g \in G$ thì

$$g \star e = e \star g = g.$$

Khi đó e được gọi là **phần tử đơn vị** của G .

2. Với mọi $g \in G$, tồn tại $g' \in G$ sao cho

$$g \star g' = g' \star g = e.$$

Khi đó g' được gọi là **phần tử nghịch đảo** của g .

3. Tính kết hợp: với mọi $a, b, c \in G$ thì

$$a \star (b \star c) = (a \star b) \star c.$$

Definition 1.17 (Nhóm Abel)

Nếu nhóm G có thêm tính giao hoán, tức là với mọi $a, b \in G$ thì $a * b = b * a$ thì G gọi là **nhóm giao hoán** (**commutative group**, **коммутативная группа**) hoặc **nhóm Abel** (**abelian group**, **абелева группа**).

Example 1.12

Xét tập hợp số nguyên \mathbb{Z} và phép cộng hai số nguyên.

1. Phần tử đơn vị là 0 vì với mọi $a \in \mathbb{Z}$ thì $a + 0 = 0 + a = a$.
2. Với mọi $a \in \mathbb{Z}$, phần tử nghịch đảo là $-a$ vì $a + (-a) = (-a) + a = 0$.
3. Phép cộng số nguyên có tính kết hợp do đó thỏa mãn điều kiện về tính kết hợp.

Như vậy $(\mathbb{Z}, +)$ tạo thành nhóm. Lưu ý do phép cộng hai số nguyên có tính giao hoán nên đây cũng là nhóm Abel.

Example 1.13

Xét tập hợp số hữu tỉ khác 0 là \mathbb{Q}^* và phép nhân hai số hữu tỉ. Do $a, b \in \mathbb{Q}^*$ nên tích $a \cdot b$ cũng khác 0 , do đó cũng thuộc \mathbb{Q}^* .

1. Phần tử đơn vị là 1 vì với mọi $a \in \mathbb{Q}^*$ thì $a \cdot 1 = 1 \cdot a = a$.
2. Với mọi $a \in \mathbb{Q}^*$, phần tử nghịch đảo là $\frac{1}{a}$ vì $a \cdot \frac{1}{a} = \frac{1}{a} \cdot a = 1$.
3. Phép nhân hai số hữu tỉ có tính kết hợp do đó thỏa mãn điều kiện về tính kết hợp.

Tương tự như nhóm $(\mathbb{Z}, +)$, nhóm (\mathbb{Q}^*, \cdot) cũng là nhóm Abel.

Definition 1.18 (Order của nhóm)

Order (hay **порядок**) của nhóm G là lực lượng (hay số phần tử, **carninality**, **мощность**) của nhóm đó và kí hiệu là $|G|$.

Đối với nhóm có vô hạn phần tử, ta quy ước order của nhóm bằng 0 , ví dụ như với hai nhóm $(\mathbb{Z}, +)$ và (\mathbb{Q}^*, \cdot) ở trên.

Nhóm con

Definition 1.19 (Nhóm con)

Cho nhóm $(G, *)$. Tập hợp $H \subset G$ được gọi là **nhóm con** (hay **subgroup**, **подгруппа**) của G nếu với mọi $a, b \in H$ thì $a * b \in H$.

Nói cách khác, toán tử $*$ đóng với các phần tử trong H .

❶ Example 1.14

Xét nhóm $(\mathbb{Z}, +)$ như trên. Ta xét tập con gồm các số chẵn của nó

$$2\mathbb{Z} = \{\dots, -4, -2, 0, 2, 4, \dots\}.$$

Ta thấy rằng tổng hai số chẵn vẫn là số chẵn, nghĩa là phép cộng số nguyên đóng trên $2\mathbb{Z}$.

Do đó $(2\mathbb{Z}, +)$ là nhóm con của $(\mathbb{Z}, +)$.

Tổng quát, mọi tập hợp có dạng $n\mathbb{Z}$ đều là nhóm con của $(\mathbb{Z}, +)$.

❶ Theorem 1.2 (Định lý Lagrange)

Order của nhóm luôn chia hết order của một nhóm con bất kì của nó.

Nhóm vòng**❶ Definition 1.20 (Nhóm vòng)**

Nhóm G được gọi là **nhóm vòng** (hay **cyclic group**, **циклическая группа**) nếu tồn tại phần tử $g \in G$ mà mọi phần tử trong G đều được biểu diễn dưới dạng g^i . Khi đó ta kí hiệu $G = \langle g \rangle$ hoặc $G = \{g, g^1, \dots, g^n\}$.

Thông thường ta quy ước $g^n = g^0 = e$.

Đối với nhóm $(\mathbb{Z}_n, +_n)$ xác định phép cộng modulo n , ta kí hiệu

$$ig = \underbrace{g + g + \dots + g}_{i \text{ lần}}.$$

Ta viết

$$G = \{1g, 2g, 3g, \dots, ng\}.$$

Phần tử g được gọi là **phần tử sinh** (hay **образующий элемент**) của nhóm vòng G .

Như vậy, số lượng phần tử sinh của \mathbb{Z}_n là $\varphi(n)$ với φ là hàm Euler. Lúc này điều kiện để phần tử j là phần tử sinh tương đương với

$$\langle j \rangle = \mathbb{Z}_n \iff (j, n) = 1.$$

❶ Definition 1.21 (Elementary abelian group)

Nhóm vòng được gọi là **elementary abelian** (hay **примарная абелева группа**) nếu bậc của nhóm là số nguyên tố.

Coset**i Definition 1.22 (Coset, lớp kề)**

Cho nhóm G và nhóm con H của G .

Coset trái của H đối với phần tử $g \in G$ là tập hợp

$$gH = \{gh : h \in H\}.$$

Tương tự, coset phải là tập hợp

$$Hg = \{hg : h \in H\}.$$

Từ đây nếu không nói gì thêm ta ngầm hiểu là coset trái.

Ví dụ với nhóm con $2\mathbb{Z}$ của \mathbb{Z} , ta thấy rằng:

1. Nếu $g \in \mathbb{Z}$ là lẻ thì khi cộng với bất kì phần tử nào của $2\mathbb{Z}$ ta nhận được số lẻ.
2. Nếu $g \in \mathbb{Z}$ là chẵn thì khi cộng với bất kì phần tử nào của $2\mathbb{Z}$ ta nhận được số chẵn.

Nói cách khác, coset của $2\mathbb{Z}$ chia tập \mathbb{Z} thành

$$\begin{aligned} 0 + 2\mathbb{Z} &= \{\dots, -4, -2, 0, 2, 4, \dots\}, \\ 1 + 2\mathbb{Z} &= \{\dots, -3, -1, 1, 3, 5, \dots\}. \end{aligned}$$

Rõ ràng hai coset trên rời nhau.

i Remark 1.2

Hai coset bắt kì hoặc rời nhau, hoặc trùng nhau.

i Chứng minh

Nếu hai coset rời nhau thì không có gì phải nói. Ta chứng minh trường hợp còn lại.

Giả sử $g_1H \cap g_2H \neq \emptyset$. Như vậy tồn tại $h_1, h_2 \in H$ mà $g_1h_1 = g_2h_2$.

Do $h_1^{-1} \in H$, ta có $g_1 = g_2h_2h_1^{-1}$, nghĩa là $g_1 \in g_2H$.

Mà mọi phần tử trong g_1H có dạng g_1h nên $g_1h = g_2h_2h_1^{-1}h$. Do H là nhóm con của G nên $h_2h_1^{-1}h \in H$.

Từ đó $g_1H \subseteq g_2H$. Tương tự ta cũng có $g_2H \subseteq g_1H$. Vậy $g_1H = g_2H$.

Normal Subgroup**i Definition 1.23 (Normal Subgroup)**

Nhóm con H của G được gọi là **normal subgroup** (hay **нормальная подгруппа**, nhóm con chuẩn tắc) nếu với mọi $g \in G$ ta có coset trái trùng với coset phải.

$$gH = Hg \quad \text{với mọi } g \in G.$$

Nếu H là normal subgroup của G ta kí hiệu $H \triangleleft G$. Khi đó, với mọi $a, b \in G$ thì $(aH)(bH) = (ab)H$.

i Definition 1.24 (Quotient Group)

Với nhóm G và normal subgroup của nó là H .

Quotient Group (hay **nhóm thương**) được kí hiệu là G/H và được định nghĩa là tập hợp các coset tương ứng với normal subgroup H .

$$G/H = \{gH : g \in H\}.$$

Ta thấy rằng điều này chỉ xảy ra nếu H là normal subgroup.

Quotient Group còn được gọi là **Factor Group** (hay **nhóm nhân tử**).

i Example 1.15

Với nhóm \mathbb{Z} và normal subgroup của nó là $2\mathbb{Z}$.

Ta thấy

$$\mathbb{Z}/2\mathbb{Z} = \{0 + 2\mathbb{Z}, 1 + 2\mathbb{Z}\}.$$

Direct sum of modules

Прямая сумма

Có hai dạng tổng là external và internal.

i Definition 1.25 (External direct sum)

Giả sử ta có các nhóm $(G_1, *)$, (G_2, \star) , ..., (G_t, \circ) . Khi đó external direct sum của các nhóm G_1, \dots, G_t là:

$$G = G_1 \times G_2 \times \dots \times G_t, \quad (G, \square).$$

Giả sử $g = (g_1, g_2, \dots, g_t) \in G$ với $g_i \in G_i$, và $g' = (g'_1, g'_2, \dots, g'_t) \in G$ với $g'_i \in G_i$. Khi đó:

$$g \square g' = (g_1 * g'_1, g_2 \star g'_2, \dots, g_t \circ g'_t).$$

i Definition 1.26 (Internal direct sum)

Giả sử ta có nhóm (G, \circ) và các nhóm con G_1, G_2, \dots, G_t của G . Khi đó internal direct sum là:

1. Với mọi $g \in G$ thì $g = g_1 \circ g_2 \circ \dots \circ g_t$ với $g_i \in G_i$.
2. Với mọi i, j mà $i \neq j$, $1 \leq i, j \leq t$ ta có

$$g_i \circ g_j = g_j \circ g_i$$

với mọi $g_i \in G_i$ và $g_j \in G_j$.

Nhóm hoán vị**Nhóm hoán vị**

Xét tập hợp $\{1, 2, \dots, n\}$.

Ta gọi S_n là tập tất cả hoán vị của tập hợp trên. Như vậy S_n có $n!$ phần tử.

Nếu ta lấy hoán vị gốc là $(1, 2, \dots, n)$, mỗi hoán vị đều có thể được biểu diễn bằng hai hàng như sau:

$$\sigma = \begin{pmatrix} 1 & 2 & \dots & n \\ \sigma(1) & \sigma(2) & \dots & \sigma(n) \end{pmatrix}.$$

Ta định nghĩa toán tử trên S_n . Với hai hoán vị σ và τ , hoán vị $\sigma * \tau$ là vị trí của σ theo τ . Nói cách khác, nếu

$$\sigma = \begin{pmatrix} 1 & 2 & \dots & n \\ \sigma(1) & \sigma(2) & \dots & \sigma(n) \end{pmatrix}$$

và

$$\tau = \begin{pmatrix} 1 & 2 & \dots & n \\ \tau(1) & \tau(2) & \dots & \tau(n) \end{pmatrix}$$

thì

$$\sigma * \tau = \begin{pmatrix} 1 & 2 & \dots & n \\ \sigma(\tau(1)) & \sigma(\tau(2)) & \dots & \sigma(\tau(n)) \end{pmatrix}$$

Tập các hoán vị S_n và toán tử như trên tạo thành một nhóm và được gọi là **nhóm hoán vị**.

Example

Xét nhóm hoán vị S_5 .

Gọi $x = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 1 & 2 & 5 \end{pmatrix}$ và $y = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 4 & 3 & 2 \end{pmatrix}$.

Khi đó, đặt $z = x * y$ thì

$$\begin{aligned} z(1) &= x(y(1)) = x(5) = 5, \\ z(2) &= x(y(2)) = x(1) = 4, \\ z(3) &= x(y(3)) = x(4) = 2, \\ z(4) &= x(y(4)) = x(3) = 1, \\ z(5) &= x(y(5)) = x(2) = 3. \end{aligned}$$

Như vậy

$$z = x * y = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 2 & 1 & 3 \end{pmatrix}.$$

Remark

Trong một hoán vị, khi biểu diễn trên hai hàng thì thứ tự viết không quan trọng, miễn là đảm bảo i tương ứng với $\sigma(i)$ trên từng cột.

❶ Example

Xét hoán vị $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 1 & 2 & 5 \end{pmatrix}$ thuộc S_5 .

Ta có $\sigma(1) = 4$, $\sigma(2) = 3$, $\sigma(3) = 1$, $\sigma(4) = 2$ và $\sigma(5) = 5$. Như vậy hai cách viết sau là giống nhau

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 1 & 2 & 5 \end{pmatrix} = \begin{pmatrix} 3 & 4 & 5 & 1 & 2 \\ 1 & 2 & 5 & 4 & 3 \end{pmatrix}.$$

Chu trình độc lập

Xét nhóm hoán vị S_n và hoán vị σ thuộc S_n .

Khi đó tồn tại các tập $\{i_1, i_2, \dots, i_k\} \subset \{1, 2, \dots, n\}$ sao cho $\sigma(i_1) = i_2$, $\sigma(i_2) = i_3$, ..., $\sigma(i_{k-1}) = \sigma(i_k)$ và $\sigma(i_k) = i_1$.

❶ Example

Xét S_5 và hoán vị $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 4 & 3 & 2 \end{pmatrix}$.

Ta thấy rằng $\sigma(1) = 5$, $\sigma(5) = 2$, $\sigma(2) = 1$. Như vậy ta có chu trình $1 \rightarrow 5 \rightarrow 2 \rightarrow 1$.

Tương tự, $\sigma(3) = 4$ và $\sigma(4) = 3$. Như vậy ta có thêm chu trình $3 \rightarrow 4 \rightarrow 3$.

Hai chu trình trên không chứa phần tử chung nên chúng được gọi là hai **chu trình độc lập**.

❶ Remark

Mọi hoán vị đều có thể viết được dưới dạng tích của các chu trình độc lập.

Chu trình có thể chứa một phần tử, tức $\sigma(i) = i$ với mọi i .

❶ Example

Hoán vị $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 4 & 3 & 2 \end{pmatrix}$ như trên thì ta có thể viết lại thành $\sigma = (1, 5, 2)(3, 4)$.

❶ Remark

Thứ tự của chu trình trong tích không quan trọng. Điều này dễ thấy vì các chu trình độc lập nhau, do đó dù viết trước hay sau thì hoán vị vẫn nằm trong chu trình đó.

Để giải thích rõ hơn, chúng ta có thể xem mỗi chu trình độc lập như một hoán vị, trong đó các phần tử không thuộc chu trình đứng yên.

Ví dụ với chu trình $(1, 5, 2) = (1, 5, 2)(3, 4)$ ở trên tương đương với

$$p_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 3 & 4 & 2 \end{pmatrix},$$

và với chu trình $(3, 4) = (3, 4)(1)(2)(5)$ tương đương với

$$p_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 4 & 3 & 5 \end{pmatrix} = \underbrace{\begin{pmatrix} 5 & 1 & 3 & 4 & 2 \\ 5 & 1 & 4 & 3 & 2 \end{pmatrix}}_{p_1} \cdot \underbrace{\begin{pmatrix} 5 & 1 & 3 & 4 & 2 \\ 5 & 1 & 4 & 3 & 2 \end{pmatrix}}_{p_2}.$$

Do đó tích của chúng (hay toán tử trên nhóm hoán vị) sẽ cho ra kết quả hoán vị ban đầu:

$$\underbrace{\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 4 & 3 & 2 \end{pmatrix}}_{\sigma} = \underbrace{\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 3 & 4 & 2 \end{pmatrix}}_{p_1} * \underbrace{\begin{pmatrix} 5 & 1 & 3 & 4 & 2 \\ 5 & 1 & 4 & 3 & 2 \end{pmatrix}}_{p_2}.$$

Quasigroup

Definition 1.27 (Quasigroup)

Tập Q và phép toán hai ngôi \star được gọi là **quasigroup** (hay **квазигруппа**) nếu với mọi $a, b \in Q$, tồn tại duy nhất hai phần tử $x, y \in Q$ sao cho

$$a \star x = b, \quad y \star a = b.$$

Example 1.20

Mọi nhóm đều là quasigroup.

Example 1.21

$(\mathbb{Z}, -)$ không phải là nhóm nhưng là quasigroup.

Chứng minh

Với mọi $a, b \in \mathbb{Z}$ ta tìm x, y sao cho $a - x = b$ và $y - a = b$.

Khi đó $x = a - b$ và $y = a + b$, nói cách khác là tồn tại hai phần tử duy nhất x, y

Remark 1.6

Quasigroup không có tính kết hợp nên chúng ta không thể định nghĩa phép tính a^n như với nhóm.

Definition 1.28 (d -quasigroup)

Xét quasigroup (Q, g) với g là ánh xạ

$$g : Q^d \rightarrow Q, \quad d \geq 2$$

được gọi là **d -quasigroup** và g được gọi là **toán tử quasigroup**.

Định nghĩa quasigroup ở đầu bài tương ứng với $d = 2$ (với g là toán tử hai ngôi).

i Definition 1.29 (Bảng Latin)

Bảng Latin là bảng gồm k hàng và k cột. Ta viết các số từ 0 tới $k - 1$ lên bảng sao cho mỗi hàng có k phần tử khác nhau và mỗi cột cũng có k phần tử khác nhau.

Ví dụ, với $k = 2$ ta có bảng

0	1
1	0

Ví dụ, với $k = 3$ ta có bảng

0	2	1
1	0	2
2	1	0

Mỗi ô được biểu diễn bởi bộ ba (тройка) (i, j, t) với

- i là vị trí hàng;
- j là vị trí cột;
- t là giá trị tại ô (i, j) .

i Definition 1.30 (Homotopy)

Giả sử (P, \star) và $(Q, *)$ là hai quasigroup. Khi đó **quasigroup homotopy** từ P tới Q là bộ ba (α, β, γ) biểu diễn ba ánh xạ từ P tới Q thỏa

$$\alpha(x) * \beta(y) = \gamma(x \star y)$$

với mọi $x, y \in P$.

i Definition 1.31 (Isotopy)

Khi cả ba ánh xạ α, β và γ đều là song ánh thì ta nói homotopy là **isotopy** (hay **изотопия**).

i Definition 1.32 (Autotopy)

Autotopy là isotopy tới chính nó, nghĩa là $P \equiv Q$.

i Remark 1.7

Isotopy là quan hệ tương đương.

$$Q_1 \sim Q_2 \iff Q_1 \text{ isotopy với } Q_2.$$

❶ Definition 1.33 (Parastrophe)

Từ toán tử ban đầu \star ta định nghĩa thêm năm toán tử khác là:

1. Toán tử \circ với $x \circ y = y \star x$ là toán tử đối của toán tử \star .
2. Toán tử \backslash với $x \backslash y = z$ tương đương với $y = x \star z$.
3. Toán tử đối của \backslash .
4. Toán tử $/$.
5. Toán tử đối của $/$.

Như vậy có tất cả sáu toán tử quasigroup và ta gọi tập các toán tử đó là **parastrophe** (hay **conjugation**, **парастрофия**).

❶ Definition 1.34 (Loop)

Loop (hay **лупа**) là quasigroup (Q, \star) với phần tử đơn vị e sao cho với mọi $x \in Q$ thì

$$e \star x = x \star e = x.$$

Khi đó mỗi phần tử trong quasigroup sẽ có phần tử nghịch đảo (inverse) trái và phải tương ứng. Lưu ý rằng hai nghịch đảo không nhất thiết bằng nhau.

❶ Definition 1.35 (Nhóm nhân)

Ta định nghĩa phép nhân (toán tử nhân)

$$\begin{aligned} L_x : Q &\rightarrow Q, & L_x(y) &= x \star y, \\ R_x : Q &\rightarrow Q, & R_x(y) &= y \star x. \end{aligned}$$

Đặt

$$\text{mult}(Q) = \langle L_q, R_q : q \in Q \rangle.$$

Ta nói $\text{mult}(Q)$ là **nhóm nhân của quasigroup** (hay **группа умножений квазигруппы**).

❶ Definition 1.36 (Chỉ số kết hợp)

Ta gọi **bộ ba kết hợp** (hay **ассоциативная тройка**) là ba phần tử $a, b, c \in Q$ sao cho

$$a \star (b \star c) = (a \star b) \star c,$$

hay tương đương với

$$R_c(L_a(b)) = L_a(R_c(b)).$$

Khi đó **chỉ số kết hợp** (hay **индекс ассоциативности**) là số lượng bộ ba kết hợp trong quasigroup.

❶ Remark 1.8

Mục tiêu của quasigroup trong mật mã học là làm yếu tính kết hợp xuống. Như vậy nếu quasigroup có càng nhiều bộ ba kết hợp thì càng dễ bị tấn công hơn.

❶ Remark 1.9

Số lượng bộ ba kết hợp của quasigroup với order n thì không nhỏ hơn n .

[TODO] Chứng minh tính chất này.

2.2.2 Homomorphisms

Group homomorphism

Đồng cấu nhóm

❶ Definition 2.3 (Homomorphism, Đồng cấu nhóm)

Xét hai nhóm (G, \star) và $(H, *)$ và một ánh xạ $f : G \rightarrow H$.

Ánh xạ f được gọi là **đồng cấu** (hay **homomorphism**) nếu với mọi g_1, g_2 thuộc G ta có $f(g_1 \star g_2) = f(g_1) * f(g_2)$.

Do g_1, g_2 là các phần tử thuộc G nên toán tử giữa chúng là \star . Trong khi đó $f(g_1), f(g_2)$ là các phần tử thuộc H nên toán tử giữa chúng là $*$.

❶ Remark 2.2

1. Gọi e_G là phần tử đơn vị của G và e_H là phần tử đơn vị của H . Khi đó $f(e_G) = e_H$.
2. Với mọi phần tử $g \in G$, nếu g^{-1} là nghịch đảo của g trong G thì $f(g^{-1}) = f(g)^{-1}$.

❶ Chứng minh

1. Nếu e_G là phần tử đơn vị của G thì với mọi $g \in G$ ta có $g \star e_G = e_G \star g = g$. Ta lấy f cả ba vế và theo định nghĩa homomorphism thu được

$$f(g \star e_G) = f(e_G \star g) = f(g) \Rightarrow f(g) * f(e_G) = f(e_G) * f(g) = f(g).$$

Đẳng thức trên đúng với mọi $g \in G$ nên đúng với mọi $f(g)$, suy ra $f(e_G)$ là phần tử đơn vị trong nhóm $(H, *)$ và do đó $f(e_G) = e_H$.

2. Từ việc tìm ra phần tử đơn vị, ta cũng chứng minh được tính chất nghịch đảo trên.

Các loại homomorphism

Tương tự như ánh xạ, chúng ta có các loại homomorphism sau

❶ Definition 2.4 (Monomorphism, Đơn cấu)

Ánh xạ được gọi là **đơn cấu** (hay **monomorphism**) nếu nó là ánh xạ one-to-one (đơn ánh). Nói cách khác, với mọi $g_1, g_2 \in G$ mà $g_1 \neq g_2$ thì $f(g_1) \neq f(g_2)$.

❷ Definition 2.5 (Epimorphism, Toàn cấu)

Ánh xạ được gọi là **toàn cấu** (hay **epimorphism**) nếu nó là ánh xạ onto (tổn ánh). Nói cách khác, với mọi $h \in H$ thì tồn tại $g \in G$ mà $f(g) = h$.

❸ Definition 2.6 (Isomorphism, Đẳng cấu)

Ánh xạ được gọi là **đẳng cấu** (hay **isomorphism**) nếu nó là ánh xạ one-to-one và onto (song ánh). Nói cách khác, ánh xạ này vừa là đơn cấu, vừa là toàn cấu.

❶ Theorem 2.2 (Định lí Cayley)

Mọi nhóm hữu hạn đều đẳng cấu (isomorphism) với một nhóm con nào đó của nhóm hoán vị.

❷ Chứng minh định lí Cayley

Giả sử ta có nhóm hữu hạn $G = \{g_1, g_2, \dots, g_n\}$.

Với mỗi $g \in G$, ta xây dựng hoán vị φ_g theo g :

$$\begin{pmatrix} g_1 & g_2 & \dots & g_i & \dots & g_n \\ g_1g & g_2g & \dots & g_ig & \dots & g_ng \end{pmatrix} = \varphi_g$$

Ta chọn $g', g'' \in G$. Khi đó:

$$\begin{aligned} \varphi_{g'g''} &= \begin{pmatrix} g_1 & g_2 & \dots & g_i & \dots & g_n \\ g_1g'g'' & g_2g'g'' & \dots & g_ig'g'' & \dots & g_ng'g'' \end{pmatrix} \\ &= \begin{pmatrix} g_1 & g_2 & \dots & g_i & \dots & g_n \\ g_1g' & g_2g' & \dots & g_ig' & \dots & g_ng' \end{pmatrix} \\ &\quad \times \begin{pmatrix} g_1g' & g_2g' & \dots & g_ig' & \dots & g_ng' \\ (g_1g')g'' & (g_2g')g'' & \dots & (g_ig')g'' & \dots & (g_ng')g'' \end{pmatrix}. \end{aligned}$$

Do

$$\begin{pmatrix} g_1 & g_2 & \dots & g_i & \dots & g_n \\ g_1g' & g_2g' & \dots & g_ig' & \dots & g_ng' \end{pmatrix} = \varphi_{g'},$$

và

$$\begin{pmatrix} g_1g' & g_2g' & \dots & g_i g' & \dots & g_n g' \\ (g_1g')g'' & (g_2g')g'' & \dots & (g_i g')g'' & \dots & (g_n g')g'' \end{pmatrix} = \varphi(g'')$$

nên $\varphi_{g'g''} = \varphi(g') \cdot \varphi(g'')$ nên φ là đồng cấu (homomorphism).

Để chứng minh φ là song ánh, ta chứng minh φ là đơn ánh và toàn ánh.

Giả sử $\varphi(g) = \varphi(g')$. Theo định nghĩa hoán vị thì $g = g'$ nên φ là đơn ánh.

Giả sử ta có hoán vị

$$\sigma = \begin{pmatrix} g_1 & g_2 & \dots & g_n \\ g_1g' & g_2g' & \dots & g_n g' \end{pmatrix},$$

ta nhân với g'^{-1} thì tìm được hoán vị ngược của σ . Như vậy φ là toàn ánh.

Kết luận: φ là song ánh và là đồng cấu (isomorphism).

❶ Definition 2.7 (Automorphism, Tự đồng cấu)

Ánh xạ được gọi là **tự đồng cấu** (hay automorphism) nếu nó là song ánh từ nó lên chính nó. Ta ký hiệu tự đồng cấu nhóm G là $\text{Aut}(G)$.

Hạt nhân và ảnh

Xét một homomorphism f từ nhóm (G, \star) tới nhóm $(H, *)$.

❶ Definition 2.8 (Kernel, Hạt nhân)

Hạt nhân (hay kernel) của f là tập hợp các phần tử của G cho ảnh là e_H , kí hiệu là $\ker f$. Nói cách khác

$$\ker f = \{g \in G : f(g) = e_H\}.$$

Như vậy $\ker f$ là tập con của G .

❶ Remark 2.3

$K = \ker f$ là normal subgroup của G .

❶ Chứng minh

Để chứng minh, ta thấy rằng theo định nghĩa homomorphism, với $g_1, g_2 \in K$ thì $f(g_1) = f(g_2) = e_H$.

Ta có

$$f(g_1 \star g_2) = f(g_1) * f(g_2) = e_H * e_H = e_H.$$

Như vậy $g_1 \star g_2 \in K$ nên K là nhóm con của G .

Tiếp theo để chứng minh K là normal subgroup, ta chứng minh $gKg^{-1} = K$ với mọi $g \in G$.

Do $gKg^{-1} = \{g \star k \star g^{-1} : k \in K\}$, lấy f mỗi phần tử bên trong ta có

$$f(g \star k \star g^{-1}) = f(g) * f(k) * f(g^{-1}) = f(g) * e_H * f(g^{-1}) = f(g) * f(g^{-1}),$$

mà theo tính chất của homomorphism thì

$$f(g^{-1}) = f(g)^{-1} \Rightarrow f(g \star k \star g^{-1}) = f(g) * f(g)^{-1} = e_H,$$

suy ra $g \star k \star g^{-1} \in K$ với mọi $g \in G$, với mọi $k \in K$. Do đó $gKg^{-1} = K$ và ta có điều phải chứng minh.

❶ Definition 2.9 (Image, Ảnh)

Ảnh (hay **image**) của f là tập hợp tất cả giá trị nhận được khi biến các phần tử thuộc G thành phần tử thuộc H . Nói cách khác

$$\text{im } f = \{f(g) : g \in G\}.$$

Như vậy $\text{im } f$ là tập con của H .

Dựa trên hai khái niệm này, chúng ta có một định lý quan trọng trong lý thuyết nhóm là **Định lí thứ nhất về sự đẳng cấu** (First isomorphism theorem).

❶ Theorem 2.3 (Định lí thứ nhất về sự đẳng cấu)

Với hai nhóm (G, \star) và $(H, *)$. Xét homomorphism $f : G \rightarrow H$. Khi đó $\text{im } f$ đẳng cấu (isomorphism) với nhóm thương $G / \ker f$.

❶ Chứng minh

Gọi G, H là hai nhóm và homomorphism $f : G \rightarrow H$.

Đặt $K = \ker f$. Ta xét biến đổi

$$\theta : \text{im } f \rightarrow G/K, f(g) \rightarrow gK$$

với $g \in G$.

Ta cần chứng minh biến đổi này là ánh xạ xác định (well-defined, nghĩa là tuân theo quy tắc ánh xạ, mỗi phần tử tập nguồn biến thành **một và chỉ một** phần tử tập đích), là homomorphism, là đơn ánh và là toàn ánh.

Đầu tiên ta chứng minh ánh xạ xác định. Giả sử ta có $g_1K = g_2K$, do g_1 và g_2 thuộc cùng coset nên $g_1^{-1}g_2 \in K$, hay $f(g_1^{-1}g_2) = e_H$.

Với f là homomorphism, ta có

$$f(g_1^{-1}g_2) = f(g_1^{-1})f(g_2) = f(g_1)^{-1}f(g_2) = e_H.$$

Suy ra $f(g_1) = f(g_2)$. Như vậy nếu $f(g_1) = f(g_2)$ thì $\theta(f(g_1)) = \theta(f(g_2))$.

Tiếp theo ta chứng minh θ là homomorphism. Do K là normal subgroup của G nên với mọi g_1, g_2 thuộc G thì $g_1g_2K = (g_1K)(g_2K)$.

Do $f(g_1g_2) = f(g_1)f(g_2)$ nên

$$\theta(f(g_1g_2)) = g_1g_2K = (g_1K)(g_2K) = \theta(f(g_1))\theta(f(g_2)).$$

Suy ra θ là homomorphism.

Để thấy với mọi $g \in G$ ta đều tìm được $f(g)$ và gK tương ứng. Do đó θ là toàn ánh.

Để chứng minh θ là đơn ánh, giả sử $g_1K = g_2K$ ta có $g_1^{-1}g_2 \in K$ nên $f(g_1^{-1}g_2) = e_H$, suy ra

$$f(g_1^{-1})f(g_2) = e_H \Rightarrow f(g_1)^{-1}f(g_2) = e_H \Rightarrow f(g_1) = f(g_2).$$

Như vậy θ là đơn ánh.

Kết luận, θ là song ánh. Định lí thứ nhất về sự đẳng cấu được chứng minh.

❶ Example 2.4 (Bài tập sưu tầm từ LAPLAS)

Chứng minh rằng $\mathrm{GL}_n(\mathbb{C})/H \cong \mathbb{R}^+$, với H là nhóm con các ma trận có định thức bằng 1.

❶ Giải

Để ý rằng H là nhóm con chuẩn tắc của $\mathrm{GL}_n(\mathbb{C})$. Xét ánh xạ:

$$f : \mathrm{GL}_n(\mathbb{C}) \rightarrow \mathbb{R}^+, f(\mathbf{A}) = |\det(\mathbf{A})|.$$

Vì $\det(\mathbf{A} \cdot \mathbf{B}) = \det(\mathbf{A}) \cdot \det(\mathbf{B})$ nên f là đồng cấu nhóm. Khi đó với mọi số thực dương r , tồn tại ma trận $\mathbf{A} \in \mathrm{GL}_n(\mathbb{C})$ sao cho $f(\mathbf{A}) = |\det(\mathbf{A})| = r$, ví dụ như

$$\begin{pmatrix} r & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}.$$

Như vậy f cũng là toàn cầu.

Ở đây $\ker f = H$ nên theo định lí thứ nhất về sự đẳng cấu, ta có

$$\mathrm{GL}_n(\mathbb{C})/H \cong \mathbb{R}^+.$$

2.2.3 Tác động nhóm

Tác động nhóm

Tác động nhóm (Group Action) cho phép chúng ta đếm những cấu hình tổ hợp mà việc vét cạn rồi loại bỏ sẽ tốn nhiều công sức cũng như sai sót.

Cho tập hợp M và nhóm G . Ta nói G **tác động trái** lên M với ánh xạ:

$$\alpha : G \times M \rightarrow M$$

thỏa mãn hai tiên đề sau:

- identity: $\alpha(e, m) = m$ với mọi $m \in M$ và e là phần tử đơn vị của G ;
- compatibility: $\alpha(g, \alpha(h, m)) = \alpha(gh, m)$.

Ta thường kí hiệu $\alpha(g, m)$ bởi $g(m)$ hay thậm chí đơn giản hơn là gm . Kí hiệu gm sẽ được sử dụng từ đây về sau.

Khi đó hai tiên đề trên tương đương với:

- identity: $em = m$ với mọi $m \in M$;
- compatibility: $g(hm) = (gh)m$ với mọi $m \in M$ và $g, h \in G$.

❶ Definition 3.1 (Nhóm con ổn định)

Với phần tử $m \in M$ cho trước, tập hợp các phần tử $g \in G$ mà $gm = m$ được gọi là **nhóm con ổn định** (hay **stabilizer**) của nhóm G . Ta kí hiệu

$$G_m = \{g \in G : gm = m\}.$$

❶ Definition 3.2 (Quỹ đạo)

Quỹ đạo (hay **orbit**) của phần tử $m \in M$ là tập hợp

$$G(m) = \{gm : g \in G\}.$$

❶ Remark 3.2

Hai orbit của hai phần tử bất kì rời nhau, hoặc trùng nhau.

❶ Chứng minh

Giả sử ta có $m_1, m_2 \in M$ mà $G(m_1) \cap G(m_2) \neq \emptyset$.

Khi đó tồn tại $g_1, g_2 \in G$ để $g_1m_1 = g_2m_2$, suy ra $m_1 = g_1^{-1}g_2m_2$.

Thêm nữa, mọi phần tử trong $G(m_1)$ có dạng gm_1 nên $gm_1 = gg_1^{-1}g_2m_2$, suy ra $G(m_1) \subseteq G(m_2)$.

Chứng minh tương tự ta cũng có $G(m_2) \subseteq G(m_1)$ nên $G(m_1) \equiv G(m_2)$.

❶ Remark 3.3

Tập hợp M là giao của các orbit rời nhau. Giả sử ta có t orbit rời nhau $G(m_1), G(m_2), \dots, G(m_t)$ thì

$$M = G(m_1) \cup G(m_2) \cup \dots \cup G(m_t).$$

❶ Example 3.5

Cho nhóm S_3 có 6 phần tử $(1)(2)(3)$, $(1)(2, 3)$, $(2)(1, 3)$, $(3)(1, 2)$, $(1, 2, 3)$, $(1, 3, 2)$.

Xét tập hợp $M = \{1, 2, 3\}$. Khi đó, xét từng hoán vị trên, ta có:

$$G_1 = \{(1)(2)(3), (1)(2, 3)\}$$

và

$$G(1) = \{1, 2, 3\}.$$

Ta nhận thấy $G(1) = G(2) = G(3)$, và $|G| = 6 = |G_1| \cdot |G(1)|$.

Hay nói cách khác, $|G(m)| = [G : G_m]$ với G_m là stabilizer của phần tử m và $[G : G_m]$ là subgroup index của $G_m \subset G$, và bằng $\frac{|G|}{|G_m|}$ nếu là nhóm hữu hạn.

❶ Definition 3.3

Hai phần tử $m, n \in M$ được gọi là có quan hệ với nhau dưới tác động của nhóm G nếu tồn tại phần tử $g \in G$ sao cho $m = gn$.

Ta kí hiệu là $m\tilde{G}n$.

❶ Remark 3.4

Quan hệ được định nghĩa như trên là quan hệ tương đương.

❶ Chứng minh

Để chứng minh một quan hệ là tương đương, ta cần chứng minh tính phản xạ, đối xứng và bắc cầu.

Đối với tính phản xạ, mọi vector đều có quan hệ với chính nó qua phần tử đơn vị $e \in G$.

Đối với tính đối xứng, nếu m có quan hệ với n thì tồn tại $g \in G$ sao cho $m = gn$. Theo tính chất nhóm thì tồn tại phần tử g^{-1} là nghịch đảo của g trong G . Do đó $g^{-1}m = n$. Nói cách khác n cũng có quan hệ với m . Như vậy ta có tính đối xứng.

Đối với tính bắc cầu, nếu m có quan hệ với n thì tồn tại $g_1 \in G$ sao cho $m = g_1n$. Tiếp theo, nếu n có quan hệ với p thì tồn tại $g_2 \in G$ sao cho $n = g_2p$, suy ra

$$m = g_1n = g_1(g_2p) = (g_1g_2)p.$$

Do $g_1, g_2 \in G$ nên $g_1g_2 \in G$. Như vậy m cũng có quan hệ với p nên quan hệ có tính bắc cầu.

Vậy quan hệ được định nghĩa như trên là quan hệ tương đương.

Bổ đề Burnside

Các trạng thái khác nhau của tập hợp M là **tương đương nhau** nếu chúng nằm trong cùng lớp tương đương dưới tác động của nhóm G .

Các ví dụ về bổ đề Burnside và định lý Polya được tham khảo tại [6].

Lemma 3.1 (Bổ đề Burnside)

Với nhóm G tác động lên tập hợp M , ta có:

$$t_G = \frac{1}{|G|} \sum_{g \in G} |M^g|,$$

trong đó:

- t_G là số lớp tương đương của tập M dưới tác động của nhóm G ;
- $|M^g|$ là số điểm bất động của tập M dưới tác động của phần tử g , nghĩa là $M^g = \{m \in M : gm = m\}$.

Chứng minh

Xét tập hợp

$$S = \{(g, m) \in G \times M : gm = m\}.$$

Ta đếm số phần tử của S theo hai cách.

Cách 1: đếm theo từng phần tử \$g in G\$.

Với mỗi phần tử g , số phần tử $m \in M$ cố định dưới tác động của g là $|M^g|$. Khi đó lấy tổng các phần tử g lại ta được

$$|S| = \sum_{g \in G} |M^g|.$$

Cách 2: đếm theo từng phần tử \$m in M\$.

Với mỗi m , stabilizer

$$G_m = \{g \in G : gm = m\}$$

có số phần tử là $|G_m|$. Khi đó

$$|S| = \sum_{m \in M} |G_m|.$$

Ở trên mình đã nói về một công thức là

$$|G| = |G(m)| \cdot |G_m|,$$

tương đương với $|G_m| = \frac{|G|}{|G(m)|}$.

Giả sử M có n orbit là $G(m_1), G(m_2), \dots, G(m_n)$. Với mỗi orbit $G(m_i)$ thì mọi $m \in G(m_i)$ có cùng kích thước $|G_m|$. Khi đó

$$\sum_{m \in G(m_i)} |G_m| = \sum_{m \in G(m_i)} \frac{|G|}{|G(m_i)|} = |G|.$$

Cộng tất cả quỹ đạo lại ta có

$$\sum_{m \in M} |G_m| = \underbrace{|G| + \cdots + |G|}_{n \text{ lầ}} = n \cdot |G|.$$

Kết hợp hai cách đếm suy ra

$$\sum_{g \in G} |M^g| = \sum_{m \in M} |G_m| = n \cdot |G|,$$

tương đương với

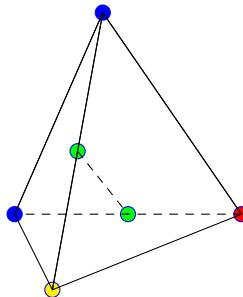
$$n = \frac{1}{|G|} \sum_{g \in G} |M^g|.$$

Ở đây $n = t_G$ và ta có điểm phải chứng minh.

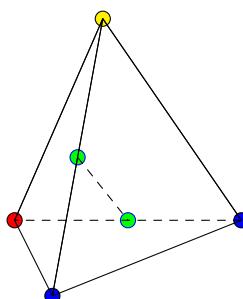
Bài toán tô màu bốn đỉnh tứ diện

Cho hình tứ diện đều. Ta tô bốn đỉnh của nó bằng ba màu xanh, đỏ, vàng. Hỏi có bao nhiêu cách tô như vậy?

Ta cần lưu ý một điều, hai cách tô là tương đương nhau (giống nhau) nếu tồn tại một phép quay các đỉnh biến cách tô này thành cách tô kia (ví dụ như [hình 2.14](#) và [hình 2.15](#)).



Hình 2.14: Cách tô 1



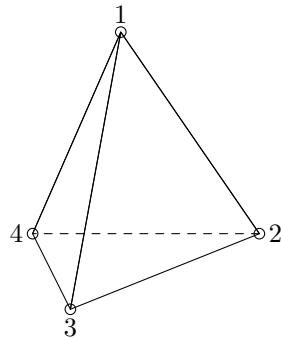
Hình 2.15: Cách tô 2

Như hình trên ta thấy nếu chọn trục quay là đường thẳng nối trung điểm hai cạnh đối diện (hai điểm xanh

lá) thì đỉnh trên và đỉnh dưới đổi chỗ cho nhau (xanh và vàng), đỉnh trái và đỉnh phải đổi chỗ cho nhau (xanh và đỏ).

Ta giải bài này như sau:

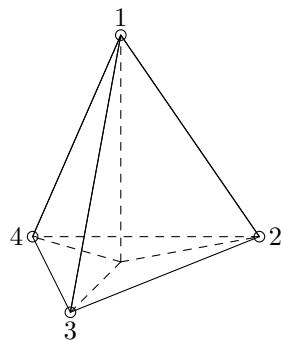
Đầu tiên ta đánh số các đỉnh của tứ diện như [hình 2.16](#).



Hình 2.16: Đánh số các đỉnh tứ diện

Ta có ba trường hợp biến đổi sau:

Trường hợp 1. Giữ nguyên một đỉnh và trực quay là đường thẳng đi qua đỉnh đó và tâm của mặt đối diện.

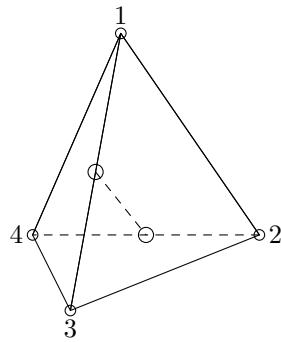


Hình 2.17: Trường hợp 1

Khi đó phép quay (ngược chiều đồng hồ) tương ứng hoán vị $(1)(2, 3, 4)$ (quay 60 độ) và $(1)(2, 4, 3)$ (quay 120 độ).

Do ta chọn một đỉnh cố định thì ta có 4 cách chọn, và với mỗi cách chọn đỉnh cố định ta có thể quay hai cách nên ta có tổng là 8 hoán vị.

Trường hợp 2. Ta chọn trung điểm hai cạnh đối nhau và nối lại thành trực quay như hình trong ví dụ. Khi đó tương ứng với hoán vị $(1, 4)(2, 3)$.



Hình 2.18: Trường hợp 2

Ta có $\frac{C_4^2}{2!} = 3$ hoán vị.

Trường hợp 3. Hoán vị đồng nhất $(1)(2)(3)(4)$.

Tóm lại, tập hợp M ở đây là tập hợp 4 đỉnh của tứ diện, và nhóm tác động lên M là nhóm con 12 phần tử của S_4 .

Như vậy, ví dụ với hoán vị $(1)(2, 3, 4)$, nếu ta muốn sau phép quay giữ nguyên trạng thái (hay nói cách khác là tìm M^g) thì ta tô màu đỉnh 1 tùy ý, các đỉnh 2 – 3 – 4 chung màu (cũng tùy ý).

Suy ra ta có $3 \cdot 3$ cách tô. Tương tự với các hoán vị dạng $(1, 4)(2, 3)$.

Như vậy có tất cả

$$t_G = \frac{1}{12}(1 \cdot 3^4 + 8 \cdot 3^2 + 3 \cdot 3^2) = 15$$

cách tô màu khác nhau.

Tổng quát, nếu có k màu thì số lớp tương đương là

$$t_G = \frac{1}{12} (1 \cdot k^4 + 8 \cdot k^2 + 3 \cdot k^2) = \frac{1}{12}(k^4 + 11k^2).$$

Tác động nhóm lên vector

Xét nhóm G và không gian vector \mathbb{F}_2^n , $n \in \mathbb{N}$. Khi đó hai vector \mathbf{x} và \mathbf{y} thuộc \mathbb{F}_2^n được gọi là **quan hệ với nhau** nếu tồn tại $g \in G$ mà $\mathbf{x} = g\mathbf{y}$.

Ví dụ, xét nhóm hoán vị S_3 . Giả sử các vector trong \mathbb{F}_2^3 có dạng

$$\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{F}_2^3.$$

Khi đó vector $(1, 0, 0)$ có quan hệ với $(0, 0, 1)$ với hoán vị $(1, 3)(2)$. Cụ thể là $(x_1, x_2, x_3) \xrightarrow{(1,3)(2)} (x_3, x_2, x_1)$.

Tương tự, vector $(1, 0, 0)$ cũng có quan hệ với $(0, 1, 0)$ với hoán vị $(1, 2)(3)$.

Thêm nữa, vector $(1, 0, 0)$ có quan hệ với chính nó qua hoán vị đồng nhất $(1)(2)(3)$.

Câu hỏi đặt ra là, có bao nhiêu lớp tương đương dưới tác động của nhóm S_3 ?

Để giải quyết vấn đề này ta sử dụng bồ đề Burnside.

Nhóm S_3 có các hoán vị

$$S_3 = \{(1)(2)(3), (1, 2)(3), (1, 3)(2), (2, 3)(1), (1, 3, 2), (1, 2, 3)\}.$$

Lần lượt xét từng hoán vị. Đầu tiên, với (1)(2)(3) thì các phần tử trong vector đúng yên. Do đó dưới tác động của hoán vị này, x_1 biến thành x_1 , x_2 biến thành x_2 và x_3 biến thành x_3 . Số cách chọn cho mỗi x_i là 2 nên theo quy tắc nhân ta có $2^3 = 8$ cách.

Tiếp theo, với hoán vị (1, 2)(3) thì $x_1 \rightarrow x_2$, $x_2 \rightarrow x_1$ và $x_3 \rightarrow x_3$. Do đó x_1 và x_2 có cùng giá trị (nằm cùng chu trình), thành ra có 2 cách chọn, x_3 cũng có 2 cách chọn nên tổng số cách là $2 \cdot 2 = 4$. Hoán vị (1, 3)(2) và (2, 3)(1) tương tự.

Với hoán vị (1, 2, 3) thì $x_1 \rightarrow x_2$, $x_2 \rightarrow x_3$ và $x_3 \rightarrow x_1$ nên $x_1 = x_2 = x_3$, có 2 cách chọn trong trường hợp này. Hoán vị (1, 3, 2) tương tự.

Như vậy, theo bổ đề Burnside, số lớp tương đương các vector trong \mathbb{F}_2^3 là

$$t(\mathcal{S}_3) = \frac{1}{6}(1 \cdot 2^3 + 3 \cdot 2^2 + 2 \cdot 2) = 4.$$

Thật vậy, ta có thể chia các vector thành 4 lớp tương đương là

$$\{000\}, \{001, 010, 011\}, \{011, 101, 110\}, \{111\}.$$

Ngoài nhóm \mathcal{S}_3 ra còn các nhóm khác cũng tác động lên các vector. Một số nhóm hay được sử dụng là:

1. Nhóm general linear: gồm các ma trận khả nghịch $n \times n$ trên \mathbb{F}_2 . Tác động nhóm lúc này là phép nhân ma trận $A \in \text{GL}(n, 2)$ với vector $x \in \mathbb{F}_2^n$, hay $A \cdot x$.
2. Nhóm general affine: gồm các ma trận khả nghịch $n \times n$ trên \mathbb{F}_2 và vector bất kì trong \mathbb{F}_2^n . Tác động nhóm lúc này là biến đổi affine $A \cdot x + b$ với $A \in \text{GL}(n, 2)$ và $b \in \mathbb{F}_2^n$.

Remark 3.5

Số lượng phần tử của nhóm $\text{GL}(n, 2)$ là

$$(2^n - 1) \cdot (2^n - 2) \cdots (2^n - 2^{n-1}).$$

Ví dụ, khi $n = 3$ thì

$$|\text{GL}(3, 2)| = (2^3 - 1) \cdot (2^3 - 2) \cdot (2^3 - 4) = 168$$

ma trận.

Tác động nhóm lên hàm boolean

Ta tiếp tục xét nhóm G và không gian vector \mathbb{F}_2^n , $n \in \mathbb{N}$. Khi đó hai hàm boolean n biến $f(x_1, \dots, x_n)$ và $g(x_1, \dots, x_n)$ được gọi là **quan hệ với nhau** nếu tồn tại $\tilde{g} \in G$ mà $g(x) = f(\tilde{g}x)$ với mọi $x \in \mathbb{F}_2^n$.

Ta cũng xét hoán vị \mathcal{S}_3 làm ví dụ. Ta cũng lần lượt xét các phần tử của nhóm.

Đặt f_0, f_1, \dots, f_7 lần lượt là các giá trị hàm f với các vector $x \in \mathbb{F}_2^3$.

Đầu tiên, với (1)(2)(3), ta có bảng chuyển vector như [hình 2.19](#).

x_1	x_2	x_3	f
0	0	0	f_0
0	0	1	f_1
0	1	0	f_2
0	1	1	f_3
1	0	0	f_4
1	0	1	f_5
1	1	0	f_6
1	1	1	f_7

x_1	x_2	x_3	f
0	0	0	f_0
0	0	1	f_1
0	1	0	f_2
0	1	1	f_3
1	0	0	f_4
1	0	1	f_5
1	1	0	f_6
1	1	1	f_7

Hình 2.19: Hoán vị (1)(2)(3)

Ta thấy rằng $f_0 \rightarrow f_0$, $f_1 \rightarrow f_1$, ..., $f_7 \rightarrow f_7$ nên có 8 chu trình. Vậy số lượng cách chọn là 2^8 .

Tiếp theo, xét các hoán vị dạng (1)(2,3), ta có bảng chuyển vector như [hình 2.20](#).

x_1	x_2	x_3	f
0	0	0	f_0
0	0	1	f_1
0	1	0	f_2
0	1	1	f_3
1	0	0	f_4
1	0	1	f_5
1	1	0	f_6
1	1	1	f_7

x_1	x_3	x_2	f
0	0	0	f_0
0	1	0	f_2
0	0	1	f_1
0	1	1	f_3
1	0	0	f_4
1	1	0	f_6
1	0	1	f_5
1	1	1	f_7

Hình 2.20: Hoán vị (1)(2,3)

Ta thấy rằng $f_0 \rightarrow f_0$, $f_1 \rightarrow f_2 \rightarrow f_1$, $f_3 \rightarrow f_3$, $f_4 \rightarrow f_4$, $f_5 \rightarrow f_6 \rightarrow f_5$, $f_7 \rightarrow f_7$. Ở đây có 6 chu trình nên số cách chọn là 2^6 .

Tiếp theo ta xét các hoán vị dạng (1, 2, 3) ([Hình 2.21](#)).

x_1	x_2	x_3	f	$(1, 2, 3)$	x_2	x_3	x_1	f
0	0	0	f_0		0	0	0	f_0
0	0	1	f_1		0	1	0	f_2
0	1	0	f_2		1	0	0	f_4
0	1	1	f_3		1	1	0	f_6
1	0	0	f_4		0	0	1	f_1
1	0	1	f_5		0	1	1	f_3
1	1	0	f_6		1	0	1	f_5
1	1	1	f_7		1	1	1	f_7

Hình 2.21: Hoán vị $(1, 2, 3)$

Ta thấy rằng $f_0 \rightarrow f_0$, $f_1 \rightarrow f_2 \rightarrow f_4 \rightarrow f_1$, $f_3 \rightarrow f_6 \rightarrow f_5 \rightarrow f_3$, $f_7 \rightarrow f_7$ nên ở đây có 4 chu trình. Số cách chọn là 2^4 .

Như vậy theo bô đề Burnside, số lớp hàm bool tương đương dưới tác động của nhóm S_3 là

$$t(S_3) = \frac{1}{6}(2^8 + 3 \cdot 2^6 + 2 \cdot 2^4) = 80.$$

Định lý Polya

Với mỗi hoán vị trong tập G , ta viết dưới dạng các chu trình độc lập

$$\underbrace{(g_1)(g_2) \dots (g_{t_1})}_{t_1} \underbrace{(g_{j_1}g_{j_2})(g_{j_3}g_{j_4}) \dots}_{t_2} \dots$$

Nếu ta viết hoán vị dưới dạng các chu trình rời nhau, ta gọi

Kí hiệu	Ý nghĩa
t_1	số chu trình có độ dài 1
t_2	số chu trình có độ dài 2
...	tương tự
t_n	số chu trình có độ dài n

Khi đó, **cycle index** (hay **chỉ số chu trình**) của hoán vị ứng các biến z_1, z_2, \dots, z_n là

$$I_g(z_1, z_2, \dots, z_n) = z_1^{t_1} z_2^{t_2} \cdots z_n^{t_n}.$$

Example 3.6

Xét hoán vị $(1, 2, 3)(4)(5)(6, 7) \in S_7$.

Ta có hai chu trình độ dài 1, một chu trình độ dài 2 và một chu trình độ dài 3 và không có chu trình độ dài 4, 5, 6, 7.

Do đó chỉ số chu trình là

$$I_g(z_1, z_2, z_3) = z_1^2 z_2^1 z_3^1.$$

Remark 3.6

Bất kì hoán vị nào thuộc S_n đều thỏa

$$1 \cdot t_1 + 2 \cdot t_2 + \dots + n \cdot t_n = n.$$

Definition 3.4 (Cyclic index, chỉ số chu trình)

Chỉ số chu trình của nhóm G là:

$$P_G(z_1, z_2, \dots, z_n) = \frac{1}{G} \sum_{g \in G} I_g(z_1, z_2, \dots, z_n).$$

Nhìn lại ví dụ về tứ diện bên trên, các đỉnh nằm trong cùng chu trình cần được tô cùng màu. Từ đó ta có chỉ số chu trình

$$P_G(z_1, z_2, z_3) = \frac{1}{12} (z_1^4 + 8z_1z_3 + 3z_2^2).$$

Cho mỗi $z_i = 3$ ta có kết quả phép tính theo bổ đề Burnside.

Định lý Polya là một mở rộng cho bổ đề Burnside, cho phép chúng ta đếm số lớp tương đương thỏa mãn điều kiện nhất định (về số lượng phần tử).

Ví dụ với hình tứ diện như trên nhưng ta thêm điều kiện tô hai đỉnh màu vàng, một đỉnh màu đỏ và một đỉnh màu xanh.

Ta kí hiệu tập R là tập hợp các trạng thái có thể nhận của mỗi phần tử $m \in M$.

$$R = \{r_1, r_2, \dots, r_c\}.$$

Ở ví dụ trên thì $R = \{\text{đỏ, xanh, vàng}\}$.

Ta thay mỗi z_i trong chỉ số chu trình bằng tổng $\sum_{r \in R} r^i$.

Example 3.7

Giả sử ta tô màu bốn đỉnh tứ diện với hai màu $R = \{r_1, r_2\}$.

Với z_1 ta thay bằng $r_1 + r_2$.

Với z_2 ta thay bằng $r_1^2 + r_2^2$.

Với z_3 ta thay bằng $r_1^3 + r_2^3$.

Khi đó P_G tương đương với

$$\frac{1}{12} [(r_1 + r_2)^4 + 8 \cdot (r_1 + r_2)(r_1^3 + r_2^3) + 3 \cdot (r_1^2 + r_2^2)^2].$$

Ta khai triển P_G (lưu ý là ở đây không có tính giao hoán phép nhân).

$$\begin{aligned}(r_1 + r_2)^4 &= r_1 r_1 r_1 r_1 + r_1 r_1 r_1 r_2 + r_1 r_1 r_2 r_1 + r_1 r_1 r_2 r_2 \\&\quad + r_1 r_2 r_1 r_1 + r_1 r_2 r_1 r_2 + r_1 r_2 r_2 r_1 + r_1 r_2 r_2 r_2 \\&\quad + r_2 r_1 r_1 r_1 + r_2 r_1 r_1 r_2 + r_2 r_1 r_2 r_1 + r_2 r_1 r_2 r_2 \\&\quad + r_2 r_2 r_1 r_1 + r_2 r_2 r_1 r_2 + r_2 r_2 r_2 r_1 + r_2 r_2 r_2 r_2.\end{aligned}$$

Mình thấy rằng có 16 cấu hình khác nhau tương ứng 16 cách tô hai màu cho bốn đỉnh. Tương tự

$$\begin{aligned}(r_1 + r_2)(r_1^3 + r_2^3) &= r_1^4 + r_1 r_2^3 + r_2 r_1^3 + r_2^4 \\&= r_1 r_1 r_1 r_1 + r_1 r_2 r_2 r_2 + r_2 r_1 r_1 r_1 + r_2 r_2 r_2 r_2.\end{aligned}$$

Cuối cùng là

$$\begin{aligned}(r_1^2 + r_2^2)^2 &= r_1^4 + r_1^2 r_2^2 + r_2^2 r_1^2 + r_2^4 \\&= r_1 r_1 r_1 r_1 + r_1 r_1 r_2 r_2 + r_2 r_2 r_1 r_1 + r_2 r_2 r_2 r_2.\end{aligned}$$

Việc không có tính giao hoán với phép nhân làm biến thức công kẽm và phức tạp.

Do đó mình thêm một tập hợp W là vành giao hoán, và xét ánh xạ $w : R \mapsto W$ với: $w(r_i) = w_i$.

Khi đó nếu thay r_i bởi w_i vào bên trên biến thức sẽ rất đẹp:

$$P_G(w_1, w_2) = \frac{1}{12} [(w_1 + w_2)^4 + 8(w_1 + w_2)(w_1^3 + w_2^3) + 3(w_1^2 + w_2^2)^2].$$

Khai triển và thu gọn ta có

$$\begin{aligned}P_G(w_1, w_2) &= \frac{1}{12} [12w_1^4 + 12w_1^3 w_2 + 12w_1^2 w_2^2 + 12w_1 w_2^3 + 12w_2^4] \\&= w_1^4 + w_1^3 w_2 + w_1^2 w_2^2 + w_1 w_2^3 + w_2^4.\end{aligned}$$

Ở đây, định lý Polya nói rằng, số mũ của w_i thể hiện số lượng phần tử của tập M nhận giá trị r_i , và hệ số trước mỗi toán hạng là số lớp tương đương tương ứng với số lượng phần tử của tập M nhận các giá trị r_i .

Nói cách khác:

- có 1 lớp tương đương mà 4 đỉnh nhận màu r_1 ;
- có 1 lớp tương đương mà 3 đỉnh nhận màu r_1 và 1 đỉnh nhận màu r_2 ;
- có 1 lớp tương đương mà 2 đỉnh nhận màu r_1 và 2 đỉnh nhận màu r_2 ;
- có 1 lớp tương đương mà 1 đỉnh nhận màu r_1 và 3 đỉnh nhận màu r_2 ;
- có 1 lớp tương đương mà 4 đỉnh nhận màu r_2 .

Quay lại vấn đề tô bốn đỉnh tứ diện với ba màu xanh, đỏ, vàng. Tìm số cách tô hai đỉnh màu vàng, một đỉnh màu đỏ và một đỉnh màu xanh.

Đặt

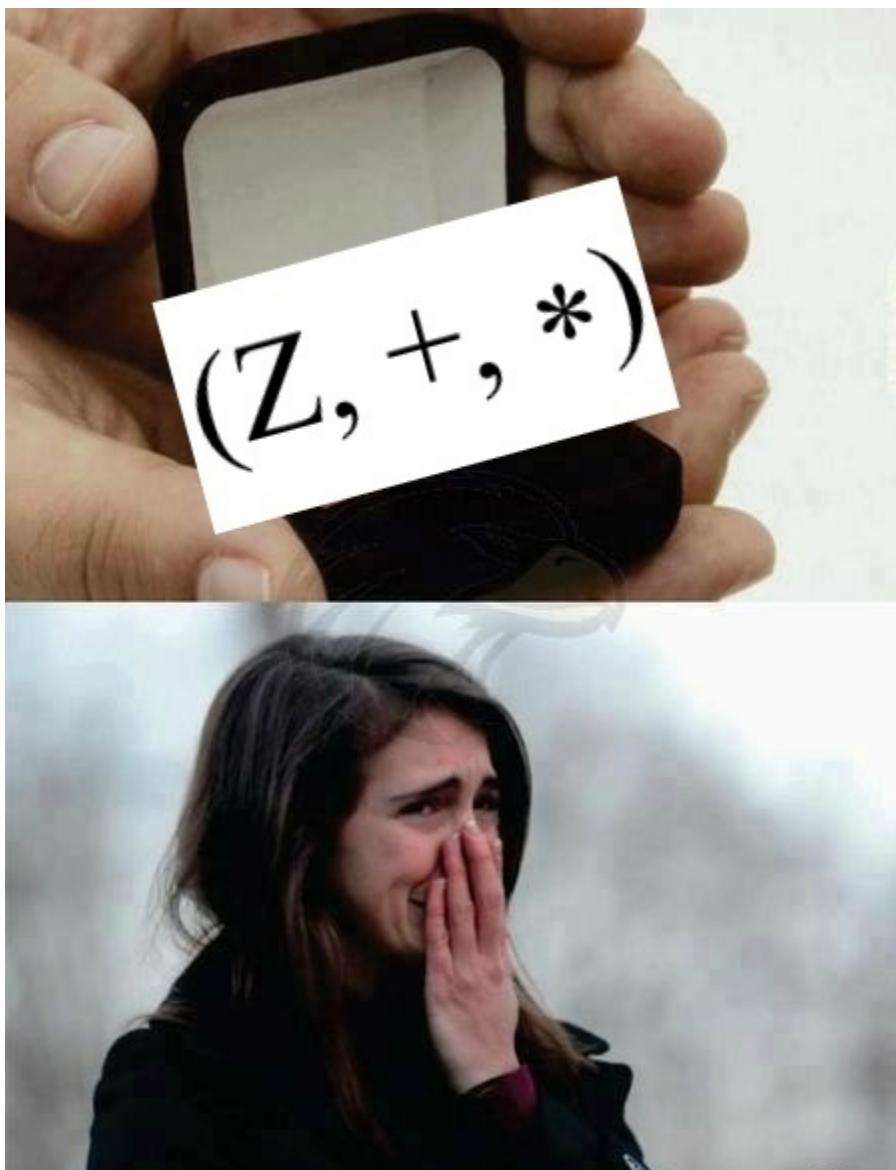
$$w(\text{vàng}) = x, w(\text{đỏ}) = y, w(\text{xanh}) = z.$$

Ta có:

$$P_G = \frac{1}{12} [(x + y + z)^4 + 8 \cdot (x + y + z)(x^3 + y^3 + z^3) + 3 \cdot (x^2 + y^2 + z^2)^2].$$

Như vậy đề bài tương ứng việc tìm hệ số của hạng tử x^2yz trong biểu thức trên. Kết quả là 1.

2.2.4 Vành



Hình 2.22: Nguồn: https://vk.com/wall-91031095_97017

Vành

i Definition 4.1 (Vành)

Cho tập hợp R , trên đó ta định nghĩa hai toán tử **cộng** (kí hiệu là $+$) và **nhân** (kí hiệu là \times).

Khi đó, $(R, +, \times)$ tạo thành **vành** (hay **ring**, **кольцо**) nếu

1. $(R, +)$ là nhóm Abel.

2. (R, \times) có tính kết hợp với phép nhân: với mọi $a, b, c \in R$ thì $a \times (b \times c) = (a \times b) \times c$.
3. Tính phân phối của phép cộng và phép nhân: với mọi $a, b, c \in R$ thì $(a + b) \times c = a \times c + b \times c$.

Tóm lại, $(R, +, \times)$ là vành nếu nó là nhóm Abel đối với phép cộng và có tính kết hợp với phép nhân.

Ta thường kí hiệu 0_R (hoặc ngắn gọn là 0) là phần tử đơn vị của phép cộng $(R, +)$ và gọi là **phần tử trung hòa**.

Khi đó phần tử nghịch đảo của phép cộng gọi là **phần tử đối** và được kí hiệu là $-a$, chỉ phần tử đối của phần tử a .

❶ Remark 4.1

Phép nhân ở đây không nhất thiết có phần tử đơn vị, hay phần tử nghịch đảo như trong định nghĩa nhóm. Trong trường hợp này (R, \times) gọi là **semigroup** (hay **nửa nhóm**).

❶ Property 4.1 (Tính chất của vành)

1. Với mọi $a \in R$ thì $a \times 0_R = 0_R \times a = 0_R$.
2. Với mọi $a, b \in R$ thì $(-a) \times b = -(a \times b)$.

❶ Chứng minh

Để chứng minh hai tính chất trên ta dùng định nghĩa vành.

1. Với mọi $a \in R$, ta có

$$a \times 0_R = a \times (0_R + 0_R) = a \times 0_R + a \times 0_R.$$

Rút gọn $a \times 0_R$ hai vế ta có $a \times 0_R = 0_R$. Tương tự cho $0_R \times a = 0_R$.

2. Vì $(-a) + a = 0_R$ với mọi $a \in R$, nhân b hai vế và dùng tính chất đầu suy ra

$$(-a) \times b + a \times b = 0_R \times b = 0_R.$$

Chuyển vế ta có $(-a) \times b = -(a \times b)$.

❶ Definition 4.2 (Vành với đơn vị)

Nếu có phần tử $1_R \neq 0_R \in R$ sao cho với mọi $r \in R$ ta đều có

$$1_R \times r = r \times 1_R = r$$

thì 1_R được gọi là **phần tử đơn vị** đối với phép nhân và R được gọi là **vành với đơn vị** (hay **ring with identity**, **кольцо с единицей**).

Ta kí hiệu 1_R (hoặc ngắn gọn là 1) là **phần tử đơn vị** đối với phép nhân (R, \times) .

Từ phần tử đơn vị đối với phép nhân ta có khái niệm **đặc số** (hay **số đặc trưng**, **characteristic**) của vành với đơn vị.

❶ Definition 4.3 (Characteristic)

Xét trường R với phần tử đơn vị là 1 và phần tử trung hòa là 0. Số dương p nhỏ nhất sao cho

$$\underbrace{1 + 1 + \dots + 1 + 1}_{p \text{ lần}} = 0$$

được gọi là **đặc số** (hay **characteristic**, **характеристика**) của R .

❶ Definition 4.4 (Vành giao hoán)

Nếu ta có tính giao hoán đối với phép nhân, nghĩa là với mọi $a, b \in R$ đều thỏa

$$a \times b = b \times a,$$

thì ta nói R là **vành giao hoán** (hay **commutative ring**, **коммутативное кольцо**).

2.2.5 Trường

Trường

❶ Definition 5.1 (Trường)

Cho tập hợp F và hai toán tử hai ngôi trên F là phép cộng $+$ và phép nhân \times . Khi đó $(F, +, \times)$ là **trường** (hay **field**, **поля**) nếu

1. $(F, +, \times)$ là vành giao hoán với đơn vị.
2. Với mọi phần tử $f \neq 0_F$, tồn tại nghịch đảo f^{-1} của f đối với phép nhân, nghĩa là

$$f \times f^{-1} = f^{-1} \times f = 1_F.$$

Nói cách khác, (F, \times) là nhóm Abel. Trên trường ta thực hiện được bốn phép tính cộng, trừ, nhân, chia.

❶ Example 5.1

Các tập hợp sau với phép cộng và nhân là trường.

1. Tập hợp số thực \mathbb{R} .
2. Tập hợp các số phức \mathbb{C} .
3. Tập hợp các số dạng $a + b\sqrt{2}$ với $a, b \in \mathbb{Q}$.

Những trường trên được gọi là **trường vô hạn** vì có vô số phần tử.

Ngược lại, chúng ta cũng có các **trường hữu hạn**.

Trường hữu hạn

Trường hữu hạn modulo nguyên tố

Cho p là số nguyên tố. Khi đó tập hợp các số dư khi chia cho p cùng với phép cộng và nhân modulo p tạo thành trường.

1 Chứng minh

Xét tập hợp các số dư khi chia cho p là

$$S = \{0, 1, \dots, p-2, p-1\}.$$

Ta thấy rằng với mọi $a, b \in S$ thì $a + b \pmod{p}$ và $a \cdot b \pmod{p}$ đều thuộc S .

1. Vì $0 + a = a + 0 = a \pmod{p}$ với mọi $a \in S$ nên 0 là phần tử đơn vị của phép cộng.
2. Với mọi $a \in S$, ta có $(p-a) + a = a + (p-a) \equiv 0 \pmod{p}$ nên phần tử nghịch đảo của a đối với phép cộng là $p-a \in S$.
3. Phép cộng modulo có tính kết hợp.
4. Phép cộng modulo có tính giao hoán.

Như vậy $(S, +)$ là nhóm Abel.

Tiếp theo, ta thấy rằng phép cộng và nhân có tính phân phối trên modulo.

Đồng thời phép nhân modulo cũng có tính kết hợp. Do đó $(S, +, \cdot)$ là vành.

1. Phần tử đơn vị của phép nhân là 1 .
2. Phép nhân modulo có tính giao hoán.
3. Do mọi phần tử thuộc S đều nguyên tố cùng nhau với p nên luôn tồn tại nghịch đảo của phần tử khác 0 trong S .

Kết luận: $(S, +, \cdot)$ là trường.

Ta thường kí hiệu trường này là $\text{GF}(p)$ (GF là viết tắt của Galois Field để tưởng nhớ người có đóng góp quan trọng trong lý thuyết nhóm).

Trường hữu hạn modulo đa thức

Xét các đa thức với hệ số nguyên

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0.$$

Ta thấy rằng phép cộng và nhân hai đa thức tạo thành một vành giao hoán với đơn vị là đa thức $f(x) \equiv 1$.

Thêm nữa vành này có vô số phần tử. Ta cần một phương án để số phần tử là hữu hạn, và đồng thời là trường.

Với p là số nguyên tố và n là số nguyên dương. Mình xét các đa thức có bậc tối đa là $n-1$ với hệ số nằm trong tập hợp các số dư khi chia cho p . Như vậy mình có p^n đa thức như vậy.

1 Example 5.2

Với $p = 3$ và $n = 2$. Khi đó các đa thức có thể có là

$$\{0, 1, 2, x, x + 1, x + 2, 2x, 2x + 1, 2x + 2\}.$$

Tương tự với việc modulo cho một số nguyên tố, ở đây mình xét phép cộng và nhân trên modulo một đa thức tối giản (irreducible polynomial) có bậc n (vì khi modulo một đa thức bậc bất kì cho đa thức bậc n ta có đa thức bậc nhỏ hơn n).

Đồng thời hệ số của đa thức từ phép cộng và nhân cũng được modulo p (nằm trong $\text{GF}(p)$).

Với trường hợp $p = 3$ và $n = 2$ ở trên mình có thể chọn đa thức modulo là $m(x) = x^2 + 2x + 2$.

Sau đây là bảng phép cộng hai đa thức bậc nhỏ hơn 2 trong modulo $m(x)$.

	0	1	2	x	$x + 1$	$x + 2$	$2x$	$2x + 1$	$2x + 2$
0	0	1	2	x	$x + 1$	$x + 2$	$2x$	$2x + 1$	$2x + 2$
1	1	2	0	$x + 1$	$x + 2$	x	$2x + 1$	$2x + 2$	$2x$
2	2	0	1	$x + 2$	x	$x + 1$	$2x + 2$	$2x$	$2x + 1$
x	x	$x + 1$	$x + 2$	$2x$	$2x + 1$	$2x + 2$	0	1	2
$x + 1$	$x + 1$	$x + 2$	x	$2x + 1$	$2x + 2$	$2x$	1	2	0
$x + 2$	$x + 2$	x	$x + 1$	$2x + 2$	$2x$	$2x + 1$	2	0	1
$2x$	$2x$	$2x + 1$	$2x + 2$	0	1	2	x	$x + 1$	$x + 2$
$2x + 1$	$2x + 1$	$2x + 2$	$2x$	1	2	0	$x + 1$	$x + 2$	x
$2x + 2$	$2x + 2$	$2x$	$2x + 1$	2	0	1	$x + 2$	x	$x + 1$

Tương tự, sau đây là bảng phép nhân hai đa thức bậc nhỏ hơn 2 trong modulo $m(x)$.

	0	1	2	x	$x + 1$	$x + 2$	$2x$	$2x + 1$	$2x + 2$
0	0	0	0	0	0	0	0	0	0
1	0	1	2	x	$x + 1$	$x + 2$	$2x$	$2x + 1$	$2x + 2$
2	0	2	1	$2x$	$2x + 2$	$2x + 1$	x	$x + 2$	$x + 1$
x	0	$2x$	$x + 1$	$2x$	$2x + 1$	1	$2x + 2$	2	$x + 2$
$x + 1$	0	$x + 1$	$2x + 2$	$2x + 1$	2	x	$x + 2$	$2x$	1
$x + 2$	0	$x + 2$	$2x + 1$	1	x	$2x + 2$	2	$x + 1$	$2x$
$2x$	0	$2x$	x	$2x + 2$	$x + 2$	2	$x + 1$	1	$2x + 1$
$2x + 1$	0	$2x + 1$	$x + 2$	2	$2x$	$x + 1$	1	$2x + 2$	x
$2x + 2$	0	$2x + 2$	$x + 1$	$x + 2$	1	$2x$	$2x + 1$	x	2

Ta thấy rằng bảng phép nhân đối xứng qua đường chéo chính. Điều này chứng tỏ phép nhân có tính giao hoán. Thêm nữa ở mỗi hàng hoặc cột khác 0 đều có 9 phần tử khác nhau.

Ideal

Definition 5.2 (Ideal)

Xét vần $(R, +, \times)$. Một tập con I của R được gọi là **ideal trái** (hay **left ideal**, **левый идеал**) nếu:

1. $(I, +)$ là nhóm con của $(R, +)$.
2. Với mọi $r \in R$, với mọi $x \in I$ thì $rx \in I$.

Ta định nghĩa tương tự với ideal phải, khi đó $xr \in I$. Từ đây về sau nếu không nói gì thêm nghĩa là mình xét ideal trái.

❶ Definition 5.3 (Principal Ideal)

Nếu $I = a$ với a là phần tử nào đó trong R thì I được gọi là **principal ideal** (hay **ideal chính**, **главный идеал**).

Nói cách khác, nếu có một phần tử trong R "sinh" ra được I thì I là principal.

❶ Definition 5.4 (Maximal Ideal)

Nếu I là một ideal của R và không tồn tại tập con I' mà $I \subset I' \subset R$ (tập con thực thụ) thì I được gọi là **maximal ideal** (hay **максимальный идеал**).

❶ Remark 5.1

Xét vành số nguyên \mathbb{Z} . Khi đó mọi ideal của \mathbb{Z} đều là principal.

❶ Chứng minh

Giả sử ideal I của \mathbb{Z} có phần tử dương nhỏ nhất là n .

Theo định nghĩa của ideal thì với mọi $q \in \mathbb{Z}$ ta có $qn \in I$.

Nếu phần tử $a \in I$, theo phép chia Euclid ta có $a = qn + r$ với $0 \leq r < n$, mà $a \in I$ và $qn \in I$ nên $r = a - qn \in I$.

Tuy nhiên phần tử dương nhỏ nhất thuộc I là n , do đó $r = 0$.

Nói cách khác mọi phần tử $a \in I$ đều có dạng qn với $q \in \mathbb{Z}$.

Vậy mọi ideal đều là principal.

❶ Remark 5.2

Ideal I của \mathbb{Z} là maximal khi và chỉ khi $I = n\mathbb{Z}$ với n là số nguyên tố.

❶ Chứng minh

Ta chứng minh chiều thuận, chiều ngược tương tự. Sử dụng phản chứng, ta giả sử n là hợp số. Khi đó $n = n_1 n_2$ với $n_1 \geq n_2 > 1$.

Khi đó $n\mathbb{Z} \subset n_1\mathbb{Z} \subset \mathbb{Z}$, suy ra ideal không phải maximal. Ta có điều phải chứng minh.

❶ Theorem 5.1

Gọi R là vành giao hoán với đơn vị. Khi đó, nếu I là ideal của R thì R/I là trường khi và chỉ khi I là maximal ideal.

❶ **Chứng minh**

Ta chứng minh điều kiện cần và điều kiện đủ.

Điều kiện cần. Ta có I là maximal ideal. Ta thấy rằng $a + I \neq 0$ khi và chỉ khi $a \notin I$, vì nếu $a \in I$ thì tồn tại $-a \in I$. Theo định nghĩa vành thì aR cũng là ideal nên $I + aR$ là ideal, mà $a \notin I$ và $a \in I + aR$ nên suy ra $I \subset I + aR$.

Ta lại có I là maximal nên $I + aR = R$, do đó tồn tại $n \in I$ và $b \in R$ sao cho $n + ab = 1$. Tóm lại là tồn tại nghịch đảo của phép nhân, do đó R/I là trường.

Điều kiện đủ. Với R/I là trường. Ta giả sử I không là maximal ideal. Khi đó tồn tại I' sao cho $I \subset I' \subset R$.

Khi đó tồn tại phần tử $a \in I'$ và $a \notin I$ mà $a + I \neq 0$. Do đó $(a + I)(b + I) = 1 + I$, suy ra tồn tại $n \in I \subset I'$ sao cho $ab = 1 + n$. Vì $a, b \in I'$ nên $1 \in I'$, từ đó $1 \in R$ nên I' không phải maximal.

❶ **Example 5.3**

Xét tập hợp \mathbb{Z} là vành giao hoán với đơn vị. Nếu n là số nguyên tố thì $n\mathbb{Z}$ là maximal ideal (đã chứng minh ở trên).

Khi đó tập $\mathbb{Z}/n\mathbb{Z}$ là trường hữu hạn modulo nguyên tố gồm các phần tử $\{0, 1, \dots, p-1\}$.

Ring kernel và ring homomorphism

Xét vành $(R, +, \times)$. Khi đó:

- với mọi $a \in R$, $a \times 0 = 0 \times a = 0$;
- $(-a) \times b = -(a \times b)$.

❶ **Definition 5.5 (Ring homomorphism)**

Xét hai vành là $(R_1, +, \times)$ và (R_2, \boxplus, \otimes) và ánh xạ $f : R_1 \rightarrow R_2$.

Ánh xạ f được gọi là **homomorphism** trên hai vành nếu f là homomorphism trên phép cộng và phép nhân, nghĩa là:

- với mọi $a, b \in R_1$, $f(a) \boxplus f(b) = f(a + b)$;
- với mọi $a, b \in R_1$, $f(a) \otimes f(b) = f(a \times b)$.

❶ Definition 5.6 (Hạt nhân)

Hạt nhân (hay **kernel**, **ядро**) của f là

$$\ker f = \{x : x \in R_1, f(x) = 0_2\}$$

với 0_2 là phần tử trung hòa của R_2 .

❷ Theorem 5.2

$\ker f$ là một ideal của R_1 .

❸ Chứng minh

Ta có $f(0_1) = 0_2$ theo định nghĩa homomorphism. Do đó với mọi $a \in R_1$ và với mọi $b \in \ker f$ thì

$$f(a) \otimes f(b) = f(a) \otimes 0_2 = 0_2 = f(a \times b).$$

Do đó $a \times b \in \ker f$, suy ra $\ker f$ là ideal trái của R_1 .

Tương tự cho với mọi $a \in R_1$ và với mọi $b \in \ker f$ thì

$$f(b) \otimes f(a) = 0_2 = f(b \times a),$$

suy ra $b \times a \in \ker f$ nên $\ker f$ cũng là ideal phải của R_1 .

Kết luận: $\ker f$ là ideal của R_1 .

2.2.6 Lý thuyết Galois

Tài liệu tham khảo trong phần này lấy từ sách [7] và khóa học Math 4120 [8] (bài giảng Visual Group Theory).

Extension Fields

Extension Fields

❶ Definition (Mở rộng trường)

Trường E được gọi là **mở rộng trường** (hay **extension field**) của trường F nếu F là trường con của E . Khi đó F được gọi là **trường cơ sở** (hay **base field**) và kí hiệu $F \subset E$.

❷ Example

Trường nào nhỏ nhất chứa \mathbb{Q} và $\sqrt{2}$?

Đáp án là trường

$$\mathbb{Q}(\sqrt{2}) = \{a + b\sqrt{2} : a, b \in \mathbb{Q}\}.$$

Việc chứng minh $\mathbb{Q}(\sqrt{2})$ là trường khá đơn giản, phần tử nghịch đảo đối với phép nhân của $a + b\sqrt{2}$ là

$$\frac{a}{a^2 - 2b^2} + \frac{-2b}{a^2 - 2b^2}\sqrt{2}.$$

❶ Example

Trường nào nhỏ nhất chứa \mathbb{Q} và i (ở đây i là đơn vị ảo, $i^2 = -1$)?

Đáp án là trường

$$\mathbb{Q}(i) = \{a + bi : a, b \in \mathbb{Q}\}.$$

Tương tự, ở đây phần tử nghịch đảo đối với phép nhân của $a + bi$ là

$$\frac{a}{a^2 + b^2} + \frac{-b}{a^2 + b^2}i.$$

Ở đây $\mathbb{Q}(\sqrt{2})$ và $\mathbb{Q}(i)$ đều là mở rộng của \mathbb{Q} và đều là tập con của \mathbb{C} . Tuy nhiên hai trường này không phải tập con của nhau.

Như vậy, bằng việc mở rộng \mathbb{Q} với $\sqrt{2}$ ta có trường $\mathbb{Q}(\sqrt{2})$.

Tương tự, bằng việc mở rộng \mathbb{Q} với i ta có trường $\mathbb{Q}(i)$.

Vậy trường nào chứa \mathbb{Q} , $\sqrt{2}$ và i ?

❶ Example

Trường chứa cả \mathbb{Q} , $\sqrt{2}$ và i là tập hợp

$$\mathbb{Q}(\sqrt{2}, i) = \{a + b\sqrt{2} + ci + d\sqrt{2}i : a, b, c, d \in \mathbb{Q}\}.$$

Trường trên có thể suy ra từ logic sau. Ta đã có $\mathbb{Q}(i)$ chứa \mathbb{Q} và i . Ta muốn thêm $\sqrt{2}$ vào trường $\mathbb{Q}(i)$ nên ta sẽ muốn mở rộng $\mathbb{Q}(i)$ lên $\mathbb{Q}(i)(\sqrt{2})$.

Khi đó $\mathbb{Q}(i)(\sqrt{2})$ tương tự sẽ có dạng

$$\mathbb{Q}(i)(\sqrt{2}) = \{\alpha + \beta\sqrt{2} : \alpha, \beta \in \mathbb{Q}(i)\}.$$

Nói cách khác $\alpha = a + bi$ và $\beta = c + di$, $a, b, c, d \in \mathbb{Q}$, nên ta có

$$\alpha + \beta\sqrt{2} = a + bi + c\sqrt{2} + d\sqrt{2}i, \quad a, b, c, d \in \mathbb{Q}.$$

Khi đó ta viết

$$\mathbb{Q}(i)(\sqrt{2}) = \mathbb{Q}(\sqrt{2}, i) = \{a + b\sqrt{2} + ci + d\sqrt{2}i : a, b, c, d \in \mathbb{Q}\}.$$

❷ Remark

- $\mathbb{Q}(\sqrt{2})$ là trường con của \mathbb{R} nhưng $\mathbb{Q}(i)$ không phải.

2. $\mathbb{Q}(i)$ nhỏ hơn \mathbb{C} rất nhiều (không chứa $\sqrt{2}$).
3. $\mathbb{Q}(\sqrt{2})$ chứa tất cả nghiệm của đa thức $f(x) = x^2 - 2$ trên \mathbb{Q} . Do đó $\mathbb{Q}(\sqrt{2})$ được gọi là **trường phân rã** của đa thức $f(x)$.

Biểu diễn quan hệ giữa các trường qua lattice

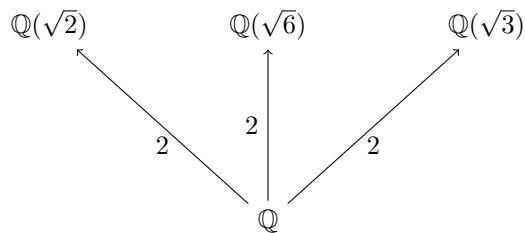
Ở phần trước, $\mathbb{Q}(\sqrt{2})$ là mở rộng của \mathbb{Q} và với hai phần tử $a, b \in \mathbb{Q}$ xác định một phần tử $a + b\sqrt{2} \in \mathbb{Q}(\sqrt{2})$. Do $a + b\sqrt{2} = a \cdot 1 + b \cdot \sqrt{2}$ nên $\{1, \sqrt{2}\}$ là **cơ sở** (hay **basis**) của $\mathbb{Q}(\sqrt{2})$. Cơ sở chứa hai phần tử nên ta nói **bậc của mở rộng đơn** (hay **degree**) từ \mathbb{Q} lên $\mathbb{Q}(\sqrt{2})$ là 2.

Tương tự, mở rộng từ \mathbb{Q} lên $\mathbb{Q}(i)$ cũng có bậc là 2 với cơ sở là $\{1, i\}$.

Ta kí hiệu bậc của mở rộng đơn là

$$[E : F].$$

Như vậy ta cũng các trường $\mathbb{Q}(\sqrt{2})$, $\mathbb{Q}(\sqrt{3})$ và $\mathbb{Q}(\sqrt{6})$ là các mở rộng đơn bậc 2 của \mathbb{Q} (Hình 2.23).



Hình 2.23: Mở rộng trường \mathbb{Q} lên $\mathbb{Q}(\sqrt{2})$, $\mathbb{Q}(\sqrt{3})$ và $\mathbb{Q}(\sqrt{6})$

Do $\sqrt{6} = \sqrt{2} \cdot \sqrt{3}$ dẫn ta tới câu hỏi, trường nào nhỏ nhất chứa cả $\sqrt{2}$ và $\sqrt{3}$?

Thực hiện tương tự với $\mathbb{Q}(\sqrt{2}, i)$ ở trên ta có trường

$$\mathbb{Q}(\sqrt{2}, \sqrt{3}) = \mathbb{Q}(\sqrt{2})(\sqrt{3}) = \{a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6} : a, b, c, d \in \mathbb{Q}\}$$

là trường nhỏ nhất chứa cả $\sqrt{2}$ và $\sqrt{3}$.

Ở đây, $\mathbb{Q}(\sqrt{2}, \sqrt{3})$ là mở rộng bậc 4 của \mathbb{Q} với cơ sở là $\{1, \sqrt{2}, \sqrt{3}, \sqrt{6}\}$.

Bằng việc mở rộng từ $\mathbb{Q}(\sqrt{2})$ lên $\mathbb{Q}(\sqrt{2}, \sqrt{3}) = \mathbb{Q}(\sqrt{2})(\sqrt{3})$ ta cũng có đây là mở rộng bậc 2 (tương tự chứng minh phía trên cho $\mathbb{Q}(\sqrt{2}, i)$). Như vậy mở rộng từ $\mathbb{Q}(\sqrt{3})$ lên $\mathbb{Q}(\sqrt{2}, \sqrt{3})$ cũng là mở rộng bậc 2.

Vậy mở rộng từ $\mathbb{Q}(\sqrt{6})$ lên $\mathbb{Q}(\sqrt{2}, \sqrt{3})$ là bậc mấy? Để xác định ta cần chứng minh nhận xét sau

❶ Remark

$\mathbb{Q}(\sqrt{2}, \sqrt{3}) \equiv \mathbb{Q}(\sqrt{2} + \sqrt{3})$, trong đó $\mathbb{Q}(\sqrt{2} + \sqrt{3})$ là trường nhỏ nhất chứa \mathbb{Q} và $\sqrt{2} + \sqrt{3}$.

❶ Chứng minh

Ta chứng minh $\mathbb{Q}(\sqrt{2}, \sqrt{3}) \subset \mathbb{Q}(\sqrt{2} + \sqrt{3})$.

Do $\sqrt{2} + \sqrt{3}$ nên nghịch đảo $\frac{1}{\sqrt{2} + \sqrt{3}} = \sqrt{3} - \sqrt{2} \in \mathbb{Q}(\sqrt{2} + \sqrt{3})$.

Khi đó, $\frac{1}{2} \cdot (\sqrt{2} + \sqrt{3}) \pm \frac{1}{2} \cdot (\sqrt{3} - \sqrt{2}) \in \mathbb{Q}(\sqrt{2} + \sqrt{3})$.

Vẽ trái sẽ bằng $\sqrt{2}$ hoặc $\sqrt{3}$. Như vậy $\sqrt{2}, \sqrt{3} \in \mathbb{Q}(\sqrt{2} + \sqrt{3})$.

Phép nhân trên trường cho ta $\sqrt{2} \cdot \sqrt{3} = \sqrt{6} \in \mathbb{Q}(\sqrt{2} + \sqrt{3})$.

Như vậy với mọi $a, b, c, d \in \mathbb{Q}$ ta có $a + b\sqrt{2} + c\sqrt{3} + d\sqrt{6} \in \mathbb{Q}(\sqrt{2} + \sqrt{3})$. Điều này tương đương với $\mathbb{Q}(\sqrt{2}, \sqrt{3}) \subset \mathbb{Q}(\sqrt{2} + \sqrt{3})$.

Nhưng mà $\mathbb{Q}(\sqrt{2} + \sqrt{3})$ là trường nhỏ nhất chứa \mathbb{Q} và $\sqrt{2} + \sqrt{3}$, và $\mathbb{Q}(\sqrt{2}, \sqrt{3})$ là trường nhỏ nhất chứa $\sqrt{2}$ và $\sqrt{3}$, kéo theo chứa cả $\sqrt{2} + \sqrt{3}$ nên $\mathbb{Q}(\sqrt{2} + \sqrt{3}) \equiv \mathbb{Q}(\sqrt{2}, \sqrt{3})$.

Ta có $\mathbb{Q}(\sqrt{6}) = \{a + b\sqrt{6} : a, b \in \mathbb{Q}\}$. Khi đó

$$\mathbb{Q}(\sqrt{6})(\sqrt{2} + \sqrt{3}) = \{\alpha + \beta(\sqrt{2} + \sqrt{3}) : \alpha, \beta \in \mathbb{Q}(\sqrt{6})\}. \quad (2.1)$$

Đặt $\alpha = a + b\sqrt{6}$ và $\beta = c + d\sqrt{6}$, như vậy mỗi phần tử trong $\mathbb{Q}(\sqrt{6})(\sqrt{2} + \sqrt{3})$ có dạng

$$\begin{aligned} a + b\sqrt{6} + (c + d\sqrt{6})(\sqrt{2} + \sqrt{3}) &= a + (c + 3d)\sqrt{2} + (c + 2d)\sqrt{3} + b\sqrt{6} \\ &= a' + b'\sqrt{2} + c'\sqrt{3} + d'\sqrt{6}, \end{aligned}$$

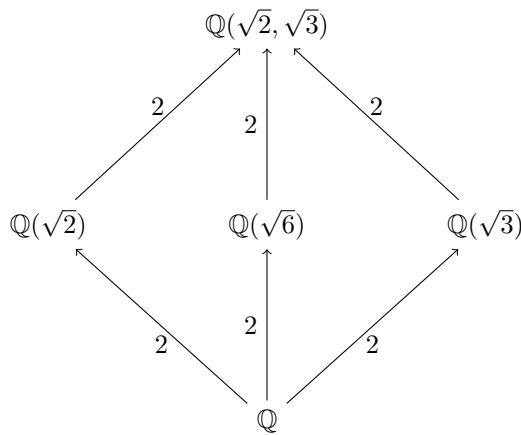
với $a \rightarrow a'$, $c + 3d \rightarrow b'$, $c + 2d \rightarrow c'$ và $b \rightarrow d'$.

Biến đổi này tương đương với hệ phương trình tuyễn tính

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} a' \\ b' \\ c' \\ d' \end{pmatrix}$$

Mã trận trên khả nghịch trên \mathbb{Q} , do đó $\mathbb{Q}(\sqrt{6})(\sqrt{2} + \sqrt{3}) \equiv \mathbb{Q}(\sqrt{2}, \sqrt{3})$. Ở dạng biểu diễn (2.1) ta suy ra mở rộng từ $\mathbb{Q}(\sqrt{6})$ lên $\mathbb{Q}(\sqrt{2}, \sqrt{3})$ là mở rộng bậc 2.

Sơ đồ mở rộng trường bây giờ như sau



Splitting Field**i Definition (Trường phân rã)**

Xét trường F và đa thức khác hằng

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

trên $F[x]$, ở đây $a_i \in F$ với mọi $i = 1, 2, \dots$

Trường mở rộng E của trường F được gọi là **trường phân rã** (hay **splitting field**) của $p(x)$ nếu tồn tại các phần tử $\alpha_1, \alpha_2, \dots, \alpha_n$ thuộc E sao cho $E = F(\alpha_1, \alpha_2, \dots, \alpha_n)$ và

$$p(x) = (x - \alpha_1)(x - \alpha_2) \cdots (x - \alpha_n).$$

Khi đó ta nói đa thức $p(x) \in F[x]$ phân rã (split) trong E nếu nó phân tích thành các nhân tử bậc nhất (tuyến tính) trong $E[x]$.

Nói nôm na, nếu đa thức có hệ số trong một trường F nào đó (tức thuộc $F[x]$) thì các nghiệm của nó nằm trong một trường lớn hơn chứa F .

i Example

Đa thức $f(x) = x^2 - 2$ trên $\mathbb{Q}[x]$ không có nghiệm trên \mathbb{Q} . Tuy nhiên $\mathbb{Q}(\sqrt{2})$ là trường chứa \mathbb{Q} và các nghiệm của $f(x)$ là $\pm\sqrt{2}$. Vì vậy $\mathbb{Q}(\sqrt{2})$ là trường phân rã của $f(x)$.

i Example

Đa thức $g(x) = x^2 + 1$ trên $\mathbb{Q}[x]$ không có nghiệm trên \mathbb{Q} . Tuy nhiên $g(x)$ có hai nghiệm là $\pm i$ và $\mathbb{Q}(i)$ chứa cả \mathbb{Q} và $\pm i$ nên $\mathbb{Q}(i)$ là một trường phân rã của $g(x)$.

Ở đây, bậc của đa thức $f(x) = x^2 - 2$ bằng với bậc của mở rộng trường từ \mathbb{Q} lên $\mathbb{Q}(\sqrt{2})$.

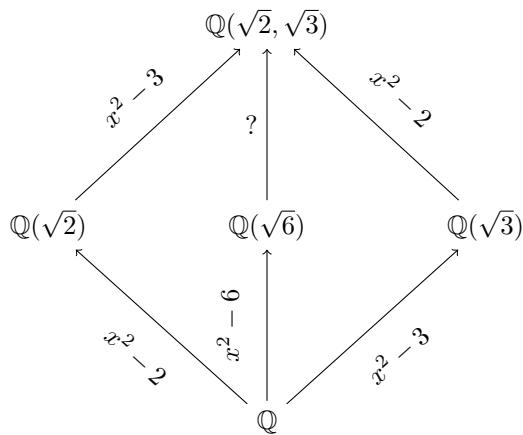
Tương tự, bậc của mở rộng trường từ \mathbb{Q} lên $\mathbb{Q}(\sqrt{3})$ bằng bậc đa thức $g(x) = x^2 - 3$, và bậc của mở rộng trường từ \mathbb{Q} lên $\mathbb{Q}(\sqrt{6})$ bằng với bậc đa thức $h(x) = x^2 - 6$.

Ở trên ta đã chứng minh bậc của mở rộng trường từ $\mathbb{Q}(\sqrt{2})$ lên $\mathbb{Q}(\sqrt{2}, \sqrt{3})$ là 2, điều này tương ứng với bậc của đa thức

$$k(x) = (x^2 - 2)(x^2 - 3) = x^4 - 5x + 6.$$

Tổng kết lại, nếu E là trường phân rã của F trên đa thức $f(x) \in F[x]$ thì bậc của mở rộng trường từ F lên E bằng với bậc của $f(x)$.

Từ sơ đồ mở rộng trường bên trên ta có sơ đồ với đa thức tương ứng.



Vậy còn mở rộng từ $\mathbb{Q}(\sqrt{6})$ thành $\mathbb{Q}(\sqrt{2}, \sqrt{3})$? Ở trên ta đã chứng minh rằng đây là mở rộng bậc 2. Vậy chúng ta cần tìm một đa thức bậc 2 có hệ số trong $\mathbb{Q}(\sqrt{6})$ mà không có nghiệm trong $\mathbb{Q}(\sqrt{6})$.

Quay lại một chút, ta đã mở rộng $\mathbb{Q}(\sqrt{6})$ lên $\mathbb{Q}(\sqrt{2}, \sqrt{3})$ với sự trợ giúp của phần tử $\sqrt{2} + \sqrt{3}$. Từ đây ta có thể xây dựng đa thức

$$m(x) = x^2 - (\sqrt{2} + \sqrt{3})^2 = x^2 - 5 - 2\sqrt{6}.$$

Đa thức $m(x)$ có hệ số trong $\mathbb{Q}(\sqrt{6})$ nhưng không có nghiệm trong $\mathbb{Q}(\sqrt{6})$, mà trong $\mathbb{Q}(\sqrt{2}, \sqrt{3})$. Ta có điều phải chứng minh.

Field automorphism

Remark

Tập hợp tất cả tự đẳng cấu (automorphism) trên trường F là nhóm với toán tử là hợp của các hàm. Kí hiệu là $Aut(F)$.

Chứng minh

Dễ thấy nếu σ và τ là tự đẳng cấu trên F thì $\sigma\tau$ và σ^{-1} cũng là tự đẳng cấu trên F (đều là song ánh). Ánh xạ đồng nhất hiển nhiên là tự đẳng cấu nên sẽ là phần tử đơn vị của nhóm.

Remark

Gọi E là mở rộng trường của F . Khi đó tập hợp tất cả tự đẳng cấu trên E mà cố định phần tử của F tạo thành một nhóm, nghĩa là tập hợp tất cả các tự đẳng cấu $\sigma : E \rightarrow E$ sao cho $\sigma(\alpha) = \alpha$ với mọi $\alpha \in F$ tạo thành một nhóm.

Chứng minh

Từ nhận xét ở trên thì tập hợp tất cả tự đẳng cấu trên E tạo thành một nhóm $Aut(E)$. Như vậy để chứng minh nhận xét này thì chúng ta chỉ cần chỉ ra tập hợp tất cả tự đẳng cấu của E mà cố định phần

tử trong F là một nhóm con của $\text{Aut}(E)$.

Gọi σ và τ là hai tự đẳng cấu của E sao cho $\sigma(\alpha) = \alpha$ và $\tau(\alpha) = \alpha$ với mọi $\alpha \in F$.

Khi đó $\sigma(\tau(\alpha)) = \sigma(\alpha) = \alpha$ nên ánh xạ hợp $\sigma\tau$ cũng nằm trong tập các tự đẳng cấu trên E mà cố định phần tử trên F .

Ngoài ra, $\sigma^{-1}(\alpha) = \alpha$ và ánh xạ đồng nhất cố định mọi phần tử trên E . Như vậy tập tất cả tự đẳng cấu trên E mà cố định mọi phần tử trong F là nhóm con của $\text{Aut}(E)$.

Ở trên ta kí hiệu nhóm tất cả các tự đẳng cấu trên E là $\text{Aut}(E)$. Bây giờ ta sẽ định nghĩa nhóm Galois.

❶ Definition (Nhóm Galois)

Gọi E là mở rộng trường của F . Ta nói **nhóm Galois** (hay **Galois group**) của E trên F là nhóm tất cả tự đẳng cấu của E mà cố định phần tử trên F , nghĩa là

$$G(E/F) = \{\sigma \in \text{Aut}(E) : \sigma(\alpha) = \alpha \text{ với mọi } \sigma \in F\}.$$

Nếu $f(x)$ là đa thức trong $F[x]$ và E là trường phân rã của $f(x)$ theo F thì ta định nghĩa nhóm Galois của $f(x)$ theo $G(E/F)$.

❷ Example

Liên hợp của số phức xác định bởi ánh xạ quen thuộc

$$\sigma : a + bi \rightarrow a - bi$$

là một tự đẳng cấu trên tập các số phức \mathbb{C} . Vì $\mathbb{R} \subset \mathbb{C}$ và

$$\sigma(a) = \sigma(a + 0i) = a - 0i = a$$

nên ánh xạ σ thuộc $G(\mathbb{C}/\mathbb{R})$.

❸ Example

Xét các trường $\mathbb{Q} \subset \mathbb{Q}(\sqrt{5}) \subset \mathbb{Q}(\sqrt{3}, \sqrt{5})$. Khi đó với $a, b \in \mathbb{Q}(\sqrt{5})$ thì ánh xạ

$$\sigma(a + b\sqrt{3}) = a - b\sqrt{3}$$

là tự đẳng cấu trên $\mathbb{Q}(\sqrt{3}, \sqrt{5})$ mà cố định $\mathbb{Q}(\sqrt{5})$.

Thật vậy, giả sử $a = s + t\sqrt{5}$ và $b = u + v\sqrt{5}$ với $s, t, u, v \in \mathbb{Q}$. Khi đó

$$\sigma(a + b\sqrt{3}) = a - b\sqrt{3} = s + t\sqrt{5} - (u + v\sqrt{5})\sqrt{3} = (s - u\sqrt{3}) + (t - v\sqrt{3})\sqrt{5}.$$

Ta có

$$a + b\sqrt{3} = (s + u\sqrt{3}) + (t + v\sqrt{3})\sqrt{5} \in \mathbb{Q}(\sqrt{5})$$

khi và chỉ khi $s + u\sqrt{3} \in \mathbb{Q}$ và $t + v\sqrt{3} \in \mathbb{Q}$. Nói cách khác $u = v = 0$. Như vậy

$$\sigma(a + b\sqrt{3}) = \sigma(s + t\sqrt{5}) = s + t\sqrt{5}.$$

Phần tử $s + t\sqrt{5} \in \mathbb{Q}(\sqrt{5})$ nên σ có định các phần tử trên $\mathbb{Q}(\sqrt{5})$.

Tương tự ta cũng có ánh xạ

$$\tau(a + b\sqrt{5}) = a - b\sqrt{5}$$

là tự đẳng cấu trên $\mathbb{Q}(\sqrt{3}, \sqrt{5})$ mà không có định $\mathbb{Q}(\sqrt{3})$.

Tuy nhiên ánh xạ hợp $\mu = \sigma\tau$ không có định cả $\mathbb{Q}(\sqrt{3})$ lẫn $\mathbb{Q}(\sqrt{5})$.

Chúng ta cũng chứng minh được rằng $\{id, \sigma, \tau, \mu\}$ là nhóm Galois của $\mathbb{Q}(\sqrt{3}, \sqrt{5})$ trên \mathbb{Q} vì các ánh xạ đó có định phần tử thuộc \mathbb{Q} (không chứa căn).

Như vậy nhóm Galois của $\mathbb{Q}(\sqrt{3}, \sqrt{5})$ có 4 phần tử, tức là $|G(\mathbb{Q}(\sqrt{3}, \sqrt{5})/\mathbb{Q})| = 4$. Ở trên phần trường phần rõ ràng cũng đã chỉ ra rằng $[\mathbb{Q}(\sqrt{3}, \sqrt{5}) : \mathbb{Q}] = 4$ nên ở đây

$$|G(\mathbb{Q}(\sqrt{3}, \sqrt{5})/\mathbb{Q})| = [\mathbb{Q}(\sqrt{3}, \sqrt{5}) : \mathbb{Q}] = 4.$$

Một điều nữa là trường $\mathbb{Q}(\sqrt{3}, \sqrt{5})$ cũng chính là không gian vector trên \mathbb{Q} với cơ sở là tập $\{1, \sqrt{3}, \sqrt{5}, \sqrt{15}\}$. Tập này cũng có 4 phần tử.

Một câu hỏi tự nhiên được đặt ra ở đây: việc số lượng phần tử của các tập hợp ở đây bằng nhau là ngẫu nhiên hay có lý do?

Remark

Cho E là một mở rộng trường của F và $f(x)$ là đa thức trên $F[x]$. Khi đó mỗi tự đẳng cấu trong $G(E/F)$ xác định một hoán vị các nghiệm của $f(x)$ trên E .

Chứng minh

Đặt

$$f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

và giả sử $\alpha \in E$ là một nghiệm của $f(x)$. Khi đó với tự đẳng cấu $\sigma \in G(E/F)$ ta có

$$\begin{aligned} 0 &= \sigma(0) \\ &= \sigma(f(\alpha)) \\ &= \sigma(a_0 + a_1\alpha + a_2\alpha^2 + \cdots + a_n\alpha^n) \\ &= \sigma(a_0) + \sigma(a_1\alpha) + \sigma(a_2\alpha^2) + \cdots + \sigma(a_n\alpha^n) \\ &= \sigma(a_0) + \sigma(a_1)\sigma(\alpha) + \sigma(a_2)\sigma(\alpha^2) + \cdots + \sigma(a_n)\sigma(\alpha^n). \end{aligned}$$

Trong đó hai dòng cuối là từ định nghĩa của tự đẳng cấu. Thêm nữa $\sigma(\alpha^k) = \sigma(\alpha)^k$ và do $\sigma \in G(E/F)$ nên $\sigma(a_i) = a_i$ với mọi $0 \leq i \leq n$. Như vậy có thể suy ra

$$0 = a_0 + a_1\sigma(\alpha) + a_2\sigma(\alpha)^2 + \cdots + a_n\sigma(\alpha)^n,$$

nói cách khác $\sigma(\alpha)$ cũng là nghiệm của $f(x)$.

Cho E là mở rộng trường của trường F . Hai phần tử $\alpha, \beta \in E$ được gọi là **liên hợp** trên F nếu chúng có cùng đa thức tối thiểu. Ở đây, đa thức tối thiểu được hiểu là đa thức có bậc nhỏ nhất nhận α và β làm nghiệm.

Ví dụ, trên trường $\mathbb{Q}(\sqrt{2})$ thì hai phần tử $\sqrt{2}$ và $-\sqrt{2}$ là liên hợp trên \mathbb{Q} vì chúng đều là nghiệm của đa thức tối giản $x^2 - 2$.

Remark

Nếu α và β liên hợp trên F thì tồn tại đẳng cấu $\sigma : F(\alpha) \rightarrow F(\beta)$ sao cho σ đồng nhất các phần tử trên F .

Khi đó ta có định lí quan trọng sau về số phần tử của nhóm Galois. Phần chứng minh hiện tại mình bỏ qua vì cần một vài bối cảnh khá dài.

Theorem

Đặt $f(x)$ là đa thức trên $F[x]$ và giả sử E là trường phân rã của $f(x)$ trên F . Nếu $f(x)$ không có nghiệm bội thì

$$|G(E/F)| = [E : F].$$

2.2.7 Bảng thuật ngữ lý thuyết nhóm

Bảng 2.1: Ý nghĩa thuật ngữ

Kí hiệu	Ý nghĩa
S_n	Nhóm đối xứng n phần tử
$\ker f$	Hạt nhân (kernel) của ánh xạ f
$\text{im } f$	Ảnh (Image) của ánh xạ f
\cong	Đẳng cấu nhóm (group isomorphism)

Bảng 2.2: Bảng thuật ngữ

Tiếng Việt	Tiếng Anh	Tiếng Nga
nhóm	group	группа
nhóm con	subgroup	подгруппа
nhóm đối xứng	symmetric group	симметричная группа
nhóm hoán vị	permutation group	группа постановок
chu trình	cycle	цикл
chu trình độc lập	independent cycles	независимые циклы
đồng cấu nhóm	group homomorphism	гомоморфизм групп
đơn cấu	monomorphism	
toàn cấu	epimorphism	
đẳng cấu	isomorphism	
tự đẳng cấu	automorphism	
hạt nhân (đồng cấu)	kernel	ядро
ảnh (đồng cấu)	image	
tác động nhóm	group action	действие групп
vành	ring	кольцо
vành với đơn vị	ring with identity	кольцо с единицей
vành giao hoán	commutative ring	коммутативное кольцо
trường	field	поле
trường hữu hạn	finite field	конечное поле
mở rộng trường	extension field	
trường phân rã	splitting field	

2.3 Hình học

2.3.1 Hình học giải tích

Theo dòng lịch sử

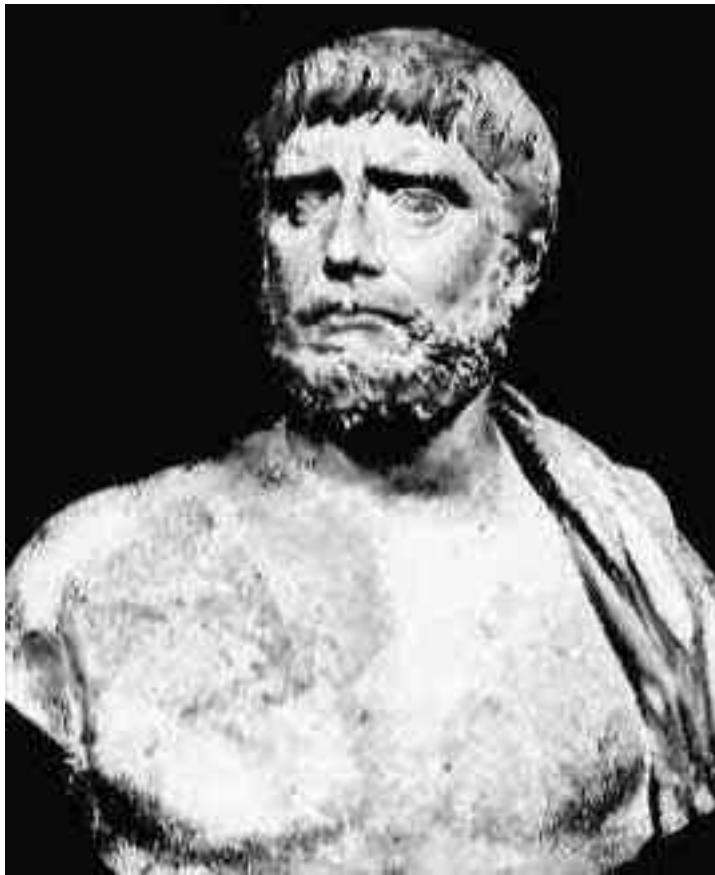
Hình học xuất hiện từ thời xa xưa, xuất phát từ những nhu cầu thực tế nhất của con người là đo đạc để phân chia đất đai, xây dựng, canh tác, ... Từ đó con người đã có nhận thức rất sớm về quan hệ song song và vuông góc giữa hai đường thẳng.

Một cách hình ảnh (mà thật ra hình học là môn học về hình ảnh) thì hai đường thẳng song song không cắt nhau dù có kéo dài chúng ra vô tận. Các đường thẳng song song luôn có nhiều điều thú vị, cả ở mặt phẳng Euclid lẫn trong không gian. Đầu tiên phải kể đến định lý mang tên triết gia vĩ đại của Hy Lạp: Thales.

Thales của Miletus

Thales của Miletus được cho rằng sinh vào khoảng năm 624 Trước Công nguyên (TCN) và mất năm 547 TCN tại Miletus (Thổ Nhĩ Kì ngày nay)¹.

¹ <https://mathshistory.st-andrews.ac.uk/Biographies/Thales/>



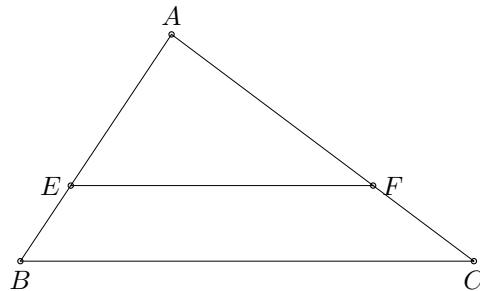
Hình 2.24: Thales của Miletus

Ông được xem là nhà triết học đầu tiên khi không cố gắng giải thích tự nhiên bằng thần thoại hay các thế lực siêu nhiên như trước. Trường phái triết học do ông sáng lập, trường phái Milet, cho rằng mọi vật có nguồn gốc từ nước. Nhà triết học nổi tiếng Aristotle đánh giá rằng Thales là người sáng lập ra **triết học duy vật sơ khai**.

Trong toán học, Thales được biết tới với định lý mang tên ông về các đường song song. Định lý Thales được phát biểu như sau:

➊ Theorem 1.3 (Định lý Thales)

Trong một tam giác, đường thẳng song song với một cạnh chia trên hai cạnh còn lại các đoạn thẳng tương ứng tỉ lệ.



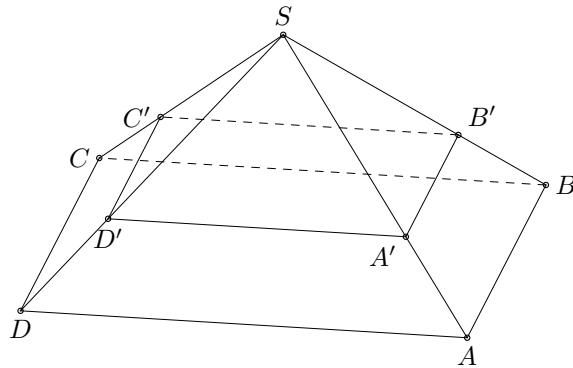
Hình 2.25: Định lý Thales trên mặt phẳng

Theo định lý Thales, nếu EF song song với BC thì ta có $\frac{AE}{AB} = \frac{AF}{AC} = \frac{EF}{BC}$ (thales1).

Không dừng lại ở mặt phẳng, khi mở rộng lên không gian định lý Thales cũng cho chúng ta một kết quả quan trọng khi nói tới các mặt phẳng song song nhau.

1 Theorem 1.4 (Định lý Thales trong không gian)

Trong khối chóp, mặt phẳng song song mặt đáy chia các cạnh nối từ đỉnh hình chóp tới các đỉnh của mặt phẳng đáy các đoạn thẳng tương ứng tỉ lệ.



Hình 2.26: Định lý Thales trong không gian

Theo định lý Thales, nếu mặt phẳng $(ABCD)$ song song với mặt phẳng $(A'B'C'D')$ thì $\frac{SA}{SA'} = \frac{SB}{SB'} = \frac{SC}{SC'} = \frac{SD}{SD'}$ (thales2).

Pythagoras của Samos

Khi nhắc tới vuông góc, chúng ta thường nhớ tới định lý ngày nào được học ở thời học sinh: định lý Pythagoras. Định lý này nói về quan hệ giữa độ dài các cạnh trong một tam giác vuông. Định lý tuy đơn giản nhưng có ý nghĩa rất quan trọng trong đời sống và khoa học của con người suốt chiều dài lịch sử. Đây cũng là tiền đề cho định lý mang tính lịch sử của nhân loại: định lý cuối cùng của Fermat.



Hình 2.27: Pythagoras của Samos

Pythagoras của Samos cũng là nhà triết học Hy Lạp cổ, được cho rằng sinh vào khoảng năm 570 TCN và mất năm 490 TCN².

Ông được học tập từ nhà triết học Thales và cũng có nhiều đóng góp cho sự phát triển của toán học, thiên văn học và âm nhạc. Tuy nhiên khác với thầy mình, trường phái triết học của ông cho rằng những con số là nguồn gốc của vạn vật và sử dụng những con số để giải thích những hiện tượng khoa học. Từ đây, các lý thuyết về âm nhạc được ra đời, cụ thể là các mối liên hệ về tần số với sự rung của dây nhạc cụ.

Ông là một trong những người hiếm hoi cho phép cả phụ nữ đi học ở lớp của mình vào thời ấy. Điều đó giúp phổ biến toán học nói riêng và kiến thức nói chung tới nhiều tầng lớp nhân dân. Tuy nhiên ông cũng có một hội kín rất thú vị. Như đã nói ở trên, trường phái triết học Pythagoras có gắng giải thích nguồn gốc vạn vật bằng những con số. Điều này đã dẫn họ tới những khám phá động trời vào thời ấy.

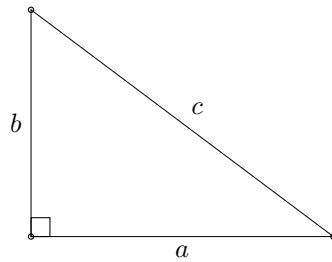
Một trong những khám phá đó là về sự tồn tại của số vô tỉ dựa vào định lý mang tên ông. Lịch sử đã chỉ ra rằng trước Pythagoras, người Babylon và Ai Cập đã tìm ra rất nhiều bộ số nguyên (a, b, c) thỏa mãn $a^2 + b^2 = c^2$ là độ dài ba cạnh tam giác vuông. Định lý Pythagoras mà ngày nay chúng ta biết được phát biểu rằng:

i Theorem 1.5 (Định lý Pythagoras)

Trong một tam giác vuông, bình phương độ dài cạnh huyền bằng tổng bình phương độ dài hai cạnh góc vuông.

² <https://mathshistory.st-andrews.ac.uk/Biographies/Pythagoras/>

Như vậy nếu gọi độ dài cạnh huyền là c , độ dài hai cạnh góc vuông lần lượt là a và b thì $a^2 + b^2 = c^2$ (`{numref}`pythagoras1``).



Hình 2.28: Định lý Pythagoras

Nếu $a = b = 1$ thì sao? Khi đó bình phương độ dài cạnh huyền $c^2 = 2$. Tuy nhiên không thể tìm ra một số hữu tỉ nào để bình phương lên là 2 cả. Phát hiện này là một chấn động đối với thời Pythagoras và ông yêu cầu tất cả thành viên trong hội phải giữ kín bí mật về sự phát hiện này. Tuy nhiên thông tin vẫn lọt ra ngoài và truyền thuyết kể rằng ông đã xử tử tội chết cho thành viên của hội không tuân thủ.

Pythagoras đã đưa một khái niệm cực kì quan trọng trong toán học, gọi là **chứng minh** (hay **proof, доказательство**). Để chứng minh một mệnh đề là đúng, chúng ta cần các mệnh đề (thường đơn giản hơn) đúng trước đó. Bằng các phép suy luận thích hợp dựa trên các mệnh đề đúng trước đó, chúng ta có thể kết luận rằng mệnh đề cần chứng minh là đúng. Phép chứng minh có thể gọi là "xương sống" của toán học, vì nếu không có một phép chứng minh đúng đắn thì một mệnh đề không thể được xác định được là có đúng hay không. Trong trường hợp của Fermat, khi ông đưa ra định lý Fermat nhưng không kèm chứng minh (vì lề sách quá chặt nên không viết lời giải được) thì chúng ta không thể biết định lý Fermat có đúng hay không (?).

Nếu việc suy luận dựa trên các mệnh đề, hoặc định lý, đã đúng trước đó, thì phải có một lúc nào đó việc này dừng lại. Chúng ta không thể suy ngược tới vô hạn lần được. Do đó chúng ta cần những mệnh đề luôn đúng nhưng tính đúng đắn của nó được kiểm nghiệm trong thực tiễn. Chúng được gọi là **tiên đề** (hay **axiom, аксиома**). Nhân vật tiếp theo được đề cập tới sẽ dẫn chúng ta tới hệ thống tiên đề làm nền tảng cho hình học.

Euclid của Alexandria

Đúng vậy, Euclid là người đặt nền móng cho hình học với bộ sách nổi tiếng **Elements** của mình. Trong bộ sách này đề cập tới những tiên đề, định lý làm nền tảng cho bộ môn hình học và vẫn còn ý nghĩa cho tới tận ngày nay. Những gì viết trong đó không quá xa lạ với những gì được giảng dạy trong nhà trường.



Hình 2.29: Euclid của Alexandria

Euclid của Alexandria sinh vào khoảng năm 325 TCN và mất vào khoảng năm 265 TCN³. Thông tin về ông không có nhiều. Nhưng chỉ mỗi bộ sách **Elements** cũng đủ để người đời sau cho rằng ông là người có ảnh hưởng nhất trong 2000 năm lịch sử phát triển của toán học.

Năm tiên đề cơ bản của hình học được ông phát biểu trong bộ **Elements** được phát biểu như sau:

1. Qua hai điểm bất kì luôn vẽ được một đường thẳng
2. Đường thẳng có thể kéo dài vô hạn về cả hai phía
3. Ta có thể xác định một đường tròn bằng tâm và bán kính của nó
4. Mọi góc vuông đều bằng nhau
5. Nếu một đường thẳng cắt hai đường thẳng khiến tổng hai góc trong cùng phía nhỏ hơn hai vuông thì hai đường thẳng đó chắc chắn sẽ cắt nhau tại một điểm nào đó

Tiên đề số 5 là rắc rối và phức tạp nhất. Nó không thực sự tự nhiên và có nhiều sự vướng mắc. Đây chính là tiên đề cho sự ra đời của hình học phi-Euclid hơn 1500 năm sau.

Bộ **Elements** của Euclid bao gồm 13 quyển. Trong đó đề cập tới rất nhiều vấn đề của hình học, từ những phần tử đơn giản nhất cấu tạo nên hình học là điểm, đoạn thẳng, đường thẳng, tới những hình học lớn hơn như hình chữ nhật, hình tròn, đa giác, mặt phẳng. Thậm chí ông cũng đã có những dấu chân ở hình học không gian như hình chóp, hình cầu, hình nón ([2], [9]).

³ <https://mathshistory.st-andrews.ac.uk/Biographies/Euclid/>

Phương pháp tọa độ trong mặt phẳng

Cuộc cách mạng trong hình học xảy ra khi nhà toán học lăng tử René Descartes phát minh ra hệ tọa độ và từ đó mọi đối tượng hình học có thể được biểu diễn bởi các phương pháp đại số như phương trình, đẳng thức.



Hình 2.30: René Descartes (1596-1650)

Danh mục thuật ngữ và kí hiệu

Đầu tiên chúng ta thống nhất các thuật ngữ cũng như kí hiệu được sử dụng kể từ đây.

Điểm là đơn vị cơ bản của hình học. Bất kì đối tượng hình học nào cũng là một *tập hợp điểm*. Điểm được kí hiệu bởi chữ in hoa, ví dụ như A , B_1 , B_2 .

Đường thẳng đi qua hai điểm phân biệt cho trước. Đường thẳng có thể kéo dài vô hạn về hai phía. Đường thẳng được kí hiệu bởi chữ in thường hoặc chữ Hy Lạp trong ngoặc đơn, ví dụ như (d) , (Δ) .

Đoạn thẳng chỉ phần đường thẳng nằm giữa hai điểm và bao gồm hai điểm đó.

Nửa đường thẳng chỉ phần đường thẳng nằm một phía của một điểm trên đường thẳng và chỉ kéo dài vô hạn về phía đó.

Vector là đoạn thẳng có hướng. Với điểm đầu là A và điểm cuối là B thì vector từ A tới B được kí hiệu là \overrightarrow{AB} . Để chỉ một vector không cần biết điểm đầu và điểm cuối ta dùng chữ thường in đậm, ví dụ như \mathbf{a} .

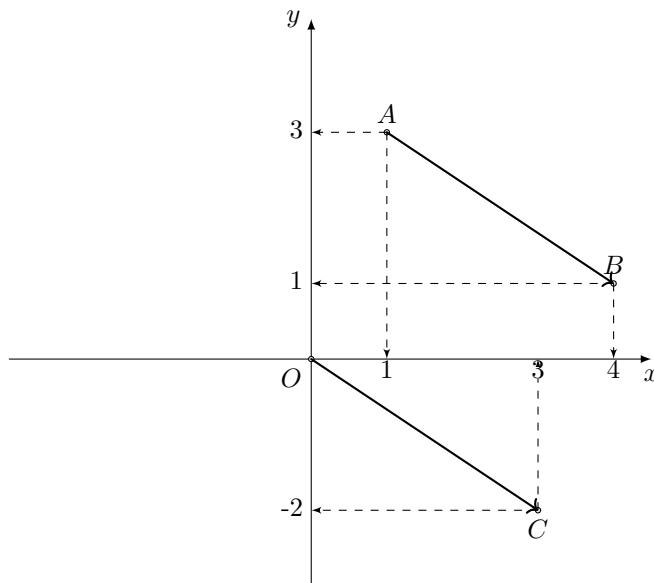
Góc giữa hai vector \overrightarrow{OA} và \overrightarrow{OB} là góc $\angle AOB$ và kí hiệu là $(\overrightarrow{OA}, \overrightarrow{OB})$.

Tương tự đối với vector \mathbf{a} và \mathbf{b} thì góc giữa chúng kí hiệu là (\mathbf{a}, \mathbf{b}) .

Vector trong mặt phẳng

Trong hệ tọa độ Oxy với tâm O và hai trục Ox (trục hoành) và Oy (trục tung) vuông góc nhau, đặt $O = (0, 0)$ là tọa độ của tâm O .

Tiếp theo, mọi điểm trong mặt phẳng Euclid đi liền với cặp số (x, y) chỉ tọa độ của điểm đó. Ví dụ $A = (1, 3)$, $B = (4, 1)$.



Hình 2.31: Tọa độ của điểm trong mặt phẳng

Tọa độ của điểm cũng là tọa độ của vector từ O tới điểm đó. Ở [Hình 2.31](#) thì $\overrightarrow{OA} = (1, 3)$ và $\overrightarrow{OB} = (4, 1)$. Tọa độ của vector \overrightarrow{AB} khi đó sẽ là

$$\overrightarrow{AB} = \overrightarrow{OB} - \overrightarrow{OA} = (4, 1) - (1, 3) = (3, -2).$$

Cũng theo [Hình 2.31](#) thì ta thấy $\overrightarrow{AB} = \overrightarrow{OC} = (3, -2)$.

Như vậy, nếu ta có hai điểm $A = (x_A, y_A)$ và $B = (x_B, y_B)$ thì tọa độ vector \overrightarrow{AB} là

$$\overrightarrow{AB} = (x_B - x_A, y_B - y_A).$$

Tích vô hướng của hai vector $\mathbf{a} = (x_1, y_1)$ và $\mathbf{b} = (x_2, y_2)$ được định nghĩa là

$$\mathbf{a} \cdot \mathbf{b} = x_1 x_2 + y_1 y_2.$$

Ta cũng có thể kí hiệu tích vô hướng là $\langle \mathbf{a}, \mathbf{b} \rangle$ nhưng mình sẽ không dùng kí hiệu này.

Ta kí hiệu $\|\mathbf{a}\|$ là độ dài (chuẩn Euclid, Euclid norm) của vector \mathbf{a} . Trong hệ tọa độ Descartes vuông góc, theo định lý Pythagoras, độ dài của vector là độ dài cạnh huyền tam giác vuông (Hình 2.31). Như vậy, độ dài đoạn thẳng AB với $A = (x_A, y_A)$ và $B = (x_B, y_B)$ là

$$AB = \|\overrightarrow{AB}\| = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}.$$

Khi đó cosin góc giữa hai vector \mathbf{a} và \mathbf{b} là

$$\cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|} = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} \cdot \sqrt{x_2^2 + y_2^2}}.$$

Nếu góc giữa hai vector bằng 90 độ thì hai vector được gọi là vuông góc nhau. Khi đó tích vô hướng $\mathbf{a} \cdot \mathbf{b} = 0$.

Phương trình đường thẳng trong mặt phẳng

Theo tiên đề Euclid, một đường thẳng được xác định khi biết hai điểm phân biệt thuộc đường thẳng đó. Trong hệ tọa độ, chúng ta có hai cách tìm phương trình đường thẳng.

Sử dụng vector pháp tuyến. Vector pháp tuyến của đường thẳng là vector vuông góc với mọi vector có phương là đường thẳng đó. Giả sử $\mathbf{v} = (a, b)$ là vector pháp tuyến của đường thẳng đi qua điểm $M_0 = (x_0, y_0)$. Khi đó đường thẳng đi qua qua M_0 nhận \mathbf{v} làm vector pháp tuyến là *tập hợp điểm* $M = (x, y)$ trên mặt phẳng sao cho $\mathbf{v} \cdot \overrightarrow{M_0 M} = 0$. Điều này tương đương với

$$\mathbf{v} \cdot \overrightarrow{M_0 M} = a \cdot (x - x_0) + b \cdot (y - y_0) = 0.$$

Sử dụng vector chỉ phương. Vector chỉ phương của đường thẳng là vector có phương song song với đường thẳng đó. Giả sử $\mathbf{v}' = (a', b')$ là vector chỉ phương của đường thẳng đi qua điểm $M_0 = (x_0, y_0)$. Khi đó đường thẳng đi qua M_0 nhận \mathbf{v}' làm vector chỉ phương là *tập hợp điểm* $M = (x, y)$ trên mặt phẳng sao cho \mathbf{v}' cùng phương với $\overrightarrow{M_0 M}$. Điều này tương đương với

$$\mathbf{v}' \parallel \overrightarrow{M_0 M} \Leftrightarrow \frac{x - x_0}{a'} = \frac{y - y_0}{b'}.$$

1. Cả hai cách biểu diễn khi khai triển ra đều có dạng $ax + by + c = 0$ với c là hằng số. Đây được gọi là *dạng tổng quát* của phương trình đường thẳng.
2. Cách viết $\frac{x - x_0}{a'} = \frac{y - y_0}{b'}$ được gọi là *dạng chính tắc* của phương trình đường thẳng.
3. Dạng chính tắc của phương trình đường thẳng còn có một tác dụng đặc biệt khác

$$\frac{x - x_0}{a'} = \frac{y - y_0}{b'} = t$$

với $t \in \mathbb{R}$. Khi đó tọa độ $M = (x, y)$ có thể được biểu diễn dưới dạng

$$\begin{cases} x = x_0 + a't \\ y = y_0 + b't \end{cases}, \quad t \in \mathbb{R}.$$

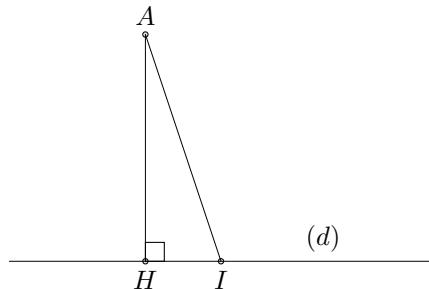
Đây được gọi là *phương trình dạng tham số*.

Chúng ta chú ý rằng nếu đường thẳng song song với một trong hai trục tọa độ thì vector chỉ phương của nó sẽ cùng phương với vector đơn vị $(1, 0)$ hoặc $(0, 1)$. Do đó không thể viết dưới dạng chính tắc được (không chia cho 0) nhưng có thể viết dưới dạng tổng quát hoặc dạng tham số.

Khoảng cách giữa điểm và đường thẳng

Nhắc lại một chút kiến thức cơ sở. **Khoảng cách** từ một điểm A nằm ngoài đường thẳng (d) là độ dài đoạn thẳng AH với $H \in (d)$ sao cho AH nhỏ nhất ([Hình 2.32](#)).

Khi đó H được gọi là **hình chiếu** của A lên đường thẳng (d) và AH là **khoảng cách** từ A tới (d) . Do AH là đoạn thẳng có độ dài ngắn nhất, điều này xảy ra khi $AH \perp (d)$.



Hình 2.32: Hình chiếu và khoảng cách tới đường thẳng

Như vậy, để tìm hình chiếu của điểm A lên đường thẳng (d) , ta dựng đường thẳng đi qua điểm A và vuông góc với (d) .

Giả sử phương trình đường thẳng (d) với vector pháp tuyến $\mathbf{v} = (a, b)$ là $(d) : ax + by + c = 0$.

Gọi (d') là đường thẳng đi qua $A = (x_0, y_0)$ và vuông góc với d . Do \mathbf{v} là vector pháp tuyến của (d) nên \mathbf{v} là vector chỉ phương của (d') . Khi đó phương trình dạng tham số của (d') là

$$\begin{cases} x = x_0 + at \\ y = y_0 + bt \end{cases}, \quad t \in \mathbb{R}.$$

Gọi H là hình chiếu của A lên (d) . Khi đó H là giao điểm của (d) và (d') . Vì $H \in (d')$ nên tọa độ của H có dạng $(x_0 + at, y_0 + bt)$ với t nào đó thuộc \mathbb{R} . Chúng ta sẽ đi tìm t này.

Vì $H \in (d)$ nên ta thay tọa độ của H vừa tìm được vào phương trình của (d) thu được

$$a(x_0 + at) + b(y_0 + bt) + c = 0 \Leftrightarrow t = -\frac{ax_0 + by_0 + c}{a^2 + b^2}.$$

Như vậy là ta đã tìm được t từ đó xác định được tọa độ của H .

Từ đây ta tính được khoảng cách từ A tới (d) , hay nói cách khác là độ dài đoạn thẳng AH . Ta có $A = (x_0, y_0)$ và $H = (x_0 + at, y_0 + bt)$ nên $\overrightarrow{AH} = (at, bt)$, suy ra

$$\begin{aligned} AH &= \|\overrightarrow{AH}\| = \sqrt{(at)^2 + (bt)^2} = |t| \sqrt{a^2 + b^2} \\ &= \left| -\frac{ax_0 + by_0 + c}{a^2 + b^2} \right| \cdot \sqrt{a^2 + b^2} = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}. \end{aligned}$$

Đạo hàm

Phép tính vi tích phân đã được con người nghiên cứu từ lâu. Câu chuyện về ai là người phát minh ra phép tính vi tích phân: Newton hay Leibniz, được coi là một trong những vụ tranh cãi đáng xấu hổ nhất lịch sử toán học. Nhưng họ cũng đã để lại một mảnh đất màu mỡ cho toán học về sau.

Cơ học và sự ra đời của đạo hàm

Trường phái Newton sử dụng đạo hàm như công cụ khảo sát vận tốc từ quãng đường. Ở bậc trung học chúng ta biết rằng **vận tốc trung bình** bằng quãng đường chia thời gian. Tuy nhiên điều đó chỉ đúng cho **chuyển động thẳng đều**. Nếu quãng đường là một hàm số phụ thuộc thời gian (quãng đường là $s(t)$ với t là thời gian) thì điều đó không đúng nữa.

Do quãng đường phụ thuộc thời gian nên có thể là vận tốc cũng phụ thuộc thời gian? Hợp lý đây. Nhưng với mỗi một giá trị thời gian t cho ta một vị trí $s(t)$ trên trực số, còn vận tốc thì không thể phụ thuộc một giá trị thời gian được. Rõ ràng vật phải di chuyển một quãng đường từ thời gian t_0 tới t_1 thì mới có vận tốc trên quãng đường đó chứ?

Cách tiếp cận ở đây là, chúng ta cho sự thay đổi thời gian, tức hiệu $\Delta t = t_1 - t_0$, rất nhỏ. Khi đó vật đi từ $s(t_0)$ tới $s(t_1)$, vậy là chúng ta có thể tính vận tốc với công thức $v = \frac{s(t_1) - s(t_0)}{t_1 - t_0}$. Do Δt rất nhỏ, hay **tiến về 0**, thì vận tốc gần như xảy ra vào đúng một thời điểm. Do đó vận tốc lúc này được gọi là **vận tốc tức thời**. Đó cũng chính là ý nghĩa cơ học và sự ra đời của đạo hàm theo trường phái Newton.

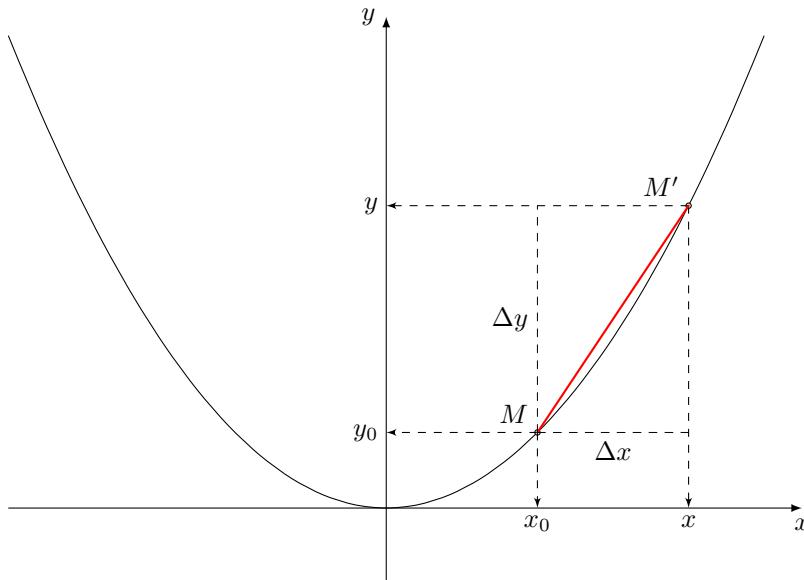
Ý nghĩa hình học của đạo hàm

Xét hàm số $y = f(x)$ liên tục trên khoảng (a, b) chứa điểm x_0 .

Gọi $M' = (x, y)$ là một điểm thuộc hàm số $y = f(x)$. Khi đó đạo hàm của $f(x)$ tại x_0 là giới hạn

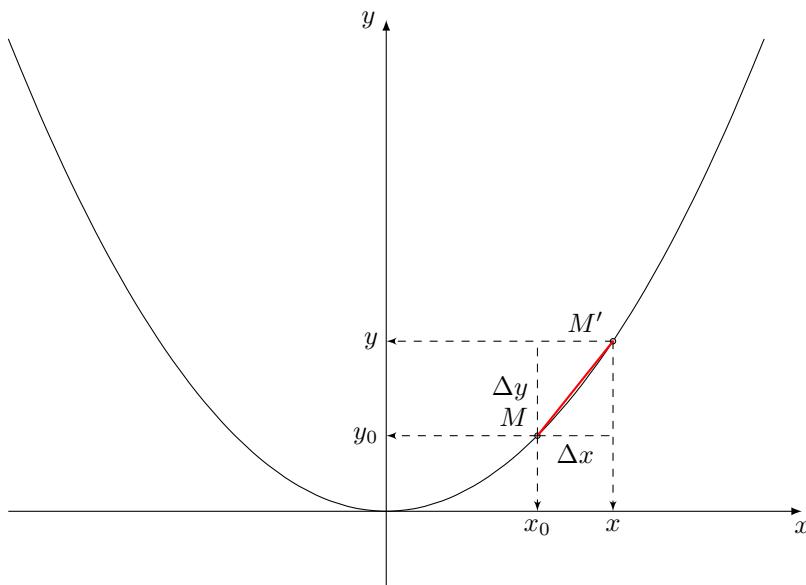
$$\lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0} = \lim_{\Delta x \rightarrow 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} = \lim_{\Delta x \rightarrow 0} \frac{\Delta y}{\Delta x}.$$

Xét Hình 2.33, tỉ số $\Delta y / \Delta x$ là tangent của góc hợp bởi trực hoành Ox và đường thẳng MM' .



Hình 2.33: Hệ số góc (trường hợp 1)

Tiếp theo, xét Hình 2.34, ta thấy đường thẳng MM' ngày càng tiến sát lại với đường cong. Như vậy, khi Δx tiến tới 0 thì đường thẳng MM' cắt đường cong tại hai điểm càng sát nhau. Đến khi hai điểm đó trùng nhau, đường thẳng MM' chỉ đi qua đúng một điểm thuộc đường cong và khi đó MM' trở thành tiếp tuyến của đường cong tại điểm $M = (x_0, y_0)$.



Hình 2.34: Hệ số góc (trường hợp 2)

Khi đó $f'(x_0)$ là tangent của góc hợp bởi MM' và trục hoành Ox , hay nói cách khác là **hệ số góc** (hay **угловой коэффициент**) của đường tiếp tuyến. Thêm nữa $f'(x_0) = \frac{\Delta y}{\Delta x} = \frac{y - y_0}{x - x_0}$ nên phương trình đường tiếp tuyến đi qua $M = (x_0, y_0)$ là

$$y = f'(x_0)(x - x_0) + y_0.$$

Tích phân

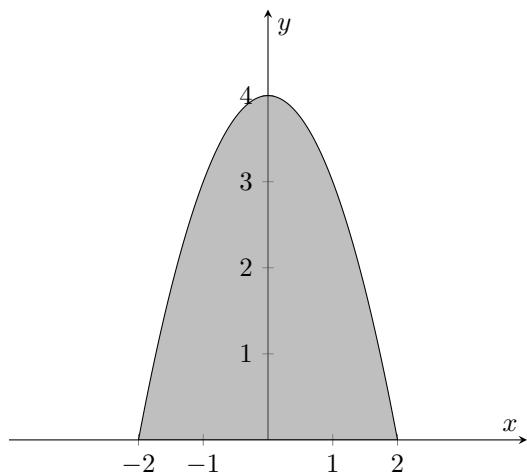
Tích phân là khái niệm quan trọng trong giải tích. Sau đây sẽ trình bày cách tính tích phân theo tổng Riemann.

Tích phân và phân chia diện tích

Xét phương trình của một đường cong $y = f(x) > 0$ trên đoạn $[a; b]$.

Theo định nghĩa, tích phân từ a tới b là diện tích phần hình phẳng giới hạn bởi đường cong $y = f(x)$, trục hoành Ox và hai trục đứng $x = a, x = b$.

Ở Hình 2.35, diện tích phần tô màu xám là tích phân từ -2 tới 2 của hàm số $f(x) = -x^2 + 4$.



Hình 2.35: Tích phân từ -2 tới 2 của $f(x) = -x^2 + 4$

Chúng ta có thể tính diện tích hình chữ nhật, hình thang, hình vuông. Vậy có cách nào để tính diện tích một hình giới hạn bởi các đường cong bất kì không?

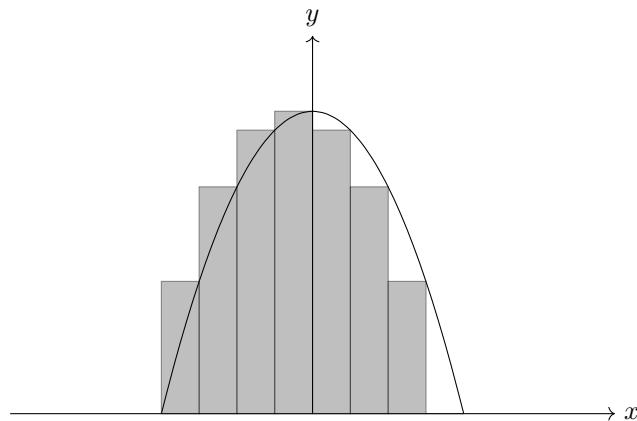
Có đấy! Chúng ta sẽ tính xấp xỉ bằng tổng diện tích các hình chữ nhật.

Ví dụ với hàm số $f(x) = -x^2 + 4$ ở trên, ta chia đoạn $[a; b]$ thành n phần bằng nhau

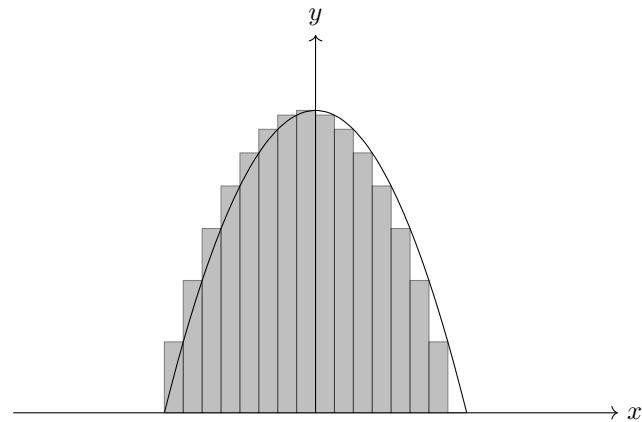
$$a = x_0 < x_1 < \dots < x_{n-1} < x_n = b.$$

Trong đó $x_{i+1} - x_i$ cố định và bằng $\frac{b-a}{n}$.

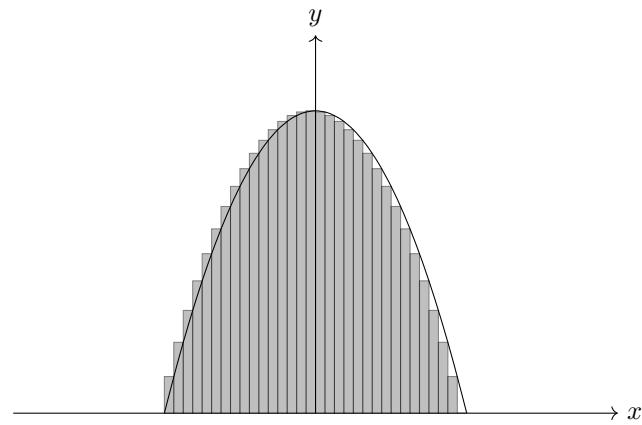
Đối với Hình 2.36 ta xấp xỉ bằng 7 hình chữ nhật. Đối với Hình 2.37 ta xấp xỉ bằng 15 hình chữ nhật. Đối với Hình 2.38 ta xấp xỉ bằng 31 hình chữ nhật.



Hình 2.36: Xấp xỉ diện tích bởi 7 hình chữ nhật



Hình 2.37: Xấp xỉ diện tích bởi 15 hình chữ nhật



Hình 2.38: Xấp xỉ diện tích bởi 31 hình chữ nhật

Càng dùng nhiều hình chữ nhật, tổng diện tích của chúng càng gần với diện tích cần tìm, hay là tích phân cần tìm.

Ở ba hình trên, mỗi hình chữ nhật trong đó có chiều rộng bằng nhau là $\frac{b-a}{n}$ với n là số đoạn. Chiều dài là $f(x_i)$ với $x_i = a + \frac{b-a}{n}i$, $i = 1, 2, \dots, n$ (biên sau).

Cụ thể hơn, hình chữ nhật từ x_{i-1} tới x_i sẽ có chiều dài là $f(x_i)$ và chiều rộng là $\frac{b-a}{n}$. Ở đây lưu ý rằng việc chọn chiều dài không bắt buộc phải chọn biên sau. Chúng ta hoàn toàn có thể chọn chiều dài là $f(x_{i-1})$, hoặc $\max f(x)$, $\min f(x)$ trên đoạn $[x_{i-1}; x_i]$.

Khi đó, tổng diện tích của các hình chữ nhật là

$$\sum_{i=1}^n (x_i - x_{i-1}) \cdot f(x_i) = \sum_{i=1}^n \frac{b-a}{n} f(x_i).$$

Khi số lượng hình chữ nhật tăng lên tới vô hạn thì tổng diện tích sẽ tiến tới diện tích chính xác của hình

cần tìm, hay nói cách khác là tích phân. Do đó kết quả sẽ là

$$\int_a^b f(x) dx = \lim_{n \rightarrow \infty} \sum_{i=1}^n \frac{b-a}{n} f(x_i), \quad x_i = a + \frac{b-a}{n} i.$$

Ví dụ tính tích phân qua tổng Riemann

Ví dụ, tính tích phân từ -2 tới 2 của hàm số $f(x) = -x^2 + 4$ ở trên.

Ta có $b = 2$ và $a = -2$ nên

$$\begin{aligned} \frac{b-a}{n} f(x_i) &= \frac{4}{n} \left(-\left(-2 + \frac{4}{n} i \right)^2 + 4 \right) \\ &= \frac{4}{n} \left(-4 + \frac{16}{n} i - \frac{16}{n^2} i^2 + 4 \right) \\ &= \frac{64}{n} \left(\frac{i}{n} - \frac{i^2}{n^2} \right). \end{aligned}$$

Tính tổng i từ 1 tới n ta có $\sum_{i=1}^n i = \frac{n(n+1)}{2}$.

Tính tổng i^2 từ 1 tới n ta có $\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$.

Từ đây ta suy ra

$$\begin{aligned} \sum_{i=1}^n \frac{64}{n} \left(\frac{i}{n} - \frac{i^2}{n^2} \right) &= \frac{64}{n^2} \sum_{i=1}^n i - \frac{64}{n^3} \sum_{i=1}^n i^2 \\ &= -\frac{64}{n^2} \cdot \frac{n(n+1)}{2} - \frac{64}{n^3} \cdot \frac{n(n+1)(2n+1)}{6}. \end{aligned}$$

Khi n tiến tới vô cực thì biểu thức trên tiến tới $\frac{64}{2} - \frac{64 \cdot 2}{6} = \frac{32}{3}$. Đây chính là giá trị của tích phân $\int_{-2}^2 (-x^2 + 4) dx$.

Ba đường Conic

Ba đường Conic bao gồm ellipse, hyperbol và parabol.

Ellipse

Definition 1.37 (Ellipse)

Đường ellipse là tập hợp các điểm sao cho tổng khoảng cách từ nó tới hai điểm cố định là hằng số.

Nói cách khác, với hai điểm cố định F_1 và F_2 , tập hợp các điểm M sao cho

$$MF_1 + MF_2 = 2a,$$

với a là hằng số tạo thành đường ellipse.

Ở trên hệ tọa độ, nếu ta chọn F_1 và F_2 nằm trên Ox và đối xứng qua Oy , tức là $F_1 = (-c, 0)$ và $F_2 = (c, 0)$ với $c \geq 0$, thì các điểm $M = (x, y)$ nằm trên ellipse thỏa

$$MF_1 + MF_2 = \sqrt{(x+c)^2 + y^2} + \sqrt{(x-c)^2 + y^2} = 2a,$$

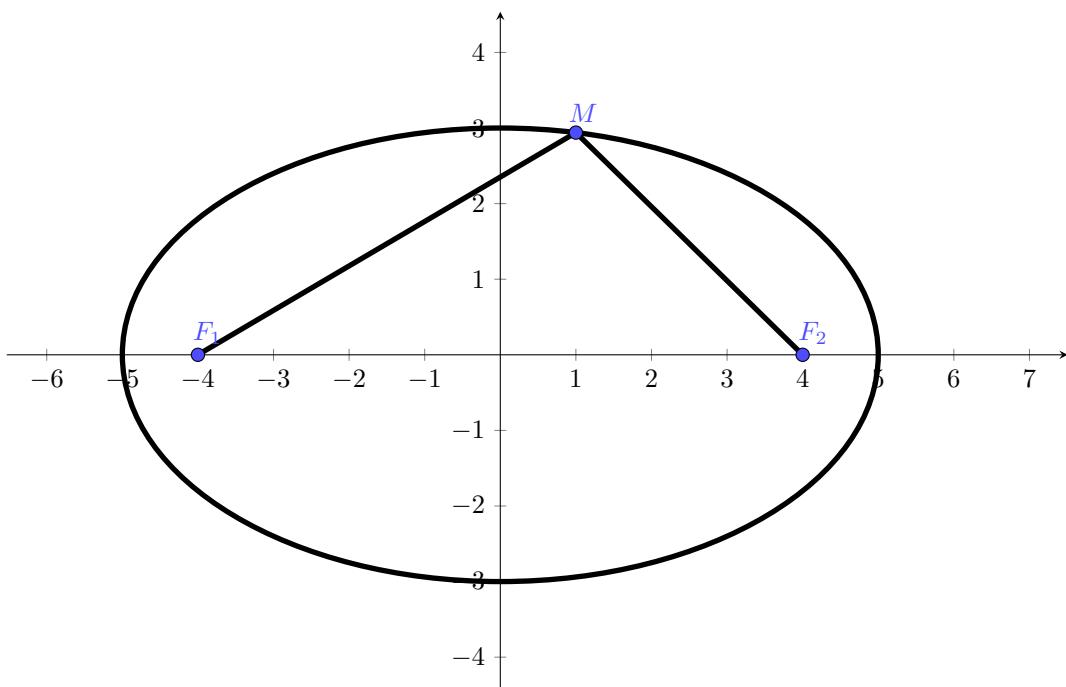
tương ứng với biến đổi thành phương trình

$$\frac{x^2}{a^2} + \frac{y^2}{a^2 - c^2} = 1.$$

Đặt $b^2 = a^2 - c^2$ thì phương trình của ellipse trở thành

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1.$$

Phương trình này gọi là **phương trình chính tắc**.



Hình 2.39: Ellipse với phương trình $\frac{x^2}{25} + \frac{y^2}{9} = 1$

Trong phương trình

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

thì a là khoảng cách từ tâm tới hai biên trái hoặc phải, nên a là **độ dài bán trục lớn**.

Tương tự, b là **độ dài bán trục nhỏ** (khoảng cách từ tâm tới hai biên trên dưới).

Từ cách đặt $b^2 = a^2 - c^2$ tương đương với $c^2 = a^2 - b^2$ thì c gọi là **tiêu cự** của ellipse.

Các điểm F_1 và F_2 gọi là **tiêu điểm** của ellipse.

Với ví dụ trên $\frac{x^2}{25} + \frac{y^2}{9} = 1$ thì $a = 5$, $b = 3$. Ta suy ra $c = 4$ (lưu ý là $a, b > 0$ và $c \geq 0$).

Các đỉnh nằm ở các tọa độ $(-a, 0)$, $(a, 0)$, $(0, b)$, $(0, -b)$.

Các tiêu điểm nằm ở $(-c, 0)$, $(c, 0)$.

❶ Remark 1.10

Khi $c = 0$, tức là hai tiêu điểm trùng nhau, ta có đường tròn.

Tâm sai của ellipse là $e = \frac{c}{a} < 1$.

Hyperbol

❶ Definition 1.38 (Hyperbol)

Đường hyperbol là tập hợp các điểm sao cho giá trị tuyệt đối hiệu số khoảng cách từ nó tới hai điểm cố định là hằng số.

Nói cách khác, với hai điểm cố định F_1 và F_2 , tập hợp các điểm M sao cho

$$|MF_1 - MF_2| = 2a,$$

với a là hằng số tạo thành đường hyperbol.

Ở trên hệ tọa độ, nếu ta chọn F_1 và F_2 nằm trên Ox và đối xứng qua Oy , tức là $F_1 = (-c, 0)$ và $F_2 = (c, 0)$, thì các điểm $M = (x, y)$ nằm trên hyperbol thỏa

$$|MF_1 - MF_2| = |\sqrt{(x+c)^2 + y^2} - \sqrt{(x-c)^2 + y^2}| = 2a,$$

tương ứng với biến đổi thành phương trình

$$\frac{x^2}{a^2} - \frac{y^2}{a^2 - c^2} = 1.$$

Đặt $b^2 = a^2 - c^2$ thì phương trình của hyperbol trở thành

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1.$$

Đường hyperbol cắt trục Ox tại hai điểm $A_1 = (-a, 0)$ và $A_2 = (a, 0)$.

Tiêu điểm của hyperbol ở $F_1 = (-c, 0)$ và $F_2 = (c, 0)$.

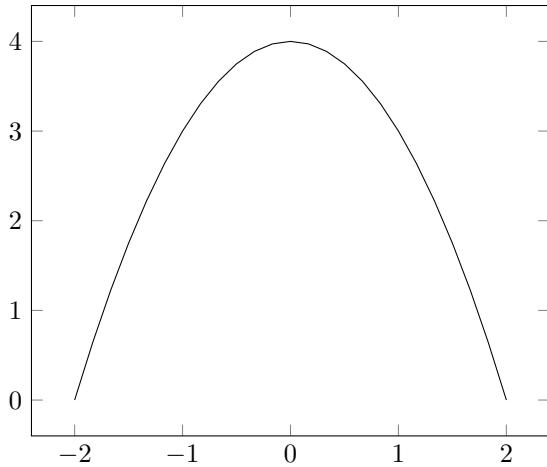
Đường hyperbol có hai tiệm cận là đường thẳng $y = \frac{b}{a}x$ và $y = -\frac{b}{a}x$.

Tâm sai của hyperbol là $e = \frac{c}{a} > 1$.

Parabol

❶ Definition 1.39 (Parabol)

Đường parabol là tập hợp các điểm cách đều một điểm cố định và một đường thẳng cố định.

Hình 2.40: Parabol với phương trình $y = -x^2 + 4$

Nghĩa là, với điểm cố định F và đường thẳng cố định (d) , parabol là tập hợp các điểm M sao cho

$$MF = d(M, (d))$$

với $d(M, (d))$ là khoảng cách từ M tới đường thẳng (d) .

Phép dời tọa độ cho phép ta dời một hình parabol có đỉnh ở bất kì điểm nào về gốc tọa độ.

Tức là, không mất tính tổng quát, ta chỉ cần xét các parabol dạng $y = ax^2$ là đủ.

Điểm cố định ở trên được gọi là **tiêu điểm**. Đường thẳng cố định ở trên gọi là **đường chuẩn**.

Parabol có tính đối xứng nên tiêu điểm nằm trên Oy . Đặt tọa độ của tiêu điểm là $F = (0, f)$.

Đường chuẩn nằm ngang nên ta có parabol là các điểm $M = (x, y)$ sao cho

$$MF = \sqrt{x^2 + (y - f)^2}, \quad d(M, (d)) = y + f,$$

trường hợp M trùng với đỉnh nên điều kiện của parabol xảy ra tương đương với M cách đều tiêu điểm và đường chuẩn, nghĩa là đường chuẩn có dạng $y = -f$.

Do đó $\sqrt{x^2 + (y - f)^2} = y + f$. Bình phương và biến đổi ta thu gọn được $f = \frac{1}{4a}$.

Thường thì ta đặt $p = f$, khi đó phương trình parabol trở thành $x^2 = 4py$.

Đây là dạng chính tắc của parabol với trục đối xứng dọc.

Tâm sai của parabol là $e = \frac{c}{a} = 1$.

Phép biến hình

Trong thực tế chúng ta hay gặp các vấn đề về việc di dời một hình nào đó sang một vị trí khác trong mặt phẳng, không gian và phải đảm bảo giữ nguyên một số quan hệ nhất định. Trong đó cơ bản nhất và được ứng dụng rộng rãi là phép dời hình và phép đồng dạng.

Phép dời hình

❶ Definition 1.40 (Phép dời hình)

Phép dời hình từ hình \mathcal{H} thành hình \mathcal{H}' là một ánh xạ f biến mỗi điểm thuộc hình \mathcal{H} thành điểm thuộc hình \mathcal{H}' sao cho khoảng cách giữa hai điểm bất kỳ trong \mathcal{H} bảo toàn khi qua \mathcal{H}' .

Nói cách khác, với mọi điểm $A, B \in \mathcal{H}$, ánh xạ f biến A thành A' và B thành B' , nghĩa là $A', B' \in \mathcal{H}'$, thì $A'B' = AB$.

Một số phép dời hình cơ bản là dời hình theo vector (dời theo một hướng nhất định), đối xứng qua trực, đối xứng qua tâm, quay quanh tâm hoặc quay quanh trực nào đó.

Phép dời hình theo vector

Phép dời hình theo vector $\vec{v} \neq \vec{0}$ (phép tịnh tiến) biến điểm A thành điểm A' sao cho $\overrightarrow{AA'} = \vec{v}$.

Dễ thấy đây là phép dời hình vì với mọi A, B biến thành A', B' ta có

$$\overrightarrow{A'B'} = \overrightarrow{A'A} + \overrightarrow{AB} + \overrightarrow{BB'},$$

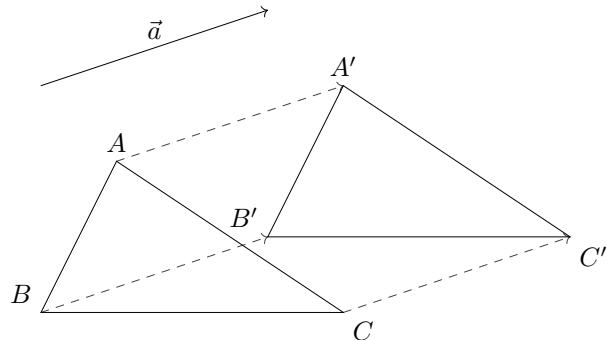
mà ta có

$$\overrightarrow{A'A} = -\vec{v} = -\overrightarrow{BB'}$$

nên

$$\overrightarrow{A'B'} = \overrightarrow{AB}.$$

Vector bằng nhau thì độ dài cũng bằng nhau. Ta có điều phải chứng minh.

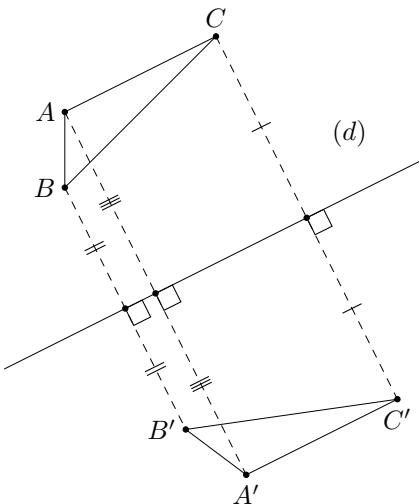


Hình 2.41: Tịnh tiến theo vector $\text{vec}\{\mathbf{a}\}$

Phép đối xứng qua đường thẳng cố định

Cho đường thẳng cố định (d).

Phép đối xứng qua đường thẳng (d) biến điểm A thành điểm A' sao cho AA' cắt (d) tại trung điểm AA' và đường thẳng đi qua AA' vuông góc với (d).

Hình 2.42: Đổi xứng qua đường thẳng (d)

Phép đổi xứng qua tâm cố định

Cho điểm cố định O .

Phép đổi xứng tâm O biến điểm A thành điểm A' sao cho $\overrightarrow{OA} = -\overrightarrow{OA'}$. Nói cách khác O là trung điểm đoạn thẳng AA' .

Phép quay quanh tâm cố định

Cho điểm cố định O .

Phép quay (mặc định là ngược chiều đồng hồ) quanh tâm O theo một góc cố định φ biến điểm A thành điểm A' sao cho

$$\widehat{(\overrightarrow{OA}, \overrightarrow{OA'})} = \varphi.$$

Trên mặt phẳng chúng ta có thể biểu diễn phép quay dưới hệ tọa độ như sau.

Giả sử vector \overrightarrow{OA} có độ dài là r và hợp với trục Ox một góc α .

Khi đó, giả sử tọa độ của $\overrightarrow{OA} = (x, y)$ thì ta có

$$\begin{aligned} x &= r \cos \alpha, \\ y &= r \sin \alpha. \end{aligned}$$

Nếu ta quay vector này quanh gốc tọa độ, ngược chiều kim đồng hồ một góc φ thì thực ra góc (mới) hợp bởi vector $\overrightarrow{OA'}$ và trục Ox là $\alpha + \varphi$.

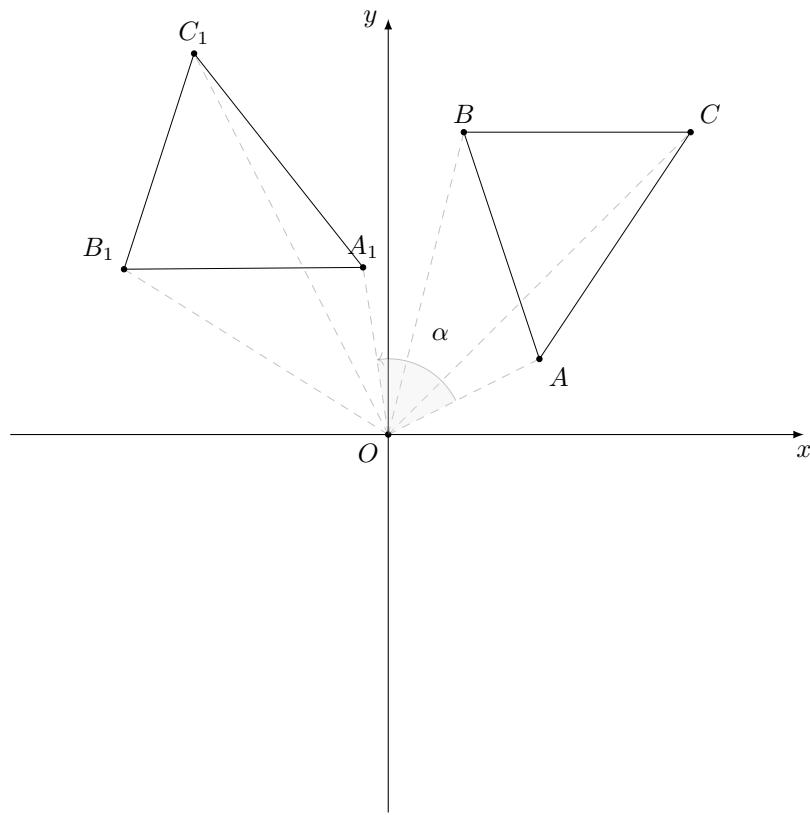
Do đó

$$\begin{aligned} x' &= r \cos(\alpha + \varphi), \\ y' &= r \sin(\alpha + \varphi). \end{aligned}$$

Khi khai triển ra

$$\begin{aligned} x' &= r \cos \alpha \cos \varphi - r \sin \alpha \sin \varphi = x \cos \varphi - y \sin \varphi \\ y' &= r \sin \alpha \cos \varphi + r \cos \alpha \sin \varphi = y \cos \varphi + x \sin \varphi. \end{aligned}$$

Dễ thấy, phép quay bảo toàn khoảng cách từ tâm O tới điểm đó. Nghĩa là $OA = OA'$.



Phép biến hình trên mặt phẳng tọa độ

Gọi $A = (x, y)$ là tọa độ ban đầu của điểm và $A' = f(A) = (x', y')$ là tọa độ điểm A' , là kết quả của phép biến hình f lên điểm A .

Chúng ta lần lượt xét các phép biến hình đã liệt kê ở trên và chuyển về phép nhân ma trận.

Các phép nhân ma trận cho phép chúng ta hợp các phép biến đổi liên tiếp thành một biến đổi lớn.

Giả sử \mathbf{A}_T là ma trận ứng với phép tịnh tiến, \mathbf{A}_R là ma trận ứng với phép quay.

Khi đó với phép nhân ma trận, nếu ta muốn thực hiện liên tiếp phép tịnh tiến và phép quay thì ta có $\mathbf{A}_T \cdot \mathbf{A}_R$.

Hợp các phép dời hình không có tính giao hoán, cũng như phép nhân ma trận. Do đó thứ tự thực hiện phép dời hình khác nhau thì thứ tự nhân ma trận cũng khác nhau.

Một yêu cầu về phép biến đổi tuyến tính là không có vector tự do, nghĩa là biến đổi có dạng

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathbf{A} \cdot \begin{pmatrix} x \\ y \end{pmatrix},$$

với \mathbf{A} là ma trận $n \times n$.

Phép dời hình theo vector

Đặt $\vec{v} = (a, b)$ là vector tịnh tiến. Khi đó $\overrightarrow{AA'} = \vec{v}$ sẽ tương đương với

$$x' - x = a, \quad y' - y = b,$$

hay tương đương với $x' = x + a$ và $y' = y + b$.

Dễ thấy rằng kết quả có thể viết ở dạng:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix}.$$

Tuy nhiên chúng ta cần một biến đổi tuyến tính không có vector $\begin{pmatrix} a \\ b \end{pmatrix}$.

Lúc này, thay vì sử dụng ma trận 2×2 thì ta chuyển thành 3×3 và công thức trở thành:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

Ma trận $\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix}$ là ma trận tịnh tiến (translation vector) tương ứng với phép tịnh tiến điểm $A = (x, y)$ theo vector $\vec{v} = (a, b)$.

Phép đối xứng qua đường thẳng cố định

Tương tự việc tìm tọa độ hình chiếu của một điểm lên một đường thẳng cho trước, ở đây chúng ta tìm hình chiếu rồi lấy đối xứng điểm ban đầu qua tâm là hình chiếu.

Gọi $(d) : ax+by+c=0$ là đường thẳng bất kì. Gọi $M = (x_M, y_M)$ là điểm cần lấy đối xứng và $N = (x_N, y_N)$ là điểm đối xứng của M qua đường thẳng (d) .

Gọi $I = (x_I, y_I)$ là hình chiếu của M lên (d) . Khi đó I là trung điểm của đoạn thẳng MN và $MN \perp (d)$.

Do đường thẳng $MN \perp (d)$ và đường thẳng (d) có vector pháp tuyến là $\mathbf{v} = (a, b)$ nên \mathbf{v} là vector chỉ phương của đường thẳng MN .

Như vậy đường thẳng MN có phương trình tham số

$$\begin{cases} x = x_M + at \\ y = y_M + bt \end{cases}, \quad t \in \mathbb{R}.$$

Do $I \in MN$ nên $x_I = x_M + at_0$ và $y_I = y_M + bt_0$. Vì $I \in (d)$ nên thay tọa độ điểm I vào phương trình đường thẳng (d) ta có

$$a(x_M + at_0) + b(y_M + bt_0) + c = 0 \iff t_0 = -\frac{ax_M + by_M + c}{a^2 + b^2}.$$

Như vậy tọa độ điểm I là

$$\begin{aligned} x_I &= x_M + at_0 = x_M - a \frac{ax_M + by_M + c}{a^2 + b^2} = \frac{b^2 x_M - aby_M - ac}{a^2 + b^2}, \\ y_I &= y_M + bt_0 = y_M - b \frac{ax_M + by_M + c}{a^2 + b^2} = \frac{-abx_M + a^2 y_M - bc}{a^2 + b^2}. \end{aligned}$$

Vì I là trung điểm MN nên

$$\begin{aligned} x_N &= 2x_I - x_M = \frac{2(b^2 x_M - aby_M - ac)}{a^2 + b^2} - x_M = \frac{b^2 - a^2}{a^2 + b^2} x_M - \frac{2ab}{a^2 + b^2} y_M - \frac{2ac}{a^2 + b^2}, \\ y_N &= 2y_I - y_M = \frac{2(-abx_M + a^2 y_M - bc)}{a^2 + b^2} - y_M = -\frac{2ab}{a^2 + b^2} x_M + \frac{a^2 - b^2}{a^2 + b^2} y_M - \frac{2bc}{a^2 + b^2}. \end{aligned}$$

Như vậy ta cũng có thể biểu diễn phép đổi xứng trực bằng phép nhân ma trận

$$\begin{pmatrix} x_N \\ y_N \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{b^2 - a^2}{a^2 + b^2} & -\frac{2ab}{a^2 + b^2} & -\frac{2ac}{a^2 + b^2} \\ -\frac{2ab}{a^2 + b^2} & \frac{a^2 - b^2}{a^2 + b^2} & -\frac{2bc}{a^2 + b^2} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_M \\ y_M \\ 1 \end{pmatrix}.$$

Phép đổi xứng qua tâm cố định

Theo định nghĩa ở trên, giả sử tâm O có tọa độ (a, b) . Điều kiện $\overrightarrow{OA} = \overrightarrow{OA'}$ tương đương với:

$$x - a = -(x' - a), \quad y - a = -(y' - b),$$

hay tương đương với

$$x' = -x + 2a, \quad y' = -y + 2b.$$

Tương tự bên trên, ta muốn một phép nhân ma trận mà không cộng thêm vector ngoài. Ta có phép nhân

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 2a \\ 0 & -1 & 2b \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Như vậy, ma trận $\begin{pmatrix} -1 & 0 & 2a \\ 0 & -1 & 2b \\ 0 & 0 & 1 \end{pmatrix}$ là ma trận đổi xứng qua tâm $O = (a, b)$.

Phép quay quanh tâm cố định

Từ công thức của phép quay quanh tâm là gốc tọa độ:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix},$$

ta thấy rằng trong các phép biến đổi trên ta cần ma trận 3×3 thay vì 2×2 nên phép quay cũng cần ma trận 3×3 để có thể hợp với các phép biến hình khác.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

Như vậy ma trận $\begin{pmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$ thể hiện phép quay quanh tâm O là gốc tọa độ.

2.3.2 Đường đoán thời

Lời nói đầu

Động lực để tác giả viết bài này là sau khi đọc về sự ra đời phép tính vi tích phân cùng vụ tranh cãi đáng xấu hổ trong lịch sử toán học giữa Newton và Leibniz, cùng với bài toán của Johann Bernoulli.

Bài viết này được tham khảo nhiều nguồn¹². Đây là tài liệu học tập cá nhân. Tác giả hy vọng rằng bài viết nhỏ này sẽ giúp ích được cho các bạn học sinh, sinh viên đam mê toán và vật lý (mặc dù tác giả không phải dân lý hihi).

¹² Miguel A. Lerma, *A simple derivation of the equation for the brachistochrone curve*, URL -<https://sites.math.northwestern.edu/~mlerma/papers-and-preprints/brachistochrone.pdf>

¹³ Lê Quang Ánh, *Gia đình Bernoulli: một dòng họ Toán học*, trang 7, ULR - <https://rosetta.vn/lequanganh/gia-dinh-bernoulli-mot-dong-ho-toan-hoc/>

Bối cảnh lịch sử

Thế kỷ 17 đã chứng kiến một drama có thể gọi là đáng xấu hổ nhất lịch sử toán học. Hai nhà toán học có ảnh hưởng rất lớn lại vướng vào một vụ kiện tụng và tranh cãi khó coi để xem ai là người phát minh ra phép tính vi tích phân. Vâng, chúng ta đang nói đến Newton và Leibniz. Vào thời điểm đó có một nhà toán học xuất sắc thuộc một dòng họ cũng gồm rất nhiều nhân vật xuất sắc đã đưa ra một bài toán đố cho các nhà toán học trên thế giới. Bài toán đó đã chứng minh được ưu thế vượt trội trong phương pháp vi tích phân của Leibniz.

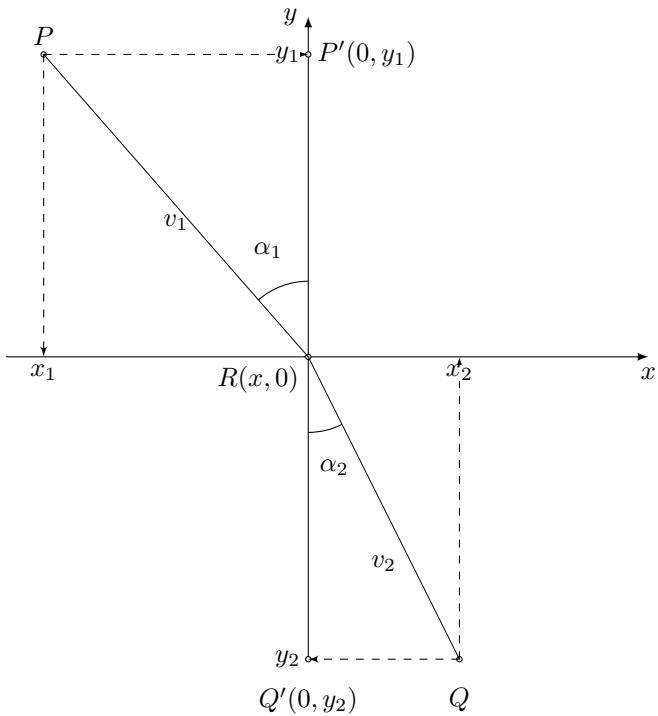
Nhà toán học xuất sắc đó là Johann Bernoulli, thuộc dòng họ Bernoulli nổi tiếng. Bài toán đó được phát biểu như sau:

Cho hai điểm A và B nằm trong mặt phẳng thẳng đứng P (A cao hơn B). Hãy xác định đường nối hai điểm A và B và nằm trong mặt phẳng P sao cho một điểm chỉ chịu trọng lực chạy từ A đến B trong thời gian ngắn nhất.

Chúng ta đã biết rằng đường đi ngắn nhất giữa hai điểm là đoạn thẳng nối hai điểm đó. Tuy nhiên trong bài toán của Johann Bernoulli thì đại lượng ngắn nhất cần tìm không phải khoảng cách giữa hai điểm mà là thời gian di chuyển giữa hai điểm. Mục tiêu cần làm ở bài toán này là xác định đường đi (hay quỹ đạo) thời gian ngắn nhất đó. Do đó bài toán này được gọi là bài toán **đường đoạn thời** (hay brachistochrone curve).

Để giải bài toán này chúng ta cần một định luật cũng về thời gian ngắn nhất. Đó là nguyên lý thời gian ngắn nhất của Fermat và một hệ quả của nó là định luật Snell-Descartes.

Định luật Snell-Descartes



Hình 2.43: Định luật Snell-Descartes

Nguyên lý thời gian ngắn nhất của Fermat phát biểu rằng

Khi ánh sáng truyền từ môi trường này sang môi trường khác thì nó luôn truyền đi theo đường nhanh nhất.

Hệ quả của nguyên lý của Fermat là định luật Snell-Descartes mà chúng ta thường thấy ở chương trình vật lý ở phổ thông dưới dạng định luật khúc xạ ánh sáng

$$\frac{\sin \alpha_1}{\sin \alpha_2} = \frac{v_1}{v_2},$$

với α_1 và α_2 lần lượt là góc hợp bởi tia vào và tia ra với pháp tuyến tại điểm tới, v_1 và v_2 là vận tốc truyền trong môi trường ở nửa trên và nửa dưới Ox ([Hình 2.43](#)).

Để chứng minh định luật trên, ta thấy rằng v_1 là vận tốc khi di chuyển từ điểm P tới điểm R nên thời gian t_1 đi từ điểm P tới R là

$$t_1 = \frac{\|\overrightarrow{PR}\|}{v_1} = \frac{\sqrt{(x - x_1)^2 + y_1^2}}{v_1}.$$

Lưu ý rằng tia sáng không truyền tới gốc tọa độ $O(0,0)$ mà truyền tới một điểm $R(x,0)$ là vì điểm bắt đầu là $P(x_1, y_1)$ và ánh sáng truyền đi theo đường nhanh nhất (theo nguyên lý Fermat) nên không có gì đảm bảo rằng nó sẽ truyền tới $O(0,0)$.

Tương tự, thời gian t_2 đi từ điểm R tới Q là

$$t_2 = \frac{\|\overrightarrow{RQ}\|}{v_2} = \frac{\sqrt{(x - x_2)^2 + y_2^2}}{v_2}.$$

Kết hợp hai phương trình của t_1 và t_2 lại thì tổng thời gian di chuyển từ P tới Q biểu diễn theo x là

$$T(x) = t_1 + t_2 = \frac{\sqrt{(x - x_1)^2 + y_1^2}}{v_1} + \frac{\sqrt{(x - x_2)^2 + y_2^2}}{v_2}.$$

Đạo hàm theo x ta có

$$T'(x) = \frac{x - x_1}{v_1 \sqrt{(x - x_1)^2 + y_1^2}} + \frac{x - x_2}{v_2 \sqrt{(x - x_2)^2 + y_2^2}}.$$

Để ý rằng $x > x_1$ nên $x - x_1 = \|\overrightarrow{PP'}\|$. Tương tự $x_2 - x = \|\overrightarrow{QQ'}\|$. Để tìm cực trị ta cho đạo hàm bằng 0 rồi tính đạo hàm cấp 2. Ta có

$$T'(x) = 0 \Leftrightarrow \frac{\|\overrightarrow{PP'}\|}{v_1 \|\overrightarrow{PR}\|} - \frac{\|\overrightarrow{QQ'}\|}{v_2 \|\overrightarrow{RQ}\|} = 0 \Leftrightarrow \frac{\sin \alpha_1}{v_1} - \frac{\sin \alpha_2}{v_2} = 0.$$

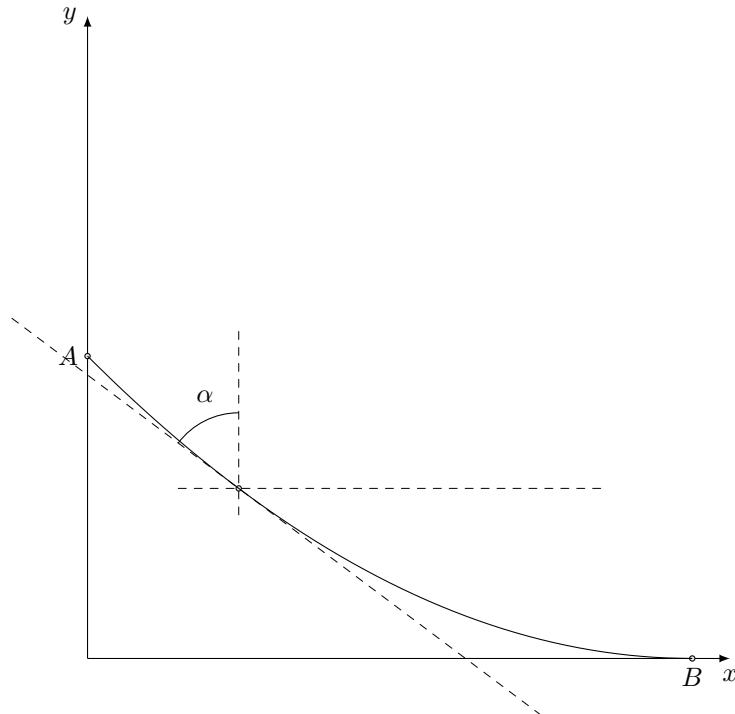
Như vậy $\frac{\sin \alpha_1}{v_1} = \frac{\sin \alpha_2}{v_2}$. Đạo hàm cấp 2 tương ứng là

$$T''(x) = \frac{y_1^2}{v_1((x - x_1)^2 + y_1^2)} + \frac{y_2^2}{v_2((x - x_2)^2 + y_2^2)} > 0.$$

Do đó x thỏa $T'(x) = 0$ ở trên là cực tiểu và định luật Snell-Descartes được chứng minh.

Đường cong Cycloid

Đáp án cho bài toán mà Johann Bernoulli đặt ra là đường cong cycloid. Sau đây sẽ trình bày cách giải bài toán mà Johann Bernoulli phát biểu.



Hình 2.44: Đường cycloid

Phương của vận tốc tức thời tại một điểm khi một vật đi theo một quỹ đạo đường cong là tiếp tuyến với đường cong tại điểm đó. Khi đó góc α trong định luật Snell-Descartes sẽ có liên hệ với hệ số góc của tiếp tuyến với đường cong. Nói rõ hơn, góc hợp bởi tiếp tuyến và trục Ox là $\alpha + \frac{\pi}{2}$ và hệ số góc của tiếp tuyến là $\tan\left(\alpha + \frac{\pi}{2}\right) = \frac{dy}{dx}$ (Hình 2.44).

Ta có $\tan\left(\alpha + \frac{\pi}{2}\right) = -\cot\alpha$ và $1 + \cot^2\alpha = \frac{1}{\sin^2\alpha}$ nên

$$\frac{1}{\sin^2\alpha} = 1 + \cot^2\alpha = 1 + \left(\frac{dy}{dx}\right)^2.$$

Giả sử tọa độ của A là (x_0, y_0) . Khi một điểm di chuyển từ A tới B , gọi (x, y) là tọa độ của điểm đó trên đường cong. Theo định luật bảo toàn cơ năng thì

$$mgy_0 = \frac{1}{2}mv^2 + mgy,$$

với v là vận tốc tức thời tại điểm (x, y) và mgy là thế năng tại điểm đó. Như vậy ta có

$$v^2 = 2g(-y + y_0).$$

Theo định luật Snell-Descartes thì $\frac{v}{\sin\alpha}$ là một hằng số khi nằm trong cùng môi trường. Do đó tồn tại số r cố định sao cho $\frac{v^2}{\sin^2\alpha} = 4gr$. Từ hai biểu thức của v^2 và $\frac{1}{\sin^2\alpha}$ ở trên ta có

$$\frac{v^2}{\sin^2\alpha} = 2g(-y + y_0) \left(1 + \left(\frac{dy}{dx}\right)^2\right) = 4gr.$$

Suy ra

$$\left(\frac{dy}{dx}\right)^2 = \frac{2r}{y_0 - y} - 1. \quad (2.2)$$

Tới đây ta thấy rằng bậc của dy và dx ở vế trái là giống nhau, trong khi vế phải chỉ có y mà không có x . Do đó "bắt chước" cách đổi biến của đường tròn, đặt

$$\begin{cases} x = a\theta + b \cos \theta \\ y = c + d \sin \theta \end{cases}$$

với a, b, c, d là các số thực cần tìm, θ là góc hợp bởi Oy và đoạn thẳng nối tâm O và điểm trên đường cong (theo góc α). Lưu ý rằng khi thay $\theta = 0$ và $\theta = \pi/2$ vào hai phương trình trên ta phải thu được hai điểm trên hai trục tọa độ.

Lấy vi phân hai phương trình trên ta có

$$\begin{cases} dx = a - b \sin \theta d\theta, \\ dy = d \cos \theta d\theta. \end{cases}$$

Thay vào phương trình (2.2) ta được

$$\frac{d^2 \cos^2 \theta}{(a - b \sin \theta)^2} = \frac{2r}{y_0 - c - d \sin \theta} - 1 = \frac{2r - y_0 + c + d \sin \theta}{y_0 - c - d \sin \theta}.$$

Do $\cos^2 \theta = 1 - \sin^2 \theta = (1 - \sin \theta)(1 + \sin \theta)$ nên ta muốn chọn a và b có thể rút gọn được cho tử số.

Trường hợp 1. $a = b$, ta thu được

$$\frac{d^2(1 + \sin \theta)}{a^2(1 - \sin \theta)} = \frac{(2r - y_0 + c) + d \sin \theta}{(y_0 - c) - d \sin \theta}.$$

Ta sẽ muốn đồng nhất hệ số tự do và hệ số trước $\sin \theta$ để dễ tính toán sau này. Do đó một cách chọn đơn giản là $2r - y_0 + c = d$ và $y_0 - c = d$. Suy ra $r = d$. Thu gọn phương trình ta được

$$\frac{d^2(1 + \sin \theta)}{a^2(1 - \sin \theta)} = \frac{1 + \sin \theta}{1 - \sin \theta}.$$

Như vậy $a^2 = d^2$ nên $a = d$ hoặc $a = -d$. Ta xét trường hợp $a = d$, trường hợp $a = -d$ cũng cho kết quả tương tự (không thỏa mãn).

Ta có $a = b = d = r$ và $c = y_0 - d = y_0 - r$. Phương trình đường cong trong tọa độ cực sẽ là

$$\begin{cases} x = r(\theta + \cos \theta), \\ y = (y_0 - r) + r \sin \theta. \end{cases}$$

Với $\theta = 0$ thì $(x, y) = (r, y_0 - r)$. Với $\theta = \pi/2$ thì $(x, y) = (\pi r/2, y_0)$.

Tới đây chúng ta có thể thêm bớt một hằng số để "kéo" các tọa độ về trực.

Ta đưa tọa độ khi $\theta = 0$ về Oy thì $x' = x - r$. Tương tự tọa độ khi $\theta = \pi/2$ sẽ về Ox nên $y' = y - y_0$. Như vậy tọa độ (mới) cho hai trường hợp θ là $(0, -r)$ và $(\pi r/2 - 1, 0)$ nhưng vì r là số dương (bán kính) nên $(0, -r)$ nằm dưới trục Ox , không phù hợp với hình vẽ.

Trường hợp 2. $a = -b$, ta thu được

$$\frac{d^2(1 - \sin \theta)}{a^2(1 + \sin \theta)} = \frac{(2r - y_0 + c) + d \sin \theta}{(y_0 - c) - d \sin \theta}.$$

Tương tự, để đồng nhất và rút gọn hệ số cho hợp với bên trái ta chọn $2r - y_0 + c = -d$ và $y_0 - c = -d$. Suy ra $d = -r$. Thu gọn phương trình ta được

$$\frac{d^2(1 - \sin \theta)}{a^2(1 + \sin \theta)} = \frac{1 - \sin \theta}{1 + \sin \theta}.$$

Như vậy $a^2 = d^2$ nên $a = d$ hoặc $a = -d$. Ta xét trường hợp $a = -d$.

Khi đó $a = -b = -d = r$ và $c = y_0 + d = y_0 - r$. Phương trình đường cong trong tọa độ cực sẽ là

$$\begin{cases} x = r(\theta - \cos \theta), \\ y = (y_0 - r) - r \sin \theta. \end{cases}$$

Với $\theta = 0$ thì $(x, y) = (-r, y_0)$. Với $\theta = \pi/2$ thì $(x, y) = (r(\pi/2 - 1), y_0 - 2r)$.

Tới đây ta cũng thêm bớt một hằng số vào hoành độ và tung độ để "kéo" các tọa độ về trực.

Ta đưa tọa độ khi $\theta = 0$ về Oy thì $x' = x + r$. Tương tự ta đưa tọa độ khi $\theta = \pi/2$ về Ox thì $y' = y - y_0 + 2r$. Khi đó tọa độ (mới) là $(0, 2r)$ và $(\pi r/2, 0)$. Điều này phù hợp với yêu cầu bài toán và tương đương với phương trình trong tọa độ cực

$$\begin{cases} x = r(\theta - \cos \theta) + r = r(1 + \theta - \cos \theta) \\ y = (y_0 - r) - r \sin \theta - (y_0 - 2r) = r(1 - \sin \theta) \end{cases}$$

với $0 \leq \theta \leq \frac{\pi}{2}$.

Đây chính là kết quả cần tìm. Thêm nữa vị trí ban đầu của vật là $(0, y_0)$ và tọa độ theo phương trình là $(0, 2r)$ nên suy ra $y_0 = 2r$.

Phương trình phụ thuộc thời gian

Trong phương trình đường cong có sự tham gia của bán kính r cố định và góc quét θ . Chúng ta cần mối liên hệ giữa các phương trình theo thời gian.

Nhắc lại, vận tốc tức thời tại một điểm có phương trùng với tiếp tuyến với đường cong tại điểm đó. Do đó $v = \frac{\sqrt{(dy)^2 + (dx)^2}}{dt}$ xác định vận tốc tức thời với quãng đường là $(dy)^2 + (dx)^2$ là bình phương khoảng cách trong mặt phẳng. Từ đây suy ra

$$\begin{aligned} v^2 &= \left(\frac{dy}{dt}\right)^2 + \left(\frac{dx}{dt}\right)^2 = r^2 \cos^2 \theta \left(\frac{d\theta}{dt}\right)^2 + r^2(1 + \sin \theta)^2 \left(\frac{d\theta}{dt}\right)^2 \\ &= 2r^2(1 + \sin \theta) \left(\frac{d\theta}{dt}\right)^2. \end{aligned}$$

Từ bên trên và $y_0 = 2r$ ta có

$$v^2 = 2g(y_0 - y) = 2g(y_0 - r + r \sin \theta) = 2gr(1 + \sin \theta).$$

Suy ra

$$2r^2(1 + \sin \theta) \left(\frac{d\theta}{dt}\right)^2 = 2gr(1 + \sin \theta),$$

hay

$$\left(\frac{d\theta}{dt}\right)^2 = \frac{g}{r} \Rightarrow \frac{d\theta}{dt} = \sqrt{\frac{g}{r}} = \text{const.}$$

Như vậy $\theta = \sqrt{\frac{g}{r}}t = \omega t$. Ở đây t là thời gian tính từ lúc bắt đầu thả vật từ điểm A . Cuối cùng phương trình phụ thuộc thời gian của đường cong cycloid là

$$\begin{cases} x = r(1 + \omega t - \cos \omega t), \\ y = r(1 - \sin \omega t). \end{cases}$$

Trong đó, r là bán kính cố định (bằng nửa độ cao ban đầu y_0 của vật), $\omega = \frac{g}{r}$ là tần số góc, y_0 là độ cao ban đầu của vật (tung độ điểm A).

2.3.3 Hình học affine

Không gian affine

Cho \mathcal{V} là một không gian vector trên trường \mathbb{F} và \mathcal{A} là một tập khác rỗng mà các phần tử của nó gọi là **điểm**.

Giả sử ta có ánh xạ φ sao cho:

$$\varphi : \mathcal{A} \times \mathcal{A} \rightarrow \mathcal{V}, (M, N) \mapsto \varphi(M, N)$$

thỏa hai điều kiện sau:

1. Với mọi điểm $M \in \mathcal{A}$ và vector $\vec{v} \in \mathcal{V}$, có duy nhất một điểm $N \in \mathcal{A}$ sao cho $\varphi(M, N) = \vec{v}$.
2. Với ba điểm M, N, P bất kì ta có

$$\varphi(M, N) + \varphi(N, P) = \varphi(M, P).$$

Khi đó, ta nói \mathcal{A} là một **không gian affine**. Nếu ta nói đầy đủ thì \mathcal{A} là không gian affine trên trường \mathbb{F} liên kết với không gian vector \mathcal{V} bởi ánh xạ liên kết φ .

Khi đó, \mathcal{V} được gọi là **không gian vector liên kết** với \mathcal{A} (hay **không gian nền** của \mathcal{A}) và kí hiệu là $\vec{\mathcal{A}}$.

Ánh xạ φ được gọi là **ánh xạ liên kết**. Ta kí hiệu $\varphi(M, N) = \overrightarrow{MN}$. Khi đó hai điều kiện trên được viết lại thành:

1. Với mọi điểm $M \in \mathcal{A}$ và vector $\vec{v} \in \vec{\mathcal{A}}$ thì tồn tại duy nhất một điểm $N \in \mathcal{A}$ sao cho $\overrightarrow{MN} = \vec{v}$.
2. Với ba điểm M, N, P bất kì ta có

$$\overrightarrow{MN} + \overrightarrow{NP} = \overrightarrow{MP}.$$

Biểu thức ở điều kiện 2 được gọi là **hệ thức Chales**. Hệ thức này đóng vai trò quan trọng trong các đại lượng có hướng (vector, góc định hướng, ...).

Nếu $\mathbb{F} = \mathbb{R}$ thì ta gọi là không gian affine thực.

Nếu $\mathbb{F} = \mathbb{C}$ thì ta gọi là không gian affine phức.

Nếu cần nhấn mạnh trường \mathbb{F} thì ta gọi là \mathbb{F} -không gian affine.

Kí hiệu một không gian affine là $(\mathcal{A}, \vec{\mathcal{A}}, \varphi)$.

Ta có thể ghi tắt là $\mathcal{A}(\mathbb{F})$ hoặc \mathcal{A} .

Nếu $\vec{\mathcal{A}}$ là không gian vector n chiều thì ta nói \mathcal{A} là không gian affine n chiều và kí hiệu là \mathcal{A}^n . Khi đó

$$\dim \mathcal{A} = \dim \vec{\mathcal{A}}.$$

Example 3.8

Hệ tọa độ trong không gian \mathbb{R}^3 mà chúng ta học ở phổ thông. Khi đó:

1. $\mathcal{A} = \mathbb{R}^3$ là tập hợp tất cả điểm trong không gian.
2. $\vec{\mathcal{A}}$ là tập các vector trong \mathbb{R}^3 . Về mặt hình học thì vector từ điểm A tới điểm B là mũi tên có hướng từ A tới B .

Example 3.9

Gọi \mathcal{V} là không gian vector trên trường \mathbb{F} . Ánh xạ

$$\varphi : \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{V}, (\vec{u}, \vec{v}) \mapsto \varphi(\vec{u}, \vec{v}) = \vec{v} - \vec{u}$$

thỏa mãn định nghĩa vì:

1. Với mọi vector $\vec{u} \in \mathcal{V}$ và $\vec{w} \in \mathcal{V}$, tồn tại duy nhất vector \vec{v} sao cho $\varphi(\vec{u}, \vec{v}) = \vec{w}$, tương đương với $\vec{v} - \vec{u} = \vec{w}$, hay $\vec{v} = \vec{u} + \vec{w}$.
2. Với ba vector $\vec{u}, \vec{v}, \vec{w}$ bất kì thuộc \mathcal{V} ta có

$$\varphi(\vec{u}, \vec{v}) + \varphi(\vec{v}, \vec{w}) = \vec{v} - \vec{u} + \vec{w} - \vec{v} = \vec{w} - \vec{u} = \varphi(\vec{u}, \vec{w}).$$

Ở *Ví dụ 3.9*, \mathcal{V} là không gian vector liên kết với chính nó nên ta nói φ xác định một **cấu trúc affine chính tắc** trên không gian vector \mathcal{V} , hay \mathcal{V} là **không gian affine với cấu trúc affine chính tắc**.

Tính chất của không gian affine. Với mọi điểm M, N, P, Q bất kì thuộc \mathcal{A} ta có:

1. $\overrightarrow{MN} = \vec{0}$ khi và chỉ khi $M \equiv N$.
2. $\overrightarrow{MN} = -\overrightarrow{NM}$.
3. $\overrightarrow{MN} = \overrightarrow{PQ}$ khi và chỉ khi $\overrightarrow{MP} = \overrightarrow{NQ}$.
4. $\overrightarrow{MN} = \overrightarrow{PN} - \overrightarrow{PM}$.

Đây là các công thức cơ bản được học ở phổ thông và ở đây cũng áp dụng.

Phẳng

Một số khái niệm ở phổ thông dưới dạng phẳng:

1. Điểm là 0-phẳng.
2. Đường thẳng là 1-phẳng.
3. Mặt phẳng là 2-phẳng.
4. Không gian ba chiều là 3-phẳng.

Chúng ta sẽ định nghĩa **phẳng** là mở rộng cho các khái niệm trên.

Đầu tiên chúng ta cần một vài nhận xét từ hình học phổ thông.

Trong mặt phẳng \mathbb{R}^2 , mỗi đường thẳng được xác định khi biết một điểm thuộc nó và một vector chỉ phương \vec{v} . Khi đó đường thẳng đi qua điểm M_0 và nhận \vec{v} làm vector chỉ phương là tập các điểm $M \in \mathbb{R}^2$ sao cho $\overrightarrow{MM_0} = \alpha \vec{v}$ với $\alpha \in \mathbb{R}$. Nói cách khác là vector $\overrightarrow{MM_0}$ cùng phương với vector \vec{v} .

1 Remark

Trên mặt phẳng, đường thẳng đi qua điểm M_0 và nhận \vec{v} làm vector chỉ phương là tập hợp

$$d = \{M \in \mathbb{R}^2 : \overrightarrow{MM_0} = \alpha \cdot \vec{v} \text{ với } \alpha \in \mathbb{R}\}.$$

Trong không gian \mathbb{R}^3 , tương tự, một đường thẳng xác định khi biết một điểm thuộc nó và một vector chỉ phương của nó.

1 Remark

Trong không gian \mathbb{R}^3 , đường thẳng đi qua điểm M_0 và nhận \vec{v} làm vector chỉ phương là tập hợp

$$d = \{M \in \mathbb{R}^3 : \overrightarrow{MM_0} = \alpha \cdot \vec{v} \text{ với } \alpha \in \mathbb{R}\}.$$

Chúng ta đã biết ở phô thông rằng, qua ba điểm không thẳng hàng có duy nhất một mặt phẳng đi qua. Ba điểm không thẳng hàng thì tạo thành tam giác, giả sử là M, N và P . Ta lấy hai cạnh làm hai đường thẳng cho mặt phẳng. Khi đó mặt phẳng trong không gian được xác định bởi điểm M và hai vector \overrightarrow{MN} và \overrightarrow{MP} .

1 Remark

Một mặt phẳng trong \mathbb{R}^3 xác định khi biết một điểm M_0 thuộc nó và một cặp vector \vec{u}, \vec{v} của nó. Khi đó mặt phẳng là tập hợp

$$p = \{M \in \mathbb{R}^3 : \overrightarrow{MM_0} = \alpha \cdot \vec{u} + \beta \cdot \vec{v} \text{ với } \alpha, \beta \in \mathbb{R}\}.$$

Ta định nghĩa tổng quát cho phẳng

1 Definition (Phẳng)

Cho $(\mathcal{A}, \vec{\mathcal{A}}, \varphi)$ là một không gian affine, M_0 là một điểm thuộc \mathcal{A} và $\vec{\alpha}$ là một không gian vector con của $\vec{\mathcal{A}}$. Tập hợp

$$\alpha = \{M \in \mathcal{A} : \overrightarrow{MM_0} \in \vec{\alpha}\}$$

được gọi **phẳng** (hay **plane**) đi qua M_0 với không gian chỉ phương $\vec{\alpha}$, hoặc phương $\vec{\alpha}$.

Nếu $\dim \vec{\alpha} = m$ thì ta nói α là **phẳng m chiều** hay **m -phẳng** và viết $\dim \alpha = m$. Như vậy $\dim \alpha = \dim \vec{\alpha}$.

Trước đây chúng ta gọi điểm, đường, mặt phẳng, nhưng khi số chiều cao hơn thì chúng ta không thể "chế" từ vựng mãi được. Khi đó chúng ta quy về khái niệm phẳng, 1-phẳng là đường thẳng, 2-phẳng là mặt phẳng.

Siêu phẳng (hay **hyperplane**) là tên gọi của phẳng có độ chiều là 1, tức là nếu số chiều của không gian là n thì số chiều của siêu phẳng là $n - 1$.

1 Remark

- Nếu α là phẳng đi qua điểm M thì $M \in \alpha$ và với mọi P, Q thuộc α ta có $\overrightarrow{PQ} = \overrightarrow{MQ} - \overrightarrow{MP}$ thuộc

α .

2. 0-phẳng là tập chỉ gồm một điểm. Do đó ta có thể xem một điểm là một 0-phẳng.
3. Điểm M_0 trong định nghĩa phẳng có vai trò tổng quát, nghĩa là mọi điểm M_0 trong α có ý nghĩa như nhau trong định nghĩa phẳng.
4. Giả sử α là phẳng đi qua M với phương $\vec{\alpha}$, β là phẳng đi qua N với phương $\vec{\beta}$. Khi đó $\alpha \subset \beta$ khi và chỉ khi $M \in \beta$ và $\vec{\alpha} \subset \vec{\beta}$. Suy ra $\alpha \equiv \beta$ khi và chỉ khi $P \in \beta$ (hoặc $Q \in \alpha$) và $\vec{\alpha} \equiv \vec{\beta}$.
5. Nếu α là phẳng với phương $\vec{\alpha}$ thì α là không gian affine liên kết với $\vec{\alpha}$ bởi ánh xạ liên kết

$$\varphi_{\alpha \times \alpha} : \alpha \times \alpha \rightarrow \vec{\alpha}.$$

Vì vậy ta có thể xem phẳng là không gian affine con.

Để xác định một không gian vector thì ta chỉ cần một cơ sở của nó. Do đó để xác định phương $\vec{\alpha}$ của m -phẳng α thì ta cũng chỉ cần biết một cơ sở là đủ.

Do một không gian vector có thể có nhiều cơ sở khác nhau, một m -phẳng chỉ có một không gian chỉ phương duy nhất nhưng có thể có nhiều cơ sở khác nhau.

Độc lập affine và phụ thuộc affine

❶ Definition (Độc lập và phụ thuộc affine)

Hệ $m + 1$ điểm

$$\{A_0, A_1, \dots, A_m\}$$

với $m \geq 1$ của không gian affine \mathcal{A} được gọi là **độc lập affine** nếu hệ m vector

$$\{\overrightarrow{A_0A_1}, \overrightarrow{A_0A_2}, \dots, \overrightarrow{A_0A_m}\}$$

của $\vec{\mathcal{A}}$ là một hệ vector độc lập tuyến tính.

Ngược lại, hệ điểm không độc lập affine được gọi là **phụ thuộc affine**.

1. Tập chỉ gồm một điểm A_0 được quy ước là luôn độc lập affine.
2. Trong định nghĩa điểm A_0 bình đẳng như các điểm khác vì hệ vector

$$\{\overrightarrow{A_0A_1}, \overrightarrow{A_0A_1}, \dots, \overrightarrow{A_0A_m}\}$$

độc lập affine thì hệ vector

$$\{\overrightarrow{A_iA_0}, \overrightarrow{A_iA_{i-1}}, \overrightarrow{A_iA_{i+1}}, \dots, \overrightarrow{A_iA_m}\}$$

cũng độc lập affine.

❷ Chứng minh

Xét tổ hợp tuyến tính

$$I = \lambda_1 \overrightarrow{A_0A_1} + \lambda_2 \overrightarrow{A_0A_2} + \dots + \lambda_m \overrightarrow{A_0A_m} = \vec{0}.$$

Do hệ vector độc lập tuyến tính nên $\lambda_1 = \lambda_2 = \dots = \lambda_m = 0$. Khi đó, biến đổi I ta có

$$\begin{aligned} I &= \lambda_1(\overrightarrow{A_i A_1} - \overrightarrow{A_i A_0}) + \lambda_2(\overrightarrow{A_i A_2} - \overrightarrow{A_i A_0}) + \dots + \lambda_{i-1}(\overrightarrow{A_i A_{i-1}} - \overrightarrow{A_i A_0}) \\ &\quad + \lambda_i \overrightarrow{A_0 A_i} + \lambda_{i+1}(\overrightarrow{A_i A_{i+1}} - \overrightarrow{A_i A_0}) + \dots + \lambda_m(\overrightarrow{A_i A_m} - \overrightarrow{A_i A_0}) \\ &= \lambda_1 \overrightarrow{A_i A_1} + \lambda_2 \overrightarrow{A_i A_2} + \dots + \lambda_m \overrightarrow{A_i A_m} \\ &\quad - \underbrace{(\lambda_1 + \dots + \lambda_m) \overrightarrow{A_i A_0}}_{\vec{0}} \\ &= \lambda_1 \overrightarrow{A_i A_1} + \lambda_2 \overrightarrow{A_i A_2} + \dots + \lambda_m \overrightarrow{A_i A_m} = \vec{0} \end{aligned}$$

với $\lambda_1 = \dots = \lambda_m = 0$. Như vậy hệ vector mới cũng độc lập tuyến tính.

3. Hệ điểm $\{A_0, \dots, A_m\}$ phụ thuộc affine khi và chỉ khi hệ vector

$$\{\overrightarrow{A_0 A_1}, \dots, \overrightarrow{A_0 A_m}\}$$

phụ thuộc tuyến tính.

4. Hệ con của một hệ độc lập thì độc lập, nhưng hệ con của một hệ phụ thuộc chưa chắc phụ thuộc.

Theorem

Trong không gian affine n chiều \mathcal{A}^n , với $0 < m \leq n + 1$ thì luôn tồn tại các hệ m điểm độc lập. Mọi hệ gồm hơn $n + 1$ điểm đều phụ thuộc.

Giao của các phẳng. Bao affine

Cho $\{\alpha_i : i \in I\}$ là một họ không rỗng các phẳng trong không gian affine \mathcal{A} .

Theorem

Nếu $\bigcap_{i \in I} \alpha_i \neq \emptyset$ thì $\bigcap_{i \in I} \alpha_i$ là một phẳng có phương là $\bigcap_{i \in I} \vec{\alpha}_i$.

Chứng minh

Vì $\bigcap_{i \in I} \alpha_i \neq \emptyset$ nên tồn tại điểm M thuộc $\bigcap_{i \in I} \alpha_i$, như vậy $M \in \alpha_i$ với mọi $i \in I$.

Nếu ta có điểm $N \in \bigcap_{i \in I} \alpha_i$ thì $N \in \alpha_i$ với mọi $i \in I$. Suy ra $\overrightarrow{MN} \in \vec{\alpha}_i$ với mọi $i \in I$. Từ đó

$$\overrightarrow{MN} \in \bigcap_{i \in I} \vec{\alpha}_i \iff \bigcap_{i \in I} \alpha_i = \{N \in \mathcal{A} : \overrightarrow{MN} \in \bigcap_{i \in I} \vec{\alpha}_i\},$$

nghĩa là $\bigcap_{i \in I} \alpha_i$ là phẳng đi qua M với không gian chỉ phương là $\bigcap_{i \in I} \vec{\alpha}_i$.

Definition (Phẳng giao)

Phẳng $\bigcap_{i \in I} \alpha_i$ trong định lí trên được gọi là **phẳng giao** của các phẳng α_i .

Từ định nghĩa ta thấy rằng $\bigcap_{i \in I} \alpha_i$ là phẳng lớn nhất (theo nghĩa quan hệ bao hàm) chứa trong tất cả các phẳng α_i với $i \in I$.

❶ Definition

Cho X là một tập con khác rỗng của không gian affine \mathcal{A} . Khi đó giao của mọi phẳng chứa X trong \mathcal{A} sẽ là một phẳng, gọi là **bao affine** của X . Kí hiệu là $\langle X \rangle$.

Bao affine $\langle X \rangle$, theo quan hệ bao hàm, của tập X là phẳng bé nhất chứa X .

❶ Definition (Phẳng tổng)

Cho $\{\alpha_i : i \in I\}$ là một họ không rỗng các phẳng. Bao affine của tập hợp $\bigcup_{i \in I} \alpha_i$ được gọi là **phẳng tổng** (hay **tổng**) của các phẳng α_i . Kí hiệu là $\sum_{i \in I} \alpha_i$.

Phẳng tổng là phẳng bé nhất (có số chiều nhỏ nhất) chứa tất cả α_i với $i \in I$. Khi I là tập hữu hạn, giả sử $I = \{1, 2, \dots, n\}$ thì ta viết

$$\alpha_1 + \alpha_2 + \dots + \alpha_n \text{ hay } \sum_{i=1}^n \alpha_i$$

để biểu diễn tổng của các phẳng α_i .

❶ Remark

Nếu X là một hệ hữu hạn điểm

$$X = \{M_0, M_1, \dots, M_n\}$$

thì tổng $M_0 + \dots + M_n$ là phẳng có số chiều nhỏ nhất đi qua các điểm này. Ở đây ta xem mỗi điểm M_i là 0-phẳng. Hơn nữa

$$\dim(M_0 + M_1 + \dots + M_n) = \text{rank}(\overrightarrow{M_0 M_1}, \dots, \overrightarrow{M_0 M_n}).$$

Do đó nếu hệ điểm $\{M_0, \dots, M_n\}$ độc lập thì

$$\dim(M_0 + M_1 + \dots + M_n) = n.$$

❶ Theorem

Cho α và β là hai phẳng. Nếu $\alpha \cap \beta \neq \emptyset$ thì với mọi điểm M thuộc α và với mọi điểm N thuộc β ta có $\overrightarrow{MN} = \vec{\alpha} + \vec{\beta}$.

Ngược lại, nếu ta có điểm $M \in \alpha$ và điểm $N \in \beta$ sao cho $\overrightarrow{MN} = \vec{\alpha} + \vec{\beta}$ thì $\alpha \cap \beta \neq \emptyset$.

❶ Chứng minh

Giả sử $\alpha \cap \beta \neq \emptyset$. Khi đó tồn tại điểm $P \in \alpha \cap \beta$. Với mọi điểm $M \in \alpha$ ta có $\overrightarrow{MP} \in \vec{\alpha}$, tương tự với mọi điểm $N \in \beta$ ta có $\overrightarrow{PN} \in \vec{\beta}$. Suy ra

$$\overrightarrow{MP} + \overrightarrow{PN} = \overrightarrow{MN} = \vec{\alpha} + \vec{\beta}.$$

Ngược lại, giả sử có điểm $M \in \alpha$ và điểm $N \in \beta$ sao cho $\overrightarrow{MN} = \vec{\alpha} + \vec{\beta}$. Khi đó $\overrightarrow{MN} = \vec{u} + \vec{v}$ với $\vec{u} \in \vec{\alpha}$ và $\vec{v} \in \vec{\beta}$. Điều này xảy ra nếu tồn tại duy nhất điểm $P \in \alpha$ sao cho $\overrightarrow{MP} = \vec{u}$, tương tự tồn tại duy nhất điểm $Q \in \beta$ sao cho $\overrightarrow{QN} = \vec{v}$.

Khi đó, vì

$$\overrightarrow{MN} = \vec{u} + \vec{v} = \overrightarrow{MP} + \overrightarrow{QN} = \overrightarrow{MP} - \overrightarrow{NQ}.$$

Chuyển về và đổi dấu ta có

$$\overrightarrow{MP} = \overrightarrow{MN} + \overrightarrow{NQ} = \overrightarrow{MQ}$$

nên $P \equiv Q$. Như vậy $\alpha \cap \beta \neq \emptyset$ vì có ít nhất một điểm $P \equiv Q$ thuộc giao của hai phẳng.

❷ Theorem

Giả sử α và β là hai phẳng với phương lần lượt là $\vec{\alpha}$ và $\vec{\beta}$. Khi đó:

1. Nếu $\alpha \cap \beta \neq \emptyset$ thì

$$\dim(\alpha + \beta) = \dim \alpha + \dim \beta = \dim(\alpha \cap \beta).$$

2. Nếu $\alpha \cap \beta = \emptyset$ thì

$$\dim(\alpha + \beta) = \dim \alpha + \dim \beta = \dim(\vec{\alpha} + \vec{\beta}) + 1.$$

❸ Chứng minh công thức 1

Nếu $\alpha \cap \beta \neq \emptyset$ thì theo Định lí 3.2 ta có $\alpha \cap \beta$ là một phẳng có phương $\vec{\alpha} \cap \vec{\beta}$.

Lấy điểm $M \in \alpha \cap \beta$ và gọi γ là phẳng đi qua M với phương $\vec{\gamma} = \vec{\alpha} + \vec{\beta}$. Ta có $\alpha \subset \gamma$ và $\beta \subset \gamma$.

Ngoài ra nếu có phẳng γ' chứa α và β thì $M \in \gamma'$ và phương của γ' phải chứa $\vec{\alpha}$ và $\vec{\beta}$. Nói cách khác ta có $\gamma \subset \gamma'$.

Như vậy γ là phẳng nhỏ nhất chứa α và β , tức là $\gamma = \alpha + \beta$. Do đó

$$\begin{aligned}\dim(\alpha + \beta) &= \dim \gamma = \dim \vec{\gamma} \\ &= \dim(\vec{\alpha} + \vec{\beta}) = \dim \vec{\alpha} + \dim \vec{\beta} \\ &= \dim \alpha + \dim \beta \stackrel{?}{=} \dim(\alpha \cap \beta).\end{aligned}$$

➊ [TODO] Chứng minh công thức 2

[TODO]

Vị trí tương đối**➊ Definition (Vị trí tương đối giữa hai phẳng)**

Hai phẳng α và β được gọi là **cắt nhau** cấp r nếu $\alpha \cap \beta$ là một r -phẳng.

Hai phẳng α và β được gọi là **chéo nhau** cấp r nếu $\alpha \cap \beta = \emptyset$ và $\dim(\vec{\alpha} \cap \vec{\beta}) = r$.

Hai phẳng α và β được gọi là **song song** (với nhau) nếu $\vec{\alpha} \subset \vec{\beta}$ hoặc $\vec{\beta} \subset \vec{\alpha}$.

➊ Example

Xét không gian ba chiều \mathbb{R}^3 .

1. Hai đường thẳng "cắt nhau" là hai 1-phẳng cắt nhau cấp 0 (tại một điểm). Tổng của chúng là mặt phẳng duy nhất xác định bởi hai đường thẳng đó.
2. Hai mặt phẳng "cắt nhau" là hai 2-phẳng cắt nhau cấp 1 (đường thẳng chung). Tổng của chúng là toàn bộ \mathbb{R}^3 .

Theo *Định lí 3.4*, trong \mathbb{R}^3 không tồn tại hai mặt phẳng chéo nhau cấp 0 hoặc 1.

➊ Theorem

Cho hai phẳng song song α và β . Nếu $\alpha \cap \beta \neq \emptyset$ thì $\alpha \subset \beta$ hoặc $\beta \subset \alpha$.

➊ Chứng minh

Do α và β có điểm chung nên theo *Định lí 3.2* $\alpha \cap \beta$ là một phẳng có phương $\vec{\alpha} \cap \vec{\beta}$.

Do α song song β nên $\vec{\alpha} \subset \vec{\beta}$ hoặc $\vec{\beta} \subset \vec{\alpha}$.

Nếu $\vec{\alpha} \subset \vec{\beta}$ thì $\vec{\alpha} \cap \vec{\beta} = \vec{\alpha}$, suy ra $\alpha \cap \beta = \alpha$, hay $\alpha \subset \beta$.

Tương tự, nếu $\vec{\beta} \subset \vec{\alpha}$ thì $\beta \subset \alpha$.

➊ Theorem

Qua một điểm M có một và chỉ một m -phẳng song song với m -phẳng α đã cho.

➊ Chứng minh

Gọi β là m -phẳng đi qua điểm M với phương $\vec{\alpha}$.

Khi đó β là phẳng m chiềng song song với α . Nếu β' cũng là m -phẳng đi qua A và song song với α thì suy ra $\beta = \beta'$ (và cũng bằng α).

Do β và β' có điểm chung nên theo Định lí 3.5 ta có $\beta \equiv \beta'$.

i Theorem

Trong không gian affine n chiều \mathcal{A}^n cho một siêu phẳng α và một m -phẳng β với $1 \leq m \leq n - 1$. Khi đó α và β hoặc song song, hoặc cắt nhau theo một $(m - 1)$ -phẳng.

Mục tiêu và tọa độ affine

i Definition (Mục tiêu affine)

Cho \mathcal{A}^n là một không gian affine n chiều. Hết

$$\{O, \vec{e}_1, \vec{e}_2, \dots, \vec{e}_n\}$$

gồm một điểm $O \in \mathcal{A}^n$ và một cơ sở $\{\vec{e}_1, \dots, \vec{e}_n\}$ của \mathcal{A}^n được gọi là **mục tiêu affine** (hay **mục tiêu**) của \mathcal{A}^n .

Điểm O được gọi là **gốc**, các vector \vec{e}_i được gọi là **vector cơ sở thứ i** với $i = 1, 2, \dots, n$.

Giả sử $\{O, \vec{e}_i\}$ là một mục tiêu của không gian affine \mathcal{A}^n .

Khi đó với mọi điểm $M \in \mathcal{A}^n$, vector $\overrightarrow{OM} \in \mathcal{A}^n$ nên ta có biểu diễn tuyến tính của \overrightarrow{OM} qua cơ sở $\{\vec{e}_i\}$:

$$\overrightarrow{OM} = \sum_{i=1}^n x_i \vec{e}_i.$$

Khi đó, nhắc lại kiến thức đại số tuyến tính, vector \overrightarrow{OM} có tọa độ (x_1, \dots, x_n) đối với cơ sở $\{\vec{e}_i\}$, x_i là các phần tử thuộc \mathbb{F} với $i = 1, \dots, n$.

Bộ (x_1, \dots, x_n) được gọi là **tọa độ** của M trong mục tiêu $\{O, \vec{e}_i\}$, và x_i được gọi là **tọa độ thứ i** . Kí hiệu là $M(x_i)$.

Giả sử M có tọa độ (x_i) và N có tọa độ (y_i) thì đối với mục tiêu $\{O, \vec{e}_i\}$ ta có

$$\overrightarrow{MN} = \overrightarrow{ON} - \overrightarrow{OM} = (y_i - x_i).$$

Đây là tọa độ của \overrightarrow{MN} trong mục tiêu $\{O, \vec{e}_i\}$.

i Remark

Giả sử trên \mathcal{A}^n đã chọn được mục tiêu cố định $\{O, \vec{e}_i\}$. Xét ánh xạ

$$\varphi : A \rightarrow \mathbb{F}^n, M \mapsto (x_i)$$

với (x_i) là tọa độ của M trong mục tiêu. Khi đó φ là song ánh và mỗi điểm được đồng nhất với một phần tử của \mathbb{F}^n . Lúc này chúng ta đã đại số hóa hình học.

Remark

Xét mục tiêu affine $\{O, \vec{e}_i\}$ của \mathcal{A}^n và gọi $E_i \in \mathcal{A}$ là các điểm sao cho $\overrightarrow{OE_i} = \vec{e}_i$.

Khi đó hệ điểm $\{O, E_1, \dots, E_n\}$ độc lập affine. Ngược lại, một hệ điểm gồm $n+1$ độc lập $\{O, E_1, \dots, E_n\}$ độc lập affine xác định một mục tiêu affine $\{O, \vec{e}_i\}$ với $\vec{e}_i = \overrightarrow{OE_i}$.

Theo định nghĩa ta có $O = (0, \dots, 0)$ và $E_i = (0, \dots, 0, 1, 0, \dots, 0)$ với số 1 ở vị trí i .

Remark

Siêu phẳng đi qua n điểm độc lập $O, E_1, \dots, E_{i-1}, E_{i+1}, \dots, E_n$ được gọi là **siêu phẳng tọa độ thứ i** .
Để thấy M thuộc siêu phẳng tọa độ thứ i khi và chỉ khi $x_i = 0$ với x_i là tọa độ thứ i của M .

Công thức đổi mục tiêu

Giả sử trong không gian affine \mathcal{A}^n có hai mục tiêu $\{O, \vec{e}_i\}$ và $\{O', \vec{e}'_i\}$.

Mỗi điểm M sẽ có tọa độ khác nhau ứng với mỗi mục tiêu (x_i) và (x'_i) . Ta cần tìm mối liên hệ giữa chúng.

Do \vec{e}_i là cơ sở của không gian vector nên mọi vector trong không gian vector có thể biểu diễn dưới dạng tổ hợp tuyến tính của các vector trong cơ sở, như vậy

$$\overrightarrow{OO'} = \sum_{i=1}^n b_i \vec{e}_i$$

với $b_i \in \mathbb{F}$.

Biểu diễn các vector trong cơ sở \vec{e}'_i bởi tổ hợp tuyến tính các vector trong cơ sở \vec{e}_i :

$$\begin{cases} \vec{e}'_1 = c_{11}\vec{e}_1 + c_{12}\vec{e}_2 + \dots + c_{1n}\vec{e}_n \\ \vec{e}'_2 = c_{21}\vec{e}_1 + c_{22}\vec{e}_2 + \dots + c_{2n}\vec{e}_n \\ \vdots \\ \vec{e}'_n = c_{n1}\vec{e}_1 + c_{n2}\vec{e}_2 + \dots + c_{nn}\vec{e}_n \end{cases}$$

hay viết ngắn gọn là

$$\vec{e}'_i = \sum_{j=1}^n c_{ij} \vec{e}_j$$

với $i = 1, 2, \dots, n$.

Điểm M có tọa độ trong hai mục tiêu trên lần lượt là (x_i) và (x'_i) , nghĩa là

$$\overrightarrow{OM} = \sum_{i=1}^n x_i \vec{e}_i, \quad \overrightarrow{O'M} = \sum_{i=1}^n x'_i \vec{e}'_i.$$

Ta có

$$\begin{aligned}
 \sum_{i=1}^n x_i \vec{e}_i &= \overrightarrow{OM} = \overrightarrow{OO'} + \overrightarrow{O'M} \\
 &= \sum_{i=1}^n b_i \vec{e}_i + \sum_{i=1}^n x'_i \vec{e}'_i \\
 &= \sum_{i=1}^n b_i \vec{e}_i + \sum_{i=1}^n x'_i \sum_{j=1}^n c_{ij} \vec{e}_j \\
 &= \sum_{i=1}^n b_i \vec{e}_i + \sum_{i=1}^n x'_i \sum_{j=1}^n c_{ji} x'_j \\
 &= \sum_{i=1}^n \left(\sum_{j=1}^n c_{ji} x'_j + b_i \right) \vec{e}_i.
 \end{aligned}$$

Ở đây mình biến đổi

$$\sum_{i=1}^n x'_i \sum_{j=1}^n c_{ij} \vec{e}_j = \sum_{i=1}^n \sum_{j=1}^n x'_i c_{ij} \vec{e}_j = \sum_{j=1}^n \vec{e}_j \sum_{i=1}^n c_{ij} x'_i,$$

và để đồng bộ với tổng phía trước là $\sum_{i=1}^n b_i \vec{e}_i$ thì mình đổi j thành i và j thành i nên sẽ có

$$\sum_{i=1}^n x'_i \sum_{j=1}^n c_{ij} \vec{e}_j = \sum_{i=1}^n \vec{e}_i \sum_{j=1}^n c_{ji} x'_j.$$

Khi đó

$$x_i = \sum_{j=1}^n c_{ji} x'_j + b_i$$

với $i = 1, 2, \dots, n$. Điều này tương đương với

$$\begin{aligned}
 x_1 &= c_{11} x'_1 + c_{21} x'_2 + \dots + c_{n1} x'_n + b_1 \\
 x_2 &= c_{12} x'_1 + c_{22} x'_2 + \dots + c_{n2} x'_n + b_2 \\
 &\dots \\
 x_n &= c_{1n} x'_1 + c_{2n} x'_2 + \dots + c_{nn} x'_n + b_n
 \end{aligned}$$

Chúng ta có thể viết dưới dạng ma trận là

$$(x_1, \dots, x_n) = (x'_1, \dots, x'_n) \cdot C + (b_1, \dots, b_n)$$

với C là ma trận

$$C = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \ddots & \ddots & \ddots & \ddots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix}.$$

Hệ phương trình tuyến tính có nghiệm khi $\det C \neq 0$.

Các công thức trên được gọi là công thức đổi tọa độ (hay công thức đổi mục tiêu) và ma trận đổi tọa độ C từ mục tiêu $\{O, \vec{e}_i\}$ sang $\{O', \vec{e}'_i\}$.

Phương trình của m -phẳng

Phương trình tham số

Cho \mathcal{A}^n là một không gian affine n chiều với mục tiêu $\{O, \vec{e}_i\}$ cho trước, và α là một m -phẳng đi qua điểm M với phương $\vec{\alpha}$ sao cho $0 < m < n$.

Giả sử

$$\overrightarrow{OP} = \sum_{i=1}^n b_i \vec{e}_i$$

và $\{\vec{\alpha}_1, \dots, \vec{\alpha}_m\}$ là một cơ sở của $\vec{\alpha}$ với

$$\vec{\alpha}_j = \sum_{i=1}^n a_{ij} \vec{e}_i, \quad j = 1, \dots, m.$$

Điểm P có tọa độ (x_i) đối với mục tiêu $\{O, \vec{e}_i\}$ thuộc α khi và chỉ khi $\overrightarrow{MP} \in \vec{\alpha}$, tức là có các số $t_j \in \mathbb{F}$ sao cho

$$\overrightarrow{MP} = \sum_{j=1}^m t_j \vec{\alpha}_j.$$

Ta có

$$\overrightarrow{MP} = \overrightarrow{OP} - \overrightarrow{OM} = \sum_{i=1}^n x_i \vec{e}_i - \sum_{i=1}^n b_i \vec{e}_i = \sum_{i=1}^n (x_i - b_i) \vec{e}_i.$$

Ta cũng có

$$\begin{aligned} \overrightarrow{MP} &= \sum_{j=1}^m t_j \vec{\alpha}_j = \sum_{j=1}^m t_j \sum_{i=1}^n a_{ij} \vec{e}_i \\ &= \sum_{i=1}^n \left(\sum_{j=1}^m t_j a_{ij} \right) \vec{e}_i \\ &\Rightarrow x_i = \sum_{j=1}^m t_j a_{ij} + b_i. \end{aligned}$$

Hệ phương trình trở thành

$$\begin{cases} x_1 = a_{11}t_1 + a_{12}t_2 + \dots + a_{1m}t_m + b_1 \\ x_2 = a_{21}t_1 + a_{22}t_2 + \dots + a_{2m}t_m + b_2 \\ \vdots \\ x_n = a_{n1}t_1 + a_{n2}t_2 + \dots + a_{nm}t_m + b_n \end{cases}$$

hay dưới dạng ma trận là

$$(x_1, \dots, x_n) = A \cdot (t_1, \dots, t_m) + (b_1, \dots, b_n).$$

Khi đó ta có thể viết phương trình tham số dưới dạng vector

$$\vec{x} = t_1 \vec{\alpha}_1 + t_2 \vec{\alpha}_2 + \dots + t_m \vec{\alpha}_m + \vec{b}.$$

Các công thức trên tương đương nhau và được gọi là phương trình tham số của m -phẳng α , còn các phần tử t_j với $j = 1, \dots, m$ được gọi là các tham số.

[TODO] Phương trình tổng quát

Tâm tỉ cự và tỉ số đơn

Cho họ điểm

$$\{M_1, M_2, \dots, M_n\} \subset \mathcal{A}^n$$

và họ hệ số $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ với $\lambda \in \mathbb{F}$ thỏa điều kiện

$$\lambda = \lambda_1 + \lambda_2 + \dots + \lambda_n \neq 0.$$

Khi đó với điểm O tùy ý của \mathcal{A}^n ta có

$$\frac{1}{\lambda} (\lambda_1 \overrightarrow{OM_1} + \lambda_2 \overrightarrow{OM_2} + \dots + \lambda_n \overrightarrow{OM_n})$$

là một vector xác định của \mathcal{A}^n . Do đó tồn tại duy nhất một điểm $G \in \mathcal{A}^n$ sao cho

$$\overrightarrow{OG} = \frac{1}{\lambda} (\lambda_1 \overrightarrow{OM_1} + \lambda_2 \overrightarrow{OM_2} + \dots + \lambda_n \overrightarrow{OM_n}).$$

Khi đó điểm G được gọi **tâm tỉ cự** của họ điểm $\{M_1, M_2, \dots, M_n\}$ gắn với hệ số $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$.

i Theorem

Điểm G là tâm tỉ cự của họ điểm $\{M_1, M_2, \dots, M_n\}$ gắn với hệ số $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ khi và chỉ khi G thỏa mãn hệ thức

$$\lambda_1 \overrightarrow{GM_1} + \lambda_2 \overrightarrow{GM_2} + \dots + \lambda_n \overrightarrow{GM_n} = \vec{0}.$$

i Chứng minh

Từ định nghĩa tâm tỉ cự ta có

$$\begin{aligned} \overrightarrow{OG} &= \frac{1}{\lambda} (\lambda_1 \overrightarrow{OM_1} + \lambda_2 \overrightarrow{OM_2} + \dots + \lambda_n \overrightarrow{OM_n}) \\ &= \frac{1}{\lambda} [\lambda_1 (\overrightarrow{OG} + \overrightarrow{GM_1}) + \lambda_2 (\overrightarrow{OG} + \overrightarrow{GM_2}) + \dots + \lambda_n (\overrightarrow{OG} + \overrightarrow{GM_n})] \\ &= \frac{\lambda_1 + \lambda_2 + \dots + \lambda_n}{\lambda} \overrightarrow{OG} + \frac{1}{\lambda} (\lambda_1 \overrightarrow{GM_1} + \overrightarrow{GM_2} + \dots + \overrightarrow{GM_n}), \end{aligned}$$

mà $\lambda_1 + \lambda_2 + \dots + \lambda_n = \lambda$ nên tối giản \overrightarrow{OG} hai vế ta có điều phải chứng minh.

i Remark

Từ định lí trên ta cũng chứng minh được hai hệ quả sau:

1. Tâm tỉ cự không phụ thuộc điểm O được chọn mà phụ thuộc họ điểm $\{M_1, M_2, \dots, M_n\}$ và họ hệ số $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$.
2. Khi thay họ hệ số $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ bởi $\{k\lambda_1, k\lambda_2, \dots, k\lambda_n\}$ với $k \neq 0$ thì tâm tỉ cự không thay đổi.

Definition

Tâm tỉ cự G của hệ điểm $\{M_1, M_2, \dots, M_n\}$ gắn với họ hệ số $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ mà

$$\lambda_1 = \lambda_2 = \dots = \lambda_n = 1$$

thì G được gọi là **trọng tâm** của hệ điểm.

Khi đó, với mọi điểm $O \in \mathcal{A}^n$ ta có

$$\overrightarrow{OG} = \frac{1}{n} \sum_{i=1}^n \overrightarrow{OM_i},$$

hay

$$\sum_{i=1}^n \overrightarrow{GM_i} = \vec{0}.$$

Remark

Mọi hệ m điểm có trọng tâm khi và chỉ khi **đặc số** (characteristic) của trường \mathbb{F} phải khác m .

Khi $\mathbb{F} = \mathbb{R}$ hoặc \mathbb{C} thì mọi hệ hữu hạn điểm đều có trọng tâm.

Theorem

Tập hợp tất cả các tâm tỉ cự với họ các hệ số khác nhau của hệ điểm $\{M_1, \dots, M_n\}$ trong không gian affine \mathcal{A}^n chính là phẳng

$$\alpha = M_1 + M_2 + \dots + M_n.$$

Ánh xạ affine

Definition

Cho hai không gian affine \mathcal{A} và \mathcal{A}' trên cùng trường \mathbb{F} và ánh xạ $\varphi : \mathcal{A} \rightarrow \mathcal{A}'$ sao cho với mọi $M, N \in \mathcal{A}$ ta có

$$\overrightarrow{f(M)f(N)} = \varphi(\overrightarrow{MN})$$

thì f được gọi là **ánh xạ affine** liên kết với φ .

Ánh xạ φ được gọi là **ánh xạ tuyến tính liên kết** hay **ánh xạ nền** của ánh xạ affine f .

Theo định nghĩa, mỗi ánh xạ affine chỉ có một ánh xạ tuyến tính liên kết. Tuy nhiên một ánh xạ tuyến tính có thể liên kết với nhiều ánh xạ affine.

❶ Example

Ánh xạ đồng nhất $\text{Id}_{\mathcal{A}} : \mathcal{A} \rightarrow \mathcal{A}$, $f(M) = M$ với mọi $M \in \mathcal{A}$ của không gian affine là một ánh xạ affine. Ánh xạ tuyến tính liên kết với $\text{Id}_{\mathcal{A}}$ chính là ánh xạ đồng nhất $\text{Id}_{\vec{\mathcal{A}}}$ của $\vec{\mathcal{A}}$.

❶ Example

Ánh xạ hằng $f : \mathcal{A} \rightarrow \mathcal{A}'$ biến mọi điểm của \mathcal{A} thành một điểm cố định nào đó của \mathcal{A}' là một ánh xạ affine. Ánh xạ tuyến tính liên kết \vec{f} của f là ánh xạ không \mathcal{O} , biến mọi vector thành vector $\vec{0}$.

❶ Example

Phép chiếu song song: trong không gian affine n chiều \mathcal{A}^n cho m -phẳng α và $(n-m)$ -phẳng β sao cho $\vec{\alpha} \cap \vec{\beta} = \{\vec{0}\}$.

Dựa vào định lí về số chiều của phẳng tổng ta chứng minh được rằng $\alpha \cap \beta \neq \emptyset$. Do $\vec{\alpha} \cap \vec{\beta} = \{\vec{0}\}$ ta suy ra $\alpha \cap \beta$ là 0-phẳng, tức là giao của α và β là tập chỉ có một điểm.

Giả sử M là một điểm bất kì của \mathcal{A}^n . Gọi α' là m -phẳng đi qua M và song song α , gọi β' là $(n-m)$ -phẳng đi qua M và song song với β . Theo lập luận trên, α' và β giao nhau tại một điểm duy nhất là M_β , tương tự β' cắt α tại một điểm duy nhất là M_α . Các ánh xạ

$$\rho_\alpha : \mathcal{A}^n \rightarrow \alpha, M \mapsto M_\alpha$$

và

$$\rho_\beta : \mathcal{A}^n \rightarrow \beta, M \mapsto M_\beta$$

lần lượt được gọi là **phép chiếu song song** lên phẳng α theo phương β và phép chiếu song song lên phẳng β theo phương α .

Ta gọi α là cơ sở và β là phương chiếu của phép chiếu ρ_α .

Ta gọi β là cơ sở và α là phương chiếu của phép chiếu ρ_β .

Ta sẽ chứng minh ρ_α là ánh xạ affine.

❷ Chứng minh

Từ giả thiết $\vec{\alpha} \cup \vec{\beta} = \vec{\mathcal{A}}$, gọi $\rho_{\vec{\alpha}}$ là phép chiếu lên thành phần thứ nhất

$$\rho_{\vec{\alpha}} : \vec{\mathcal{A}} \rightarrow \vec{\alpha}.$$

Với mọi $M, N \in \mathcal{A}$ ta có

$$\overrightarrow{\rho_\alpha(M)\rho_\alpha(N)} = \overrightarrow{M_\alpha N_\alpha}.$$

Hơn nữa

$$\rho_{\vec{\alpha}}(\overrightarrow{MN}) = \rho_{\vec{\alpha}}(\overrightarrow{MM_\alpha}) + \rho_{\vec{\alpha}}(\overrightarrow{M_\alpha N_\alpha}) + \rho_{\vec{\alpha}}(\overrightarrow{N_\alpha N}) = \overrightarrow{M_\alpha N_\alpha}.$$

Từ đây suy ra ρ_α là ánh xạ liên kết với $\rho_{\bar{\alpha}}$. Để thấy (?) ρ_α và ρ_β có tính chất

$$\rho_\alpha^2 = \rho_\alpha, \quad \rho_\beta^2 = \rho_\beta, \quad \rho_\alpha \rho_\beta = \rho_\beta \rho_\alpha = h$$

với h là ánh xạ hằng, biến mọi điểm thành giao điểm của α và β .

1 Definition

Nếu ánh xạ affine f là một đơn ánh thì ta nói f là **đơn cấu affine**. Tương tự cho **toàn cấu affine** và **đảng cấu affine**.

Khi $\mathcal{A} = \mathcal{A}'$, tức là $f : \mathcal{A} \rightarrow \mathcal{A}$ thì ta nói f là một **tự đồng cấu affine** (hay **automorphism**) của \mathcal{A} .

Nếu có một đẳng cấu affine từ \mathcal{A} vào \mathcal{A}' thì ta nói \mathcal{A} và \mathcal{A}' đẳng cấu affine với nhau và kí hiệu là $\mathcal{A} \cong \mathcal{A}'$.

Quan hệ đẳng cấu giữa các không gian affine là quan hệ tương đương và hai không gian affine đẳng cấu khi và chỉ khi chúng có cùng số chiều.

1 Theorem

Cho $f : \mathcal{A} \rightarrow \mathcal{A}'$ là ánh xạ affine. Khi đó

1. Nếu f là đơn cấu, toàn cấu hoặc đẳng cấu affine thì ánh xạ liên kết:math: \vec{f}

theo thứ tự là đơn cấu, toàn cấu và đẳng cấu tuyến tính.

2. Nếu f là đẳng cấu affine thì ánh xạ ngược $f^{-1} : \mathcal{A}' \rightarrow \mathcal{A}$ cũng là đẳng cấu affine và $\overrightarrow{f^{-1}} = (\vec{f})^{-1}$.

1 Theorem

Nếu $f : \mathcal{A} \rightarrow \mathcal{A}'$ và $g : \mathcal{A}' \rightarrow \mathcal{A}''$ là các ánh xạ affine lần lượt liên kết với các ánh xạ tuyến tính \vec{f} và \vec{g} thì $g \circ f$ là ánh xạ affine liên kết với $\vec{g} \circ \vec{f}$, tức là

$$\overrightarrow{g \circ f} = \vec{g} \circ \vec{f}.$$

1 Theorem

Cho $f : \mathcal{A} \rightarrow \mathcal{A}'$ là ánh xạ affine liên kết với ánh xạ tuyến tính \vec{f} . Khi đó

1. Nếu α là một phẳng trong \mathcal{A} với phương $\vec{\alpha}$ thì $f(\alpha)$ là một phẳng trong \mathcal{A}' với phương $\vec{f}(\vec{\alpha})$ và $\dim f(\alpha) \geq \dim \alpha$. Trong trường hợp f là đơn cấu thì $\dim f(\alpha) = \dim \alpha$.
2. Giả sử α' là một phẳng trong \mathcal{A}' với phương $\vec{\alpha}'$. Nếu $f^{-1}(\alpha')$ khác rỗng thì $f^{-1}(\alpha')$ là một phương trong \mathcal{A} với phương $\overrightarrow{f^{-1}}(\vec{\alpha}')$.

Sự xác định ánh xạ affine

Ánh xạ tuyến tính hoàn toàn xác định nếu biết được ảnh của một cơ sở. Ánh xạ affine cũng vậy, nó hoàn toàn xác định nếu biết được ảnh của một mục tiêu.

Theorem

Cho \mathcal{A} và \mathcal{A}' là các \mathbb{F} -không gian affine, $\varphi : \vec{\mathcal{A}} \rightarrow \vec{\mathcal{A}'}$ là ánh xạ tuyến tính, $M \in \mathcal{A}$ và $M' \in \mathcal{A}'$. Khi đó tồn tại duy nhất một ánh xạ affine $f : \mathcal{A} \rightarrow \mathcal{A}'$ sao cho $f(M) = M'$ và $\vec{f} = \varphi$.

Nói cách khác, ánh xạ affine hoàn toàn xác định khi biết ánh xạ tuyến tính liên kết và một cặp điểm tương ứng.

Remark

Nếu $\dim \mathcal{A} = \dim \mathcal{A}'$ và φ là đẳng cấu tuyến tính thì f là một đẳng cấu affine.

Corollary

Cho \mathcal{A} và \mathcal{A}' là hai \mathbb{F} -không gian affine, $\{O, \vec{e}_1, \dots, \vec{e}_n\}$ là mục tiêu của \mathcal{A} , $O' \in \mathcal{A}'$ và $\{\vec{e}'_1, \dots, \vec{e}'_n\}$ là một hệ vector trong $\vec{\mathcal{A}'}$. Khi đó tồn tại duy nhất một ánh xạ affine $f : \mathcal{A} \rightarrow \mathcal{A}'$ sao cho

$$f(O) = O', \text{ và } \vec{f}(\vec{e}_i) = \vec{e}'_i$$

với mọi $i = 1, \dots, n$.

Nói cách khác, ánh xạ affine hoàn toàn được xác định bởi ảnh của một mục tiêu.

Hơn nữa, nếu $\dim \mathcal{A} = \dim \mathcal{A}'$ và $\{\vec{e}'_1, \dots, \vec{e}'_n\}$ là một cơ sở của \mathcal{A}' thì f là một đẳng cấu affine.

2.3.4 Hình học phi Euclid

Giới thiệu

Euclid là một trong những nhà toán học vĩ đại nhất lịch sử loài người, người đã viết quyển *Elements*, trình bày các mệnh đề làm cơ sở cho hình học, gọi là **các tiên đề**.

Chúng ta luôn cần một chứng minh để khẳng định hoặc bác bỏ bất kì mệnh đề toán học nào. Thông thường, việc chứng minh mệnh đề sẽ dựa trên các mệnh đề đã đúng trước đó. Tuy nhiên dễ thấy rằng việc truy ngược như vậy không thể thực hiện vô hạn mà phải tới một "điểm neo" nào đó, nghĩa là các mệnh đề đúng không cần chứng minh. Các mệnh đề đó chính là các tiên đề. Tuy nhiên các tiên đề phải đúng, ít nhất là được kiểm nghiệm trên thực tiễn.

Các tiên đề Euclid có năm tiên đề chính. Trong số đó có tiên đề thứ 5, viết số La Mã thành *tiên đề V*, là rắc rối và thiếu tự nhiên nhất:

Nếu hai đường thẳng tạo ra với một đường thẳng thứ ba cắt chúng hai góc trong cùng phía có tổng nhỏ hơn hai góc vuông thì khi kéo dài hai đường thẳng đó về phía ấy đến một lúc nào đó chúng sẽ phải cắt nhau.

Tiên đề này có hai điểm kì quái.

Đầu tiên là nó khá dài dòng. Các tiên đề thường là những điều hiển nhiên mà ta thấy được và là cơ sở cho hình học. Do đó tiên đề thường ngắn gọn, chỉ rõ một hiện tượng.

Ví dụ:

- qua hai điểm phân biệt trên mặt phẳng chỉ vẽ được một đường thẳng;
- đường thẳng kéo dài vô hạn về hai phía;
- mọi góc vuông đều bằng nhau.

Điểm kì quái thứ hai là tiên đề này khá rối rắm và chúng ta cần cẩn thận suy xét ý nghĩa của nó. Trong khi đó, như đã nói ở trên, tiên đề chỉ ra những quan sát thực tiễn và là cơ sở cho hình học (hoặc bất kì lĩnh vực toán học khác). Do đó tiên đề cần đơn giản, dễ nắm bắt.

Điều thú vị là hệ thống tiên đề của Euclid đã đúng vững 2000 năm mà không có vấn đề gì. Các nhà toán học thời đó, mỗi lần gặp các vấn đề về hình học, sẽ tham khảo các tài liệu do Euclid để lại. Còn có thông tin rằng những nghiên cứu dám nghi ngờ hoặc phản đối hệ tiên đề Euclid sẽ bị các nhà khoa học xung quanh bác bỏ, khá giống với tòa án dị giáo thời Trung Cổ.

Tuy nhiên, ba nhà toán học kiệt xuất đã xây dựng nên hình học phi Euclid độc lập nhau. Hình học phi Euclid không bác bỏ hình học Euclid, mà chúng bổ trợ cho nhau, làm đầy đủ các khía cạnh hình học.

Phần này mình tham khảo từ series [Путь к геометрии Лобачевского](#). Trong tiếng Việt tên series nghĩa là "Đường tới hình học Lobachevsky". Series gồm 6 phần nhưng mình chỉ tham khảo 5 phần đầu. Phần cuối là [TODO] sau này nghiên cứu thêm.

Sự ra đời hình học phi Euclid

Tiên đề V có vẻ rối rắm và dài dòng. Do đó nhiều nhà toán học cũng nghĩ rằng, có khi nào đây là một định lý chứ không phải tiên đề không? Như vậy họ đã cố gắng chứng minh tiên đề V bằng các tiên đề còn lại. Kết quả là họ ... thất bại.

Nhà toán học người Nga Lobachevsky có một cách tiếp cận khác. Ông này kiểu: "Hmm, giả sử mình thay tiên đề này bởi phủ định của nó rồi đi chứng minh các định lý hình học thì có khi nào phát sinh mâu thuẫn không nhỉ?". Nghĩa là ông ấy giữ nguyên tất cả tiên đề Euclid trừ tiên đề V, và thay tiên đề V thành phủ định của nó, rồi đi chứng minh các định lý hình học và hy vọng tìm ra mâu thuẫn toán học nào đó.

Kết quả là Lobachevsky đã thất bại (trong việc tìm ra mâu thuẫn).

Ông đã chứng minh hàng chục định lý mà không phát hiện điều gì bất thường.

Nếu vậy, phải chăng hình học phi Euclid khi thay tiên đề V bởi phủ định của nó, cũng chặt chẽ như hình học Euclid?

Phần sau mình sẽ đề cập tới hình học phi Euclid trong hệ tọa độ, cụ thể là các mặt cong trên nền tảng hình học Lobachevsky.

Tích vô hướng trong không gian Euclidean

Với hai vector $\mathbf{x} = (x_1, x_2, x_3)$ và $\mathbf{y} = (y_1, y_2, y_3)$, tích vô hướng của hai vector được định nghĩa là

$$\langle \mathbf{x}, \mathbf{y} \rangle = x_1y_1 + x_2y_2 + x_3y_3.$$

Độ dài của vector \mathbf{x} , kí hiệu là $\|\mathbf{x}\|$ và được tính bởi công thức

$$\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}.$$

Khoảng cách giữa hai điểm $A = \mathbf{x} = (x_1, x_2, x_3)$ và $B = \mathbf{y} = (y_1, y_2, y_3)$ là

$$AB = d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|.$$

Độ dài đường cong trong không gian Euclide

Trong bài viết về tích phân đường mòn đã tìm công thức tính độ dài đường trên mặt phẳng Euclide. Khi đó, nếu đường cong γ xác định các tọa độ bởi tham số t , nghĩa là

$$\gamma(t) = \begin{cases} x = x(t) \\ y = y(t) \end{cases}$$

thì độ dài đường cong từ t_0 tới t_1 (tức là $t_0 \leq t \leq t_1$) là tích phân

$$l(\gamma) = \int_{\gamma} dl = \int_{t_0}^{t_1} \sqrt{dx^2 + dy^2} = \int_{t_0}^{t_1} \sqrt{(x'(t))^2 + (y'(t))^2} dt.$$

Tương tự, đối với đường cong trong không gian Euclide $Oxyz$ có tọa độ là hàm số theo tham số t , nghĩa là

$$\gamma(t) = \begin{cases} x = x(t) \\ y = y(t) \\ z = z(t) \end{cases}$$

thì độ dài đường cong từ t_0 tới t_1 là tích phân

$$l(\gamma) = \int_{\gamma} dl = \int_{t_0}^{t_1} \sqrt{dx^2 + dy^2 + dz^2}.$$

Ở đây một biểu thức quan trọng sẽ được sử dụng xuyên suốt về sau là

$$dl = \sqrt{dx^2 + dy^2 + dz^2},$$

nhưng chúng ta sẽ bình phương để dễ tính toán

$$dl^2 = dx^2 + dy^2 + dz^2.$$

Nếu chúng ta có nhiều tham số thì sử dụng công thức đạo hàm riêng. Giả sử $x = x(u, v)$ và $y = y(u, v)$ thì

$$\begin{aligned} dx &= \frac{\partial x}{\partial u} du + \frac{\partial x}{\partial v} dv = \dots \\ dy &= \frac{\partial y}{\partial u} du + \frac{\partial y}{\partial v} dv = \dots \\ dz &= \frac{\partial z}{\partial u} du + \frac{\partial z}{\partial v} dv = \dots \end{aligned}$$

Khi đó

$$dl^2 = (a(u, v)du + b(u, v)dv)^2.$$

Mặt cầu

Mặt cầu là tập hợp các điểm cách một điểm cố định một khoảng không đổi trong không gian.

Mặt cầu tâm $O(x_0, y_0, z_0)$ với bán kính r là tập hợp các điểm $M(x, y, z)$ thỏa phương trình

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2.$$

Chúng ta sẽ đưa tâm mặt cầu về gốc tọa độ $O(0, 0, 0)$ để thuận tiện tính toán về sau. Phương trình sẽ trở thành

$$x^2 + y^2 + z^2 = r^2.$$

Một cách khác để biểu diễn các điểm của mặt cầu là sử dụng tọa độ cầu, nghĩa là biểu diễn tọa độ (x, y, z) thông qua hai tham số (θ, ϕ) (ở đây không có r vì bán kính đã cố định rồi). Khi đó, tọa độ sẽ có dạng

$$\begin{cases} x = r \sin \theta \cos \phi \\ y = r \sin \theta \sin \phi \\ z = r \cos \theta, \end{cases},$$

trong đó $0 \leq \theta, \phi \leq 2\pi$.

Khoảng cách giữa hai điểm trên mặt cầu

Trên mặt phẳng, khoảng cách giữa hai điểm là độ dài phần đường thẳng nằm giữa hai điểm đó. Tuy nhiên, trên mặt cầu thì không phải là đường thẳng mà chúng ta thường biết nữa.

Sử dụng công thức về độ dài đường cong bên trên có thể xác định được khoảng cách giữa hai điểm trên mặt cầu (trong tọa độ cực) là

$$\begin{aligned} dx &= r \cos \theta \cos \phi \, d\theta - r \sin \theta \sin \phi \, d\phi \\ dy &= r \cos \theta \sin \phi \, d\theta + r \sin \theta \cos \phi \, d\phi \\ dz &= -r \sin \theta \, d\theta \end{aligned}$$

Từ đây ta suy ra

$$dl^2 = dx^2 + dy^2 + dz^2 = r^2(d\theta^2 + \sin^2 \theta \, d\phi^2).$$

Ví dụ, mình muốn tính độ dài đường xích đạo, là vòng có độ dài lớn nhất trên mặt cầu, ứng với $\theta = \pi/2$ và $0 \leq \phi \leq 2\pi$, theo công thức trên

$$\sin \frac{\pi}{2} = 1, \quad d\theta = 0,$$

nên

$$l(\gamma) = \int dl = \int r \sqrt{0^2 + \sin^2(\pi/2) \, d\phi^2} = \int_0^{2\pi} r \, d\phi = 2\pi r.$$

Đường thẳng trên mặt cầu

Mỗi mặt phẳng nếu cắt mặt cầu sẽ được một "đường". Đường đó là tập hợp các điểm thỏa mãn hệ phương trình

$$\begin{cases} x^2 + y^2 + z^2 = 1 \\ ax + by + cz = 0. \end{cases}$$

Ở đây lấy đường tròn đơn vị và mặt phẳng đi qua gốc tọa độ cho đơn giản, (a, b, c) là các số cố định.

Khi đó hệ tọa độ cầu sẽ là

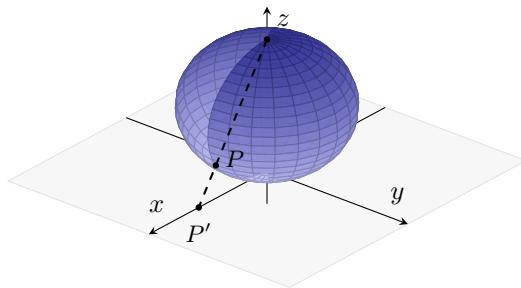
$$\begin{aligned} x &= \cos \phi \cos t - \sin \phi \sin t \cos \theta \\ y &= \sin \phi \cos t \cos \theta \\ z &= \sin t \sin \theta. \end{aligned}$$

Trong đó θ và ϕ cố định, còn t thay đổi từ 0 tới 2π trên vòng tròn cắt.

Trên mặt cầu không tồn tại hai đường song song.

Phép chiếu lập thể

Phép chiếu lập thể (Stereographic projection, Стереографическая проекция) là phép chiếu mặt cầu lên mặt phẳng.

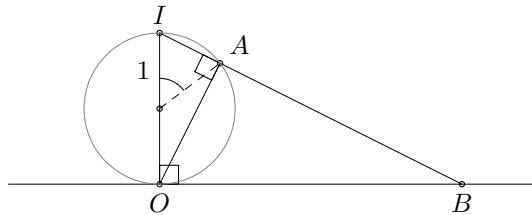


Ở hình trên, xét mặt cầu

$$x^2 + y^2 + (z - 1)^2 = 1$$

và mặt phẳng $z = 0$.

Chọn điểm cực bắc $I(0, 0, 2)$ làm tâm phép chiếu. Khi đó, phép chiếu lập thể biến mỗi điểm P thuộc mặt cầu thành điểm P' thuộc mặt phẳng chiếu sao cho I, P và P' thẳng hàng. Nói cách khác, P' là giao điểm của đường thẳng IP và mặt phẳng chiếu.



Trong trường hợp này, khi chiếu một vòng tròn song song mặt phẳng lên mặt phẳng ta được hình tròn. Lúc này, góc ϕ sẽ không thay đổi khi chiếu lên mặt phẳng, nhưng bán kính sẽ thay đổi, không phải là bán kính mặt cầu ban đầu.

Để tìm bán kính hình tròn ở mặt phẳng chiếu, trong tam giác vuông IOB ta có

$$OB = IO \cdot \cot \angle IBO = IO \cdot \cot \angle IOA = IO \cdot \cot \left(\frac{\angle ICA}{2} \right) = IO \cdot \cot \frac{\theta}{2}.$$

Một công thức lượng giác quen thuộc

$$\cos^2 \alpha = \frac{1 + \cos 2\alpha}{2}, \quad \sin^2 \alpha = \frac{1 - \cos 2\alpha}{2},$$

suy ra

$$\cot^2 \alpha = \frac{\cos^2 \alpha}{\sin^2 \alpha} = \frac{(1 + \cos 2\alpha)/2}{(1 - \sin 2\alpha)/2} = \frac{1 + \cos 2\alpha}{1 - \cos 2\alpha}.$$

Như vậy

$$\cot \frac{\theta}{2} = \sqrt{\frac{1 + \cos \theta}{1 - \cos \theta}}$$

và thay $IO = 2$ vào ta có

$$\begin{cases} r = 2\sqrt{\frac{1 + \cos \theta}{1 - \cos \theta}} = OB \\ \phi' = \phi \end{cases}$$

là các tham số của ảnh trên mặt phẳng qua phép chiếu lật thể. Tọa độ của điểm trên mặt phẳng chiếu là

$$\begin{cases} u = r \cos \phi' = r(\theta) \cos \phi \\ v = r \sin \phi' = r(\theta) \sin \phi. \end{cases}$$

Ở trên, bình phương độ dài đường cong trên mặt cầu được tính theo vi phân

$$dl^2 = dx^2 + dy^2 + dz^2$$

và ở tọa độ cầu là

$$dl^2 = d\theta^2 + \sin^2 \phi d\phi^2$$

với $r = 1$.

Bây giờ chúng ta tìm cách thay $d\theta$ và $d\phi$ từ kết quả phép chiếu lật thể vào bình phương vi phân đường cong. Để làm điều đó thì cần biểu diễn ngược lại θ theo r , còn $\phi' = \phi$ nên không cần xét tới.

Ta có

$$r = 2\sqrt{\frac{1 + \cos \theta}{1 - \cos \theta}} \implies \cos \theta = \frac{r^2 - 4}{r^2 + 4} \iff \theta = \arccos\left(\frac{r^2 - 4}{r^2 + 4}\right).$$

Lúc này, đạo hàm của $\arccos(x) = -\frac{1}{\sqrt{1-x^2}}$ nên mình suy ra

$$\begin{aligned} d\theta &= -\frac{1}{\sqrt{1-\left(\frac{r^2-4}{r^2+4}\right)^2}} \cdot \left(\frac{r^2-4}{r^2+4}\right)' dr \\ &= -\frac{1}{\frac{1}{r^2+4} \cdot \sqrt{(r^2+4)^2 - (r^2-4)^2}} \cdot \frac{2r(r^2+4) - 2r(r^2-4)}{(r^2+4)^2} dr \\ &= -\frac{16r}{(r^2+4) \cdot \sqrt{16r^2}} dr. \end{aligned}$$

Bình phương hai vế suy ra

$$d\theta^2 = \frac{16}{(r^2+4)^2} dr^2.$$

Tương tự, với $\sin^2 \theta d\phi^2$ thì ta thay biểu diễn $\cos \theta$ theo r bên trên vào

$$\sin^2 \theta = 1 - \cos^2 \theta = 1 - \left(\frac{r^2-4}{r^2+4}\right)^2 = \frac{16r^2}{(r^2+4)^2}.$$

Như vậy ta có

$$dl^2 = \frac{16}{(r^2+4)^2} (dr^2 + r^2 d\phi^2).$$

Ở trên chúng ta muốn dời tọa độ (r, ϕ) sang tọa độ (u, v) với

$$\begin{cases} u = r \cos \phi \\ v = r \sin \phi \end{cases} \iff \begin{cases} r^2 = u^2 + v^2 \\ \phi = \arctan \frac{u}{v} \end{cases}$$

và

$$dl^2 = \frac{16}{(u^2 + v^2 + 4)^2} (du^2 + dv^2).$$

Câu hỏi. Ảnh của các đường thẳng trên mặt cầu (các vòng tròn lớn) lên mặt phẳng là gì?

Trả lời. Đối với các đường không đi qua cực bắc (điểm I) thì ảnh là hình tròn, còn các đường đi qua cực bắc thì ảnh là đường thẳng đi qua gốc tọa độ.

[TODO] Tìm phương trình của hình chiếu.

Không gian giả Euclid

Trong không gian Euclid, tích vô hướng của hai vector $\mathbf{x} = (x_1, x_2, x_3)$ và $\mathbf{y} = (y_1, y_2, y_3)$ là

$$\mathbb{R}^3 : \langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 + x_2 y_2 + x_3 y_3.$$

Trong không gian giả Euclid, tích vô hướng sẽ là

$$\mathbb{R}_1^3 : \langle \mathbf{x}, \mathbf{y} \rangle = -x_1 y_1 + x_2 y_2 + x_3 y_3.$$

Như vậy, độ dài vector $\|\mathbf{x}\|^2 = \langle \mathbf{x}, \mathbf{x} \rangle$ trong không gian giả Euclid có thể là số âm.

Giả cầu (Pseudosphere)

Để thấy rằng do mặt cầu là tập hợp các điểm cách điểm cố định một đoạn không đổi r nên cũng tương đương với mặt cầu là tập các vector có độ dài bằng r , nghĩa là

$$S^2 = \{\mathbf{x} : \langle \mathbf{x}, \mathbf{x} \rangle = r^2\} \subset \mathbb{R}^3.$$

Bây giờ ta định nghĩa giả cầu là tập các điểm thỏa phương trình

$$-x^2 + y^2 + z^2 = -r^2,$$

hay tương đương là

$$L^2 = \{\mathbf{x} : \langle \mathbf{x}, \mathbf{x} \rangle = -r^2\} \subset \mathbb{R}_1^3.$$

Để tham số hóa giả cầu chúng ta cần thống nhất hệ thống tọa độ trong không gian giả Euclid \mathbb{R}_1^3 :

1. Tích vô hướng trong \mathbb{R}_1^3 là

$$\langle \mathbf{x}, \mathbf{y} \rangle = -x_1 y_1 + x_2 y_2 + x_3 y_3.$$

2. Độ dài vector (chuẩn Euclid) vẫn là tích vô hướng của chính nó

$$\|\mathbf{x}\| = \langle \mathbf{x}, \mathbf{x} \rangle.$$

3. Vi phân độ dài đường cong

$$dl^2 = \|(dx, dy, dz)\| = -dx^2 + dy^2 + dz^2.$$

Tiếp theo chúng ta sẽ biểu diễn giả cầu trong tọa độ cực

$$L^2 = \begin{cases} x = \cosh u \\ y = \sinh u \cos \theta \\ z = \sinh u \sin \theta \end{cases}$$

Ở đây các công thức có vẻ hơi lạ vì không phải các hàm lượng giác (trigonometric functions) như chúng ta hay gặp, mà là các hàm hyperbol (hyperbolic function) được định nghĩa như sau:

$$\sinh x = \frac{e^x - e^{-x}}{2}, \quad \cosh x = \frac{e^x + e^{-x}}{2}.$$

Như vậy ta có thể tìm được các vi phân

$$\begin{aligned} dx &= \frac{\partial x}{\partial u} du + \frac{\partial x}{\partial \theta} d\theta = \sinh u du \\ dy &= \frac{\partial y}{\partial u} du + \frac{\partial y}{\partial \theta} d\theta = \cosh u \cos \theta du - \sinh u \sin \theta d\theta \\ dz &= \frac{\partial z}{\partial u} du + \frac{\partial z}{\partial \theta} d\theta = \cosh u \sin \theta du + \sinh u \cos \theta d\theta \end{aligned}$$

Như vậy ta có

$$dl^2 = -dx^2 + dy^2 + dz^2 = du^2 + \sinh^2 u d\theta,$$

ở đây lưu ý là

$$\cosh^2 u - \sinh^2 u = \frac{(e^u + e^{-u})^2 - (e^u - e^{-u})^2}{4} = \frac{4e^u e^{-u}}{4} = 1.$$

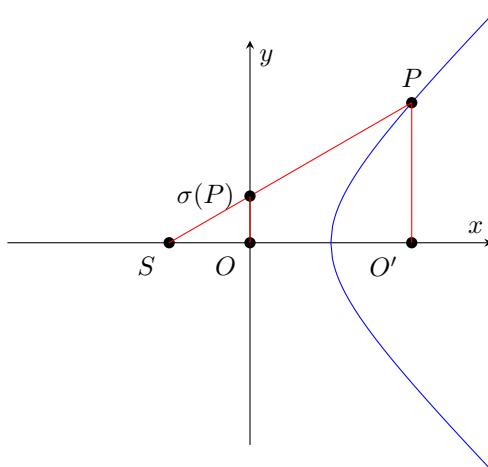
Tương tự, bây giờ, vòng tròn lớn trong không gian giả Euclid là giao giữa mặt phẳng và giả cầu, tức là

$$\{-x^2 + y^2 + z^2 = -1, ax + by + cz = 0\}$$

với a, b, c cố định.

Phép chiếu lập thể trên giả cầu

Khi chúng ta cắt giả cầu với một mặt phẳng (lát cắt), ví dụ $z = 0$, chúng ta sẽ thấy một đường hyperbol.



Hình trên là phép chiếu lên mặt phẳng $x = 0$ và chúng ta đang nhìn từ trên xuống (theo hướng $z = 0$).

Kí hiệu phép chiếu lập thể tâm S biến điểm P trên giả cầu thành điểm $\sigma(P)$. Lúc này chúng ta sẽ thiết lập ánh xạ $\sigma : (x, y, z) \rightarrow (u, v)$ với $(x, y, z) \in L^2$ là điểm thuộc giả cầu.

Trong trường hợp này các thông số ban đầu là

$$\begin{aligned} S &= (-1, 0, 0), \\ P &= (x, y, z), \\ -x^2 + y^2 + z^2 &= -1, \\ \sigma(P) &= (u, v). \end{aligned}$$

Đầu tiên phương trình tham số của đường thẳng SP là

$$\begin{cases} x = x(t) = -1 + (x+1)t \\ y = y(t) = yt \\ z = z(t) = zt \end{cases}$$

Đường thẳng SP cắt mặt phẳng $x = 0$ tại $x(t_0) = 0$, như vậy

$$-1 + (x+1)t_0 = 0 \iff t_0 = \frac{1}{x+1} \implies (y, z)(t_0) = \left(\frac{y}{x+1}, \frac{z}{x+1} \right).$$

Lúc này tọa độ (u, v) chính là $(y, z)(t_0)$, nói cách khác

$$(y, z)(t_0) = (u, v) \implies \begin{cases} y = (x+1)u \\ z = (x+1)v \end{cases}$$

Bây giờ thay ngược y và z vào biểu thức giả cầu (hyperbolic) ta được

$$\begin{aligned} -x^2 + y^2 + z^2 &= -1 \\ -x^2 + (x+1)^2 u^2 + (x+1)^2 v^2 &= -1 \\ -x^2 + (x+1)^2 (u^2 + v^2) + 1 &= 0 \\ -x^2 + (x^2 + 2x + 1)(u^2 + v^2) + 1 &= 0 \\ (u^2 + v^2 - 1)x^2 + 2(u^2 + v^2)x + (u^2 + v^2 + 1) &= 0 \end{aligned}$$

Xem đây là phương trình bậc hai theo x , tính delta ta có

$$\Delta' = (u^2 + v^2)^2 - (u^2 + v^2 - 1)(u^2 + v^2 + 1) = (u^2 + v^2)^2 - ((u^2 + v^2)^2 - 1) = 1.$$

Nghiệm của phương trình bậc hai khi đó là

$$x_{1,2} = \frac{-(u^2 + v^2) \pm 1}{u^2 + v^2 - 1}.$$

Như vậy

$$x_1 = -1, \quad x_2 = \frac{1 + u^2 + v^2}{1 - u^2 - v^2}.$$

Nghiệm thứ hai cho phép biểu diễn y và z theo u và v :

$$\begin{aligned} x + 1 &= \frac{1 + u^2 + v^2}{1 - u^2 - v^2} + 1 = \frac{2}{1 - u^2 - v^2} \\ \implies \begin{cases} x = \frac{1 + u^2 + v^2}{1 - u^2 - v^2} \\ y = (x+1)u = \frac{2u}{1 - u^2 - v^2} \\ z = (x+1)v = \frac{2v}{1 - u^2 - v^2} \end{cases} \end{aligned}$$

Bây giờ chúng ta đã có thể tính được độ dài hình chiếu. Đầu tiên ta vi phân các biến x, y và z theo u và v :

$$\begin{aligned} dx &= \frac{\partial x}{\partial u} du + \frac{\partial x}{\partial v} dv = \frac{4u du + 4v dv}{(1 - u^2 - v^2)^2} \\ dy &= \frac{\partial y}{\partial u} du + \frac{\partial y}{\partial v} dv = \frac{2(1 + u^2 - v^2)}{(1 - u^2 - v^2)^2} du + \frac{4uv}{(1 - u^2 - v^2)^2} dv \\ dz &= \frac{\partial z}{\partial u} du + \frac{\partial z}{\partial v} dv = \frac{4uv}{(1 - u^2 - v^2)^2} du + \frac{2(1 - u^2 + v^2)}{(1 - u^2 - v^2)^2} dv \end{aligned}$$

Khi đó bình phương vi phân đường cong dl^2 là

$$\begin{aligned} (1 - u^2 - v^2)^4 dl^2 &= (1 - u^2 - v^2)^4 \cdot (-dx^2 + dy^2 + dz^2) \\ &= -(4u du + 4v dv)^2 + (2(1 + u^2 - v^2) du + 4uv dv)^2 + (4uv du + 2(1 + v^2 - u^2) dv)^2 \\ &= (-16u^2 + 4(1 + u^2 - v^2)^2 + 16u^2v^2) du^2 \\ &\quad + (-32uv + 16uv(1 + u^2 - v^2) + 16uv(1 - u^2 + v^2)) dudv \\ &\quad + (-16v^2 + 16u^2v^2 + 4(1 - u^2 + v^2)^2) dv^2 \\ &= 4(1 - u^2 - v^2)^2 du^2 + 0 dudv + 4(1 - u^2 - v^2)^2 dv^2 \end{aligned}$$

Như vậy thu được

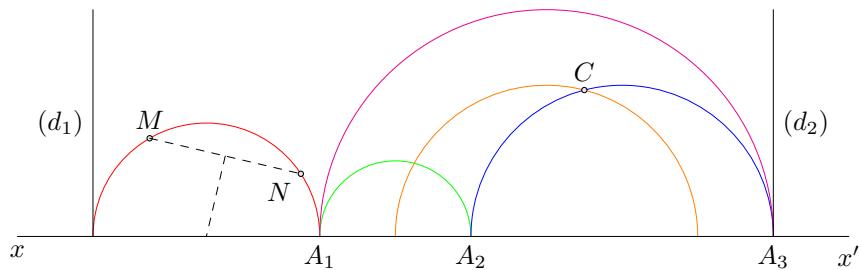
$$dl^2 = \frac{4}{(1 - u^2 - v^2)^2} (du^2 + dv^2).$$

Cách xây dựng của Poincare

Xét đường thẳng xx' . Đường thẳng xx' chia mặt phẳng làm hai phần, ta gọi là *nửa trên* và *nửa dưới*.

Bây giờ ta xây dựng các đối tượng phi Euclid sau:

- 1) **Điểm** phi Euclid là tất cả điểm ở nửa mặt phẳng trên chia bởi xx' nhưng không tính các điểm trên xx'
- 2) **Đường thẳng** phi Euclid là mọi nửa đường thẳng vuông góc với xx' và nửa đường tròn tâm nằm trên xx' không tính các điểm nằm trên xx' .
- 3) Nửa mặt phẳng trên chia bởi đường thẳng xx' được gọi là **mặt phẳng phi Euclid**.



Cách xây dựng này thỏa mãn các tiên đề Euclid (trừ tiên đề V):

- qua hai điểm phân biệt chỉ vẽ được một đường thẳng: với hai điểm A và B bất kì thuộc mặt phẳng phi Euclid, nếu $\$AB \perp\!\!\!\perp xx'\$$ thì ta vẽ đường thẳng $\$AB\$$, ngược lại ta vẽ đường tròn tâm của $\$AB\$$ cắt xx' tại I và vẽ đường tròn tâm I , bán kính $\$IA\$$. Như vậy các đường thẳng được xác định duy nhất
- đường thẳng kéo dài vô hạn về hai phía: trong hai trường hợp, đường thẳng tiệm cận xx' nhưng không chạm vào xx' . Do đó đường thẳng kéo dài vô hạn về xx' (tương tự việc chứng minh đoạn $(0; 1)$ có điểm bằng với \mathbb{R} của Cantor).

Từ hình vẽ trên có thể thấy có nhiều hơn một đường thẳng song với $\$(d_2)\$$ mà đi qua điểm C là đường tròn màu cam và đường tròn màu xanh nước suối.

Trong trường hợp này, tổng ba góc của tam giác có thể có số đo tùy ý, thậm chí bằng 0. Trên hình vẽ, tam giác tạo bởi A_1 , A_2 và A_3 , giới hạn bởi các đường tròn màu hướng, màu xanh lá cỏ và màu xanh nước suối, tạo thành tam giác có tổng ba góc bằng 0.

2.4 Giải tích

2.4.1 Giải tích



Hình 2.45: Karl Theodor Wilhelm Weierstrass (1815-1897)

Phần này mình lấy từ các sách giáo khoa toán của bậc THPT thời mình còn đi học (trước 2018), chủ yếu là cuốn [1].

Giới hạn

Giới hạn của dãy số

Definition 1.41 (Giới hạn hữu hạn của dãy số)

Cho dãy số $\{a_n\}$. Ta nói dãy $\{a_n\}$ có giới hạn hữu hạn L nếu với mọi $\varepsilon > 0$, tồn tại $n_0 \in \mathbb{N}$ sao cho với mọi $n \geq n_0$ thì

$$|a_n - L| < \varepsilon.$$

Kí hiệu: $\lim_{n \rightarrow \infty} a_n = L$.

Example 1.22

Xét dãy số cho bởi công thức $a_n = \frac{1}{n}$. Ta chứng minh dãy số có giới hạn hữu hạn là 0.

Với mọi $\varepsilon > 0$ tùy ý, ta cần chứng minh tồn tại số $n_0 \geq 1$ sao cho với mọi $n \geq n_0$ thì $|a_n - 0| < \varepsilon$.

Nói cách khác $a_{n_0} < \varepsilon$, hay tương đương với

$$\frac{1}{n_0} < \varepsilon \Leftrightarrow n_0 > \frac{1}{\varepsilon}.$$

Vậy ta chỉ cần chọn n_0 thỏa bất đẳng thức trên (luôn tìm được).

Kết luận: $\lim_{n \rightarrow \infty} a_n = 0$.

Definition 1.42 (Dãy số có giới hạn vô cực)

Cho dãy số $\{a_n\}$. Ta nói dãy số có giới hạn ở dương vô cực nếu với mọi $M > 0$, tồn tại $n_0 \in \mathbb{N}$ sao cho với mọi $n \geq n_0$ thì $a_n > M$.

Nói cách khác, nếu ta chọn một số M rất lớn bất kì, thì mọi số hạng của dãy số kể từ một số hạng nào đó trở đi luôn lớn hơn M . Định nghĩa về dãy số có giới hạn ở âm vô cực cũng tương tự.

Giới hạn của hàm số

Để định nghĩa giới hạn của hàm số $y = f(x)$ khi x tiến tới x_0 ta có hai loại định nghĩa.

Definition 1.43 (Giới hạn hàm số qua giới hạn dãy số)

Xét hàm số $f(x)$. Ta nói hàm số có giới hạn hữu hạn L khi x tiến tới x_0 , nếu với mọi dãy số $\{x_n\}$ mà $\lim_{n \rightarrow \infty} x_n = x_0$, thì $\lim_{n \rightarrow \infty} f(x_n) = L$.

Định nghĩa này tuân theo giới hạn của dãy số. Khi đó mọi phần tử của dãy số từ một số hạng nào đó trở đi cho giá trị $f(x_n)$ tiến về L .

Định nghĩa của hàm số theo kiểu Cauchy (hay còn được gọi là ngôn ngữ $\delta - \varepsilon$) là kiểu định nghĩa phổ biến được giảng dạy trong nhà trường.

Definition 1.44 (Giới hạn hàm số kiểu Cauchy)

Xét hàm số $f(x)$. Ta nói hàm số có giới hạn hữu hạn L khi x tiến tới x_0 , nếu với mọi $\varepsilon > 0$, tồn tại $\delta > 0$ sao cho với mọi x mà $|x - x_0| < \delta$ thì $|f(x) - L| < \varepsilon$.

Kí hiệu: $\lim_{x \rightarrow x_0} f(x) = L$.

Ta có thể thấy ở đây x tiến về x_0 (khá giống định nghĩa giới hạn hàm số) và $f(x)$ tương ứng tiến về L .

Tương tự ta cũng có giới hạn hàm số ở vô cực.

i Definition 1.45 (Giới hạn hàm số ở vô cực)

Với hàm số $f(x)$, ta nói hàm số có giới hạn tại dương vô cực khi x tiến về x_0 nếu với mọi $M > 0$, tồn tại $\delta > 0$ sao cho với mọi x mà $|x - x_0| < \delta$ thì $f(x) > M$.

Kí hiệu: $\lim_{x \rightarrow x_0} f(x) = +\infty$.

i Definition 1.46 (Giới hạn một bên)

Ta nói hàm số $f(x)$ có giới hạn phải L tại x_0 khi x tiến về bên phải x_0 nếu với mọi $\varepsilon > 0$, tồn tại $\delta > 0$ sao cho với mọi $0 < x - x_0 < \delta$ thì $|f(x) - L| < \varepsilon$.

Kí hiệu: $\lim_{x \rightarrow x_0^+} f(x) = L$.

Nghĩa là chúng ta chỉ xét giới hạn khi x tiến tới x_0 từ bên phải $x > x_0$. Tương tự cho giới hạn trái.

Lưu ý rằng trong nhiều trường hợp, mặc dù cùng tiến tới x_0 nhưng giới hạn trái và giới hạn phải có thể không bằng nhau.

i Example 1.23

Xét hàm số $y = \frac{1}{x}$. Ta thấy hàm số không xác định tại $x = 0$, và giới hạn trái và phải khác nhau do

$$\lim_{x \rightarrow 0^+} = +\infty, \quad \lim_{x \rightarrow 0^-} = -\infty.$$

Tính liên tục của hàm số

Cho hàm số $f(x)$ xác định trên miền D và x_0 là một điểm thuộc D .

i Definition 1.47 (Hàm số liên tục tại một điểm)

Ta nói hàm số $f(x)$ liên tục tại x_0 nếu

$$\lim_{x \rightarrow x_0} f(x) = f(x_0).$$

Định nghĩa tương tự cho liên tục trái và liên tục phải (ta lấy giới hạn một bên).

Như vậy, có ba khả năng hàm số không liên tục tại một điểm.

1. Hàm số không xác định tại x_0 .
2. Hàm số xác định tại x_0 nhưng giới hạn tại đó không bằng $f(x_0)$.
3. Giới hạn trái và giới hạn phải không bằng nhau.

Nếu hàm số không liên tục tại x_0 , ta gọi hàm số bị **gián đoạn** tại x_0 .

Nếu hàm số liên tục tại mọi điểm trên khoảng $(a; b)$ thì ta nói hàm số liên tục trên khoảng đó.

Tính đơn điệu và cực trị

Đầu tiên chúng ta cần một định lý về tính đơn điệu của hàm số khả vi.

i Theorem 1.6

Xét hàm số $f(x)$ khả vi trên khoảng $(a; b)$. Nếu $f'(x) > 0$ với mọi $x \in (a; b)$ thì $f(x)$ đồng biến trên $(a; b)$.

i Chứng minh

Theo định nghĩa đạo hàm thì

$$f'(x) = \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} > 0.$$

Như vậy, nếu $\Delta x > 0$ thì $f(x + \Delta x) - f(x) > 0$. Ta cũng có $x + \Delta x > x$ nên từ đây suy ra $f(x)$ đồng biến trên $(a; b)$.

Tương tự:

1. Nếu $f'(x) < 0$ với mọi $x \in (a; b)$ thì $f(x)$ nghịch biến trên $(a; b)$.
2. Nếu $f'(x) = 0$ với mọi $x \in (a; b)$ thì $f(x)$ có giá trị không đổi trên $(a; b)$.

i Definition 1.48 (Cực tiểu của hàm số)

Xét hàm số $f(x)$ liên tục trên khoảng $(a; b)$. Điểm $(x_0, f(x_0))$ được gọi là **cực tiểu** của hàm số $f(x)$ nếu tồn tại một lân cận U chứa x_0 nằm trong khoảng $(a; b)$ sao cho với mọi $x \in U$ thì $f(x) \geq f(x_0)$.

i Definition 1.49 (Cực đại của hàm số)

Xét hàm số $f(x)$ liên tục trên khoảng $(a; b)$. Điểm $(x_0, f(x_0))$ được gọi là **cực đại** của hàm số $f(x)$ nếu tồn tại một lân cận U chứa x_0 nằm trong khoảng $(a; b)$ sao cho với mọi $x \in U$ thì $f(x) \leq f(x_0)$.

Theo định nghĩa cực tiểu thì chỉ cần tồn tại lân cận chứa x_0 mà $f(x) \geq f(x_0)$ thì điểm đó là cực tiểu. Như vậy một hàm số có thể có nhiều cực tiểu, tương tự cũng có thể có nhiều cực đại.

Lưu ý rằng cực đại và cực tiểu không phải điểm chỉ giá trị lớn nhất hay giá trị nhỏ nhất của hàm số. Nó chỉ lớn nhất hoặc nhỏ nhất trong vùng lân cận đó theo định nghĩa, nên người ta còn gọi là cực trị địa phương.

Từ đó chúng ta có tính chất của cực trị.

❶ Remark 1.11

Xét hàm số $f(x)$ khả vi trên khoảng $(a; b)$ và một điểm $x_0 \in (a; b)$. Gọi $f'(x)$ là đạo hàm của $f(x)$ trên $(a; b)$. Khi đó điểm $x_0 \in (a; b)$ được gọi là

1. Nếu $f'(x)$ đổi chiều từ âm sang dương khi đi qua x_0 thì $(x_0, f(x_0))$ là điểm cực tiểu.
2. Nếu $f'(x)$ đổi chiều từ dương sang âm khi đi qua x_0 thì $(x_0, f(x_0))$ là điểm cực đại.

❶ Definition 1.50 (Dãy Cauchy)

Dãy $\{x_n\}$ được gọi là dãy Cauchy nếu với mọi $\varepsilon > 0$, tồn tại $N_0 \in \mathbb{N}$ sao cho, với mọi $m, n > N_0$ thì $|x_m - x_n| < \varepsilon$.

❶ Theorem 1.7 (Tiêu chuẩn Cauchy)

Dãy số $\{x_n\}$ có giới hạn hữu hạn khi và chỉ khi nó là dãy Cauchy.

Đạo hàm**Đạo hàm****❶ Definition (Đạo hàm)**

Cho hàm số $f(x)$ xác định trên miền D và x_0 là điểm thuộc D . Ta nói hàm số $f(x)$ có đạo hàm tại x_0 (hoặc khả vi tại x_0) nếu tồn tại giới hạn hữu hạn

$$\lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0}. \quad (2.3)$$

Kí hiệu đạo hàm của f tại x_0 là $f'(x_0)$.

Lưu ý rằng nếu giới hạn trên không phải là giới hạn hữu hạn (không tồn tại hoặc tiến tới vô cực) thì hàm số không có đạo hàm tại điểm x_0 .

❶ Example

Tính đạo hàm của hàm số $f(x) = x^3 + 2x^2 - 4$ tại $x_0 = 4$.

Ta khai triển

$$\begin{aligned} \frac{f(x) - f(x_0)}{x - x_0} &= \frac{f(x) - f(4)}{x - 4} \\ &= \frac{x^3 + 2x^2 - 4 - (4^3 + 2 \cdot 4^2 - 4)}{x - 4} \\ &= \frac{(x^3 - 4^3) + 2(x^2 - 4^2)}{x - 4} \\ &= \frac{(x - 4)(x^2 + 4x + 16) + 2(x - 4)(x + 4)}{x - 4} \\ &= x^2 + 4x + 16 + 2(x + 4). \end{aligned}$$

Cho x tiến tới 4 thì ta có đạo hàm tại $x = 4$ là

$$\begin{aligned} f'(4) &= \lim_{x \rightarrow 4} \frac{f(x) - f(4)}{x - 4} \\ &= \lim_{x \rightarrow 4} (x^2 + 4x + 16 + 2(x + 4)) \\ &= 4^2 + 4 \cdot 4 + 16 + 2 \cdot (4 + 4) = 64. \end{aligned}$$

Example

Xét hàm số $f(x) = x^2 + 1$ trên \mathbb{R} . Tìm đạo hàm tại $x_0 \in \mathbb{R}$.

Ta có

$$f(x) - f(x_0) = x^2 + 1 - (x_0^2 + 1) = (x - x_0)(x + x_0).$$

Khi đó $\frac{f(x) - f(x_0)}{x - x_0} = x + x_0$ nên ta có $\lim_{x \rightarrow x_0} (x + x_0) = 2x_0$.

Trong định nghĩa ở (2.3), nếu ta đặt

$$\Delta x = x - x_0, \quad \Delta y = y - y_0 = f(x) - f(x_0),$$

ta gọi Δx là **số gia của biến** x , tương tự Δy là **số gia của biến** y .

Trong định nghĩa, x tiến tới x_0 tương đương với Δx tiến tới 0. Chuyển về x_0 ta có $x = x_0 + \Delta x$ và từ đó $f(x) = f(x_0 + \Delta x)$. Định nghĩa đạo hàm ở trên có thể được viết lại

$$f'(x_0) = \lim_{\Delta x \rightarrow 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} = \lim_{\Delta x \rightarrow 0} \frac{\Delta y}{\Delta x}.$$

Nếu hàm số có đạo hàm tại mọi điểm trên khoảng (a, b) thì ta nói hàm số khả vi trên khoảng đó.

Ví dụ đối với hàm số $f(x) = x^3 + 2x^2 - 4$ như trên. Với mọi $x_0 \in \mathbb{R}$ ta có

$$\begin{aligned} f'(x_0) &= \lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0} \\ &= \lim_{x \rightarrow x_0} \frac{x^3 + 2x^2 - 4 - (x_0^3 + 2x_0^2 - 4)}{x - x_0} \\ &= \lim_{x \rightarrow x_0} \frac{(x^3 - x_0^3) + 2(x^2 - x_0^2)}{x - x_0} \\ &= \lim_{x \rightarrow x_0} (x^2 + xx_0 + x_0^2) + 2(x + x_0) \\ &= x_0^2 + x_0 \cdot x_0 + x_0^2 + 2(x_0 + x_0) = 3x_0^2 + 4x_0. \end{aligned}$$

Ta thấy rằng giới hạn trên luôn tồn tại với mọi $x_0 \in \mathbb{R}$ nên thay x_0 thành x ta có đạo hàm $f'(x) = 3x^2 + 4x$ của $f(x)$ trên \mathbb{R} .

Remark

Từ định nghĩa ta thấy rằng nếu $f(x)$ khả vi tại x_0 thì nó cũng liên tục tại x_0 . Tuy nhiên chiều ngược lại không đúng. Ví dụ với hàm số $y = |x|$, hàm số liên tục tại $x = 0$ nhưng giới hạn (đạo hàm) phải là 1, còn giới hạn (đạo hàm) trái là -1.

Về mặt hình ảnh, khi hàm số khả vi tại một điểm thì đồ thị sẽ "trơn", không gấp khúc tại điểm đó.

❶ Definition (Đạo hàm của hàm số trên một nửa khoảng hay một đoạn)

Cho hàm số $f(x)$ xác định trên tập K , trong đó K là một nửa khoảng hay một đoạn.

Hàm số $f(x)$ được gọi là có đạo hàm trên nửa khoảng $K = [a; b)$ nếu nó có đạo hàm tại mọi điểm thuộc $(a; b)$ và có đạo hàm phải tại $x = a$. Định nghĩa tương tự với $K = [a; +\infty)$.

Hàm số $f(x)$ được gọi là có đạo hàm trên nửa khoảng $K = (a; b]$ nếu nó có đạo hàm tại mọi điểm thuộc $(a; b)$ và có đạo hàm trái tại $x = b$. Định nghĩa tương tự với $K = (-\infty; b]$.

Hàm số $f(x)$ được gọi là có đạo hàm trên đoạn $[a; b]$ nếu nó có đạo hàm tại mọi điểm thuộc khoảng $(a; b)$, có đạo hàm phải tại $x = a$ và có đạo hàm trái tại $x = b$.

Đạo hàm của hàm số hợp

❶ Theorem

Nếu hàm số $h = h(x)$ có đạo hàm tại điểm x_0 và hàm số $y = g(h)$ có đạo hàm tại điểm $h_0 = h(x_0)$ thì hàm số hợp $f(x) = g(h(x))$ có đạo hàm tại điểm x_0 và

$$f'(x) = g'(h_0) \cdot h'(x_0).$$

Nếu giả thiết trên đúng với mọi điểm x thuộc tập xác định J thì hàm số hợp $y = f(x)$ có đạo hàm trên J và

$$f'(x) = g'(h(x)) \cdot h'(x).$$

Công thức trên còn được viết gọn là

$$f'_x = g'_h \cdot h'_x.$$

❷ Chứng minh

Theo định nghĩa đạo hàm thì

$$h(x_0 + a) - h(x_0) = h'(x_0) \cdot a + \varepsilon(a) \cdot a,$$

với $\varepsilon(a) \rightarrow 0$ khi $a \rightarrow 0$. Ở đây a đóng vai trò như Δx trong định nghĩa.

Tương tự cho hàm g ta có

$$g(h(x_0) + b) - g(h(x_0)) = g'(h(x_0)) \cdot b + \eta(b) \cdot b,$$

với $\eta(b) \rightarrow 0$ khi $b \rightarrow 0$.

Bây giờ xét

$$\begin{aligned} g(h(x_0 + a)) - g(h(x_0)) &= g(h(x_0) + h'(x_0) \cdot a + \varepsilon(a) \cdot a) - g(h(x_0)) \\ &= g(h(x_0) + c) - g(h(x_0)) = g'(h(x_0)) \cdot c + \eta(c) \cdot c \end{aligned}$$

với $h'(x_0) \cdot a + \varepsilon(a) \cdot a = c$.

Ta thấy rằng khi $a \rightarrow 0$ thì $\frac{c}{a} \rightarrow h'(x_0)$ và $c \rightarrow 0$, như vậy $\eta(n) \rightarrow 0$.

Từ đây suy ra

$$\frac{g(h(x_0 + a)) - g(h(x_0))}{a} \rightarrow g'(h(x_0)) \cdot \frac{c}{a} + 0 \rightarrow g'(h(x_0)) \cdot h'(x_0)$$

khi $a \rightarrow 0$.

Đạo hàm của hàm số ngược

Giả sử hàm số $f : I \rightarrow J$ là một hàm khả nghịch, nghĩa là có hàm ngược. Khi đó nếu f có đạo hàm khác 0 tại điểm $f^{-1}(x_0)$ thì f^{-1} cũng có đạo hàm tại điểm x_0 theo đẳng thức

$$(f^{-1})'(x_0) = \frac{1}{f'(f^{-1}(x_0))}.$$

Chúng ta sẽ không chứng minh ở đây vì chứng minh khá phức tạp.

Về mặt hình học, vì đồ thị của hàm số $y = f(x)$ và $y = f^{-1}(x)$ đối xứng với nhau qua đường thẳng $y = x$ nên theo công thức trên, nếu hệ số góc của tiếp tuyến đồ thị hàm số $y = f(x)$ tại điểm (x_0, y_0) là k thì hệ số góc của tiếp tuyến đồ thị hàm số $y = f^{-1}(x)$ tại điểm (y_0, x_0) là $\frac{1}{k}$.

Vi phân

Trong cách kí hiệu

$$f'(x) = \lim_{\Delta x \rightarrow 0} \frac{\Delta y}{\Delta x},$$

ta thay Δy thành dy và Δx thành dx thì vi phân được định nghĩa là

$$f'(x) = \frac{dy}{dx} \Leftrightarrow dy = f'(x) dx.$$

Cách kí hiệu vi phân có ý nghĩa là về trái là vi phân theo biến y và về phải là vi phân theo biến x . Do $y = f(x)$ nên khi vi phân hai về sẽ cho ra $dy = f'(x) dx$ (về trái là đa thức bậc 1 theo biến y).

Ví dụ phương trình $y^2 = x^3 + 4x - 7$ thì khi vi phân hai về ta có

$$(y^2)' dy = (x^3 + 4x - 7) dx \Leftrightarrow 2y dy = (3x^2 + 4) dx.$$

Một số định lí về giá trị trung bình

Theorem 1.9 (Bổ đề Fermat)

Cho f là một hàm số có đạo hàm trên $(a; b)$. Nếu $x_0 \in (a; b)$ là một điểm cực trị của f thì ta có $f'(x_0) = 0$.

Chứng minh

Ta chứng minh trong trường hợp x_0 là điểm cực tiểu. Trường hợp điểm cực đại tương tự.

Hàm f có đạo hàm trên $(a; b)$ nên tại điểm x_0 nó có đạo hàm bên trái và đạo hàm bên phải, và hai đạo hàm này bằng nhau.

Ta có $f'(x_0^+) = \lim_{x \rightarrow x_0^+} \frac{f(x) - f(x_0)}{x - x_0}$. Vì $x \rightarrow x_0^+$ nghĩa là $x > x_0$ (x tiến tới x_0 từ bên phải), và do x_0 là cực tiểu $f(x) - f(x_0) \geq 0$ nên phân số dưới dấu giới hạn lớn hơn 0. Suy ra $f'(x_0^+) \geq 0$.

Hoàn toàn tương tự ta chứng minh được $f'(x_0^-) \leq 0$. Và do $f'(x_0^+) = f'(x_0^-) = f'(x_0)$ nên $f'(x_0) = 0$.

Ta có điều phải chứng minh.

i Theorem 1.10 (Định lí Rolle)

Xét hàm số f liên tục trên đoạn $[a; b]$, có đạo hàm trên khoảng $(a; b)$ và $f(a) = f(b)$. Khi đó tồn tại c thuộc $(a; b)$ sao cho $f'(c) = 0$.

i Chứng minh

Để chứng minh định lí Rolle chúng ta cần bổ đề Fermat và một tính chất của hàm liên tục

Nếu f là một hàm số liên tục trên đoạn $[a; b]$ thì f đạt giá trị lớn nhất và giá trị nhỏ nhất trên đoạn đó.

Nếu cả hai giá trị lớn nhất (GTLN) và giá trị nhỏ nhất (GTNN) của f đều đạt tại biên thì do giả thiết $f(a) = f(b)$ ta suy ra GTLN = GTNN. Như vậy f là hàm hằng nên $f'(c) = 0$ với mọi $c \in (a; b)$.

Ngược lại, có một trong hai giá trị đó đạt được tại điểm $c \in (a; b)$. Khi đó c là điểm cực trị nên theo bổ đề Fermat thì $f'(c) = 0$.

Về mặt vật lí, định lí Rolle cho biết rằng nếu một chất di chuyển động trên đường thẳng bắt đầu từ điểm a và quay lại điểm xuất phát ở thời điểm b thì có một thời điểm c nào đó thuộc $(a; b)$ mà chất di chuyển lại, kể cả ta không biết tốc độ của chất di chuyển như nào.

Định lí Rolle là công cụ giúp khảo sát, đánh giá số nghiệm của các phương trình rất tốt.

Khi mở rộng định lí Rolle chúng ta được định lí Lagrange. Nếu trong khoảng thời gian từ a tới b , chất di chuyển trên đường thẳng từ vị trí $s(a)$ tới $s(b)$ (so với gốc tọa độ) thì vận tốc trung bình trong thời gian này là $\frac{s(b) - s(a)}{b - a}$. Định lí Lagrange nói rằng tồn tại thời điểm c thuộc $(a; b)$ sao cho vận tốc tức thời tại thời điểm này bằng vận tốc trung bình trên cả quãng đường $s(a)$ tới $s(b)$.

i Theorem 1.11 (Định lí Lagrange)

Xét hàm số f liên tục trên đoạn $[a; b]$, có đạo hàm trên khoảng $(a; b)$. Khi đó tồn tại c thuộc $(a; b)$ sao cho $f'(c)(b - a) = f(b) - f(a)$.

i Chứng minh

Xét hàm

$$g(x) = f(x) - \frac{f(b) - f(a)}{b - a} \cdot (x - a).$$

Khi đó ta có $g(a) = f(a)$ và

$$g(b) = f(b) - \frac{f(b) - f(a)}{b - a} \cdot (b - a) = f(b) - (f(b) - f(a)) = f(a).$$

Do g liên tục trên $[a; b]$, có đạo hàm trên $(a; b)$ và $g(a) = g(b)$ nên theo định lí Rolle tồn tại $c \in (a; b)$ sao cho $g'(c) = 0$. Ta lại có

$$g'(x) = f'(x) - \frac{f(b) - f(a)}{b - a},$$

nên thay c và $g'(x)$ ta có

$$0 = g'(c) = f'(c) - \frac{f(b) - f(a)}{b - a} \iff f'(c) = \frac{f(b) - f(a)}{b - a}.$$

Với cách tiếp cận và trình bày như trên thì định lí Lagrange được gọi là *định lí giá trị trung bình*.

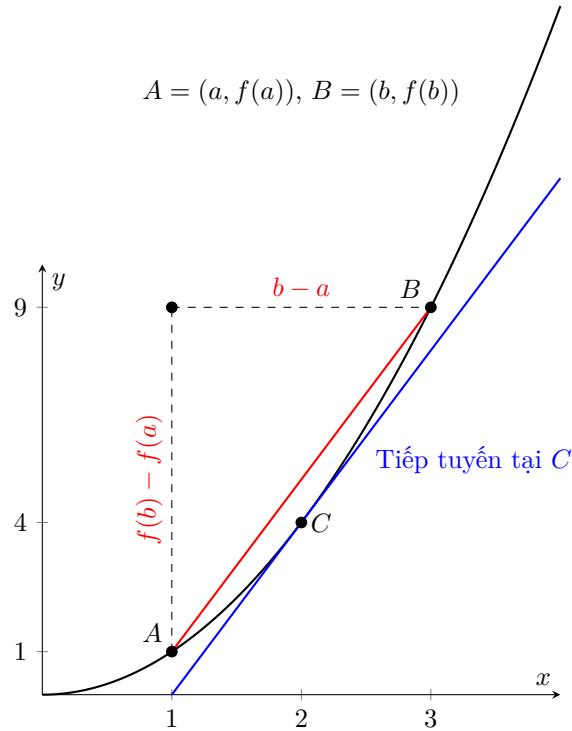
Thay đổi cách kí hiệu, đặt $a = x$ và $b = x + \Delta x$ thì ta có

$$f(x + \Delta x) - f(x) = f'(c) \cdot \Delta x$$

với $c \in (x, x + \Delta x)$. Lúc này định lí Lagrange được gọi là *định lí về số gia hữu hạn*.

Về mặt hình học có thể thấy $\frac{f(b) - f(a)}{b - a}$ là hệ số góc của dây cung nối hai điểm $(a, f(a))$ và $(b, f(b))$ của đồ thị hàm số $y = f(x)$, còn $f'(c)$ là hệ số góc của tiếp tuyến tại điểm có hoành độ $x = c$ thuộc đường cong. Khi đó định lí Lagrange có ý nghĩa

Nếu hàm số f liên tục trên $[a; b]$ và có đạo hàm trên $(a; b)$ thì tồn tại điểm $c \in (a; b)$ sao cho tiếp tuyến tại điểm $(c, f(c))$ song song với dây cung AB nối hai điểm $(a, f(a))$ và $(b, f(b))$.



Hình 2.46: Minh họa hình học của định lí Lagrange

Ở biểu thức của định lí Lagrange, đặt $g(x) = x$ thì $g(b) = b$, $g(a) = a$ và $g'(x) = 1$ với mọi $x \in (a; b)$. Khi đó ta có

$$f'(c) = \frac{f(b) - f(a)}{b - a} \iff \frac{f'(c)}{1} = \frac{f'(c)}{g'(c)} = \frac{f(b) - f(a)}{g(b) - g(a)}.$$

Nếu ta thay $g(x)$ là một hàm số khả vi tùy ý thì biểu thức này còn đúng không?

❶ Theorem 1.12 (Định lí Cauchy)

Nếu $f(x)$ và $g(x)$ là hai hàm số liên tục trên $[a; b]$, có đạo hàm trên $(a; b)$ và $g'(x) \neq 0$ với mọi $x \in (a; b)$ thì tồn tại $c \in (a; b)$ sao cho

$$\frac{f(b) - f(a)}{g(b) - g(a)} = \frac{f'(c)}{g'(c)}.$$

❷ Chứng minh

Ta xét hàm số

$$h(x) = (f(b) - f(a)) \cdot (g(x) - g(a)) - (g(b) - g(a)) \cdot (f(x) - f(a))$$

thì ta có $h(a) = h(b) = 0$. Hơn nữa hàm số $h(x)$ cũng liên tục trên $[a; b]$ và có đạo hàm trên $(a; b)$. Áp dụng định lí Rolle cho hàm số $h(x)$ thì tồn tại số $c \in (a; b)$ sao cho $h'(x) = 0$. Vì

$$h'(x) = (f(b) - f(a)) \cdot g'(x) - (g(b) - g(a)) \cdot f'(x)$$

nên suy ra

$$h'(c) = (f(b) - f(a)) \cdot g'(c) - (g(b) - g(a)) \cdot f'(c) = 0$$

và ta có điều phải chứng minh.

❶ Theorem 1.13 (Quy tắc L'Hôpital)

Xét V là lân cận của điểm x_0 , $f(x)$ và $g(x)$ là hai hàm số liên tục trên V và có đạo hàm trên $V \setminus \{x_0\}$. Giả sử

$$f(x_0) = g(x_0) = 0, \lim_{x \rightarrow x_0} \frac{f'(x)}{g'(x)} = a.$$

Khi đó ta có

$$\lim_{x \rightarrow x_0} \frac{f(x)}{g(x)} = a.$$

Quy tắc L'Hôpital là công cụ khử dạng vô định $\frac{0}{0}$ rất hiệu quả. Tuy nhiên chúng ta cũng cần chú ý rằng nếu không phải dạng vô định thì quy tắc L'Hôpital không còn đúng.

Tính lồi, lõm và điểm uốn của đồ thị

❶ Definition 1.53 (Hàm lồi, lõm)

Giả sử \mathbb{I} là một khoảng trong \mathbb{R} . Hàm số f được gọi là **lõm** trên \mathbb{I} nếu và chỉ nếu với mọi $\alpha, \beta \geq 0$ mà $\alpha + \beta = 1$, ta đều có

$$f(\alpha x + \beta y) \leq \alpha f(x) + \beta f(y)$$

với mọi $x, y \geq 0$.

Trong trường hợp ngược lại (đổi chiều bất đẳng thức) thì ta nói f là hàm **lồi** trên \mathbb{I} .

Trong một số sách toán ở nước ngoài thì khái niệm lồi, lõm của hàm số được hiểu ngược lại. Khi đó hàm lõm ở định nghĩa trên được gọi là hàm *lồi (lồi dưới)* và hàm lồi ở định nghĩa trên gọi là hàm *lõm (lồi trên)*.

Ví dụ ở bài viết [Выпуклость и точки перегиба](#) (Tính lồi và điểm uốn) của Moscow State University (MSU) mang tên Lomonosov thì định nghĩa hàm lồi bên trên được MSU gọi là *lồi trên* (*выпуклая сверх*), còn định nghĩa hàm lõm bên trên được MSU gọi là *lồi dưới* (*выпуклая сниз*).

Ví dụ ở trang wikipedia [Concave function](#) thì định nghĩa hàm lồi bên trên (dấu \geq) được wikipedia định nghĩa là hàm lõm (concave). Tương tự, ở trang wikipedia [Convex function](#) thì định nghĩa hàm lõm bên trên (dấu \leq) được wikipedia định nghĩa là hàm lồi (convex).

Về mặt hình học, các bạn có thể xem $y = x^2$ là một ví dụ của hàm lõm và $y = -x^2$ là một ví dụ của hàm lồi. Sau đây chúng ta sẽ phân tích một số tính chất từ định nghĩa hàm lồi, lõm để đưa ra khía cạnh hình học này.

❶ Theorem 1.14

Giả sử $f(x)$ có đạo hàm trên \mathbb{I} . Khi đó $f(x)$ lõm trên \mathbb{I} khi và chỉ khi $f'(x)$ là hàm tăng không nghịch đảo (knn) trên \mathbb{I} .

❶ Chứng minh chiều thuận

Ta có $f(x)$ là hàm lõm trên \mathbb{I} và ta cần chứng minh $f'(x)$ tăng khanh trên \mathbb{I} , nghĩa là

$$f'(x_1) \leq f'(x_2)$$

với mọi $x_1, x_2 \in \mathbb{I}$ mà $x_1 < x_2$.

Từ định nghĩa hàm lõi, ta chọn $\alpha = \frac{x_2 - x}{x_2 - x_1}$ và $\beta = \frac{x - x_1}{x_2 - x_1}$ với $x_1 < x < x_2$ thì khi đó $\alpha, \beta > 0$ và $\alpha + \beta = 1$. Ta có

$$f(x) = f\left(\frac{x_2 - x}{x_2 - x_1}x_1 + \frac{x - x_1}{x_2 - x_1}x_2\right) \leq \frac{x_2 - x}{x_2 - x_1}f(x_1) + \frac{x - x_1}{x_2 - x_1}f(x_2).$$

Ta biến đổi bất đẳng thức như sau

$$\begin{aligned} (x_2 - x_1)f(x) &\leq (x_2 - x)f(x_1) + (x - x_1)f(x_2) \\ \iff (x_2 - x + x - x_1)f(x) &\leq (x_2 - x)f(x_1) + (x - x_1)f(x_2) \\ \iff (x_2 - x)[f(x) - f(x_1)] &\leq (x - x_1)[f(x_2) - f(x)] \\ \iff \frac{f(x) - f(x_1)}{x - x_1} &\leq \frac{f(x_2) - f(x)}{x_2 - x}. \end{aligned} \tag{2.4}$$

Cho $x \rightarrow x_1^+$ thì vế phải BĐT là đạo hàm phải tại x_1 , nghĩa là

$$f'(x_1) \leq \frac{f(x_2) - f(x_1)}{x_2 - x_1}.$$

Tương tự, cho $x \rightarrow x_2^-$ thì vế trái BĐT là đạo hàm trái tại x_2 , nghĩa là

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1} \leq f'(x_2).$$

Từ hai BĐT vừa rồi suy ra $f'(x_1) \leq f'(x_2)$. Ta có điều phải chứng minh.

❶ Chứng minh chiều ngược

Ở phần trên ta dùng biến đổi tương đương để có (2.4). Như vậy nếu chứng minh được (2.4) thì cũng đồng nghĩa $f(x)$ là hàm lõm.

Theo định lí Lagrange tồn tại các số x_3 và x_4 sao cho $x_1 < x_3 < x < x_4 < x_2$ và

$$\frac{f(x) - f(x_1)}{x - x_1} = f'(x_3), \quad \frac{f(x_2) - f(x)}{x_2 - x} = f'(x_4).$$

Vì $x_3 < x_4$ và $f'(x)$ tăng khanh trên \mathbb{I} nên ta có $f'(x_3) \leq f'(x_4)$, suy ra

$$\frac{f(x) - f(x_1)}{x - x_1} \leq \frac{f(x_2) - f(x)}{x_2 - x}.$$

Như vậy (2.4) được chứng minh.

Một hệ quả quan trọng hay được sử dụng của định lí này như sau.

1 Corollary 1.1

Nếu $f(x)$ liên tục trên \mathbb{I} và có đạo hàm cấp hai trên \mathbb{I} thì $f(x)$ lõm trên \mathbb{I} khi và chỉ khi $f''(x) \geq 0$ với mọi $x \in \mathbb{I}$.

Một số tính chất của hàm lồi và hàm lõm là:

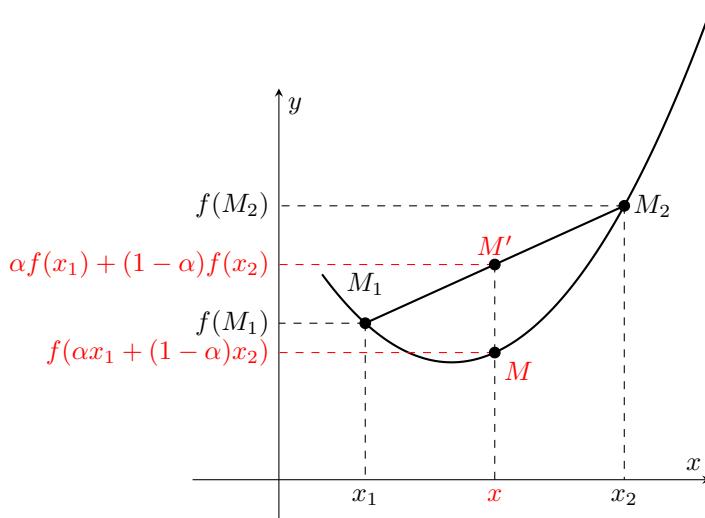
1. Nếu $f(x)$ có đạo hàm và lõm trên khoảng \mathbb{I} thì tiếp tuyến tại mọi điểm thuộc \mathbb{I} của nó đều nằm phía dưới đồ thị của hàm $f(x)$.
2. Nếu $f(x)$ có đạo hàm và lồi trên khoảng \mathbb{I} thì tiếp tuyến tại mọi điểm thuộc \mathbb{I} của nó đều nằm phía trên đồ thị của hàm $f(x)$.
3. Nếu $f(x)$ là hàm lõm trên đoạn $[a; b]$ thì ta có $f(x) \leq \max\{f(a), f(b)\}$ với mọi $x \in [a; b]$. Nói cách khác, giá trị lớn nhất của $f(x)$ trên $[a; b]$ đạt được tại đầu mút của đoạn $[a; b]$.
4. Tương tự, nếu $f(x)$ là hàm lồi trên đoạn $[a; b]$ thì giá trị nhỏ nhất của nó trên đoạn này đạt được tại đầu mút của đoạn $[a; b]$.

Ý nghĩa hình học của hàm lõm

Giả sử hàm $f(x)$ lõm trên \mathbb{I} . Lấy bất kì $x_1, x_2 \in \mathbb{I}$ với $x_1 < x_2$.

Gọi $M_1(x_1, f(x_1))$ và $M_2(x_2, f(x_2))$ là hai điểm trên đồ thị hàm số $y = f(x)$.

Khi đó điểm $M'(x, y)$ nằm trên đoạn thẳng M_1M_2 khi và chỉ khi có số $\alpha \in [0, 1]$ sao cho $\overrightarrow{M_2M'} = \alpha \overrightarrow{M_2M_1}$ như trên hình vẽ sau.



Điều kiện này tương đương với

$$\begin{cases} x - x_2 = \alpha(x_1 - x_2), \\ y - f(x_2) = \alpha[f(x_1) - f(x_2)]. \end{cases}$$

hay

$$\begin{cases} x = \alpha x_1 + (1 - \alpha)x_2, \\ y = \alpha f(x_1) + (1 - \alpha)f(x_2). \end{cases}$$

Vì $f(x)$ lõm trên \mathbb{I} nên ta có

$$y_M = f(\alpha x_1 + (1 - \alpha)x_2) \leq \alpha f(x_1) + (1 - \alpha)f(x_2) = y'_M.$$

Kết quả này cho chúng ta kết quả:

Hàm số $f(x)$ lõm trên \mathbb{I} khi và chỉ khi với mọi cặp điểm M_1, M_2 thuộc đồ thị hàm số $y = f(x)$, cung M_1M_2 của đồ thị luôn nằm phía dưới đoạn thẳng M_1M_2 .

Điểm uốn của đồ thị

1 Definition 1.54 (Điểm uốn của đồ thị)

Giả sử hàm số $f(x)$ có đạo hàm trên khoảng $(a; b)$ chứa x_0 . Nếu đồ thị (C) của hàm số $y = f(x)$ lõm trên một trong hai khoảng (a, x_0) , (x_0, b) và lồi trên khoảng còn lại thì $U(x_0, f(x_0))$ được gọi là **điểm uốn** (hay **inflection point**, **точка перегиба**) của đồ thị (C) .

Nói cách khác, qua điểm uốn lồi thành lõm hoặc lõm thành lồi.

1 Remark 1.13

Tiếp tuyến tại điểm uốn đi xuyên qua đồ thị.

Điều này có nghĩa là trên một trong hai khoảng $(a; x_0)$, $(x_0; b)$ thì tiếp tuyến tại x_0 nằm phía dưới đồ thị và trên khoảng còn lại thì tiếp tuyến đó nằm phía trên đồ thị.

Từ kết quả trên về tính lồi, lõm chúng ta có định lí sau là công cụ hữu ích để xác định điểm uốn của đồ thị.

1 Theorem 1.15

Giả sử hàm số $f(x)$ có đạo hàm cấp hai trên khoảng \mathbb{I} chứa điểm x_0 . Nếu $f''(x_0) = 0$ và $f''(x)$ đổi dấu khi x đi qua điểm x_0 thì $U(x_0, f(x_0))$ là một điểm uốn của đồ thị hàm số $y = f(x)$.

2.4.2 Phương trình vi phân

Phương trình vi phân bậc nhất

Phương trình vi phân bậc nhất có dạng:

$$P(x, y) dx + Q(x, y) dy = 0.$$

Nghiệm tổng quát có dạng $y = \varphi(x, c)$ khả vi và thỏa mãn điều kiện của phương trình vi phân.

1. Hàm $\varphi(x, c)$ là nghiệm của phương trình vi phân với hằng số c .
2. Nếu $y(x_0) = y_0$ thì có thể tìm được c_0 . Khi đó nghiệm được gọi là **nghiệm riêng** $y = \varphi(x, c_0)$.

❶ Definition 2.10 (Phương trình tách biến)

Phương trình tách biến (hay разделяющие переменные) là phương trình có dạng:

$$P_1(x) \cdot Q_1(y) dx + P_2(x) \cdot Q_2(y) dy = 0.$$

Để giải phương trình tách biến ta chia hai vế cho $Q_1(y) \cdot P_2(x) \neq 0$ và giải

$$\int \frac{P_1(x)}{P_2(x)} dx + \int \frac{Q_2(y)}{Q_1(y)} dy = 0.$$

Nếu $Q_1(y) \cdot P_2(x) = 0$ thì ta giải từng hàm riêng $Q_1(y) = 0$ hoặc $P_2(x) = 0$.

❷ Definition 2.11 (Hàm thuần nhất bậc n)

Hàm thuần nhất bậc n với tham số λ bất kì ta luôn có $f(\lambda x, \lambda y) = \lambda^n f(x, y)$.

Ví dụ, $f(x, y) = x^2 - 2xy$. Với các hàm thuần nhất ta có định nghĩa phương trình vi phân thuần nhất.

❸ Definition 2.12 (Phương trình vi phân thuần nhất)

Phương trình vi phân $y' = f(x, y)$ được gọi là thuần nhất nếu hàm $f(x, y)$ là hàm thuần nhất bậc 0. Khi đó $y' = \varphi\left(\frac{y}{x}\right)$.

Để giải dạng này ta đặt $\frac{y}{x} = u$. Như vậy $y = ux$ và $y' = u'x + ux'$.

Phương trình tuyến tính và phương trình Bernoulli

❹ Definition 2.13 (Phương trình vi phân tuyến tính)

Phương trình vi phân được gọi là tuyến tính bậc nhất nếu nó có dạng $y' + p(x) \cdot y = g(x)$.

Để giải phương trình tuyến tính ta có hai phương pháp là phương pháp Bernoulli và phương pháp Lagrange.

Đối với phương pháp Bernoulli, ta đặt $y = uv$ với $u = u(x)$ và $v = v(x)$ là một hàm được chọn từ lớp các hàm thỏa mãn với hằng số cố định. Khi đó $y' = u'v + uv'$.

Phương trình vi phân khi này có dạng:

$$\begin{aligned} y' + p(x) \cdot y &= g(x) \\ \Leftrightarrow u'v + v'u + p(x) \cdot uv &= g(x) \\ \Leftrightarrow u'v + u(v' + p(x) \cdot v) &= g(x) \end{aligned}$$

Ta chọn v sao cho $v' + p(x) \cdot v = 0$. Điều này tương đương với $\frac{dv}{dx} + p(x) \cdot v = 0$ nên $\ln|v| = - \int p(x) dx + \ln|C|$.

Ở đây C là hằng số.

Để đơn giản, ta chọn $C = 1$ thì $v = e^{- \int p(x) dx}$.

Suy ra $u'v = u'e^{-\int p(x) dx} = g(x)$ và biến đổi

$$\begin{aligned} \frac{du}{dx} \cdot e^{-\int p(x) dx} &= g(x) \\ \iff u &= \int g(x) \cdot e^{-\int p(x) dx} dx + C \\ \iff y &= uv = \dots \end{aligned}$$

Đối với phương pháp Lagrange, ta xét phương trình vi phân tuyến tính thuần nhất tương ứng với $y' + p(x) \cdot y = g(x)$ là $y' + p(x) \cdot y = 0$.

Khi đó ta biến đổi $\frac{dy}{dx} = -p(x) \cdot y$ và giải được $y = ce^{-\int p(x) dx}$ với c là hằng số.

Ta thay c bởi $c(x)$ là hàm theo x :

$$\begin{aligned} y &= c(x) \cdot e^{-\int p(x) dx} \\ \iff y' &= c'(x) \cdot e^{-\int p(x) dx} + c(x) \cdot e^{-\int p(x) dx} \cdot (-p(x)) \end{aligned}$$

và thay vào phương trình vi phân ban đầu:

$$\begin{aligned} c'(x) \cdot e^{-\int p(x) dx} + c(x) \cdot e^{-\int p(x) dx} \cdot (-p(x)) + c(x) \cdot e^{-\int p(x) dx} \cdot p(x) &= g(x) \\ \iff c'(x) \cdot e^{-\int p(x) dx} &= g(x) \\ \iff dc(x) &= e^{\int p(x) dx} \cdot g(x) dx. \end{aligned}$$

Tới đây ta tìm được $c(x)$ để có y .

Example 2.5

Giải phương trình vi phân

$$y dx = (y^2 - x) dy.$$

Đầu tiên ta biến đổi phương trình

$$\begin{aligned} y dx &= (y^2 - x) dy \iff \frac{dx}{dy} + \frac{x}{y} = y \\ \iff x' + \frac{1}{y} &= y. \end{aligned} \tag{2.5}$$

Đối với phương pháp Bernoulli, đặt $x = uv$ với $u = u(y)$ và $v = v(y)$. Khi đó ta có $x' = u'v + uv'$.

Từ (2.5) suy ra

$$\begin{aligned} u'v + v'u + \frac{1}{y} \cdot uv &= y \\ \iff u'v + u \left(v' + \frac{1}{y} \cdot v \right) &= y. \end{aligned}$$

Ta tìm v sao cho $v' + \frac{1}{y}v = 0$, tương đương với

$$\frac{dv}{dy} = -\frac{v}{y} \iff \dots \iff v = \frac{1}{y}.$$

Thay vào (2.5) ta được

$$u'v = u' \cdot \frac{1}{y} = y \iff u' = y^2 \implies u = \frac{y^3}{3} + C.$$

Như vậy

$$x = uv = \left(\frac{y^3}{3} + c \right) \cdot \frac{1}{y} = \frac{y^2}{3} + \frac{C}{y}.$$

Đối với phương pháp Lagrange, ta xét dạng tuyến tính thuận nhất là $x' + \frac{1}{y} \cdot x = 0$.

Khi đó

$$\frac{dx}{x} = -\frac{dy}{y} \iff x = \frac{c}{y}$$

với c là hằng số. Bây giờ thay c thành $c(y)$ ta có $x = \frac{c(y)}{y}$.

Thay vào (2.5) ta có

$$\begin{aligned} & \frac{c'(y)y - c(y)}{y^2} + \frac{1}{y} \cdot \frac{c(y)}{y} = y \\ \iff & \frac{c'(y)}{y} - \cancel{\frac{c(y)}{y^2}} + \cancel{\frac{c(y)}{y^2}} = y \\ \implies & c(y) = \int y^2 dy = \frac{y^3}{3} + C. \end{aligned}$$

Như vậy ta có kết quả

$$x = \frac{c(y)}{y} = \left(\frac{y^3}{3} + C \right) \cdot \frac{1}{y} = \frac{y^2}{3} + \frac{C}{y}.$$

❶ Definition 2.14 (Phương trình Bernoulli)

Phương trình Bernoulli là phương trình có dạng:

$$y' + p(x) \cdot y = g(x) \cdot y^n, \quad n \in \mathbb{N}, n \neq 0, n \neq 1.$$

Khi $n = 0$ thì phương trình trở thành phương trình tuyến tính.

Để giải phương trình Bernoulli ta chia hai vế cho y^n thì được:

$$\frac{y'}{y^n} + \frac{p(x)}{y^{n-1}} = g(x).$$

Đặt $\frac{1}{y^{n-1}} = z$ thì

$$z' = \frac{dz}{dx} = (1-n) \frac{1}{y^n} y'.$$

Như vậy $\frac{1}{y^n} \cdot y' = \frac{z'}{1-n}$ và thay vào phương trình ban đầu ta có

$$\frac{1}{1-n} z' + p(x) \cdot z = g(x).$$

Từ đây ta giải phương trình tuyến tính.

Phương trình vi phân toàn phần. Nhân tử tích phân

i Definition 2.15 (Phương trình vi phân toàn phần)

Phương trình vi phân được gọi là **toàn phần** (hay **уравнение в полных дифференциалах**) nếu về trái có vi phân toàn phần là hàm $u(x, y)$, nghĩa là:

$$P(x, y) dx + Q(x, y) dy = du(x, y).$$

i Theorem 2.4

Điều kiện cần và đủ để biểu thức

$$\Delta = P(x, y) dx + Q(x, y) dy,$$

với $P(x, y)$ và $Q(x, y)$ và đạo hàm riêng $\frac{\partial P}{\partial y}$ và $\frac{\partial Q}{\partial x}$ liên tục trên tập D nào đó, là phương trình vi phân toàn phần:

$$\frac{\partial P}{\partial y} = \frac{\partial Q}{\partial x}.$$

i Chứng minh

Để chứng minh điều kiện cần, đặt Δ là biểu thức có vi phân toàn phần, nghĩa là

$$\begin{aligned} P(x, y) dx + Q(x, y) dy &= du(x, y) \\ \iff du(x, y) &= \frac{\partial u}{\partial x} dx + \frac{\partial u}{\partial y} dy, \end{aligned}$$

với $P(x, y) = \frac{\partial u}{\partial x}$ và $Q(x, y) = \frac{\partial u}{\partial y}$.

Suy ra $\frac{\partial P}{\partial y} = \frac{\partial^2 u}{\partial x \partial y}$ và $\frac{\partial Q}{\partial x} = \frac{\partial^2 u}{\partial y \partial x}$.

Khi đó $\frac{\partial P}{\partial y} = \frac{\partial Q}{\partial x}$ và ta có điều phải chứng minh.

Để chứng minh điều kiện đủ, trên tập D ta có $\frac{\partial P}{\partial y} = \frac{\partial Q}{\partial x}$.

Lúc này ta chứng minh tồn tại hàm $u(x, y)$ trên D mà

$$du(x, y) = P(x, y) dx + Q(x, y) dy.$$

Giả sử ngược lại, $\frac{\partial u}{\partial x} = P(x, y)$ và $\frac{\partial u}{\partial y} = Q(x, y)$.

Ở đây, ta cố định y và lấy tích phân theo x thì được:

$$u(x, y) = \int P(x, y) dx + \varphi(y),$$

với $c = \varphi(y)$ là hằng số (do y cố định nên $\varphi(y)$ cũng cố định). Suy ra ta có:

$$\begin{aligned} \frac{\partial u}{\partial y} &= \left(\int P(x, y) dx \right)'_y + \varphi'(y) \\ \iff Q(x, y) &= \left(\int P(x, y) dx \right)'_y + \varphi'(y) \\ \iff \varphi'(y) &= Q(x, y) - \left(\int P(x, y) dx \right)'_y. \end{aligned}$$

Vẽ phải khi đạo hàm theo x là:

$$\begin{aligned} &\frac{\partial}{\partial x} \left(Q(x, y) - \frac{\partial}{\partial y} \left(\int P(x, y) dx \right) \right) \\ &= \frac{\partial Q}{\partial x} - \frac{\partial}{\partial x} \left(\frac{\partial}{\partial y} \left(\int P(x, y) dx \right) \right) \\ &= \frac{\partial Q}{\partial x} - \frac{\partial}{\partial y} \left(\frac{\partial}{\partial x} \left(\int P(x, y) dx \right) \right) \\ &= \frac{\partial Q}{\partial x} - \frac{\partial}{\partial y}(P) = 0, \end{aligned}$$

từ đây suy ra:

$$\varphi(y) = \int \left(Q(x, y) - \frac{\partial}{\partial y} \left(\int P(x, y) dx \right) \right) dy + c,$$

với c là hằng số.

Khi điều kiện $\frac{\partial P}{\partial y} = \frac{\partial Q}{\partial x}$ không thỏa mãn, ta nhân thêm hàm $t(x, y)$ để thỏa điều kiện:

$$\frac{\partial}{\partial y} (t(x, y) \cdot P(x, y)) = \frac{\partial}{\partial x} (t(x, y) \cdot Q(x, y)).$$

Hàm $t(x, y)$ khi đó được gọi là **nhân tử tích phân** (hay **интегрирующий множитель**). Lúc này phương trình biến đổi thành:

$$\begin{aligned} \frac{\partial t}{\partial y} \cdot P + \frac{\partial P}{\partial y} \cdot t &= \frac{\partial t}{\partial x} \cdot Q + \frac{\partial Q}{\partial x} \cdot t \\ \iff \frac{\partial t}{\partial y} \cdot P - \frac{\partial t}{\partial x} \cdot Q &= t \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right). \end{aligned}$$

Lúc này ta chỉ cần tìm hàm t chỉ phụ thuộc x hoặc y . Ví dụ với $t = t(x)$ ta có thể biến đổi:

$$\begin{aligned} -\frac{dt}{dx} Q &= t \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) \\ \iff \frac{dt}{t} &= \frac{\partial P / \partial y - \partial Q / \partial x}{Q} dx \\ \iff t(x) &= \exp \left(\int \frac{\partial P / \partial y - \partial Q / \partial x}{Q} dx \right). \end{aligned}$$

Tương tự, $t(y) = \exp \left(\int \frac{\partial Q/\partial x - \partial P/\partial y}{P} dy \right)$.

Phương trình Lagrange và Clairaut

Definition 2.16 (Phương trình Lagrange)

Phương trình Lagrange (уравнение Лагранжа) là phương trình có dạng:

$$y = x\varphi(y') + \psi(y'),$$

với φ và ψ là hai hàm số với $y' = \frac{dy}{dx}$.

Để giải phương trình Lagrange ta đặt $p = y'$. Khi đó $y = x\varphi(p) + \psi(p)$. Lấy vi phân hai vế ta có:

$$\begin{aligned} \frac{dy}{dx} &= \varphi(p) + x\varphi'(p)p' + \psi'(p)p' \\ &= \varphi(p) + x\varphi'(p)\frac{dp}{dx} + \psi'(p)\frac{dp}{dx}. \end{aligned}$$

Thay vào điều kiện ban đầu ta được:

$$(p - \varphi(p))\frac{dx}{dp} - x\varphi'(x) = \psi'(p).$$

Đây là phương trình vi phân tuyến tính theo $x = x(p)$.

Giải phương trình tìm được $x = \lambda(p, c)$ với c là hằng số, thay lại điều kiện $p = y'$ tìm được $y = \gamma(x, c)$.

Chú ý rằng khi thay vào điều kiện ban đầu ta chia cho dp nên trước đó phải xét trường hợp $dp = 0$, nói cách khác $p = p_0$ là hằng số và $p - \varphi(p) = 0$.

Nghiệm $y = x\varphi(p_0) + \psi(p_0)$ gọi là **nghiệm đặc trưng** (hay **особое решение**).

Definition 2.17 (Phương trình Clairaut)

Phương trình Clairaut (уравнение Клеро) có dạng:

$$y = xy' + \psi(y').$$

Đây là phương trình Lagrange khi $\varphi(y') = y'$.

Để giải phương trình này, đặt $y' = p$ thì $y = xp + \psi(p)$. Khi đó:

$$\begin{aligned} \frac{dy}{dx} &= p + xp' + \psi'(p)p' \\ \iff p &= p + x\frac{dp}{dx} + \psi'(p)\frac{dp}{dx} \\ \iff (x + \psi'(p))\frac{dp}{dx} &= 0. \end{aligned}$$

Nếu $\frac{dp}{dx} = 0$ thì $p = c$ là hằng số. Nghiệm tổng quát là $y = cx + \psi(c)$.

Nếu $x + \psi'(p) = 0$ thì $x = -\psi'(p)$. Suy ra $y = xp + \psi(p)$ là nghiệm đặc trưng và không có dạng tổng quát.

2.4.3 Chuỗi số

Xét dãy số $\{a_n\}$. Đặt

$$S_n = a_1 + a_2 + \dots + a_n.$$

Khi đó $\{S_n\}$ là chuỗi số. Tương tự như sự hội tụ hoặc phân kỳ của dãy số, ta cũng quan tâm đến sự hội tụ và phân kỳ của chuỗi số.

i Definition 3.15 (Chuỗi số hội tụ)

Chuỗi số $\{S_n\}$ được gọi là **hội tụ** nếu tồn tại giới hạn hữu hạn $L = \lim_{n \rightarrow \infty} S_n$.

Ngược lại ta gọi là chuỗi phân kỳ, nghĩa là $\lim_{n \rightarrow \infty} S_n = \infty$ hoặc không tồn tại $\lim_{n \rightarrow \infty} S_n$.

i Example 3.14

Xét dãy số $a_n = \frac{1}{2^n}$ với $n = 1, 2, \dots$

Khi đó

$$\begin{aligned} S_n &= \frac{1}{2} + \frac{1}{2^2} + \dots + \frac{1}{2^n} \\ &= \frac{1}{2} \cdot \frac{1 - \frac{1}{2^n}}{1 - \frac{1}{2}} \longrightarrow \frac{1}{2} \cdot \frac{1}{1 - \frac{1}{2}} = 1 \end{aligned}$$

khi $n \rightarrow \infty$. Như vậy S_n là chuỗi hội tụ.

Tiêu chuẩn Cauchy

Theo định nghĩa, chuỗi số hội tụ khi tồn tại giới hạn hữu hạn. Do đó ta có thể viết theo ngôn ngữ $\delta - \varepsilon$ như đối với dãy số.

i Theorem 3.14 (Tiêu chuẩn Cauchy)

Chuỗi số $\sum_{i=1}^{\infty} a_i$ hội tụ nếu với mọi $\varepsilon > 0$, tồn tại $N \in \mathbb{N}$, sao cho với mọi $n \geq m \geq N$, ta có

$$\left| \sum_{i=n}^m a_i \right| < \varepsilon.$$

i Chứng minh

Do $\sum_{i=m}^n a_i = \sum_{i=1}^n a_i - \sum_{i=1}^{m-1} a_i$ nên chuỗi số hội tụ từ một điểm m nào đó trở đi thì chuỗi hội tụ. Tương tự cho phân kỳ.

❶ Corollary 3.2

Nếu chuỗi số $\sum_{n=1}^{\infty} a_i$ hội tụ thì $\lim_{n \rightarrow \infty} a_n = 1$.

❶ Chứng minh

Theo tiêu chuẩn Cauchy, với mọi $\varepsilon > 0$, do chuỗi hội tụ nên tồn tại $N \in \mathbb{N}$ sao cho với mọi $n \geq m \geq N$ ta có $\left| \sum_{i=m}^n a_i \right| < \varepsilon$.

Nếu ta chọn $m = n$ thì điều kiện trở thành với mọi $\varepsilon > 0$, tồn tại $N \in \mathbb{N}$ sao cho với mọi $n \geq N$ ta có $|a_n| < \varepsilon$. Nói cách khác $\lim_{n \rightarrow \infty} a_n = 0$ (định nghĩa giới hạn dãy số).

Dựa vào tiêu chuẩn Cauchy ta có một số tiêu chuẩn hội tụ hữu dụng như sau.

❶ Theorem 3.15 (Tiêu chuẩn thứ nhất về sự hội tụ)

Xét hai chuỗi số $\sum_{n=1}^{\infty} a_n$ và $\sum_{n=1}^{\infty} b_n$. Khi điều kiện $0 \leq a_n \leq b_n$ thỏa mãn, ta có các kết quả sau:

1. Nếu $\sum b_n$ hội tụ thì $\sum a_n$ cũng hội tụ.
2. Nếu $\sum a_n$ phân kỳ thì $\sum b_n$ cũng phân kỳ.

❶ Chứng minh

Ta thấy rằng nếu $\sum b_n$ hội tụ thì với mọi $\varepsilon > 0$, tồn tại $N \in \mathbb{N}$ sao cho với mọi $n \geq m \geq N$, $\left| \sum_{i=m}^n a_i \right| < \varepsilon$.

Do $0 \leq a_i \leq b_i$ nên $\left| \sum_{i=m}^n a_i \right| < \left| \sum_{i=m}^n b_i \right| < \varepsilon$. Như vậy chuỗi $\sum a_n$ cũng hội tụ.

❶ Theorem 3.16 (Tiêu chuẩn so sánh)

Xét hai chuỗi số dương $\sum_{n=1}^{\infty} a_n$ và $\sum_{n=1}^{\infty} b_n$. Nếu tồn tại giới hạn hữu hạn $\lim_{n \rightarrow \infty} \frac{a_n}{b_n} = L$ thì hai dãy số trên cùng hội tụ hoặc cùng phân kỳ.

❶ Chứng minh

Do dãy số $\frac{a_n}{b_n}$ có giới hạn hữu hạn L nên với mọi $\varepsilon > 0$, tồn tại $N \in \mathbb{N}$ sao cho với mọi $n \geq N$, ta có $\left| \frac{a_n}{b_n} - L \right| < \varepsilon$. Tương đương với $b_n(-\varepsilon + L) < a_n < b_n(\varepsilon + L)$. Từ tiêu chuẩn thứ nhất về sự hội tụ và

bất đẳng thức thứ hai suy ra nếu chuỗi $\sum b_n$ hội tụ thì chuỗi $\sum a_n$ cũng hội tụ.

Tương tự, với bất đẳng thức thứ nhất, nếu $\sum b_n$ phân kỳ thì $\sum a_n$ cũng phân kỳ.

Theorem 3.17 (Tiêu chuẩn tích phân Cauchy)

Xét chuỗi số dương $\sum_{n=1}^{\infty} a_n$. Nếu a_n là dãy số có dạng $a_n = f(n)$, ta chuyển qua xét hàm số $f(x)$.

Nếu hàm số $f(x)$ thỏa mãn:

1. $f(x)$ không tăng.
2. $\int_0^{\infty} f(x) dx$ hữu hạn.

Khi đó chuỗi số $\sum_{n=1}^{\infty} a_n$ hội tụ.

2.4.4 Đạo hàm

Đạo hàm một số hàm nhiều biến

Hàm số cho giá trị là số vô hướng

Giả sử ta có vector hàng $\mathbf{x} = (x_1, \dots, x_n)$ và hàm số f có biến là vector \mathbf{x} . Nói cách khác là $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $f(\mathbf{x}) = f(x_1, \dots, x_n)$.

Khi đó đạo hàm riêng của hàm f theo vector \mathbf{x} cũng là một vector (nếu \mathbf{x} là vector hàng thì đạo hàm riêng cũng là vector hàng và ngược lại) và được kí hiệu

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} & \dots & \frac{\partial f}{\partial x_n} \end{pmatrix}$$

Ví dụ, đối với hàm tuyến tính

$$f(\mathbf{x}) = a_1 x_1 + \dots + a_n x_n = \mathbf{a} \cdot \mathbf{x}^\top$$

thì ta thấy rằng $\frac{\partial f}{\partial x_i} = a_i$. Khi đó

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} & \dots & \frac{\partial f}{\partial x_n} \end{pmatrix} = (a_1, \dots, a_n) = \mathbf{a}.$$

Ta thấy rằng $f(\mathbf{x}) = \mathbf{a} \cdot \mathbf{x}^\top = \mathbf{x} \cdot \mathbf{a}^\top$. Do đó

$$\nabla(\mathbf{a} \cdot \mathbf{x}^\top) = \nabla(\mathbf{x} \cdot \mathbf{a}^\top) = \mathbf{a}.$$

Đạo hàm riêng cấp hai được cho bởi ma trận được gọi là ma trận Hessian.

$$\nabla^2 f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \ddots & \cdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}.$$

Theo tính chất của đạo hàm riêng cấp hai có thể thấy ma trận trên là ma trận đối xứng.

Nếu đầu vào là một ma trận, hay $f : \mathbb{R}^{n \times m} \rightarrow \mathbb{R}$, $f(\mathbf{X})$ thì ta làm tương tự

Giả sử

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \cdots & \cdots & \ddots & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}.$$

Khi đó đạo hàm của hàm f theo ma trận \mathbf{X} là

$$\nabla f(\mathbf{X}) = \begin{pmatrix} \frac{\partial f}{\partial x_{11}} & \frac{\partial f}{\partial x_{12}} & \cdots & \frac{\partial f}{\partial x_{1m}} \\ \frac{\partial f}{\partial x_{21}} & \frac{\partial f}{\partial x_{22}} & \cdots & \frac{\partial f}{\partial x_{2m}} \\ \cdots & \cdots & \ddots & \cdots \\ \frac{\partial f}{\partial x_{n1}} & \frac{\partial f}{\partial x_{n2}} & \cdots & \frac{\partial f}{\partial x_{nm}} \end{pmatrix}.$$

Như vậy đạo hàm theo ma trận cũng là ma trận cùng cỡ với ma trận đầu vào.

Hàm số cho giá trị là vector

Xét hàm vector

$$F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x}))$$

với $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$ và các hàm $f_i(\mathbf{x})$ là hàm từ \mathbb{R}^n tới \mathbb{R} . Khi đó hàm vector F là hàm từ \mathbb{R}^n tới \mathbb{R}^m .

Nếu f_i là các hàm tuyến tính như trên thì hàm F là một ánh xạ tuyến tính, hay tương đương với phép nhân ma trận $F(\mathbf{x}) = \mathbf{x} \cdot \mathbf{A}$. Ở đây \mathbf{x} là vector hàng, còn \mathbf{A} là ma trận $n \times m$.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \cdots & \cdots & \ddots & \cdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{pmatrix}.$$

Ở đây,

$$f(\mathbf{x}) = f_i(x_1, x_2, \dots, x_n) = a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n.$$

Nếu đặt $\mathbf{a}_i = (a_{i1}, a_{i2}, \dots, a_{in})$ thì ma trận \mathbf{A} có các cột là \mathbf{a}_i^\top . Nói cách khác

$$\mathbf{A} = (\mathbf{a}_1^\top \quad \mathbf{a}_2^\top \quad \cdots \quad \mathbf{a}_m^\top).$$

Nếu ta xét từng cột của ma trận \mathbf{A} thì hoàn toàn giống trường hợp trên. Giả sử với cột đầu tiên (ứng với f_1) ta có

$$f_1(\mathbf{x}) = (x_1 \quad x_2 \quad \cdots \quad x_n) \cdot \begin{pmatrix} a_{11} \\ a_{12} \\ \vdots \\ a_{1n} \end{pmatrix} = \mathbf{x} \cdot \mathbf{a}_1^\top.$$

Đạo hàm của f_1 theo vector \mathbf{x} là

$$\nabla f_1(\mathbf{x}) = (a_{11} \quad a_{12} \quad \cdots \quad a_{1n}) = \mathbf{a}_1.$$

Xếp các hàm f_i từ trên xuống dưới, ta có được đạo hàm của hàm F theo vector \mathbf{x} là

$$\nabla F(\mathbf{x}) = \begin{pmatrix} \nabla f_1(\mathbf{x}) \\ \nabla f_2(\mathbf{x}) \\ \vdots \\ \nabla f_m(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_m \end{pmatrix} = \mathbf{A}^\top$$

2.4.5 Tích phân

Tích phân đường

Tích phân đường trên mặt phẳng

Tích phân đường dùng để tính độ dài đường cong $f(x)$ trên đoạn $[a; b]$ nào đó thuộc tập xác định.

Sau đây mình sẽ dùng tổng vô hạn để giải thích một số công thức tích phân đường được học ở trường.

Giả sử mình chia đoạn $[a; b]$ thành n phần bằng nhau

$$a = x_0 < x_1 < \cdots < x_n = b$$

$$\text{với } x_{i+1} - x_i = \frac{b-a}{n}.$$

Gọi các điểm

$$L_0 = (x_0, f(x_0) = y_0), L_1 = (x_1, f(x_1) = y_1), \dots, L_n = (x_n, f(x_n) = y_n).$$

Khi đó độ dài đường cong là tổng độ dài các đoạn thẳng $L_0L_1, L_1L_2, \dots, L_{n-1}L_n$ khi n tiến tới vô cùng.

Độ dài đoạn thẳng $L_{i-1}L_i$ là khoảng cách từ điểm $(x_{i-1}, f(x_{i-1}))$ tới $(x_i, f(x_i))$, nghĩa là

$$L_{i-1}L_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} = |x_i - x_{i-1}| \cdot \sqrt{1 + \left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right)^2}.$$

Khi n tiến tới vô cùng thì $\Delta x = x_i - x_{i-1} = \frac{b-a}{n}$ tiến tới 0. Các bạn có thấy biểu thức $\frac{y_i - y_{i-1}}{x_i - x_{i-1}}$ khi $x_i - x_{i-1}$ tiến tới 0 quen không? Chính là định nghĩa đạo hàm $f'(x) = \lim_{\Delta x \rightarrow 0} \Delta y / \Delta x$ mà chúng ta học ở phổ thông.

Kí hiệu l là độ dài đường cong trên. Do l là tổng của những đoạn $L_{i-1}L_i$ cực nhỏ nên có thể nói

$$l = \underbrace{\Delta l_1}_{L_0L_1} + \underbrace{\Delta l_2}_{L_1L_2} + \cdots + \underbrace{\Delta l_n}_{L_{n-1}L_n}$$

khi n tiến tới vô cùng.

Điều này có nghĩa là nếu $y = f(x)$ thì chúng ta có

$$\Delta l_i = |x_i - x_{i-1}| \cdot \sqrt{1 + \left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right)^2} = \Delta x_i \cdot \sqrt{1 + (\Delta y_i / \Delta x_i)^2}$$

với $\Delta y_i = y_i - y_{i-1}$, và $\Delta x_i = x_i - x_{i-1}$ tiến về 0. Thay các Δ bởi vi phân ta sẽ có công thức

$$dl = \sqrt{1 + (dy/dx)^2} dx = \sqrt{1 + (f'(x))^2} dx.$$

Công thức này có thể dùng khi y phụ thuộc vào x hay $y = f(x)$.

Làm ngược lại quá trình trên, thay x thành y và y thành x chúng ta có công thức trên nhưng theo dy là

$$dl = \sqrt{1 + (f'(y))^2} dy.$$

Vậy còn trường hợp x và y là hai hàm số theo tham số t ? Ví dụ như cung tròn bán kính bằng 1 cho bởi $x = x(t) = \cos(t)$ và $y = y(t) = \sin(t)$, trong đó $t \in [a; b] \subset [0, 2\pi]$.

Cách tiếp cận vẫn như vậy, ta chia đoạn $[a; b]$ thành n phần bằng nhau

$$a = t_0 < t_1 < \dots < t_n = b$$

với $t_{i+1} - t_i = \frac{b-a}{n}$.

Khi đó mỗi đoạn thẳng $L_{i-1}L_i$ sẽ có dạng

$$\begin{aligned} \Delta l_i &= L_{i-1}L_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \\ &= (t_i - t_{i-1}) \cdot \sqrt{\left(\frac{x_i - x_{i-1}}{t_i - t_{i-1}}\right)^2 + \left(\frac{y_i - y_{i-1}}{t_i - t_{i-1}}\right)^2} \\ &= \Delta t_i \cdot \sqrt{(\Delta x_i / \Delta t_i)^2 + (\Delta y_i / \Delta t_i)^2}. \end{aligned}$$

Khi n tiến tới vô cực thì Δt_i tiến về 0, mà x và y là các hàm số theo biến t nên nói cách khác $\Delta x_i / \Delta t_i$ chính là đạo hàm của hàm số $x = x(t)$, tương tự $\Delta y_i / \Delta t_i$ là đạo hàm của hàm số $y = y(t)$.

Thay các Δ bởi vi phân thì ta có công thức

$$dl = \sqrt{(x'(t))^2 + (y'(t))^2} dt.$$

2.5 Xác suất thống kê

2.5.1 Xác suất thống kê

Xác suất là gì?

❶ Definition 1 (Định nghĩa cỗ điển của xác suất)

Định nghĩa thống kê của xác suất nói rằng, giả sử trong một phép thử có n khả năng có thể xảy ra. Xét một biến cố A xảy ra khi thực hiện phép thử có k khả năng xảy ra. Khi đó xác suất của biến cố A kí hiệu là $P(A)$ và được tính:

$$P(A) = \frac{k}{n}.$$

Dễ thấy, do biến cố A là một trường hợp nhỏ trong tổng thể tất cả trường hợp khi thực hiện phép thử, nên $0 \leq P(A) \leq 1$.

$$0 \leq P(A) \leq 1$$

với mọi biến cố A bất kì.

Example 1

Xét phép thử tung hai đồng xu. Gọi A là biến cố hai đồng xu cùng mặt.

Ta kí hiệu S là đồng xu sấp, N là đồng xu ngửa. Khi đó các trường hợp có thể xảy ra của phép thử là $S - S, S - N, N - S, N - N$ (4 trường hợp).

Trong khi đó, các trường hợp có thể xảy ra của biến cố A là $S - S, N - N$ (2 trường hợp).

Kết luận: $P(A) = \frac{2}{4} = \frac{1}{2}$.

Chúng ta gọi tập hợp tất cả các trường hợp khi thực hiện phép thử là **không gian mẫu** và kí hiệu là Ω . Mỗi phần tử trong không gian mẫu được gọi là **biến cố sơ cấp**. Trong ví dụ trên:

$$\Omega = \{S - S, S - N, N - S, N - N\}.$$

Tập hợp các trường hợp có thể xảy ra của biến cố gọi là **không gian biến cố** và kí hiệu là Ω_A . Trong ví dụ trên, $\Omega_A = \{S - S, N - N\}$.

Như vậy, $P(A) = \frac{|\Omega_A|}{|\Omega|}$.

Example 2

Tung hai con súc sắc cân đối và đồng chất. Tính xác suất tổng số nút của hai con súc sắc bằng 4.

Việc tung mỗi con súc sắc có 6 trường hợp. Do đó $|\Omega| = 6^2 = 36$.

Gọi A là biến cố tổng số nút của hai con súc sắc bằng 4. Ta có các trường hợp là $4 = 1 + 3 = 3 + 1 = 2 + 2$ (3 trường hợp).

Như vậy $|\Omega_A| = 3$ và $P(A) = \frac{3}{36} = \frac{1}{12}$.

Ba tiên đề về sự nhất quán của xác suất

Tiên đề 1. Nếu A là một sự kiện và kí hiệu $P(A)$ là **xác suất của A** thì:

$$0 \leq P(A) \leq 1.$$

Tiên đề 2. Nếu A là một sự kiện và kí hiệu \bar{A} là **sự kiện phủ định** của A thì:

$$P(A) + P(\bar{A}) = 1.$$

Tiên đề 3. Với hai sự kiện A và B , nếu hai sự kiện A và B không thể cùng xảy ra thì xác suất của sự kiện "xảy ra A hoặc B " bằng tổng các xác suất của A và B :

$$P(A \cap B) = 0 \Rightarrow P(A \cup B) = P(A) + P(B).$$

Không gian xác suất

Definition 2 (Không gian xác suất)

Một **không gian xác suất** là một tập hợp Ω cùng với:

1) Một họ \mathcal{S} các tập con của Ω , thỏa mãn các tính chất sau:

- $\Omega \in \mathcal{S}$;
- nếu $A, B \in \mathcal{S}$ thì $A \cup B \in \mathcal{S}$, $A \cap B \in \mathcal{S}$ và $\bar{A} = \Omega \setminus A \in \mathcal{S}$.

Một họ như vậy được gọi là một **đại số** các tập con của Ω .

Trong trường hợp Ω là tập có vô hạn phần tử thì ta cần thêm điều kiện sau: nếu A_i , $i = 1, 2, 3, \dots$ là một dãy vô hạn các phần tử của \mathcal{S} thì hợp $\bigcup_{i=1}^{\infty} A_i$ cũng thuộc họ \mathcal{S} .

Với điều kiện thêm này, \mathcal{S} được gọi là một **sigma-đại số**. Các phần tử của \mathcal{S} được gọi là tập hợp con **đo được** của không gian xác suất.

2) Một hàm số thực $P : \mathcal{S} \rightarrow \mathbb{R}$, được gọi là **phân bố xác suất** hay **độ đo xác suất** trên Ω , thỏa mãn các tính chất sau:

i) với mọi $A \in \mathcal{S}$ ta có:

$$0 \leq P(A) \leq 1$$

ii)

$$P(\emptyset) = 0, P(\Omega) = 1$$

iii) nếu $A \cap B = \emptyset$ thì:

$$P(A \cup B) = P(A) + P(B).$$

Tổng quát hơn, nếu A_i , với $i = 1, 2, 3, \dots$ là một dãy các tập hợp con đo được và chúng đôi một không giao nhau thì:

$$P\left(\bigcup_i A_i\right) = \sum_i P(A_i).$$

Một số lưu ý:

1. Không gian xác suất Ω còn được gọi là **không gian mẫu** (hay sample space) và nó là mô hình toán học trừu tượng cho vấn đề tính toán xác suất đang được quan tâm. Mỗi phần tử của Ω có thể được gọi là một **sự kiện thành phần** (hay **biến cố sơ cấp**, elementary event). Nếu A là một phần tử trong Ω thì ta cũng có thể viết $P(A)$ và hiểu là $P(\{A\})$, trong đó $\{A\}$ là tập con của Ω chứa duy nhất một phần tử A . Mỗi sự kiện là một tập con của Ω nên có thể chứa nhiều (thậm chí vô hạn) sự kiện thành phần. Không nhất thiết tập con nào của Ω cũng đo được (nằm trong họ \mathcal{S}) và chúng ta sẽ chỉ quan tâm đến các tập con đo được.
2. Trong toán học, một đại số là một tập hợp với các phép tính cộng, trừ và nhân (vành). Các tính chất của họ \mathcal{S} trong định nghĩa không gian xác suất khiến nó là một đại số theo nghĩa:
 - phần tử 0 trong \mathcal{S} là tập rỗng;
 - phần tử đơn vị trong \mathcal{S} là tập Ω ;

- phép nhân trong \mathcal{S} là phép giao $A \times B = A \cap B$;
- phép cộng trong \mathcal{S} là phép

$$A + B = (A \cup B) \setminus (A \cap B) = (A \setminus B) \cup (B \setminus A).$$

Đại số này có đặc số (số đặc trưng, characteristic) bằng 2, tức là $2A = A + A = 0$ với mọi A . Vì lý do này mà phép cộng và phép trừ là một. Chúng ta muốn \mathcal{S} là một đại số để thuận tiện thực hiện tính toán số học.

- Đẳng thức $P\left(\bigcup_i A_i\right) = \sum_i P(A_i)$ được gọi là **tính chất sigma** của xác suất. Tính chất sigma là **tính chất cộng vô hạn**: khi có một dãy vô hạn các tập con không giao nhau thì xác suất của hợp của chúng cũng bằng tổng vô hạn của các xác suất của các tập con. Tính chất sigma chính là tính chất cho phép chúng ta lấy giới hạn cho việc tính toán xác suất.

Ví dụ, nếu $A_1 \subset A_2 \subset \dots$ là một dãy tăng các tập con của Ω và $A = \lim_{n \rightarrow \infty} A_n = \bigcup_{n=1}^{\infty} A_n$, thì ta có thể viết $P(A) = \lim_{n \rightarrow \infty} P(A_n)$ bởi vì:

$$\begin{aligned} P(A) &= P(A_1 \cup \bigcup_{n=1}^{\infty} (A_{n+1} \setminus A_n)) = P(A_1) + \sum_{n=1}^{\infty} P(A_{n+1} \setminus A_n) \\ &= P(A_1) + \lim_{n \rightarrow \infty} P(A_{n+1} \setminus A_n) = P(A_1) + \lim_{n \rightarrow \infty} (P(A_{n+1}) - P(A_1)). \end{aligned}$$

Đẳng thức $P\left(\bigcup_i A_i\right) = \sum_i P(A_i)$ không được suy ra từ $P(A \cup B) = P(A) + P(B)$ mà là một tiên đề trong xác suất. Tiên đề này được đưa ra bởi nhà toán học người Nga Andrei Nikolaievich Kolmogorov.

Phép cộng xác suất mở rộng

Ở tiên đề 3 bên trên, hai biến cỗ khi đó được gọi là **xung khắc**, nghĩa là nếu biến cỗ này xảy ra thì biến cỗ kia chắc chắn không xảy ra. Nói cách khác giao của chúng bằng rỗng.

Ta còn có thể kí hiệu giao hai biến cỗ $P(A \cap B)$ là $P(AB)$.

Một trường hợp đơn giản nhất của hai biến cỗ xung khắc là **biến cỗ đối**.

Example 3

Tung một đồng xu và gọi A là biến cỗ đồng xu ra mặt ngửa. Khi đó biến cỗ đối của A , kí hiệu là \bar{A} là biến cỗ ra mặt sấp. Ở đây $A \cup \bar{A} = \Omega$ và $A \cap \bar{A} = \emptyset$.

Từ đó:

$$1 = P(\Omega) = P(A \cup \bar{A}) = P(A) + P(\bar{A}),$$

nói cách khác $P(\bar{A}) = 1 - P(A)$.

Xét hai tập hợp A và B . Số phần tử của phép hợp hai tập hợp trong trường hợp tổng quát được tính như sau:

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

Tương tự, xác suất của phép cộng xác suất đối với hai biến cỗ có giao khác rỗng là:

$$P(A \cup B) = P(A) + P(B) - P(AB).$$

Xét các tập hợp A_1, A_2, \dots, A_n . Khi đó, số phần tử khi hợp các tập hợp này là:

$$\begin{aligned} |A_1 \cup A_2 \cup \dots \cup A_n| &= |A_1| + |A_2| + \dots + |A_n| - \sum_{i,j} |A_i \cap A_j| \\ &\quad + \sum_{i,j,k} |A_i \cap A_j \cap A_k| + \dots \\ &= \sum_{i=1}^n (-1)^{i+1} \sum_{j_1, j_2, \dots, j_i} |A_{j_1} \cap A_{j_2} \cap \dots \cap A_{j_i}|. \end{aligned}$$

Tương tự, ta có phép cộng xác suất:

i Theorem 1 (Phép cộng xác suất mở rộng)

$$P(A_1 \cup A_2 \cup \dots \cup A_n) = \sum_{i=1}^n (-1)^{i+1} \sum_{j_1, j_2, \dots, j_i} P(A_{j_1} \cap A_{j_2} \cap \dots \cap A_{j_i}).$$

Mô hình xác suất với vô hạn các sự kiện

i Example 4 (Phân phối Poisson)

Giả sử tỉ lệ số khách hàng trung bình đến siêu thị trong một đơn vị thời gian cố định là λ .

Phân phối Poisson $P(n) = e^{-\lambda} \cdot \frac{\lambda^n}{n!}$ thể hiện xác suất có n khách hàng đến siêu thị theo tỉ lệ thời gian λ .

Ở phân phối Poisson, n nhận tất cả giá trị nguyên không âm $0, 1, \dots$ cũng như thỏa điều kiện:

$$\sum_{n=0}^{\infty} P(n) = \sum_{n=0}^{\infty} e^{-\lambda} \cdot \frac{\lambda^n}{n!} = e^{-\lambda} \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} = e^{-\lambda} \cdot e^{\lambda} = 1.$$

Ở biến đổi trên, $\sum_{n=0}^{\infty} \frac{\lambda^n}{n!} = e^{\lambda}$ là khai triển Taylor.

i Example 5

Giả sử ta biết rằng có một xe hơi X đang đậu trên một khúc phố Z và ta quan tâm đến vị trí của X trên khúc phố đó. Ta có thể mô hình X bằng một điểm và Z là một đoạn thẳng và lấy đoạn thẳng đó làm không gian xác suất $\Omega = [a; b]$, $a, b \in \mathbb{R}$, $a < b$ (mô hình xác suất liên tục này có lực lượng continuum, không đếm được).

Sự kiện "xe hơi đỗ chỗ nào đó trên khúc phố" chuyển thành sự kiện "điểm x nằm trong một đoạn thẳng con nào đó trên đoạn thẳng $\Omega = [a; b]$ ".

Ta có thể chọn phân bố xác suất đều trên $\Omega = [a; b]$ theo nghĩa sau: xác suất mỗi đoạn thẳng con trên Ω tỷ lệ thuận với độ dài của đoạn thẳng con đó, hay $P([c; d]) = (d - c)/(b - a)$.

Ánh xạ giữa các không gian xác suất

Definition 3 (Ánh xạ bảo toàn xác suất)

Một ánh xạ $\phi : (\Omega_1, P_1) \rightarrow (\Omega_2, P_2)$ từ một không gian xác suất (Ω_1, P_1) vào một không gian xác suất (Ω_2, P_2) được gọi là **ánh xạ bảo toàn xác suất** nếu nó bảo toàn độ đo xác suất, nghĩa là với mọi tập con $B \subset \Omega_2$ do được, ta có:

$$P_1(\phi^{-1}(B)) = P_2(B).$$

Hơn nữa, nếu ϕ là một song ánh modulo những tập có xác suất bằng 0, nghĩa là tồn tại các tập con $A \in \Omega_1$, $B \in \Omega_2$ sao cho $P_1(A) = P_2(B) = 0$ và $\phi : \Omega_1 \setminus A \rightarrow \Omega_2 \setminus B$ là song ánh bảo toàn xác suất, thì ϕ được gọi là **đẳng cấu xác suất**, và ta nói rằng (Ω_1, P_1) đẳng cấu xác suất với (Ω_2, P_2) .

Theorem 2

Nếu (Ω_1, P_1) là một không gian xác suất và $\phi : \Omega_1 \rightarrow \Omega_2$ là một ánh xạ tùy ý thì tồn tại một độ đo xác suất P_2 sao cho ánh xạ $\phi : (\Omega_1, P_1) \rightarrow (\Omega_2, P_2)$ là ánh xạ bảo toàn xác suất.

Ta xây dựng P_2 theo công thức: với mỗi tập con $B \subset \Omega_2$, nếu tồn tại $P_1(\phi^{-1}(B))$ thì ta đặt

$$P_2(B) = P_1(\phi^{-1}(B)).$$

Độ đo xác suất P_2 như trên gọi là **push-forward** của P_1 qua ánh xạ ϕ , hay **phân bố xác suất cảm sinh** từ P_1 qua ánh xạ ϕ .

Câu hỏi: chứng minh rằng quan hệ đẳng cấu xác suất giữa các không gian xác suất là một quan hệ tương đương.

Giải: vì ϕ là song ánh, đổi với tính phản xạ chúng ta lấy ánh xạ đồng nhất, tính đổi xứng thì ánh xạ ngược của song ánh (vẫn là song ánh), bắc cầu thì ta hợp hai song ánh vẫn là song ánh.

Tích của các không gian xác suất

Nếu (Ω_1, P_1) và (Ω_2, P_2) là hai không gian xác suất thì tích $\Omega_1 \times \Omega_2$ cũng có một độ đo xác suất P được xác định bởi P_1 và P_2 bằng công thức: nếu $A_1 \subset \Omega_1$ và $A_2 \subset \Omega_2$ nằm trong các sigma-đại số tương ứng của P_1 và P_2 thì:

$$P(A_1 \times A_2) = P_1(A_1) \times P_2(A_2).$$

Sigma-đại số của P chính là sigma-đại số sinh bởi các tập con của $\Omega_1 \times \Omega_2$ có dạng $A_1 \times A_2$ như trên.

Tương tự ta có thể định nghĩa tích trực tiếp của n không gian xác suất hay thập chí một dãy vô hạn các không gian xác suất.

Theorem 3

Hai phép chiếu tự nhiên từ tích $(\Omega_1, P_1) \times (\Omega_2, P_2)$ của hai không gian xác suất xuống (Ω_1, P_1) và (Ω_2, P_2) là hai ánh xạ bảo toàn xác suất.

Xác suất có điều kiện

Xác suất có điều kiện

i Definition 4 (Xác suất có điều kiện)

Xét hai biến cố A và B . Khi đó xác suất xảy ra của biến cố B với điều kiện biến cố A xảy ra là:

$$P(A|B) = \frac{P(AB)}{P(B)}.$$

Ý nghĩa của công thức trên có thể hiểu là, việc biến cố A xảy ra dựa trên cơ sở biến cố B đã xảy ra, do đó không gian mẫu sẽ giảm xuống còn B và biến cố giảm còn AB .

Dựa trên định nghĩa của xác suất có điều kiện có thể đưa ra nhận xét sau.

i Remark 1

Từ công thức trên có thể thấy sự tương đương:

$$P(AB) = P(B) \cdot P(A|B) = P(A) \cdot P(B|A).$$

Nhận xét trên cho thấy sự liên hệ của hai biến cố. Nói cách khác việc xảy ra của biến cố này sẽ ảnh hưởng đến biến cố kia và ngược lại.

Tổng quát, nếu n biến cố A_i , $i = 1, \dots, n$ không độc lập thì:

$$\begin{aligned} P(A_1 A_2 \cdots A_n) &= P(A_1) \cdot P(A_2 | A_1) \cdot P(A_3 | A_2 A_1) \cdots \\ &\quad P(A_n | A_1 A_2 \cdots A_{n-1}). \end{aligned}$$

Sử dụng nhận xét trên có thể chứng minh công thức này bằng quy nạp. Về mặt ý nghĩa, biến cố A_1 xảy ra đầu tiên. Tiếp theo đó A_2 xảy ra với điều kiện A_1 đã xảy ra. Tiếp theo nữa, A_3 xảy ra khi cả A_1 và A_2 xảy ra, chính là $A_1 A_2$.

Tương tự như vậy, A_i xảy ra với điều kiện tất cả A_1, \dots, A_{i-1} đều đã xảy ra, là biến cố giao $A_1 \dots A_{i-1}$.

Cũng từ nhận xét trên, các biến cố có vai trò như nhau nên việc đổi vị trí không thay đổi kết quả $P(A_1 \dots A_n)$.

Sự độc lập và phụ thuộc của các sự kiện

Nếu hai biến cố không ảnh hưởng việc xảy ra của nhau thì ta gọi là biến cố độc lập.

i Definition 5 (Biến cố độc lập)

Hai biến cố được gọi là **độc lập** nếu việc xảy ra của biến cố này không ảnh hưởng đến việc xảy ra của biến cố kia, hay:

$$P(A) = P(A|B) = P(AB)/P(B).$$

Viết cách khác là:

$$P(AB) = P(A) \cdot P(B).$$

Khi đó, giả sử ta có một họ \mathcal{M} các sự kiện.

i Definition 6 (Họ các sự kiện độc lập)

Họ \mathcal{M} được gọi là một **họ các sự kiện độc lập** nếu như với bất kì số tự nhiên k nào và bất kì sự kiện A_1, \dots, A_k khác nhau nào trong họ \mathcal{M} ta cũng có:

$$P(A_1 \cdots A_k) = P(A_1) \cdot P(A_2) \cdots P(A_k).$$

i Remark 2

Nếu ta có một họ các sự kiện độc lập thì các sự kiện trong họ độc lập đôi một với nhau. Nhưng ngược lại chưa chắc: có những họ không độc lập mà trong đó các sự kiện độc lập từng đôi một với nhau!

Công thức xác suất toàn phần

i Definition 7 (Hệ biến cố đầy đủ)

Xét phép thử có không gian mẫu là Ω . Một hệ các biến cố A_1, A_2, \dots, A_n là một **hệ biến cố đầy đủ** (hoặc **phân hoạch**) của Ω nếu chúng thỏa các điều kiện:

- $A_1 \cup A_2 \cup \cdots \cup A_n = \Omega$;
- $A_i \cap A_j = \emptyset$ với mọi $i \neq j$.

i Example 6

Một hệ biến cố đầy đủ đơn giản là $\Omega = \{A, \bar{A}\}$ gồm biến cố A và biến cố đối của A là \bar{A} .

Khi có một hệ biến cố đầy đủ, ta có thể tính xác suất của một biến cố bất kì nếu biết xác suất của các biến cố trong hệ biến cố đầy đủ và xác suất có điều kiện tương ứng.

i Theorem 4 (Công thức xác suất toàn phần)

Gọi A_1, A_2, \dots, A_n là một hệ biến cố đầy đủ của Ω . Khi đó, với biến cố B bất kì trong phép thử:

$$P(B) = P(A_1) \cdot P(B|A_1) + \cdots + P(A_n) \cdot P(B|A_n).$$

i Example 7

Đề bài. Trong một lớp học có 15 bạn nam và 10 bạn nữ. Trong đó có 5 bạn nam biết chơi bóng chuyền và 2 bạn nữ biết chơi bóng chuyền. Chọn ngẫu nhiên một bạn trong lớp, tính xác suất bạn đó biết chơi bóng chuyền.

Giải. Bài này có thể giải đơn giản bằng việc xác định số bạn biết chơi bóng chuyền là $5 + 2 = 7$ (5 nam và 2 nữ), trong khi không gian mẫu là $15 + 10 = 25$ nên kết quả là $\frac{7}{25}$.

Bài này nhằm đưa ra cái nhìn đơn giản về việc xác định điều kiện để thành lập hệ biến cố đầy đủ và giải bài toán.

Ở đây chúng ta cần tìm một hệ biến cố đầy đủ. Gọi A_1 là biến cố bạn được chọn là nam và A_2 là biến cố bạn được chọn là nữ.

Như vậy $\Omega = \{A_1, A_2\}$ thỏa định nghĩa về hệ biến cố đầy đủ.

$$\text{Để thấy } P(A_1) = \frac{15}{25} = \frac{3}{5} \text{ và } P(A_2) = \frac{10}{25} = \frac{2}{5}.$$

Tiếp theo, gọi B là biến cố bạn được chọn biết chơi bóng chuyền.

$$\text{Khi đó, xác suất một bạn biết chơi bóng chuyền với điều kiện bạn đó là nam bằng } P(B|A_1) = \frac{5}{15} = \frac{1}{3}.$$

$$\text{Tương tự, xác suất một bạn biết chơi bóng chuyền với điều kiện bạn đó là nữ bằng } P(B|A_2) = \frac{2}{10} = \frac{1}{5}.$$

Theo công thức xác suất toàn phần, xác suất bạn được chọn ngẫu nhiên biết chơi bóng chuyền là $P(B)$ và ta tính được

$$P(B) = P(A_1) \cdot P(B|A_1) + P(A_2) \cdot P(B|A_2) = \frac{3}{5} \cdot \frac{1}{3} + \frac{2}{5} \cdot \frac{1}{5} = \frac{7}{25}.$$

Ở đây, nếu chúng ta đảo điều kiện đề bài, ví dụ như bạn được chọn ngẫu nhiên là nữ *với điều kiện bạn đó biết chơi bóng chuyền* thì sao? Đề bài lúc này tương đương việc tính $P(A_2|B)$.

Để trả lời câu hỏi này chúng ta sử dụng công thức Bayes.

Công thức Bayes

❶ Theorem 5 (Công thức Bayes)

Xét hệ biến cố đầy đủ $\{A_1, A_2, \dots, A_n\}$ của không gian xác suất. Với biến cố B bất kì ta có **công thức Bayes**:

$$P(A_i|B) = \frac{P(A_i)P(B|A_i)}{\sum_{j=1}^n P(A_j)P(B|A_j)}$$

với mọi $1 \leq i \leq n$.

Công thức Bayes thực ra là công thức xác suất có điều kiện $P(B) \cdot P(A|B) = P(A) \cdot P(B|A)$, trong đó ta thay $P(B)$ bởi công thức xác suất toàn phần.

Như vậy, để trả lời cho câu hỏi trên, ta có

$$P(A_2|B) = \frac{P(A_2) \cdot P(B|A_2)}{P(B)} = \frac{\frac{2}{5} \cdot \frac{1}{5}}{\frac{7}{25}} = \frac{2}{7}.$$

Biến ngẫu nhiên**Biến ngẫu nhiên**

Xét phép thử với không gian mẫu Ω . Với mỗi biến cố sơ cấp $\omega \in \Omega$ ta liên kết với một số thực $\xi(\omega) \in \mathbb{R}$ thì ξ được gọi là **biến ngẫu nhiên** (hay **random variable**, BNN).

❶ Definition 8 (Biến ngẫu nhiên)

Biến ngẫu nhiên ξ của một phép thử với không gian mẫu Ω là ánh xạ:

$$\xi = \xi(\omega), \quad \omega \in \Omega.$$

Giá trị $\xi(\omega)$ được gọi là một giá trị của biến ngẫu nhiên ξ .

- Nếu $\xi(\omega)$ là một tập hữu hạn $\{\xi_1, \xi_2, \dots, \xi_n\}$ hay tập vô hạn đếm được thì ξ được gọi là **biến ngẫu nhiên rời rạc**.
- Nếu $\xi(\omega)$ là một khoảng của \mathbb{R} hay toàn bộ \mathbb{R} thì ξ được gọi là **biến ngẫu nhiên liên tục**.

Phân bố xác suất của biến ngẫu nhiên**❶ Definition 9 (Hàm phân phối xác suất)**

Hàm phân phối của biến ngẫu nhiên ξ là hàm số $F(x)$, xác định bởi:

$$F(x) = P(\xi \leq x), \quad x \in \mathbb{R}.$$

Ở đây ta viết gọn $P(\xi \leq x)$ từ $P(\{\omega : \xi(\omega) \leq x\})$. Tập hợp $\{\omega : \xi(\omega) \leq x\}$ có thể không thuộc một biến cố nào, do đó có thể là tập rỗng (ứng với xác suất là 0).

Tính chất của hàm phân phối

Tính chất 1. Hàm phân phối $F(x)$ không giảm trên mọi đoạn thẳng.

❶ Chứng minh

Đặt $x_2 > x_1$. Ta thấy rằng

$$\{\xi \leq x_2\} = \{\xi \leq x_1\} + \{x_1 < \xi \leq x_2\}.$$

Do đó nếu ta lấy xác suất thì cũng có

$$P(\xi \leq x_2) = P(\xi \leq x_1) + P(x_1 < \xi \leq x_2).$$

Xác suất luôn không âm, hay $P(x_1 < \xi \leq x_2) \geq 0$, suy ra $P(\xi \leq x_2) \geq P(\xi \leq x_1)$, hay $F(x_2) \geq F(x_1)$.

Tính chất 2. $\lim_{x \rightarrow -\infty} F(x) = 0$.

Tính chất 3. $\lim_{x \rightarrow +\infty} F(x) = 1$.

Tính chất 4. Hàm phân phối $F(x)$ liên tục phải trên toàn trực số.

Để chứng minh các tính chất 2, 3, 4 chúng ta cần các tiên đề của sự liên tục (continuity axioms) và sẽ không đề cập ở đây.

Biến ngẫu nhiên rời rạc

Cho BNN rời rạc

$$\xi = \xi(\omega), \xi = \{a_1, a_2, \dots, a_n, \dots\}.$$

Giả sử $a_1 < a_2 < \dots < a_n < \dots$ với xác suất tương ứng là $P(\xi = a_i) = p_i, i = 1, 2, \dots$

Ta có thể biểu diễn biến ngẫu nhiên và xác suất tương ứng của nó bằng bảng phân phối xác suất của ξ .

ξ	a_1	a_2	\dots	a_n	\dots
P	p_1	p_2	\dots	p_n	\dots

Rõ ràng rằng $p_n \geq 0$ với mọi n . Hơn nữa:

$$\sum_{n=1}^{\infty} p_n = 1.$$

Không gian mẫu lúc này là hợp của các tập biến ngẫu nhiên rời rạc:

$$\Omega = \{\xi = a_1\} \cup \{\xi = a_2\} \cup \dots$$

Các biến ngẫu nhiên xung khắc nhau (vì ξ không thể nhận hai giá trị khác nhau cùng lúc), do đó xác suất cả không gian mẫu là

$$1 = P(\Omega) = P(\xi = a_1) + P(\xi = a_2) + \dots = p_1 + p_2 + \dots$$

Definition 10 (Phân bố Bernoulli)

Phân bố xác suất cho không gian sinh bởi đúng một sự kiện A và phủ định của nó \bar{A} , hay $\Omega = \{A, \bar{A}\}$. Nếu xác suất xảy ra sự kiện A là p thì xác suất xảy ra \bar{A} là $1 - p$.

Phân bố Bernoulli được đặt tên theo nhà toán học Jacob Bernoulli (1654-1705).

Definition 11 (Phân phối nhị thức)

Biến ngẫu nhiên ξ được gọi là có **phân phối nhị thức** với tham số p, n , với $p \in [0; 1]$ và n là số tự nhiên, nếu ξ nhận các giá trị $0, 1, \dots, n$ và

$$P(\xi = k) = C_n^k p^k q^{n-k}, \quad k = 0, 1, \dots, n,$$

ở đây $q = 1 - p$.

Example 8

Một bài kiểm tra có 100 câu hỏi trắc nghiệm bốn đáp án. Xác suất chọn ngẫu nhiên đúng đáp án của mỗi câu hỏi thì giống nhau và bằng $\frac{1}{4}$.

Ở đây xác suất chọn ngẫu nhiên đúng đáp án của một câu hỏi bất kì là $p = \frac{1}{4}$, và số lượng câu hỏi là $n = 100$.

Gọi ξ là biến ngẫu nhiên số câu hỏi trả lời đúng. Khi đó ξ nhận các giá trị $0, 1, \dots, 100$.

Do đó bài toán này có phân phối nhị thức và

$$P(\xi = k) = C_{100}^k \left(\frac{1}{4}\right)^k \left(\frac{3}{4}\right)^{100-k}.$$

Definition 12 (Phân bố xác suất đều)

Phân bố xác suất P trên không gian xác suất hữu hạn với N phần tử $\Omega = \{A_1, \dots, A_n\}$ được gọi là **phân bố xác suất đều** nếu như

$$P(A_1) = \dots = P(A_n) = 1/N.$$

Khái niệm phân bố đều không mở rộng được lên các không gian xác suất có số phần tử là vô hạn và đếm được vì 1 chia vô cùng bằng 0 mà tổng của chuỗi vô hạn số 0 vẫn bằng 0 chứ không bằng 1.

Remark 3

Phân bố xác suất đều có **tính đối xứng, cân bằng** hay **hoán vị** được của các sự kiện thành phần.

Definition 13 (Phân phối Poisson)

Biến ngẫu nhiên ξ được gọi là có **phân phối Poisson** với tham số λ , nếu ξ nhận các giá trị $0, 1, \dots, n$ và

$$P(\xi = k) = \frac{\lambda^k \cdot e^{-\lambda}}{k!}, \quad k = 0, 1, \dots, n.$$

Tham số λ thể hiện số lần trung bình mà một sự kiện xảy ra trong một khoảng thời gian nhất định. Khi đó, nếu một biến ngẫu nhiên có số lần xuất hiện trung bình của một sự kiện trong thời gian t thì nó có phân phối Poisson với tham số λt , với λ là số lần trung bình trong một đơn vị thời gian.

Biến ngẫu nhiên liên tục**Definition 14 (Biến ngẫu nhiên liên tục)**

Biến ngẫu nhiên ξ được gọi là **liên tục**, nếu nó nhận giá trị tại mọi điểm thuộc một đoạn liên tục nào đó trên trục số, và tồn tại một hàm số không âm $p(x)$ sao cho với mọi đoạn $[a, b]$ (hữu hạn hoặc vô

hạn) ta có

$$P(a \leq \xi \leq b) = \int_a^b p(x) dx$$

Hàm $p(x)$ được gọi là **hàm mật độ** của biến ngẫu nhiên ξ .

Tương tự biến ngẫu nhiên rời rạc, $p(x) \geq 0$ với mọi $x \in \mathbb{R}$ và khi hai cận là vô cực thì biến ngẫu nhiên bao quát toàn bộ không gian mẫu, nghĩa là

$$\int_{-\infty}^{+\infty} p(x) dx = 1.$$

Từ định nghĩa của hàm phân phối $F(x) = P(\xi \leq x)$ ta có hai tính chất của hàm mật độ:

1. $F(x) = \int_{-\infty}^x p(x) dx.$
2. $p(x) = F'(x).$

Tính chất thứ nhất là từ định nghĩa hàm phân phối. Tính chất thứ hai suy ra từ việc cận trên của tích phân là hữu hạn.

Hàm mật độ của biến ngẫu nhiên rời rạc

Biến ngẫu nhiên rời rạc có bảng xác suất sau:

x	x_1	x_2	\dots	x_n	\dots
P	p_1	p_2	\dots	p_n	\dots

Khi đó hàm mật độ của X là:

$$f(x) = \begin{cases} p_i & \text{khi } x = x_i, \\ 0 & \text{khi } x \neq x_i, \text{ với mọi } i. \end{cases}$$

Remark 4

Ta có các lưu ý sau:

- $p_i \geq 0, \sum p_i = 1, i = 1, 2, \dots$
- $P(a < X \leq b) = \sum_{a < x_i \leq b} p_i.$

Hàm mật độ của biến ngẫu nhiên liên tục

➊ Definition 15

Hàm số $f: \mathbb{R} \mapsto \mathbb{R}$ được gọi là **hàm mật độ** của biến ngẫu nhiên liên tục X nếu:

$$P(a \leq X \leq b) = \int_a^b f(x) dx, \forall a, b \in \mathbb{R}.$$

➋ Remark 5

Với mọi $x \in \mathbb{R}$, $f(x) \geq 0$ và $\int_{-\infty}^{+\infty} f(x) dx = 1$.

Ý nghĩa hình học. Xác suất của biến ngẫu nhiên X nhận giá trị trong $[a; b]$ bằng diện tích hình thang cong giới hạn bởi $x = a$, $x = b$, $y = f(x)$ và Ox .

2.5.2 Machine Learning

Mình học ML đa phần từ blog machinelearningcoban.com của anh Vũ Hữu Tiệp [10].

Và mình nhận ra rằng mình không học được ML. :)))

Trong nhiều vấn đề đại số tuyến tính, đặc biệt là phép toán trên ma trận và vector, mình không hiểu công thức nên cách giải quyết của mình luôn là ... viết ra hết. Ở các bài viết trong phần này mình lý giải theo góc nhìn của mình.

Phần này không phải chuyên môn của mình và mình chỉ tự kỷ ở đây :)))

Linear Regression

Giả sử ta có N điểm dữ liệu đầu vào $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ với $\mathbf{x}_i \in \mathbb{R}^d$. Ứng với từng điểm dữ liệu đầu vào \mathbf{x}_i ta có một đầu ra y_i , nghĩa là ta có N cặp dữ liệu (\mathbf{x}_i, y_i) . Khi đó y_i được gọi là **nhãn** (hay **label**) tương ứng với điểm dữ liệu \mathbf{x}_i .

Mục tiêu là xây dựng hàm số $\hat{y} = f(x_1, x_2, \dots, x_d)$ sao cho tổng sai số của y_i và \hat{y}_i là nhỏ nhất, tức là

$$\sum_{i=1}^N \|y_i - \hat{y}_i\|^2 \rightarrow \min.$$

Để hàm số đạt giá trị nhỏ nhất (hoặc lớn nhất) ta tìm cực trị của hàm số và khảo sát. Tuy nhiên không phải hàm số nào cũng đạo hàm được. Một cách tiếp cận đơn giản là sử dụng hàm tuyến tính, dễ xây dựng và luôn khả vi. Ta đặt

$$\hat{y} = f(x_1, x_2, \dots, x_d) = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_d x_d.$$

Lúc này, hàm mất mát có dạng

$$\mathcal{L} = \sum_{i=1}^N \|y_i - (w_0 + w_1 x_{i1} + w_2 x_{i2} + \dots + w_d x_{id})\|^2.$$

Bình phương chuẩn Euclid chính là bình phương của vector. Do đó dưới dấu tổng là các hàm số bình phương. Khi đạo hàm riêng theo w_j ta có

$$\frac{\partial \mathcal{L}}{\partial w_j} = \sum_{i=1}^N 2x_{ij} \cdot [y_i - (w_0 + w_1x_{i1} + w_2x_{i2} + \dots + w_dx_{id})]$$

với $1 \leq j \leq d$.

Với $j = 0$ có chút khác biệt:

$$\frac{\partial \mathcal{L}}{\partial w_0} = \sum_{i=1}^N 2 \cdot [y_i - (w_0 + w_1x_{i1} + \dots + w_dx_{id})].$$

Ta cho các đạo hàm riêng $\frac{\partial \mathcal{L}}{\partial w_j}$ bằng 0 thì được

$$\begin{aligned} \sum_{i=1}^N x_{ij}(w_0 + w_1x_{i1} + w_2x_{i2} + \dots + w_dx_{id}) &= \sum_{i=1}^N x_{ij}y_i \\ \Leftrightarrow w_0 \sum_{i=1}^N x_{ij} + w_1 \sum_{i=1}^N x_{ij}x_{i1} + w_2 \sum_{i=1}^N x_{ij}x_{i2} \\ &\quad + \dots + w_d \sum_{i=1}^N x_{ij}x_{id} = \sum_{i=1}^N x_{ij}y_i. \end{aligned}$$

Bây giờ chúng ta cần biểu diễn các dấu tổng lại thành dạng đại số (ma trận, vector) vì chúng sẽ được sử dụng để nhân với vector $\mathbf{w} = (w_0, w_1, \dots, w_d)$.

Ta có

$$\sum_{i=1}^N x_{ij} = (1 \quad 1 \quad \dots \quad 1) \cdot \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{Nj} \end{pmatrix}.$$

Ta cũng có

$$\sum_{i=1}^N x_{ij}x_{i1} = (x_{11} \quad x_{21} \quad \dots \quad x_{N1}) \cdot \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{Nj} \end{pmatrix}.$$

Cứ tương tự như vậy, ta xếp các dấu sigma thành dạng cột thì tương đương với

$$\begin{pmatrix} * & \sum_{i=1}^N x_{ij} & * \\ * & \sum_{i=1}^N x_{ij}x_{i1} & * \\ \vdots & \vdots & \vdots \\ * & \sum_{i=1}^N x_{ij}x_{id} & * \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_{11} & x_{21} & \dots & x_{N1} \\ \dots & \dots & \ddots & \dots \\ x_{1d} & x_{2d} & \dots & x_{Nd} \end{pmatrix} \cdot \begin{pmatrix} * & x_{1j} & * \\ * & x_{2j} & * \\ \vdots & \vdots & \vdots \\ * & x_{Nj} & * \end{pmatrix}.$$

Ghép các cột theo thứ tự j từ 0 tới d ta có

$$\begin{aligned} & \begin{pmatrix} w_0 & w_1 & \cdots & w_d \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_{11} & x_{21} & \cdots & x_{N1} \\ \cdots & \cdots & \ddots & \cdots \\ x_{1d} & x_{2d} & \cdots & x_{Nd} \end{pmatrix} \\ & \quad \times \begin{pmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \cdots & \cdots & \ddots & \cdots \\ 1 & x_{N1} & \cdots & x_{Nd} \end{pmatrix} \\ & = \begin{pmatrix} y_1 & y_2 & \cdots & y_N \end{pmatrix} \cdot \begin{pmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \cdots & \cdots & \ddots & \cdots \\ 1 & x_{N1} & \cdots & x_{Nd} \end{pmatrix}. \end{aligned}$$

Hay nói cách khác, nếu ta đặt $\mathbf{w} = (w_0, w_1, \dots, w_d)$ là ma trận hàng, \mathbf{X} là ma trận có các hàng là các input, thì phương trình trên được viết lại là $\mathbf{w}\mathbf{X}^T\mathbf{X} = \mathbf{y}\mathbf{X}$.

Nếu đặt $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ và $\mathbf{b} = \mathbf{y}\mathbf{X}$ thì đây là hệ phương trình theo các ẩn w_0, w_1, \dots, w_d . Tuy nhiên không phải lúc nào \mathbf{A} cũng khả nghịch nên chúng ta sẽ sử dụng một khái niệm gọi là **giả nghịch đảo** (hay **pseudo-inverse**) để tìm nghiệm cho hệ phương trình.

Kí hiệu \mathbf{A}^\dagger là giả nghịch đảo của ma trận \mathbf{A} . Khi đó nghiệm của hệ phương trình là $\mathbf{w} = \mathbf{b}\mathbf{A}^\dagger$.

K-Means clustering

Một công việc thường được quan tâm khi xử lý dữ liệu là phân loại một nhóm các đối tượng thành những nhóm nhỏ hơn theo những tiêu chí nhất định.

Tương tự như phần trước, chúng ta có N điểm dữ liệu \mathbf{x}_i thuộc \mathbb{R}^d . Ta muốn phân cụm các vector này vào những cluster (cụm) sao cho chúng gần nhau nhất (về mặt khoảng cách Euclid).

Giả sử ta muốn phân N điểm dữ liệu trên vào $K < N$ cluster. Ta cần tìm các điểm $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_K$ là tâm của các cụm, sao cho tổng khoảng cách từ các điểm \mathbf{x}_i tới tâm cluster mà nó được phân vào là nhỏ nhất. Nghĩa là ứng với center \mathbf{m}_1 ta cần tìm các điểm $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots, \mathbf{x}_{i_t}$ sao cho $\sum_{j=1}^t \|\mathbf{x}_{i_j} - \mathbf{m}_1\|^2$ nhỏ nhất. Tương tự cho các tâm khác.

Nhưng câu chuyện phức tạp ở đây là, tâm nằm ở đâu để có thể bao quát các điểm? Tâm được chọn phải có tính tổng quát, và việc phân các điểm vào cluster tương ứng với tâm thực hiện như thế nào?

Một kỹ thuật thường được sử dụng là **one-hot**. Với mỗi điểm dữ liệu \mathbf{x}_i ta thêm một label $\mathbf{y}_i = (y_{i1}, \dots, y_{iK})$. Điểm \mathbf{x}_i sẽ thuộc cluster j khi $y_{ij} = 1$, không thuộc thì bằng 0. Như vậy chỉ có đúng một phần tử của \mathbf{y}_i bằng 1, còn lại bằng 0, suy ra ràng buộc của $\mathbf{y}_i = (y_{i1}, y_{i2}, \dots, y_{iK})$ là $y_{ij} \in \{0, 1\}$ và $\sum_{j=1}^K y_{ij} = 1$.

Khi đó, ta mong muốn phân các điểm \mathbf{x}_i vào cluster \mathbf{m}_k để khoảng cách tới tâm \mathbf{m}_k là ngắn nhất, hay $\|\mathbf{x}_i - \mathbf{m}_k\|^2 \rightarrow \min$. Thêm nữa, với cách kí hiệu y_{ij} như trên, biểu thức tương đương với

$$\|\mathbf{x}_i - \mathbf{m}_k\|^2 = y_{ik} \|\mathbf{x}_i - \mathbf{m}_k\|^2 = \sum_{j=1}^K y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|^2$$

vì điểm \mathbf{x}_i sẽ thuộc cluster \mathbf{m}_k nào đó với $1 \leq k \leq K$.

Sai số cho toàn bộ dữ liệu lúc này sẽ là

$$\mathcal{L}(\mathbf{Y}, \mathbf{M}) = \sum_{i=1}^N \sum_{j=1}^K y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|^2.$$

Ta cần tối ưu \mathbf{Y} và \mathbf{M} . Việc tối ưu hai ma trận cùng lúc là rất khó thậm chí bất khả thi. Do đó chúng ta có một cách tiếp cận khác là luân phiên cố định một bên và tối ưu bên còn lại. Từ đó công việc được chia làm hai bước.

Bước 1. Cố định \mathbf{M} , tìm \mathbf{Y} .

Giả sử ta đã biết các center $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_K$. Lúc này ta cần phân các điểm \mathbf{x}_i vào cluster gần nó nhất. Để thấy rằng center gần nó nhất sẽ có khoảng cách Euclid ngắn nhất. Do đó ta tìm j sao cho $\|\mathbf{x}_i - \mathbf{m}_j\|^2$ đạt nhỏ nhất. Không cần thiết phải tính căn bậc hai để giảm độ phức tạp.

Bước 2. Cố định \mathbf{Y} , tìm \mathbf{M} .

Khi đã biết \mathbf{Y} tức là ta đã biết điểm nào được phân vào cluster nào. Khi đó ta cần tìm tâm cho từng cluster. Gọi $l(\mathbf{m}_j)$ là hàm tổng bình phương khoảng cách các điểm trong cluster tới tâm \mathbf{m}_j , nghĩa là

$$l(\mathbf{m}_j) = \sum_{i=1}^N y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|^2.$$

Mục tiêu của chúng ta là tối ưu tâm \mathbf{m}_j . Do đó ta đạo hàm theo vector \mathbf{m}_j thu được

$$\frac{\partial l(\mathbf{m}_j)}{\partial \mathbf{m}_j} = \sum_{i=1}^N 2 \cdot y_{ij} (\mathbf{x}_i - \mathbf{m}_j).$$

Cho đạo hàm bằng 0 và biến đổi ta có

$$\begin{aligned} 2 \sum_{i=1}^N y_{ij} (\mathbf{x}_i - \mathbf{m}_j) &= 0 \\ \iff \mathbf{m}_j \sum_{i=1}^N y_{ij} &= \sum_{i=1}^N y_{ij} \mathbf{x}_i \\ \iff \mathbf{m}_j &= \frac{\sum_{i=1}^N y_{ij} \mathbf{x}_i}{\sum_{i=1}^N y_{ij}}. \end{aligned}$$

Để ý rằng, $\sum_{i=1}^N y_{ij}$ là số lượng điểm trong cluster, và $\sum_{i=1}^N y_{ij} \mathbf{x}_i$ là tổng các điểm trong cluster. Như vậy \mathbf{m}_j là trung bình cộng các điểm trong cluster j .

Algorithm 1 (Thuật toán K-Means clustering)

Input: Dữ liệu \mathbf{X} (có N điểm dữ liệu) và số cluster K

Output: Các center \mathbf{M} và label \mathbf{y} cho mỗi điểm dữ liệu

1. Chọn K điểm bất kì làm các cluster ban đầu.
2. Phân mỗi điểm dữ liệu vào cluster gần nó nhất (cố định M , tìm Y).
3. Nếu việc phân dữ liệu vào các cluster ở bước 2 không thay đổi so với trước đó thì dừng thuật toán.
4. Cập nhật center mới cho mỗi cluster bằng cách lấy trung bình cộng các điểm trong cluster (cố định Y , tìm M).
5. Quay lại bước 2.

Gradient Descent

Trong nhiều trường hợp chúng ta thường không thể tìm nghiệm của phương trình đạo hàm để từ đó tìm các cực trị địa phương. Một phương pháp hiệu quả là gradient descent.

Hàm một biến

Giả sử x^* là local extremum (cực trị địa phương) của hàm số $f(x)$. Khi đó chúng ta xây dựng dãy số $\{x_n\}$ hội tụ về x^* . Ý tưởng thực hiện là dựa trên nhận xét, nếu x_n nằm bên phải x^* thì x_{n+1} nằm giữa x^* và x_n . Ta đã biết nếu x^* là một điểm cực trị thì $f'(x) > 0$ với $x > x^*$ (trường hợp cực tiểu), mà x_n đi từ bên phải sang bên trái (ngược chiều Ox nên mang dấu âm). Từ đó chúng ta có công thức chung sau

$$x_{n+1} = x_n - \eta f'(x_n).$$

Trong đó η là một số dương nhỏ, gọi là *learning rate* (tốc độ học).

Ta chọn x_0 là một điểm bất kì. Tuy nhiên việc chọn x_0 cũng có thể ảnh hưởng đến tốc độ hội tụ.

Ví dụ với hàm số $f(x) = x^2 + 5 \sin x$. Ta có đạo hàm là $f'(x) = 2x + 5 \cos x$. Việc giải phương trình đạo hàm bằng 0 là điều không dễ dàng. Do đó gradient descent tỏ ra hiệu quả trong trường hợp này.

Chọn $\eta = 0,1$ và $x_0 = 5$. Sau đó chọn $\eta = 0,1$ và $x_0 = -5$. Ta thấy trường hợp sau tốn ít vòng lặp hơn do $x_0 = -5$ gần điểm cực trị hơn (≈ -1.11).

Hàm nhiều biến

Lúc này đầu vào của hàm số là một vector \mathbf{x} . Đặt $\nabla f(\mathbf{x})$ là đạo hàm của hàm f theo vector \mathbf{x} . Tương tự, ta xây dựng dãy vector $\{\mathbf{x}_n\}$ hội tụ về cực trị \mathbf{x}^* . Công thức lúc này là

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \eta \cdot \nabla f(\mathbf{x}_n).$$

Ta đã biết đạo hàm của hàm số theo vector cũng là vector cùng cõi. Do đó giả sử $f(\mathbf{x}) = f(x_1, x_2, \dots, x_n)$ thì đạo hàm của nó là

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right).$$

Với ví dụ là bài toán Linear Regression, lúc này hàm mất mát là

$$\mathcal{L} = \frac{1}{2N} \sum_{i=1}^N \|y_i - \mathbf{x}_i \mathbf{w}^T\|^2 = \frac{1}{2N} \|\mathbf{y} - \mathbf{X} \mathbf{w}^T\|^2.$$

Đạo hàm của hàm mất mát là

$$\nabla \mathcal{L} = \frac{1}{N} (\mathbf{w} \mathbf{X}^T - \mathbf{y}) \mathbf{X}.$$

Lúc này, với vector khởi đầu \mathbf{w}_0 chúng ta xây dựng dãy $\{\mathbf{w}_n\}$ tới khi nhận được $\mathbf{w}_n/d < \varepsilon$, với d là độ dài vector \mathbf{w} .

Perceptron Learning Algorithm

Một trong những nhiệm vụ quan trọng nhất của ML là phân loại (tiếng Anh - classification).

Perceptron là thuật toán phân loại cho trường hợp đơn giản nhất khi có hai lớp. Nếu ta có các điểm dữ liệu cho trước trong không gian d chiều, ta muốn tìm một siêu phẳng (hình học affine gọi là $(d-1)$ -phẳng) chia các điểm dữ liệu đó thành hai phần. Sau đó khi có một điểm dữ liệu mới ta chỉ cần bỏ nó vào bên này hoặc bên kia của siêu phẳng.

Trong dạng này, mỗi điểm dữ liệu được biểu diễn ở dạng cột của ma trận. Giả sử các điểm dữ liệu là $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$, với $\mathbf{x}_i \in \mathbb{R}^d$, thì ma trận dữ liệu là

$$\mathbf{X} = (\mathbf{x}_1^\top \quad \mathbf{x}_2^\top \quad \cdots \quad \mathbf{x}_N^\top).$$

Ta gọi nhãn tương ứng với N điểm dữ liệu trên là vector $\mathbf{y} = (y_1, y_2, \dots, y_N)$ với $y_i = 1$ nếu \mathbf{x}_i thuộc class xanh, và $y_i = -1$ nếu \mathbf{x}_i thuộc class đỏ.

Một siêu phẳng có phương trình là

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + \dots + w_d x_d = \mathbf{w} \cdot \mathbf{x}^\top.$$

Một điểm thuộc nửa không gian (tạm gọi là *bên này*) đối với siêu phẳng thì $f_{\mathbf{w}}(\mathbf{x}) < 0$, nếu thuộc nửa *kia* thì $f_{\mathbf{w}}(\mathbf{x}) > 0$, nếu nằm trên phẳng thì bằng 0.

Gọi $\text{label}(\mathbf{x})$ là nhãn của điểm \mathbf{x} . Khi đó điểm \mathbf{x} thuộc một trong hai bên của phẳng nên $\text{label}(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x}^\top)$ với sgn là hàm dấu. Ta quy ước $\text{sgn}(0) = 1$.

Khi một điểm bị phân loại sai class thì ta nói điểm đó bị **misclassified**. Ý tưởng của thuật toán là làm giảm thiểu số lượng điểm bị misclassified qua nhiều lần lặp. Đặt

$$J_1(\mathbf{w}) = \sum_{\mathbf{x}_i \in \mathcal{M}} (-y_i \cdot \text{sgn}(\mathbf{w} \cdot \mathbf{w}_i^\top)),$$

trong đó \mathcal{M} là tập các điểm bị misclassified (tập này sẽ thay đổi theo \mathbf{w}).

Nếu \mathbf{x}_i bị misclassified thì y_i và $\text{sgn}(\mathbf{w} \cdot \mathbf{x}_i^\top)$ ngược dấu nhau. Nói cách khác, $-y_i \cdot \text{sgn}(\mathbf{w} \cdot \mathbf{x}_i^\top) = 1$. Từ đó $J_1(\mathbf{w})$ là hàm đếm số lượng điểm bị misclassified. Ta thấy rằng $J_1(\mathbf{w}) \geq 0$ nên ta cần tối ưu để hàm này đạt giá trị nhỏ nhất bằng 0. Khi đó không điểm nào bị misclassified.

Tuy nhiên có một vấn đề. Hàm $J_1(\mathbf{w})$ là hàm rời rạc (hàm sgn) nên rất khó tối ưu vì không thể tính đạo hàm. Do đó chúng ta cần một cách tiếp cận khác, một hàm mượt mà hơn.

Nếu ta bỏ đi hàm sgn thì có hàm

$$J(\mathbf{w}) = \sum_{\mathbf{x}_i \in \mathcal{M}} (-y_i \cdot \mathbf{w} \cdot \mathbf{x}^\top).$$

Nhận xét. Một điểm bị misclassified nằm càng xa biên giới (siêu phẳng) thì giá trị $\mathbf{w} \cdot \mathbf{x}_i^\top$ càng lớn, tức là hàm J đi ra xa so với giá trị nhỏ nhất. Hàm J cũng đạt min ở 0 nên ta cũng có thể dùng hàm này để loại bỏ các điểm bị misclassified.

Lúc này hàm $J(\mathbf{x})$ khả vi nên ta có thể dùng GD hoặc SGD để tìm nghiệm cho bài toán.

Nếu xét tại một điểm thì

$$J(\mathbf{w}, \mathbf{x}_i, y_i) = -y_i \cdot \mathbf{w} \cdot \mathbf{x}_i^\top \Rightarrow \frac{\partial J}{\partial \mathbf{w}} = -y_i \mathbf{x}_i.$$

Khi đó quy tắc để cập nhật là $\mathbf{w} = \mathbf{w} + \eta \cdot y_i \cdot \mathbf{x}_i$ với η là learning rate (thường chọn bằng 1). Nói cách khác ta đang xây dựng dãy $\{\mathbf{w}_n\}$ hội tụ lại nghiệm bài toán với công thức $\mathbf{w}_{i+1} = \mathbf{w}_i + \eta \cdot y_i \cdot \mathbf{x}_i$.

Thuật toán Perceptron Learning Algorithm (PLA) có thể được mô tả như sau:

1. Chọn ngẫu nhiên vector \mathbf{w} với w_i xấp xỉ 0.
2. Duyệt ngẫu nhiên qua các \mathbf{x}_i :
 - nếu \mathbf{x}_i được phân lớp đúng, tức $\text{sgn}(\mathbf{w} \cdot \mathbf{x}_i^\top) = y_i$ thì ta không cần làm gì;
 - nếu \mathbf{x}_i bị misclassified, ta cập nhật \mathbf{w} theo công thức $\mathbf{w} = \mathbf{w} + \eta \cdot y_i \cdot \mathbf{x}_i$.
3. Kiểm tra xem có bao nhiêu điểm bị misclassified. Nếu không còn điểm nào thì ta dừng thuật toán, ngược lại thì quay lại bước 2.

2.6 Đại số tuyến tính

2.6.1 Ma trận

Trong các bài viết của về đại số tuyến tính:

1. Vector sẽ được kí hiệu bởi chữ thường in đậm, ví dụ \mathbf{v}, \mathbf{x} .
2. Ma trận sẽ được kí hiệu bởi chữ hoa in đậm, ví dụ \mathbf{A}, \mathbf{B} .
3. Các đại lượng vô hướng (số) được kí hiệu bởi chữ thường không in đậm, ví dụ x_1, N, t .

Bảng thuật ngữ và kí hiệu

Kí hiệu	Ý nghĩa
\mathbf{A}^\top	Ma trận chuyển vị (transpose) của ma trận \mathbf{A}
\mathbf{A}^{-1}	Ma trận nghịch đảo (inverse) của ma trận \mathbf{A}
rank \mathbf{A}	Hạng (rank) của ma trận \mathbf{A}
det \mathbf{A}	Định thức (determinant) ma trận \mathbf{A}
$\mathbf{x} \cdot \mathbf{y}$	Tích vô hướng hai vector \mathbf{x} và \mathbf{y}
dim \mathcal{V}	Số chiều (dimension) không gian vector \mathcal{V}
$\ \mathbf{x}\ $	Chuẩn Euclid (Euclidean norm) vector \mathbf{x}

Định thức ma trận

➊ Definition 1.55 (Nghịch thế)

Cho tập hợp $A = \{1, 2, \dots, n\}$ và xét hoán vị σ trên A .

Ta gọi hai phần tử i và j tạo thành **nghịch thế** (hay **inversion**) nếu $i < j$ và $\sigma(i) > \sigma(j)$.

Đặt $\sigma = \begin{pmatrix} 1 & 2 & \dots & n \\ k_1 & k_2 & \dots & k_n \end{pmatrix}$ là một hoán vị của A . Ta kí hiệu $P(\sigma)$ là số lượng nghịch thế của σ và đặt $(-1)^{P(\sigma)} = \text{sign } \sigma$.

➋ Example 1.26

Với $n = 4$, $A = \{1, 2, 3, 4\}$.

Xét hoán vị $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix}$.

Ta nhận thấy các cặp nghịch thế $(1, 2), (1, 3), (1, 4), (2, 3)$ gồm bốn cặp nghịch thế. Vậy $P(\sigma) = 4$ và $\text{sign } \sigma = (-1)^4 = 1$.

➌ Definition 1.56 (Định thức)

Khi đó **định thức** của ma trận

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

được định nghĩa là:

$$\det(\mathbf{A}) = \sum_{(i_1, i_2, \dots, i_n)} a_{1,i_1} \cdot a_{2,i_2} \cdot \dots \cdot a_{n,i_n} \cdot \text{sign}\sigma$$

với mọi hoán vị $\sigma = \begin{pmatrix} 1 & 2 & \dots & n \\ i_1 & i_2 & \dots & i_n \end{pmatrix}$ của tập $\{1, 2, \dots, n\}$. Như vậy có $n!$ phần tử cho tổng trên.

❶ Example 1.27

Tính định thức ma trận $\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$.

Xét hoán vị $\sigma_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$. Khi đó $P(\sigma_1) = 0$,

$$a_{11} \cdot a_{22} \cdot a_{33} \cdot (-1)^0 = 1 \cdot 5 \cdot 9 \cdot 1 = 45.$$

Xét hoán vị $\sigma_2 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}$. Khi đó $P(\sigma_2) = 1$,

$$a_{11} \cdot a_{23} \cdot a_{32} \cdot (-1)^1 = 1 \cdot 6 \cdot 8 \cdot (-1) = -48.$$

Xét hoán vị $\sigma_3 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$. Khi đó $P(\sigma_3) = 1$,

$$a_{12} \cdot a_{21} \cdot a_{33} \cdot (-1)^1 = 2 \cdot 4 \cdot 9 \cdot (-1) = -72.$$

Xét hoán vị $\sigma_4 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$. Khi đó $P(\sigma_4) = 2$,

$$a_{12} \cdot a_{23} \cdot a_{31} \cdot (-1)^2 = 2 \cdot 6 \cdot 7 \cdot 1 = 84.$$

Xét hoán vị $\sigma_5 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$. Khi đó $P(\sigma_5) = 2$,

$$a_{13} \cdot a_{21} \cdot a_{32} \cdot (-1)^2 = 3 \cdot 4 \cdot 8 \cdot 1 = 96.$$

Xét hoán vị $\sigma_6 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$. Khi đó $P(\sigma_6) = 3$,

$$a_{13} \cdot a_{22} \cdot a_{31} \cdot (-1)^3 = 3 \cdot 5 \cdot 7 \cdot (-1) = -105.$$

Như vậy

$$\det(A) = 45 - 48 - 72 + 84 + 96 - 105 = 0.$$

Định thức của ma trận còn được định nghĩa theo **đề quy** như sau.

Với ma trận 1×1 là $\mathbf{A} = (a_{11})$ thì $\det(\mathbf{A}) = a_{11}$.

Với ma trận 2×2 là $\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ thì $\det(\mathbf{A}) = a_{11}a_{22} - a_{21}a_{12}$.

Với ma trận $n \times n$, gọi M_{ij} là ma trận có được từ ma trận \mathbf{A} khi bỏ đi hàng i và cột j của ma trận \mathbf{A} và kí hiệu $A_{ij} = (-1)^{i+j} \det(M_{ij})$.

❶ Theorem 1.16 (Định lý Laplace)

Định lý Laplace cho phép ta khai triển định thức của ma trận cấp n thành tổng các ma trận cấp $n-1$.

Khai triển theo cột j :

$$\det(\mathbf{A}) = \sum_{i=1}^n a_{ij} A_{ij} = a_{1j} A_{1j} + a_{2j} A_{2j} + \cdots + a_{nj} A_{nj}, \quad j = \overline{1, n}.$$

Khai triển theo hàng i :

$$\det(\mathbf{A}) = \sum_{j=1}^n a_{ij} A_{ij} = a_{i1} A_{i1} + a_{i2} A_{i2} + \cdots + a_{in} A_{in}, \quad i = \overline{1, n}.$$

Ma trận nghịch đảo

❶ Definition 1.57 (Ma trận nghịch đảo)

Ma trận \mathbf{A}^{-1} được gọi là **ma trận nghịch đảo** của ma trận vuông \mathbf{A} nếu

$$\mathbf{A}^{-1} \cdot \mathbf{A} = \mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{I},$$

trong đó \mathbf{I} là ma trận đơn vị cùng kích thước với \mathbf{A} .

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} [(A_{ij})_n]^\top = \frac{1}{\det(\mathbf{A})} \begin{pmatrix} A_{11} & A_{21} & \cdots & A_{n1} \\ A_{12} & A_{22} & \cdots & A_{n2} \\ \cdots & \cdots & \cdots & \cdots \\ A_{1n} & A_{2n} & \cdots & A_{nn} \end{pmatrix},$$

trong đó, A_{ij} cũng được định nghĩa tương tự như khi tính định thức bằng khai triển theo dòng hoặc cột. Gọi M_{ij} là ma trận có được từ ma trận \mathbf{A} khi bỏ đi hàng i và cột j của ma trận \mathbf{A} và kí hiệu $A_{ij} = (-1)^{i+j} \det(M_{ij})$.

Như vậy, điều kiện cần và đủ để một ma trận có nghịch đảo là định thức khác 0.

Hạng của ma trận

❶ Definition 1.58 (Hạng của ma trận)

Cho ma trận $\mathbf{A}_{m \times n}$. **Hạng** của ma trận là cấp của ma trận con lớn nhất có định thức khác 0.

Nghĩa là, một ma trận vuông mà là ma trận con (lấy một phần của ma trận ban đầu) kích thước $r \times r$ mà có định thức khác 0 và r lớn nhất, thì **hạng** của ma trận đó là r .

1 Remark 1.14

Ma trận con kích thước $r \times r$ là ma trận con của ma trận kích thước $m \times n$ nên $r \leq \min(m, n)$.

1 Example 1.28

Ma trận $A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 1 & 2 & 4 \end{pmatrix}$ có định thức $\det(A) = 0$.

Nhưng ma trận con của A là $B = \begin{pmatrix} 2 & 3 \\ 2 & 4 \end{pmatrix}$ (lấy dòng 1 và 3, lấy cột 2 và 3) có định thức $\det(B) = 2 \neq 0$, do đó

$$r = \text{rank } (A) = 2,$$

với $\text{rank } (A)$ nghĩa là hạng của A .

2.6.2 Toán tử tuyến tính

1 Definition 2.18 (Ánh xạ tuyến tính)

Toán tử tuyến tính (hay **linear operator**, **линейный оператор**) là một ánh xạ

$$A : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

thỏa hai điều kiện:

1. Với mọi $u, v \in \mathbb{R}^n$ thì $A(u) + A(v) = A(u + v)$.
2. Với mọi $\alpha \in \mathbb{R}$ và $u \in \mathbb{R}^n$ thì $A(\alpha u) = \alpha A(u)$.

Nếu A là một ma trận cỡ $m \times n$ thì đây là một ánh xạ tuyến tính được biểu diễn bởi phép nhân ma trận với vector $A \cdot x = y$.

Ở đây $x \in \mathbb{R}^n$ và $y \in \mathbb{R}^m$ là các vector cột.

1 Definition 2.19 (Hạt nhân)

Hạt nhân (hay **kernel**, **ядро**) của ánh xạ tuyến tính A là tập hợp nghiệm của hệ thuần nhất và được ký hiệu là $\ker(A)$. Nói cách khác

$$\ker(A) = \{x \in \mathbb{R}^n : A \cdot x = \mathbf{0}\}.$$

1 Definition 2.20 (Ánh)

Ánh (hay **image**, **образ**) của ánh xạ tuyến tính A là tập hợp tất cả giá trị có thể của phép nhân ma

trận và được kí hiệu là $\text{im}(A)$. Nói cách khác

$$\text{im}(A) = \{A \cdot x : x \in \mathbb{R}^n\}.$$

❶ Property 2.1

Ánh xạ tuyến tính $A: \mathbb{R}^n \rightarrow \mathbb{R}^m$ có tính chất:

1. $\dim(\ker A) + \dim(\text{im } A) = n$.

2.6.3 Trị riêng và vector riêng

Trị riêng và vector riêng

❶ Definition (Trị riêng và vector riêng)

Xét toán tử tuyến tính được biểu diễn bởi ma trận A . Khi đó vector v khác không được gọi là **vector riêng** (hay **eigenvector**) của ma trận nếu tồn tại phần tử λ sao cho

$$Av = \lambda v.$$

Giá trị λ khi đó gọi là **trị riêng** (hay **eigenvalue**) tương ứng với vector riêng v .

Chuyển về đẳng thức trên ta có $(A - \lambda I) \cdot v = 0$. Ở đây I là ma trận cùng cỡ với A và có các phần tử ở hàng i và cột i bằng 1 (ma trận đơn vị).

Như vậy, để phương trình có nghiệm khác không thì ma trận $A - \lambda I$ suy biến, hay $\det(A - \lambda I) = 0$.

Mỗi nghiệm λ của phương trình $\det(A - \lambda I) = 0$ là một trị riêng. Với mỗi trị riêng λ ta tìm được các vector riêng v tương ứng.

❶ Property (Một số tính chất của trị riêng và vector riêng)

Giả sử đối với ma trận A có $n \times n$ thì phương trình đặc trưng có đầy đủ n nghiệm thực, ta có các tính chất sau:

1. $\text{tr } A = \lambda_1 + \lambda_2 + \dots + \lambda_n$.
2. $\det A = \lambda_1 \cdot \lambda_2 \cdots \lambda_n$.

❶ Property (Tính chất liên quan đến rank và trace)

1. $\text{tr}(AB) = \text{tr}(BA)$.
2. $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$.

Bài tập

Bài 1. Cho vector cột $\mathbf{v} \in \mathbb{R}^n$. Đặt $\mathbf{A} = \mathbf{v} \cdot \mathbf{v}^\top$. Tìm $\text{spa}\mathbf{A}$.

Các cột của \mathbf{A} có dạng $\mathbf{v} \cdot \mathbf{v}_1, \mathbf{v} \cdot \mathbf{v}_2, \dots, \mathbf{v} \cdot \mathbf{v}_n$. Như vậy các cột đều tỉ lệ với cột đầu nên $\text{rank}\mathbf{A} = 1$.

Suy ra $\dim \ker \mathbf{A} = n - 1$ và do đó $\lambda = 0$ là nghiệm bậc $n - 1$ trong phương trình đặc trưng.

Như vậy phương trình đặc trưng còn một nghiệm $\lambda \neq 0$.

Do

$$(\mathbf{v} \cdot \mathbf{v}^\top) \mathbf{x} = \lambda \mathbf{x} \Leftrightarrow \mathbf{v}(\mathbf{v}^\top \cdot \mathbf{x}) = \lambda \mathbf{x}.$$

Đặt $\mathbf{v}^\top \cdot \mathbf{x} = \alpha$ thì $\alpha \mathbf{v} = \lambda \mathbf{x}$. Suy ra $\mathbf{x} = \mathbf{v}$ và do đó $\alpha = \lambda = \|\mathbf{v}\|^2$.

Vậy $\text{spa}\mathbf{A} = \{\|\mathbf{v}\|^2, 0, 0, \dots, 0\}$.

Bài 3. Cho ma trận $\mathbf{A}_{3 \times 3}$. Biết rằng $\text{tr}\mathbf{A} = \text{tr}\mathbf{A}^{-1} = 0$ và $\det \mathbf{A} = 1$. Chứng minh rằng $\mathbf{A}^3 = \mathbf{I}$.

Phương trình đặc trưng có dạng $P_3(\lambda) = -\lambda^3 + a_2\lambda^2 + a_1\lambda + a_0$.

Theo tính chất trên thì $a_2 = \sum \lambda = \text{tr}\mathbf{A} = 0$.

Do λ là trị riêng nên $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$. Do \mathbf{A} khả nghịch nên $\frac{1}{\lambda}\mathbf{x} = \mathbf{A}^{-1}\mathbf{x}$.

Nghĩa là $\frac{1}{\lambda}$ là trị riêng của ma trận \mathbf{A}^{-1} . Suy ra $\frac{1}{\lambda_1} + \frac{1}{\lambda_2} + \frac{1}{\lambda_3} = \text{tr}\mathbf{A}^{-1} = 0$.

Từ đó suy ra $\lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1 = 0$.

Cuối cùng $\det \mathbf{A} = \lambda_1 \cdot \lambda_2 \cdot \lambda_3 = 1$.

Vậy phương trình đặc trưng là $P_3(\lambda) = -\lambda^3 + 1$. Theo định lý Cayley-Hamilton thì $P_3(\mathbf{A}) = -\mathbf{A}^3 + \mathbf{I} = \mathbf{0}$, hay $\mathbf{A}^3 = \mathbf{I}$.

Bài 4. Cho ma trận $\mathbf{A}_{n \times n}$, $\mathbf{A}_{ij} \geq 0$. Giả sử ma trận có đủ n trị riêng thực. Chứng minh rằng $\lambda_1 + \lambda_2 + \dots + \lambda_n \geq 0$ với mọi \mathbf{k} in \mathbb{N} .

Ta thấy rằng với $k = 1$ thì $\lambda_1 + \dots + \lambda_n = \text{tr}(\mathbf{A}) \geq 0$.

Vì λ_i là thỏa phương trình $\mathbf{A}\mathbf{x} = \lambda_i\mathbf{x}$ nên nhân hai vế cho \mathbf{A} ta có $\mathbf{A} \cdot \mathbf{A}\mathbf{x} = \mathbf{A} \cdot \lambda_i\mathbf{x}$. Tương đương với $\mathbf{A}^2\mathbf{x} = \lambda_i(\mathbf{A}\mathbf{x}) = \lambda_i^2\mathbf{x}$.

Nói cách khác, λ_i^2 là trị riêng của ma trận \mathbf{A}^2 . Thực hiện tương tự ta có λ_i^k là trị riêng của ma trận \mathbf{A}^k .

Do đó $\lambda_1^k + \dots + \lambda_n^k = \text{tr}(\mathbf{A}^k) \geq 0$.

Bài 5. Cho ma trận \mathbf{A} khả nghịch. \mathbf{X} là ma trận sao cho $\mathbf{AX} + \mathbf{XA} = \mathbf{0}$. Chứng minh rằng $\text{tr} \mathbf{X} = 0$.

Nhân bên trái hai vế cho \mathbf{A}^{-1} ta có $\mathbf{X} + \mathbf{A}^{-1}\mathbf{XA} = \mathbf{0}$. Ta biết rằng $\mathbf{A}^{-1}\mathbf{XA}$ là ma trận tương đương ma trận \mathbf{X} nên $\text{tr}(\mathbf{A}^{-1}\mathbf{XA}) = \text{tr} \mathbf{X}$.

Suy ra $\text{tr}\mathbf{X} + \text{tr}\mathbf{X} = \text{tr} \mathbf{0} = 0$. Từ đây có $\text{tr} \mathbf{X} = 0$.

2.6.4 Tổ hợp tuyến tính

Xét tập hợp các vector $\{v_1, v_2, \dots, v_d\}$ trên \mathbb{R} .

Definition 4.5 (Tổ hợp tuyến tính)

Với vector x bất kì thuộc \mathbb{R} , nếu tồn tại các số thực $\alpha_1, \alpha_2, \dots, \alpha_d$ thuộc \mathbb{R} sao cho

$$x = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_d v_d$$

thì x được gọi là **tổ hợp tuyến tính** (hay **linear combination**) của các vector $v_i, i = 1, 2, \dots, d$.

Ta thấy rằng vector không $\mathbf{0}$ là tổ hợp tuyến tính của mọi tập các vector v_i khi tất cả $\alpha_i = 0$.

Bây giờ ta xét tổ hợp tuyến tính

$$\alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_d v_d = \mathbf{0}.$$

Definition 4.6 (Độc lập tuyến tính)

Tập hợp các vector v_1, v_2, \dots, v_d được gọi là **độc lập tuyến tính** (hay **linear independent**) nếu chỉ có duy nhất trường hợp

$$\alpha_1 = \alpha_2 = \dots = \alpha_d = 0$$

thỏa tổ hợp tuyến tính trên.

Definition 4.7 (Phụ thuộc tuyến tính)

Tập các vector là **phụ thuộc tuyến tính** (hay **linear dependent**) nếu không độc lập tuyến tính. Nói cách khác tồn tại ít nhất một phần tử $\alpha_i \neq 0$.

2.6.5 Không gian vector

Không gian vector

Xét tập hợp các vector \mathcal{V} trên trường \mathbb{F} .

Ta định nghĩa hai phép tính cộng và nhân trên các vector này.

1. Phép cộng là một ánh xạ $\mathcal{V} \times \mathcal{V} \rightarrow \mathcal{V}$ sao cho với mọi $x, y \in \mathcal{V}$ thì $x + y \in \mathcal{V}$.
2. Phép nhân vô hướng là ánh xạ $\mathbb{F} \times \mathcal{V} \rightarrow \mathcal{V}$ sao cho với mọi $\alpha \in \mathbb{F}$ và $x \in \mathcal{V}$ thì $\alpha x \in \mathcal{V}$.

Nói cách khác, phép cộng hai vector và phép nhân vô hướng một số với vector cho kết quả vẫn nằm trong không gian vector đó.

Đồng thời, phép cộng và phép nhân vô hướng phải thỏa mãn các tính chất sau

1. Tính giao hoán với phép cộng: với mọi $x, y \in \mathcal{V}$, $x + y = y + x$.
2. Tính kết hợp với phép cộng: với mọi $x, y, z \in \mathcal{V}$, $x + (y + z) = (x + y) + z$.
3. Phần tử đơn vị của phép cộng: tồn tại vector không $\mathbf{0} \in \mathcal{V}$ sao cho với mọi $x \in \mathcal{V}$, $\mathbf{0} + x = x + \mathbf{0} = x$.

4. Phần tử đối của phép cộng: với mọi $\mathbf{x} \in \mathcal{V}$, tồn tại phần tử $\mathbf{x}' \in \mathcal{V}$ sao cho $\mathbf{x} + \mathbf{x}' = \mathbf{x} + \mathbf{x}' = \mathbf{0}$.
5. Phần tử đơn vị của phép nhân vô hướng: tồn tại phần tử $1_F \in \mathbb{F}$ sao cho với mọi $\mathbf{x} \in \mathcal{V}$ thì $1_F \cdot \mathbf{x} = \mathbf{x}$.
6. Tính kết hợp của phép nhân vô hướng: với mọi $\alpha, \beta \in \mathbb{F}$, với mọi $\mathbf{x} \in \mathcal{V}$ thì $\alpha(\beta\mathbf{x}) = (\alpha\beta)\mathbf{x}$.
7. Tính phân phối giữa phép cộng và nhân: với mọi $\alpha \in \mathbb{F}$, với mọi $\mathbf{x}, \mathbf{y} \in \mathcal{V}$ thì $\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{x} + \alpha\mathbf{y}$.
8. Tính phân phối giữa phép nhân vô hướng: với mọi $\alpha, \beta \in \mathbb{F}$, với mọi $\mathbf{x} \in \mathcal{V}$ thì $(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}$.

Ta thấy rằng không gian vector ở chương trình phổ thông là không gian vector xác định trên trường $\mathbb{F} = \mathbb{R}$. Khi đó $\mathcal{V} = \mathbb{R}^n$. Trong chương này sẽ làm việc với không gian vector thực \mathbb{R} .

Cơ sở và số chiều của không gian vector

Nếu trong không gian vector \mathcal{V} tồn tại các vector độc lập tuyến tính $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d$ mà tất cả các vector trong \mathcal{V} có thể biểu diễn dưới dạng tổ hợp tuyến tính của các vector \mathbf{v}_i trên, thì tập hợp các vector

$$\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d\}$$

được gọi là **cơ sở** (hay **basis**, **базис**) của không gian vector \mathcal{V} .

Khi đó

$$\mathbf{x} = \sum_{i=1}^d \alpha_i \mathbf{v}_i \quad \text{với mọi } \mathbf{x} \in \mathcal{V}.$$

Số lượng phần tử của tập hợp các vector đó (ở đây là d) gọi là **số chiều** (hay **dimension**) của không gian vector \mathcal{V} . Ta kí hiệu $\dim \mathcal{V} = d$.

Ta còn kí hiệu

$$\mathcal{V} = \text{span}\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d\}$$

và nói là không gian vector \mathcal{V} được span (hay được sinh) bởi các vector \mathbf{v}_i .

Ta thấy rằng có thể có nhiều cơ sở cho cùng một không gian vector.

Theorem

Mọi cơ sở của không gian vector \mathcal{V} đều có số phần tử bằng $\dim \mathcal{V}$.

Giả sử ta có $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d$ là một cơ sở của không gian vector \mathbb{R}^n . Khi đó nếu hệ vector $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_d$ cũng là một hệ cơ sở khi và chỉ khi tồn tại ma trận khả nghịch \mathbf{A} sao cho $\mathbf{W} = \mathbf{A} \cdot \mathbf{V}$. Ở đây \mathbf{W} là ma trận với các hàng là các vector \mathbf{w}_i . Tương tự \mathbf{V} là ma trận với các hàng là các vector \mathbf{v}_i .

Chứng minh

Ta viết các vector \mathbf{v}_i dưới dạng \mathbb{R}^n .

$$\begin{aligned} \mathbf{v}_1 &= (v_{11}, v_{12}, \dots, v_{1n}), \\ \mathbf{v}_2 &= (v_{21}, v_{22}, \dots, v_{2n}), \\ \dots &= (\dots, \dots, \dots, \dots), \\ \mathbf{v}_d &= (v_{d1}, v_{d2}, \dots, v_{dn}). \end{aligned}$$

Tương tự là các vector \mathbf{w}_i .

$$\mathbf{w}_1 = (w_{11}, w_{12}, \dots, w_{1n}),$$

$$\mathbf{w}_2 = (w_{21}, w_{22}, \dots, w_{2n}),$$

$$\dots = (\dots, \dots, \dots, \dots),$$

$$\mathbf{w}_d = (w_{d1}, w_{d2}, \dots, w_{dn}).$$

Do \mathbf{v}_i là một cơ sở của \mathbb{R}^n , mọi vector trong \mathbb{R}^n được biểu diễn dưới dạng tóm hợp tuyến tính của các \mathbf{v}_i .

Khi đó ta viết các \mathbf{w}_i dưới dạng tóm hợp tuyến tính của \mathbf{v}_i .

$$\mathbf{w}_1 = \alpha_{11}\mathbf{v}_1 + \alpha_{12}\mathbf{v}_2 + \dots + \alpha_{1d}\mathbf{v}_d$$

$$\mathbf{w}_2 = \alpha_{21}\mathbf{v}_1 + \alpha_{22}\mathbf{v}_2 + \dots + \alpha_{2d}\mathbf{v}_d$$

$$\dots = \dots$$

$$\mathbf{w}_d = \alpha_{d1}\mathbf{v}_1 + \alpha_{d2}\mathbf{v}_2 + \dots + \alpha_{dd}\mathbf{v}_d.$$

Điều này tương đương với

$$\begin{pmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \dots & \dots & \dots & \dots \\ w_{d1} & w_{d2} & \dots & w_{dn} \end{pmatrix} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1d} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2d} \\ \dots & \dots & \dots & \dots \\ \alpha_{d1} & \alpha_{d2} & \dots & \alpha_{dd} \end{pmatrix} \times \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \dots & \dots & \dots & \dots \\ v_{d1} & v_{d2} & \dots & v_{dn} \end{pmatrix}$$

Nếu \mathbf{w}_i cũng là cơ sở của \mathcal{V} , thì các vector \mathbf{v}_i cũng phải biểu diễn được dưới dạng tóm hợp tuyến tính của \mathbf{w}_i .

Nói cách khác, ma trận (α_{ij}) khả nghịch và ta có điều phải chứng minh.

Không gian vector con

Cho không gian vector $\mathcal{V} \subset \mathbb{R}^n$ với phép cộng hai vector và phép nhân vô hướng. Một tập con L của \mathcal{V} được gọi là không gian vector con nếu:

1. Với mọi \mathbf{x}, \mathbf{y} thuộc L , $\mathbf{x} + \mathbf{y} \in L$.
2. Với mọi $\alpha \in \mathbb{R}$, với mọi $\mathbf{x} \in L$, $\alpha\mathbf{x} \in L$.

Nói cách khác, phép cộng và phép nhân vô hướng **đóng** (hay **closure**) trên không gian vector con.

Remark

Trên \mathbb{R}^n , hệ phương trình tuyến tính thuần nhất có thể sinh ra một không gian vector con của \mathbb{R}^n .

Example

Xét hệ phương trình tuyến tính sau:

$$\begin{array}{cccccc} x_1 & + & 3x_2 & + & 5x_3 & + & 7x_4 = 0 \\ 2x_1 & & & + & 4x_3 & + & 2x_4 = 0 \\ 3x_1 & + & 2x_2 & + & 8x_3 & + & 7x_4 = 0 \end{array}$$

Biến đổi ma trận

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 2 & 0 & 4 & 2 \\ 3 & 2 & 8 & 7 \end{pmatrix} \sim \begin{pmatrix} 1 & 3 & 5 & 7 \\ 0 & -6 & -6 & -12 \\ 0 & -7 & -7 & -14 \end{pmatrix} \sim \begin{pmatrix} 1 & 3 & 5 & 7 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Như vậy hệ tương đương với

$$x_1 + 3x_2 + 5x_3 + 7x_4 = 0, \quad x_2 + x_3 + 2x_4 = 0.$$

Ta chọn $x_3, x_4 \in \mathbb{R}$ tự do, khi đó x_1 và x_2 được biểu diễn theo x_3 và x_4

$$x_1 = -2x_3 - x_4, \quad x_2 = -x_3 - 2x_4.$$

Mọi vector trong không gian tuyến tính khi đó có dạng

$$\begin{aligned} (x_1, x_2, x_3, x_4) &= (-2x_3 - x_4, -x_3 - 2x_4, x_3, x_4) \\ &= x_3 \cdot (-2, -1, 1, 0) + x_4 \cdot (-1, -2, 0, 1) \end{aligned}$$

Ở đây ta thấy x_3, x_4 nhận giá trị tùy ý trong \mathbb{R} , và mọi vector trong không gian nghiệm là tổ hợp tuyến tính của hai vector $(-2, -1, 1, 0)$ và $(-1, -2, 0, 1)$. Suy ra hai vector này là cơ sở của không gian nghiệm, và $\dim \mathcal{V} = 2$.

2.6.6 Không gian Euclide

Trên không gian vector \mathcal{V} chúng ta bổ sung thêm một phép toán là tích vô hướng (dot product, tích chấm) của hai vector.

Giả sử với hai vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$ và $\mathbf{y} = (y_1, y_2, \dots, y_n)$. Khi đó tích vô hướng của \mathbf{x} và \mathbf{y} là

$$\mathbf{x} \cdot \mathbf{y} = x_1y_1 + x_2y_2 + \dots + x_ny_n.$$

Một số sách kí hiệu tích vô hướng của hai vector \mathbf{x} và \mathbf{y} là $\langle \mathbf{x}, \mathbf{y} \rangle$. Trong phần này mình sẽ dùng kí hiệu $\mathbf{x} \cdot \mathbf{y}$ như trên.

Không gian vector có phép toán tích vô hướng được gọi là không gian Euclide. Khi $\mathbf{x} = \mathbf{y}$ thì căn bậc hai của kết quả tích vô hướng được gọi là **chuẩn Euclide** (hay **Euclidean norm**) và được kí hiệu

$$\|\mathbf{x}\| = \sqrt{\mathbf{x} \cdot \mathbf{x}} = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}.$$

Như vậy ta có thể viết $\|\mathbf{x}\|^2 = \mathbf{x} \cdot \mathbf{x}$.

❶ Theorem 6.2 (Bất đẳng thức Cauchy-Schwarz)

Với hai vector \mathbf{x} và \mathbf{y} bất kì ta luôn có

$$\|\mathbf{x}\| \cdot \|\mathbf{y}\| \geq |\mathbf{x} \cdot \mathbf{y}|,$$

nghĩa là tích độ dài của hai vector bất kì trong cùng không gian Euclide lớn hơn hoặc bằng tích vô hướng giữa chúng. Dấu bằng xảy ra khi và chỉ khi $\frac{x_1}{y_1} = \frac{x_2}{y_2} = \dots = \frac{x_n}{y_n}$. Nói cách khác là hai vector cùng phương.

❶ Chứng minh

Với mọi số thực t , ta luôn có

$$0 \leq \|x - ty\|^2 = x^2 - 2tx \cdot y + t^2y^2 = \|x\|^2 - 2tx \cdot y + t^2\|y\|^2.$$

Nếu xem biểu thức trên là đa thức bậc hai theo t , để đa thức lớn hơn hoặc bằng 0 với mọi $t \in \mathbb{R}$ thì ta phải có $\Delta' \leq 0$ và $\|y\|^2 > 0$ (luôn đúng). Ta có

$$\Delta' = (x \cdot y)^2 - \|x\|^2 \cdot \|y\|^2 \leq 0,$$

tương đương với $|x \cdot y| \leq \|x\| \cdot \|y\|$ (điều phải chứng minh).

Hệ cơ sở trực giao

Cho không gian Euclide \mathcal{V} và một cơ sở của nó là v_1, v_2, \dots, v_d . Thuật toán trực giao Gram-Schmidt là thuật toán biến đổi cơ sở trên thành một cơ sở mới, trong đó các vector đều trực giao nhau.

❶ Algorithm 6.1 (Thuật toán trực giao Gram-Schmidt)

Input: v_1, \dots, v_d trong \mathbb{R}^n .

Output: u_1, \dots, u_d trong \mathbb{R}^n mà $u_i \cdot u_j = 0$ với mọi $i \neq j$.

1. $u_1 \leftarrow v_1$
2. **for** $i = 2$ **to** d
 1. $w = v_i$
 2. **for** $j = i - 1$ **to** 1
 1. $\mu_{i,j} = (v_i \cdot u_j) / (u_i \cdot u_j)$
 2. $w \leftarrow w - \mu_{i,j} u_j$
 3. $u_i \leftarrow w$
3. Trả về cơ sở trực giao u_1, \dots, u_d

Nói cách khác, với $u_1 = v_1$, với mỗi $i = 2, 3, \dots, d$ ta tính vector u_i với công thức

$$u_i = v_i - \sum_{j=1}^{i-1} \mu_{i,j} u_j.$$

Ở đây $\mu_{i,j} = \frac{v_i \cdot u_j}{u_i \cdot u_j}$ là hệ số trước u_j .

❶ Example 6.8

Xét cơ sở $v_1 = (2, -2, 4)$, $v_2 = (1, -1, 0)$ và $v_3 = (5, -3, 3)$ của \mathbb{R}^3 .

Đặt $u_1 = v_1 = (2, -2, 4)$.

Ta có

$$\mu_{2,1} = \frac{v_2 \cdot u_1}{u_1 \cdot u_1} = \frac{1 \cdot 2 + (-1) \cdot (-2) + 0 \cdot 4}{2^2 + (-2)^2 + 4^2} = \frac{4}{24} = \frac{1}{6}.$$

Suy ra

$$\mathbf{u}_2 = \mathbf{v}_2 - \mu_{2,1} \mathbf{u}_1 = (1, -1, 0) - \frac{1}{6} \cdot (2, -2, 4) = \left(\frac{2}{3}, \frac{-2}{3}, \frac{-2}{3} \right).$$

Tương tự

$$\mu_{3,1} = \frac{\mathbf{v}_3 \cdot \mathbf{u}_1}{\mathbf{u}_1 \cdot \mathbf{u}_1} = \frac{5 \cdot 2 + (-3) \cdot (-2) + 3 \cdot 4}{2^2 + (-2)^2 + 4^2} = \frac{28}{24} = \frac{7}{6}.$$

Tiếp theo

$$\mu_{3,2} = \frac{\mathbf{v}_3 \cdot \mathbf{u}_2}{\mathbf{u}_2 \cdot \mathbf{u}_2} = \frac{5 \cdot \frac{2}{3} + (-3) \cdot \frac{-2}{3} + 3 \cdot \frac{-2}{3}}{\left(\frac{2}{3}\right)^2 + \left(\frac{-2}{3}\right)^2 + \left(\frac{-2}{3}\right)^2} = \frac{5}{2}.$$

$$\begin{aligned} \Rightarrow \quad \mathbf{u}_3 &= \mathbf{v}_3 - \mu_{3,1} \mathbf{u}_1 - \mu_{3,2} \mathbf{u}_2 \\ &= (5, -3, 3) - \frac{7}{6} \cdot (2, -2, 4) - \frac{5}{2} \cdot \left(\frac{2}{3}, \frac{-2}{3}, \frac{-2}{3} \right) \\ &= (1, 1, 0). \end{aligned}$$

Ta có thể kiểm chứng rằng

$$\mathbf{u}_1 \cdot \mathbf{u}_2 = 2 \cdot \frac{2}{3} + (-2) \cdot \frac{-2}{3} + 4 \cdot \frac{-2}{3} = 0.$$

Tương tự với $\mathbf{u}_1 \cdot \mathbf{u}_3 = 0$ và $\mathbf{u}_2 \cdot \mathbf{u}_3 = 0$. Thêm nữa các vector này cũng độc lập tuyến tính nên cũng là một hệ cơ sở của \mathbb{R}^3 .

Như vậy các vector $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ là một cơ sở trực giao của \mathbb{R}^3 .

❶ Remark 6.7

Cơ sở trực giao cho phép ta tính độ dài của tất cả các vector khác trong không gian vector dễ dàng hơn.

Thật vậy, giả sử $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_d$ là các vector trong cơ sở trực giao. Mọi vector \mathbf{x} trong không gian vector đều có dạng

$$\mathbf{x} = \alpha_1 \mathbf{u}_1 + \alpha_2 \mathbf{u}_2 + \dots + \alpha_d \mathbf{u}_d.$$

Khi đó

$$\|\mathbf{x}\|^2 = \mathbf{x}^2 = (\alpha_1 \mathbf{u}_1 + \alpha_2 \mathbf{u}_2 + \dots + \alpha_d \mathbf{u}_d)^2 = \sum_{i=1}^d \alpha_i^2 \mathbf{u}_i^2 + 2 \sum_{i \neq j} \mathbf{u}_i \cdot \mathbf{u}_j.$$

Do các vector trong cơ sở đều trực giao với nhau nên $\mathbf{u}_i \cdot \mathbf{u}_j = 0$ với $i \neq j$, $1 \leq i, j \leq d$. Từ đó ta có được

$$\|\mathbf{x}\|^2 = \sum_{i=1}^d \alpha_i^2 \mathbf{u}_i^2 = \sum_{i=1}^d \alpha_i^2 \|\mathbf{u}_i\|^2,$$

hay

$$\|\mathbf{x}\| = \sqrt{\alpha_1^2 \mathbf{u}_1^2 + \dots + \alpha_d^2 \mathbf{u}_d^2}.$$

Kết quả rất đơn giản, độ dài của các vector bất kì là căn bậc hai của tổ hợp độ dài các vector trong cơ sở và hệ số tương ứng.

Chứng minh thuật toán Gram-Schmidt

Cho không gian vector \mathcal{V} với cơ sở là các vector $\mathbf{v}_1, \dots, \mathbf{v}_d$. Thuật toán Gram-Schmidt biến đổi và cho kết quả là cơ sở mới $\mathbf{u}_1, \dots, \mathbf{u}_d$ sao cho các vector trong cơ sở mới này trực giao nhau đôi một.

Đặt $\mathbf{u}_1 = \mathbf{v}_1$.

Bước 1. Ta chứng minh với mọi $k \geq 2$ thì $\mathbf{u}_k \cdot \mathbf{u}_1 = 0$.

Ta có

$$\mathbf{u}_2 = \mathbf{v}_2 - \mu_{2,1}\mathbf{u}_1 = \mathbf{v}_2 - \frac{\mathbf{v}_2 \cdot \mathbf{u}_1}{\mathbf{u}_1 \cdot \mathbf{u}_1} \cdot \mathbf{u}_1.$$

Suy ra

$$\begin{aligned} \mathbf{u}_2 \cdot \mathbf{u}_1 &= \mathbf{v}_2 \cdot \mathbf{u}_1 - \frac{\mathbf{v}_2 \cdot \mathbf{u}_1}{\mathbf{u}_1 \cdot \mathbf{u}_1} \cdot (\mathbf{u}_1 \cdot \mathbf{u}_1) \\ &= \mathbf{v}_2 \cdot \mathbf{u}_1 - \mathbf{v}_2 \cdot \mathbf{u}_1 = 0. \end{aligned}$$

Như vậy với $k = 2$ thì đẳng thức đúng. Giả sử đẳng thức $\mathbf{u}_k \cdot \mathbf{u}_1 = 0$ đúng tới $k \geq 2$. Xét $k + 1$ ta có

$$\mathbf{u}_{k+1} = \mathbf{v}_{k+1} - \sum_{j=1}^k \mu_{k+1,j} \mathbf{u}_j,$$

suy ra

$$\mathbf{u}_{k+1} \cdot \mathbf{u}_1 = \mathbf{v}_{k+1} \cdot \mathbf{u}_1 - \sum_{j=1}^k \frac{\mathbf{v}_{k+1} \cdot \mathbf{u}_j}{\mathbf{u}_j \cdot \mathbf{u}_j} \cdot (\mathbf{u}_j \cdot \mathbf{u}_1).$$

Ta thấy rằng với $j = 2, \dots, k$ thì $\mathbf{u}_j \cdot \mathbf{u}_1 = 0$ theo giả thiết quy nạp. Như vậy chỉ còn lại $j = 1$ và kết quả là

$$\mathbf{u}_{k+1} \cdot \mathbf{u}_1 = \mathbf{v}_{k+1} \cdot \mathbf{u}_1 - \frac{\mathbf{v}_{k+1} \cdot \mathbf{u}_1}{\mathbf{u}_1 \cdot \mathbf{u}_1} \cdot (\mathbf{u}_1 \cdot \mathbf{u}_1) = 0.$$

Bước 2. Ta chứng minh với mọi $k \geq 3$ thì $\mathbf{u}_k \cdot \mathbf{u}_2 = 0$.

Tương tự ta sử dụng quy nạp. Ta có

$$\mathbf{u}_3 = \mathbf{v}_3 - \mu_{3,2}\mathbf{u}_2 - \mu_{3,1}\mathbf{u}_1,$$

suy ra

$$\mathbf{u}_3 \cdot \mathbf{u}_2 = \mathbf{v}_3 \cdot \mathbf{u}_2 - \mu_{3,2}\mathbf{u}_2 \cdot \mathbf{u}_2 - \mu_{3,1}\mathbf{u}_1 \cdot \mathbf{u}_2.$$

Ta đã chứng minh được $\mathbf{u}_2 \cdot \mathbf{u}_1 = 0$. Như vậy kết quả sẽ là

$$\mathbf{u}_3 \cdot \mathbf{u}_2 = \mathbf{v}_3 \cdot \mathbf{u}_2 - \frac{\mathbf{v}_3 \cdot \mathbf{u}_2}{\mathbf{u}_2 \cdot \mathbf{u}_2} \cdot (\mathbf{u}_2 \cdot \mathbf{u}_2) = 0.$$

Với giả thiết quy nạp $\mathbf{u}_k \cdot \mathbf{u}_2 = 0$ đúng tới $k \geq 3$, ta xét $k + 1$. Khi đó

$$\mathbf{u}_{k+1} = \mathbf{v}_{k+1} - \sum_{j=1}^k \mu_{k+1,j} \mathbf{u}_j,$$

suy ra

$$\mathbf{u}_{k+1} \cdot \mathbf{u}_2 = \mathbf{v}_{k+1} \cdot \mathbf{u}_2 - \sum_{j=1}^k \mu_{k+1,j} \mathbf{u}_j \cdot \mathbf{u}_2.$$

Theo giả thiết quy nạp thì mọi $j \geq 3$ đều cho kết quả $\mathbf{u}_j \cdot \mathbf{u}_2 = 0$, thêm nữa là $\mathbf{u}_1 \cdot \mathbf{u}_2 = 0$ do chứng minh trên. Như vậy trong tổng chỉ còn lại $j = 2$ và kết quả sẽ là

$$\mathbf{u}_{k+1} \cdot \mathbf{u}_2 = \mathbf{v}_{k+1} \cdot \mathbf{u}_2 - \frac{\mathbf{v}_{k+1} \cdot \mathbf{u}_2}{\mathbf{u}_2 \cdot \mathbf{u}_2} \cdot (\mathbf{u}_2 \cdot \mathbf{u}_2) = 0.$$

Từ đây có thể thấy, sử dụng phương pháp quy nạp ta có thể chứng minh được rằng với mỗi số $n \geq 2$, thì mọi $k \geq n + 1$ ta đều có $\mathbf{u}_k \cdot \mathbf{u}_n = 0$, hay nói cách khác là khi thuật toán tính \mathbf{u}_k thì nó sẽ trực giao với tất cả $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{k-1}$.

2.7 Số học

Toán học là vua của các môn khoa học, và số học là nữ hoàng.

---Carl Friedrich Gauss



Hình 2.47: Carl Friedrich Gauss (1777-1855)

2.7.1 Phép chia Euclid. Thuật toán Euclid

Phép chia Euclid

Đây là nền tảng, cơ sở của số học. Từ khi biết tới phép chia hai số nguyên, ta có thể tìm **thương** và **số dư**. Nói theo toán học, nếu ta có hai số nguyên dương a và b thì tồn tại cặp số q, r sao cho $a = qb + r$ với $0 \leq r < b$.

Khi đó, a gọi là số bị chia, b gọi là số chia, q là thương (q trong quotient) và r là số dư (r trong remainder). Đặc biệt, sự tồn tại của cặp số q và r là duy nhất. Thật vậy, nếu ta giả sử tồn tại hai cặp số (q_1, r_1) và (q_2, r_2) đều thỏa đẳng thức trên, nghĩa là

$$a = q_1b + r_1, \quad a = q_2b + r_2.$$

Trừ hai đẳng thức về theo a ta có

$$(q_1 - q_2)b + (r_1 - r_2) = 0,$$

tương đương $(r_2 - r_1) = (q_1 - q_2)b$, mà $0 \leq r_1, r_2 < b$ nên $-b < r_2 - r_1 < b$.

Như vậy chỉ có thể xảy ra trường hợp $r_2 - r_1 = 0$ (vì giá trị tuyệt đối của $r_2 - r_1$ là bội của b nên sẽ lớn hơn b , còn vé trái lại có giá trị tuyệt đối nhỏ hơn b) hay $r_2 = r_1$, kéo theo $q_1 = q_2$.

Thuật toán Euclid

Dựa trên phép chia Euclid, ta có một thuật toán hiệu quả để tìm ước chung lớn nhất giữa hai số a và b .

Kí hiệu $\gcd(a, b)$ là ước chung lớn nhất của a và b . Chúng ta thực hiện đệ quy như sau:

$$\gcd(a, b) = \begin{cases} a, & \text{nếu } b = 0 \\ \gcd(b, a \bmod b), & \text{nếu } b \neq 0. \end{cases}$$

Điểm quan trọng ở thuật toán Euclid là thuật toán chắc chắn sẽ dừng sau một số hữu hạn bước, và kết quả sẽ là ước chung lớn nhất của hai số a và b .

Chứng minh

Đặt $r_0 = a$ và $r_1 = b$. Theo phép chia Euclid tồn tại các số q_0 và r_2 sao cho $r_0 = r_1q_0 + r_2$ với $0 \leq r_2 < r_1$.

Trong thuật toán Euclid, ở bước thứ i ($i = 1, 2, \dots$) vì đã biết r_i và r_{i+1} nên ta tìm được thương q_i và số dư r_{i+2} trong phép chia r_i cho r_{i+1} .

$$\begin{aligned} r_0 &= r_1q_0 + r_2 \\ r_1 &= r_2q_1 + r_3 \\ r_2 &= r_3q_2 + r_4 \\ &\dots = \dots \\ r_i &= r_{i+1}q_i + r_{i+2} \\ &\dots = \dots \\ r_k &= r_{k+1}q_k + 0 \end{aligned}$$

Ở mỗi bước, r_{i+2} luôn nhỏ hơn r_{i+1} . Do đó cuối cùng sẽ bằng 0, và khi đó ta có ước chung lớn nhất là r_{k+1} như trên.

Example 1.29

Tìm ước chung lớn nhất của 784 và 74.

$$\begin{array}{rcl}
 r_i & = & r_{i+1} \cdot q_i + r_{i+2} \\
 784 & = & 74 \cdot 10 + 44 \\
 74 & = & 44 \cdot 1 + 30 \\
 44 & = & 30 \cdot 1 + 14 \\
 30 & = & 14 \cdot 2 + 2 \\
 14 & = & 2 \cdot 7 + 0
 \end{array}$$

Vậy $\gcd(784, 74) = 2$.

Thuật toán Euclid mở rộng**Definition 1.59 (Phương trình Diophantus)**

Cho trước các số nguyên a, b và c . Phương trình Diophantus là phương trình có dạng

$$ax + by = c$$

với x, y là các số nguyên.

Example 1.30

Giải phương trình $5x + 3y = 1$.

Ta có

$$y = \frac{1 - 5x}{3} = \frac{1 - 2x - 3x}{3} = \frac{1 - 2x}{3} - x.$$

Như vậy nếu $y \in \mathbb{Z}$ thì $\frac{1 - 2x}{3} \in \mathbb{Z}$, nghĩa là $1 - 2x$ chia hết cho 3. Vậy $1 - 2x = 3k$ với $k \in \mathbb{Z}$.

Tiếp tục, $1 - 2x = 3k$, suy ra

$$x = \frac{1 - 3k}{2} = \frac{1 - k - 2k}{2} = \frac{1 - k}{2} - k.$$

Do x nguyên nên tương tự $\frac{1 - k}{2}$ cũng nguyên, hay $1 - k = 2t$, tương đương với $k = 1 - 2t$.

Thay ngược lại ta có

$$x = \frac{1 - 3k}{2} = \frac{1 - 3(1 - 2t)}{2} = -1 + 3t.$$

Tiếp tục thay vào để tìm y thì

$$y = \frac{1 - 5x}{3} = \frac{1 - 5(-1 + 3t)}{3} = 2 - 5t.$$

Như vậy nghiệm của phương trình là tất cả các nghiệm (x, y) mà $x = -1 + 3t$, $y = 2 - 5t$ với $t \in \mathbb{Z}$.

Ở đây chúng ta đã thực hiện phép chia có dư liên tiếp để tìm nghiệm. Nói cách khác ta đã thực hiện thuật toán Euclid ở bên trên để làm giảm độ phức tạp ở mỗi bước giải.

Tổng quát ta có thuật toán Euclid mở rộng để tìm ước chung lớn nhất $\gcd(a, b)$ của hai số a, b , và **một** nghiệm của phương trình $ax + by = \gcd(a, b)$.

Ở ví dụ trên, ta đã tìm được một nghiệm của phương trình $5x + 3y = 1$ là $(-1, 2)$ khi $t = 0$. Khi đó ta có thể suy ra tất cả nghiệm (họ nghiệm) của phương trình có dạng $(-1 + 3t, 2 - 5t)$ với $t \in \mathbb{Z}$.

1 Algorithm 1.1 (Thuật toán Euclid mở rộng)

Input: $a, b \in \mathbb{Z}$

Output: $\gcd(a, b), x, y$

1. $r_0 \leftarrow a, r_1 \leftarrow b, r_2 \leftarrow 0$
2. $x_0 \leftarrow 1, x_1 \leftarrow 0, x_2 \leftarrow 0$
3. $y_0 \leftarrow 0, y_1 \leftarrow 1, y_2 \leftarrow 0$
4. While $r_1 \neq 0$
 1. $q \leftarrow r_0 \text{ div } r_1$
 2. $r_2 \leftarrow r_0 - q * r_1, r_0 \leftarrow r_1, r_1 \leftarrow r_2$
 3. $x_2 \leftarrow x_0 - q * x_1, x_0 \leftarrow x_1, x_1 \leftarrow x_2$
 4. $y_2 \leftarrow y_0 - q * y_1, y_0 \leftarrow y_1, y_1 \leftarrow y_2$
5. EndWhile
6. Return r_0, x_0, y_0

Ở thuật toán trên, r_0, r_1 và r_2 hoạt động như thuật toán Euclid chuẩn.

Ở mỗi bước, q là thương của phép chia r_0 cho r_1 , và ta sử dụng q đó để tính x_0 và y_0 mới. Kết quả cuối cùng (r_0, x_0, y_0) lần lượt là ước chung lớn nhất r_0 , và hai số x_0, y_0 thỏa mãn $ax_0 + by_0 = r_0$.

Tại sao chúng ta lại có $(x_0, x_1) = (1, 0)$ và $(y_0, y_1) = (0, 1)$?Thêm nữa, làm sao biết thuật toán hoạt động đúng?

Mục đích của chúng ta là tìm các số (x, y) sao cho $ax + by = \gcd(a, b)$. Khi đó, dựa trên thuật toán Euclid cơ bản ở trên, ta xây dựng dãy số $\{x_n\}$ và $\{y_n\}$ sao cho ở mọi bước thứ n ta đều có

$$ax_n + by_n = r_n. \quad (2.6)$$

Từ thuật toán Euclid, với r_i và r_{i+1} ở bước thứ i ta thực hiện phép chia Euclid $r_i = r_{i+1}q_i + r_{i+2}$ để tìm q_i và r_{i+2} . Từ q_i ở mỗi bước ta tính

$$x_{i+2} = x_i - x_{i+1}q_i, \quad y_{i+2} = y_i - y_{i+1}q_i.$$

Chuyển về hai phương trình trên ta có

$$x_i = x_{i+1}q_i + x_{i+2}, \quad y_i = y_{i+1}q_i + y_{i+2}.$$

Nếu thay hai phương trình vừa rồi vào (2.6) ta được

$$a(x_{i+1}q_i + x_{i+2}) + b(y_{i+1}q_i + y_{i+2}) = r_i,$$

tương đương với

$$(ax_{i+1} + by_{i+1}) \cdot q_i + (ax_{i+2} + bx_{i+2}) = r_i.$$

Do

$$ax_{i+1} + by_{i+1} = r_{i+1}, \quad ax_{i+2} + by_{i+2} = r_{i+2},$$

nên $r_{i+1}q_i + r_{i+2} = r_i$, đúng với thuật toán Euclid chuẩn ban đầu. Như vậy thuật toán mở rộng hoạt động đúng.

Bây giờ ta cần chọn (x_0, x_1) và (y_0, y_1) vì chúng ta đã đặt $r_0 = a$ và $r_1 = b$.

Ở bước thứ 0, vì

$$r_0 = a = ax_0 + by_0,$$

và ở bước thứ 1,

$$r_1 = b = ax_1 + by_1.$$

Để thấy ở bước 0 ta chọn $x_0 = 1$ và $x_1 = 0$, còn ở bước 1 ta chọn $y_0 = 0$ và $y_1 = 1$ là được.

Example 1.31

Tìm một nghiệm nguyên của phương trình $784x + 74y = 2$.

$$\begin{array}{rcl|l} r_i & = & r_{i+1} \cdot q_i & + & r_{i+2} \\ \hline 784 & = & 74 \cdot 10 & + & 44 \\ 74 & = & 44 \cdot 1 & + & 30 \\ 44 & = & 30 \cdot 1 & + & 14 \\ 30 & = & 14 \cdot 2 & + & 2 \\ 14 & = & 2 \cdot 7 & + & 0 \end{array} \left| \begin{array}{rcl|l} x_{i+2} & = & x_i & - & x_{i+1} \cdot q_i \\ 1 & = & 1 & - & 0 \cdot 10 \\ -1 & = & 0 & - & 1 \cdot 1 \\ 2 & = & 1 & - & (-1) \cdot 1 \\ -5 & = & (-1) & - & 2 \cdot 2 \end{array} \right| \left| \begin{array}{rcl|l} y_{i+2} & = & y_i & - & y_{i+1} \cdot q_i \\ -10 & = & 0 & - & 1 \cdot 10 \\ 11 & = & 1 & - & (-10) \cdot 1 \\ -21 & = & -10 & - & 11 \cdot 1 \\ 53 & = & 11 & - & (-21) \cdot 2 \end{array} \right|$$

Các bạn có thể thấy ước chung lớn nhất là số màu cam. Do đó các số x_{i+2} và y_{i+2} cũng chính là điểm dừng và mình không cần tính toán thêm.

Như vậy một nghiệm của phương trình $784x + 74y = 2$ là $(-5, 53)$.

Chúng ta cũng có một cách trình bày khác để giải phương trình nghiệm nguyên trên là sử dụng biến đổi tương đương của ma trận.

Ví dụ, để tìm một nghiệm nguyên (x, y) của phương trình $ax + by = c$ với a, b, c là các số cho trước, chúng ta viết ma trận

$$\left(\begin{array}{cc|c} 1 & 0 & a \\ 0 & 1 & b \end{array} \right)$$

và biến đổi tương đương về dạng

$$\left(\begin{array}{cc|c} * & * & c \\ * & * & 0 \end{array} \right)$$

Khi đó hai số ở hàng trên sẽ là nghiệm cần tìm.

Example 1.32

Sử dụng bài toán ở trên làm ví dụ: tìm một nghiệm nguyên của phương trình $784x + 74y = 2$.

$$\begin{array}{c} \left(\begin{array}{cc|c} 1 & 0 & 784 \\ 0 & 1 & 74 \end{array} \right) \sim \left(\begin{array}{cc|c} 1 & -10 & 44 \\ 0 & 1 & 74 \end{array} \right) \sim \left(\begin{array}{cc|c} 1 & -10 & 44 \\ -1 & 11 & 30 \end{array} \right) \\ \sim \left(\begin{array}{cc|c} 2 & -21 & 14 \\ -1 & 11 & 30 \end{array} \right) \sim \left(\begin{array}{cc|c} 2 & -21 & 14 \\ -5 & 53 & 2 \end{array} \right) \sim \left(\begin{array}{cc|c} 37 & -392 & 0 \\ -5 & 53 & 2 \end{array} \right) \end{array}$$

Về bản chất thì hai cách trình bày là giống nhau.

Bài tập sưu tầm

Câu 1 (đề kiểm tra, ITMO). Tính

$$\gcd(61^{610} + 1, 61^{671} - 1).$$

Mình thay 61 bởi biến x và thực hiện phép chia đa thức theo thuật toán Euclid.

Đầu tiên, xét phép chia $x^{671} - 1$ cho $x^{610} + 1$. Kết quả phép chia là

$$x^{671} - 1 = (x^{610} + 1) \cdot x^{61} - x^{61} - 1.$$

Tiếp theo, xét phép chia $x^{610} + 1$ cho $-x^{61} - 1$. Kết quả là

$$x^{610} + 1 = (-x^{61} - 1) \cdot (-x^{549} + x^{488} - \dots + 1) + 2.$$

Như vậy, ước chung lớn nhất của hai đa thức là 2.

Câu 2 (đề kiểm tra, ITMO). Chứng minh rằng với mọi $a, b, c \in \mathbb{N}$ thì

$$[a, b, c] = \frac{a \cdot b \cdot c \cdot (a, b, c)}{(a, b) \cdot (a, c) \cdot (b, c)},$$

trong đó

- $[a, b, c]$ là bội chung nhỏ nhất của ba số a, b, c
- (a, b, c) là ước chung lớn nhất của ba số a, b, c
- (a, b) là ước chung lớn nhất của hai số a, b .

Chưa làm ra.

Câu 3 (đề kiểm tra, ITMO). Tìm ít nhất một nghiệm nguyên của phương trình

$$311x - 28y = 2.$$

Sử dụng thuật toán Euclid:

$$\begin{array}{c} \left(\begin{array}{cc|c} 1 & 0 & 311 \\ 0 & 1 & -28 \end{array} \right) \xrightarrow{(1) \leftrightarrow (1)} \left(\begin{array}{cc|c} 1 & 11 & 3 \\ 0 & 1 & -28 \end{array} \right) \xrightarrow{(2) \leftrightarrow (2) + 10 \cdot (1)} \left(\begin{array}{cc|c} 1 & 11 & 3 \\ 10 & 111 & 2 \end{array} \right) \\ \xrightarrow{(1) \leftrightarrow (1) - (2)} \left(\begin{array}{cc|c} -9 & -100 & 1 \\ 10 & 111 & 2 \end{array} \right) \xrightarrow{(2) \leftrightarrow (2) - 2 \cdot (1)} \left(\begin{array}{cc|c} -9 & -100 & 1 \\ 28 & 311 & 0 \end{array} \right) \end{array}$$

Như vậy mình có $(-9, -100)$ là một nghiệm của phương trình

$$311x - 28y = 1.$$

Từ đó suy ra một nghiệm của phương trình

$$311x - 28y = 2$$

là $(2 \cdot (-9), 2 \cdot (-100)) = (-18, -200)$.

2.7.2 Hàm Euler

Đầu tiên chúng ta xem xét hệ thăng dư đầy đủ và hệ thăng dư thu gọn.

❶ Definition 2.21 (Hệ thăng dư đầy đủ)

Hệ thăng dư đầy đủ của số nguyên dương n là tập $\mathbb{Z}_n = \{0, 1, \dots, n - 1\}$.

Nói cách khác, hệ thăng dư đầy đủ của n là các số dư có thể có khi chia một số bất kì cho n .

❶ Definition 2.22 (Hệ thăng dư thu gọn)

Hệ thăng dư thu gọn của số nguyên dương n là tập các số a mà $1 \leq a < n$ và $(a, n) = 1$.

$$\mathbb{Z}_n^\times = \{a : 1 \leq a < n \text{ và } (a, n) = 1\}.$$

❶ Definition 2.23 (Phi hàm Euler)

Cho số nguyên dương n . Số lượng các số dương nhỏ hơn n và nguyên tố cùng nhau với n được kí hiệu bởi $\varphi(n)$ và gọi là φ hàm Euler.

Nói cách khác, φ hàm Euler là số lượng phần tử trong tập \mathbb{Z}_n^\times .

$$\varphi(n) = |\mathbb{Z}_n^\times|.$$

❶ Remark 2.4

Nếu n là số nguyên tố thì $\varphi(n) = n - 1$.

Hàm Euler có ý nghĩa quan trọng trong lý thuyết số, công cụ giúp chúng ta giải các vấn đề về số mũ trong modulo.

Tính chất hàm Euler

❶ Remark 2.5

Với $(m, n) = 1$ thì $\varphi(mn) = \varphi(m)\varphi(n)$.

❶ Chứng minh

Ta viết các số từ 1 tới mn thành bảng như sau

1	$m + 1$...	$(n - 1)m + 1$
2	$m + 2$...	$(n - 1)m + 2$
...
m	$m + m$...	$(n - 1)m + m$

Hàng r gồm các phần tử dạng $rm + k$ với $0 \leq r \leq n - 1$ và $1 \leq k \leq m$. Ta thấy rằng nếu $(rm + k, m) = 1$ thì $(k, m) = 1$.

Do đó trên mỗi hàng có $\varphi(m)$ phần tử nguyên tố cùng nhau với m .

Tiếp theo, trên các hàng vừa tìm được, do $(m, n) = 1$ nên để $(rm + k, n) = 1$ thì $(r, n) = 1$, nghĩa là có $\varphi(n)$ hàng như vậy.

Tổng kết lại, ta có $\varphi(m)\varphi(n)$ phần tử trong bảng nguyên tố cùng nhau với mn . Do đó có điều phải chứng minh.

Do tính chất này nên hàm Euler là hàm nhân tính.

Remark 2.6

Cho số nguyên dương n . Khi đó

$$\sum_{d|n} \varphi(d) = n.$$

❶ Chứng minh

Giả sử phân tích thừa số nguyên tố của n là

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}.$$

Khi đó mỗi ước d của n đều có dạng $p_1^{f_1} p_2^{f_2} \cdots p_k^{f_k}$ với $0 \leq f_i \leq e_i$, $i = 1, 2, \dots, k$.

Như vậy

$$\sum_{d|n} \varphi(d) = \sum_{0 \leq f_i \leq e_i} \varphi(p_1^{f_1} p_2^{f_2} \cdots p_k^{f_k}) = \varphi(p_1^{f_1}) \varphi(p_2^{f_2}) \cdots \varphi(p_k^{f_k})$$

Ta liên hệ một dạng biểu thức đơn giản là

$$(1+x)(1+y) = 1 + x + y + xy,$$

hoặc với 3 biến là

$$(1+x)(1+y)(1+z) = 1 + x + y + z + xy + yz + xz + xyz.$$

Tổng quát cho k biến ở trên thì biểu thức tương đương với

$$\begin{aligned} \sum_{0 \leq f_i \leq e_i} \varphi(p_1^{f_1}) \varphi(p_2^{f_2}) \cdots \varphi(p_k^{f_k}) &= (1 + \varphi(p_1) + \varphi(p_1^2) + \cdots + \varphi(p_1^{e_1})) \\ &\quad \times (1 + \varphi(p_2) + \varphi(p_2^2) + \cdots + \varphi(p_2^{e_2})) \\ &\quad \times \cdots \\ &\quad \times (1 + \varphi(p_k) + \varphi(p_k^2) + \cdots + \varphi(p_k^{e_k})). \end{aligned}$$

Ở đây ta rút gọn dễ dàng với $i = 1, 2, \dots, k$:

$$\begin{aligned} &1 + \varphi(p_i) + \varphi(p_i^2) + \cdots + \varphi(p_i^{e_i}) \\ &= 1 + p_i - 1 + p_i^2 - p_i + \cdots + p_i^{e_i} - p_i^{e_i-1} \\ &= p_i^{e_i}. \end{aligned}$$

Như vậy mỗi tổng $1 + \varphi(p_i) + \cdots$ bằng chính $p_i^{e_i}$. Nhân chúng lại với nhau ta có lại n .

Định lý Euler

❶ Theorem 2.5 (Định lý Euler)

Cho số nguyên dương n . Với mọi số nguyên a mà $(a, n) = 1$ thì

$$a^{\varphi(n)} \equiv 1 \pmod{n}.$$

❶ Chứng minh

Giả sử $S = \{a_1, a_2, \dots, a_{\varphi(n)}\}$ là hệ thặng dư thu gọn của n . Ta sẽ chứng minh rằng nếu a là số sao cho $(a, n) = 1$ thì tập hợp

$$\{aa_1 \pmod{n}, aa_2 \pmod{n}, \dots, aa_{\varphi(n)} \pmod{n}\}$$

là hoán vị của tập S .

Thật vậy, giả sử $aa_i \equiv aa_j \pmod{n}$ với $1 \leq i, j \leq \varphi(n)$ và $i \neq j$.

Do $(a, n) = 1$ nên tồn tại nghịch đảo $a' \pmod{n}$ của a . Nhân a' cho hai vế ta còn $a_i \equiv a_j \pmod{n}$.

Nói cách khác, nếu $a_i \not\equiv a_j \pmod{n}$ thì $aa_i \not\equiv aa_j \pmod{n}$, suy ra tập

$$\{aa_1, aa_2, \dots, aa_{\varphi(n)}\}$$

là hoán vị của S .

Ta nhân tất cả phần tử của S thì sẽ bằng tích phần tử của tập trên

$$aa_1 \cdot aa_2 \cdots aa_{\varphi(n)} \equiv a_1 \cdot a_2 \cdots a_{\varphi(n)} \pmod{n}.$$

Đặt $I = a_1 \cdot a_2 \cdots a_{\varphi(n)}$ thì phương trình trên tương đương với

$$a^{\varphi(n)} \cdot I \equiv I \pmod{n},$$

mà $(I, n) = 1$ do là tích các số nguyên tố cùng nhau với n nên rút gọn hai vế ta được

$$a^{\varphi(n)} \equiv 1 \pmod{n}.$$

Ta có điều phải chứng minh.

Định lý Fermat nhỏ

❶ Theorem 2.6 (Định lý Fermat nhỏ)

Cho số nguyên tố p . Với mọi số nguyên a thì

$$a^p \equiv a \pmod{p}.$$

Khi $(a, p) = 1$ thì

$$a^{p-1} \equiv 1 \pmod{p}.$$

❶ Remark 2.7

Khi $(a, p) = 1$ thì định lý Fermat là hệ quả trực tiếp từ định lý Euler.

2.7.3 Thặng dư chính phương

❶ Definition 3.17 (Số chính phương modulo p)

Xét số nguyên tố lẻ p . Số a được gọi là **số chính phương modulo p** nếu $(a, p) = 1$ và tồn tại số x sao cho $x^2 \equiv a \pmod{p}$.

Nói cách khác phương trình đồng dư $x^2 \equiv a \pmod{p}$ có nghiệm.

Chúng ta sử dụng kí hiệu Legendre (Legendre symbol) để thể hiện một số a có phải là số chính phương modulo nguyên tố p không.

❶ Definition 3.18 (Legendre symbol)

Xét p là số nguyên tố, a là số nguyên không chia hết cho p . Khi đó kí hiệu Legendre được định nghĩa là

$$\left(\frac{a}{p}\right) = \begin{cases} 1, & \text{nếu } a \text{ là số chính phương modulo } p \\ -1, & \text{nếu ngược lại.} \end{cases}$$

Một trường hợp tổng quát hơn của kí hiệu Legendre là kí hiệu Jacobi áp dụng cho số nguyên dương bất kì.

❶ Definition 3.19 (Jacobi symbol)

Xét n là số nguyên dương, a là số nguyên không chia hết cho n . Khi đó kí hiệu Jacobi được định nghĩa là

$$\left(\frac{a}{n}\right) = \begin{cases} 1, & \text{nếu } a \text{ là số chính phương modulo } n \\ -1, & \text{nếu ngược lại.} \end{cases}$$

Bài tập sưu tầm

Câu 1 (đề kiểm tra, ITMO). Số 3 có là số chính phương modulo 323 không?

Vì $323 = 17 \cdot 19$, ta sử dụng tiêu chuẩn Euler cho từng modulo 17 và 19:

$$\begin{aligned} \left(\frac{3}{17}\right) &= 3^{\frac{17-1}{2}} \equiv -1 \pmod{17}, \\ \left(\frac{3}{19}\right) &= 3^{\frac{19-1}{2}} \equiv -1 \pmod{19}. \end{aligned}$$

Như vậy 3 không là số chính phương trong modulo 17 và 19.

Kết luận: 3 không là số chính phương modulo 323.

2.7.4 Hàm Möbius

August Ferdinand Möbius là nhà toán học người Đức, đóng góp nổi tiếng của ông là dải Möbius. Tuy nhiên ở đây chúng ta xem xét một hàm số học mang tên ông.

Hàm Möbius đóng vai trò quan trọng trong việc tính các đại lượng liên quan tới số học.

Hàm Möbius

❶ Definition (Hàm Möbius)

Hàm Möbius của số nguyên dương n được định nghĩa như sau:

$$\mu(n) = \begin{cases} 1, & \text{nếu } n = 1 \\ (-1)^k, & \text{nếu } n = p_1 p_2 \dots p_k \text{ với } p_i \text{ là các số nguyên tố khác nhau} \\ 0, & \text{trong các trường hợp còn lại.} \end{cases}$$

Điều này có nghĩa là, nếu n là tích của các số nguyên tố bậc 1 thì $\mu(n) = (-1)^k$ với k là số lượng số nguyên tố trong tích. Như vậy, nếu tồn tại số nguyên tố p sao cho $p^2 \mid n$ thì $\mu(n) = 0$.

Tính chất hàm Möbius

❶ Property

1. Nếu $(n_1, n_2) = 1$ thì $\mu(n_1 \cdot n_2) = \mu(n_1) \cdot \mu(n_2)$.
2. $\sum_{d|n} \mu(d) = 0$ với $n = p_1 p_2 \cdots p_k$.

❷ Chứng minh

Với tính chất 1, ta dễ thấy rằng do n_1 và n_2 nguyên tố cùng nhau nên trong cách phân tích thừa số nguyên tố của chúng sẽ chứa các số nguyên tố khác nhau.

Khi đó $\mu(n_1)$ và $\mu(n_2)$ không bị phụ thuộc nhau và có thể tách thành phép nhân như trên.

Với tính chất 2, chúng ta lần lượt chọn d là tổ hợp của $0, 1, 2, \dots, k$ số nguyên tố:

- nếu $d = 1$ thì $\mu(d) = 1$;
- nếu $d = p_i$ thì $\mu(d) = (-1)^1 = -1$ với $i = \overline{1, k}$;
- nếu $d = p_i p_j$ với $i \neq j$ thì $\mu(d) = (-1)^2 = 1$;
- tương tự như vậy, nếu d là tích của t số nguyên tố thì $\mu(d) = (-1)^t$.

Ở mỗi trường hợp trên, do d là tổ hợp của t số nguyên tố ($0 \leq t \leq k$) nên số cách chọn số nguyên tố p_i ở mỗi trường hợp là C_k^t . Ta cộng tất cả chúng lại

$$\sum_{d|n} \mu(d) = 1 - C_k^1 + C_k^2 - \dots + (-1)^k C_k^k = 0$$

theo nhị thức Newton. Từ đó ta có điều phải chứng minh.

Công thức nghịch đảo Möbius

Giả sử ta có hai hàm f và g từ \mathbb{N} tới \mathbb{Z} . Khi đó hai cách biểu diễn sau là tương đương.

$$f(n) = \sum_{d|n} g(d) \iff g(n) = \sum_{d|n} f(d) \mu\left(\frac{n}{d}\right).$$

Nghĩa là nếu chúng ta có hai hàm số f và g thỏa phương trình đầu (biểu diễn f theo g) thì chúng ta cũng sẽ tìm được cách biểu diễn g theo f .

❶ Chứng minh

Với $d | n$, đặt $d' = \frac{n}{d}$, suy ra $d = \frac{n}{d'}$.

$$\text{Như vậy } f(d) \cdot \mu\left(\frac{n}{d}\right) = f\left(\frac{n}{d'}\right) \cdot \mu(d').$$

Ta lấy tổng lại thì

$$\sum_{d|n} f(d) \cdot \mu\left(\frac{n}{d}\right) = \sum_{d|n} f\left(\frac{n}{d'}\right) \cdot \mu(d') = \sum_{d|n} f\left(\frac{n}{d}\right) \cdot \mu(d).$$

Ở đây lưu ý rằng nếu d là ước của n thì $d' = \frac{n}{d}$ cũng là ước của n . Do đó ta hoàn toàn có thể thay thế d' bởi d trong tổng trên.

Vì $f(n) = \sum_{d|n} g(d)$ nên

$$\sum_{d|n} f\left(\frac{n}{d}\right) \cdot \mu(d) = \sum_{d|n} \mu(d) \sum_{d'|n} g(d'). \quad (2.7)$$

Dễ thấy rằng do $d | n$ và $d' | \frac{n}{d}$ nên tồn tại k, l sao cho $kd = n$ và $ld' = \frac{n}{d}$. Khi đó $n = ldd'$ và $kd = n$, suy ra $d' | n$ và $d | \frac{n}{d'}$.

Tương tự như trên, ta có thể thay thế d bởi d' và ngược lại, phương trình (2.7) tương đương:

$$\sum_{d'|n} g(d') \sum_{d|\frac{n}{d'}} \mu(d),$$

mà $\sum_{a|p} \mu(a) = 0$ nếu $p \neq 1$ và bằng 1 với $p = 1$ (đã chứng minh ở trên) nên từ đây suy ra

$$\sum_{d'|n} g(d') \sum_{d|\frac{n}{d'}} \mu(d) = \sum_{d'|n} g(d') \cdot 1 \text{ (khi } n = d') = g(n).$$

Tương tự ta cũng có công thức nghịch đảo Möbius đối với phép nhân

$$f(n) = \prod_{d|n} g(d) \iff g(n) = \prod_{d|n} f(d)^{\mu\left(\frac{n}{d}\right)}.$$

Liên hệ với hàm Euler

Nếu ta chọn $f(n) = n$ và $g(n) = \varphi(n)$ thì theo công thức nghịch đảo Möbius ta có

$$\varphi(n) = \sum_{d|n} d \cdot \mu\left(\frac{n}{d}\right)$$

do ta đã biết $\sum_{d|n} \varphi(d) = n$.

2.7.5 Định lý số dư Trung Hoa

Định lý số dư Trung Hoa (Chinese Remainder Theorem - CRT) là một trong những định lý quan trọng nhất của số học nói riêng và toán học nói chung.

Chứng minh định lý số dư Trung Hoa

Giả sử ta có hệ phương trình đồng dư

$$\begin{aligned} x &\equiv x_1 \pmod{m_1} \\ x &\equiv x_2 \pmod{m_2} \\ &\dots \\ x &\equiv x_k \pmod{m_k}. \end{aligned}$$

Trong đó $\gcd(m_i, m_j) = 1$ với mọi $i \neq j$, $1 \leq i, j \leq k$.

Khi đó định lý số dư Trung Hoa phát biểu rằng hệ phương trình đồng như này có nghiệm duy nhất trong modulo $M = m_1 m_2 \dots m_k$.

Chứng minh

Chúng ta cần chứng minh sự tồn tại và tính duy nhất của nghiệm.

Để chứng minh sự tồn tại, ta xây dựng cách tính nghiệm bằng quy nạp.

Bước cơ sở. Với $k = 2$, ta có $x \equiv x_1 \pmod{m_1}$ và $x \equiv x_2 \pmod{m_2}$.

Do $\gcd(m_1, m_2) = 1$ nên tồn tại hai số nguyên n_1, n_2 sao cho $m_1 n_1 + m_2 n_2 = 1$ (bổ đề Bézout).

Quan sát một chút, nếu ta modulo hai về $m_1 n_1 + m_2 n_2 = 1$ cho m_1 thì sẽ có $m_2 n_2 \equiv 1 \pmod{m_1}$. Như vậy

$$x \equiv x_1 \cdot 1 \pmod{m_1} \iff x \equiv x_1 \cdot (m_2 n_2) \pmod{m_1}.$$

Tương tự, $m_1 n_1 \equiv 1 \pmod{m_2}$ nên

$$x \equiv x_2 \cdot 1 \pmod{m_2} \iff x \equiv x_2 \cdot (m_1 n_1) \pmod{m_2}.$$

Từ đó ta có công thức nghiệm là

$$x \equiv x_1 m_2 n_2 + x_2 m_1 n_1 \pmod{m_1 m_2}.$$

Khi modulo cho m_1 và m_2 thì kết quả là hai phương trình đồng dư ban đầu.

Tiếp theo chúng ta sử dụng quy nạp để chứng minh với mọi $k \geq 2$ thì nghiệm của hệ phương trình đồng dư là

$$x \equiv x_1 M_1 N_1 + x_2 M_2 N_2 + \dots + x_k M_k N_k \pmod{M}. \quad (2.8)$$

Trong đó

- $M = m_1 \cdot m_2 \cdots m_k$;
- $M_i = M/m_i$;
- N_i là nghịch đảo của M_i trong modulo m_i .

Giả thiết quy nạp. Giả sử (2.8) đúng với $k \geq 2$. Đặt $M = m_1 \cdots m_k$ và

$$X_k = x_1 M_1 N_1 + \dots + x_k M_k N_k.$$

Với $k+1$ ta có

$$x \equiv X_k \pmod{M}, \quad x \equiv x_{k+1} \pmod{m_{k+1}}.$$

Tương tự với hai modulo ở bước cơ sở, do $\gcd(m_{k+1}, m_i) = 1$ với mọi $1 \leq i \leq k$ nên

$$\gcd(m_{k+1}, m_1 \cdots m_k) = \gcd(m_{k+1}, M) = 1.$$

Khi đó tồn tại các số nguyên α và β sao cho $\alpha M + \beta m_{k+1} = 1$.

Nghiệm của hệ hai phương trình đồng dư khi này là

$$x \equiv X_k \beta m_{k+1} + x_{k+1} \alpha M \pmod{M \cdot m_{k+1}}.$$

Đặt $M' = M \cdot m_{k+1} = m_1 \cdots m_k \cdot m_{k+1}$. Ở đây $M = M'/m_{k+1}$ chính là M'_{k+1} trong cách xây dựng nghiệm ở trên.

Từ đó α chính là N'_{k+1} .

Ta có

$$X_k \beta m_{k+1} = (x_1 M_1 N_1 + \dots + x_k M_k N_k) \beta m_{k+1}.$$

Để ý rằng

$$M_i = M/m_i = M'/(m_i \cdot m_{k+1}),$$

nên suy ra

$$M_i \cdot m_{k+1} = M'/m_i = M'_i.$$

Tiếp theo, do $\beta m_{k+1} \equiv 1 \pmod{M}$ và $M = m_1 \cdots m_k$ nên $\beta m_{k+1} \equiv 1 \pmod{m_i}$ với $1 \leq i \leq k$.

Ở trên ta có $N_i M_i \equiv 1 \pmod{m_i}$ nên suy ra $\beta m_{k+1} \cdot N_i M_i \equiv 1 \pmod{m_i}$, tương đương với $(\beta N_i) \cdot (M_i m_{k+1}) \equiv 1 \pmod{m_i}$.

Ta đã chứng minh ở trước $M_i m_{k+1} = M'/m_i = M'_i$ nên $\beta N_i = N'_i$. Tới đây thì ta đã hoàn thành chứng minh do

$$\begin{aligned} & X_k \cdot \beta \cdot m_{k+1} + x_{k+1} \cdot \alpha \cdot M \\ &= (x_1 M_1 N_1 + \dots + x_k M_k N_k) \cdot \beta \cdot m_{k+1} + x_{k+1} \cdot N'_{k+1} \cdot M'_{k+1} \\ &= x_1 \cdot (\beta N_1) \cdot (M_1 m_{k+1}) + \dots + x_k \cdot (\beta N_k) \cdot (M_k m_{k+1}) \\ &\quad + x_{k+1} \cdot N'_{k+1} \cdot M'_{k+1} \\ &= x_1 N'_1 M'_1 + \dots + x_k N'_k M'_k + x_{k+1} N'_{k+1} M'_{k+1}. \end{aligned}$$

Để chứng minh sự duy nhất của nghiệm, giả sử y là một nghiệm khác x của hệ phương trình đồng dư (modulo M). Khi đó $y \equiv x_i \equiv x \pmod{m_i}$. Như vậy $y = x + t_i m_i$, hay nói cách khác y khác x một bội của m_i . Do đó trong modulo m_i chỉ có thể có trường hợp $y \equiv x$ nên nghiệm tìm được ở modulo tổng là duy nhất.

Example 5.5

Tìm nghiệm của hệ phương trình đồng dư sau

$$\begin{aligned} x &\equiv 1 \pmod{3} \\ x &\equiv 4 \pmod{7}. \end{aligned}$$

Ta có $\gcd(3, 7) = 1$ và $3 \cdot 5 + 7 \cdot (-2) = 1$. Do đó nghiệm của hệ phương trình có dạng

$$x \equiv 1 \cdot 7 \cdot (-2) + 4 \cdot 3 \cdot 5 = 4 \pmod{21}.$$

Ta kiểm tra $4 \equiv 1 \pmod{3}$ và $4 \equiv 4 \pmod{7}$ thỏa mãn hệ phương trình đồng dư.

Bài tập sưu tầm

Câu 1 (đề kiểm tra, ITMO). Giải hệ đồng dư

$$\begin{cases} x \equiv 11 \pmod{56} \\ x \equiv 25 \pmod{77}. \end{cases}$$

Do $\gcd(56, 77) = 7$ nên đầu tiên cần tách mỗi phương trình thành các module nguyên tố cùng nhau.

$$x \equiv 11 \pmod{56} \Rightarrow \begin{cases} x \equiv 11 \equiv 3 \pmod{8} \\ x \equiv 11 \equiv 4 \pmod{7}. \end{cases}$$

$$x \equiv 25 \pmod{77} \Rightarrow \begin{cases} x \equiv 25 \equiv 4 \pmod{7} \\ x \equiv 25 \equiv 3 \pmod{11}. \end{cases}$$

Sử dụng định lý số dư Trung Hoa cho hệ

$$\begin{cases} x \equiv 3 \pmod{8} \\ x \equiv 4 \pmod{7} \\ x \equiv 3 \pmod{11} \end{cases}$$

giải ra nghiệm $x \equiv 179 \pmod{616}$.

Câu 2 (đề kiểm tra, ITMO). Tìm hai chữ số cuối của số

$$318^{17683732657328}.$$

Tìm hai chữ số cuối cũng có nghĩa là tính đồng dư trong modulo 100.

Thay vì tính trong modulo 100, chúng ta tính trong modulo 4 và 25 rồi dùng định lý số dư Trung Hoa để gom nghiệm lại.

Đặt $A = 318^{17683732657328}$.

Vì $318 \equiv 0 \pmod{2}$ nên $318^2 \equiv 0 \pmod{2^2}$. Nói cách khác $A \equiv 0 \pmod{4}$.

Vì $\gcd(318, 25) = 1$ nên $318^{\varphi(25)} \equiv 1 \pmod{25}$. Do $\varphi(25) = 20$ nên suy ra

$$318^{17683732657320} \equiv 1 \pmod{25}.$$

Như vậy chỉ cần tính $318^8 \pmod{25}$ nữa. Vì $318 \equiv 18 \pmod{25}$, ta tính

$$18^2 = 1 \pmod{25} \Rightarrow 18^8 \equiv 1 \pmod{25}.$$

Như vậy $318^8 \equiv 1 \pmod{25}$. Ta có hệ đồng dư

$$\begin{cases} A \equiv 0 \pmod{4} \\ A \equiv 1 \pmod{25} \end{cases}$$

Như vậy $A \equiv 76 \pmod{100}$. Vậy hai chữ số cuối là 76.

Câu 3 (đề kiểm tra, ITMO). Tìm tất cả nghiệm của phương trình

$$x^2 \equiv 1 \pmod{5929}.$$

Vì $5929 = 7^2 \cdot 11^2$ nên ta giải trên hai modulo 7^2 và 11^2 .

Trên modulo 7^2 , vì chỉ có ± 1 thỏa $x^2 \equiv 1 \pmod{7^2}$ nên cũng chỉ có ± 1 thỏa $x^2 \equiv 1 \pmod{7^2}$.

Tương tự, trên modulo 11^2 , vì chỉ có ± 1 thỏa $x^2 \pmod{11}$ nên cũng chỉ có ± 1 thỏa $x^2 \equiv 1 \pmod{11^2}$.

Ta tính $(7^2)^{-1} \equiv 42 \pmod{11^2}$ và $(11^2)^{-1} \equiv 32 \pmod{7^2}$.

Khi đó nghiệm của phương trình

$$x^2 \equiv 1 \pmod{5929}$$

cũng là nghiệm của hệ

$$\begin{cases} x \equiv \pm 1 \pmod{7^2} \\ x \equiv \pm 1 \pmod{11^2}. \end{cases}$$

Như vậy có 4 nghiệm là

$$x \equiv \pm 1 \cdot 7^2 \cdot 42 \pm 1 \cdot 32 \cdot 11^2 \equiv 1, 4115, 1814, 5928 \pmod{5929}.$$

2.7.6 Bài tập số học sâu tâm

Câu 1 (đề kiểm tra, ITMO). Chứng minh (không dùng quy nạp) rằng với mọi $n \in \mathbb{N}$ thì

$$5 \cdot 2^{3n-2} + 3^{3n-1} \vdots 19.$$

Theo định lý Fermat nhỏ thì $2^{18} \equiv 1 \pmod{19}$, hay $(2^3)^6 \equiv 1 \pmod{19}$. Tương tự $3^{18} \equiv 1 \pmod{19}$. Do đó mình sẽ xét các dạng của n là $6k, 6k+1, \dots, 6k+5$.

Trường hợp 1. $n = 6k$. Khi đó

$$\begin{aligned} 2^{3n-2} &= 2^{3 \cdot 6k-2} \equiv 2^{-2} = 5 \pmod{19} \\ 3^{3n-1} &= 3^{3 \cdot 6k-1} \equiv 3^{-1} = 13 \pmod{19} \\ \Rightarrow 5 \cdot 2^{3n-2} + 3^{3n-1} &\equiv 5 \cdot 5 + 13 \equiv 0 \pmod{19}. \end{aligned}$$

Trường hợp 2. $n = 6k+1$. Khi đó

$$\begin{aligned} 2^{3n-2} &= 2^{3 \cdot 6k+1} \equiv 2 \pmod{19} \\ 3^{3n-1} &= 3^{3 \cdot 6k+2} \equiv 3^2 = 9 \pmod{19} \\ \Rightarrow 5 \cdot 2^{3n-2} + 3^{3n-1} &\equiv 5 \cdot 2 + 9 \equiv 0 \pmod{19}. \end{aligned}$$

Trường hợp 3. $n = 6k+2$. Khi đó

$$\begin{aligned} 2^{3n-2} &= 2^{3 \cdot 6k+4} \equiv 2^4 = 16 \pmod{19} \\ 3^{3n-1} &= 3^{3 \cdot 6k+5} \equiv 3^5 = 15 \pmod{19} \\ \Rightarrow 5 \cdot 2^{3n-2} + 3^{3n-1} &\equiv 5 \cdot 16 + 15 \equiv 0 \pmod{19}. \end{aligned}$$

Trường hợp 4. $n = 6k+3$. Khi đó

$$\begin{aligned} 2^{3n-2} &= 2^{3 \cdot 6k+7} \equiv 2^7 = 14 \pmod{19} \\ 3^{3n-1} &= 3^{3 \cdot 6k+8} \equiv 3^8 = 6 \pmod{19} \\ \Rightarrow 5 \cdot 2^{3n-2} + 3^{3n-1} &\equiv 5 \cdot 14 + 6 \equiv 0 \pmod{19}. \end{aligned}$$

Trường hợp 5. $n = 6k+4$. Khi đó

$$\begin{aligned} 2^{3n-2} &= 2^{3 \cdot 6k+10} \equiv 2^{10} = 17 \pmod{19} \\ 3^{3n-1} &= 3^{3 \cdot 6k+11} \equiv 3^{11} = 10 \pmod{19} \\ \Rightarrow 5 \cdot 2^{3n-2} + 3^{3n-1} &\equiv 5 \cdot 17 + 10 \equiv 0 \pmod{19}. \end{aligned}$$

Trường hợp 6. $n = 6k + 5$. Khi đó

$$\begin{aligned} 2^{3n-2} &= 2^{3 \cdot 6k+13} \equiv 2^{13} = 3 \pmod{19} \\ 3^{3n-1} &= 3^{3 \cdot 6k+14} \equiv 3^{14} = 4 \pmod{19} \\ \Rightarrow 5 \cdot 2^{3n-2} + 3^{3n-1} &\equiv 5 \cdot 3 + 4 \equiv 0 \pmod{19}. \end{aligned}$$

Câu 2 (đề kiểm tra, ITMO). Tính

$$1! + 2! + \dots + 2023! + 2024! \pmod{10}.$$

Trong các giai thừa từ $5!$ trở đi luôn có 5 và 2 do đó luôn chia hết cho 10 . Vì vậy chỉ cần tính

$$1! + 2! + 3! + 4! = 1 + 2 + 6 + 24 \equiv 3 \pmod{10}.$$

Câu 3 (đề kiểm tra, ITMO). Chứng minh số $154^{20} + 139^{31}$ là hợp số.

Vì

$$154^{20} \equiv (-1)^{20} \equiv 1 \pmod{5} \text{ và } 139^{31} \equiv (-1)^{31} \equiv -1 \pmod{5}$$

nên số đã cho chia hết cho 5 và do đó là hợp số.

Câu 4 (đề kiểm tra, ITMO). Chứng minh số $31^{51} + 27$ là hợp số.

Đã thấy 31 là số lẻ nên lũy thừa của nó cũng là số lẻ. Tổng hai số lẻ là số chẵn nên số đã cho chia hết cho 2 và do đó là hợp số.

Câu 5 (đề kiểm tra, ITMO). Chứng minh với mọi $n \in \mathbb{N}$ thì $n^8 + 4$ là hợp số.

Ta có

$$n^8 + 4 = n^8 + 4n^4 + 4 - 4n^4 = (n^4 + 2) - (2n^2)^2 = (n^4 + 2 - 2n^2)(n^4 + 2 + 2n^2).$$

Hai biểu thức trong ngoặc luôn lớn hơn 1 nên suy ra $n^8 + 4$ là hợp số.

Câu 6 (đề kiểm tra, ITMO). Tìm tất cả số nguyên tố p sao cho $3p + 20$ và $4p + 1$ là số nguyên tố.

Chưa làm ra.

Câu 7 (đề kiểm tra, ITMO). Tìm tất cả số nguyên tố p sao cho $2p^2 + 5p - 2$ cũng là số nguyên tố.

Chưa làm ra.

Câu 8 (đề kiểm tra, ITMO). Chứng minh rằng không tồn tại đa thức P với hệ số nguyên sao cho $P(40) = 30$ và $P(19) = 24$.

Đặt

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0.$$

Khi đó

$$P(40) = a_n \cdot 40^n + a_{n-1} \cdot 40^{n-1} + \dots + a_1 \cdot 40 + a_0,$$

và

$$P(19) = a_n \cdot 19^n + a_{n-1} \cdot 19^{n-1} + \dots + a_1 \cdot 19 + a_0.$$

Vì $40^k - 19^k = (40 - 19) \cdot (\dots)$ nên $40^k - 19^k$ chia hết cho $40 - 19 = 21$ với mọi k .

Nói cách khác $P(40) - P(19)$ chia hết cho 21 , nhưng $30 - 24 = 6$ không chia hết 21 . Do đó không tồn tại đa thức P thỏa mãn đề bài.

Câu 9 (đề kiểm tra, ITMO). Tìm tập tất cả số $x \in \mathbb{Z}$ sao cho $533x \equiv 1 \pmod{17}$.

Vì $533 \equiv 6 \pmod{17}$ nên ta chỉ cần giải phương trình $6x \equiv 1 \pmod{17}$ là đủ.

Sử dụng thuật toán Euclid mở rộng có thể tính được $6^{-1} = 3 \pmod{17}$, suy ra nghiệm là

$$x \equiv 3 \pmod{17},$$

nghĩa là $x = 3 + 17k$ với $k \in \mathbb{Z}$.

Câu 10 (đề kiểm tra, ITMO). Tìm số dư của 454^{225} khi chia cho 16.

Do 454 chia hết cho 2 nên 454^4 chia hết cho 16 .

Suy ra 454^{225} chia hết cho 16 .

2.8 Toán rời rạc

2.8.1 Quan hệ hai ngôi

Quan hệ hai ngôi

i Definition (Quan hệ hai ngôi)

Xét hai tập hợp A và B . Ta gọi \mathcal{R} là một quan hệ hai ngôi trên A và B nếu $\mathcal{R} \subset A \times B$, trong đó $A \times B$ là tích Descartes của hai tập hợp.

Nếu phần tử $(a, b) \in \mathcal{R}$ với $a \in A$ và $b \in B$ thì ta nói a **có quan hệ với** b và kí hiệu $a\mathcal{R}b$.

Khi $A \equiv B$ thì ta nói \mathcal{R} là quan hệ hai ngôi trên A . Đây cũng là yếu tố quan trọng cho các khái niệm về sau.

i Example

Xét hai tập hợp $A = \{1, 2, 3, 4\}$ và $B = \{a, b, c\}$. Khi đó tích Descartes

$$\begin{aligned} A \times B = & \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c), \\ & (3, a), (3, b), (3, c), (4, a), (4, b), (4, c)\} \end{aligned}$$

Giả sử $\mathcal{R} = \{(1, a), (3, b), (4, c)\}$.

Khi đó 1 quan hệ với a do $(1, a) \in \mathcal{R}$, hay $1Ra$.

Tuy nhiên 1 không có quan hệ với b do $(1, b) \notin \mathcal{R}$.

Sau đây ta định nghĩa các loại quan hệ hai ngôi.

i Definition

Cho R là quan hệ hai ngôi trên tập A . Ta nói:

1. \mathcal{R} **phản xạ** (hay **reflexive**) nếu với mọi $x \in A$ thì $(x, x) \in \mathcal{R}$.
2. \mathcal{R} **đối xứng** (hay **symmetric**) nếu $(x, y) \in \mathcal{R}$ thì $(y, x) \in \mathcal{R}$.

3. **\mathcal{R} phản xứng** (hay **antisymmetric**) nếu $(x, y) \in \mathcal{R}$ thì $(y, x) \notin \mathcal{R}$. Nói cách khác nếu $(x, y) \in \mathcal{R}$ và $(y, x) \in \mathcal{R}$ thì $x = y$.
4. **\mathcal{R} bắc cầu** (hay **transitive**) nếu $(x, y) \in \mathcal{R}$ và $(y, z) \in \mathcal{R}$ thì $(x, z) \in \mathcal{R}$.

Quan hệ tương đương

Quan hệ tương đương giúp ta chia (phân hoạch) một tập hợp rời rạc thành các tập con mà chỉ cần một phần tử đại diện cho tập con đó là đủ để tính toán.

❶ Definition (Quan hệ tương đương)

Cho \mathcal{R} là quan hệ trên tập A . Khi đó \mathcal{R} được gọi là **quan hệ tương đương** (hay **equivalence relation**, **отношение эквивалентности**) nếu \mathcal{R} phản xạ, đối xứng và bắc cầu.

Ta có thể kí hiệu $x\mathcal{R}y$, với \mathcal{R} là quan hệ tương đương, là $x \sim y$ hoặc $x\tilde{\mathcal{R}}y$.

Tiếp theo ta định nghĩa lớp tương đương chứa phần tử x và tập thương.

❶ Definition (Lớp tương đương)

Cho \mathcal{R} là quan hệ tương đương trên tập A . Khi đó với $x \in A$, ta định nghĩa lớp tương đương chứa phần tử x là tập các phần tử của A có quan hệ với x :

$$\bar{x} = \{y \in A, y\mathcal{R}x\}.$$

❶ Definition (Tập thương)

Tập hợp các lớp tương đương như trên tạo thành tập thương.

$$A/\mathcal{R} = \{\bar{x}, x \in A\}.$$

❶ Example

Xét số nguyên dương n . Với số nguyên x và y , ta nói x có quan hệ với y nếu $n \mid (x - y)$, hay $x \equiv y \pmod{n}$. Ta kí hiệu quan hệ này là $n\mathbb{Z}$.

Quan hệ trên là quan hệ tương đương vì

1. $n \mid 0 = x - x$ với mọi $x \in \mathbb{Z}$ nên có tính phản xạ.
2. $n \mid (x - y)$ suy ra $n \mid -(x - y) = y - x$ với mọi $x, y \in \mathbb{Z}$ nên có tính đối xứng.
3. $n \mid (x - y)$ và $n \mid (y - z)$ suy ra $n \mid (x - y + y - z) = (x - z)$ nên có tính bắc cầu.

Từ đó ta có thể phân tách \mathbb{Z} thành các lớp tương đương

$$\begin{aligned}\bar{0} &= \{\dots, -2n, -n, 0, n, 2n, \dots\}, \\ \bar{1} &= \{\dots, -2n+1, -n+1, 1, n+1, 2n+1, \dots\}, \\ &\vdots \\ \bar{n-1} &= \{\dots, -n-1, -1, n-1, 2n-1, 3n-1, \dots\}.\end{aligned}$$

Tập thương của chúng ta là

$$\mathbb{Z}/n\mathbb{Z} = \{\overline{0}, \overline{1}, \dots, \overline{n-1}\}.$$

Quan hệ thứ tự

❶ Definition (Quan hệ thứ tự)

Cho quan hệ \mathcal{R} trên tập A . Ta nói \mathcal{R} là **quan hệ thứ tự** (hay **order relation**, **отношение порядка**) nếu \mathcal{R} phản xạ, phản xứng và bắc cầu.

❶ Definition

Cho tập hợp A và quan hệ \mathcal{R} trên A là quan hệ thứ tự. Nếu $x\mathcal{R}y$ thì ta ký hiệu $x \prec y$. Khi đó (A, \prec) được gọi là **tập có thứ tự** (hay **ordered set**).

Tiếp theo là một số định nghĩa quan trọng về tập hợp có thứ tự.

❶ Definition

Với (A, \prec) và $x, y \in A$.

1. Nếu $x \prec y$, ta nói y là **trội** của x , hay là x **được trội** bởi y .
2. y là **trội trực tiếp** của x nếu không tồn tại z sao cho $x \prec z \prec y$.

❶ Definition

Xét (A, \prec) .

1. x và y thuộc A được gọi là **so sánh được** nếu $x \prec y$ hoặc $y \prec x$.
2. Nếu với mọi $x, y \in A$, x và y so sánh được thì (A, \prec) được gọi là **quan hệ thứ tự toàn phần**. Ngược lại thì gọi là **quan hệ thứ tự bán phần**.

Để biểu diễn sự so sánh trong một tập hợp, ta sử dụng biểu đồ Hasse.

❶ Definition

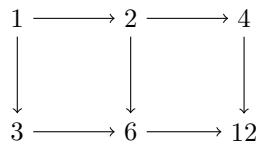
Biểu đồ Hasse của (A, \prec) với A là tập hữu hạn bao gồm

1. Tập điểm - mỗi điểm biểu diễn một phần tử của A .
2. Tập cung - vẽ một cung từ x tới y nếu y là trội trực tiếp của x .

❶ Example

Xét tập $U_{12} = \{1, 2, 3, 4, 6, 12\}$ với quan hệ $x \mathcal{R} y$ được định nghĩa bởi x là ước của y .

Theo đó, biểu đồ Hasse của quan hệ trên là [Hình 2.48](#).



Hình 2.48: Biểu đồ Hasse của U_{12}

❶ Definition

Xét quan hệ thứ tự (A, \prec) .

1. Phần tử $M \in A$ được gọi là
 1. **Tối đại** nếu $M \prec x$ thì $x = M$.
 2. **Cực đại** (hay **lớn nhất**) nếu với mọi $x \in A$ thì $x \prec M$.
2. Phần tử $m \in A$ được gọi là
 1. **Tối thiểu** nếu $x \prec m$ thì $x = m$.
 2. **Cực thiểu** (hay **nhỏ nhất**) nếu với mọi $x \in A$ thì $m \prec x$.

❶ Remark

1. Phần tử cực đại nếu có là duy nhất. Tương tự cho cực thiểu.
2. Nếu n là phần tử tối đại duy nhất thì nó cũng là cực đại. Tương tự cho tối thiểu.

Trong ví dụ U_{12} ở trên thì 1 là tối thiểu và cũng là cực thiểu, và 12 là tối đại và cũng là cực đại.

2.8.2 Phân hoạch

Số Stirling

Số Stirling loại 1

Xét tập các hoán vị trên tập hữu hạn n phần tử.

Số hoán vị chứa k chu trình độc lập là số Stirling loại 1, kí hiệu là $\mathcal{S}_n^{(k)}$.

Các bước khởi đầu là

- $\mathcal{S}_0^{(0)} = 1$ (có đúng một cách chia tập 0 phần tử vào 0 chu trình);
- $\mathcal{S}_n^{(0)} = \mathcal{S}_0^{(n)} = 0$ (không có cách nào chia tập n phần tử vào 0 chu trình và chia tập 0 phần tử vào n chu trình).

Công thức đệ quy của số Stirling là:

$$\mathcal{S}_{n+1}^{(k)} = n \cdot \mathcal{S}_n^{(k)} + \mathcal{S}_n^{(k-1)}.$$

❶ Chứng minh

Giả sử tập có $n + 1$ phần tử là $A = \{1, 2, \dots, n, n + 1\}$.

Khi đó để phân hoạch $n + 1$ phần tử này vào k chu trình độc lập thì có hai trường hợp:

- Đầu tiên xếp các phần tử $1, 2, \dots, n$ vào k chu trình độc lập thì có $S_n^{(k)}$ cách chọn. Tiếp theo ta cho phần tử $n + 1$ vào một vị trí bất kì trong k chu trình trên thì có n cách chọn. Vậy trường hợp này là $n \cdot S_n^{(k)}$.
- Ta phân n phần tử $1, 2, \dots, n$ vào $k - 1$ chu trình độc lập và phần tử $n + 1$ sẽ vào chu trình thứ k . Trường hợp này có $S_n^{(k-1)} \cdot 1$ cách chọn.

Như vậy tổng số cách phân $n + 1$ phần tử vào k chu trình độc lập là hợp của hai trường hợp trên. Ta có điều phải chứng minh.

❶ Example 12

Có bao nhiêu hoán vị của S_5 chứa đúng ba chu trình độc lập?

Ta có các trường hợp sau:

- Hoán vị có dạng $(1)(2)(3, 4, 5)$. Ta chọn hai phần tử làm hai chu trình độc lập, có $C_5^2 = 10$ cách chọn. Tiếp theo chọn ba phần tử cho chu trình cuối, có $2!$ cách chọn. Vậy trường hợp này có $10 \cdot 2 = 20$ cách chọn.
- Hoán vị có dạng $(1)(2, 3)(4, 5)$. Ta chọn một phần tử làm chu trình độc lập có $C_5^1 = 5$ cách chọn. Tiếp theo chọn hai phần tử cho chu trình tiếp theo trong 4 phần tử còn lại, có $C_4^2 = 6$ cách chọn. Hai phần tử còn lại là $C_2^2 = 1$ cách chọn. Lưu ý là hai chu trình độc lập có thể đổi chỗ cho nhau nên cần chia thêm $2!$ nữa, ví dụ $(1, 2)(3, 4)$ hoàn toàn giống với $(3, 4)(1, 2)$. Số cách chọn cho trường hợp này là $5 \cdot 6 / 2 = 15$ cách chọn.

Như vậy số hoán vị của S_5 chứa đúng ba chu trình độc lập là $20 + 15 = 35$.

Ta so sánh kết quả với số Stirling. Ta cần tìm $S_5^{(3)}$.

Ta có:

$$\begin{aligned} S_5^{(3)} &= 4 \cdot S_4^{(3)} + S_4^{(2)} \\ &= 4 \cdot (3 \cdot S_3^{(3)} + S_3^{(2)}) + 3 \cdot S_3^{(2)} + S_3^{(1)} \\ &= 4 \cdot (3 \cdot 1 + 2 \cdot S_2^{(2)} + S_2^{(1)}) + 3 \cdot (2 \cdot S_2^{(2)} + S_2^{(1)}) + 2 \cdot S_2^{(1)} + S_2^{(0)} \\ &= 4 \cdot (3 + 2 \cdot 1 + 1 \cdot S_1^{(1)} + S_1^{(0)}) + 3 \cdot (2 \cdot 1 + S_1^{(1)} + S_1^{(0)}) + S_1^{(1)} + S_1^{(0)} + 0 \\ &= 4 \cdot (3 + 2 + 1) + 3 \cdot (2 + 1 + 0) + 2 \cdot 1 + 0 + 0 = 35. \end{aligned}$$

Kết quả số Stirling khớp với việc giải bằng tổ hợp và không đòi hỏi các suy luận phức tạp.

❶ Remark 7

Số Stirling loại 1 cho phép tính số lượng phân hoạch theo đệ quy thay vì giải các bài toán tổ hợp phức tạp. Điều này hiệu quả khi các số n và k trở nên lớn.

❶ Property 1 (Tính chất của số Stirling loại 1)

$$\sum_{k=1}^n S_n^{(k)} = n!.$$

Công thức này chỉ số hoán vị có thể có của một tập n phần tử.

Số Stirling loại 2

Số Stirling loại 2 thể hiện số cách phân bổ n phần tử vào k tập hợp rời nhau, kí hiệu là $s(n, k)$.

Công thức đệ quy của số Stirling loại 2 là:

$$s(n+1, k) = k \cdot s(n, k) + s(n, k-1).$$

❷ Chứng minh

Cách chứng minh khá tương tự số Stirling loại 1. Tuy nhiên ở đây việc phân một phần tử vào một tập hợp không xét tới thứ tự, điều này khác với chu trình cần xem xét thứ tự. Như vậy ta vẫn có hai trường hợp

1. Phân các phần tử 1, 2, ..., n vào k tập hợp. Sau đó phần tử $n+1$ sẽ được phân vào một trong k tập đó nên có k cách chọn.
2. Phân các phần tử 1, 2, ..., n vào $k-1$ tập hợp và phần tử $n+1$ vào tập hợp thứ $k+1$.

Từ đây ta có công thức số Stirling loại 2.

Các bước cơ sở cho số Stirling loại 2 cũng tương tự loại 1: * $s(n, n) = 1$; * $s(n, 0) = s(0, n) = 0$.

2.8.3 Biến đổi Fourier rời rạc

Phần này mình dịch từ [11].

Giới thiệu

Cho số $t \in \mathbb{N}$ và $n = 2^t$.

Gọi R là vành với đơn vị 1. Vành R chứa phần tử 2^{-1} là nghịch đảo của phần tử 2. Hơn nữa vành R cũng chứa phần tử ζ_{2^n} là một nghiệm cố định nào đó của phương trình $x^n + 1 = 0$. Đặt $\zeta_n = \zeta_{2^n}$.

❸ Example 13

Trên \mathbb{C} thì $\zeta_{2^n} = e^{\pi i/n}$ là một nghiệm của phương trình $x^n + 1 = 0$. Khi đó $\zeta_n = \zeta_{2^n}^2 = e^{2\pi i/n}$.

❹ Lemma 1

Phần tử ζ_{2^n} sinh ra một nhóm vòng (cyclic group) với order $2n$ trong nhóm nhân của vành R .

❶ Chứng minh

Do ζ_{2n} là nghiệm của phương trình $x^n + 1 = 0$ nên $\zeta_{2n}^n = -1$, hay $\zeta_{2n}^{2^t} = -1$. Từ đó suy ra $(\zeta_{2n}^{2^t})^2 = 1$ nên order của phần tử ζ_{2n} là $2^{t+1} = 2n$.

Giả sử $(f_0, \dots, f_{n-1}) \in R^n$ là vector bất kì. Khi đó:

Biến đổi Fourier loại 1 của vector trên là vector n chiều $(\hat{f}_0, \dots, \hat{f}_{n-1})$ thuộc R^n xác định bởi:

$$\hat{f}_i = \sum_{j=0}^{n-1} \zeta_n^{ij} f_j, \quad i = 0, 1, \dots, n-1.$$

Biến đổi Fourier loại 2 của vector trên là vector n chiều $(\check{f}_1, \check{f}_3, \dots, \check{f}_{2n-1})$ thuộc R^n xác định bởi:

$$\check{f}_i = \sum_{j=0}^{n-1} \zeta_{2n}^{ij} f_j, \quad i = 1, 3, \dots, 2n-1.$$

❷ Example 14

Mình lấy ví dụ với vector $(f_0, f_1, f_2) = (1, 2, 3)$ thuộc C^3 .

Chọn nghiệm $\zeta_{2n} = e^{\pi i/3}$ của phương trình $x^3 + 1 = 0$. Khi đó $\zeta_n = \zeta_{2n}^2 = e^{2\pi i/3}$.

Biến đổi Fourier loại 1 lúc này là:

$$\begin{aligned}\hat{f}_0 &= (e^{2\pi i/3})^{0 \cdot 0} \cdot 1 + (e^{2\pi i/3})^{0 \cdot 1} \cdot 2 + (e^{2\pi i/3})^{0 \cdot 2} \cdot 3 \\ &= 1 + 2 + 3 = 6. \\ \hat{f}_1 &= (e^{2\pi i/3})^{1 \cdot 0} \cdot 1 + (e^{2\pi i/3})^{1 \cdot 1} \cdot 2 + (e^{2\pi i/3})^{1 \cdot 2} \cdot 3 \\ &= 1 + 2 \cdot e^{2\pi i/3} + 3 \cdot e^{4\pi i/3} \\ \hat{f}_2 &= (e^{2\pi i/3})^{2 \cdot 0} \cdot 1 + (e^{2\pi i/3})^{2 \cdot 1} \cdot 2 + (e^{2\pi i/3})^{2 \cdot 2} \cdot 3 \\ &= 1 + 2 \cdot e^{4\pi i/3} + 3 \cdot e^{2\pi i/3}.\end{aligned}$$

Chú ý rằng $e^{2\pi i} = 1$ nên biểu thức cuối của \hat{f}_2 rút gọn còn $e^{2\pi i/3}$ ($8 \equiv 2 \pmod{3}$).

Tiếp theo ta biểu diễn $e^{4\pi i/3}$ theo $e^{2\pi i/3}$ như sau. Ta xét

$$\begin{aligned}e^{2zi} &= \cos 2z + i \sin 2z \\ &= 2 \cos^2 z - 1 + 2i \sin z \cos z \\ &= 2 \cos z (\cos z + i \sin z) - 1 \\ &= 2 \cos z \cdot e^{zi} - 1.\end{aligned}$$

Thay $z = 2\pi/3$ vào kết quả trên ta được

$$e^{4\pi i/3} = 2 \cos \frac{2\pi}{3} \cdot e^{2\pi i/3} - 1 = -e^{2\pi i/3} - 1.$$

Như vậy kết quả của biến đổi Fourier rời rạc bên trên là

$$(\hat{f}_0, \hat{f}_1, \hat{f}_2) = (6, -e^{2\pi i/3} - 2, e^{2\pi i/3} - 1).$$

Ở ví dụ trên chúng ta tính biến đổi Fourier cho vector có độ dài không phải lũy thừa của 2. Sau đây chúng ta sẽ xem xét một ví dụ với vector có độ dài là lũy thừa của 2 và ở phần sau sẽ tối ưu tính toán với biến đổi Fourier nhanh.

Example 15

Mình lấy ví dụ với vector $(f_0, f_1, f_2, f_3) = (1, 2, 3, 4)$ thuộc \mathbb{C}^4 .

Chọn nghiệm $\zeta_{2n} = e^{i\pi/4}$ của phương trình $x^4 + 1 = 0$. Khi đó $\zeta_n = \zeta_{2n}^2 = e^{i\pi/2} = i$.

Biến đổi Fourier loại 1 lúc này là:

$$\begin{aligned}\hat{f}_0 &= 1 \cdot 1 + 2 \cdot 1 + 3 \cdot 1 + 4 \cdot 1 = 10, \\ \hat{f}_1 &= 1 \cdot 1 + 2 \cdot i + 3 \cdot (-1) + 4 \cdot (-i) = -2 - 2i, \\ \hat{f}_2 &= 1 \cdot 1 + 2 \cdot (-1) + 3 \cdot 1 + 4 \cdot (-1) = -2, \\ \hat{f}_3 &= 1 \cdot 1 + 2 \cdot (-i) + 3 \cdot (-1) + 4 \cdot i = -2 + 2i.\end{aligned}$$

Đối với vector độ dài $n = 2^2$ thì $t = 2$, chúng ta cần $n - 1$ phép cộng và n phép nhân với lũy thừa của ζ_n để tính mỗi \hat{f}_i , với $i = 0, 1, 2, 3$. Tổng quát, ta cần tất cả $(n + n - 1) \cdot n$ phép tính cộng và nhân với lũy thừa của ζ_n . Độ phức tạp để tính biến đổi Fourier rời rạc là $O(n^2)$.

Remark 8

Các phần tử \hat{f}_i là giá trị của đa thức $F(x) = \sum_{j=0}^{n-1} f_j x^j \in R[x]$ tại điểm $x = \zeta_n^i$, các phần tử \check{f}_i là giá trị của đa thức $F(x)$ tại $x = \zeta_{2n}^i$ khi i lẻ.

Lemma 2

Ta có biểu diễn ngược như sau:

$$f_i = n^{-1} \sum_{j=0}^{n-1} \hat{f}_j \zeta_n^{-ij}, \quad (2.9)$$

$$f_i = n^{-1} \sum_{\substack{1 \leq j \leq 2n-1 \\ j \text{ lẻ}}} \check{f}_j \zeta_{2n}^{-ij}, \quad (2.10)$$

với $n^{-1} = (2^{-1})^t \in R$.

Chứng minh

Đặt $k \in \mathbb{Z}$. Khi đó $\zeta_{2n}^{-k} = \zeta_{2n}^{2nr-k}$ với $r \in \mathbb{Z}$, $2nr - k \geq 0$.

Ngoài ra, vì order của phần tử ζ_n trong phép nhân là n nên ta có:

$$\sum_{j=0}^{n-1} \zeta_n^{lj} = n \quad \text{khi } l \equiv 0 \pmod{n}, \quad (2.11)$$

$$\sum_{j=0}^{n-1} \zeta_n^{lj} = 0 \quad \text{khi} \quad l \not\equiv 0 \pmod{n}. \quad (2.12)$$

Phương trình (2.9) thỏa mãn vì:

$$\sum_{j=0}^{n-1} \hat{f}_j \zeta_n^{-ij} = \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} f_k \zeta_n^{jk} \zeta_n^{-ij} = \sum_{k=0}^{n-1} f_k \sum_{j=0}^{n-1} \zeta_n^{(k-i)j} = n f_i$$

theo (2.11) và (2.12), cụ thể là khi $k - i \equiv 0 \pmod{n}$.

Đẳng thức (2.10) có được từ

$$\begin{aligned} \sum_{\substack{1 \leq j \leq 2n-1 \\ j \text{ lẻ}}} \check{f}_j \zeta_{2n}^{-ij} &= \sum_{k=0}^{n-1} \check{f}_{2k+1} \zeta_{2n}^{-i(2k+1)} \\ &= \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} f_l \zeta_{2n}^{l(2k+1)} \zeta_{2n}^{-i(2k+1)} \\ &= \sum_{l=0}^{n-1} f_l \zeta_{2n}^{l-i} \sum_{k=0}^{n-1} \zeta_n^{(l-i)k}. \end{aligned}$$

Từ (2.11) và (2.12) suy ra được đẳng thức (2.10), cũng là khi $l - i \equiv 0 \pmod{n}$.

Việc tính toán biến đổi Fourier rời rạc bằng (2.9) và (2.10) cần $O(n^2)$ phép tính cộng và nhân trong vành R . Phản sau sẽ trình bày phương pháp tính biến đổi Fourier rời rạc với độ phức tạp $O(n \log n)$. Phương pháp này được gọi là **biến đổi Fourier nhanh** (hay **Fast Fourier Transform, FFT**, быстрое преобразование Фурье).

Biến đổi Fourier nhanh

Theorem 6

Biến đổi Fourier rời rạc $(\hat{f}_0, \dots, \hat{f}_{n-1})$ có thể được tính bằng:

- nt phép cộng trong vành R ;
- nt phép nhân với lũy thừa của ζ_n trong vành R .

Biến đổi Fourier rời rạc $(\check{f}_1, \dots, \check{f}_{2n-1})$ có thể được tính bằng:

- nt phép cộng trong vành R ;
- nt phép nhân với lũy thừa của ζ_{2n} trong vành R .

Nếu có (\hat{f}_i) và (\check{f}_i) thì có thể tính được vector (f_0, \dots, f_{n-1}) bằng:

- nt phép cộng trong vành R ;
- nt phép nhân với lũy thừa của ζ_{2n} trong vành R ;
- n phép nhân với $n^{-1} \in R$.

❶ Chứng minh

Đặt

$$\begin{aligned} F(x) &= \sum_{j=0}^{n-1} f_j x^j = \sum_{\substack{0 \leq j \leq n-1 \\ j \text{ chẵn}}} f_j x^j + \sum_{\substack{0 \leq j \leq n-1 \\ j \text{ lẻ}}} f_j x^j \\ &= F_0(x^2) + x F_1(x^2), \end{aligned}$$

với $\deg F_0(x), \deg F_1(x) < \frac{n}{2} = 2^{t-1}$.

Khi đó

$$F(\zeta_n^i) = F_0(\zeta_n^{2i}) + \zeta_n^i F_1(\zeta_n^{2i}), \quad (2.13)$$

với $i = 0, \dots, n-1$.

Đặt $\zeta_{n/2} = \zeta_n^2$. Khi đó

$$\{\zeta_n^{2i} : 0 \leq i \leq n-1\} = \left\{ \zeta_{n/2}^i : 0 \leq i \leq \frac{n}{2} - 1 \right\}.$$

Dựa trên quy nạp ta sẽ chứng minh biến đổi Fourier rời rạc loại 1, tức là tính $(F(\zeta_n^0), \dots, F(\zeta_n^{n-1}))$, theo công thức (2.13) bằng n phép cộng trong vành R và n phép tính nhân với lũy thừa của ζ_n trong vành R nếu ta đã biết các giá trị $F_0(\zeta_{n/2}^i)$ và $F_1(\zeta_{n/2}^i)$ với $i = 0, \dots, \frac{n}{2} - 1$.

Khi $t = 1$ và $n = 2 = 2^t$ (bước cơ sở quy nạp) thì để tính (\hat{f}_0, \hat{f}_1) ta cần tìm $F_0 + \zeta_2^i F_1$ với F_0 và F_1 là các phần tử thuộc vành R , $i = 0, 1$. Như vậy, với $n = 2$ thì cần:

- $2 = nt$ phép nhân với lũy thừa của $\zeta_n = \zeta_2$ trong vành R , ứng với $\zeta_2^i F_1$ khi $i = 0, 1$;
- $2 = nt$ phép cộng trong vành R , ứng với $F_0 + \zeta_2^i F_1$.

Giả sử với mọi $j < t$, để tính biến đổi Fourier rời rạc loại 1 trên vector 2^j chiều ta cần:

- $2^j \cdot j$ phép nhân với lũy thừa của $\zeta_{2^j} = (\zeta_n)^{2^{t-j}}$ trong vành R ;
- $2^j \cdot j$ phép cộng trong vành R .

Khi đó, nếu $j = t$ thì việc tính $(\hat{f}_0, \dots, \hat{f}_{n-1})$ theo công thức (2.13) bao gồm tính $F_0(\zeta_{n/2}^i)$ và $F_1(\zeta_{n/2}^i)$ với $i = 0, \dots, n-1$. Nói cách khác là tính biến đổi Fourier cho vector độ dài $n/2$ gồm các hệ số $F_0(x)$ và $F_1(x)$ cộng thêm:

- n phép cộng trong vành R ;
- n phép nhân với lũy thừa của ζ_n trong vành R .

Khi đó, theo giả thiết quy nạp, việc tính vector $(\hat{f}_0, \dots, \hat{f}_{n-1})$ cần không nhiều hơn:

- n phép cộng trong vành R ;
- cộng với n phép nhân trong vành R ;
- cộng với $2 \cdot 2^{t-1} \cdot (t-1)$ phép nhân với lũy thừa của ζ_n trong vành R .

Như vậy cần tổng cộng $n + n(t-1) = nt$ phép cộng trong vành R và $n + n(t-1) = nt$ phép nhân với lũy thừa của ζ_n trong vành R . Như vậy mệnh đề đầu tiên của định lý được chứng minh.

Mệnh đề thứ hai chứng minh tương tự, ta thay ζ_n thành ζ_{2n} .

Mệnh đề thứ ba suy ra từ hai mệnh đề trên và [Bổ đề 2](#).

❶ Example 16

Xét vector $(f_0, f_1, f_2, f_3) = (1, 2, 3, 4)$ thuộc \mathbb{C}^4 như bên trên.

Ở đây, $n = 4 = 2^2$ và $t = 2$.

Chọn nghiệm $\zeta_8 = e^{i\pi/4}$ của phương trình $x^4 + 1 = 0$. Khi đó $\zeta_4 = \zeta_8^2 = e^{i\pi/2} = i$.

Ta chia vector thành hai phần:

- vector con gồm các phần tử ở vị trí chẵn là $(f_0, f_2) = (1, 3)$;
- vector con gồm các phần tử ở vị trí lẻ là $(f_1, f_3) = (2, 4)$.

Đặt $\zeta_2 = \zeta_4^2 = -1$.

Khi đó, với vector con $(f_0, f_2) = (1, 3)$ ta tính:

$$\begin{aligned} F_0 &= f_0 + \zeta_2^0 \cdot f_2 = 1 + 1 \cdot 3 = 4, \\ F_1 &= f_0 + \zeta_2^1 \cdot f_2 = 1 + (-1) \cdot 3 = -2. \end{aligned}$$

Tương tự, với vector con $(f_1, f_3) = (2, 4)$ ta tính

$$\begin{aligned} F_2 &= f_1 + \zeta_2^0 \cdot f_3 = 2 + 1 \cdot 4 = 6, \\ F_3 &= f_1 + \zeta_2^1 \cdot f_3 = 2 + (-1) \cdot 4 = -2. \end{aligned}$$

Kết hợp hai vector (F_0, F_2) và (F_3, F_4) ta tính được biến đổi Fourier nhanh.

$$\begin{aligned} \hat{f}_0 &= F_0 + \zeta_4^0 \cdot F_2 = 4 + 1 \cdot 6 = 10, \\ \hat{f}_2 &= F_0 + \zeta_4^2 \cdot F_2 = 4 + (-1) \cdot 6 = -2, \\ \hat{f}_1 &= F_1 + \zeta_4^1 \cdot F_3 = -2 + i \cdot (-2) = -2 - 2i, \\ \hat{f}_3 &= F_1 + \zeta_4^3 \cdot F_3 = -2 + (-i) \cdot (-2) = -2 + 2i. \end{aligned}$$

Ở đây, hệ số \hat{f}_i ở vị trí lẻ sẽ được tính nhờ các F_i ở vị trí lẻ cùng với lũy thừa lẻ của ζ_n , và ngược lại, hệ số \hat{f}_i ở vị trí chẵn sẽ được tính nhờ các F_i ở vị trí chẵn cùng với lũy thừa chẵn của ζ_n .

Như vậy kết quả là

$$(\hat{f}_0, \hat{f}_1, \hat{f}_2, \hat{f}_3) = (10, -2 - 2i, -2, -2 + 2i),$$

giống với ví dụ ở trên. Tuy nhiên chúng ta chỉ dùng 4 phép cộng và 4 phép nhân với ζ_2 để tính các F_i , cộng thêm 4 phép cộng và 4 phép nhân với ζ_4 để tính các \hat{f}_i . Như vậy tổng cộng ta dùng $8 = nt$ phép cộng, và $8 = nt$ phép nhân cho các lũy thừa của ζ_4 (vì ζ_2 cũng là lũy thừa của ζ_4).

Biến đổi Fourier và phép nhân đa thức

❷ Lemma 3

Một phép nhân trong vành $R[x]/(x^{2^t} + 1)$ có thể được tính bởi:

- $n = 2^t$ phép nhân trong vành R ;
- $3nt$ phép cộng trong vành R ;
- $3nt$ phép nhân với lũy thừa của ζ_{2^n} trong vành R ;

- n phép nhân với n^{-1} trong vành R .

➊ Remark 9

Bố đề không áp dụng khi $R = \mathbb{Z}$ và $R = \mathbb{Q}$ vì hai vành này không có phần tử ζ_{2n} (căn bậc n của đơn vị).

Sau đây ta chứng minh *Bố đề 3*.

➊ Chứng minh

Đặt $\sum_{i=0}^{n-1} f_i x^i$ và $\sum_{i=0}^{n-1} g_i x^i$ với $F, G \in R[x]$ và là biểu diễn của phần tử trong lớp nhân tử $R[x]/(x^{2^t} + 1)$.

Đặt $H = \sum_{i=0}^{n-1} h_i x^i \in R[x]$ sao cho $F \cdot G \equiv H \pmod{x^n + 1}$. Nói cách khác H là tích $F \cdot G$ trong vành và ta sẽ tính H .

Biến đổi Fourier loại 2 cho vector (f_0, \dots, f_{n-1}) và (g_0, \dots, g_{n-1}) cho ta đẳng thức:

$$\check{f}_i \cdot \check{g}_i = F(\zeta_{2n}^i) \cdot G(\zeta_{2n}^i) = H(\zeta_{2n}^i) = \check{h}_i$$

với i lẻ, $1 \leq i \leq 2n - 1$, vì $\zeta_{2n}^n + 1 = 0$.

Như vậy nếu ta biết tất cả \check{f}_i và \check{g}_i thì có thể tính mọi \check{h}_i với n phép nhân trong vành R .

Theo *Định lí 6*, các phần tử \check{f}_i và \check{g}_i có thể tính với $2tn$ phép cộng trong vành R và $2tn$ phép nhân với lũy thừa của ζ_{2n} trong vành R .

Cũng theo *Định lí 6*, các phần tử \check{h}_i có thể tính, với điều kiện đã biết \check{h}_i , với

- tn phép cộng trong vành R ;
- tn phép nhân với lũy thừa của ζ_{2n} trong vành R
- n phép nhân với phần tử n^{-1} trong vành R .

Ta có điều phải chứng minh.

➊ Corollary 1

Đặt T là vành giao hoán với đơn vị, $2^{-1} \in T$, $\zeta_{4n} = \zeta_{2^{t+2}} \in T$ là nghiệm phương trình $x^{2n} + 1 = 0$. Nếu $F(x), G(x) \in T[x]$, $\deg F(x) < n$, $\deg G(x) < n$ thì tích $F(x) \cdot G(x)$ có thể tính với:

- $2n$ phép nhân trong T ;
- $6n(t+1)$ phép cộng trong T ;
- $6n(t+1)$ phép nhân với lũy thừa của ζ_{4n} trong T ;
- $2n$ phép nhân với $(2^{-1})^{t+1}$ trong T .

➊ Chứng minh

Vì chẵn trên của bậc đa thức $F(x)$ và $G(x)$ nên bậc của $F(x) \cdot G(x)$ sẽ nhỏ hơn $2n$. Khi chia $F(x) \cdot G(x)$ cho $x^{2n} + 1$ ta thu được chính đa thức $F(x) \cdot G(x)$. Khi đó tích $F(x) \cdot G(x)$ trong $R[x]$ có thể tính với một phép nhân trong $R[x]/(x^{2n} + 1)$. Từ [Bố đề 1](#), ta thay n thành $2n$ và t thay thành $t + 1$, kết hợp [Bố đề 3](#) ta có điều phải chứng minh.

Đặt T là vành giao hoán với đơn vị 1 và có phần tử 2^{-1} .

Ta hiểu phép cộng trong T là cả phép cộng lăn phép trừ (cộng cho số đối). Ta cần bổ đẽ cơ sở sau.

i Lemma 4

Nếu $t \geq 2$ thì một phép nhân trong vành $T[x]/(x^{2^t} + 1)$ có thể thực hiện với:

- $O(2^t \cdot t)$ phép nhân trong T ;
- $O(2^t \cdot t \cdot \log t)$ phép cộng trong T .

i Remark 10

Kết quả trên có thể áp dụng với vành số nguyên \mathbb{Z} nếu ta xem T là vành như sau:

$$T = \left\{ \frac{m}{2^k} : m \in \mathbb{Z}, k \in \mathbb{Z}_{\geq 0} \right\}, \quad T \supseteq \mathbb{Z}.$$

Trên máy tính, mỗi phần tử của T được lưu chính xác, ví dụ dưới dạng cặp (m, k) .

Ta sẽ xem xét định lí quan trọng tiếp theo rồi quay lại chứng minh [Bố đẽ 4](#).

i Theorem 7

Phép nhân hai đa thức có bậc không quá n trong vành $T[x]$, với $n \geq 3$, được tính với $M(n) = O(n \log n)$ phép nhân trong T và $A(n) = O(n \log n \log \log n)$ phép cộng trong T .

i Chứng minh

Gọi $t \in \mathbb{N}$ là số sao cho $2^{t-1} \leq 2n < 2^t$. Dễ thấy $t \geq 2$ và $2n < 2^t < 4n$.

Do đó phép nhân hai đa thức có bậc không quá n theo modulo $x^{2^t} + 1$ là phép nhân thông thường, kết quả không thay đổi sau phép modulo.

Theo [Bố đẽ 4](#) thì ta có thể tính tích với $M(n) = O(2^t \cdot t) = O(n \log n)$ phép nhân trong T và $A(n) = O(2^t \cdot t \log t) = O(n \cdot \log n \cdot \log \log n)$ phép cộng trong T .

Tiếp theo chúng ta quay lại chứng minh [Bố đẽ 4](#).

i Chứng minh

Đặt $F = F(x)$ và $G = G(x)$ là các đa thức bậc không quá $2^t - 1$. Đặt

$$H = H(x) = \sum_{i=0}^{2^t-1} H_i x^i$$

với $H = FG \pmod{x^{2^t} + 1}$. Ta cần tính các hệ số $H_0, H_1, \dots, H_{2^t-1}$ theo các hệ số của F và G .

Đặt k là tham số tự nhiên với $1 \leq k < t$ mà ta sẽ chọn ở dưới.

Ta biểu diễn F và G dưới dạng

$$F = \sum_{i=0}^{2^{t-k}-1} f_i(x) x^{i \cdot 2^k}, \quad G = \sum_{i=0}^{2^{t-k}-1} g_i(x) x^{i \cdot 2^k}$$

với

- $f_i(x), g_i(x) \in T[x]$;
- $\deg f_i(x) \leq 2^k - 1$;
- $\deg g_i(x) \leq 2^k - 1$.

Thuật toán tính $F \cdot G$ được thực hiện theo các bước sau.

Bước 1. Nhân $\sum_{i=0}^{2^{t-k}-1} f_i(x) Y^i$ với $\sum_{i=0}^{2^{t-k}-1} g_i(x) Y^i$ trên vành $T[x, Y]/(Y^{2^{t-k}-1})$. Kết quả được biểu diễn bởi $\tilde{H} = \sum_{i=0}^{2^{t-k}-1} h_i(x) Y^i$.

Bước 2. Thay $Y = x^{2^k}$ vào \tilde{H} vào tính modulo $x^{2^t} + 1 = Y^{2^{t-k}} + 1$. Khi đó

$$F(x) \cdot G(x) = \sum_{l=0}^{2^{t-k}-1} f_l(x) x^{2^k l} \cdot \sum_{j=0}^{2^{t-k}-1} g_j(x) x^{2^k j} \equiv \sum_{i=0}^{2^{t-k}-1} h_i(x) x^{2^k i} \pmod{(x^{2^k})^{2^{t-k}} + 1}.$$

Từ bước 2 ta sẽ tính được $H(x) = \sum_{i=0}^{2^t-1} H_i x^i$. Ta cần hiểu tại sao ở bước 2 dãy $\{h_i(x)\}$ theo modulo $x^{2^t} + 1$ tìm được hệ số H_i của $H(x)$.

Ở bước 1 ta nhân

$$\sum_{l=0}^{2^{t-k}-1} f_l(x) Y^l \cdot \sum_{j=0}^{2^{t-k}-1} g_j(x) Y^j \equiv \sum_{i=0}^{2^{t-k}-1} h_i(x) Y^i \pmod{Y^{2^{t-k}} + 1}.$$

Ở đây $l + j \leq 2^{t-k+1} - 2 < 2^{t-k+1} - 1$. Khi đó có hai trường hợp có thể xảy ra.

Trường hợp 1. Nếu $0 \leq l + j \leq 2^{t-k} - 1$ thì

$$Y^l \cdot Y^j \equiv Y^{l+j} \pmod{Y^{2^{t-k}} + 1}.$$

Trường hợp 2. Nếu $2^{t-k} \leq l + j \leq 2 \cdot 2^{t-k} - 2$ thì

$$Y^{l+j} \equiv -Y^i \pmod{Y^{2^{t-k}} + 1},$$

với $i = l + j - 2^{t-k}$.

Khi đó

$$h_i(x) = \sum_{\substack{l,j \\ j+j=i}} f_l(x)g_j(x) - \sum_{\substack{l,j \\ l+j=i+2^{t-k}}} f_l(x)g_j(x),$$

với $i = 0, \dots, 2^{t-k} - 1$.

Từ đây suy ra

$$\deg h_i(x) \leq \deg f_l(x) = \deg g_j(x) \leq 2^{k+1} - 2 < 2^{k+1} - 1.$$

Vì $k < t$ nên $\deg h_i < 2^t - 1$.

Bước 2 thực hiện không nhiều hơn 2^t phép cộng trên T vì nếu chúng ta đã biết các giá trị $h_i(x) = \sum_{j=0}^{2^{k+1}-1} h_{ij}x^j$ thì khi thay $Y = x^{2^k}$ ta có biểu thức

$$\sum_{i=0}^{2^{t-k}-1} \sum_{j=0}^{2^{k+1}-1} h_{ij}x^{j+2^k i} \pmod{x^{2^t} + 1}.$$

Phép tính modulo $x^{2^t} + 1$ ta sẽ được các biểu thức với $j + 2^k i \geq 2^t$.

Khi đó, với $j + 2^k i = r2^t + l$, $0 \leq l < 2^t$, nếu thay $x^{j+2^k i}$ thành $(-1)^r \cdot x^l$ thì đại lượng $(-1)^r \cdot h_{ij}$ sẽ được thêm vào hệ số của x^l (tức là thực hiện một phép cộng hoặc trừ). Các phép cộng như vậy sẽ không lớn hơn số lượng số hạng, tức là không lớn hơn $2^{t-k} \cdot 2^{k+1} = 2^{t+1}$.

Ở đây, khi $0 \leq i \leq 2^{t-k+1}$ ta có bất đẳng thức

$$j + 2^k \cdot i \leq 2^{k+1} - 1 + 2^{t-1} - 2^k = 2^k + 2^{t-1} - 1 \leq 2^t - 1$$

vì $k \leq t - 1$. Với những số i như vậy thì phép tính modulo là không cần thiết.

Lúc này còn các giá trị i trong đoạn $2^{t-k-1} \leq i \leq 2^{t-k} - 1$, số lượng các giá trị này không nhiều hơn 2^{t-k-1} . Khi đó, cặp (i, j) ở bước 2 sẽ triệt tiêu theo modulo, và có không nhiều hơn $2^{t-k-1} \cdot 2^{k+1} = 2^t$ cặp. Như vậy ta đã chứng minh được ở bước 2 thực hiện không nhiều hơn 2^t phép cộng trong T .

Bây giờ xét bước 1. Phép nhân ta không thực hiện trong vành $T[x, Y]/(Y^{2^{t-k}} - 1)$ mà trong vành $R[Y]/(Y^{2^{t-k}} - 1)$ với $R = T[x]/(x^{2^{k+1}} + 1)$.

Trong vành R có phần tử $\zeta_{2^{k+2}} \equiv x \pmod{x^{2^{k+1}} + 1}$ là nghiệm phương trình $X^{2^{k+1}} + 1 = 0$.

Chọn $k = \left[\frac{t}{2} \right] \geq 1$. Khi đó

$$k \geq \frac{t-1}{2}, \quad k \leq \frac{t}{2} < t.$$

Vì $2^{t-k+1} \leq 2^{k+2}$ nên trong R có phần tử $\zeta_{2^{t-k+1}}$ là một lũy thừa của phần tử $\zeta_{2^{k+2}}$.

Một phép nhân cho lũy thừa của phần tử $\zeta_{2^{t-k+1}}$ trong vành R được thực hiện bởi 2^{k+1} phép cộng trong T .

Vì $x \equiv \zeta_{2^{k+2}} \pmod{x^{2^{k+1}} + 1}$ nên

$$R = \left\{ a_0 + a_1\zeta_{2^{k+2}} + \dots + a_{2^{k+1}}\zeta_{2^{k+2}}^{2^{k+1}-1} : a_i \in T, i = 0, 1, \dots, 2^{k+1} - 1 \right\}.$$

Phép nhân với $\zeta_{2^{t-k+1}}^j = \zeta_{2^{k+2}}^j$ với đẳng thức $\zeta_{2^{k+2}}^{2^{k+1}} = -1$ sẽ cho kết quả là hoán vị các hệ số $a_0, a_1, \dots, a_{2^{k+1}-1}$ và một số phần tử sẽ đổi dấu.

Ta dùng [Bước 3](#) để đánh giá độ phức tạp của một phép nhân trong vành $R[Y]/(Y^{2^{t-k}} + 1)$.

Ta cần thực hiện:

- 2^{t-k} phép nhân trong R ;
- $3 \cdot (t-k) \cdot 2^{t-k}$ phép cộng trong R (ở đây có $3 \cdot (t-k) \cdot 2^{t-k} \cdot 2^{k+1}$ phép cộng trong T vì các phần tử của vành R được biểu diễn bởi đa thức bậc không quá $2^{k+1} - 1$ trong $T[x]$);
- $3 \cdot (t-k) \cdot 2^{t-k}$ phép nhân với lũy thừa của $\zeta_{2^{t-k+1}}$;
- 2^{t-k} phép nhân với $(2^{-1})^{t-k}$ (lần nữa, cần $2^{t-k} \cdot 2^{k+1} = 2^{t+1}$ phép nhân trong T cho phần tử $(2^{-1})^{t-k}$).

Tổng cộng ta cần:

- 2^{t-k} phép nhân trong R ;
- $6 \cdot (t-k) \cdot 2^{k+1}$ phép cộng trong T ;
- 2^{t+1} phép nhân cho $(2^{-1})^{t-k}$ trong T .

Ta tính thêm 2^t phép cộng trong T ở bước 2, đặt

$$k(t) = k + 1 = \left\lceil \frac{t}{2} \right\rceil + 1 \geq 2. \quad (2.14)$$

Khi đó, một phép nhân trong $T[x]/(x^{2^{k+1}} + 1)$ được thực hiện bởi $2^{t-k(t)+1}$ phép nhân trong vành $R = T[x]/(x^{2^{k(t)}} + 1)$, cộng với không nhiều hơn $12 \cdot t \cdot 2^t$ phép cộng trong T , cộng với 2^{t+1} phép nhân trong T .

Khi $t \geq 3$ thì $k(t) < t$. Ta chuyển phép nhân ở trên trong vành $T[x]/(x^{2^t} + 1)$ thành phép nhân trong vành $T[x]/(x^{2^{k(t)}} + 1)$, và đi xuống tiếp cho tới khi gấp vành $T[x]/(x^4 + 1)$. Ở bước cuối này phép nhân bắt kì đều cần $O(1)$ phép cộng và phép nhân trong T .

Ta ký hiệu $M_1(2^j)$ và $A_1(2^j)$ là số lượng phép nhân và cộng trong T , tương ứng với số lượng cần thiết để thực hiện phép nhân ở trên trong vành $T[x]/(x^{2^j} + 1)$.

Khi đó, với $t \geq 3$ ta có bất đẳng thức

$$\begin{aligned} M_1(2^t) &\leq 2^{t-k(t)+1} \cdot M_1(2^{k(t)}) + 2^{t+1}, \\ A_1(2^t) &\leq 2^{t-k(t)+1} \cdot A_1(2^{k(t)}) + 12 \cdot t \cdot 2^t, \end{aligned}$$

với $k(t)$ định nghĩa ở (2.14).

Đặt $\alpha(j) = \frac{A_1(2^j)}{2^j}$ với $j = 2, 3, \dots$

Khi đó $\alpha(j) \leq 2\alpha(k(t)) + 12t$.

Giả sử ta có bất đẳng thức sau với $2 \leq j \leq t$:

$$\alpha(j) \leq c \cdot j \log j$$

với hằng số tuyệt đối c nào đó.

Khi đó ta có bất đẳng thức

$$\alpha(t) \leq 2c \cdot k(t) \cdot \log(k(t)) + 12t \leq 2c \left(\frac{t}{2} + 1 \right) \log \left(\frac{t}{2} + 1 \right) + 12t < c \cdot t \log t$$

khi hằng số c đủ lớn.

Như vậy

$$A_1(2^t) \leq 2^t \cdot c \cdot t \log t = O(2^t \cdot t \log t)$$

và ta đã chứng minh được ý thứ hai của *Bố đê 4* về phép cộng.

Bây giờ, lại đặt $\beta(j) = \frac{M_1(2^j)}{2^j}$ với $j = 2, 3, \dots$

Ta có bất đẳng thức

$$\beta(t) \leq 2\beta(k(t)) + 2.$$

Từ đây suy ra

$$\beta(t) \leq 2(2\beta(k(k(t))) + 2) + 2 = 2^2\beta(k(k(t))) + 2^2 + 2,$$

và tương tự như vậy.

Vì $k(t) \leq \frac{t}{2} + 1$ nên

$$k(k(t)) \leq \frac{1}{2} \left(\frac{t}{2} + 1 \right) + 1 = \frac{t}{2^2} + 1 + \frac{1}{2}; \dots; k(k(\dots(k(t)\dots))) < \frac{t}{2^j} + 2$$

với mọi $j \geq 1$. Vì vậy khi $j = \lceil \log_2 t \rceil$ ta có bất đẳng thức

$$\beta(t) \leq 2^j \cdot c_1 + 2 + 2^2 + \dots + 2^j < 2^j \cdot c_1 + 2^{j+1} \leq c_2 t$$

với c_1, c_2 là các hằng số tuyệt đối.

Như vậy $M_1(t) = O(t \cdot 2^t)$ và ta đã chứng minh xong ý đầu của *Bố đê 4*.

Ví dụ nhân đa thức với Sympy

Để ví dụ, mình sẽ viết chương trình Python nhân hai đa thức bậc 4 trên \mathbb{Q} trong modulo $x^4 + 1$.

Thuật toán ở các chứng minh trên là thuật toán Cooley-Tukey FFT.

Đầu tiên mình sẽ viết hàm đảo ngược bit (dùng cho radix-2 decimal-in-time).

```
def _reverse_bit(n: int, nbits: int) -> int:
    m = 0
    for _ in range(nbits):
        m = (m << 1) + (n & 1)
        n = n >> 1
    return m

assert _reverse_bit(3, 3) == 6
assert _reverse_bit(1, 3) == 4
```

FFT là thuật toán chia để trị -- để tính toán FFT cho vector độ dài $n = 2^t$, ta sẽ tính trên hai vector độ dài $n/2$ và kết hợp kết quả lại.

```
from sympy import pi, sin, cos, I, simplify
from sympy.codegen.cfunctions import log2
```

(continues on next page)

(continued from previous page)

```

def _mult_W(v: list[int]) -> list[int]:
    n = len(v)
    w = cos(2 * pi / n) + I * sin(2 * pi / n)
    u = [0] * n
    for i in range(0, n // 2):
        u[i] = v[i] + w**i * v[i + n // 2]
        u[i + n // 2] = v[i] + w**(i + n // 2) * v[i + n // 2]
    return u

def _fft(v: list[int]) -> list[int]:
    n = len(v)

    if n == 2:
        return _mult_W(v)
    else:
        u = _fft(v[:n // 2]) + _fft(v[n // 2:])
        return _mult_W(u)

def fft(v: list[int]) -> list[int]:
    n = len(v)
    b = int(log2(n))
    idx = [_reverse_bit(i, b) for i in range(n)]
    v = [v[idx[i]] for i in range(n)]

    return list(map(simplify, _fft(v)))

def _mult_iW(v: list[int]) -> list[int]:
    n = len(v)
    w = cos(-2 * pi / n) + I * sin(-2 * pi / n)
    u = [0] * n
    for i in range(0, n // 2):
        u[i] = v[i] + w**i * v[i + n // 2]
        u[i + n // 2] = v[i] + w**(i + n // 2) * v[i + n // 2]
    return u

def _ifft(v: list[int]) -> list[int]:
    n = len(v)
    if n == 2:
        return _mult_iW(v)
    else:
        u = _ifft(v[:n // 2]) + _ifft(v[n // 2:])
        return _mult_iW(u)

def ifft(v: list[int]) -> list[int]:
    n = len(v)
    b = int(log2(n))
    idx = [_reverse_bit(i, b) for i in range(n)]
    v = [v[idx[i]] for i in range(n)]

    return [simplify(i / n) for i in _ifft(v)]

```

Tiếp theo mình kiểm tra các hàm đã viết bên trên.

```

assert fft([2, 3, 0, 0]) == [5, 2 + 3*I, -1, 2 - 3*I]
assert ifft([5, 2 + 3*I, -1, 2 - 3*I]) == [2, 3, 0, 0]

import random
v = [random.randint(-4, 4) for _ in range(8)]
assert ifft(fft(v)) == v, fft(v)

```

Ví dụ, để nhân hai đa thức

$$a(x) = 1 + 2x - x^2 + 3x^3,$$

$$b(x) = -1 - 4x + 3x^2 - 2x^3,$$

trong modulo $x^4 + 1$, đầu tiên mình viết hệ số của mỗi đa thức thành vector theo số mũ tăng dần, như vậy

$$\mathbf{a} = (1, 2, -1, 3), \quad \mathbf{b} = (-1, -4, 3, -2).$$

Tiếp theo, mình sẽ thêm vào sau các vector \mathbf{a} và \mathbf{b} các số 0 để đạt độ dài là lũy thừa của 2 nhưng lớn hơn bậc của modulo. Ở đây modulo là $x^4 + 1$ nên mình sẽ thêm các số 0 để đạt độ dài $4 \cdot 2 = 8$.

$$\mathbf{a}' = (1, 2, -1, 3, 0, 0, 0, 0), \quad \mathbf{b}' = (-1, -4, 3, -2, 0, 0, 0, 0).$$

Tiếp theo, mình áp dụng FFT cho hai vector \mathbf{a}' và \mathbf{b}' .

```

a = [1, 2, -1, 3, 0, 0, 0, 0]
b = [-1, -4, 3, -2, 0, 0, 0, 0]

fa = fft(a)
fb = fft(b)

```

Đặt

$$\mathbf{f}_a = \text{FFT}(\mathbf{a}') = (a_0, a_1, \dots, a_7),$$

$$\mathbf{f}_b = \text{FFT}(\mathbf{b}') = (b_0, b_1, \dots, b_7).$$

Khi đó ta tính vector \mathbf{f}_c theo là tích theo cặp của \mathbf{f}_a và \mathbf{f}_b , nghĩa là

$$\mathbf{f}_c = (c_0, c_1, \dots, c_7), \quad c_i = a_i \cdot b_i.$$

Tiếp theo, ta tính \mathbf{c}' là biến đổi Fourier rời rạc ngược của \mathbf{f}_c :

$$\mathbf{c}' = \text{FFT}^{-1}(\mathbf{f}_c) = (C_0, C_1, \dots, C_7).$$

Khi đó, ta tính vector \mathbf{c} độ dài 4 theo công thức

$$\mathbf{c} = (C_i - C_{i+4})_{i=0}^3.$$

```

fc = [i * j for i, j in zip(fa, fb)]
fc = list(map(simplify, ifft(fc)))
c = [fc[i] - fc[i + 4] for i in range(4)]
print(c)

```

Kết quả là vector $(18, -17, 2, 5)$.

Kiểm tra kết quả bằng tay, mình có

$$\begin{aligned}
 a(x) \cdot b(x) &= (1 + 2x - x^2 + 3x^3) \cdot (-1 - 4x + 3x^2 - 2x^3) \\
 &= -1 - 4x + 3x^2 - 2x^3 \\
 &\quad - 2x - 8x^2 + 6x^3 - 4x^4 \\
 &\quad + x^2 + 4x^3 - 3x^4 + 2x^5 \\
 &\quad - 3x^3 - 12x^4 + 9x^5 - 6x^6 \\
 &= -1 - 6x - 4x^2 + 5x^3 - 19x^4 + 11x^5 - 6x^6.
 \end{aligned}$$

Trong modulo $x^4 + 1$ ta có thể thay thế x^4 thành -1 , như vậy kết quả trên tương đương

$$\begin{aligned}
 a(x) \cdot b(x) &= -1 - 6x - 4x^2 + 5x^3 - 19x^4 + 11x^5 - 6x^6 \\
 &= -1 - 6x - 4x^2 + 5x^3 - 19 \cdot (-1) + 11x \cdot (-1) - 6x^2 \cdot (-1) \\
 &= 18 - 17x + 2x^2 + 5x^3 \bmod x^4 + 1.
 \end{aligned}$$

Nếu viết hệ số của kết quả dưới dạng vector theo số mũ tăng dần thì mình có $(18, -17, 2, 5)$, bằng đúng phép tính với thuật toán Cooley-Tukey.

Một lý do khiến mình viết code trong khi nhiều thư viện Python như numpy, sympy, sagemath đã hỗ trợ là vì trong [11] (tài liệu tham khảo chính của bài viết này) thì DFT thuận sử dụng ζ_{2n} là nghiệm của $x^n + 1$ nên nếu xét trên \mathbb{C} thì $\zeta_{2n} = e^{i\pi/n}$. Trong khi đó các tài liệu khác như wikipedia, numpy, sympy, ... sử dụng $e^{-i\pi/n}$ để tính DFT thuận. Ngược lại, [11] sử dụng $e^{-i\pi/n}$ để tính DFT ngược, trong khi các tài liệu khác dùng $e^{i\pi/n}$. Ở đây chúng ta có thể thấy dù là cách nào thì để thực hiện phép nhân nhanh đa thức modulo $x^{2^t} + 1$ đều cần cả DFT thuận và ngược, nên kết quả không thay đổi nhờ vào tính chất của DFT.

2.8.4 Đại cương tổ hợp

Chúng ta có hai quy tắc đếm, và những công thức đếm khác đều được sinh ra cũng như tuân theo hai quy tắc này. Đó là quy tắc cộng và quy tắc nhân.

Quy tắc cộng và quy tắc nhân

Khi một công việc có thể thực hiện bằng một trong nhiều phương án, ta sử dụng quy tắc cộng. Ví dụ, nếu chúng ta muốn lấy một cây bút trong hộp có 3 cây bút đỏ, 4 cây bút xanh và 5 cây bút vàng, thì chúng ta có tất cả $3 + 4 + 5 = 12$ cách lấy.

Khi một công việc được thực hiện trên nhiều công đoạn, ở mỗi công đoạn có nhiều phương án thì ta sử dụng quy tắc nhân. Ví dụ, nếu chúng ta đi từ thành phố A tới thành phố B, giữa đường đi ngang thành phố C. Từ A tới C có 4 con đường, từ C tới B có 3 con đường thì có tất cả $4 \cdot 3 = 12$ con đường đi từ A tới B mà đi ngang qua C.

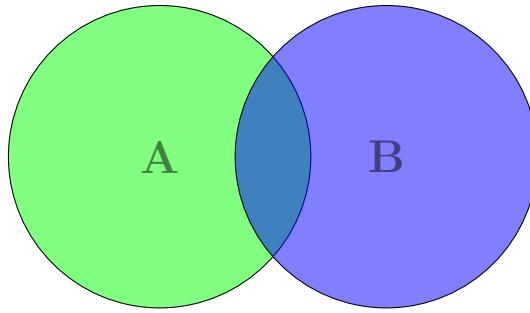
Nguyên lý bù trừ. Ở quy tắc cộng, khi các phương án rời nhau, thì ta cộng chúng lại. Tuy nhiên, khi các phương án có sự giao nhau thì chúng ta sử dụng nguyên lý bù trừ, hay còn gọi là quy tắc cộng mở rộng.

Gọi A và B là hai tập hợp. Kí hiệu $|A \cup B|$ là số lượng phần tử của A hợp B và $|A \cap B|$ là số lượng phần tử của A giao B .

Khi đó

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

Sử dụng sơ đồ Venn để mô tả công thức trên:



Hình 2.49: Nguyên lý bù trừ cho hai tập hợp

Ta thấy rằng, khi chúng ta cộng số phần tử của hai tập hợp lại với nhau thì phần giao của chúng bị trùng. Do đó ta phải trừ đi một lần phần giao thì mới có hợp của hai tập hợp.

i Example 17

Trong một lớp học có 20 học sinh. Trong đó có 13 học sinh biết chơi bóng chuyền, 15 học sinh biết chơi bóng bàn. Biết rằng học sinh nào cũng biết chơi bóng chuyền hoặc bóng bàn. Hỏi có bao nhiêu học sinh biết chơi cả hai môn bóng chuyền và bóng bàn?

i Giải

Đặt A là tập hợp các học sinh biết chơi bóng chuyền. Ta có $|A| = 13$.

Đặt B là tập hợp các học sinh biết chơi bóng bàn. Ta có $|B| = 15$.

Do lớp học có 20 học sinh và học sinh nào cũng biết chơi hoặc bóng chuyền, hoặc bóng bàn, nên $|A \cup B| = 20$.

Theo nguyên lý bù trừ, số lượng học sinh biết chơi cả hai bộ môn là

$$|A \cap B| = |A| + |B| - |A \cup B| = 13 + 15 - 20 = 8.$$

Như vậy có 8 học sinh biết chơi cả hai môn.

Ta có thể có công thức bù trừ cho ba tập hợp.

$$|A \cup B \cup C| = |A| + |B| + |C| - (|A \cap B| + |B \cap C| + |C \cap A|) + |A \cap B \cap C|.$$

Trong trường hợp tổng quát cho n tập hợp A_1, A_2, \dots, A_n thì công thức bù trừ là:

$$\begin{aligned} |A_1 \cup A_2 \cup \dots \cup A_n| &= \sum_{i=1}^n |A_i| \\ &\quad - \sum_{1 \leq i_1, i_2 \leq n, i_1 \neq i_2} |A_{i_1} \cap A_{i_2}| \\ &\quad + \sum_{1 \leq i_1, i_2, i_3, i_1 \neq i_2 \neq i_3 \neq i_1} |A_{i_1} \cap A_{i_2} \cap A_{i_3}| + \dots \\ &\quad + (-1)^{t-1} \sum_{1 \leq i_1, \dots, i_t, i_1 \neq \dots \neq i_t} |A_{i_1} \cap \dots \cap A_{i_t}| + \dots \\ &\quad + (-1)^{n-1} |A_1 \cap \dots \cap A_n|. \end{aligned}$$

Hoán vị, tổ hợp, chỉnh hợp

Xét một tập hợp n phần tử

$$A = \{a_1, a_2, \dots, a_n\}.$$

Một cách xếp n phần tử này theo thứ tự là một hoán vị của tập hợp đó. Tập hợp có n phần tử thì số hoán vị là $n!$.

Chứng minh

Xét vị trí đầu tiên, ta có thể xếp một trong n phần tử vào vị trí này.

Đối với vị trí thứ hai, vì ta đã xếp một phần tử vào vị trí đầu nên ta còn $n - 1$ phần tử có thể xếp vào vị trí thứ hai.

Tương tự, với vị trí thứ ba ta có $n - 2$ cách chọn.

Tiếp tục như vậy cho tới vị trí cuối cùng (vị trí thứ n) ta còn đúng 1 phần tử.

Do đó, theo quy tắc nhân, số cách xếp n phần tử theo thứ tự là

$$n \cdot (n - 1) \cdot (n - 2) \cdots 2 \cdot 1 = n!.$$

Ví dụ, với tập $A = \{1, 2, 3\}$ thì ta có các hoán vị là

$$\{1, 2, 3\}, \{1, 3, 2\}, \{2, 1, 3\}, \{2, 3, 1\}, \{3, 1, 2\}, \{3, 2, 1\}.$$

Chỉnh hợp là một trường hợp nhỏ hơn của hoán vị. Khi đó từ một tập hợp có n phần tử, ta lấy ra k phần tử và sắp k phần tử đó theo thứ tự. Khi đó với $k \leq n$ thì số chỉnh hợp là $\frac{n!}{(n - k)!}$.

Chứng minh

Vị trí đầu tiên ta có n cách chọn.

Vị trí thứ hai ta có $n - 1$ cách chọn.

Tương tự vậy, ta thấy rằng ở vị trí thứ i thì ta có $n - i + 1$ cách chọn (chỉ số của vị trí và số cách chọn luôn có tổng bằng $n + 1$).

Do đó, tới vị trí thứ k thì số cách chọn là $n - k + 1$.

Như vậy theo quy tắc nhân, số chỉnh hợp là

$$\begin{aligned} & n \cdot (n - 1) \cdots (n - k + 1) \\ &= \frac{n \cdot (n - 1) \cdots (n - k + 1) \cdot (n - k) \cdot (n - k - 1) \cdots 2 \cdot 1}{(n - k) \cdot (n - k - 1) \cdots 2 \cdot 1} \\ &= \frac{n!}{(n - k)!}. \end{aligned}$$

Khi chúng ta lấy ra k phần tử từ n phần tử nhưng không sắp chúng theo thứ tự, ta có tổ hợp. Do đó ta cần chia cho số hoán vị của k phần tử. Như vậy số tổ hợp k phần tử từ tập hợp n phần tử là $\frac{n!}{(n - k)! \cdot k!}$.

2.8.5 Công thức truy hồi

Dãy số là một ánh xạ từ \mathbb{N} tới \mathbb{R}

$$u : \mathbb{N} \rightarrow \mathbb{R}, \quad n \mapsto u(n).$$

Khi đó các giá trị $u(1), u(2), \dots$ được gọi là **số hạng** của dãy số. Chúng ta cũng có thể viết u_1, u_2, \dots thay vì $u(1), u(2), \dots$, hoặc thậm chí là $\{u_n\}$.

Cấp số cộng và công thức truy hồi bậc nhất

Cho dãy số $\{u_n\}$ với phần tử đầu u_0 và công sai d . Khi đó các phần tử sau đó được tính với công thức

$$u_n = u_{n-1} + d$$

với mọi $n \geq 1$ thì $\{u_n\}$ được gọi là **cấp số cộng**.

Như vậy cấp số cộng là một dãy số có dạng

$$u_n = \varphi(n, u_{n-1}).$$

Dãy số có dạng như trên gọi là **truy hồi bậc nhất** (hay **first order**).

Example 18

Dãy số $\{u_n\}$ xác định bởi

$$u_n = au_{n-1} + bn + c$$

với u_0 là số hạng đầu; a, b và c là các số thực.

Example 19

Dãy số $\{u_n\}$ xác định bởi

$$u_n = au_{n-1}^2 + bu_{n-1} + c$$

với u_0 là số hạng đầu; a, b và c là các số thực.

Ta cần chú ý rằng việc nói truy hồi bậc nhất mang nghĩa là số hạng thứ n chỉ phụ thuộc vào số hạng thứ $n-1$, chứ không phải các số hạng trước đó như số hạng thứ $n-2, n-3$, vân vân và mây mây.

Ở ví dụ thứ hai thì số hạng thứ n là hàm bậc hai theo số hạng thứ $n-1$ nhưng đây là công thức truy hồi bậc nhất.

Dãy số Fibonacci và công thức truy hồi bậc hai

Đây có lẽ là dãy số nổi tiếng nhất trong toán học. Công thức của dãy số là với hai phần tử ban đầu $u_0 = 0$ và $u_1 = 1$, các phần tử sau sẽ được tính với công thức

$$u_n = u_{n-1} + u_{n-2}$$

với mọi $n \geq 2$.

Ở đây, số hạng thứ n được tính bởi hai số hạng trước nó nên đây là ví dụ của **truy hồi bậc hai**. Dãy số truy hồi bậc hai là dãy số có dạng

$$u_n = \varphi(n, u_{n-1}, u_{n-2}).$$

Example 20

Dãy số $\{u_n\}$ được xác định bởi

$$u_n = au_{n-1} + bu_{n-2} + c$$

với u_0 và u_1 là số hạng đầu; a, b và c là các số thực.

Công thức truy hồi bậc cao

Tổng quát, nếu số hạng thứ n của dãy số $\{u_n\}$ được tính bởi k số hạng trước nó, nghĩa là

$$u_n = \varphi(n, u_{n-1}, u_{n-2}, \dots, u_{n-k})$$

thì ta gọi là **công thức truy hồi bậc k** .

Các dãy truy hồi có ứng dụng rộng rãi trong toán học và các ngành khoa học khác, tiêu biểu nhất là dãy Fibonacci ở trên. Trong khoa học máy tính, công thức truy hồi có một ứng dụng để sinh một dãy số cho nhiều mục đích khác nhau như sinh số giả ngẫu nhiên (pseudo-random), sinh khóa cho thuật toán mã hóa dòng (stream cipher). Các ứng dụng này thường sử dụng dãy truy hồi **tuyến tính**, tức là bậc của các hạng tử $u_{n-1}, u_{n-2}, \dots, u_{n-k}$ không quá 1. Nói cách khác thì

$$u_n = a_{n-1}u_{n-1} + a_{n-2}u_{n-2} + \dots + a_{n-k}u_{n-k} + \phi(n),$$

trong đó $\phi(n)$ là một hàm số nào đó không phụ thuộc các hạng tử u_{n-1}, \dots, u_{n-k} . Các hệ số a_{n-1}, \dots, a_{n-k} nằm trong trường số nào đó tùy thuộc bài toán. Ví dụ với các dãy LFSR (Linear Feedback Shift Register) để sinh dãy trong tin học ở trên thì các phép tính được thực hiện trên trường \mathbb{F}_2 . Hiện tại chúng ta sẽ khảo sát dãy số với hệ số thực.

Chúng ta quan tâm đến công thức tổng quát của dãy số truy hồi. Để thấy rằng, để tính số hạng thứ n theo truy hồi thì ta phải biết đủ k số hạng trước đó. Nhưng với mỗi số hạng trong k số hạng đó ta lại cần đi ngược về trước đó nữa cho đến khi tới các số hạng ban đầu u_0, u_1, \dots, u_{k-1} . Tuy nhiên đôi khi chúng ta quan tâm một số tính chất đại số mà cần công thức tổng quát cho $\{u_n\}$ và chỉ phụ thuộc n , nghĩa là $u_n = f(n)$. Khi đó chúng ta sẽ sử dụng **phương pháp hàm sinh** (hay generating function method, hay метод проводящих функций).

Phương pháp hàm sinh

Phương pháp hàm sinh thực hiện theo các bước sau:

1. Xác định các số hạng ban đầu u_0, u_1, \dots, u_{k-1} .
2. Gọi $G(x)$ là đa thức với hệ số là các số hạng của dãy số, tức là

$$G(x) = u_0 + u_1x + u_2x^2 + \dots + u_nx^n + \dots$$

3. Phân tích $G(x)$ thành tổng các phân thức dạng $\frac{1}{H(x)}$. Sau đó ta gom các số hạng có cùng lũy thừa của x lại để được công thức tổng quát của $\{u_n\}$. Các hàm $H(x)$ sẽ được trình bày ở phần cuối với các ví dụ, và ta gọi chúng là **hàm sinh**.

Một ví dụ hay dùng của hàm sinh là khai triển sau.

$$\frac{1}{1-mx} = 1 + mx + m^2x^2 + \cdots + m^n x^n + \cdots \quad (2.15)$$

Công thức trên có thể thu được từ khai triển Taylor-Maclaurin hoặc bằng quy nạp.

Ví dụ 1

Xét dãy số $\{u_n\}$ với $u_0 = 0, u_1 = 1$, và

$$u_n = 5u_{n-1} - 6u_{n-2} \text{ với mọi } n \geq 2.$$

Đặt

$$G(x) = u_0 + u_1 x + u_2 x^2 + \cdots + u_n x^n + \cdots \quad (2.16)$$

Khi đó, ta nhân $G(x)$ với x, x^2 và nhân thêm số hạng để khử các hạng tử theo công thức truy hồi

$$u_n - 5u_{n-1} + 6u_{n-2} = 0.$$

Cụ thể, ta tính

$$x \cdot G(x) = u_0 x + u_1 x^2 + u_2 x^3 + \cdots + u_{n-1} x^n + \cdots,$$

và

$$x^2 \cdot G(x) = u_0 x^2 + u_1 x^3 + u_2 x^4 + \cdots + u_{n-2} x^n + \cdots$$

Khi đó

$$\begin{aligned} G(x) - 5x \cdot G(x) + 6x^2 \cdot G(x) &= u_0 + u_1 x + \color{blue}{u_2 x^2} + \color{green}{u_3 x^3} + \cdots + \color{red}{u_n x^n} + \cdots \\ &\quad - 5u_0 x - \color{blue}{5u_1 x^2} - \color{green}{5u_2 x^3} - \cdots - \color{red}{5u_{n-1} x^n} + \cdots \\ &\quad + \color{blue}{6u_0 x^2} + \color{green}{6u_1 x^3} + \cdots + \color{red}{6u_{n-2} x^n} + \cdots \end{aligned}$$

Các bạn có thấy điều gì không? Các số hạng trước x^2, x^3, \dots đều bằng 0 theo công thức truy hồi. Như vậy thu gọn về trái và thay u_0, u_1 vào về phải ta có

$$(1 - 5x + 6x^2) \cdot G(x) = u_0 + u_1 x - 5u_0 x = x,$$

tương đương với

$$G(x) = \frac{x}{1 - 5x + 6x^2}.$$

Vì $1 - 5x + 6x^2$ phân tích thành nhân tử là $(1 - 2x)(1 - 3x)$ nên ta muốn phân tích $G(x)$ thành tổng

$$G(x) = \frac{\alpha}{1 - 2x} + \frac{\beta}{1 - 3x} = \frac{(\alpha + \beta) + (-3\alpha - 2\beta)x}{(1 - 2x)(1 - 3x)},$$

với α và β là hệ số cần tìm để

$$(\alpha + \beta) + (-3\alpha - 2\beta)x \equiv x.$$

Như vậy, đồng nhất hệ số ta có

$$\alpha + \beta = 0, \quad -3\alpha - 2\beta = 1,$$

giải hệ ta có $\alpha = -1$ và $\beta = 1$. Ta thu được

$$G(x) = \frac{-1}{1-2x} + \frac{1}{1-3x}.$$

Theo công thức (2.15), ta có

$$\frac{1}{1-2x} = 1 + 2x + 2^2x^2 + \cdots + 2^n x^n + \cdots,$$

và

$$\frac{1}{1-3x} = 1 + 3x + 3^2x^2 + \cdots + 3^n x^n + \cdots.$$

Thay hai khai triển trên vào $G(x)$ ta có

$$\begin{aligned} G(x) &= -(1 + 2x + 2^2x^2 + \cdots + 2^n x^n + \cdots) \\ &\quad + (1 + 3x + 3^2x^2 + \cdots + 3^n x^n + \cdots) \\ &= 0 + (3-2)x + (3^2-2^2)x^2 + \cdots + (3^n-2^n)x^n + \cdots \end{aligned}$$

Đồng nhất hệ số với (2.16) ta có

$$u_n = 3^n - 2^n.$$

Đây chính là công thức tổng quát của dãy $\{u_n\}$.

Tất nhiên là ví dụ trên có thể được giải bằng cách khác là *đa thức đặc trưng* và đây là phương pháp phổ biến ở phổ thông. Tuy nhiên một số bài toán khác không thể sử dụng đa thức đặc trưng. Trước khi đến với các bài toán như vậy thì mình sẽ liệt kê một số hàm sinh thông dụng để giải quyết các bài toán đó.

Một số hàm sinh thông dụng

$$\boxed{\frac{1}{1-mx} = 1 + mx + m^2x^2 + \cdots + m^n x^n + \cdots}$$

$$\boxed{\frac{1}{(1-x)^t} = \sum_{i=0}^{\infty} C_{t+i-1}^i x^i.}$$

Khi $t = 2$ thì $C_{i+1}^i = \frac{(i+1)!}{i! \cdot 1!} = i+1$ nên ta có kết quả

$$\boxed{\frac{1}{(1-x)^2} = \sum_{i=0}^{\infty} (i+1)x^i.}$$

Kết hợp hai công thức trên

$$\boxed{\frac{1}{(1-mx)^t} = \sum_{i=0}^{\infty} C_{t+i-1}^i m^i x^i.}$$

Ví dụ II

Cho dãy số $\{u_n\}$ xác định bởi $u_0 = 1$, $u_1 = 2$ và

$$u_n = 6u_{n-1} - 8u_{n-2} + n, \text{ với mọi } n \geq 2.$$

Tương tự, đầu tiên ta đặt $G(x)$ là đa thức có hệ số là dãy $\{u_n\}$, nghĩa là

$$G(x) = u_0 + u_1x + u_2x^2 + \cdots + u_nx^n + \cdots$$

Ta cũng nhân x và x^2 cho $G(x)$ và thu được

$$x \cdot G(x) = u_0x + u_1x^2 + u_2x^3 + \cdots + u_{n-1}x^n + \cdots,$$

và

$$x^2 \cdot G(x) = u_0x^2 + u_1x^3 + u_2x^4 + \cdots + u_{n-2}x^n + \cdots$$

Như vậy, hoàn toàn tương tự ví dụ I, mình tính được

$$\begin{aligned} G(x) - 6x \cdot G(x) + 8x^2 \cdot G(x) &= u_0 \\ &\quad + (u_1 - 6u_0)x \\ &\quad + (u_2 - 6u_1 + 8u_0)x^2 \\ &\quad + (u_3 - 6u_2 + 8u_1)x^3 \\ &\quad + \cdots \\ &\quad + (u_n - 6u_{n-1} + 8u_{n-2})x^n \\ &\quad + \cdots \end{aligned}$$

Cơ mà ở đây $u_n - 6u_{n-1} + 8u_{n-2}$ không đủ để triệt tiêu thành 0 mà cần thêm n nữa. Vậy phải làm sao?

Chúng ta sẽ cần một hàm $F(x)$ sao cho

$$\begin{aligned} G(x) - 6x \cdot G(x) + 8x^2 \cdot G(x) + F(x) &= u_0 \\ &\quad + (u_1 - 6u_0)x \\ &\quad + (u_2 - 6u_1 + 8u_0 - 2)x^2 \\ &\quad + (u_3 - 6u_2 + 8u_1 - 3)x^3 \\ &\quad + \cdots \\ &\quad + (u_n - 6u_{n-1} + 8u_{n-2} - n)x^n \\ &\quad + \cdots \end{aligned}$$

Như vậy mình gom các hệ số màu đỏ lại sẽ được

$$F(x) = -2x^2 - 3x^3 + \cdots - nx^n + \cdots$$

Khi đó

$$G(x) - 6x \cdot G(x) + 8x^2 \cdot G(x) + F(x) = u_0 + (u_1 - 6u_0)x,$$

và việc của chúng ta là tìm một hàm sinh biểu thị cho $F(x)$ nữa để có thể biểu diễn $G(x)$ như ở ví dụ I.

Ta có

$$\begin{aligned}
 F(x) &= -2x^2 - 3x^3 - \cdots - nx^n + \cdots \\
 &= 1x - x(1 + 2x + 3x^2 + \cdots + nx^{n-1} + \cdots) \\
 &= x - x(1 + x + x^2 + x^3 + \cdots + x^n + \cdots)' \\
 &= x - x \cdot \left(\frac{1}{1-x} \right)' \\
 &= x - x \cdot \frac{1}{(1-x)^2} \\
 &= \frac{x(1-2x+x^2)+x}{(1-x)^2} = -\frac{x^2(2-x)}{(1-x)^2}.
 \end{aligned}$$

Thay u_0, u_1 và $F(x)$ vào ta tính được $G(x)$ là hàm

$$(1 - 6x + 8x^2) \cdot G(x) - \frac{x^2(2-x)}{(1-x)^2} = 1 - 4x,$$

tương đương với

$$\begin{aligned}
 G(x) &= \frac{1}{1-6x+8x^2} \left[1 - 4x + \frac{x^2(2-x)}{(1-x)^2} \right] \\
 &= \frac{(1-4x)(1-2x+x^2)-2x^2+x^3}{(1-6x+8x^2)(1-x)^2} \\
 &= \frac{1-2x+x^2-4x+8x^2-4x^3+2x^2-x^3}{(1-6x+8x^2)(1-x)^2} \\
 &= \frac{1-6x+11x^2-5x^3}{(1-6x+8x^2)(1-x)^2}.
 \end{aligned}$$

Bây giờ, $1 - 6x + 8x^2$ phân tích thành nhân tử $(1 - 2x)(1 - 4x)$. Vậy mình sẽ cần tách $G(x)$ thành

$$G(x) = \frac{A}{1-2x} + \frac{B}{1-4x} + \frac{C}{1-x} + \frac{D}{(1-x)^2}$$

với A, B, C và D là các hệ số cần tìm. Quy đồng mẫu số mình có

$$G(x) = \frac{A(1-4x)(1-x)^2 + B(1-2x)(1-x)^2 + C(1-2x)(1-4x)(1-x) + D(1-2x)(1-4x)}{(1-6x+8x^2)(1-x)^2}.$$

Thu gọn tử số lại mình được

$$(A + B + C + D) + (-6A - 4B - 7C - 6D)x + (9A + 5B + 14C + 8D)x^2 + (-4A - 2B - 8C)x^3,$$

và đồng nhất hệ số thì mình cần giải hệ phương trình

$$\begin{cases} A + B + C + D = 1 \\ -6A - 4B - 7C - 6D = -6 \\ 9A + 5B + 14C + 8D = 11 \\ -4A - 2B - 8C = -5 \end{cases} \iff \begin{cases} A = -1/2 \\ B = 7/18 \\ C = 7/9 \\ D = 1/3 \end{cases}$$

Bây giờ thay các hàm sinh vào $G(x)$ (chưa cần thay A, B, C và D vội) thì mình có

$$\begin{aligned}
 G(x) &= A \cdot (1 + 2x + 2^2x^2 + \cdots + 2^n x^n + \cdots) \\
 &\quad + B \cdot (1 + 4x + 4^2x^2 + \cdots + 4^n x^n + \cdots) \\
 &\quad + C \cdot (1 + x + x^2 + \cdots + x^n + \cdots) \\
 &\quad + D \cdot (1 + 2x + 3x + \cdots + (n+1)x^n + \cdots).
 \end{aligned}$$

Nhìn vào hệ số của x^n mình có công thức tổng quát

$$u_n = A \cdot 2^n + B \cdot 4^n + C + D \cdot (n+1) = -2^{n-1} + \frac{7 \cdot 4^n}{18} + \frac{7}{9} + \frac{n+1}{3}.$$

Ví dụ III

Trong một số trường hợp "hơi lú", ví dụ như

$$u_n = 4u_{n-1} - 4u_{n-2}$$

thì nếu sử dụng đa thức đặc trưng ta có một phương trình bậc hai với nghiệm kép. Khi đó công thức tổng quát không còn ở dạng

$$u_n = \alpha \cdot z_1^n + \beta \cdot z_2^n$$

với z_1 và z_2 là hai nghiệm phân biệt của phương trình đặc trưng, mà ở một dạng khác. Phương pháp hàm sinh sẽ giúp chúng ta tìm công thức tổng quát ở trường hợp này.

Xét đa thức $G(x)$ với hệ số là các số hạng của dãy $\{u_n\}$ cho bởi công thức truy hồi ở trên:

$$G(x) = u_0 + u_1x + u_2x^2 + \cdots + u_nx^n + \cdots$$

Thực hiện tương tự bên trên, mình tính

$$x \cdot G(x) = u_0x + u_1x^2 + u_2x^3 + \cdots + u_{n-1}x^n + \cdots$$

và

$$x^2 \cdot G(x) = u_0x^2 + u_1x^3 + u_2x^4 + \cdots + u_{n-2}x^n + \cdots$$

Như vậy

$$\begin{aligned} G(x) - 4x \cdot G(x) + 4x^2 \cdot G(x) &= u_0 \\ &\quad + (u_1 - 4u_0)x \\ &\quad + \overbrace{(u_2 - 4u_1 + 4u_0)}^0 x^2 \\ &\quad + \cdots \\ &\quad + \overbrace{(u_n - 4u_{n-1} + 4u_{n-2})}^0 x^n \\ &\quad + \cdots \\ &= u_0 + (u_1 - 4u_0)x. \end{aligned}$$

Đặt $G(x)$ làm nhân tử chung và chuyển về mình có

$$G(x) = \frac{u_0 + (u_1 - 4u_0)x}{(1 - 2x)^2}.$$

Mình cần tách $G(x)$ thành tổng

$$G(x) = \frac{A}{1 - 2x} + \frac{B}{(1 - 2x)^2}.$$

Quy đồng mẫu số và đồng nhất hệ số mình cần tìm A và B thỏa mãn hệ phương trình

$$\begin{cases} A + B = u_0 \\ -2A = (u_1 - 4u_0) \end{cases} \iff \begin{cases} A = (-u_1 + 4u_0)/2 \\ B = (u_1 - 2u_0)/2 \end{cases}$$

Thay hàm sinh vào $G(x)$ mình được

$$\begin{aligned} G(x) &= A \cdot (1 + 2x + 2^2x^2 + \cdots + 2^n x^n + \cdots) \\ &\quad + B \cdot [1 + 2 \cdot 2x + 3 \cdot 2^2x^2 + \cdots + (n+1) \cdot 2^n x^n]. \end{aligned}$$

Hệ số trước x^n cho ta công thức tổng quát của dãy số

$$u_n = A \cdot 2^n + B \cdot (n+1) \cdot 2^n = 2^{n-1} \cdot [2u_0 + (u_1 - 2u_0) \cdot n].$$

2.8.6 Lý thuyết đồ thị

Phần này minh sử dụng các quyển sách dành cho học sinh chuyên Tin [12].

Các định nghĩa cơ bản

Definition 27 (Đồ thị)

Đồ thị (hay **graph**, hay **rpađ**) $G = (V, E)$ gồm một tập hợp các đỉnh V và tập hợp các cạnh E nối các đỉnh với nhau.

Đồ thị vô hướng

Example 21

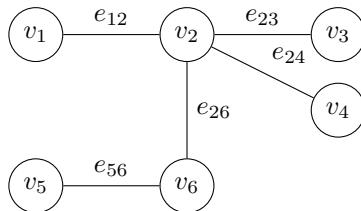
Đồ thị sau có:

1. Tập hợp các đỉnh

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}.$$

2. Tập hợp các cạnh

$$E = \{e_{12}, e_{23}, e_{24}, e_{26}, e_{56}\}.$$



Hình 2.50: Đồ thị vô hướng

Ở ví dụ trên, cạnh e_{ij} nối đỉnh v_i và đỉnh v_j . Trong trường hợp này hướng của cạnh không quan trọng nên việc viết e_{ij} và e_{ji} là tương đương. Đồ thị lúc này gọi là **đồ thị vô hướng** (hay **undirected graph**).

Hai đỉnh được gọi là **kề nhau** (hay **adjacent**) nếu có cạnh nối giữa chúng.

Ở ví dụ trên thì hai đỉnh v_1 và v_2 kề nhau, nhưng đỉnh v_1 không kề với v_3 vì không có cạnh e_{13} .

Khi đó cạnh nối hai đỉnh kề nhau được gọi là **cạnh liên thuộc** (hay **incident**).

Ở đồ thị vô hướng, ta nói **bậc** (hay **degree**) của đỉnh v là số cạnh liên thuộc với đỉnh v , và kí hiệu là $\deg(v)$.

Ở [Hình 2.50](#) ta thấy $\deg(v_1) = 1$, $\deg(v_2) = 3$, $\deg(v_3) = 1$, $\deg(v_4) = 1$, $\deg(v_5) = 1$, $\deg(v_6) = 2$. Tổng quát ta có định lí sau.

Theorem 8

Giả sử $G = (V, E)$ là đồ thị vô hướng, khi đó tổng tất cả các bậc của các đỉnh trong V sẽ bằng hai lần

số cạnh:

$$\sum_{v \in V} \deg(v) = 2|E|.$$

❶ **Chứng minh**

Khi lấy tổng tất cả các bậc thì mỗi cạnh e_{ij} sẽ được tính một lần cho đỉnh v_i và một lần cho đỉnh v_j . Từ đó suy ra kết quả.

❶ **Corollary 2**

Trong đồ thị vô hướng thì số đỉnh có bậc lẻ là số chẵn.

❶ **Chứng minh**

Giả sử V_1 là tập các đỉnh có bậc lẻ và V_2 là tập các đỉnh có bậc chẵn. Khi đó $V_1 \cup V_2 = V$ và $V_1 \cap V_2 = \emptyset$. Theo định lí trên thì

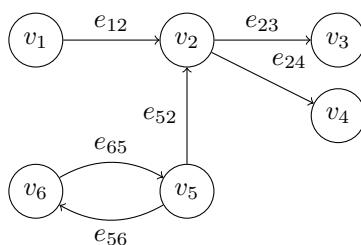
$$\sum_{v_1 \in V_1} \deg(v_1) + \sum_{v_2 \in V_2} \deg(v_2) = 2|E|$$

là số chẵn, mà tổng bậc của các đỉnh bậc chẵn $\sum_{v_2 \in V_2} \deg(v_2)$ cũng là số chẵn nên suy ra tổng $\sum_{v_1 \in V_1} \deg(v_1)$ cũng là chẵn.

Do mỗi giá trị $\deg(v_1)$ là lẻ với mọi $v_1 \in V_1$ nên tổng của chúng là chẵn khi và chỉ khi số phần tử của V_1 là chẵn. Ta có điều phải chứng minh.

Đồ thị có hướng

Nếu chúng ta quan tâm đến hướng thì khi vẽ cạnh e_{ij} ta vẽ mũi tên từ đỉnh v_i tới đỉnh v_j . Đồ thị khi đó gọi là **đồ thị có hướng** (hay directed graph).



Hình 2.51: Đồ thị có hướng

Ở hình trên:

- chỉ có cạnh từ v_1 tới v_2 là e_{12} chứ không có cạnh từ v_2 tới v_1
- có cạnh từ v_6 tới v_5 là e_{65} và cũng có cạnh từ v_5 tới v_6 là e_{56} .

Lúc này tập đỉnh là

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\},$$

và tập cạnh là

$$E = \{e_{12}, e_{23}, e_{24}, e_{52}, e_{56}, e_{65}\}.$$

Đối với đồ thị có hướng thì cạnh e_{ij} là cạnh đi ra từ đỉnh i và đi vào đỉnh j . Đỉnh i gọi là đỉnh đầu, đỉnh j gọi là đỉnh cuối.

❶ Definition 28 (Bán bậc vào, bán bậc ra)

Bán bậc ra (hay out-degree) của đỉnh v là số lượng cạnh đi ra khỏi nó và kí hiệu là $\deg^+(v)$.

Bán bậc vào (hay in-degree) của đỉnh v là số lượng cạnh đi vào nó và kí hiệu là $\deg^-(v)$.

Với ví dụ ở Hình 2.51 thì:

- $\deg^+(v_1) = 1, \deg^-(v_1) = 0;$
- $\deg^+(v_2) = 2, \deg^-(v_2) = 1;$
- $\deg^+(v_3) = 0, \deg^-(v_3) = 1;$
- $\deg^+(v_4) = 0, \deg^-(v_4) = 1;$
- $\deg^+(v_5) = 2, \deg^-(v_5) = 1;$
- $\deg^+(v_6) = 1, \deg^-(v_6) = 1.$

❷ Theorem 9

Nếu $G = (V, E)$ là đồ thị có hướng thì tổng tất cả các bán bậc ra của các đỉnh bằng tổng tất cả các bán bậc vào, và cũng bằng tổng số cung của đồ thị

$$\sum_{v \in V} \deg^+(v) = \sum_{v \in V} \deg^-(v) = |E|.$$

❸ Chứng minh

Mỗi cạnh của đồ thị đi ra từ đúng một đỉnh nên $\sum_{v \in V} \deg^+(v) = |E|$.

Tương tự, mỗi cạnh của đồ thị đi vào đúng một đỉnh nên $\sum_{v \in V} \deg^-(v) = |E|$.

Kết hợp hai đẳng thức trên ta có điều phải chứng minh.

Đường đi và chu trình

Một dãy có thứ tự các đỉnh $v_{i_0}, v_{i_1}, \dots, v_{i_n}$ sao cho $e_{i_t i_{t+1}} \in E$ với mọi $t = 0, 1, \dots, n - 1$ được gọi là **đường đi**. Lúc này đường đi có $n + 1$ đỉnh $v_{i_0}, v_{i_1}, \dots, v_{i_n}$ và n cạnh $e_{i_0 i_1}, e_{i_1 i_2}, \dots, e_{i_{n-1} i_n}$.

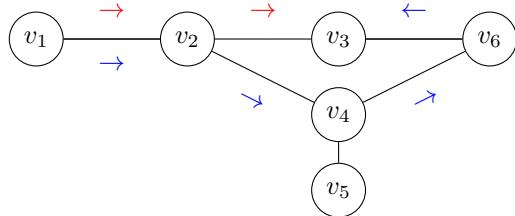
Nếu tồn tại một đường đi từ đỉnh v_{i_0} tới đỉnh v_{i_n} thì ta nói đỉnh v_{i_n} **đến được** (hay **reachable**) từ đỉnh v_{i_0} và kí hiệu là $v_{i_0} \rightsquigarrow v_{i_n}$.

1. Đỉnh v_{i_0} được gọi là đỉnh đầu của đường đi.
2. Đỉnh v_{i_n} được gọi là đỉnh cuối của đường đi.
3. Các đỉnh $v_{i_1}, \dots, v_{i_{n-1}}$ được gọi là đỉnh trong của đường đi.

Đường đi được gọi là đường đi **đơn** (hay **simple**) nếu tất cả các đỉnh trên đường đi hoàn toàn phân biệt.

Hình 2.52 thể hiện hai đường đi đơn từ đỉnh v_1 tới đỉnh v_3 :

- đường đi thứ nhất là v_1 , tới v_2 và tới v_3
- đường đi thứ hai là v_1 , tới v_2 , đi xuống v_4 , đi lên v_6 và quay lại v_3 .



Hình 2.52: Đường đi từ v_1 tới v_3

Đường đi được gọi là **chu trình** (hay **circuit**) nếu đỉnh đầu trùng với đỉnh cuối, nghĩa là $v_{i_0} = v_{i_n}$.

Đẳng cấu đồ thị

Definition 29

Hai đồ thị $G = (V, E)$ và $G' = (V', E')$ được gọi là **đẳng cấu** (hay **isomorphic**) nếu tồn tại một song ánh $f : V \rightarrow V'$ sao cho số cung nối đỉnh u với đỉnh v trên E bằng số cung nối đỉnh $f(u)$ với đỉnh $f(v)$ trên E' .

Nói cách khác, kí hiệu e_{ij} là cạnh nối đỉnh v_i và v_j trên E . Đặt $v'_i = f(v_i)$ và $v'_j = f(v_j)$ là các đỉnh thuộc V' . Khi đó e'_{ij} là cạnh thuộc E' nối đỉnh v'_i và đỉnh v'_j . Nếu giữa v_i và v_j không có cạnh nào thì cũng không có cạnh nối v'_i và v'_j .

Bài toán đẳng cấu đồ thị (graph isomorphism problem) là một trong những bài toán khó hiện nay (tháng 1 năm 2025) về việc tìm lời giải trong thời gian đa thức. Nếu cho trước hai đồ thị thì nhiệm vụ của bài toán là xác định xem hai đồ thị có đẳng cấu hay không. Rõ ràng nếu số đỉnh nhỏ thì chúng ta có thể thử từng hoán vị (song ánh) của tập đỉnh, nhưng khi số đỉnh lớn thì việc thử sai tất cả hoán vị là không thể.

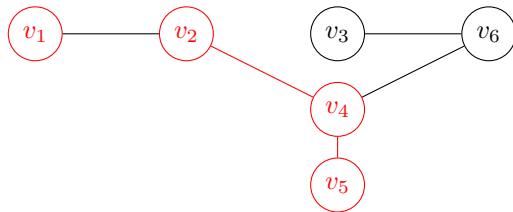
Đồ thị con

Definition 30 (Đồ thị con)

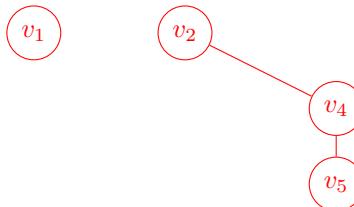
Đồ thị $G' = (V', E')$ là **đồ thị con** (hay **subgraph**, hay **подграф**) của đồ thị $G = (V, E)$ nếu $V' \subseteq V$ và $E' \subseteq E$.

Nói cách khác, đồ thị con thu được từ đồ thị ban đầu bằng việc lấy một lượng nhất định đỉnh và cạnh.

Ở đồ thị trên Hình 2.53, mình lấy các đỉnh v_1, v_2, v_4 và v_5 , và lấy các cạnh e_{24}, e_{45} thì mình có đồ thị con trên Hình 2.54. Các bạn có thể thấy ở đồ thị ban đầu thì v_1 nối với v_2 , nhưng ở đồ thị con thì không có cạnh nối giữa hai đỉnh này.



Hình 2.53: Đồ thị ban đầu

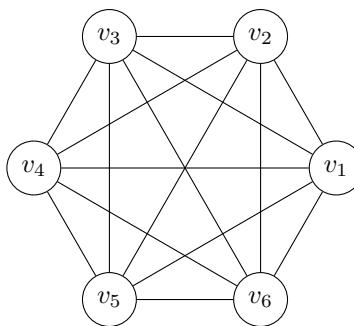


Hình 2.54: Đồ thị con

Đồ thị đầy đủ

Đồ thị vô hướng được gọi là **đầy đủ** (hay **complete**, hay **полный**) nếu mọi cặp đỉnh đều kề nhau.

Đồ thị đầy đủ gồm n đỉnh kí hiệu là K_n .

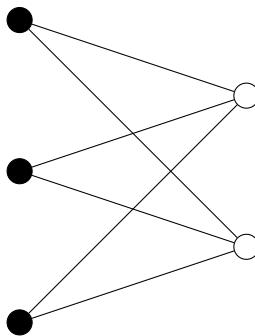
Hình 2.55: Ví dụ K_6

Dễ thấy số cạnh của đồ thị đầy đủ n đỉnh là C_n^2 .

Đồ thị hai phía

Đồ thị vô hướng được gọi là **hai phía** (hay **bipartite**) nếu tập đỉnh của nó có thể chia thành hai tập rời nhau V_1 và V_2 sao cho không tồn tại cạnh nối hai đỉnh của V_1 , và cũng không tồn tại cạnh nối hai đỉnh của V_2 .

Nếu $|V_1| = n_1$ và $|V_2| = n_2$ và giữa mọi cặp đỉnh (v_1, v_2) , trong đó $v_1 \in V_1$ và $v_2 \in V_2$, đều có cạnh nối thì đồ thị hai phía đó được gọi là đồ thị hai phía đầy đủ, kí hiệu là K_{n_1, n_2} .



Hình 2.56: Ví dụ đồ thị hai phía đầy đủ $K_{3,2}$

Theo quy tắc nhân, số cạnh của đồ thị hai phía đầy đủ là $v_1 \cdot v_2$ vì mỗi đỉnh ở V_1 đều có v_2 cạnh nối với nó, và có tất cả v_1 đỉnh trong V_1 .

Đồ thị phẳng

Đồ thị được gọi là **đồ thị phẳng** (hay **planar graph**, hay **планарный граф**) nếu chúng ta có thể vẽ đồ thị trên mặt phẳng sao cho:

1. Mỗi đỉnh tương ứng với một điểm trên mặt phẳng, không có hai đỉnh cùng tọa độ.
2. Mỗi cạnh tương ứng với một đường liên tục nối hai đỉnh và hai cạnh bất kì không giao nhau.

Phép vẽ đồ thị phẳng này được gọi là **biểu diễn phẳng** của đồ thị.

i Theorem 10 (Định lí Kuratowski)

Một đồ thị vô hướng là đồ thị phẳng khi và chỉ khi nó không chứa đồ thị con đẳng cấu với $K_{3,3}$ hoặc K_5 .

i Theorem 11 (Công thức Euler)

Nếu một đồ thị vô hướng liên thông là đồ thị phẳng và biểu diễn phẳng của đồ thị đó gồm v đỉnh và e cạnh chia mặt phẳng thành f phần thì

$$v - e + f = 2.$$

Công thức Euler này chúng ta cũng thấy ở hình học không gian đối với một khối đa diện. Nếu v là số đỉnh (vertices), e là số cạnh (edges) và f là số mặt (faces) thì $v - e + f = 2$.

1. Hình tứ diện có 4 đỉnh, 6 cạnh và 4 mặt nên $4 - 6 + 4 = 2$.
2. Hình lập phương có 8 đỉnh, 12 cạnh và 6 mặt nên $8 - 12 + 6 = 2$.

i Theorem 12

Nếu đồ thị vô hướng $G = (V, E)$ là đồ thị phẳng có ít nhất 3 đỉnh thì

$$|E| \leq 3|V| - 6.$$

Ngoài ra nếu G không có chu trình độ dài 3 thì

$$|E| \leq 2|V| - 4.$$

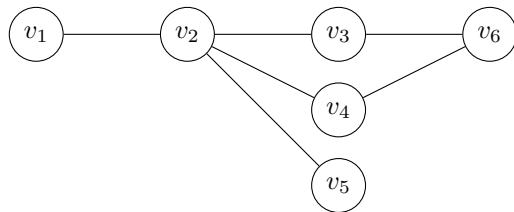
Tính liên thông của đồ thị

Tính liên thông trên đồ thị vô hướng

❶ Definition 31 (Đồ thị liên thông)

Một đồ thị vô hướng được gọi là **liên thông** (hay **connected**, hay **связанный**) nếu giữa hai đỉnh bất kỳ của đồ thị tồn tại đường đi.

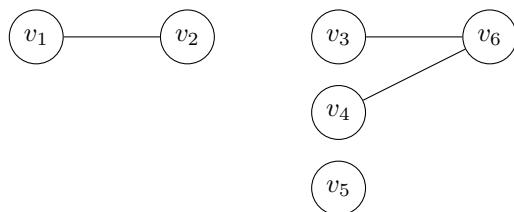
Đồ thị chỉ gồm một đỉnh duy nhất cũng được coi là đồ thị liên thông.



Hình 2.57: Đồ thị liên thông

❶ Definition 32 (Thành phần liên thông)

Cho đồ thị $G = (V, E)$. Nếu đồ thị con $G' = (V', E')$ của G là đồ thị liên thông thì G' được gọi là **thành phần liên thông** (hay **connected component**, **связанный компонент**) của đồ thị G .



Hình 2.58: Các thành phần liên thông trên đồ thị

Trên Hình 2.58 có ba thành phần liên thông:

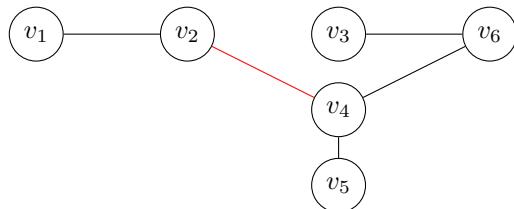
- thành phần liên thông thứ nhất gồm hai đỉnh v_1, v_2 , và cạnh e_{12}
- thành phần liên thông thứ hai gồm ba đỉnh v_3, v_4, v_6 , và các cạnh e_{36}, e_{46}
- thành phần liên thông thứ ba chỉ gồm đỉnh v_5 .

Đôi khi việc bỏ đi một đỉnh và tất cả cạnh liên thuộc với nó sẽ tăng số lượng thành phần liên thông hơn so với đồ thị ban đầu. Các đỉnh như vậy gọi là **đỉnh cắt** (hay **cut vertices**), hoặc **nút khớp** (hay **articulation nodes**).

Tương tự, khi bỏ đi một cạnh mà số lượng thành phần liên thông tăng lên thì cạnh đó gọi là **cạnh cắt** (hay **cut edges**) hoặc **cầu** (hay **bridge**).

Ở **Hình 2.59** là một đồ thị liên thông, nếu chúng ta xóa cạnh e_{24} (cạnh màu đỏ) thì chúng ta sẽ có hai thành phần liên thông:

- thành phần liên thông thứ nhất gồm hai đỉnh v_1 và v_2
- thành phần liên thông thứ hai gồm bốn đỉnh v_3, v_4, v_5 và v_6 .



Hình 2.59: Ví dụ về cầu

Đồ thị có thể có nhiều cầu. Ở **Hình 2.59**, thay vì xóa cạnh e_{24} , nếu ta xóa cạnh e_{46} thì cũng làm tăng số thành phần liên thông. Do đó cạnh e_{46} cũng là cầu. Tương tự cho cạnh e_{12}, \dots

Tính liên thông trên đồ thị có hướng

Một đồ thị có hướng được gọi là:

- **liên thông mạnh** (hay **strongly connected**) nếu tồn tại đường đi giữa hai đỉnh bất kì của đồ thị;
- **liên thông yếu** (hay **weakly connected**) nếu phiên bản vô hướng của nó là đồ thị liên thông.

Bài toán xác định các thành phần liên thông

Bài toán xác định các thành phần liên thông của đồ thị là một bài toán quan trọng trong lý thuyết đồ thị. Bài toán sẽ tìm tất cả thành phần liên thông của đồ thị vô hướng.

Để liệt kê các thành phần liên thông của đồ thị vô hướng $G = (V, E)$ ta thực hiện các bước sau:

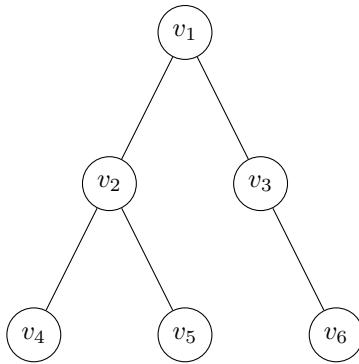
1. Bắt đầu từ một đỉnh bất kì, ta tìm tất cả đỉnh đến được từ đỉnh đó. Như vậy chúng ta tìm được một thành phần liên thông.
2. Loại những đỉnh ở thành phần liên thông đầu tiên và xét một trong những đỉnh còn lại. Lặp lại công việc ở bước 1, ta tìm tất cả đỉnh đến được từ đỉnh đang xét. Như vậy chúng ta tìm được thêm một thành phần liên thông.
3. Thực hiện đến khi đã xét hết tất cả đỉnh trong đồ thị.

Ở đây, để tìm tất cả đỉnh đến được từ một đỉnh nào đó trong đồ thị ta sử dụng thuật toán DFS (Depth First Search, Duyệt Sâu) hoặc BFS (Breadth First Search, Duyệt Rộng).

Cây

Definition 33 (Cây)

Cây (hay **tree**, hay **дерево**) là đồ thị vô hướng, liên thông, và không có chu trình đơn.



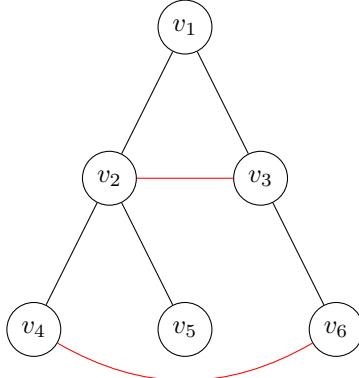
Hình 2.60: Ví dụ về cây

Definition 34 (Cây khung)

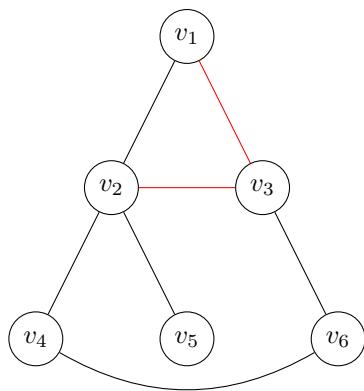
Xét đồ thị $G = (V, E)$ và $T = (V', E')$ là đồ thị con của G . Nếu T là cây thì ta gọi T là **cây khung** hoặc **cây bao trùm** (hay **spanning tree**) của đồ thị G .

Điều kiện cần và đủ để một đồ thị vô hướng có cây khung là đồ thị đó phải liên thông.

Một đồ thị có thể có nhiều cây khung. Ở [Hình 2.61](#) ta xóa đi cạnh e_{23} và e_{46} (hai cạnh màu đỏ) thì thu được một cây khung. Tương tự, ở [Hình 2.62](#) ta xóa đi hai cạnh màu đỏ là e_{13} và e_{23} thì cũng thu được một cây khung khác.



Hình 2.61: Cây khung thứ nhất



Hình 2.62: Cây khung thứ hai

i Theorem 13 (Daisy Chain Theorem)

Giả sử $T = (V, E)$ là đồ thị vô hướng với n đỉnh. Khi đó các mệnh đề sau tương đương:

1. T là cây.
2. T không chứa chu trình đơn và có $n - 1$ cạnh.
3. T liên thông và mỗi cạnh của nó đều là cầu.
4. Giữa hai đỉnh bất kì của G' đều tồn tại đúng một đường đi đơn.
5. T không chứa chu trình đơn nhưng nếu ta thêm vào một cạnh thì ta thu được chu trình đơn.
6. T liên thông và có $n - 1$ cạnh.

Các mệnh đề trên có thể xem như các định nghĩa tương đương của cây. Trong nhiều trường hợp thì tính chất của cây có thể thu được từ các định nghĩa này nên mình sẽ chứng minh chuỗi mệnh đề này.

❶ Chứng minh $1 \Rightarrow 2$

Từ T là cây, theo định nghĩa thì T không chứa chu trình đơn. Ta chứng minh cây T có n đỉnh thì sẽ có $n - 1$ cạnh bằng quy nạp.

Với $n = 1$ thì cây chỉ có thể có 0 cạnh (không nối đi đâu cả).

Giả thiết quy nạp: với $n \geq 1$ thì cây T với n đỉnh có $n - 1$ cạnh. Đặt các đỉnh là v_1, v_2, \dots, v_n .

Bây giờ ta thêm đỉnh v_{n+1} vào cây T để được cây mới T' . Ta cần chứng minh cây mới có n cạnh.

Do T liên thông nên giữa mọi cặp đỉnh u và v của T đều có đường đi, giả sử là

$$u = v_{i_1} \rightarrow v_{i_2} \rightarrow \cdots \rightarrow v_{i_k} = v.$$

Khi đó, nếu cả hai đỉnh u và v đều có cạnh nối với v_{n+1} thì ta thu được chu trình

$$v_{n+1} \rightarrow u = v_{i_1} \rightarrow v_{i_2} \rightarrow \cdots \rightarrow v_{i_k} = v \rightarrow v_{n+1},$$

như vậy sẽ không tạo thành cây. Nghĩa là từ v_{n+1} chỉ có thể nối với một trong các đỉnh v_1, \dots, v_n nên số cạnh chỉ có thể tăng lên 1 để có được cây mới. Khi đó, với $n + 1$ đỉnh ta có n cạnh, theo quy nạp ta có điều phải chứng minh.

❷ Chứng minh $2 \Rightarrow 3$

Giả sử T có k thành phần liên thông T_1, T_2, \dots, T_k . Vì T không chứa chu trình đơn nên các thành phần liên thông của T cũng không chứa chu trình đơn, tức là T_1, T_2, \dots, T_k đều là cây.

Gọi n_1, n_2, \dots, n_k lần lượt là số đỉnh của T_1, T_2, \dots, T_k thì

- $n_1 + \cdots + n_k = n$;
- cây T_1 có $n_1 - 1$ cạnh (từ chứng minh $1 \Rightarrow 2$), tương tự cây T_2 có $n_2 - 1$ cạnh, ..., T_k có $n_k - 1$ cạnh.

Cộng số lượng cạnh của k cây lại ta có tổng số cạnh của cây T , kết hợp giả thiết ban đầu ta có

$$n_1 + n_2 + \cdots + n_k - k = n - k = n - 1.$$

Như vậy $k = 1$, nghĩa là T liên thông. Từ đây, do T không có chu trình nên nếu bỏ một cạnh bất kì thì đồ thị mới cũng không có chu trình. Đồ thị mới không thể liên thông vì nếu giả sử ngược lại (phản chứng) đồ thị mới liên thông thì nó sẽ phải là cây có n đỉnh. Nhưng hiện tại đồ thị chỉ có $n - 2$ cạnh vì đã bỏ đi một cạnh, không phải $n - 1$. Do đó cạnh bỏ đi là cầu.

❸ Chứng minh $3 \Rightarrow 4$

Gọi u và v là hai đỉnh bất kì trong T . Vì T liên thông nên sẽ có một đường đi đơn từ u tới v là

$$u = v_{i_1} \rightarrow v_{i_2} \rightarrow \cdots \rightarrow v_{i_k} = v.$$

Giả sử tồn tại một đường đi đơn khác từ u tới v là

$$u = v_{j_1} \rightarrow v_{j_2} \rightarrow \cdots \rightarrow v_{j_l} = v.$$

Khi đó nếu ta xóa đi một cạnh ở đường đi đơn đầu nhưng không nằm trên đường đi đơn thứ hai thì u

vẫn tới được v nhờ vào đường đi đơn thứ hai. Nói cách khác u và v vẫn liên thông nên dữ kiện "mỗi cạnh đều là cầu" mâu thuẫn.

Nói rõ hơn, không mất tính tổng quát, giả sử đường đi đơn thứ nhất và thứ hai có cạnh chung là $v_{i_2} \rightarrow v_{i_3}$, nghĩa là $v_{i_2} = v_{j_2}$ và $v_{i_3} = v_{j_3}$. Khi đó nếu ta xóa cạnh $v_{i_2} \rightarrow v_{i_3}$ thì u vẫn tới được v và số lượng thành phần liên thông không tăng.

❶ **Chứng minh 4 \Rightarrow 5**

Cây T sẽ không chứa chu trình đơn vì khi đó sẽ có hai đường đi đơn từ hai đỉnh bất kì trên chu trình đó (chúng ta có thể tưởng tượng tượng đi cùng chiều và ngược chiều kim đồng hồ đều có thể đi từ 1 tới 3).

Khi đó, hai đỉnh bất kì u và v luôn liên thông, nghĩa là tồn tại đường đi

$$u = v_{i_1} \rightarrow v_{i_2} \rightarrow \cdots \rightarrow v_{i_k} = v.$$

Nếu ta thêm bất kì bắc kì cạnh nào nối giữa hai đỉnh trên đường đi, không mất tính tổng quát giả sử là v_{i_1} và v_{i_k} , khi đó ta có chu trình vì

$$u = v_{i_1} \rightarrow v_{i_2} \rightarrow \cdots \rightarrow v_{i_k} = v \rightarrow v_{i_1}.$$

Như vậy cứ thêm vào một cạnh ta sẽ có chu trình.

❶ **Chứng minh 5 \Rightarrow 6**

Giữa hai đỉnh bất kì u và v của T có hai trường hợp:

- u và t được nối bởi một cạnh;
- u và t không kề nhau (không có cạnh nối). Ở trường hợp này, nếu chúng ta vẽ cạnh nối u với v thì theo giả thiết sẽ tạo ra chu trình. Điều này chỉ xảy ra khi và chỉ khi có một đường đi đơn từ u tới v .

Như vậy trong cả hai trường hợp thì luôn tồn tại đường đi từ u tới v , nói cách khác là hai đỉnh liên thông. Điều này đúng với mọi đỉnh trong đồ thị nên T liên thông.

Do ta đã chứng minh được T liên thông, và giả thiết là T không chứa chu trình đơn, nên suy ra T là cây. Như vậy T có $n - 1$ cạnh.

❶ **Chứng minh 6 \Rightarrow 1**

Sử dụng phản chứng, giả sử T không là cây. Khi đó T không liên thông hoặc T có chu trình. Do giả thiết T liên thông nên ta chỉ xét trường hợp T có chu trình.

Nếu ta bỏ một cạnh trên chu trình này thì T vẫn liên thông. Nếu T vẫn còn chu trình thì ta lại bỏ đi một cạnh của chu trình đó. Tiếp tục cho đến khi T không còn chu trình nào. Lúc này T là cây nhưng lại có ít hơn $n - 1$ cạnh do chúng ta đã bỏ ít nhất một cạnh. Điều này vô lí vì ở chứng minh trên (1 \Rightarrow 2), nếu T là cây thì T có $n - 1$ cạnh. Như vậy theo phản chứng suy ra T là cây.

1 Theorem 14

Số cây khung của đồ thị đầy đủ K_n là n^{n-2} .

Đồ thị Euler và đồ thị Hamilton**Đồ thị Euler**

Leonhard Euler là người đầu tiên mô hình hóa bài toán rác rưởi này thành một hệ thống hoàn chỉnh là Lý thuyết đồ thị hiện nay.

1 Definition 35 (Chu trình Euler)

Chu trình đi qua tất cả các cạnh của đồ thị, mỗi cạnh đúng một lần, được gọi là **chu trình Euler** (hay **Euler circuit**, **Euler circle**, **Euler tour**).

1 Definition 36 (Đường đi Euler)

Đường đi đi qua tất cả các cạnh của đồ thị, mỗi cạnh đúng một lần, được gọi là **đường đi Euler**.

Lúc này, bài toán 7 cây cầu nổi tiếng của thành phố Konigsberg (nay là thành phố Kaliningrad thuộc Liên bang Nga) trở thành bài toán xác định xem có tồn tại chu trình Euler hay không.

1 Definition 37 (Đồ thị Euler)

Đồ thị có chu trình Euler được gọi là **đồ thị Euler** (hay **Eulerian graph**, **unicursal graph**).

1 Definition 38 (Đồ thị nửa Euler)

Đồ thị có đường đi Euler được gọi là **đồ thị nửa Euler** (hay **Semi-Eulerian graph**, **Traversable graph**).

Các định lí và thuật toán trên đồ thị Euler**1 Theorem 15**

Một đồ thị vô hướng liên thông $G = (V, E)$ có chu trình Euler khi và chỉ khi mọi đỉnh của nó đều có bậc chẵn.

1 [TODO] Chứng minh

Chiều thuận. Nếu G có chu trình Euler thì khi đi theo chu trình đó, mỗi đỉnh sẽ đi vào một lần và đi ra một lần nên bậc của nó tăng lên 2. Áp dụng cho mỗi lần gấp một đỉnh thì cuối cùng bậc của mỗi đỉnh phải là số chẵn

[TODO] **Chiều ngược.** Nếu mọi đỉnh của đồ thị đều có bậc chẵn, ta cần xây dựng cách tìm chu trình Euler.

i Corollary 3

Một đồ thị vô hướng liên thông $G = (V, E)$ có đường đi Euler khi và chỉ khi nó có đúng hai đỉnh bậc lẻ.

i Chứng minh

Điều kiện của đường đi Euler "lỏng hơn" chu trình vì điểm đầu không cần phải trùng với điểm cuối. Do đó nếu G có đường đi Euler thì đỉnh bắt đầu và kết thúc sẽ có bậc lẻ, các đỉnh còn lại có bậc chẵn. Ngược lại nếu đồ thị liên thông có đúng hai đỉnh bậc lẻ, khi đó ta vẽ cạnh nối hai đỉnh đó thì tất cả đỉnh của đồ thị đều có bậc chẵn, nghĩa là tồn tại chu trình Euler. Loại bỏ cạnh đó thì ta có đường đi Euler.

[TODO] Chu trình Euler và đường đi Euler của đồ thị có hướng liên thông yêu.

Đồ thị Hamilton

Khái niệm đường đi và chu trình Hamilton được đưa ra bởi William Rowan Hamilton (1856).

i Definition 39 (Chu trình Hamilton và đồ thị Hamilton)

Đồ thị $G = (V, E)$ được gọi là **đồ thị Hamilton** (hay **Hamilton graph**) nếu tồn tại chu trình đơn đi qua tất cả các đỉnh.

Chu trình đơn đi qua tất cả các đỉnh gọi là **chu trình Hamilton** (hay **Hamiltonian circuit**, **Hamilton circle**).

Người ta quy ước rằng đồ thị chỉ gồm một đỉnh là đồ thị Hamilton nhưng đồ thị gồm hai đỉnh liên thông không phải là đồ thị Hamilton.

Mật mã học cũng khó

3.1 Đại số Boolean và mật mã học

3.1.1 Giới thiệu

Giới thiệu

Trung tâm của stream cipher và block cipher là các hàm boolean.

Ở những bài viết ở phần "Đại số boolean và mật mã học" này mình sẽ mô tả các đặc trưng khi xây dựng các hệ mật mã dạng dòng (stream cipher) và khối (block cipher) từ các hàm boolean.

Nhắc lại, hàm boolean n biến là ánh xạ f từ $\{0, 1\}^n$ tới $\{0, 1\}$. Ở phần này mình sẽ sử dụng kí hiệu trường \mathbb{F}_2 . Như vậy hàm boolean trên n biến là ánh xạ

$$f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2, \quad f(x_1, x_2, \dots, x_n) = y.$$

Tiếp theo, khi "ghép" các hàm boolean lại ta có **hàm boolean vector** (hay **vectorial Boolean function**). Như vậy hàm boolean vector là ánh xạ

$$F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m, \quad f(x_1, x_2, \dots, x_n) = (y_1, y_2, \dots, y_m) \in \mathbb{F}_2^m.$$

Như vậy chúng ta có thể coi mỗi hàm $y_i = f_i(x_1, \dots, x_n)$ là một hàm boolean nên khi ghép cạnh nhau chúng ta có hàm boolean vector.

x_1	x_2	\cdots	x_n	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	\cdots	$f_m(\mathbf{x})$
0	0	\cdots	0	$f_1(0, \dots, 0)$	$f_2(0, \dots, 0)$	\cdots	$f_m(0, \dots, 0)$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
1	1	\cdots	1	$f_1(1, \dots, 1)$	$f_2(1, \dots, 1)$	\cdots	$f_m(1, \dots, 1)$

Ở đây $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{F}_2^n$.

Bảng kí hiệu

Kí hiệu	Ý nghĩa
\mathcal{F}_n	Tập hợp tất cả hàm boolean n biến
\mathcal{L}_n	Tập hợp tất cả hàm boolean affine n biến
\mathcal{C}_n	Tập hợp tất cả hàm boolean tuyến tính n biến
$\deg f$	Bậc của hàm boolean f ở dạng chuẩn tắc đại số (ANF)
$\text{wt}(f)$	Trọng số của hàm boolean f
N_f	Nonlinearity của hàm boolean f
$W_f(\mathbf{a})$	Hệ số Walsh của hàm f ứng với vector \mathbf{a}

Các tính chất mật mã của hàm boolean**Bậc đại số cao**

Tham số $\deg f$ phải cao. Điều này đặc biệt quan trọng trong các stream cipher sử dụng LFSR.

Nonlinearity cao

Nonlinearity cực kì quan trọng trong việc chống phá mã tuyến tính (linear cryptanalysis). Nonlinearity càng cao, dấu vết tuyến tính càng thấp.

Hàm boolean có nonlinearity cực đại được gọi là **hàm bent** (hay **bent function**).

Theo phần đại số boolean ở trước thì

$$N_f \leq 2^{n-1} - \frac{1}{2} \cdot 2^{n/2-1}$$

khi n chẵn.

Điều kiện cần và đủ để tồn tại hàm boolean n biến là n chẵn.

Nếu n lẻ thì không tồn tại hàm bent n biến. Tuy nhiên chúng ta vẫn có thể xem xét các hàm có nonlinearity N_f lớn nhất và gọi chúng là **Almost Bent (AB)**.

Khi đó

$$N_f \leq 2^{n-1} - 2^{(n-1)/2}.$$

Bài tập

- Chứng minh rằng khoảng cách từ hàm boolean f với n biến tới hàm boolean affine

$$l_{\mathbf{a}, b}(\mathbf{x}) = a_1x_1 \oplus \dots \oplus a_nx_n \oplus b$$

với $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{F}_2^n$ và $b \in \mathbb{F}_2$ được tính theo công thức:

$$\begin{aligned} d(f, l_{\mathbf{a}, 0}(\mathbf{x})) &= 2^{n-1} - \frac{1}{2}W_f(\mathbf{a}), \\ d(f, l_{\mathbf{a}, 1}(\mathbf{x})) &= 2^{n-1} + \frac{1}{2}W_f(\mathbf{a}). \end{aligned}$$

- Chứng minh rằng nonlinearity của hàm boolean f bắt kì được tính bởi công thức

$$N_f = 2^{n-1} - \frac{1}{2} \max_{\mathbf{y}} |W_f(\mathbf{y})|.$$

- Chứng minh rằng hàm boolean f là hàm bent khi và chỉ khi $W_f(\mathbf{y}) = \pm 2^{n/2}$ với mọi vector \mathbf{y} .

Balanced

Hàm boolean được gọi là **balanced** (hay cân bằng, сбалансированный) nếu nhận giá trị 0 và 1 nhiều như nhau. Như vậy nếu hàm boolean f trên n biến cân bằng khi và chỉ khi

$$\text{wt}(f) = 2^{n-1}.$$

Bài tập: Xác định số lượng hàm boolean cân bằng có n biến.

r -resilient

Đặt r là số nguyên không âm nhỏ hơn n . Hàm boolean f với n biến được gọi là r -resilient (hay r -устойчивой) nếu với mọi hàm con mà nhận được từ việc cố định r biến thì đều là hàm cân bằng.

Hàm boolean này có độ an toàn cao hơn so với hàm cân bằng, giúp chống lại cách tấn công correlation cryptanalysis.

Correlation immune

Hàm boolean f với n biến được gọi là **correlation immune of order r** (корреляционно-иммунной порядка r , tạm dịch là *kháng tương quan bậc r*) với $1 \leq r \leq n$ nếu với mọi hàm con $f_{i_1, \dots, i_r}^{a_1, \dots, a_r}$ nhận được từ việc cố định r biến thì đều thỏa đẳng thức

$$\text{wt}(f_{i_1, \dots, i_r}^{a_1, \dots, a_r}) = \frac{\text{wt}(f)}{2^r}.$$

Bài tập

- Chứng minh rằng hàm boolean f là r -resilient khi và chỉ khi nó cân bằng và correlation immune bậc r .
- (**Định lí Siegenthaler I**, 1984). Chứng minh rằng nếu hàm boolean f là correlation immune bậc r thì $\deg f + r \leq n$.
- (**Định lí Siegenthaler II**) Chứng minh rằng nếu hàm boolean f là r -resilient và $r \leq n - 2$ thì $\deg f + r \leq n - 1$.

Chứng minh cho các định lí Siegenthaler có thể tìm ở [13].

- Chứng minh rằng hàm boolean f là correlation immune bậc r khi và chỉ khi $W_f(\mathbf{y}) = 0$ với mọi vector \mathbf{y} thỏa $1 \leq \text{wt}(\mathbf{y}) \leq r$.

Năm 2007 D. Г. Фон–Дер–Флаacc tìm được chặn trên cho correlation immune của hàm boolean không cân bằng.

- (**Định lí D. Г. Фон–Дер–Флаacc**, [14]). Gọi f là hàm boolean không cân bằng có correlation immune bậc r khác 0. Chứng minh rằng $r \leq (2n/3) - 1$.
- Chứng minh rằng với hàm boolean r -resilient f sao cho $r \leq n - 2$ thì ta có bất đẳng thức $N_f \leq 2^{n-1} - 2^{r+1}$ [15].

Algebraic immune

Tính chất này được giới thiệu vào năm 2004.

Algebraic immune (tạm dịch là *kháng đại số*) của hàm boolean f là số d nhỏ nhất sao cho tồn tại hàm boolean g bậc d , không đồng nhất với 0, thỏa mãn $fg = 0$ hoặc $(f \oplus \mathbf{1})g = 0$.

Algebraic immune của hàm f được kí hiệu là $\text{Al}(f)$.

Ví dụ algebraic immune hàm $f(\mathbf{x}) = x_1x_2x_3 \oplus x_1$ bằng 1, vì ta có thể chọn $g(\mathbf{x}) = x_1 \oplus 1$. Khi đó $fg = (x_1x_2x_3 \oplus x_1)(x_1 \oplus 1) = 1$.

Bài tập

1. Chứng minh rằng algebraic immune của hàm boolean f bất kì với n biến không vượt quá giá trị $\lceil n/2 \rceil$.
2. Chứng minh một số tính chất cơ bản của algebraic immune:
 - $\text{AI}(f) \leq \deg f$;
 - $\text{AI}(f \cdot g) \leq \text{AI}(f) + \text{AI}(g)$;
 - $\text{AI}(f \oplus g) \leq \text{AI}(f) + \text{AI}(g)$;
 - $\text{AI}(f) = \text{AI}(g)$ nếu g nhận được từ f qua một biến đổi affine trên các biến, nghĩa là $g(\mathbf{x}) = f(\mathbf{Ax} \oplus \mathbf{b})$ với \mathbf{A} là ma trận khả nghịch bậc n và \mathbf{b} là vector.
3. Xác định giá trị của $\text{AI}(f)$ với các hàm boolean sau và tìm hàm g tương ứng:
 - $f(\mathbf{x}) = x_1x_2x_4 \oplus x_1x_2 \oplus 1$;
 - $f(\mathbf{x}) = 0$;
 - $f(\mathbf{x}) = 1$;
 - $f(\mathbf{x}) = x_1 \cdots x_k$ với $k = 1, 2, \dots, n$;
 - $f(\mathbf{x}) = x_1 \oplus \dots \oplus x_n$;
 - $f(\mathbf{x}) = x_1x_2 \oplus \dots \oplus x_{n-1}x_n$ (tổng tất cả cấp tích);
 - $f(\mathbf{x}) = x_1x_2x_3x_4 \oplus x_5x_6$.
4. Cho ví dụ hàm boolean f với giá trị algebraic immune nhỏ nhất, nghĩa là $\text{AI}(f) = d$ với $d = 1, 2, \dots, k$.

Differentially δ -uniform

Differential δ -uniform

Khái niệm này lần đầu được định nghĩa trong [16].

Hàm boolean vector $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ gọi là **differentially δ - uniform** nếu với mọi vector \mathbf{a} khác không và vector \mathbf{b} bất kì thì phương trình

$$F(\mathbf{x}) \oplus F(\mathbf{x} \oplus \mathbf{a}) = \mathbf{b}$$

có không quá δ nghiệm với δ là số nguyên dương.

Để ý rằng nếu phương trình có nghiệm là \mathbf{x} thì cũng có nghiệm $\mathbf{x} \oplus \mathbf{a}$. Số δ càng nhỏ thì phép biến đổi của thuật toán mã hóa càng ít có dấu hiệu vi sai, tăng khả năng kháng phá mã vi sai.

Một cách tổng quát ta có định nghĩa sau.

Definition (Differential δ -uniform)

Hàm boolean vector từ \mathbb{F}_p^n tới \mathbb{F}_p^m được gọi là **differential δ - uniform** nếu với mọi $\mathbf{a} \in \mathbb{F}_p^n$ khác không và với mọi \mathbb{F}_p^m thì phương trình

$$F(\mathbf{x} + \mathbf{a}) - F(\mathbf{x}) = \mathbf{b}$$

có không quá δ nghiệm.

Trong mật mã học thường dùng $p = 2$. Thông thường các hàm boolean tập trung vào việc xây dựng các S-box nên n thường là 4 hoặc 8.

Perfect Nonlinear và Almost Perfect Nonlinear

❶ Definition (Hàm Perfect Nonlinear)

Hàm boolean vector F từ \mathbb{F}_p^n tới \mathbb{F}_p^m được gọi là hàm **Perfect Nonlinear (PN)** nếu phương trình

$$F(\mathbf{x} + \mathbf{a}) - F(\mathbf{x}) = \mathbf{b}$$

có đúng p^{n-m} nghiệm với mọi vector $\mathbf{a} \in \mathbb{F}_p^n$ khác không và $\mathbf{b} \in \mathbb{F}_p^m$.

Số lượng hàm PN rất ít. Đối với các giá trị n và p thường được sử dụng trong mật mã thậm chí không tồn tại hàm PN. Do đó chúng ta sẽ nối lỏng điều kiện thành hàm Almost Perfect Nonlinear (APN).

❶ Definition (Hàm Almost Perfect Nonlinear)

Hàm boolean vector F từ \mathbb{F}_p^n tới \mathbb{F}_p^m được gọi là hàm **Almost Perfect Nonlinear (APN)** nếu phương trình

$$F(\mathbf{x} + \mathbf{a}) - F(\mathbf{x}) = \mathbf{b}$$

có không quá hai nghiệm với mọi $\mathbf{a} \in \mathbb{F}_p^n$ khác không và với mọi $\mathbf{b} \in \mathbb{F}_p^m$.

Bài toán khó hiện nay là xây dựng hàm APN là song ánh với số biến n chẵn. Đặc biệt là n có dạng lũy thừa của 2.

Như vậy, theo định nghĩa có thể thấy điều tương đương sau

- APN là differential 2-uniform.
- PN là differential 1-uniform khi $n = m$.

Hoán vị APN

Từ trước tới nay có ba phương pháp xây dựng hoán vị APN trên \mathbb{F}_2^n . Tuy nhiên cả ba phương pháp chỉ hoạt động trên n lẻ. Câu hỏi về việc xây dựng hoán vị APN tới giờ vẫn là vấn đề mở với n chẵn, đặc biệt là n có dạng lũy thừa của 2 như đã nói ở trên.

Bài tập

Chứng minh hàm $S : \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8$ của S-box trong thuật toán AES là differentially 4-uniform.

Bài tập này cho thấy rằng S-box tốt thì δ sẽ nhỏ nhất có thể nhằm kháng phá mã vi sai.

3.1.2 Đại số Boolean

Boolean (hay luận lý) chỉ giá trị đúng hoặc sai của mệnh đề nào đó.

Theo cách hiểu cơ bản, boolean gồm hai giá trị 0 hoặc 1 (sai hoặc đúng). Chương này tham khảo chính từ [17], [18] và [19].

Một số kí hiệu hay dùng:

1. Để chỉ tập hợp tất cả hàm boolean n biến ta dùng \mathcal{F}_n .
2. Để chỉ tập hợp tất cả hàm boolean affine n biến ta dùng \mathcal{A}_n .
3. Để chỉ tập hợp tất cả hàm boolean tuyến tính n biến ta dùng \mathcal{L}_n .

Hàm boolean f đối với các biến x_1, x_2, \dots, x_n là hàm số nhận giá trị trong \mathbb{F}_2^n và trả về giá trị thuộc \mathbb{F}_2 .

Nói cách khác f là ánh xạ từ \mathbb{F}_2^n tới \mathbb{F}_2 .

Ta kí hiệu hàm boolean n biến là $f(x_1, x_2, \dots, x_n)$.

Do $x_i \in \mathbb{F}_2$ nên ta có 2^n vector (bộ số) (x_1, x_2, \dots, x_n) . Giá trị của hàm f lại nằm trong tập $\{0, 1\}$ nên ứng với mỗi vector có thể có 2 giá trị của hàm boolean, và ta có 2^n vector nên số lượng hàm boolean có thể có là 2^{2^n} .

Một số toán tử boolean hay dùng: đối, AND, OR, XOR, NAND, NOR, kéo theo, tương đương.

Để biểu diễn hàm boolean chúng ta dùng bảng chân trị. Bảng chân trị tương ứng với các toán tử boolean trên là:

		AND	OR	XOR	NAND	NOR
x_1	x_2	$x_1 \cdot x_2$	$x_1 \vee x_2$	$x_1 \oplus x_2$	$x_1 x_2$	$x_1 \downarrow x_2$
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	1	0
1	1	1	1	0	0	0

Toán tử đối làm đổi giá trị của hàm bool (0 thành 1 và 1 thành 0), kí hiệu \bar{x} .

Bảng 3.1: Toán tử đối

x	\bar{x}
0	1
1	0

Bảng 3.2: Toán tử kéo theo và tương đương

x_1	x_2	$x_1 \rightarrow x_2$	$x_1 \sim x_2$
0	0	1	1
0	1	1	0
1	0	0	0
1	1	1	1

Toán tử tương đương còn chỉ sự tương đương của hai mệnh đề logic.

Khi hai biểu thức logic có cùng bảng chân trị thì hai mệnh đề đó tương đương nhau. Do đó ta có thể viết một số kết quả như sau (từ các bảng chân trị cơ bản trên):

- $x_1 | x_2 \sim \overline{x_1 \cdot x_2}$. Ở đây ta đổi dấu từng giá trị hàm boolean $x_1 \cdot x_2$;
- $x_1 \downarrow x_2 \sim \overline{x_1 \vee x_2}$. Tương tự ta đổi dấu từng giá trị hàm boolean $x_1 \vee x_2$.

Hàm boolean

Algebraic Normal Form

Đặt $f(\mathbf{x})$ là hàm boolean n biến. Với số $m \leq n$ thì

$$\begin{aligned} f(x_1, \dots, x_n) = & \bigoplus_{a_1, \dots, a_m \in \mathbb{F}_2} (x_1 \oplus a_1 \oplus 1) \times \cdots \times \\ & \times (x_m \oplus a_m \oplus 1) \cdot f(a_1, \dots, a_m, x_{m+1}, \dots, x_n) \end{aligned}$$

Chứng minh

Chọn bộ (b_1, \dots, b_m) bất kì thuộc \mathbb{F}_2^m .

Thay x_i bởi b_i với $i = 1, \dots, m$ thì

$$f(b_1, \dots, b_m, x_{m+1}, x_m) = \bigoplus_{a_1, \dots, a_m \in \mathbb{F}_2} (b_1 \oplus a_1 \oplus 1) \cdots (b_m \oplus a_m \oplus 1) \cdot f(a_1, \dots, a_m, x_{m+1}, \dots, x_n).$$

Ở vế phải, tích $\prod_{i=1}^m (b_i \oplus a_i \oplus 1) = 1$ khi và chỉ khi $b_i \oplus a_i \oplus 1 = 1$ với mọi $i = 1, \dots, m$.

Nói cách khác là khi $b_i \equiv a_i$ thì ta còn f ở vế phải, còn các trường hợp kia thì bằng 0. Do đó ta có điều phải chứng minh.

Khi đó, $f(a_1, \dots, a_m, x_{m+1}, \dots, x_n)$ được gọi là **hệ số khai triển của hàm f theo các biến x_1, \dots, x_m** .

Example 2.6

Xét $f(x_1, x_2) = x_1 x_2 \oplus 1$. Với $m = 1$, ta có

$$\begin{aligned} a_1 = 0 \Rightarrow (x_1 \oplus 0 \oplus 1) \cdot f(0, x_2) &= (x_1 \oplus 1) \cdot 1 = x_1 \oplus 1, \\ a_1 = 1 \Rightarrow (x_1 \oplus 1 \oplus 1) \cdot f(1, x_2) &= x_1 \oplus (x_2 \oplus 1). \end{aligned}$$

Như vậy

$$f(x_1, x_2) = (x_1 \oplus 1) \oplus (x_1 \cdot (x_2 \oplus 1)).$$

Nếu khai triển vế phải ra chúng ta thấy bằng với hàm f ban đầu.

Tương ứng với m biến ta có 2^m hệ số khai triển.

Đặt f_1, \dots, f_{2^m} là các hệ số khai triển hàm f theo m biến bất kì. Khi đó

$$\text{wt}(f) = \sum_{i=1}^{2^m} \text{wt}(f_i). \quad (3.1)$$

Definition 2.24 (Algebraic Normal Form)

Với hàm boolean n biến $f(x_1, x_2, \dots, x_n)$, **algebraic normal form** (hay ANF, **dạng chuẩn tắc đại số**, алгебраическая нормальная форма) tương ứng với hàm bool đó là cách biểu diễn đa thức đó

dưới dạng tổng các tích như sau

$$f(x_1, x_2, \dots, x_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_1 x_2 \oplus \dots \oplus a_k x_1 x_2 \dots x_n$$

với $a_i \in \{0, 1\}$.

Ta thấy rằng có n biến, do đó có 2^n hệ số a_i với mỗi $i = 0, 1, \dots, 2^n - 1$.

Trong các tài liệu tiếng Nga thì ANF còn được gọi là **đa thức Zhegalkin** (hay **полином Жегалкина**)

❶ Definition 2.25 (Bậc của đa thức Zhegalkin)

Tương tự như bậc của một đa thức đại số thông thường, bậc của đa thức Zhegalkin là bậc của đơn thức chứa nhiều biến x_i nhất. Kí hiệu là $\deg(f)$.

❶ Example 2.7

Xét hàm boolean $f(x, y, z) = 1 \oplus x \oplus yz \oplus xyz$. Khi đó $\deg(f) = 3$ vì đơn thức chứa nhiều biến nhất là xyz có 3 đơn thức.

Xét hàm boolean $f(x, y, z) = 1 \oplus z \oplus zy \oplus xy$. Khi đó $\deg(f) = 2$ vì đơn thức chứa nhiều biến nhất là zy (cũng có thể xét xy).

❶ Definition 2.26 (Trọng số của hàm boolean)

Trọng số (hay **weight**, **вес**) của hàm boolean n biến $f(x_1, x_2, \dots, x_n)$ là số lượng giá trị khác 0 của hàm f .

Kí hiệu là $\text{wt}(f)$.

❶ Example 2.8

Hàm boolean $f(x, y) = (0, 1, 0, 1)$ có trọng số $\text{wt}(f) = 2$.

Hàm boolean $f(x, y, z) = (1, 0, 1, 1, 1, 0, 0, 1)$ có trọng số $\text{wt}(f) = 5$.

❶ Property 2.2 (Một số tính chất của trọng số)

Gọi f là hàm boolean n biến. Khi đó:

1. $0 \leq \text{wt}(f) \leq 2^n$.
2. $\text{wt}(f \oplus \mathbf{1}) = 2^n - \text{wt}(f)$.

3. Nếu h cũng là một hàm boolean n biến thì

$$\text{wt}(f \oplus h) = \text{wt}(f) + \text{wt}(h) - 2\text{wt}(fh).$$

4. $\text{wt}(f)$ nhận giá trị lẻ khi và chỉ khi $\deg(f) = n$.

❶ Definition 2.27 (Hàm boolean cân bằng)

Nếu hàm boolean n biến f có trọng số bằng 2^{n-1} thì f được gọi là hàm boolean **cân bằng** (hay **balanced**, **сбалансированная**).

❷ Remark 2.8

Ta nói hàm boolean g giả phụ thuộc vào biến y nếu $g(x_1, \dots, x_n, y) = f(x_1, \dots, x_n)$. Khi đó $\text{wt}(g) = 2\text{wt}(f)$.

❸ Chứng minh

Do

$$g(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n, 1) = f(x_1, \dots, x_n)$$

nên ta có điều phải chứng minh.

❹ Remark 2.9

Đặt $f(x_1, \dots, x_n)$ và $g(y_1, \dots, y_m)$ là các hàm boolean phụ thuộc vào tập các biến không giao nhau. Khi đó:

1. Nếu f và g là các hàm số khác hằng 1 thì $f \cdot g$ không là hàm cân bằng.
2. f hoặc g cân bằng khi và chỉ khi $f \oplus g$ cân bằng.

❺ Chứng minh

Đặt $\mathbf{x} = (x_1, \dots, x_n)$ và $\mathbf{y} = (y_1, \dots, y_m)$.

1. Đặt $\text{wt}(f) = r < 2^n$ và $\text{wt}(g) = s < 2^m$. Vì $f(\mathbf{x}) \cdot g(\mathbf{y}) = 1$ khi và chỉ khi $f(\mathbf{x}) = g(\mathbf{y}) = 1$ nên $\text{wt}(f \cdot g) = r \cdot s$.

Do đó nếu $f \cdot g$ cân bằng thì $2^{n+m-1} = \text{wt}(f \cdot g) = r \cdot s$.

Như vậy $r = 2^k$ và $s = 2^l$ với $k \leq n - 1$ và $l \leq m - 1$.

Suy ra $r \cdot s \leq 2^{n+m-2}$. Điều này vô lý vì $r \cdot s = 2^{n+m-1} > 2^{n+m-2}$. Như vậy giả sử ban đầu $r < 2^n$ là sai, tương tự với s và ta có điều phải chứng minh.

2. Chú ý rằng $f(\mathbf{x}) \oplus g(\mathbf{y}) = 1$ khi và chỉ khi $f(\mathbf{x}) \neq g(\mathbf{y})$, suy ra

$$\text{wt}(f \oplus g) = \text{wt}(f) \cdot \text{wt}(\bar{g}) + \text{wt}(\bar{f}) \cdot \text{wt}(g).$$

Điều kiện đủ. Giả sử hàm f cân bằng, suy ra

$$\text{wt}(f) = \text{wt}(\bar{f}) = 2^{n-1}.$$

Như vậy

$$\text{wt}(f \oplus g) = 2^{n-1} \cdot \text{wt}(g) + 2^{n-1} \cdot \text{wt}(\bar{g}) = 2^{n-1} \cdot 2^m.$$

Vậy $f \oplus g$ là hàm cân bằng.

Điều kiện cần. Giả sử $\text{wt}(f) = r \neq 2^{n-1}$ và $\text{wt}(g) = s$. Như vậy

$$\text{wt}(f \oplus g) = r(2^m - s) + s(2^n - r) = 2^{n+m-1}$$

do $f \oplus g$ là hàm cân bằng. Tiếp theo

$$s = \frac{2^{n+m-1} - 2^m \cdot r}{2^n - 2r} = 2^{m-1}.$$

Vậy g là hàm cân bằng.

Đặt

$$f(x_1, \dots, x_n) = \bigoplus_{a_1, \dots, a_n \in \mathbb{F}_2} g(a_1, \dots, a_n) \cdot x_1^{a_1} \cdots x_n^{a_n}. \quad (3.2)$$

Hàm g khi đó được gọi là **hệ số ANF** của hàm f .

Ánh xạ $\mu(f) = g$ được gọi là **biến đổi Möbius** (hay **преобразование Мёбиуса**).

Example 2.9

Cho hàm bool $f(x, y) = x \vee y$. Ta có bảng chân trị sau.

x	y	$f(x, y)$
0	0	0
0	1	1
1	0	1
1	1	1

Bảng chân trị này tương đương với đa thức Zhegalkin

$$f(x, y) = x \oplus y \oplus xy.$$

ANF ở ví dụ trên có thể được viết lại

$$f(x, y) = 0 \cdot x^0 y^0 \oplus 1 \cdot x^0 y^1 \oplus 1 \cdot x^1 y^0 \oplus 1 \cdot x^1 y^1.$$

Như vậy biến đổi Möbius của hàm f là

x	y	$f(x, y)$	$g = \mu(f)$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Ta ký hiệu $\mathbf{x}^{\mathbf{a}} = x_1^{a_1} \cdots x_n^{a_n}$ với

- $\mathbf{x} = (x_1, \dots, x_n)$;

- $\mathbf{a} = (a_1, \dots, a_n)$.

Do $x_i^{a_i} = 1$ khi và chỉ khi $a_i \leq x_i$, ta có $\mathbf{x}^{\mathbf{a}} = 1$ khi và chỉ khi $\mathbf{a} \preceq \mathbf{x}$ theo nghĩa $a_i \leq x_i$ với $i = 1, \dots, n$.

Ta có thể viết lại (3.2) là

$$f(\mathbf{x}) = \bigoplus_{\mathbf{a} \in \mathbb{F}_2^n} g(\mathbf{a}) \cdot \mathbf{x}^{\mathbf{a}} = \bigoplus_{\mathbf{a} \in \mathbb{F}_2^n} g(\mathbf{a}).$$

Remark 2.10 (Biến đổi Möbius)

Đặt $f \in \mathcal{F}_n$ và $g = \mu(f)$. Khi đó với mọi $\mathbf{a} \in \mathbb{F}_2^n$ ta có

$$g(\mathbf{a}) = \bigoplus_{\mathbf{x} \preceq \mathbf{a}} f(\mathbf{x}).$$

Chứng minh

Ta chứng minh bằng quy nạp theo trọng số của \mathbf{a} .

Ở bước cơ sở khi trọng số bằng không, $g(\mathbf{0}) = f(\mathbf{0})$ với $\mathbf{0}$ là vector chứa n số 0.

Giả thiết quy nạp: giả sử mệnh đề đúng với mọi vector \mathbf{a} có trọng số nhỏ hơn p .

Khi \mathbf{a} có trọng số bằng p , ta có

$$\begin{aligned} f(\mathbf{a}) &= \bigoplus_{\mathbf{x} \preceq \mathbf{a}} g(\mathbf{x}) = \left(\bigoplus_{\mathbf{x} \prec \mathbf{a}} g(\mathbf{x}) \right) \oplus g(\mathbf{a}) \quad (\text{tách thành phần nhỏ hơn và bằng } \mathbf{a}) \\ &= \left(\bigoplus_{\mathbf{x} \prec \mathbf{a}} \bigoplus_{\mathbf{y} \preceq \mathbf{x}} f(\mathbf{y}) \right) \oplus g(\mathbf{a}) \quad (\text{sử dụng giả thiết quy nạp thay } g(\mathbf{x})). \end{aligned}$$

Đặt

$$S = \bigoplus_{\mathbf{x} \prec \mathbf{a}} \bigoplus_{\mathbf{y} \preceq \mathbf{x}} f(\mathbf{y}) = \bigoplus_{\mathbf{y} \prec \mathbf{a}} f(\mathbf{y}) \bigoplus_{\mathbf{y} \preceq \mathbf{x} \prec \mathbf{a}} 1 = \bigoplus_{\mathbf{y} \prec \mathbf{a}} f(\mathbf{y}).$$

Đẳng thức thứ hai đúng là do \mathbf{y} nhận tất cả vector từ $\mathbf{0}$ tới \mathbf{x} mà $\mathbf{x} \prec \mathbf{a}$ nên thực chất có thể thay \mathbf{x} thành \mathbf{y} .

Đẳng thức cuối đúng là do $2^{\text{wt}(\mathbf{a}) - \text{wt}(\mathbf{y})} - 1$ là số lẻ nào đó mà $\mathbf{y} \preceq \mathbf{x} \prec \mathbf{a}$, suy ra $g(\mathbf{a}) = S \oplus f(\mathbf{a}) = \bigoplus_{\mathbf{y} \preceq \mathbf{a}} f(\mathbf{y})$.

Corollary 2.1

$$\mu(\mu(f)) = f.$$

➊ Remark 2.11

$$g(\mathbf{1}) = \bigoplus_{\mathbf{x} \in \mathbb{F}_2^n} f(\mathbf{x})$$

với $\mathbf{1}$ là vector có n số 1.

➋ Remark 2.12

Nếu $f \in \mathcal{F}_n$ và $\deg f = d \geq 1$ thì

$$2^{n-d} \leq \text{wt}(f) \leq 2^n - 2^{n-d}.$$

➌ Chứng minh

Đặt $x_{i_1} \cdots x_{i_d}$ là đơn thức có bậc cao nhất ở ANF. Khai triển f thành $n - d$ biến và đặt $f_1, \dots, f_{2^{n-d}}$ là các hệ số khai triển.

Ở ANF, mỗi hệ số đều có x_{i_1}, \dots, x_{i_d} nên mọi hệ số đều khác hằng, suy ra

$$1 \leq \text{wt}(f_i) \leq 2^d - 1$$

với $i = 1, \dots, 2^{n-d}$. Ta có điều phải chứng minh.

Nói riêng, nếu $\deg f = 1$ thì f là hàm cân bằng.

Phụ thuộc tuyến tính**➍ Definition 2.28**

Hàm $f(x_1, \dots, x_n)$ được gọi là **linear dependent** (hay **линейно зависим**) vào biến x_i nếu f có thể biểu diễn ở dạng

$$f(x_1, \dots, x_n) = g(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) \oplus x_i$$

với $g \in \mathcal{F}_{n-1}$.

Theo trường hợp riêng ở trên thì nếu f linear dependent vào một biến bất kì thì f là hàm cân bằng.

Ta có thể diễn đạt định nghĩa trên theo cách khác:

$$f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) \neq f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n).$$

➎ Definition 2.29

Hàm $f(x_1, \dots, x_n)$ được gọi là **quasi-linear dependent** (hay **квазилинейно зависит**) trên cặp biến x_i và x_j nếu f đổi giá trị khi ta đảo giá trị ở vị trí i và j .

Nói cách khác, ta có

$$f(x_1, \dots, x_i, \dots, x_j, \dots, x_n) = \bar{f}(x_1, \dots, \bar{x}_i, \dots, \bar{x}_j, \dots, x_n)$$

với $x_i, x_j \in \mathbb{F}_2$.

➊ Remark 2.13

Hàm $f(x_1, \dots, x_n, y, z)$ là hàm quasi-linear dependent trên biến y và z khi và chỉ khi f có dạng

$$f(x_1, \dots, x_n, y, z) = g(x_1, \dots, x_n, y \oplus z) \oplus y.$$

➋ Chứng minh

Điều kiện cần. Đặt $\mathbf{x} = \mathbb{F}_2^n$. Khi đó

$$\begin{aligned} f(\mathbf{x}, 0, 0) &= g(\mathbf{x}, 0), & f(\mathbf{x}, 1, 1) &= g(\mathbf{x}, 0) \oplus 1 = \bar{f}(\mathbf{x}, 0, 0) \\ f(\mathbf{x}, 0, 1) &= g(\mathbf{x}, 1), & f(\mathbf{x}, 1, 0) &= g(\mathbf{x}, 1) \oplus 1 = \bar{f}(\mathbf{x}, 1, 0). \end{aligned}$$

Như vậy f là quasi-linear dependent.

Điều kiện đủ. Khai triển f theo y và z , sau đó thay $(bm\{x\}, 0, 0)$ bởi $f(\mathbf{x}, 0, 0) \oplus 1$, tương tự $f(\mathbf{x}, 0, 1)$ thành $f(\mathbf{x}, 0, 1) \oplus 1$. Nói cách khác là

$$\begin{aligned} f(\mathbf{x}, y, z) &= (y \oplus 1) \cdot (z \oplus 1) \cdot f(\mathbf{x}, 0, 0) \\ &\quad \oplus (y \oplus 1) \cdot z \cdot f(\mathbf{x}, 0, 1) \\ &\quad \oplus y \cdot (z \oplus 1) \cdot f(\mathbf{x}, 1, 0) \\ &\quad \oplus y \cdot z \cdot f(\mathbf{x}, 1, 1). \end{aligned} \tag{3.3}$$

Ta gom hai nhóm:

$$\begin{aligned} &(y \oplus 1) \cdot (z \oplus 1) \cdot f(\mathbf{x}, 0, 0) \oplus y \cdot z \cdot f(\mathbf{x}, 1, 1) \\ &= (yz \oplus y \oplus z \oplus 1) \cdot f(\mathbf{x}, 0, 0) \oplus yz \cdot (f(\mathbf{x}, 0, 0) \oplus 1) \\ &= (y \oplus z \oplus 1) \cdot f(\mathbf{x}, 0, 0) \oplus yz, \end{aligned}$$

và

$$\begin{aligned} &(y \oplus 1) \cdot z \cdot f(\mathbf{x}, 0, 1) \oplus y \cdot (z \oplus 1) \cdot f(\mathbf{x}, 1, 0) \\ &= (y \oplus 1) \cdot z \cdot f(\mathbf{x}, 0, 1) \oplus y \cdot (z \oplus 1) \cdot (f(\mathbf{x}, 0, 1) \oplus 1) \\ &= (y \oplus z) \cdot f(\mathbf{x}, 0, 1) \oplus yz \oplus y. \end{aligned}$$

Tóm lại, phương trình khai triển ở (3.3) sẽ tương đương với

$$f(\mathbf{x}, y, z) = \underbrace{(y \oplus z \oplus 1) \cdot f(\mathbf{x}, 0, 0)}_{g(\mathbf{x}, y \oplus z)} \oplus \underbrace{(y \oplus z) \cdot f(\mathbf{x}, 0, 1)}_{g(\mathbf{x}, y \oplus z)} \oplus y.$$

➊ Remark 2.14

Nếu hàm boolean quasi-linear dependent trên hai biến bất kì thì hàm boolean đó cân bằng.

➋ Chứng minh

Đặt $\mathbf{x} \in \mathbb{F}_2^n$. Do $f(\mathbf{x}, y, z) \in \mathcal{F}_{n+2}$ và f quasi-linear dependent trên hai biến y và z nên theo [Nhận xét 2.13](#) thì f có dạng

$$f(\mathbf{x}, y, z) = g(\mathbf{x}, y \oplus z) \oplus y.$$

Do tổng (XOR) của f có phần tuyến tính nên f cân bằng.

Chúng ta cũng có thể chứng minh theo cách khác bằng khai triển f theo y và z

$$\begin{aligned} f(\mathbf{x}, y, z) &= y \cdot z \cdot (g(\mathbf{x}, 0) \oplus 1) \oplus y \cdot (z \oplus 1) \cdot (g(\mathbf{x}, 1) \oplus 1) \\ &\quad \oplus (y \oplus 1) \cdot z \cdot g(\mathbf{x}, 1) \oplus (y \oplus 1) \cdot (z \oplus 1) \cdot g(\mathbf{x}, 0). \end{aligned}$$

Theo đẳng thức (3.1) thì

$$\text{wt}(f) = \text{wt}(\bar{g}(\mathbf{x}, 0)) + \text{wt}(\bar{g}(\mathbf{x}, 1)) + \text{wt}(g(\mathbf{x}, 1)) + \text{wt}(g(\mathbf{x}, 0)) = 2^{n+1}.$$

➌ Example 2.10 (hàm quasi-linear dependent)

Hàm boolean

$$f(x, y, z) = y \oplus xz \oplus xy$$

quasi-linear dependent trên hai biến y và z .

Cách tìm đa thức Zhegalkin từ bảng chân trị

Ta có nhiều phương pháp để tìm đa thức Zhegalkin của một hàm boolean từ bảng chân trị.

Phương pháp tam giác

Ở hàng đầu ta viết các phần tử bảng chân trị từ trái sang phải. Với n biến sẽ có 2^n ô.

Hàng thứ hai có $2^n - 1$ ô. Phần tử dưới sẽ bằng XOR của 2 phần tử ngay trên nó (tạo thành tam giác). Tiếp tục như vậy tới khi ta có hàng cuối chỉ có 1 ô.

x	y	f	
0	0	0	
0	1	1	
1	0	1	
1	1	1	

Hình 3.1: Phương pháp tam giác

Khi đó, tương ứng với các biến, nếu biến đó là 1 thì đơn thức sẽ chứa biến đó, 0 thì không ghi. Ở ví dụ trên, nếu $(x, y) = (0, 0)$ thì không có gì (phần tử 1), $(x, y) = (0, 1)$ tương ứng với đơn thức y trong đa thức Zhegalkin, $(x, y) = (1, 0)$ tương ứng đơn thức x . Cuối cùng $(x, y) = (1, 1)$ tương ứng đơn thức xy .

Hệ số trước mỗi đơn thức là phần tử đầu tiên bên trái theo bảng kim tự tháp. Như vậy đa thức Zhegalkin là:

$$f(x, y) = 0 \cdot 1 \oplus 1 \cdot y \oplus 1 \cdot x \oplus 1 \cdot xy = x \oplus y \oplus xy.$$

Đa thức Zhegalkin đóng vai trò quan trọng trong nhiều lĩnh vực, bao gồm cả toán học, vật lý, khoa học máy tính, vì AND và XOR là hai toán tử đại số cơ bản, do đó biểu diễn đa thức Zhegalkin được gọi là dạng chuẩn tắc đại số như ở trên đề cập.

Một ví dụ khác của đa thức Zhegalkin với hàm 3 biến x, y và z như hình sau:

x	y	z	f	
0	0	0	0	
0	0	1	1	
0	1	0	1	
0	1	1	1	
1	0	0	0	
1	0	1	0	
1	1	0	1	
1	1	1	1	

Như vậy ứng với hàm boolean f thì đa thức Zhegalkin là:

$$f(x, y, z) = z \oplus yz \oplus x \oplus xz \oplus xyz.$$

Phương pháp Möbius

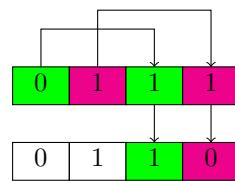
Phương pháp này cho phép chúng ta tính hệ số đa thức Zhegalkin như phương pháp tam giác nhưng nhanh hơn và đỡ sai sót hơn.

Đầu tiên chúng ta chia đôi bảng chân trị thành hai nửa trái phải. Nửa trái giữa nguyên, mỗi phần tử ở nửa phải được XOR (cộng modulo 2) với phần tử tương ứng ở nửa trái.

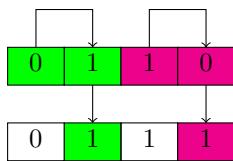
Ví dụ với hàm $f(x, y) = (0, 1, 1, 1)$ như trên.

Bước 1, ta giữ nguyên hai phần tử đầu 0 và 1. Phần tử thứ ba (mới) bằng phần tử thứ ba (cũ) XOR với phần tử đầu, nghĩa là $0 \oplus 1 = 1$. Phần tử thứ tư (mới) bằng phần tử thứ tư (cũ) XOR với phần tử thứ hai, nghĩa là $1 \oplus 1 = 0$.

Tiếp theo, chúng ta xử lý như trên cho hai phần tử bên nửa trái (hai phần tử bên nửa phải xử lý tương tự).



Hình 3.2: Bước 1



Hình 3.3: Bước 2

Như vậy ta có kết quả là $(0, 1, 1, 1)$, tương ứng với các đơn thức $1, y, x, xy$ (như trên). Vậy đa thức Zhegalkin là $f(x, y) = x \oplus y \oplus xy$.

Hàm affine và hàm tuyến tính

Hàm affine và hàm tuyến tính

❶ Definition (Hàm boolean affine)

Xét hàm boolean n biến $f(x_1, x_2, \dots, x_n)$. Khi đó f được gọi là hàm boolean **affine** nếu nó có dạng

$$f(x_1, x_2, \dots, x_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \dots \oplus a_n x_n.$$

Khi $a_0 = 0$ thì ta gọi là hàm boolean **tuyến tính (linear)**.

Ta thấy rằng chỉ có các hạng tử dạng $a_i x_i$ xuất hiện trong biểu diễn đa thức Zhegalkin tương ứng của hàm boolean đó, hay nói cách khác hàm boolean là affine khi $\deg(f) = 1$.

❷ Example

Hàm boolean $f(x, y) = x \oplus y$ là hàm boolean affine và cũng tuyến tính.

Hàm boolean $f(x, y) = x \oplus xy$ không là hàm boolean affine.

Số lượng hàm affine và hàm tuyến tính

Ở phần trên ta đã tính được

$$|\mathcal{F}_n| = 2^{2^n}.$$

Số lượng hàm boolean affine là số cách chọn các hệ số a_0, a_1, \dots, a_n . Như vậy cần chọn $n + 1$ số trong \mathbb{F}_2 nên

$$|\mathcal{A}_n| = 2^{n+1}.$$

Đối với hàm boolean tuyến tính thì chọn từ a_1 tới a_n nên cần chọn n số, suy ra

$$|\mathcal{L}_n| = 2^n.$$

Hàm đơn điệu

i Definition 2.31 (Vector so sánh được)

Xét hai vector $\mathbf{a} = (a_1, a_2, \dots, a_n)$ và $\mathbf{b} = (b_1, b_2, \dots, b_n)$ với $a_i, b_i \in \mathbb{F}_2$.

Ta nói \mathbf{a} so sánh được nhỏ hơn \mathbf{b} nếu với mọi $i = 1, 2, \dots, n$ ta có $a_i \leq b_i$. Kí hiệu $\mathbf{a} \preceq \mathbf{b}$.

i Example 2.12

Ta có $(1, 0, 0) \preceq (1, 0, 1)$, còn $(1, 0, 0)$ và $(0, 1, 0)$ thì không so sánh được (vì trí thứ nhất và thứ hai).

i Definition 2.32 (Hàm boolean đơn điệu)

Hàm boolean n biến $f(x_1, x_2, \dots, x_n)$ được gọi là hàm **đơn điệu** (hay **monotone**) nếu với mọi vector $(a_1, a_2, \dots, a_n) \preceq (b_1, b_2, \dots, b_n)$ thì ta có

$$f(a_1, a_2, \dots, a_n) \leq f(b_1, b_2, \dots, b_n).$$

i Example 2.13

Xét hàm boolean $f(x, y) = (0, 1, 0, 1)$.

Ta thấy rằng:

- $(0, 0) \preceq (0, 1)$ và $f(0, 0) = 0 \leq 1 = f(0, 1)$;
- $(0, 0) \preceq (1, 0)$ và $f(0, 0) = 0 \leq 0 = f(1, 0)$;
- $(0, 0) \preceq (1, 1)$ và $f(0, 0) = 0 \leq 1 = f(1, 1)$;
- $(0, 1)$ và $(1, 0)$ không so sánh được nên bỏ qua;
- $(0, 1) \preceq (1, 1)$ và $f(0, 1) = 1 \leq 1 = f(1, 1)$;
- $(1, 0) \preceq (1, 1)$ và $f(1, 0) = 0 \leq 1 = f(1, 1)$.

Vậy đây là hàm đơn điệu.

Disjunctive và Conjunctive Normal Form

Khi xử lý các mạch logic có ba toán tử chúng ta đặc biệt quan tâm là AND, OR, NOT. Điều này khác với đại số boolean ở phần trước chỉ quan tâm phép XOR và AND - phép cộng và nhân trong \mathbb{F}_2 .

Ở phần này kí hiệu:

- NOT của biến a là $\neg a$;
- AND của hai biến a và b là $a \wedge b$. Chúng ta cũng có thể kí hiệu là $a \cdot b$ hoặc ab như đại số boolean ở phần trước.
- OR của hai biến a và b là $a \vee b$. Chúng ta **KHÔNG** dùng kí hiệu $a + b$ ở đây vì phép cộng đã được dùng ở phần ANF.

Phép AND và OR có tính giao hoán, kết hợp.

Một số luật logic

Sau đây là một số "hàng đẳng thức đáng nhớ" trong logic:

1. Luật bù:

- $a \vee \neg a = 1;$
- $a \wedge \neg a = 0.$

2. Luật hấp thụ:

- $x \vee (x \wedge y) = x;$
- $x \wedge (x \vee y) = x.$

3. Luật đồng nhất:

- $x \vee 0 = x;$
- $x \wedge 1 = x.$

4. Luật giao hoán:

- $x \vee y = y \vee x;$
- $x \wedge y = y \wedge x.$

5. Luật kết hợp:

- $x \vee (y \vee z) = (x \vee y) \vee z;$
- $x \wedge (y \wedge z) = (x \wedge y) \wedge z.$

6. Luật phân phối:

- $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z);$
- $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z).$

7. Luật De Morgan:

- $\neg(x \wedge y) = \neg x \vee \neg y;$
- $\neg(x \vee y) = \neg x \wedge \neg y.$

Tổng của tích

Nếu biểu thức boolean được biểu diễn ở dạng tổng của các tích thì ta gọi là **disjunctive normal form** (hay **DNF**).

Example 2.14

$$(a \wedge \neg b \wedge \neg c) \vee (a \wedge b) \vee (b \wedge \neg c).$$

Ở đây, mỗi tích sẽ bao gồm các **literal** (hay **đơn tử**). Ví dụ với tích $(a \wedge \neg b \wedge \neg c)$ thì các literal sẽ là a , $\neg b$ và $\neg c$.

Mỗi tích sẽ được gọi là **term** (hay **hạng tử**). Trong ví dụ trên có ba term là $(a \wedge \neg b \wedge \neg c)$, $(a \wedge b)$ và $(b \wedge \neg c)$.

Biểu thức boolean với n biến x_1, \dots, x_n ở dạng **full disjunctive normal form** (hay **full DNF**) mỗi term của nó có đủ n literal.

Ở ví dụ trên thì các term $a \wedge b$ và $b \wedge \neg c$ chỉ có hai literal nên không phải là full DNF.

i Example 2.15 (Ví dụ về full DNF)

$$(a \wedge b) \vee (a \wedge \neg b) \vee (\neg a \wedge b).$$

Để chuyển một hàm boolean về dạng full DNF chúng ta có thể dùng bảng chân trị.

Đặt $f(x_1, \dots, x_n)$ là hàm boolean cần đưa về dạng full DNF.

1. Nếu $f = 0$ thì ta không xét.
2. Nếu $f = 1$, với $i = 1, \dots, n$ ta xây dựng term như sau:
 - nếu $x_i = 0$ thì literal là $\neg x_i$;
 - nếu $x_i = 1$ thì literal là x_i .

i Example 2.16

Sử dụng ví dụ ở trên

$$f(a, b, c) = (a \wedge \neg b \wedge \neg c) \vee (a \wedge b) \vee (b \wedge \neg c).$$

Bảng chân trị của f sẽ có dạng

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Ở đây f nhận giá trị 1 khi (a, b, c) là các bộ $(0, 1, 0)$, $(1, 0, 0)$, $(1, 1, 0)$ và $(1, 1, 1)$.

- đối với $(0, 1, 0)$ thì term tương ứng là $\neg a \wedge b \wedge \neg c$;
- đối với $(1, 0, 0)$ thì term tương ứng là $a \wedge \neg b \wedge \neg c$;
- đối với $(1, 1, 0)$ thì term tương ứng là $a \wedge b \wedge \neg c$;
- đối với $(1, 1, 1)$ thì term tương ứng là $a \wedge b \wedge c$.

Như vậy hàm f có thể viết lại là

$$f(a, b, c) = (\neg a \wedge b \wedge \neg c) \vee (a \wedge \neg b \wedge \neg c) \vee (a \wedge b \wedge \neg c) \vee (a \wedge b \wedge c).$$

Tích của các tổng

Nếu biểu thức boolean được biểu diễn ở dạng tích của các tổng thì ta gọi là **conjunctive normal form** (hay **CNF**).

Example 2.17

$$(a \vee \neg b \vee \neg c) \wedge (a \vee b) \wedge (b \vee \neg c).$$

Tương tự, mỗi tổng sẽ bao gồm các **literal** (hay **đơn tử**). Ví dụ với tổng $(a \vee \neg b \vee \neg c)$ thì các literal sẽ là a , $\neg b$ và $\neg c$.

Mỗi tích sẽ được gọi là **term** (hay **hạng tử**). Trong ví dụ trên có ba term là $(a \vee \neg b \vee \neg c)$, $(a \vee b)$ và $(b \vee \neg c)$. Biểu thức boolean với n biến x_1, \dots, x_n ở dạng **full conjunctive normal form** (hay **full CNF**) mỗi term của nó có đủ n literal.

Ở ví dụ trên thì các term $a \vee b$ và $b \vee \neg c$ chỉ có hai literal nên không phải là full CNF.

Tương tự, để chuyển một hàm boolean về dạng full CNF chúng ta có thể dùng bảng chân trị nhưng làm ngược lại so với DNF.

Đặt $f(x_1, \dots, x_n)$ là hàm boolean cần đưa về dạng full CNF.

1. Nếu $f = 1$ thì ta không xét.
2. Nếu $f = 0$, với $i = 1, \dots, n$ ta xây dựng term như sau:
 - nếu $x_i = 0$ thì literal là x_i ;
 - nếu $x_i = 1$ thì literal là $\neg x_i$.

Example 2.18

Sử dụng ví dụ ở trên

$$f(a, b, c) = (a \vee \neg b \vee \neg c) \wedge (a \vee b) \wedge (b \vee \neg c).$$

Bảng chân trị của f sẽ có dạng

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Ở đây f nhận giá trị 0 khi (a, b, c) là các bộ $(0, 0, 0)$, $(0, 0, 1)$, $(0, 1, 1)$, và $(1, 0, 1)$.

- đối với $(0, 0, 0)$ thì term tương ứng là $a \vee b \vee c$;
- đối với $(0, 0, 1)$ thì term tương ứng là $a \vee b \vee \neg c$;

- đối với $(0, 1, 1)$ thì term tương ứng là $a \vee \neg b \vee \neg c$;
- đối với $(1, 0, 1)$ thì term tương ứng là $\neg a \vee b \vee \neg c$.

Như vậy hàm f có thể viết lại là

$$f(a, b, c) = (a \vee b \vee c) \wedge (a \vee b \vee \neg c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee b \vee \neg c).$$

Chúng ta có thể sử dụng đoạn code sau của SageMath để tìm full CNF (các bạn có thể xem ở [đây](#)):

```
import sage.logic.propcalc as propcalc
s = propcalc.formula("(a | ~b | ~c) & (a | b) & (b | ~c)")
s.convert_cnf()
print(s)
```

Phương pháp bìa Karnaugh tìm biểu thức tối thiểu

Chúng ta đã thấy rằng từ một biểu thức DNF hoặc CNF thì tìm được biểu thức full DNF hoặc full CNF.

Tuy nhiên trong thực tế chúng ta quan tâm đến biểu thức ít term, và mỗi term có ít literal nhất có thể. Do đó chúng ta cần một phương án rút gọn biểu thức boolean.

Đối với hàm ba hoặc bốn biến chúng ta có thể sử dụng **bìa Karnaugh** (hay **Karnaugh map**, **капта Капно**).

Ở phần sau mình sẽ trình bày cách tìm tất cả biểu thức tối thiểu đối với DNF. Đối với CNF làm ngược lại.

Trường hợp ba biến

Giả sử ta có biểu thức boolean $f(x_1, x_2, x_3)$.

Ta viết bảng gồm 2 hàng (ứng với 2 giá trị của x_1 thuộc $\{0, 1\}$) và 4 cột (ứng với 4 giá trị x_2x_3 thuộc $\{00, 01, 10, 11\}$).

Quan trọng

Hai ô kề nhau khác nhau đúng **MỘT** bit.

				x_2x_3
	00	01	11	10
x_1	0			
1				

Hình 3.4: Bìa Karnaugh

Ở [Hình 3.4](#), x_2x_3 được viết theo thứ tự 00, 01, 11 rồi mới tới 10. Điều này đảm bảo với lưu ý ở trên là hai ô kề nhau khác nhau đúng một bit.

Tiếp theo, ta điền giá trị của hàm f vào bảng. Ở đây, ô ở hàng x_1 và ở cột x_2x_3 sẽ là giá trị $f(x_1, x_2, x_3)$.

		x_2x_3				
		00	01	11	10	
x_1		0	0	1	1	1
		1	0	0	1	0

Hình 3.5: Điền giá trị vào bìa Karnaugh

Lấy ví dụ DNF ở trên với bảng chân trị tương ứng:

$$f(x_1, x_2, x_3) = (x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2) \vee (x_2 \wedge \neg x_3).$$

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Khi đó bìa Karnaugh sẽ là

		x_2x_3			
		00	01	11	10
x_1		0	1	1	1
		1		1	

Tiếp theo chúng ta xác định các té bào, là các term với số lượng literal nhỏ nhất có thể.

Đối với dạng DNF, chúng ta quan tâm các ô có giá trị 1. Ta gom các ô có giá trị 1 kề nhau tạo thành hình chữ nhật có 2^i ô với $i = 0, 1, \dots$

Ở hình trên có ba té bào theo thứ tự là x_1, x_2 và x_3 (Hình 3.6):

- 001 và 011;
- 011 và 010;

- 011 và 111.

		x_2x_3			
		00	01	11	10
		0	1	1	1
x_1				1	
	1				

Hình 3.6: Các tế bào của biểu thức f

Mỗi tế bào sẽ tương ứng với term ở các vị trí giống nhau. Nói cách khác:

- tế bào màu đỏ 001 và 011 giống nhau ở bit đầu và bit thứ ba, tương ứng x_1 và x_3 nên term là $\neg x_1 \wedge x_3$. Ta đặt $T_1 = \neg x_1 \wedge x_3$;
- tế bào màu xanh lá 011 và 010 giống nhau ở bit đầu và bit thứ hai, tương ứng x_1 và x_2 nên term là $\neg x_1 \wedge x_2$. Ta đặt $T_2 = \neg x_1 \wedge x_2$;
- tế bào màu vàng 011 và 111 giống nhau ở bit thứ hai và thứ ba, tương ứng x_2 và x_3 nên term là $x_2 \wedge x_3$. Ta đặt $T_3 = x_2 \wedge x_3$.

Luôn nhớ rằng trong DNF thì giá trị 1 ta giữ nguyên (ví dụ x_3) còn giá trị 0 thì ta đảo (ví dụ `$neg x_1$`).

Tiếp theo, ta xét các tế bào có số ô nhiều nhất trước. Ở đây cả ba tế bào đều có hai ô nên tương đương nhau.

- Chọn tế bào T_1 . Ta gạch bỏ tất cả ô của tế bào T_1 .

		x_2x_3			
		00	01	11	10
		0	1	1	1
x_1				1	
	1				

Hình 3.7: Bìa Karnaugh sau khi gạch bỏ T_1

- Trong các tế bào còn lại, ta chọn tế bào chưa bị gạch hết và có số ô cao nhất. Ở đây chúng ta có hai phương án:

- chọn T_2 , ta gạch bỏ T_2 . Tiếp tục quá trình với T_3 , ta gạch bỏ T_3 . Khi đó công thức tối thiểu là $T_1 \rightarrow T_2 \rightarrow T_3$;
- chọn T_3 , ta gach bỏ T_3 . Tiếp tục quá trình với T_2 , ta gach bỏ T_2 . Khi đó công thức tối thiểu là $T_1 \rightarrow T_3 \rightarrow T_2$.

Như vậy ta có hai công thức tối thiểu là $T_1 \rightarrow T_2 \rightarrow T_3$ và $T_1 \rightarrow T_3 \rightarrow T_2$, nghĩa là:

$$\begin{aligned} f(x_1, x_2, x_3) &= (\neg x_1 \wedge x_3) \vee (\neg x_1 \wedge x_2) \vee (x_2 \wedge x_3) \\ &= (\neg x_1 \wedge x_3) \vee (x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2). \end{aligned}$$

Hai công thức hóa ra ... giống nhau. Đó là do biểu thức ban đầu chứ không phải tất cả trường hợp đều như vậy.

Khoảng cách Hamming

❶ Definition 2.33 (Khoảng cách Hamming giữa hai vector)

Với hai vector \mathbf{x}, \mathbf{y} thuộc \mathbb{F}_2^n , đặt

$$d(\mathbf{x}, \mathbf{y}) = \text{wt}(\mathbf{x} \oplus \mathbf{y})$$

là **khoảng cách Hamming** giữa hai vector \mathbf{x} và \mathbf{y} . Trong đó $\text{wt}(\mathbf{z})$ là trọng số vector \mathbf{z} .

❶ Definition 2.34 (Khoảng cách Hamming từ vector tới tập vector)

Xét $M \subseteq \mathbb{F}_2^n$. Khi đó với mọi $\mathbf{x} \in \mathbb{F}_2^n$, ta nói khoảng cách từ \mathbf{x} tới M là

$$d(\mathbf{x}, M) = \min_{\mathbf{y} \in M} d(\mathbf{x}, \mathbf{y}).$$

❶ Definition 2.35 (Khoảng cách Hamming giữa hai hàm boolean)

Xét hai hàm boolean n biến là $f(x_1, x_2, \dots, x_n)$ và $g(x_1, x_2, \dots, x_n)$. Khi đó khoảng cách Hamming từ hàm f tới hàm g là

$$d(f, g) = \text{wt}(f \oplus g) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} f(\mathbf{x}) \oplus g(\mathbf{x}).$$

Nonlinearity của hàm boolean

Biến đổi Fourier

Với mỗi vector $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{F}_2^n$, ta kí hiệu $\langle \mathbf{a}, \mathbf{x} \rangle$ là hàm sau:

$$\langle \mathbf{a}, \mathbf{x} \rangle = a_1 x_1 \oplus a_2 x_2 \oplus \dots \oplus a_n x_n. \quad (3.4)$$

Mỗi hàm boolean $f(\mathbf{x}) = f(x_1, x_2, \dots, x_n)$ sẽ được biểu diễn dưới dạng duy nhất với

$$f(\mathbf{x}) = 2^{-n} \sum_{\mathbf{a} \in \mathbb{F}_2^n} F_f(\mathbf{a}) \cdot (-1)^{\langle \mathbf{a}, \mathbf{x} \rangle}, \quad (3.5)$$

trong đó

$$F_f(\mathbf{a}) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} f(\mathbf{x}) \cdot (-1)^{\langle \mathbf{a}, \mathbf{x} \rangle}. \quad (3.6)$$

Khi đó, tập hợp $\{F_f(\mathbf{a}), \mathbf{a} \in \mathbb{F}_2^n\}$ được gọi là **phổ Fourier** (hay **spectre Fourier**) của hàm boolean f .

❶ Remark 2.15

Đầu tiên ta có nhận xét rằng, với vector z cố định thì

$$\sum_{\mathbf{a} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{a}, z \rangle} = \begin{cases} 0, & \text{nếu } z \neq \mathbf{0} \\ 2^n, & \text{nếu } z = \mathbf{0}. \end{cases} \quad (3.7)$$

❶ Chứng minh

Để chứng minh nhận xét này, ta thấy rằng nếu $z \neq \mathbf{0}$ thì có ít nhất một bit $z_i \neq 0$.

Ta chọn vector

$$\Delta = (0, \dots, 0, 1, 0, \dots, 0)$$

với bit 1 nằm ở vị trí i .

Khi đó với mọi vector $\mathbf{a} \in \mathbb{F}_2^n$ tồn tại duy nhất vector $\mathbf{a}' \in \mathbb{F}_2^n$ sao cho $\mathbf{a} \oplus \mathbf{a}' = \Delta$.

Ta suy ra $\langle \mathbf{a} \oplus \mathbf{a}', z \rangle = \langle \Delta, z \rangle = 1$ vì $z_i \cdot 1 = 1$, các vị trí còn lại $z_j \cdot 0 = 0$.

Lý do ta chọn vector Δ như vậy là vì

$$\langle \mathbf{a} \oplus \mathbf{a}', z \rangle = \langle \mathbf{a}, z \rangle \oplus \langle \mathbf{a}', z \rangle = 1,$$

tương đương với $\langle \mathbf{a}, z \rangle = 1 \oplus \langle \mathbf{a}', z \rangle$.

Do đó $\langle \mathbf{a}, z \rangle$ và $\langle \mathbf{a}', z \rangle$ là hai bit khác nhau, dẫn tới $(-1)^{\langle \mathbf{a}, z \rangle}$ và $(-1)^{\langle \mathbf{a}', z \rangle}$ là hai số trái dấu nhau nên tổng chúng là 0. Chúng ta có $2^n/2$ cặp như vậy và tổng cuối cùng là 0.

Trong trường hợp $z = \mathbf{0}$ thì $\langle \mathbf{a}, z \rangle = 0$ với mọi $\mathbf{a} \in \mathbb{F}_2^n$. Do đó $(-1)^{\langle \mathbf{a}, z \rangle} = (-1)^0 = 1$ với mọi vector \mathbf{a} . Hàm boolean n biến có 2^n vector \mathbf{a} nên tổng là $2^n \cdot 1 = 2^n$.

Đẳng thức (3.7) đã được chứng minh. Ta quay lại bài toán.

Chứng minh

Với mọi vector $\mathbf{x} \in \mathbb{F}_2^n$, ta khai triển từ vế phải của (3.5) và từ (3.6):

$$\begin{aligned} & 2^{-n} \sum_{\mathbf{a} \in \mathbb{F}_2^n} F_f(\mathbf{a}) \cdot (-1)^{\langle \mathbf{a}, \mathbf{x} \rangle} \\ &= 2^{-n} \sum_{\mathbf{a} \in \mathbb{F}_2^n} \left(\sum_{\mathbf{y} \in \mathbb{F}_2^n} f(\mathbf{y}) \cdot (-1)^{\langle \mathbf{a}, \mathbf{y} \rangle} \right) \cdot (-1)^{\langle \mathbf{a}, \mathbf{x} \rangle} \\ &= 2^{-n} \sum_{\mathbf{y} \in \mathbb{F}_2^n} f(\mathbf{y}) \sum_{\mathbf{a} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{a}, \mathbf{y} \oplus \mathbf{x} \rangle}. \end{aligned}$$

Theo (3.7), nếu ta coi $\mathbf{y} \oplus \mathbf{x} = \mathbf{z}$ thì

$$\sum_{\mathbf{a} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{a}, \mathbf{y} \oplus \mathbf{x} \rangle} = \begin{cases} 0, & \text{nếu } \mathbf{y} \oplus \mathbf{x} \neq \mathbf{0} \\ 2^n, & \text{nếu } \mathbf{y} \oplus \mathbf{x} = \mathbf{0}, \end{cases}$$

nghĩa là trong tổng trên thì chỉ có \mathbf{y} thỏa $\mathbf{y} \oplus \mathbf{x} = \mathbf{0}$ thì $f(\mathbf{y})$ không bị triệt tiêu. Nói cách khác là $\mathbf{y} = \mathbf{x}$ và do đó tổng trên còn lại $2^{-n}(f(\mathbf{x}) \cdot 2^n) = f(\mathbf{x})$ và ta có điều phải chứng minh.

❶ Example 2.19

Xét hàm boolean $f(x_1, x_2) = (1, 0, 0, 1)$.

Xét $\mathbf{a} = (0, 0)$. Ta có:

Với $\mathbf{x} = (0, 0)$,

$$f(\mathbf{x}) = 1, \langle \mathbf{a}, \mathbf{x} \rangle = 0 \cdot 0 + 0 \cdot 0 = 0.$$

Với $\mathbf{x} = (0, 1)$,

$$f(\mathbf{x}) = 0, \langle \mathbf{a}, \mathbf{x} \rangle = 0 \cdot 0 + 0 \cdot 1 = 0.$$

Với $\mathbf{x} = (1, 0)$,

$$f(\mathbf{x}) = 0, \langle \mathbf{a}, \mathbf{x} \rangle = 0 \cdot 1 + 0 \cdot 0 = 0.$$

Với $\mathbf{x} = (1, 1)$,

$$f(\mathbf{x}) = 1, \langle \mathbf{a}, \mathbf{x} \rangle = 0 \cdot 1 + 0 \cdot 1 = 0.$$

Ta suy ra

$$F_f(\mathbf{a}) = 1 \cdot (-1)^0 + 0 \cdot (-1)^0 + 0 \cdot (-1)^0 + 1 \cdot (-1)^0 = 2$$

khi $\mathbf{a} = (0, 0)$.

Tương tự, ta có các giá trị $F_f(\mathbf{a})$ sau:

- với $\mathbf{a} = (0, 1)$, $F_f(\mathbf{a}) = F_f(0, 1) = 0$;
- với $\mathbf{a} = (1, 0)$, $F_f(\mathbf{a}) = F_f(1, 0) = 0$;
- với $\mathbf{a} = (1, 1)$, $F_f(\mathbf{a}) = F_f(1, 1) = 2$.

Bây giờ ta đã có đủ $F_f(\mathbf{a})$ với $\mathbf{a} \in \mathbb{F}_2^n$ nên ta có thể kiểm chứng với mọi $\mathbf{x} \in \mathbb{F}_2^n$ thỏa công thức (3.5).

Biên đổi Walsh-Hadamard

Với mỗi hàm boolean $f(x_1, x_2, \dots, x_n) = f(\mathbf{x})$ ta định nghĩa một hàm tương ứng như sau:

$$f^*(\mathbf{x}) = (-1)^{f(\mathbf{x})}.$$

Ta định nghĩa $\langle \mathbf{a}, \mathbf{x} \rangle$ như trên.

Khi đó hàm $f^*(\mathbf{x})$ sẽ có dạng

$$f^*(\mathbf{x}) = 2^{-n} \sum_{\mathbf{a} \in \mathbb{F}_2^n} W_f(\mathbf{a})(-1)^{\langle \mathbf{a}, \mathbf{x} \rangle}, \quad (3.8)$$

trong đó

$$W_f(\mathbf{a}) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} (-1)^{f(\mathbf{x}) \oplus \langle \mathbf{a}, \mathbf{x} \rangle}. \quad (3.9)$$

Tập hợp $\{W_f(\mathbf{a}), \mathbf{a} \in \mathbb{F}_2^n\}$ được gọi là **phổ Walsh** (hay **spectre Walsh**) của hàm $f(\mathbf{x})$.

Các giá trị $W_f(\mathbf{a})$ được gọi là **hệ số Walsh**.

Chứng minh

Tương tự như trên, ta khai triển vế phải của (3.8) và thay (3.9) vào

$$\begin{aligned} & 2^{-n} \sum_{\mathbf{a} \in \mathbb{F}_2^n} W_f(\mathbf{a}) (-1)^{\langle \mathbf{a}, \mathbf{x} \rangle} \\ &= 2^{-n} \sum_{\mathbf{a} \in \mathbb{F}_2^n} \left(\sum_{\mathbf{y} \in \mathbb{F}_2^n} (-1)^{f(\mathbf{y}) \oplus \langle \mathbf{a}, \mathbf{y} \rangle} \right) (-1)^{\langle \mathbf{a}, \mathbf{x} \rangle} \\ &= 2^{-n} \sum_{\mathbf{y} \in \mathbb{F}_2^n} (-1)^{f(\mathbf{y})} \sum_{\mathbf{a} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{a}, \mathbf{y} \oplus \mathbf{x} \rangle}. \end{aligned}$$

Cũng từ (3.7), tương tự như trên ta có

$$\sum_{\mathbf{a} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{a}, \mathbf{y} \oplus \mathbf{x} \rangle} = \begin{cases} 0, & \text{nếu } \mathbf{y} \oplus \mathbf{x} \neq \mathbf{0} \\ 2^n, & \text{nếu } \mathbf{y} \oplus \mathbf{x} = \mathbf{0}. \end{cases}$$

Do đó trong các $\mathbf{y} \in \mathbb{F}_2^n$ thì chỉ có $\mathbf{y} = \mathbf{x}$ không bị triệt tiêu nên kết quả là

$$2^{-n} \cdot (-1)^{f(\mathbf{x})} \cdot 2^n = (-1)^{f(\mathbf{x})} = f^*(\mathbf{x}).$$

Các hệ số Walsh liên hệ với nhau bởi công thức

$$\sum_{\mathbf{a} \in \mathbb{F}_2^n} W_f(\mathbf{a}) W_f(\mathbf{a} \oplus \mathbf{d}) = \begin{cases} 2^{2n}, & \mathbf{d} = \mathbf{0} \\ 0, & \mathbf{d} \neq \mathbf{0}. \end{cases}$$

Trường hợp $\mathbf{d} = \mathbf{0}$ được gọi là **đẳng thức Parcel**:

$$\sum_{\mathbf{a} \in \mathbb{F}_2^n} (W_f(\mathbf{a}))^2 = 2^{2n}.$$

Tính chất của biến đổi Walsh-Hadamard

1. $W_f(\mathbf{0}) = 2^n - 2\text{wt}(f)$, và $W_f(\mathbf{0}) = 0$ khi và chỉ khi f là hàm cân bằng.
2. Nếu $g = \bar{f}$ thì $W_g(\mathbf{a}) = -W_f(\mathbf{a})$ với mọi $\mathbf{a} \in \mathbb{F}_2^n$.
3. Nếu $g(\mathbf{x}) = f(\mathbf{x} \oplus \mathbf{b})$ với vector \mathbf{b} nào đó, thì

$$\begin{aligned} W_g(\mathbf{a}) &= \sum_x (-1)^{f(\mathbf{x} \oplus \mathbf{b}) \oplus \langle \mathbf{a}, \mathbf{x} \rangle} \\ &= \sum_x (-1)^{f(\mathbf{x}) \oplus \langle \mathbf{a}, \mathbf{x} \oplus \mathbf{b} \rangle} \\ &= (-1)^{\langle \mathbf{a}, \mathbf{b} \rangle} W_f(\mathbf{a}). \end{aligned}$$

4. Đặt $f \in \mathcal{F}_n$, $\mathbf{b} \in \mathbb{F}_2^n$, $g(\mathbf{x}) = f(\mathbf{x}) \oplus \langle \mathbf{b}, \mathbf{x} \rangle$. Khi đó

$$\begin{aligned} W_g(\mathbf{a}) &= \sum_x (-1)^{f(\mathbf{x}) \oplus \langle \mathbf{b}, \mathbf{x} \rangle \oplus \langle \mathbf{a}, \mathbf{x} \rangle} \\ &= \sum_x (-1)^{f(\mathbf{x}) \oplus \langle \mathbf{b} \oplus \mathbf{a}, \mathbf{x} \rangle} \\ &= W_f(\mathbf{b} \oplus \mathbf{a}). \end{aligned}$$

5. Đặt $f(\mathbf{x}) = c$ là hằng số. Hàm $c \oplus \langle \mathbf{a}, \mathbf{x} \rangle$ là hàm tuyến tính với mọi $\mathbf{a} \neq \mathbf{0}$. Kết quả là $W_f(\mathbf{a}) = 0$ với mọi vector \mathbf{a} khác không, trừ khi

$$W_f(\mathbf{0}) = 2^n - 2\text{wt}(f) = \begin{cases} -2^n, & \text{nếu } c = 1, \\ 2^n, & \text{nếu } c = 0. \end{cases}$$

6. Đặt $f(\mathbf{x}) = c \oplus \langle \mathbf{b}, \mathbf{x} \rangle$ là hàm affine. Khi đó, theo tính chất 4 và 5 suy ra $W_f(\mathbf{a}) = 0$ với mọi $\mathbf{a} \neq \mathbf{0}$, và $W_f(\mathbf{b}) = (-1)^c \cdot 2^n$.
7. Đặt $f(\mathbf{x}, \mathbf{y}) = g(\mathbf{x}) \oplus h(\mathbf{y})$ với $g \in \mathcal{F}_n$ và $h \in \mathcal{F}_m$ và hai tập hợp biến \mathbf{x} và \mathbf{y} không giao nhau. Nói cách khác f là hàm boolean $n+m$ biến. Khi đó với mọi $\mathbf{a} \in \mathbb{F}_2^n$ và $\mathbf{b} \in \mathbb{F}_2^m$ thì

$$\begin{aligned} W_f(\mathbf{ab}) &= \sum_{\mathbf{x}, \mathbf{y}} (-1)^{g(\mathbf{x}) \oplus h(\mathbf{y}) \oplus \langle \mathbf{a}, \mathbf{x} \rangle \oplus \langle \mathbf{b}, \mathbf{y} \rangle} \\ &= \sum_{\mathbf{x}} (-1)^{g(\mathbf{x}) \oplus \langle \mathbf{a}, \mathbf{x} \rangle} \sum_{\mathbf{y}} (-1)^{h(\mathbf{y}) \oplus \langle \mathbf{b}, \mathbf{y} \rangle} \\ &= W_g(\mathbf{a}) \cdot W_h(\mathbf{b}). \end{aligned}$$

8. Đặt $f(x_1, \dots, x_n)$ giả phu thuộc vào biến x_i . Khi đó $W_f(\mathbf{a}) = 0$ với mọi vector \mathbf{a} mà $a_i = 1$.

Nói cách khác, nếu đặt $\mathbf{x}' = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$ và $\mathbf{a}' = (a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$ và để ý rằng $\langle \mathbf{a}, \mathbf{x} \rangle = \langle \mathbf{a}', \mathbf{x}' \rangle \oplus a_i x_i$. Khi đó nếu $a_i = 1$ thì

$$\begin{aligned} W_f(\mathbf{a}) &= \sum_{\mathbf{x}} (-1)^{f(\mathbf{x}) \oplus \langle \mathbf{a}, \mathbf{x} \rangle} \\ &= \sum_{\substack{\mathbf{x}, \\ x_i=0}} (-1)^{f(\mathbf{x}) \oplus \langle \mathbf{a}', \mathbf{x}' \rangle} + \sum_{\substack{\mathbf{x}, \\ x_i=1}} (-1)^{f(\mathbf{x}) \oplus \langle \mathbf{a}', \mathbf{x}' \rangle \oplus 1} = 0. \end{aligned}$$

Liên hệ giữa hệ số Fourier và hệ số Walsh

Remark 2.16

Quan hệ giữa hệ số Fourier và hệ số Walsh là biểu thức sau

$$W_f(\mathbf{a}) = 2^n \Delta(\mathbf{a}) - 2F_f(\mathbf{a})$$

với

$$\Delta(\mathbf{a}) = \begin{cases} 1, & \text{nếu } \mathbf{a} = \mathbf{0} \\ 0, & \text{nếu } \mathbf{a} \neq \mathbf{0}. \end{cases}$$

Chứng minh

Ta có

$$\begin{aligned} W_f(\mathbf{a}) + 2F_f(\mathbf{a}) &= \sum_{\mathbf{x} \in \mathbb{F}_2^n} (-1)^{f(\mathbf{x}) \oplus \langle \mathbf{a}, \mathbf{x} \rangle} + 2 \sum_{\mathbf{x} \in \mathbb{F}_2^n} f(\mathbf{x}) (-1)^{\langle \mathbf{a}, \mathbf{x} \rangle} \\ &= \sum_{\mathbf{x} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{a}, \mathbf{x} \rangle} [(-1)^{f(\mathbf{x})} + 2f(\mathbf{x})]. \end{aligned}$$

Để ý rằng, nếu $f(\mathbf{x}) = 0$ thì

$$(-1)^{f(\mathbf{x})} + 2f(\mathbf{x}) = (-1)^0 + 2 \cdot 0 = 1,$$

còn nếu $f(\mathbf{x}) = 1$ thì

$$(-1)^{f(\mathbf{x})} + 2f(\mathbf{x}) = (-1)^1 + 2 \cdot 1 = 1.$$

Nói cách khác biểu thức trên trở thành

$$W_f(\mathbf{a}) + 2F_f(\mathbf{a}) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{a}, \mathbf{x} \rangle}.$$

Từ (3.5) ta có

$$\sum_{\mathbf{x} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{a}, \mathbf{x} \rangle} = \begin{cases} 0, & \text{nếu } \mathbf{a} \neq \mathbf{0} \\ 2^n, & \text{nếu } \mathbf{a} = \mathbf{0}. \end{cases}$$

Như vậy nếu đặt $\Delta(\mathbf{a}) = \begin{cases} 1, \text{nếu } \mathbf{a} = \mathbf{0} \\ 0, \text{nếu } \mathbf{a} \neq \mathbf{0} \end{cases}$ thì ta có điều phải chứng minh.

❶ Remark 2.17

Khi $\mathbf{a} = \mathbf{0}$ thì với mọi $\mathbf{x} \in \mathbb{F}_2^n$ ta đều có $\langle \mathbf{a}, \mathbf{x} \rangle = 0$. Do đó

$$F_f(\mathbf{0}) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} f(\mathbf{x})(-1)^{\langle \mathbf{a}, \mathbf{x} \rangle} = \sum_{\mathbf{x} \in \mathbb{F}_2^n} f(\mathbf{x})(-1)^0 = \text{wt}(f),$$

suy ra

$$W_f(\mathbf{0}) = 2^n - 2\text{wt}(f) \Leftrightarrow \text{wt}(f) = 2^{n-1} - \frac{1}{2}W_f(\mathbf{0}). \quad (3.10)$$

Hàm Bent

Ta kí hiệu \mathcal{A} là tập hợp tất cả các hàm boolean affine với n biến, nghĩa là

$$\mathcal{A} = \{a_0 \oplus a_1x_1 \oplus \cdots \oplus a_nx_n \mid a_0, a_1, \dots, a_n \in \mathbb{F}_2\}.$$

❶ Definition 2.36 (Nonlinearity của hàm boolean)

Nonlinearity của hàm boolean f bất kì được định nghĩa là khoảng cách Hamming từ f tới \mathcal{A} , hay nói cách khác $N_f = d(f, \mathcal{A})$.

❶ Remark 2.18

Xét hàm f với n biến và phô Walsh tương ứng của hàm f là $\{W_f(\mathbf{a}), \mathbf{a} \in \mathbb{F}_2^n\}$. Khi đó

$$N_f = 2^{n-1} - \frac{1}{2} \max_{\mathbf{a} \in \mathbb{F}_2^n} |W_f(\mathbf{a})|. \quad (3.11)$$

➊ Remark 2.19

Từ đẳng thức Parcel ta có

$$\begin{aligned} 2^n \cdot \left(\max_{\mathbf{a} \in \mathbb{F}_2^n} (W_f(\mathbf{a}))^2 \right) &\geq \sum_{\mathbf{a} \in \mathbb{F}_2^n} (W_f(\mathbf{a}))^2 = 2^{2n} \\ \Leftrightarrow \max_{\mathbf{a} \in \mathbb{F}_2^n} (W_f(\mathbf{a}))^2 &\geq \frac{2^{2n}}{2^n} = 2^n \\ \Leftrightarrow \max_{\mathbf{a} \in \mathbb{F}_2^n} |W_f(\mathbf{a})| &\geq 2^{n/2}. \end{aligned}$$

Từ nhận xét trên và từ định nghĩa nonlinearity ở trên ta có

$$N_f \leq 2^{n-1} - \frac{1}{2} 2^{n/2}.$$

Hàm f khiếu dẫu bằng xảy ra được gọi là **hàm Bent**. Điều kiện cần và đủ để tồn tại hàm Bent f có n biến là khi $n = 2k$, tức là n chẵn.

Tính chất quan trọng của hàm Bent là với mọi vector \mathbf{a} thì

$$W_f(\mathbf{a}) = \pm 2^{n/2}.$$

➋ Example 2.20

Với $n = 2$, hàm $f(x_1, x_2) = x_1 \oplus x_1 x_2$ là hàm Bent.

Ta có thể tính toán và thấy rằng $W_f(0, 0) = 2$, $W_f(0, 1) = -2$, $W_f(1, 0) = 2$ và $W_f(1, 1) = 2$.

3.1.3 Block cipher

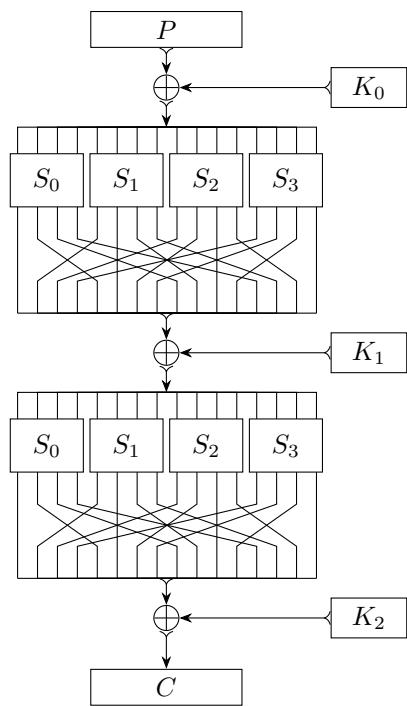
Chương này nói về các loại mã khối.

SP network

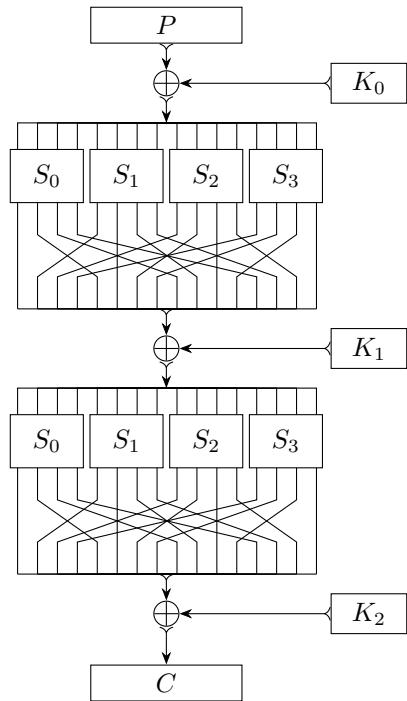
SP network

Mạng SP (SP network) là một cách xây dựng thuật toán mã hóa bên cạnh Feistel.

Mạng SP thường sử dụng một S-box làm nhiệm vụ xáo trộn và một P-box làm nhiệm vụ hoán vị. Hai nhiệm vụ có vẻ khá giống nhau, tuy nhiên S-box thường là một ánh xạ không tuyến tính và đôi khi làm giảm số bit đầu ra so với đầu vào (S-box của DES), còn P-box thường làm việc hoán vị vị trí các bit đầu vào (P-box của DES). Có thể thấy rằng P-box là ánh xạ tuyến tính (nhân với ma trận mà mỗi hàng và mỗi cột chứa đúng một số 1, còn lại là 0).



Hình 3.8: Ví dụ SP network



Hình 3.9: Ví dụ SP network

Tính chất của SP network

Các thuật toán được xây dựng dựa trên SP network thường gồm hai phần:

1. Phần không tuyến tính (S-box).
2. Phần tuyến tính.

Đối với mỗi phép biến đổi của SP network đều cần có biến đổi ngược của nó cho quá trình giải mã. Đây là điểm khác nhau cơ bản đối với mô hình Feistel.

Một số thuật toán sử dụng SP network

Một số thuật toán sử dụng SP network: AES (tiêu chuẩn mã hóa Hoa Kì, định nghĩa trong FIPS 197), Kuznyechik (tiêu chuẩn mã hóa Nga, định nghĩa trong GOST 34.12.2015, version 128 bit).

AES

AES biến đổi theo khối 128 bit, sử dụng mô hình mạng SPN.

Bốn phép biến đổi chính là Add Round Key, Substitute Bytes, Shift Rows và Mix Columns.

Quá trình giải mã sử dụng phép biến đổi ngược của bốn phép biến đổi trên là Inverse Sub Bytes, Inverse Shift Rows, Inverse Mix Columns. Đối với Add Round Key bản thân là phép XOR nên phép biến đổi ngược là chính nó.

AES hỗ trợ key với các kích thước: 128 bit, 192 bit và 256 bit.

Đối với kích thước khóa 128 bit, AES dùng hàm Expand Key để mở rộng khóa thành 44 words, mỗi word có 32 bits, với key 128 bit thành 11 cụm khóa con. Mỗi 4 words làm tham số cho một phép Add Round Key.

Mỗi block bốn rỗng byte p_0, p_1, \dots, p_{15} được tổ chức dưới dạng một ma trận 4×4 (gọi là **ma trận state**)

$$\begin{pmatrix} p_0 & p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 & p_7 \\ p_8 & p_9 & p_{10} & p_{11} \\ p_{12} & p_{13} & p_{14} & p_{15} \end{pmatrix} \longrightarrow \begin{pmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{pmatrix}$$

$$\begin{pmatrix} p_0 & p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 & p_7 \\ p_8 & p_9 & p_{10} & p_{11} \\ p_{12} & p_{13} & p_{14} & p_{15} \end{pmatrix} \longrightarrow \begin{pmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{pmatrix}$$

1. Các phép biến đổi Add Round Key, Substitute Bytes, Shift Rows, Mix Columns được thực hiện trên ma trận 4×4 này.
2. Các phép tính số học trong AES được thực hiện trong $GF(2^8)$ với đa thức tối giản là $f(x) = x^8 + x^4 + x^3 + x + 1$.

Substitute Bytes

Substitute Bytes

Ta sử dụng một bảng tra cứu 16×16 (S-box).

1. Diễn các số từ 0 tới 255 theo từng hàng.

2. Thay thế mỗi byte trong bảng bằng nghịch đảo trong $\text{GF}(2^8)$. Quy ước $(00)^{-1} = 00$.
3. Với mỗi byte trong bảng, ta kí hiệu 8 bit là $b_7b_6b_5b_4b_3b_2b_1b_0$. Thay thế mỗi b_i bằng b'_i như sau

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i,$$

với c_i là bit thứ i của số `0x63`.

Việc tính trên tương đương với phép nhân trên ma trận $\text{GF}(2)$ là $B' = XB + C$

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Ma trận X là ma trận khả nghịch, do đó phép biến đổi S-box là song ánh (one-to-one và onto mapping).

Dựa vào bảng S-box, Substitute Bytes thực hiện như sau: mỗi byte trong ma trận state S dưới dạng thập lục phân là xy sẽ được thay bằng giá trị ở hàng x và cột y của S-box.

Inverse Sub Bytes

Ta cần xây dựng bảng Inverse Sub Bytes (IS-box).

Việc xây dựng bảng này giống với bảng S-box ở bước 1 và 2. Tại bước 3:

$$b_i = b'_{(i+2) \bmod 8} \oplus b'_{(i+5) \bmod 8} \oplus b'_{(i+7) \bmod 8} \oplus d_i,$$

với d_i là bit thứ i của số `0x05`.

Ý nghĩa của Substitute Bytes

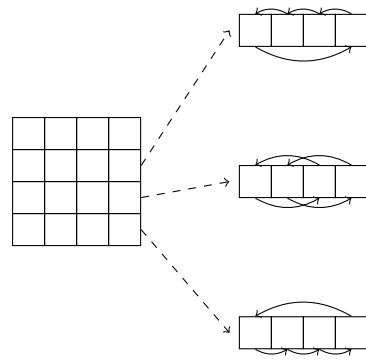
Bảng S-box dùng để chống lại known-plaintext và là bước duy nhất trong bốn bước không có quan hệ tuyến tính.

Shift Rows

Shift Rows

Trong Shift Rows, các dòng của ma trận state được biến đổi như sau:

1. Dòng thứ nhất giữ nguyên.
2. Dòng 2 dịch vòng trái 1 ô.
3. Dòng 3 dịch vòng trái 2 ô.
4. Dòng 4 dịch vòng trái 3 ô.



Inverse Shift Rows

Các dòng thứ 2, 3, 4 dịch phải tương ứng 1, 2, 3 ô.

Ý nghĩa

Xáo trộn các byte để tạo ra các cột cho Mix Columns.

Mix Columns

Mix Columns

Mix cols biến đổi từng cột của ma trận state một cách độc lập bằng phép nhân đa thức. Giả sử cột đầu tiên của ma trận state viết dưới dạng đa thức là

$$f(z) = s_{00}z^3 + s_{10}z^2 + s_{20}z + s_{30},$$

với $z \in \text{GF}(2^8)$.

Khi đó $f(z)$ sẽ được nhân với $a(z) = 3z^3 + z^2 + z + 2$, lưu ý rằng tất cả hệ số, phép cộng và nhân thực hiện trên $\text{GF}(2^8)$, và sau đó modulo cho $n(z) = z^4 + 1$.

Bốn byte hệ số của kết quả sẽ thay thế cho bốn byte tương ứng trong cột. Nếu viết dưới dạng ma trận, ta có

$$\begin{bmatrix} s'_{00} \\ s'_{10} \\ s'_{20} \\ s'_{30} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{00} \\ s_{10} \\ s_{20} \\ s_{30} \end{bmatrix}.$$

Lưu ý rằng các số 01, 02, 03 tuy viết dưới dạng thập phân nhưng khi tính toán phải ở dạng $\text{GF}(2^8)$. Việc sử dụng 1, 2, 3 giúp tăng tốc độ tính toán vì 1 và 2 chỉ cần phép dịch bit, còn 3 là XOR của 1 và 2.

Inverse Mix Columns

Lúc này ma trận nghịch đảo có dạng

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}.$$

Ý nghĩa

Mỗi cột mới chỉ phụ thuộc cột ban đầu. Cùng với sự kết hợp Shift Rows sau một vài vòng biến đổi (cụ thể là 2, các bạn có thể thử chứng minh), 128 bit kết quả phụ thuộc vào tất cả 128 bit ban đầu. Từ đó tạo ra tính khuếch tán (diffusion).

Add Round Key

Add Round Key

128 bit của ma trận state được XOR với 128 bit của khóa con từng vòng (4 dword 32 bit). Phép biến đổi ngược của Add Round Key là chính nó.

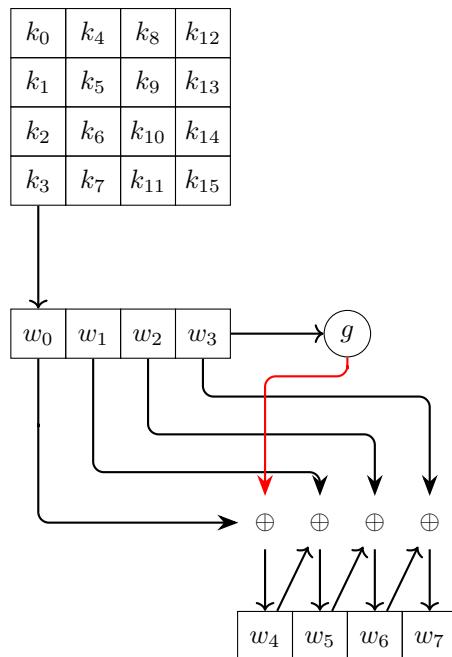
Ý nghĩa

Sự kết hợp với khóa tạo ra tính làm rối (confusion).

Expand Key

Expand Key

Đầu vào của thao tác Expand Key là 16 bytes (4 words) của khóa, sinh ra một mảng 44 words (176 bytes) sử dụng cho 11 vòng AES, mỗi vòng 4 words.



Từ 4 word đầu vào $w_0 w_1 w_2 w_3$, lần lặp đầu sinh ra $w_4 w_5 w_6 w_7$, lần lặp thứ hai sinh ra $w_8 w_9 w_{10} w_{11}$, ...

Algorithm 3.1

1. if $i \bmod 4 = 0$
 1. $g \leftarrow SubWord(RotWord(w_{i-1})) \oplus Rcon[i/4]$

```

2.  $w_i = w_{i-4} \oplus g$ 
2. else
    1.  $w_i = w_{i-4} \oplus w_{i-1}$ 
3. endif

```

Trong đó:

1. *RotWord* dịch vòng trái 1 bit, nghĩa là $b_0 b_1 b_2$ trở thành $b_1 b_2 b_0$.
2. *SubWord* thay mỗi byte trong word bằng bảng S-box.
3. *Rcon* là một mảng hằng số gồm 10 words tương ứng với 10 vòng AES. 4 bytes của một phần tử $Rcon[j]$ là $RC[j], 0, 0, 0$ với $RC[j]$ là mảng 10 bytes như sau

j	1	2	3	4	5	6	7	8	9	10
$RC[j]$	1	2	4	8	10	20	40	80	18	36

Ý nghĩa của Expand Key

Dùng để chống lại known-plaintext (giống Sub Bytes dùng S-box). Đặc điểm của Expand Key gồm:

1. Biết một số bit của khóa hay khóa con không thể tính được các bit còn lại.
2. KHÔNG THỂ tính ngược.
3. Khuếch tán: mỗi bit của khóa chính tác động lên tất cả khóa con.

Kết luận

Mã hóa AES đơn giản và có thể chạy trên các chip 8 bit.

AES cung cấp ba biến thể cho độ dài khóa là:

- 128 bits: 44 words 4 bytes cho 10 vòng (11 lần ARK);
- 192 bits: 52 words 4 bytes cho 12 vòng (13 lần ARK);
- 256 bits: 60 words 4 bytes cho 14 vòng (15 lần ARK).

Về phép Mix Columns

Sau đây mình sẽ nói về việc phép nhân trên đa thức có hệ số trong $GF(2^8)$ lại tương đương với phép nhân ma trận trong Mix Columns ở trên.

Giả sử ma trận trạng thái trước khi bước vào phép tính Mix Column của AES là

$$\begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ c_8 & c_9 & c_{10} & c_{11} \\ c_{12} & c_{13} & c_{14} & c_{15} \end{pmatrix}.$$

Phép tính Mix Column lấy mỗi cột của ma trận trạng thái trên làm tham số cho đa thức với hệ số trong $GF(2^8)$ và nhân với đa thức $c(z) = 2 + z + z^2 + 3z^3$ rồi modulo cho $z^4 + 1$.

Giả sử với cột đầu tiên, ta viết hệ số theo thứ tự bậc tăng dần $d(z) = c_0 + c_4z + c_8z^2 + c_{12}z^3$.

Tính trong GF(2⁸):

$$\begin{aligned}
 c(z) \cdot d(z) &= (2 + z + z^2 + 3z^3) \cdot (c_0 + c_4z + c_8z^2 + c_{12}z^3) \\
 &= 2c_0 + 2c_4z + 2c_8z^2 + 2c_{12}z^3 \\
 &\quad + c_0z + c_4z^2 + c_8z^3 + c_{12}z^4 \\
 &\quad + c_0z^2 + c_4z^3 + c_8z^4 + c_{12}z^5 \\
 &\quad + 3c_0z^3 + 3c_4z^4 + 3c_8z^5 + 3c_{12}z^6 \\
 &= 2c_0 + (2c_4 + c_0)z + (2c_8 + c_4 + c_0)z^2 \\
 &\quad + (2c_{12} + c_8 + c_4 + 3c_0)z^3 + (c_{12} + c_8 + 3c_4)z^4 \\
 &\quad + (c_{12} + 3c_8)z^5 + 3c_{12}z^6.
 \end{aligned}$$

Trong GF(2⁸) thì mọi phần tử đều có tính chất $2x^n = 0$, tương đương với $x^n = -x^n$. Do đó

$$\begin{aligned}
 z^6 \pmod{z^4 + 1} &\equiv -z^2 \equiv z^2 \\
 z^5 \pmod{z^4 + 1} &\equiv -z \equiv z \\
 z^4 \pmod{z^4 + 1} &\equiv -1 \equiv 1.
 \end{aligned}$$

Suy ra

$$\begin{aligned}
 c(z) \cdot d(z) &= 2c_0 + (2c_4 + c_0)z + (2c_8 + c_4 + c_0)z^2 \\
 &\quad + (2c_{12} + c_8 + c_4 + 3c_0)z^3 + (c_{12} + c_8 + 3c_4) \\
 &\quad + (c_{12} + 3c_8)z + 3c_{12}z^2 \\
 &= (c_{12} + c_8 + 3c_4 + 2c_0) + (c_{12} + 3c_8 + 2c_4 + c_0)z \\
 &\quad + (3c_{12} + 2c_8 + c_4 + c_0)z^2 + (2c_{12} + c_8 + c_4 + 3c_0)z^3.
 \end{aligned}$$

Như vậy xét hệ số lần lượt trước 1, z , z^2 và z^3 thì tương đương với phép nhân ma trận

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_4 \\ c_8 \\ c_{12} \end{pmatrix}.$$

Đây chính là kết quả cần tìm.

Kuznyechik

Kuznyechik là một thuật toán mã hóa khối, đối xứng như AES. Kuznyechik tiếng Nga là Кузнечик, có nghĩa là chảo nấu. Tuy nhiên trong văn bản quốc tế thì chúng ta giữ nguyên tên gọi là hệ mã Kuznyechik.

Mã khối Kuznyechik biến đổi trên khối 128 bit, độ dài khóa là 256 bit, biến đổi trên 9 vòng. Kuznyechik sử dụng mô hình SPN tương tự như AES, trở thành chuẩn mã hóa của Nga và được định nghĩa trong GOST R 34.12-2015.

Một điểm đặc biệt là quá trình biến đổi qua các vòng sử dụng mạng SPN, tuy nhiên thuật toán sinh khóa con cho các vòng sử dụng mô hình Feistel.

Phần này tham khảo chính từ [20].

Mã hóa

Gọi k_i là khóa con của vòng thứ i , $i = 0, 1, \dots, 9$. Ta có các động tác biến đổi sau:

Phép biến đổi X

Hàm $X : \mathbb{F}_2^{128} \rightarrow \mathbb{F}_2^{128}$ biến đổi trên block 128 bits.

Ta chia block đầu vào thành 16 cụm 8 bit, kí hiệu

$$a = a_0 \| a_1 \| \dots \| a_{14} \| a_{15}$$

với ký tự $\|$ chỉ việc nối các chuỗi bit (concatenate). Tương tự k_i cũng được chia thành 16 cụm 8 bits. Khi đó,

$$X(k_i, a) = k_{i,0} \oplus a_0 \| k_{i,1} \oplus a_1 \| \dots \| k_{i,14} \oplus a_{14} \| k_{i,15} \oplus a_{15}. \quad (3.12)$$

Nói cách khác, chúng ta xor 128 bit của khối đầu vào và 128 bit của khóa con k_i .

Phép biến đổi S

Hàm $S : \mathbb{F}_2^{128} \rightarrow \mathbb{F}_2^{128}$.

Block đầu vào tiếp tục được chia thành 16 cụm 8 bits. Mỗi cụm sẽ đi qua một bảng tra cứu SBox (gọi là bảng π). Sau đó ta nối các kết quả với nhau.

$$S(a) = \pi(a_0) \| \pi(a_1) \| \dots \| \pi(a_{14}) \| \pi(a_{15}). \quad (3.13)$$

Bảng π được định nghĩa sẵn và không tuyến tính, nên đây là bước không tuyến tính của thuật toán.

Phép biến đổi L

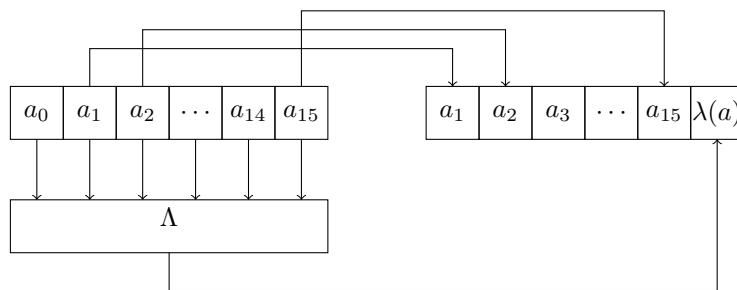
Hàm $L : \mathbb{F}_2^{128} \rightarrow \mathbb{F}_2^{128}$.

Block đầu vào vẫn được chia thành 16 cụm 8 bit. Tuy nhiên ở đây mỗi cụm 8 bit biểu diễn một đa thức trong trường $GF(2^8)$ với đa thức tối giản là $g(x) = x^8 + x^7 + x^6 + x + 1$. Những phép tính cộng và nhân sau đây cũng được thực hiện trên trường $GF(2^8)$ này.

$$\begin{aligned} \lambda(a) = & 148a_{15} + 32a_{14} + 133a_{13} + 16a_{12} \\ & + 194a_{11} + 192a_{10} + a_9 + 251a_8 \\ & + a_7 + 192a_6 + 194a_5 + 16a_4 \\ & + 133a_3 + 32a_2 + 148a_1 + a_0. \end{aligned} \quad (3.14)$$

Tiếp theo, ta định nghĩa hàm $\Lambda : \mathbb{F}_2^{128} \rightarrow \mathbb{F}_2^{128}$ như sau

$$a = a_0 \| a_1 \| \dots \| a_{14} \| a_{15} \rightarrow a_1 \| a_2 \| \dots \| a_{15} \| \lambda(a) = \Lambda(a).$$



Hình 3.10: Hàm λ

Lưu ý rằng sau khi tính toán trên hàm λ , đa thức trên $GF(2^8)$ được chuyển trở lại thành cụm 8 bit sau đó mới nối vào dãy a_1, a_2, \dots, a_{15} như mô tả ở [Hình 3.10](#).

Cuối cùng, hàm L ban đầu là thực hiện hàm Λ 16 lần.

$$L(a) = \underbrace{\Lambda(\dots \Lambda(a) \dots)}_{16 \text{ lần}}.$$

Như vậy, phép biến đổi trên vòng thứ i với khóa con k_i là

$$R(k_i, a) = L(S(X(a))) \quad (3.15)$$

với $i = 0, 1, \dots, 8$.

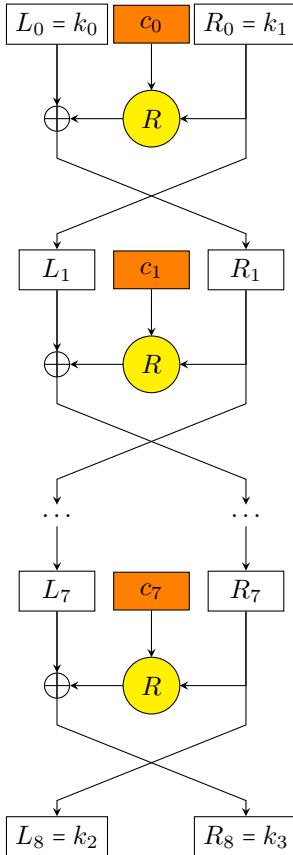
Ở vòng thứ 10 ta XOR với khóa con k_9 nữa: $X(k_9, a)$.

Thuật toán sinh khóa con

Để sinh khóa con cho 10 lần XOR, thuật toán Kuznyechik dùng mô hình Feistel. Đầu tiên ta định nghĩa hàm $F(c, a)$. Với c bất kì thuộc \mathbb{F}_2^{128} và $a = a_0 \| a_1$ thuộc \mathbb{F}_2^{256} . Hàm $F(c, a)$ biến phần tử thuộc $\mathbb{F}_2^{128} \times \mathbb{F}_2^{256}$ thành phần tử thuộc \mathbb{F}_2^{256} bằng đẳng thức

$$F(c, a) = a_1 \| a_0 \oplus R(c, a_1)$$

với hàm R được định nghĩa ở phương trình (3.15).



Hình 3.11: Biến đổi từ khóa $k_0 \| k_1$ thành $k_2 \| k_3$

Với khóa đầu vào là $k \in \mathbb{F}_2^{256}$, mình kí hiệu ở dạng ghép hai chuỗi 128 bits $k = k_0 \| k_1$ với $k_0, k_1 \in \mathbb{F}_2^{128}$ là những khóa con mở đầu. Khi đó các khóa con cho 10 phép XOR là k_0, k_1, \dots, k_9 .

Thuật toán sinh khóa con được sử dụng như sau

$$k_{2i+2} \| k_{2i+3} = F(c_{8i+7}, \dots, F(c_{8i}, k_{2i} \| k_{2i+1})),$$

với $i = 0, 1, 2, 3$. Thuật toán có thể mô tả ở [Hình 3.11](#).

Theo đó, các số c_0, c_1, \dots, c_7 được sử dụng để sinh khóa $k_2 \| k_3$ từ $k_0 \| k_1$. Tương tự, c_8, c_9, \dots, c_{15} được dùng để sinh khóa $k_4 \| k_5$ từ khóa $k_2 \| k_3$. Các số c_0, c_1, \dots, c_{31} được định nghĩa trong tiêu chuẩn.

So sánh Kuznyechik với AES

Điểm giống nhau là cả hai thuật toán đều có phần tuyến tính và phần không tuyến tính. Về phần tuyến tính, đối với AES là các động tác Shift Rows, Mix Columns và Add Round Key, còn đối với Kuznyechik là hàm X và L bên trên. Về phần không tuyến tính đều là việc sử dụng một bảng tra cứu SBox của riêng thuật toán đó.

Điểm khác nhau đầu tiên là cách xây dựng ma trận tính toán. Nếu ta xem xét Shift Rows và Mix Columns dưới dạng phép nhân ma trận trên $\text{GF}(2^8)$ thì ta thấy rằng ma trận chứa nhiều số 0 nhất có thể. Điều này giúp tăng tốc độ tính toán. Về phần Kuznyechik, phép tính ở hàm λ cũng thực hiện trên $\text{GF}(2^8)$ nhưng không chứa bất kì số 0 nào. Điều này làm tăng độ phức tạp tính toán nhưng cũng làm tăng tính an toàn.

Điểm khác nhau tiếp theo là việc sinh khóa con. AES sử dụng thuật toán sinh khóa con từng vòng từ toàn bộ 256 bit ban đầu. Trong khi Kuznyechik sử dụng mô hình Feistel, theo đó với 256 bit ban đầu được sử dụng cho hai vòng đầu, cứ mỗi hai khóa con sẽ sinh ra hai khóa con tiếp theo. Như vậy thuật toán sinh khóa con ít phức tạp hơn AES (Kuznyechik cần 5 lần sinh khóa còn AES 14 lần).

PRESENT

PRESENT là mã khôi lightweight, nghĩa là được thiết kế cho các thiết bị có bộ xử lý nhỏ gọn, tập trung vào độ dài khóa không quá lớn (tối đa 128 bits) nhưng vẫn đảm bảo an toàn và có thể thực hiện trên các thiết bị có bộ xử lý 8 bit (ví dụ embedded systems).

Phần này minh tham khảo trong tài liệu về xây dựng PRESENT cipher ở [\[21\]](#).

Mô tả thuật toán

PRESENT gồm 32 động tác addRoundKey, duyệt qua 31 vòng.

Algorithm 3.2

1. generateRoundKeys()
2. **for** $i = 1$ **to** 31 **do**

 - 1. addRoundKey(STATE, K_i)
 - 2. sBoxLayer(STATE)
 - 3. pLayer(STATE)

3. end for
4. addRoundKey(STATE, K_{32})

PRESENT sử dụng mạng SP để mã hóa. Kích thước khôi là 64 bits. PRESENT hỗ trợ hai kích thước khóa là 80 và 128 bits. Trong [\[21\]](#), các tác giả khuyến nghị sử dụng khóa 80 bits cho phù hợp với cài đặt phần

cứng. Điều này dễ hiểu vì kích thước khóa lớn thì vấn đề lưu trữ trên bộ nhớ và tính toán sẽ chiếm nhiều không gian, ngoài ra tốc độ tính toán sẽ bị giảm vì khả năng của CPU không lớn.

addRoundKey

Giả sử round key ở vòng thứ i là

$$K_i = \kappa_{63}^i \dots \kappa_0^i,$$

với $1 \leq i \leq 32$, cùng với STATE hiện tại là $b_{63} \dots b_0$, khi đó addRoundKey chính là phép XOR bit

$$b_j \rightarrow b_j \kappa_j^i$$

với $0 \leq j \leq 63$.

sBoxLayer

PRESENT sử dụng S-box biến đổi 4 bits thành 4 bits, nghĩa là $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$. S-box được cho bởi bảng sau.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

STATE được biểu diễn ở dạng $b_{63} \dots b_0$ sẽ được chia thành 16 đoạn, mỗi đoạn 4 bits và từng đoạn sẽ đi qua S-box. Sau đó các đoạn được nối lại theo thứ tự thành khối có 64 bits mới.

pLayer

Hoán vị bit (bit permutation) được thực hiện theo bảng sau với bit thứ i của STATE sẽ trở thành bit thứ $P(i)$ của kết quả.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51

i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55

i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59

i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Thực ra, việc hoán vị bit tương ứng với một phép nhân ma trận. Ma trận này có tính chất là trên mỗi hàng và mỗi cột có duy nhất một phần tử 1, các phần tử còn lại bằng 0. Ví dụ với 4 bit,

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} b_2 \\ b_3 \\ b_0 \\ b_1 \end{pmatrix}.$$

Phép nhân ma trận tương ứng với bảng hoán vị

i	0	1	2	3
$P(i)$	1	0	3	2

key schedule

Ở phần này chúng ta xem xét phiên bản khóa 80 bits của PRESENT.

Giả sử khóa đầu vào được lưu trong thanh ghi K và được biểu diễn ở dạng $k_{79}k_{78}\dots k_0$.

Tại vòng thứ i , khóa K_i là 64 bits ngoài cùng của K , nghĩa là ở vòng thứ i ta có

$$K_i = \kappa_{63} \dots \kappa_0 = k_{79}k_{78}\dots k_{16}.$$

Sau khi trích xuất round key K_i , thanh ghi K sẽ được cập nhật theo các bước sau

1. $[k_{79}k_{78}\dots k_1k_0] = [k_{18}k_{17}\dots k_{20}k_{19}]$ - dịch vòng trái 61 vị trí.
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$ - dùng S-box biến đổi 4 bits thành 4 bits.
3. Gọi **round_counter** là i ở dạng 5 bits, khi đó

$$[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus \text{round_counter}.$$

Mô hình Feistel

Mô tả mô hình Feistel

Trong mô hình Feistel, mỗi block đầu vào được chia thành hai nửa trái phải có số bit bằng nhau.

Một cách hình thức, giả sử plaintext đầu vào là P có $2n$ bit, thì P được chia thành nửa trái L_0 và R_0 , mỗi nửa có n bits.

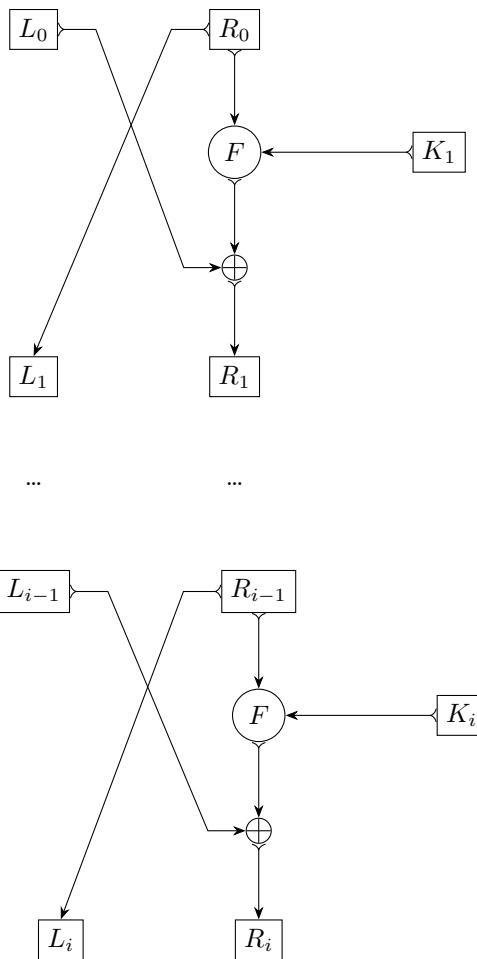
Mỗi hệ mã Feistel có một hàm $F(R_{i-1}, K_i)$ nhận đầu vào là nửa phải ở vòng thứ $(i-1)$ và khóa K_i ở vòng thứ i . Hàm F được gọi là **round function**.

Ở mỗi vòng $i = 1, 2, \dots$, hai nửa trái phải mới sẽ được tính theo công thức

$$L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus F(R_{i-1}, K_i),$$

trong đó:

- L_{i-1} và R_{i-1} là nửa trái và nửa phải ở vòng thứ $(i-1)$;
- L_i và R_i là nửa trái và nửa phải ở vòng thứ i ;
- K_i là khóa con được dùng ở vòng i ;
- F là round function.



Hình 3.12: Mô hình Feistel

Nếu thuật toán kết thúc sau r vòng (với r khóa con) thì ciphertext tương ứng sẽ là $C = L_r \| R_r$.

Tính chất của mô hình Feistel

Mô hình Feistel không đòi hỏi round function F phải khả nghịch. Giả sử ciphertext là $C = L_r \| R_r$. Khi đó dựa trên công thức mã hóa ở trên thì có thể dễ dàng suy ra công thức giải mã là

$$R_{i-1} = L_i, \quad L_{i-1} = R_i \oplus F(R_i, K_i),$$

với $i = r - 1, \dots, 0$.

Plaintext nhận được sẽ là $P = L_0 \| R_0$.

Một số thuật toán sử dụng mô hình Feistel

Mô hình Feistel chuẩn

Mô hình Feistel chuẩn là mô hình ở trên. Các thuật toán sử dụng mô hình Feistel chuẩn có thể kể đến là: DES, Magma (tiêu chuẩn mã hóa Nga, định nghĩa trong GOST 34.12.2015, version 64 bit), CAST-128 (tiêu chuẩn mã hóa Canada).

Mô hình Feistel tổng quát

Mô hình Feistel tổng quát (generalized Feistel model) có nhiều biến thể. Trong đó khối đầu vào không được chia thành hai nửa trái phải mà có thể được chia thành bốn phần $P = X_0\|X_1\|X_2\|X_3$, mỗi phần có số bit bằng nhau.

Sau đó, ở mỗi vòng, bốn phần ở vòng i nhận được từ bốn phần ở vòng $(i - 1)$, thông thường là từ việc giữ lại ba phần (nhưng ở vị trí khác) và XOR phần còn lại với một round function lấy tham số là ba phần kia cùng với khóa con ở vòng i . Ví dụ

$$\begin{aligned} X_0^{(i)} &= X_1^{(i-1)}, \\ X_1^{(i)} &= X_2^{(i-1)}, \\ X_2^{(i)} &= X_3^{(i-1)}, \\ X_3^{(i)} &= X_0^{(i-1)} \oplus F(X_1^{(i-1)}, X_2^{(i-1)}, X_3^{(i-1)}, K_i), \end{aligned}$$

trong đó:

- $X_0^{(i)}$, $X_1^{(i)}$, $X_2^{(i)}$, $X_3^{(i)}$ là bốn phần ở vòng thứ i ;
- $X_0^{(i-1)}$, $X_1^{(i-1)}$, $X_2^{(i-1)}$, $X_3^{(i-1)}$ là bốn phần ở vòng thứ $(i - 1)$;
- K_i là khóa con ở vòng thứ i ;
- F là round function nhận bốn đầu vào gồm ba phần ở vòng thứ $(i - 1)$ và khóa con ở vòng thứ i .

Lưu ý rằng trên đây chỉ trình bày một dạng biến thể của mô hình Feistel tổng quát.

Một số thuật toán sử dụng mô hình Feistel tổng quát: SM4 (tiêu chuẩn mã hóa mạng không dây của Trung Quốc).

Magma

Hệ mật mã Magma được chính phủ Xô Viết chọn làm chuẩn mã hóa. Cũng giống như hệ mật mã DES, Magma sử dụng mô hình Feistel cho các vòng mã hóa, được định nghĩa trong GOST 34.12-2015, trước đây gọi là GOST 28147-89.

Phần này tham khảo chính từ [22].

Độ dài khóa là 256 bits. Độ dài khối là 64 bits. Magma biến đổi trên 32 vòng để cho ra ciphertext.

Magma thực hiện biến đổi trên 32 vòng Feistel. Khối đầu vào 64 bits được chia thành hai nửa trái phải, mỗi nửa 32 bits.

Key schedule

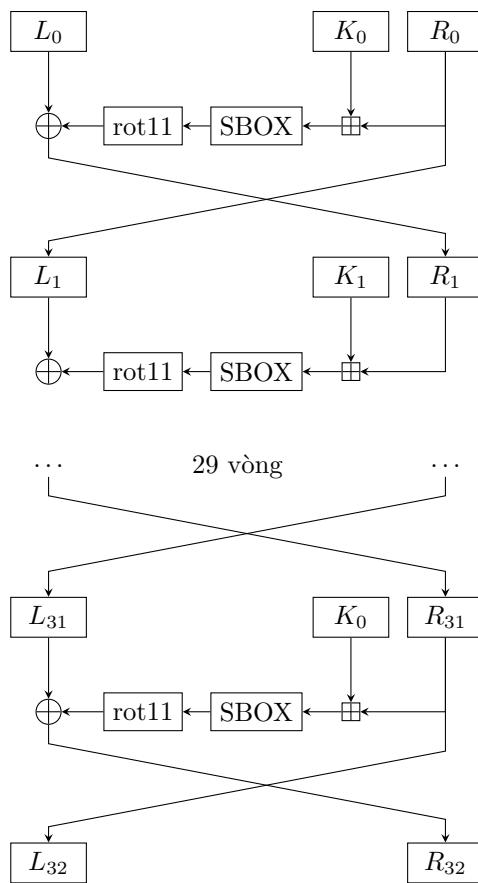
Khóa 256 bit được chia thành 8 cụm khóa con, mỗi khóa con 32 bit.

Nếu ta ký hiệu 256 bit của khóa là $k_0k_1\dots k_{254}k_{255}$ thì ta có các khóa con là

$$\underbrace{k_0\dots k_{31}}_{K_0}\underbrace{k_{32}\dots k_{63}}_{K_1}\dots\underbrace{k_{224}\dots k_{255}}_{K_7}.$$

Từ vòng 1 tới 24 sử dụng lần lượt các khóa K_0 , K_1 , ..., K_7 rồi lặp lại thứ tự đó.

Từ vòng 25 tới 32 sử dụng theo thứ tự ngược lại, từ K_7 , K_6 , ..., K_0 .



Hình 3.13: Mô hình mă khối Magma

Round function

Như ta đã biết, trong mô hình Feistel, mỗi khối plaintext được chia thành hai nửa trái phải (L_0, R_0). Sau đó ở mỗi vòng biến đổi thì

$$L_{i+1} = R_i, \quad R_{i+1} = L_i \oplus f(R_i, K_i),$$

với $i = 0, 1, \dots$ và K_i là khóa con ở vòng i .

Hàm f của Magma khá đơn giản, bao gồm ba động tác là cộng modulo 2^{32} , SBox và xoay 11 bits.

Đối với việc cộng modulo 2^{32} , ta xem block R_i và K_i như những số nguyên 32 bit, cộng chúng lại và modulo 2^{32} , nghĩa là $(R_i + K_i) \bmod 2^{32}$.

Đặt $T_i = (R_i + K_i) \bmod 2^{32}$. Như vậy T_i có 32 bits. Ta chia 32 bits này thành 8 cụm 4 bits. Ứng với mỗi cụm 4 bits chúng ta cho qua một hoán vị. Như vậy cần 8 hoán vị (SBox). SBox được sử dụng chung cho tất cả vòng.

Theo wiki thì SBox có thể bí mật. Tuy nhiên việc mã hóa và giải mã cần sử dụng SBox giống nhau. Do đó thiết bị mã hóa và giải mã có cùng cơ chế pseudo-random để sinh ra SBox giống nhau.

SBox được quy định theo tiêu chuẩn chính phủ Nga là

```
sbox = [  
    0xC, 0x4, 0x6, 0x2, 0xA, 0x5, 0xB, 0x9, 0xE, 0x8, 0xD, 0x7, 0x0, 0x3, 0xF, 0x1],  
    (continues on next page)
```

(continued from previous page)

```
[0x6, 0x8, 0x2, 0x3, 0x9, 0xA, 0x5, 0xC, 0x1, 0xE, 0x4, 0x7, 0xB, 0xD, 0x0, 0xF],  

[0xB, 0x3, 0x5, 0x8, 0x2, 0xF, 0xA, 0xD, 0xE, 0x1, 0x7, 0x4, 0xC, 0x9, 0x6, 0x0],  

[0xC, 0x8, 0x2, 0x1, 0xD, 0x4, 0xF, 0x6, 0x7, 0x0, 0xA, 0x5, 0x3, 0xE, 0x9, 0xB],  

[0x7, 0xF, 0x5, 0xA, 0x8, 0x1, 0x6, 0xD, 0x0, 0x9, 0x3, 0xE, 0xB, 0x4, 0x2, 0xC],  

[0x5, 0xD, 0xF, 0x6, 0x9, 0x2, 0xC, 0xA, 0xB, 0x7, 0x8, 0x1, 0x4, 0x3, 0xE, 0x0],  

[0x8, 0xE, 0x2, 0x5, 0x6, 0x9, 0x1, 0xC, 0xF, 0x4, 0xB, 0x0, 0xD, 0xA, 0x3, 0x7],  

[0x1, 0x7, 0xE, 0xD, 0x0, 0x5, 0x8, 0x3, 0x4, 0xF, 0xA, 0x6, 0x9, 0xC, 0xB, 0x2],  

]
```

Nếu T_i được viết dưới dạng 32 bits là $t_{31}t_{30}\dots t_1t_0$ thì SBox tương ứng của nó là

$$\underbrace{t_{31}\dots t_{28}}_{S_7} \underbrace{t_{27}\dots t_{24}}_{S_6} \underbrace{\dots t_7\dots t_4}_{S_1} \underbrace{t_3\dots t_0}_{S_0}$$

Nói cách khác, $t_{4i+3}t_{4i+2}t_{4i+1}t_{4i}$ dùng S_{7-i} với $i = 0, 1, 2, \dots, 7$.

Cuối cùng, việc xoay trái 11 bits (rot11) chỉ đơn giản là đưa 11 bit đầu về cuối và đưa 21 bit cuối lên đầu.

Để giải mã ta vẫn sử dụng round function như lúc mã hóa, chỉ cần viết với thứ tự ngược lại là được. Như vậy

$$R_i = L_{i+1}, \quad L_i = R_{i+1} \oplus f(L_{i+1}, K_i)$$

do $R_i = L_{i+1}$. Lưu ý rằng khóa con lúc này là 0 tới 7 cho 8 vòng đầu, và 7 về 0 (rồi lặp lại) cho 24 vòng sau.

SMS4

SM4 (trước đây là SMS4) là thuật toán mã hóa khối của Trung Quốc để bảo vệ mạng không dây. Thuật toán được phát triển vào năm 2006.

Phần này mình dịch từ [17] hoặc bản dịch tiếng Anh là [23].

Input, output và độ dài khóa của SMS4 đều là 128 bit. Thuật toán gồm 32 vòng.

Plaintext 128 bit sẽ được chia thành 4 phần độ dài 32 bit (gọi là **word**). Giả sử plaintext là P có 128 bit sẽ được biểu diễn thành

$$P = (X^0, X^1, X^2, X^3).$$

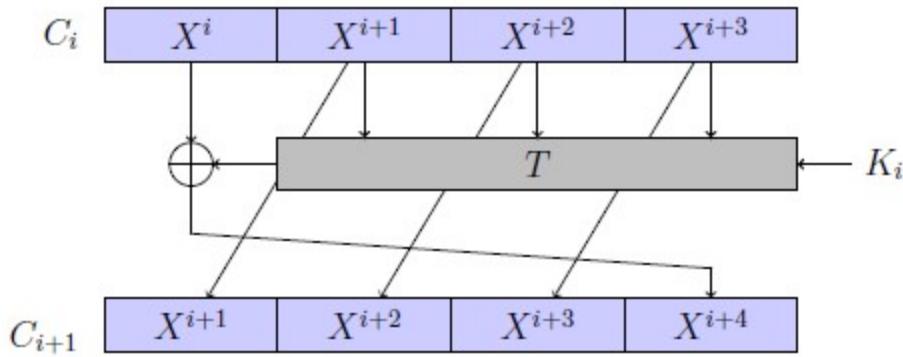
Từ khóa ban đầu là K , thuật toán sinh khóa sinh ra các khóa K_0, \dots, K_{31} . Mỗi khóa con độ dài 32 bit.

SMS4 sử dụng **mô hình Feistel tổng quát** (hay **generalized Feistel model**).

Mã hóa

Ở vòng thứ i , với $i = 0, 1, \dots, 31$, ta tính word mới X^{i+4} theo công thức

$$X^{i+4} = X^i \oplus T(X^{i+1} \oplus X^{i+2} \oplus X^{i+3} \oplus K_i).$$



Hình 3.14: Một vòng SMS4

Ciphertext sẽ là viết ngược của bốn word ở vòng cuối cùng, nói cách khác là

$$C = (X^{35}, X^{34}, X^{33}, X^{32}).$$

Round function

Biến đổi T gồm hai phần là hoán vị (không tuyến tính) τ và hoán vị (tuyến tính) L . Nói cách khác là $T(\cdot) = L(\tau(\cdot))$.

Hoán vị không tuyến tính có dạng

$$\tau(X) = \tau(a_0, a_1, a_2, a_3) = (\text{SBox}(a_0), \text{SBox}(a_1), \text{SBox}(a_2), \text{SBox}(a_3)),$$

với a_i là các byte của X , nghĩa là X có 32 bits sẽ được chia thành bốn khối độ dài 8 bits là a_0, a_1, a_2 và a_3 .

Bảng S-box có thể xem ở một trong hai tài liệu trên.

Hoán vị tuyến tính có dạng

$$L(X) = X \oplus (X \lll 2) \oplus (X \lll 10) \oplus (X \lll 18) \oplus (X \lll 24),$$

trong đó \lll là phép dịch vòng bit sang trái.

Thuật toán sinh khóa con

Khóa K ban đầu (128 bit) được chia thành bốn word

$$K = (MK_0, MK_1, MK_2, MK_3).$$

Các khóa con K_0, K_1, \dots, K_{31} được sinh ra theo quy tắc sau. Với mỗi $i = 0, 1, \dots, 31$ thì

$$K_i = HK_{i+4} = HK_i \oplus T'(HK_{i+1} \oplus HK_{i+2} \oplus HK_{i+3} \oplus CK_i),$$

với $HK_i = MK_i \oplus FK_i$ với $i = 0, 1, 2, 3$. Trong đó FK là một mảng được định nghĩa sẵn

$$\begin{aligned} FK_0 &= (\text{a3b1bac6}), FK_1 = (\text{56aa3350}), \\ FK_2 &= (\text{677d9197}), FK_3 = (\text{b27022dc}). \end{aligned}$$

Hàm T' khác với hàm T ở trên, thay vì dùng L thì dùng L' có dạng

$$L'(X) = X \oplus (X \lll 13) \oplus (X \lll 23).$$

CK cũng là một mảng cố định. CK_i với $i = 0, 1, \dots, 31$ là bảng sau

00070e15,	1c232a31,	383f464d,	545b6269,
70777e85,	8c939aa1,	a8afb6bd,	c4cbd2d9,
e0e7eef5,	fc030a11,	181f262d,	343b4249,
50575e65,	6c737a81,	888f969d,	a4abb2b9,
c0c7ced5,	dce3eaf1,	f8ff060d,	141b2229,
30373e45,	4c535a61,	686f767d,	848b9299,
a0a7aeb5,	bcc3cad1,	d8dfe6ed,	f4fb0209,
10171e25,	2c333a41,	484f565d,	646b7279.

Giải mã

Để giải mã ta dùng round function ở trên nhưng theo thứ tự ngược lại.

ARX

ARX là cách gọi chung của những mô hình mã hóa khối chỉ sử dụng ba toán tử Addition (A, phép cộng modulo 2^n), Rotation (R, dịch vòng bit), và XOR (X, toán tử xor).

Giả sử ta có hai số nguyên a và b có n bit.

Khi đó, nếu ta biểu diễn $a \in \mathbb{Z}_{2^n}$ dưới dạng nhị phân

$$a = a_0 + 2a_1 + 2^2a_2 + \cdots + 2^{n-1}a_{n-1}$$

thì số nguyên a cũng tương ứng với vector

$$(a_0, a_1, a_2, \dots, a_{n-1}) \in \mathbb{F}_2^n.$$

Phép cộng modulo 2^n giữa hai số nguyên a và b được định nghĩa là

$$[a \boxplus b = (a + b) \bmod 2^n.]$$

Thông thường n cố định và là lũy thừa của 2. Ví dụ trong hệ mã Magma thì $n = 32$.

Với số nguyên dương r cố định, phép dịch vòng trái và phải r bit lần lượt là

$$\boxed{\begin{aligned} a \lll r &= (a_r, a_{r+1}, \dots, a_{n-1}, a_0, a_1, \dots, a_{r-1}), \\ a \ggg r &= (a_{n-r}, a_{n-r+1}, \dots, a_{n-1}, a_0, a_1, \dots, a_{n-r-1}). \end{aligned}}$$

Đối với phép XOR thì ta thực hiện phép cộng modulo 2 theo từng vị trí trong vector

$$[a \oplus b = (a_0 \oplus b_0, a_1 \oplus b_1, \dots, a_{n-1} \oplus b_{n-1}).]$$

Dưới góc độ phá mã vi sai, phép dịch vòng bit và phép XOR là hai biến đổi tuyến tính, và phép cộng modulo 2^n là biến đổi không tuyến tính. Do đó phép cộng modulo 2^n có thể loại bỏ sự có mặt của S-box trong thuật toán, phù hợp với các thiết bị lightweight vì không phải dùng bộ nhớ để lưu trữ S-box và các chip hiện nay đều hỗ trợ thực hiện các phép tính ARX.

Nhược điểm của ARX là có nhiều phần chưa được nghiên cứu rộng rãi, chẳng hạn, mặc dù phép cộng modulo 2^n là biến đổi không tuyến tính nhưng có quan hệ gì với phép XOR hay không. Nói cách khác, có nhiều điểm chưa rõ ràng đối với các toán tử ARX ở thời điểm hiện tại (2025).

Một số thuật toán lightweight sử dụng ARX: họ SPECK/SIMON.

3.1.4 Stream cipher

Chương này nói về mã dòng.

Linear Feedback Shift Register

Linear Feedback Shift Register (LFSR) là một ứng dụng quan trọng của hàm boolean để sinh ra một chuỗi các giá trị (giả ngẫu nhiên, pseudorandom).

Linear Feedback Shift Register

Xét hàm boolean n biến $f(x_0, x_1, \dots, x_{n-1})$. Khi đó với các giá trị (bit) khởi tạo x_0, x_1, \dots, x_{n-1} thuộc \mathbb{F}_2 , ta có thể sinh ra bit ở các vị trí tiếp theo

$$x_n = f(x_0, x_1, \dots, x_{n-1}).$$

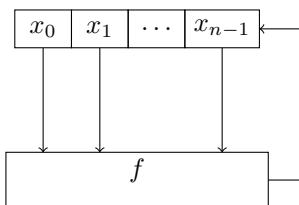
Tương tự:

$$x_{n+1} = f(x_1, x_2, \dots, x_n), \quad x_{n+2} = f(x_2, x_3, \dots, x_{n+1}), \quad x_{n+3} = f(x_3, x_4, \dots, x_{n+2}), \dots$$

Tổng quát, để sinh bit thứ $n + i$ thì đầu vào sẽ là các bit $x_i, x_{i+1}, \dots, x_{i+n-1}$.

$$x_{n+i} = f(x_i, x_{i+1}, \dots, x_{i+n-1}).$$

Theo [Hình 3.15](#), kết quả của hàm f sẽ được nối vào sau của dãy bit. Theo đó dãy bit sẽ luôn có dạng $x_0, x_1, \dots, x_n, \dots$



Hình 3.15: Feedback Shift Register

Qua hàm f , các vector trong \mathbb{F}_2^n sẽ chuyển trạng thái lẫn nhau theo công thức

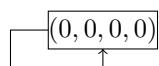
$$(x_{n-1}, x_{n-2}, \dots, x_1, x_0) \rightarrow (x_n, x_{n-1}, \dots, x_2, x_1).$$

Example

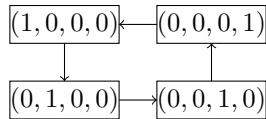
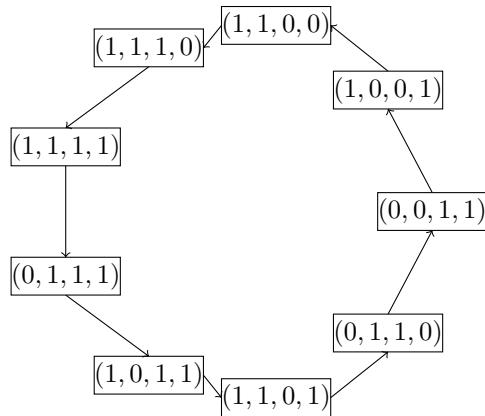
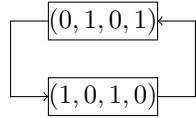
Xét hàm boolean $f(x_3, x_2, x_1, x_0) = x_3x_2 \oplus x_0$. Bảng chân trị của hàm f là

$$f(x_3, x_2, x_1, x_0) = (0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0).$$

Ví dụ, với $(0, 0, 0, 1)$, ta có $f(0, 0, 0, 1) = 1$ nên $(0, 0, 0, 1)$ biến đổi thành $(1, 0, 0, 0)$. Như vậy chúng ta có các chuyển đổi đối với hàm f được thể hiện thành 4 chu trình ở các hình bên dưới.



Hình 3.16: Chu trình thứ nhất của f

Hình 3.17: Chu trình thứ hai của f Hình 3.18: Chu trình thứ ba của f Hình 3.19: Chu trình thứ tư của f

Ta thấy rằng tập các vector $\mathbf{x} = (x_i, x_{i+1}, \dots, x_{i+n-1})$ có 2^n trường hợp. Do đó sẽ có một lúc nào đó (số i nào đó) mà vector \mathbf{x} trở lại đúng vector ban đầu. Như vậy tồn tại i sao cho

$$(x_0, x_1, \dots, x_{n-1}) = (x_i, x_{i+1}, \dots, x_{i+n-1}).$$

Từ đó dãy bit tiếp theo được sinh ra sẽ giống hệt trước đó nên số i nhỏ nhất thỏa mãn điều kiện được gọi là **chu kì** của Feedback Shift Register.

Trong các hàm boolean thì hàm boolean tuyến tính được quan tâm nhiều nhất để sinh ra chuỗi bit Feedback Shift Register. Do đó từ đây ta tập trung vào các hàm boolean tuyến tính và Linear Feedback Shift Register.

Nhắc lại, hàm boolean tuyến tính là hàm boolean có dạng

$$f(x_0, x_1, \dots, x_{n-1}) = a_0x_0 \oplus a_1x_1 \oplus \dots \oplus a_{n-1}x_{n-1}.$$

Trong đó $a_i \in \mathbb{F}_2$ là các hệ số cho trước. Ta định nghĩa đa thức đặc trưng cho hàm boolean tuyến tính như sau.

Definition (Đa thức đặc trưng)

Xét hàm boolean tuyến tính

$$f(x_0, x_1, \dots, x_{n-1}) = a_0x_0 \oplus a_1x_1 \oplus \dots \oplus a_{n-1}x_{n-1}.$$

Khi đó đa thức đặc trưng tương ứng với hàm f là đa thức có hệ số trong \mathbb{F}_2

$$P(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1} + x^n.$$

Do hàm boolean tuyến tính có tính chất là $f(bm\{0\}) = 0$ nên chu kì tối đa có thể đạt được là $2^n - 1$. Ta có một vài định nghĩa sau để một LFSR đạt được chu kì tối đa.

❶ Definition (Đa thức primitive)

Xét đa thức

$$P(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1} + x^n$$

thuộc \mathbb{F}_{2^n} . Ta đã biết trường \mathbb{F}_{2^n} có $2^n - 1$ phần tử khác không. Đặt $p = 2^n - 1$. Đa thức $P(x)$ được gọi là **primitive** khi với mọi ước nguyên tố q của p thì:

$$x^s \not\equiv 1 \pmod{P(x)}, \quad \text{với } s = \frac{p}{q} = \frac{2^n - 1}{q}.$$

❶ Definition (Đa thức đặc trưng với chu kì cực đại)

Đa thức

$$P(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1} + x^n$$

được gọi là **đa thức đặc trưng với chu kì cực đại** nếu đa thức đó tối giản và là đa thức primitive.

❶ Example

Xét đa thức $f(x) = x^4 + x^3 + 1$. Ta có thể xác định xem đa thức này có sinh ra LFSR với chu kì tối đa hay không mà không cần tìm đồ thị chuyển trạng thái của LFSR.

Đầu tiên ta chứng minh $f(x)$ là đa thức tối giản. Thật vậy, giả sử ngược lại, $f(x)$ là tích của hai đa thức bậc nhỏ hơn 4. Hai trường hợp có thể xảy ra là $f(x)$ chia hết cho đa thức tối giản bậc 1 hoặc bậc 2.

Các đa thức tối giản bậc 1 là x và $x + 1$. Ta có thể kiểm chứng rằng $f(x)$ không chia hết cho bất kì đa thức nào ở trên. Tương tự, đa thức tối giản bậc 2 (trong \mathbb{F}_{2^4}) là $x^2 + x + 1$ và $f(x)$ cũng không chia hết cho đa thức này. Như vậy ta có thể kết luận rằng $f(x)$ là đa thức tối giản.

Trong \mathbb{F}_{2^4} có $2^4 - 1 = 15$ phần tử khác 0. Các ước nguyên tố của 15 là 3 và 5. Ta thấy rằng

$$x^3 \not\equiv 1 \text{ và } x^5 = x \cdot x^4 = x \cdot (x^3 + 1) = x^4 + x = x^3 + x + 1 \not\equiv 1.$$

Như vậy $f(x)$ là đa thức primitive.

Như vậy ta có thể kết luận rằng đa thức $f(x)$ sinh ra LFSR với chu kì tối đa.

Thuật toán Berlekamp-Massey

Thuật toán Berlekamp-Massey là thuật toán tìm đa thức sinh có chu kì ngắn nhất sinh ra một dãy LFSR cho trước.

Bài toán ở đây là, giả sử chúng ta có một dãy bit

$$u_0, u_1, \dots, u_{l-1}$$

là một dãy bit được sinh giả ngẫu nhiên (pseudorandom). Làm thế nào từ đoạn bit này ta xây dựng được đa thức đặc trưng của LFSR mà sinh ra tất cả các bit sau đó? Hơn nữa l phải nhỏ nhất có thể.

Nhắc lại, đa thức đặc trưng là đa thức có dạng

$$P(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1} + a_nx^n$$

thì hàm boolean tuyến tính sinh ra bit tiếp theo sẽ có dạng

$$x_n = f(x_0, x_1, \dots, x_{n-1}) = a_0x_0 \oplus a_1x_1 \oplus \dots \oplus a_{n-1}x_{n-1}.$$

Tổng quát, công thức để sinh ra bit thứ $n+i$ sẽ là

$$x_{n+i} = f(x_i, x_{i+1}, \dots, x_{n+i-1}) = a_0x_i \oplus a_1x_{i+1} \oplus \dots \oplus a_{n-1}x_{n+i-1}$$

với $i = 0, 1, \dots$

Algorithm (Thuật toán Berlekamp-Massey)

Đầu vào là dãy bit u_0, u_1, \dots, u_{l-1} .

Đầu ra là đa thức đặc trưng bậc nhỏ nhất mà sinh ra toàn bộ dãy bit trên, bắt đầu từ các bit u_0, u_1, \dots nhất định.

Bước -1. Gọi n_0 là vị trí đầu tiên mà $u_{n_0} = 1$ (các bit được đánh số từ 0). Khi đó đặt $P_{-1}(x) = 1$ và $k_{-1} = 1$. Nếu $P_{-1}(x)$ sinh ra toàn bộ dãy bit thì ta dừng thuật toán. Ngược lại thì tiếp tục bước 1.

Bước 0. Đặt $P_0(x) = x^{n_0+1} \oplus P_{-1}(x)$ với n_0 là vị trí bit 1 đầu tiên và đặt $k_0 = \deg P_0(x) = n_0 + 1$.

Ở mỗi bước từ đây trở đi, gọi m là số sao cho

$$k_{-1} = k_0 = \dots = k_{m-1} < k_m = k_{m+2} = \dots$$

Bước n . Để tìm $P_n(x)$ ta tính $a = m - k_{m-1}$ và $b = n - k_{n-1}$.

1. Nếu $a \geq b$ thì

$$P_n(x) = P_{n-1}(x) \oplus x^{a-b}P_{m-1}(x).$$

2. Nếu $a < b$ thì

$$P_n(x) = x^{b-a}P_{n-1}(x) \oplus P_{m-1}.$$

Trong quá trình tìm $P_n(x)$, khi nào bậc k_n của $P_n(x)$ thỏa mãn điều kiện số m thì ta cập nhật lại số m .

Ở mỗi bước, nếu $P_n(x)$ sinh ra toàn bộ dãy bit thì ta dừng thuật toán.

Để xem cách hoạt động của thuật toán Berlekamp-Massey ta sẽ giải ví dụ sau.

Xét dãy bit 111100100.

Đặt $n_0 = 0$, $P_{-1}(x) = 1$ và $k_{-1} = 0$.

Theo $P_{-1}(x) = 1$ thì $1 \rightarrow 1 \rightarrow 1 \rightarrow 1$, nghĩa là bit đầu thành bit thứ hai, bit thứ hai thành bit thứ ba và thứ ba thành thứ tư. Từ đó suy ra

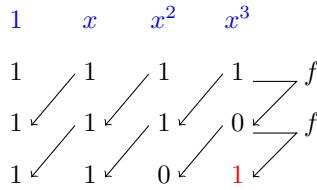
$$P_0(x) = P_1(x) = P_2(x) = P_3(x) = x^{0+1} \oplus 1 = x \oplus 1.$$

Do $k_{-1} < k_0 = k_1 = k_2 = k_3$ ($0 < 1$) nên $m = 0$.

Để tìm $P_4(x)$, ta có $a = m - k_{m-1} = 0 - 0 = 0$ và $b = n - k_{n-1} = 4 - 1 = 3$. Do $a < b$ nên

$$P_4(x) = x^3 P_3(x) \oplus P_{-1}(x) = x^3 \cdot (x \oplus 1) \oplus 1 = x^4 \oplus x^3 \oplus 1.$$

Do $k_4 > k_3$ ($4 > 1$) nên $m = 4$.



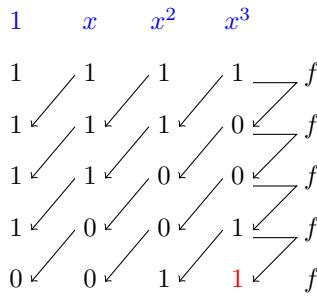
Hình 3.20: LFSR tương ứng $P_4(x)$

Trong hình trên, hàng đầu từ trái sang phải là u_0, u_1, u_2, u_3 . Hàng thứ hai từ trái sang phải là u_1, u_2, u_3, u_4 . Hàng thứ ba là u_2, u_3, u_4, u_5 , nhưng $u_5 = 0$ theo dây ban đầu nên LFSR sinh ra 1 là chưa đúng, thuật toán chuyển sang bước tiếp theo.

Để tìm $P_5(x)$, ta có $a = m - k_{m-1} = 4 - 1 = 3$ và $b = n - k_{n-1} = 5 - 4 = 1$. Suy ra

$$P_5(x) = P_4(x) \oplus x^2 P_3(x) = x^4 \oplus x^3 \oplus 1 \oplus x^2 \cdot (x \oplus 1) = x^4 \oplus x^2 \oplus 1.$$

Theo $P_5(x) = x^4 \oplus x^2 \oplus 1$ thì LFSR sẽ hoạt động như hình dưới đây. Cũng theo hình dưới thì $P_6(x) = P_5(x) = x^4 \oplus x^2 \oplus 1$.

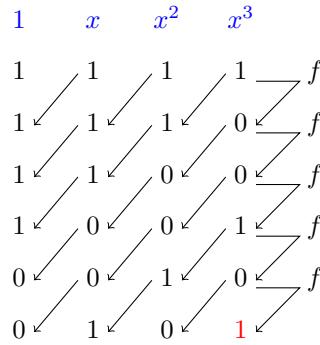


Hình 3.21: LFSR tương ứng $P_5(x)$ và $P_6(x)$

Để tìm $P_7(x)$, ta có $a = m - k_{m-1} = 3$ và $b = n - k_{n-1} = 7 - 4 = 3$. Do $a \geq b$ nên

$$P_7(x) = P_6(x) \oplus P_3(x) = x^4 \oplus x^2 \oplus 1 \oplus x \oplus 1 = x^4 \oplus x^2 \oplus x.$$

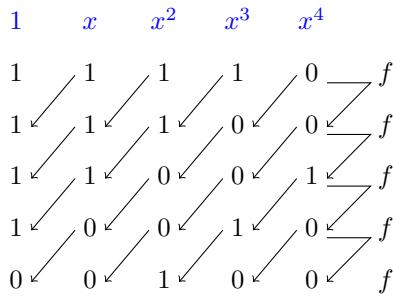
Khi đó LFSR sẽ được sinh như hình.

Hình 3.22: LFSR tương ứng $P_7(x)$

Để tìm $P_8(x)$, $a = 3$ và $b = 8 - 4 = 4$. Do $a < b$ nên

$$P_8(x) = xP_7(x) \oplus P_3(x) = x^5 \oplus x^3 \oplus x^2 \oplus x \oplus 1.$$

Lúc này LFSR sẽ là

Hình 3.23: LFSR tương ứng $P_8(x)$

Như vậy đa thức sinh ra dãy bit ban đầu là $x^5 \oplus x^3 \oplus x^2 \oplus x \oplus 1$. Thuật toán Berlekamp-Massey tìm ra đa thức đặc trưng với độ phức tạp là 8 (tìm tới $P_8(x)$).

Ví dụ trên có thể được kiểm tra bởi chương trình Python ở [đây](#).

3.1.5 Message Authentication Code

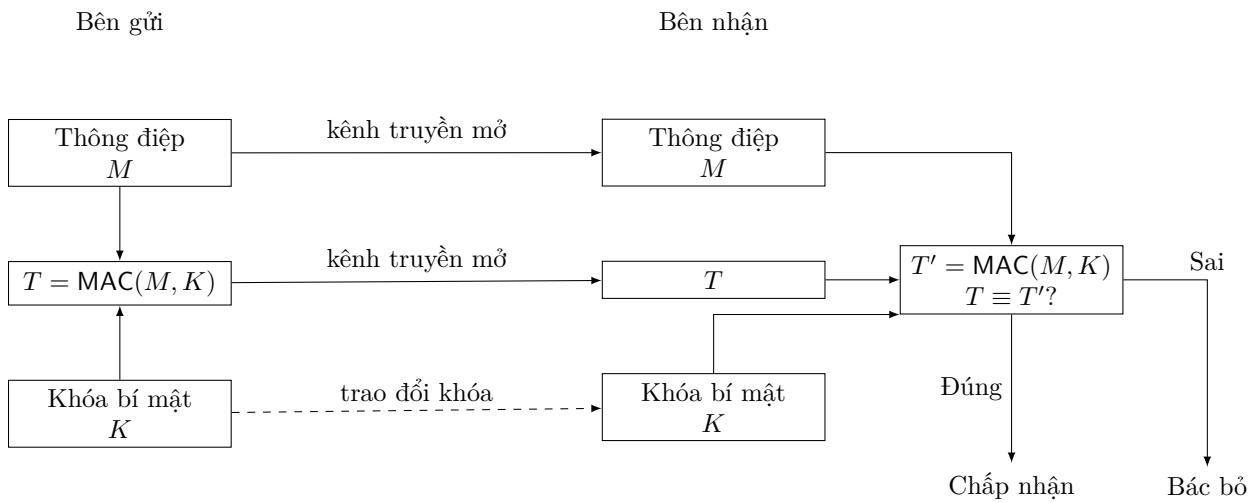
Điểm yếu của các hệ mật mã đối xứng là ở tính không từ chối. Bên nhận ciphertext không có cách nào xác minh được bên gửi, cũng như bên gửi hoàn toàn có thể chối bỏ rằng mình đã gửi ciphertext đi.

Để giải quyết vấn đề này người ta đã nghĩ ra một phương án là **Message Authentication Code** (viết tắt là **MAC**, tiếng Nga là **ИМПОВСТАВКА**). Một số tài liệu tiếng Việt mình đọc thì MAC được dịch là *mã chứng thực thông điệp*.

Khi đó, MAC gồm ba thành phần:

1. Thuật toán tạo khóa bí mật K .
2. Thuật toán tạo ra tag T là thông tin chứng thực từ khóa bí mật K và thông điệp M , nói cách khác là $T = \text{MAC}(K, M)$.

3. Thuật toán kiểm tra: với T , K và M , thuật toán trả về *chấp nhận* hoặc *bác bỏ*. Bên nhận sẽ tính toán $T' = \text{MAC}(K, M)$ và so sánh với T . Nếu $T \equiv T'$ thì chấp nhận thông tin không bị sửa đổi và được gửi từ một bên sở hữu khóa bí mật K .

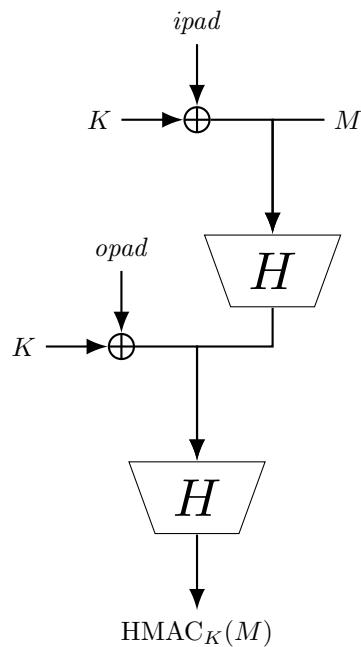


Hình 3.24: Sơ đồ tạo MAC

Nhìn chung, MAC giống với quy trình tạo chữ ký điện tử.

Hash-based Message Authentication Code (HMAC)

HMAC là một cách tiếp cận để xây dựng MAC dựa trên hàm băm nên có tên gọi hash-based MAC.



Hình 3.25: Hash-based message authentication code

HMAC được định nghĩa trong RFC 2104 [24] bởi công thức

$$\text{HMAC}(K, M) = H((K' \oplus \text{opad}) \| H(K' \oplus \text{ipad}) \| M),$$

trong đó:

- H là một hàm băm mật mã;
- M là thông điệp cần chứng thực;
- K là khóa bí mật;
- K' là khóa với độ dài cố định được tạo từ khóa bí mật K ;
- opad là một dãy các byte 0x5c, viết tắt của outer padding;
- ipad là một dãy các byte 0x36, viết tắt của inner padding.

Độ dài cố định (block-size) ở mô tả trên phụ thuộc vào hàm băm mật mã H .

Mô hình HMAC cho phép chứng thực thông điệp nhưng không cho phép mã hóa (encrypt). Một ý tưởng cho việc này là chúng ta mã hóa trước rồi chứng thực bản mã nhận được. Đây là cách tiếp cận **Encrypt-then-MAC** và được ứng dụng khá rộng rãi, ví dụ như ở [25]. Chúng ta cũng có thể làm ngược lại, chứng thực bản rõ trước và sau đó mã hóa, gọi là **MAC-then-encrypt**. Tuy nhiên MAC-then-encrypt không được sử dụng nhiều.

Một ứng dụng tiêu biểu của Encrypt-then-MAC là cơ chế GCM (Galois/Counter Mode) được sử dụng rộng rãi khi đi kèm với thuật toán AES. Trước tiên chúng ta sẽ xem xét bài toán tổng quát hơn là Authentication Encryption (tạm dịch là *mã hóa có chứng thực*).

Authentication Encryption

Đầu tiên ta thống nhất các kí hiệu sau.

Kí hiệu	Ý nghĩa
P	bản rõ (plaintext)
K	khóa cho thuật toán mã hóa đối xứng
K'	khóa dùng để tạo MAC
$\text{Enc}_K(P)$	hàm mã hóa đối xứng bản rõ P với khóa bí mật K
C	bản mã (ciphertext) khi mã hóa bản rõ P với khóa bí mật K
$C = \text{Enc}_K(P)$	
A	thông tin để chứng thực (Authentication Data)

Authentication Encryption (AE)

Authentication Encryption (hay **AE**) có thể biểu diễn bởi hàm

$$T = \text{MAC}(K', C),$$

khi đó MAC được tạo bởi khóa K' , bản mã C .

Thông thường, K' sẽ được sinh ra từ K hoặc cả hai đều được sinh từ một mật khẩu (password) hoặc passphrase nào đó.

Authentication Encryption with Associated Data (AEAD)

Tương tự với AE nhưng lúc này chúng ta thêm một đoạn thông tin gọi là **Associated Data** hoặc **Authentication Data**. Lúc này MAC sẽ được tính bởi công thức

$$T = \text{MAC}(K', C, A),$$

với A là thông tin chứng thực (authentication data).

Hiện nay trong các giao thức mật mã thì AEAD là bắt buộc đi kèm với thuật toán mã hóa đối xứng.

Ở TLS v1.3 thì đối với thuật toán AES có hai phương án AEAD là **GCM** (Galois/Counter Mode) và **CCM** (CBC-MAC). Đối với các cipher suites sử dụng các thuật toán tiêu chuẩn GOST (Liên bang Nga) sử dụng phương án AEAD là **MGM** (Multilinear Galois Mode). Chúng ta cần lưu ý rằng các AEAD được định nghĩa trong các *khuyến nghị* (*recommendation*) chứ không phải trong các *tiêu chuẩn* [26, 27, 28]. Nói cách khác, khi phát triển sản phẩm mật mã thì đây không phải quy định bắt buộc nhưng về mặt an toàn thì chúng ta nên áp dụng vì những mode khác luôn có khuyết điểm (ECB, CBC, v.v.).

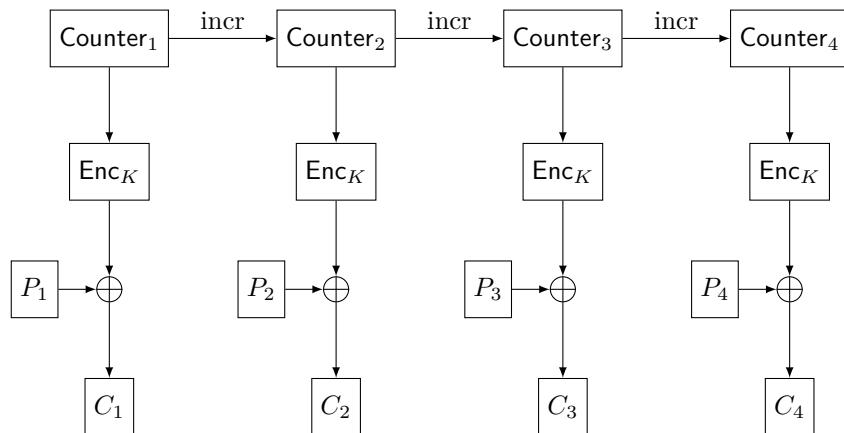
Phần sau mình sẽ trình bày cách tính MAC dựa trên GCM, CCM và MGM.

Các loại AEAD

Sơ lược về mã hóa với bộ đếm (counter)

Cả ba phương án AEAD sau đây đều sử dụng CTR (Counter Mode) để mã hóa. Sau đó việc tính toán MAC được thực hiện theo những cách khác nhau.

Cơ chế mã hóa với bộ đếm như hình sau.



Hình 3.26: Mã hóa theo mode CTR

Bắt đầu với giá trị Counter₁, các counter sau đó được tạo ra khi tăng dần bộ đếm với hàm `incr` nào đó:

$$\text{Counter}_{i+1} = \text{incr}(\text{Counter}_i)$$

với $i = 1, 2, \dots$

Khi đó, nếu P_1, P_2, \dots, P_n là các khối bản rõ thì các khối bản mã tương ứng C_1, C_2, \dots, C_n được tính bởi

$$C_i = P_i \oplus \text{Enc}_K(\text{Counter}_i).$$

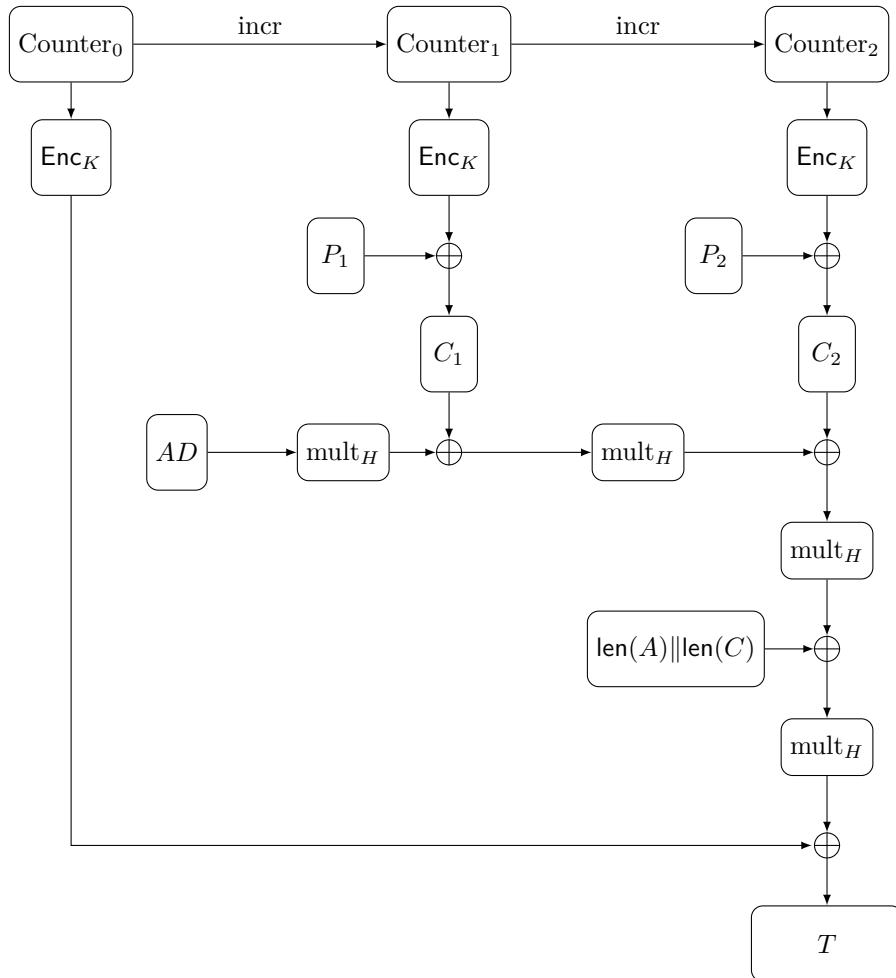
Hàm `incr` là một hàm tăng nào đó, không nhất thiết là cộng 1.

Điều quan trọng khi chúng ta sử dụng CTR là không được sử dụng lại Counter₁, các bạn có thể tìm về lỗ hỏng *reuse nonce*. Nếu sử dụng lại counter thì chúng ta sẽ bị tấn công dạng known-plaintext.

Galois/Counter Mode (GCM)

Đây là dạng AEAD được sử dụng rộng rãi nhất hiện nay (ít ra là mình thấy vậy :v).

GCM sử dụng CTR để mã hóa và sử dụng các phép tính trên trường Galois để tạo MAC nên có tên gọi là Galois/Counter Mode.



Bắt đầu với Counter₀, hay còn gọi là **nonce**, các giá trị bộ đếm tiếp theo được tính bởi việc tăng giá trị trước đó bởi hàm incr. Việc mã hóa sử dụng CTR giống như đã trình bày ở trên.

Để tính MAC, giả sử chúng ta có m khối associated data là A_1, A_2, \dots, A_m và n khối bản mã C_1, C_2, \dots, C_n . Khi đó các khối A_i và C_j có 128 bit được biểu diễn thành các đa thức trong GF(2¹²⁸) với đa thức tối giản là $x^{128} + x^7 + x^2 + x + 1$.

Đặt

$$A = A_1 \| A_2 \| \cdots \| A_m, \quad C = C_1 \| C_2 \| \cdots \| C_n.$$

Đặt $H = \text{Enc}_K(0^{128})$. Đây là phần tử trường Galois dùng để tạo MAC nên có thể nói H chính là K' ở phần mô tả MAC bên trên.

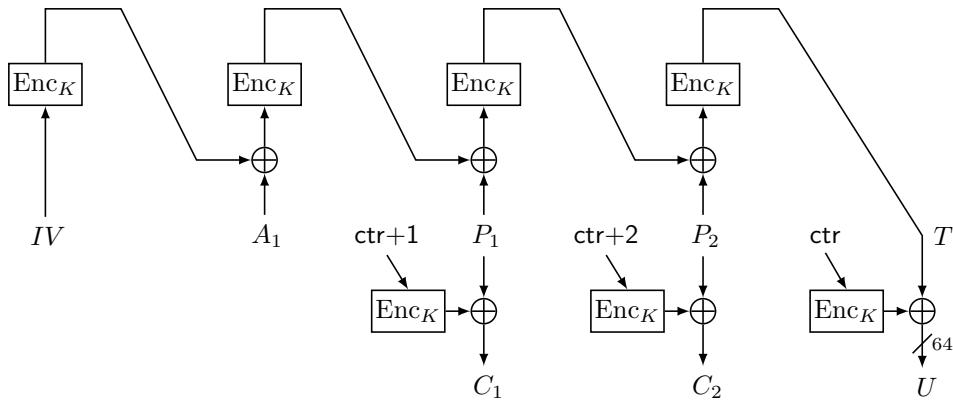
MAC T sẽ được tính bởi công thức

$$\begin{aligned} T = & A_1 H^{n+m+2} \oplus A_2 H^{n+m+1} \oplus \cdots A_m H^{n+3} \\ & \oplus C_1 H^{n+2} \oplus C_2 H^{n+1} \oplus \cdots \oplus C_n H^2 \\ & \oplus LH \oplus \text{Enc}_K(\text{Counter}_0), \end{aligned}$$

với $L = \text{len}(A) \parallel \text{len}(C)$, nghĩa là độ dài cả đoạn A là $\text{len}(A)$ sẽ được biểu diễn bằng dãy 64 bit và tương tự, độ dài cả đoạn C là $\text{len}(C)$ sẽ được biểu diễn bằng dãy 64 bit. Khi nối hai đoạn đó lại ta có khối L có 128 bit.

Ở hình minh họa ở trên thì $m = 1$ và $n = 2$ (có một khối AD và hai khối bản mã).

CBC-MAC



Hình 3.27: Mã hóa theo CCM

Việc mã hóa cũng sử dụng bộ đếm CTR, ở đây là $\text{ctr} + i$ với $i = 1, 2, \dots$ còn ctr thì dùng để tính MAC sau.

Tương tự, ta giả sử có m khối AD và n khối bản mã. Khi đó với một IV (có công thức tạo nhưng mình không nói ở đây) thì đầu tiên ta đặt $B_0 = \text{Enc}_K(\text{IV})$, ta tính

$$B_i = \text{Enc}_K(A_i \oplus B_{i-1})$$

với $i = 1, 2, \dots, m$. Phần này là CBC dành cho AD.

Ta tiếp tục tính CBC cho các bản rõ P_1, P_2, \dots, P_n bằng

$$B_{i+m} = \text{Enc}_K(P_i \oplus B_{i+m-1})$$

với $i = 1, 2, \dots, n$.

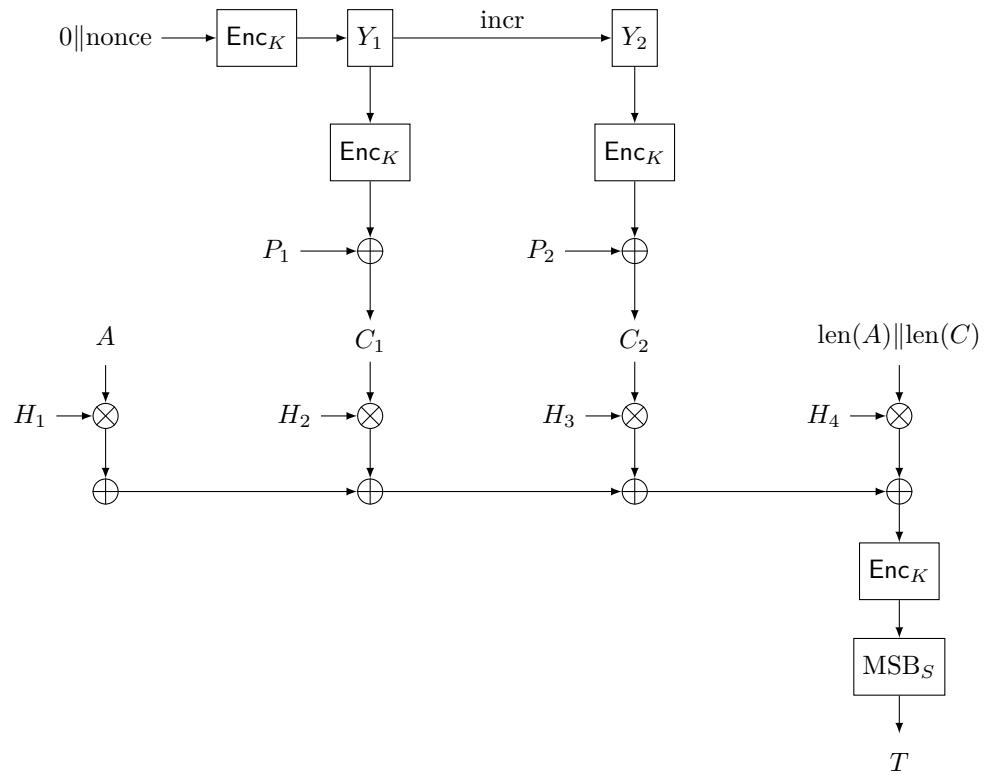
Tag T sẽ là B_{n+m} và MAC là 64 bit cao nhất (MSB) của $\text{Enc}_K(\text{ctr}) \oplus T$, nghĩa là

$$U = \text{MSB}_{64}(\text{Enc}_K(\text{ctr}) \oplus T).$$

Ở đây MAC được tạo bởi CBC nên có tên gọi là CBC-MAC.

Multilinear Galois Mode (MGM)

Đây là AEAD sử dụng cho các GOST cipher suite cho TLS v1.3 và là mở rộng của GCM. Ở đây thay vì chúng ta chỉ nhân với một phần tử H như GCM mà là một dãy H_1, H_2, \dots nên có tên gọi là multilinear Galois.



Hình 3.28: Mã hóa theo MGM

Đối với MGM chúng ta cần một đoạn 127 bit gọi là nonce và sẽ sử dụng để mã hóa lẩn tính MAC.

Để mã hóa, đặt $Y_1 = \text{Enc}_K(0 \parallel \text{nonce})$. Đây là điểm khởi đầu của bộ đếm. Khi đó các phần tử bộ đếm được sinh bởi quy tắc

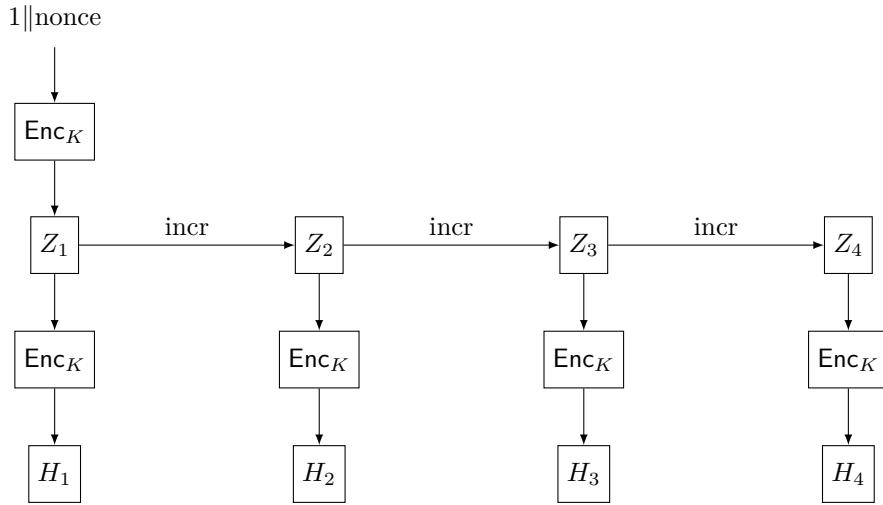
$$Y_{i+1} = \text{incr}_r(Y_i), \quad i = 1, 2, \dots, n.$$

Bản mã sẽ là

$$C_i = \text{Enc}_K(Y_i) \oplus P_i, \quad i = 1, 2, \dots, n.$$

Tiếp theo, đặt $Z_1 = \text{Enc}_K(1 \parallel \text{nonce})$ và

$$Z_{i+1} = \text{incr}_l(Z_i), \quad i = 1, 2, \dots, n + m + 1.$$

Hình 3.29: Quá trình sinh dây (H_i)

Chúng ta tính một dãy H_1, H_2, \dots theo quy tắc sau (Hình 3.29)

$$H_i = \text{Enc}_K(Z_i), \quad i = 1, 2, \dots, n + m + 1.$$

Dãy H_i sẽ được dùng để tính MAC. Gọi

$$\mathcal{T} = A_1 H_1 \oplus A_2 H_2 \oplus \dots \oplus A_m H_m \oplus C_1 H_{m+1} \oplus \dots \oplus C_n H_{n+m} \oplus L H_{n+m+1},$$

với $L = \text{len}(A) \parallel \text{len}(C)$ như GCM. Một điều lưu ý ở đây là MGM có thể sử dụng cho thuật toán với độ dài khối là 64 bit (Magma) và 128 bit (Kuznyechik). Khi đó, nếu độ dài khối là 64 bit thì đa thức tối giản là $x^{64} + x^4 + x^3 + x + 1$ và nếu độ dài khối là 128 bit thì đa thức tối giản là $x^{128} + x^7 + x^2 + x + 1$.

Khi đó, MAC là 64 bit cao (MSB) của kết quả $\text{Enc}_K(\mathcal{T})$, nghĩa là

$$T = \text{MSB}_{64}(\text{Enc}_K(\mathcal{T})).$$

Hàm incr_l và incr_r hoạt động theo nguyên tắc, giả sử chúng ta có một số 128 bit là $L \parallel R$, trong đó L và R đều có 64 bit. Khi đó

- $\text{incr}_l(L \parallel R) = L' \parallel R$ với $L' = (L + 1) \bmod 2^{64}$;
- $\text{incr}_r(L \parallel R) = L \parallel R'$ với $R' = (R + 1) \bmod 2^{64}$.

Ở đây L' (hoặc R') được biểu diễn bởi dãy 64 bit và gắn vào R (hoặc L) ban đầu để có một dãy 128 bit.

3.2 Phá mã

3.2.1 Phá mã vi sai

Phá mã vi sai (differential cryptanalysis) đã làm chuẩn mã hóa DES không còn đủ an toàn.

Trong các chuẩn mã hóa hiện đại về sau, khả năng kháng phá mã vi sai và phá mã tuyến tính trở thành tiêu chuẩn đánh giá độ an toàn của thuật toán mã hóa.

Nhập môn phá mã vi sai

Phần này mình tham khảo ở [29]. Bạn này viết khá nhiều bài nhập môn cryptanalysis trên block cipher nên rất tốt để tham khảo.

Differential (vi sai)

1 Definition 1.63

Gọi \mathbb{F}_2^n và \mathbb{F}_2^m là hai không gian vector trên \mathbb{F}_2 với số chiều lần lượt là n và m . Gọi S là ánh xạ từ \mathbb{F}_2^n tới \mathbb{F}_2^m . Với mỗi vector $\mathbf{a}, \mathbf{b} \in \mathbb{F}_2^n$, ta nói **input differential** của S là $\delta = \mathbf{a} \oplus \mathbf{b}$ và **output differential** của S là $\Delta = S(\mathbf{a}) \oplus S(\mathbf{b})$, trong đó \oplus là phép XOR.

Trên thực tế, nếu trường được xét không phải \mathbb{F}_2 mà là một trường \mathbb{F} bất kì thì input differential là $\mathbf{b} - \mathbf{a}$ và output differential là $S(\mathbf{b}) - S(\mathbf{a})$. Trong mật mã chúng ta thường hay làm việc trên các không gian vector nhị phân nên phép trừ cũng chính là phép cộng (XOR) trên \mathbb{F}_2 .

Ánh xạ S thường được sử dụng trong S-box là ánh xạ không tuyến tính, nghĩa là chúng ta không có tính chất $S(\mathbf{a} \oplus \mathbf{b}) = S(\mathbf{a}) \oplus S(\mathbf{b})$. Tuy nhiên khi phân tích phân bố của $\delta = \mathbf{a} \oplus \mathbf{b}$ và $\Delta = S(\mathbf{a}) \oplus S(\mathbf{b})$ ta có thể trích ra các thông tin phục vụ tấn công. Do sử dụng các thông tin thống kê từ differential nên cách tấn công này được gọi là **differential attack** (hay **phá mã vi sai**).

Toy cipher

1. Độ dài khối là 4 bits.
2. Độ dài khóa là 8 bits.

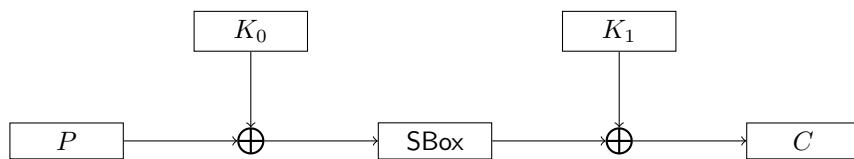
Mình gọi plaintext 4 bits là P và khóa 8 bits là $K = K_0 \| K_1$, trong đó K_0 và K_1 có 4 bits và $\|$ là toán tử ghép chuỗi.

S-box của toy cipher là ánh xạ \mathbb{F}_2^4 tới \mathbb{F}_2^4 theo bảng sau.

\mathbf{x}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S(\mathbf{x})$	3	14	1	10	4	9	5	6	8	11	15	2	13	12	0	7

Quá trình mã hóa mỗi khối plaintext 4 bit P thành ciphertext C cũng 4 bit là:

$$P \rightarrow P \oplus K_0 \rightarrow S(P \oplus K_0) \rightarrow S(P \oplus K_0) \oplus K_1 = C.$$



Hình 3.30: Sơ đồ toy cipher

Phân tích vi sai

Tiếp theo chúng ta phân tích sự phân bố vi sai của S-box và biểu diễn thành bảng.

Trong bảng này, phần tử ở hàng i và cột j thể hiện số lượng cặp $(\mathbf{a}, \mathbf{b}) \in \mathbb{F}_2^4 \times \mathbb{F}_2^4$ sao cho $\mathbf{a} \oplus \mathbf{b} = i$ và $S(\mathbf{a} \oplus \mathbf{b}) = j$.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	2	0	4	0	0	0	2	0	0	0	2	0	6	0	
2	0	2	2	0	2	0	0	2	0	2	0	2	0	2	0	
3	0	0	2	0	2	0	0	0	2	4	0	4	0	0	2	
4	0	0	0	0	2	4	0	6	0	0	0	0	2	0	2	
5	0	0	2	0	2	0	2	2	2	0	4	0	0	0	2	
6	0	0	2	2	0	2	2	0	4	0	0	0	2	0	2	
7	0	0	0	2	0	2	0	0	2	0	0	4	0	0	2	
8	0	2	0	0	0	6	0	0	2	2	0	2	0	0	0	
9	0	0	2	2	2	4	0	4	0	0	0	0	0	0	0	
10	0	2	0	0	2	0	0	0	2	2	2	0	4	0	2	
11	0	4	2	2	0	0	0	0	0	4	2	2	0	0	0	
12	0	2	4	0	2	0	0	0	0	0	2	0	2	2	0	
13	0	2	0	2	0	0	2	2	0	2	2	0	0	0	4	
14	0	0	0	2	0	0	2	0	0	2	0	4	2	4	0	
15	0	0	0	0	2	0	4	2	0	0	0	0	2	6	0	

Chúng ta quan tâm tới những hàng có nhiều giá trị 0 và có một phần tử lớn nhất.

Cụ thể thì ở bảng này:

- Nếu input differential là 0 thì output differential là 0 với xác suất $16/16 = 1$.
- Nếu input differential là 1 thì output differential là 13 với xác suất $6/16$.
- Nếu input differential là 4 thì output differential là 7 với xác suất $6/16$.
- Nếu input differential là 8 thì output differential là 5 với xác suất $6/16$.
- Nếu input differential là 15 thì output differential là 14 với xác suất $6/16$.

Khi input differential là 0, nói cách khác là $\mathbf{a} \oplus \mathbf{b} = 0$, tương đương với $\mathbf{a} = \mathbf{b}$. Suy ra $S(\mathbf{a}) = S(\mathbf{b})$ nên output differential luôn là 0 = $S(\mathbf{a}) \oplus S(\mathbf{b})$. Tính chất này không hữu dụng.

Tiếp theo, giả sử ta mã hóa plaintext P_1 và nhận được ciphertext tương ứng là C_1 .

Tương tự ta mã hóa plaintext P'_1 và nhận được ciphertext tương ứng là C'_1 .

Theo cấu trúc cipher, $P_1 \oplus K_0$ và $P'_1 \oplus K_0$ sẽ là các đầu vào của S-box. Do đó input differential là

$$\delta = (P_1 \oplus K_0) \oplus (P'_1 \oplus K_0) = P_1 \oplus P'_1.$$

Điều này có nghĩa là input differential không phụ thuộc vào khóa và đây là tính chất quan trọng cho phá mã vi sai.

Theo quan sát số 2 về bảng phân bố vi sai ở trên, nếu input differential là 1 thì output differential là 13 với xác suất $6/16$.

Giả sử ta chọn các plaintext P_1 và P'_1 sao cho $P_1 \oplus P'_1 = 1$.

Đặt $I_1 = S(P_1 \oplus K_0)$ và $I'_1 = S(P'_1 \oplus K_0)$. Khi đó output differential của S-box là $I_1 \oplus I'_1$ và bằng 13 với xác suất $6/16$.

Để ý rằng $C_1 = I_1 \oplus K_1$ và $C'_1 = I'_1 \oplus K_1$ nên

$$C_1 \oplus C'_1 = (I_1 \oplus K_1) \oplus (I'_1 \oplus K_1) = I_1 \oplus I'_1$$

chính là output differential.

Remark 1.15

Nếu hai plaintext P_1 và P'_1 thỏa mãn $P_1 \oplus P'_1 = 1$ thì hai ciphertext tương ứng C_1 và C'_1 sẽ cho giá trị $C_1 \oplus C'_1 = 13$ với xác suất $6/16$.

Nói cách khác, trung bình 16 cặp plaintext mà $P_1 \oplus P'_1 = 16$ thì sẽ có 6 cặp cho kết quả $C_1 \oplus C'_1 = 13$.

Chosen plaintext

Dựa vào nhận xét trên, chiến thuật phá mã vi sai thực hiện như sau:

1. Chọn ngẫu nhiên plaintext P_1 và tính $P'_1 = P_1 \oplus 1$.
2. Mã hóa P_1 thu được C_1 , mã hóa P'_1 thu được C'_1 .
3. Nếu $C_1 \oplus C'_1 = 13$ thì ta đã tìm được một cặp plaintext "tốt". Nếu không thì ta quay lại bước 1.

Xác suất $6/16$ đảm bảo rằng việc tìm kiếm sẽ không mất thời gian vì xác suất này khá lớn so với phần còn lại. :))

Khi chúng ta đã tìm được cặp plaintext (P_1, P'_1) mà $C_1 \oplus C'_1 = 13$, chúng ta đã sẵn sàng khôi phục khóa con K_0 .

Đầu tiên, ta liệt kê tất cả cặp $(\mathbf{a}, \mathbf{b}) \in \mathbb{F}_2^4 \times \mathbb{F}_2^4$ sao cho $\mathbf{a} \oplus \mathbf{b} = 1$ và $S(\mathbf{a}) \oplus S(\mathbf{b}) = 13$. Các cặp đó là

$$\mathcal{S}_0 = \{(0, 1), (1, 0), (4, 5), (5, 4), (10, 11), (11, 10)\}.$$

Do phép XOR có tính đối xứng nên tập \mathcal{S}_0 cũng có những cặp đối xứng. Do đó ta chỉ cần lấy phần tử đầu của mỗi cặp và đặt

$$\mathcal{A} = \{0, 1, 4, 5, 10, 11\}.$$

Mỗi phần tử $\mathbf{a} \in \mathcal{A}$ có tính chất $S(\mathbf{a}) \oplus S(\mathbf{a} \oplus 1) = 13$.

Theo cấu trúc của toy cipher thì $S(P_1 \oplus K_0) \oplus S(P'_1 \oplus K_0) = 13$.

Như vậy $\mathbf{a} = P_1 \oplus K_0$ và $\mathbf{a} \oplus 1 = P'_1 \oplus K_0$. Từ đây ta tìm được các khả năng có thể có của khóa con K_0 ứng với mỗi giá trị \mathbf{a} .

Với mỗi giá trị K_0 , ta tính được $K_1 = C_1 \oplus S(P_1 \oplus K_0)$. Do \mathcal{A} có 6 phần tử nên ta sẽ có 6 trường hợp khóa $K = K_0 \| K_1$.

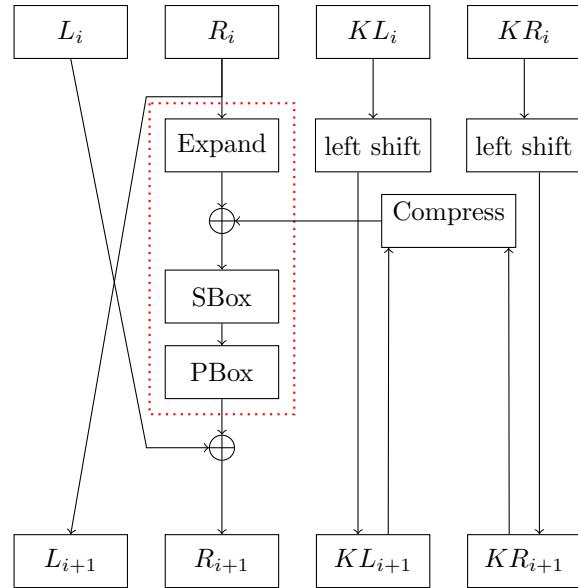
Để tìm ra khóa ta có thể thử mã hóa P_1 với từng khóa. Nếu kết quả là C_1 thì ta đã khôi phục đúng khóa.

Differential cryptanalysis ở toy cipher đã giảm số lượng khóa cần thử từ $2^8 = 256$ trường hợp xuống còn 6 trường hợp.

Phá mã vi sai trên TinyDES

Mô tả TinyDES

TinyDES là một phiên bản thu nhỏ của chuẩn mã hóa DES. TinyDES là mã hóa khôi theo mô hình Feistel, kích thước khối là 8 bit, kích thước khóa cũng là 8 bit. Mỗi vòng khóa con có độ dài 6 bit.



Hình 3.31: Một vòng TinyDES

Mã TinyDES khá đơn giản. Theo mô hình Feistel, khối đầu vào 8 bit được chia thành hai nửa trái phải 4 bit. Nửa phải sẽ đi qua các hàm Expand, SBox và PBox, sau đó XOR với nửa trái để được nửa phải mới. Còn nửa trái mới là nửa phải cũ. Tóm lại công thức mô hình Feistel là:

$$L_{i+1} = R_i, \quad R_{i+1} = L_i \oplus F(R_i, K_{i+1})$$

với $i = 1, 2, 3$ tương ứng 3 vòng với đầu vào của khối là (L_0, R_0) .

Chúng ta cần các động tác sau:

1. Expand: mở rộng và hoán vị R_i từ 4 bits lên 6 bits. Giả sử 4 bits của R_i là $b_0b_1b_2b_3$ thì kết quả sau khi Expand là $b_2b_3b_1b_2b_1b_0$.
 2. SBox: gọi 6 bits đầu vào là $b_0b_1b_2b_3b_4b_5$. Khi đó ta tra cứu theo bảng SBox với b_0b_5 chỉ số **hàng**, $b_1b_2b_3b_4$ chỉ số **cột**. Nói cách khác bảng SBox có 4 hàng, 16 cột. Kết quả của SBox là một số 4 bit.
 3. PBox: là hàm hoán vị 4 bits $b_0b_1b_2b_3$ thành $b_2b_0b_3b_1$.

Như vậy, hàm F của mô hình Feistel đối với mã TinyDES là:

$$F(R_i, K_i) = \text{PBox}(\text{SBox}(\text{Expand}(R_i) \oplus K_{i+1})).$$

Để sinh khóa con cho 3 vòng, khóa ban đầu được chia thành hai nửa trái phải lần lượt là KL_0 và KR_0 . TinyDES thực hiện như sau:

1. Vòng 1: KL_0 và KR_0 được dịch vòng trái 1 bit để được KL_1 và KR_1 ;
 2. Vòng 2: KL_1 và KR_1 được dịch vòng trái 2 bit để được KL_2 và KR_2 ;
 3. Vòng 3: KL_2 và KR_2 được dịch vòng trái 1 bit để được KL_3 và KR_3 .

Khi đó, khóa K_i ở vòng thứ i (với $i = 1, 2, 3$) là hoán vị và nén 8 bits của KL_i và KR_i lại thành 6 bits.

Đặt 8 bits khi ghép $KL_i \parallel KR_i$ là $k_0k_1k_2k_3k_4k_5k_6k_7$, kết quả là 6 bits $k_5k_1k_3k_2k_7k_0$.

tinydes.py

```
# tindexs.py

sbox = [
    0xE, 0x4, 0xD, 0x1, 0x2, 0xF, 0xB, 0x8, 0x3, 0xA, 0x6, 0xC, 0x5, 0x9, 0x0, 0x7,
    0x0, 0xF, 0x7, 0x4, 0xE, 0x2, 0xD, 0x1, 0xA, 0x6, 0xC, 0xB, 0x9, 0x5, 0x3, 0x8,
    0x4, 0x1, 0xE, 0x8, 0xD, 0x6, 0x2, 0xB, 0xF, 0xC, 0x9, 0x7, 0x3, 0xA, 0x5, 0x0,
    0xF, 0xC, 0x8, 0x2, 0x4, 0x9, 0x1, 0x7, 0x5, 0xB, 0x3, 0xE, 0xA, 0x0, 0x6, 0xD
]

def Xor(a: list[int], b: list[int]) -> list[int]:
    return [x^y for x, y in zip(a, b)]

def Expand(R: list[int]) -> list[int]:
    return [R[2], R[3], R[1], R[2], R[1], R[0]]

def SBox(R: list[int]) -> list[int]:
    row = int("".join(map(str, [R[0], R[5]])), 2)
    col = int("".join(map(str, R[1:5])), 2)

    return list(map(int, format(sbox[row*16 + col], "04b")))

def PBox(R: list[int]) -> list[int]:
    return [R[2], R[0], R[3], R[1]]

def PBox_inv(R: list[int]) -> list[int]:
    return [R[1], R[3], R[0], R[2]]

def Compress(K: list[int], round: int) -> list[int]:
    left, right = K[:4], K[4:]
    if round == 0 or round == 2:
        left = left[1:] + left[:1]
        right = right[1:] + right[:1]
    elif round == 1:
        left = left[2:] + left[:2]
        right = right[2:] + right[:2]

    Ki = left + right
    return left, right, [Ki[5], Ki[1], Ki[3], Ki[2], Ki[7], Ki[0]]

def encrypt_block(plaintext: list[int], key: list[int]) -> list[int]:
    keys = [key]
    left, right = key[:4], key[4:]
    for i in range(3):
        left, right, key = Compress(left + right, i)
        keys.append(key)
```

```

left, right = plaintext[:4], plaintext[4:]
for i in range(3):
    left, right = right, Xor(left, PBox(SBox(Xor(Expand(right), keys[i+1]))))

return left + right

#print(encrypt_block([0, 1, 0, 1, 1, 0, 0], [1, 0, 0, 1, 1, 0, 1]))

```

Phá mã vi sai trên TinyDES

Giả sử X_1 và X_2 là hai khối input có cùng số bit.

Ta định nghĩa vi sai của X_1 và X_2 là $X = X_1 \oplus X_2$.

Xét các phép biến đổi trong TinyDES

Phép XOR key

Gọi K là khóa con ở vòng nào đó trong thuật toán. Khi đó nếu đặt $Y_1 = X_1 \oplus K$ và $Y_2 = X_2 \oplus K$ thì vi sai của output là $Y = Y_1 \oplus Y_2 = X_1 \oplus X_2$. Như vậy K không tác động lên vi sai và đây là tính chất quan trọng để chúng ta phá mã vi sai.

Phép PBox

Phép PBox bảo toàn số bit (hoán vị 4 bits thành 4 bits) và cách xây dựng hoán vị là một biến đổi tuyến tính. Việc hoán vị 4 bits $b_0 b_1 b_2 b_3$ thành $b_2 b_0 b_3 b_1$ tương đương với phép nhân ma trận

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} b_2 \\ b_0 \\ b_3 \\ b_1 \end{pmatrix}$$

Do đó nếu đặt $Y_1 = \text{PBox}(X_1)$ và $Y_2 = \text{PBox}(X_2)$ thì

$$Y_1 \oplus Y_2 = \text{PBox}(X_1) \oplus \text{PBox}(X_2) = \text{PBox}(X_1 \oplus X_2).$$

Như vậy nếu vi sai input là cố định thì vi sai output cũng cố định do tính tuyến tính.

Phép Expand

Tương tự, phép Expand cũng là biến đổi tuyến tính và nếu đặt $Y_1 = \text{Expand}(X_1)$ và $Y_2 = \text{Expand}(X_2)$ thì

$$Y_1 \oplus Y_2 = \text{Expand}(X_1) \oplus \text{Expand}(X_2) = \text{Expand}(X_1 \oplus X_2).$$

Cũng tương tự, nếu vi sai input là cố định thì vi sai output cũng cố định.

Phép SBox

Phép SBox là một biến đổi không tuyến tính với input 6 bits và output 4 bits.

Đặt $Y_1 = \text{SBox}(X_1)$ và $Y_2 = \text{SBox}(X_2)$.

Với mỗi $X = X_1 \oplus X_2$ cố định thì cứ một giá trị X_1 sẽ có duy nhất một giá trị X_2 cho ra vi sai X . Tuy nhiên vi sai output $Y_1 \oplus Y_2$ phân bố không đều nhau.

Thực hiện bruteforce đơn giản trên SBox với vi sai input $X = X_1 \oplus X_2$ có 6 bits, ta tìm được sự phân bố vi sai output $Y = Y_1 \oplus Y_2$.

Chúng ta mong muốn rằng trên một hàng có càng ít phần tử khác 0 càng tốt. Từ đó sự phân bố xác suất sẽ dễ kiểm soát hơn.

check_sbox.py

```
# check_sbox.py

import tinydes

def int_to_vec6(n: int) -> list[int]:
    return list(map(int, format(n, "06b")))

def int_to_vec8(n: int) -> list[int]:
    return list(map(int, format(n, "08b")))

def vec_to_int(v: list[int]) -> int:
    return int("".join(map(str, v)), 2)

# Know about distribution of differential input-output
dist = []
for _ in range(2**6):
    X = int_to_vec6(_)
    row = [0] * 16
    for __ in range(2**6):
        X1 = int_to_vec6(__)
        X2 = tinydes.Xor(X, X1)
        Y1 = tinydes.SBox(X1)
        Y2 = tinydes.SBox(X2)
        Y = tinydes.Xor(Y1, Y2)
        row[vec_to_int(Y)] += 1
    dist.append(row)

for i, row in enumerate(dist):
    print(f'Row = {row}')
    print(f'Row {i} has {row.count(0)} zero elements')
    print(f'Element that has maximal probability is {row.index(max(row))} with prob
→{max(row)}')
    print()
```

Sau khi xem bảng phân phối vi sai input và output ta có thể thấy được rằng:

1. Nếu vi sai input $X = 0$ thì chắc chắn vi sai output $Y = 0$.
2. Nếu vi sai input $X = 16$ thì có 9 vi sai output Y khác 0.
3. Nếu vi sai input $X = 52$ thì có 8 vi sai output Y khác 0.

Dựa trên nhận xét này, chúng ta sẽ tấn công trên các X_1, X_2 mà $X = X_1 \oplus X_2 \in \{16, 52\}$.

Xét hai hàng 16 và 52, ta thấy rằng:

1. Nếu vi sai input $X = 16$ thì vi sai output $Y = 7$ là cao nhất với xác suất $14/64$.
2. Nếu vi sai input $X = 52$ thì vi sai output $Y = 2$ là cao nhất với xác suất $16/64$.

Hàm F

Như vậy, phép XOR key, phép PBox và phép Expand cho xác suất đều nhau với các cặp vi sai (X, Y) . Trong khi đó thì SBox lại cho xác suất các cặp vi sai (X, Y) không đều nhau.

Đặt $Y_1 = F(X_1)$ và $Y_2 = F(X_2)$.

Đi từ trong ra ngoài (Expand, tới SBox, tới PBox và cuối cùng là F) ta thấy rằng:

- vi sai input của hàm F chính là vi sai input của Expand;
- vi sai output của Expand là vi sai input của SBox (không phụ thuộc vào khóa);
- vi sai output của SBox là vi sai input của PBox;
- vi sai output của PBox là vi sai output của hàm F .

Ta có thể đưa ra nhận xét về xác suất vi sai output $Y = Y_1 \oplus Y_2$ từ vi sai $X = X_1 \oplus X_2$ như sau:

1. Nếu vi sai input của F là 0 \Rightarrow vi sai output của Expand là 0 \Rightarrow vi sai output của SBox chắc chắn là 0 \Rightarrow vi sai output của PBox chắc chắn là 0 \Rightarrow vi sai output của hàm F chắc chắn là 0.
2. Nếu vi sai input của F là 1 \Rightarrow vi sai output của Expand là 16 \Rightarrow vi sai output của SBox là 7 với xác suất $14/64$ \Rightarrow vi sai output của PBox là 11 với xác suất là $14/64$ \Rightarrow hàm F là 11 với xác suất $14/64 = 7/32$.
3. Nếu vi sai input của F là 3 \Rightarrow vi sai output của Expand là 52 \Rightarrow vi sai output của SBox là 2 với xác suất $16/64$ \Rightarrow vi sai output của PBox là 8 với xác suất $16/64$ \Rightarrow hàm F là 8 với xác suất $16/64 = 1/4$.

Nói chung, chúng ta chọn output của Expand (input cho SBox) giống với xác suất cao nhất với phân tích SBox ở trên kia.

Chosen plaintext

Differential attack là một dạng chosen plaintext, trong đó chúng ta tận dụng các xác suất ở trên.

Chosen plaintext phần một

Do tính chất vi sai, chúng ta sẽ mong muốn tìm những cặp (plaintext, ciphertext) (P_1, C_1) và (P_2, C_2) nào đó mà vi sai input $P_1 \oplus P_2$ và vi sai output $C_1 \oplus C_2$ có thể tối ưu xác suất trên.

Giả sử chúng ta xét trường hợp 3 ở trên, khi vi sai input của F là 3 thì vi sai output của F là 8 với xác suất $1/4$. Đây là xác suất lớn nhất nên ta mong muốn trong 3 vòng của TinyDES sẽ tận dụng được càng nhiều càng tốt.

Chúng ta đi từ dưới lên. Đặt

$$L_3 = R_2, \quad R_3 = L_2 \oplus F(R_2, K_3),$$

và

$$L'_3 = R'_2, \quad R'_3 = L'_2 \oplus F(R'_2, K_3).$$

Chọn $R_2 \oplus R'_2 = 3$ thì $F(R_2, K_3) \oplus F(R'_2, K_3) = 8$ với xác suất $1/4$. Vì là bước cuối nên ta hy vọng ciphertext cuối cùng sẽ càng ít phức tạp càng tốt. Do đó có thể lựa chọn $L_2 = L'_2 = 0$ để bảo toàn vi sai sau khi vòng 3 kết thúc.

Ở vòng 3, xác suất để vi sai input bằng 3 và vi sai output bằng 8 là $1/4$.

Tiếp theo, đặt:

$$L_2 = R_1, \quad R_2 = L_1 \oplus F(R_1, K_2),$$

và

$$L'_2 = R'_1, \quad R'_2 = L'_1 \oplus F(R'_1, K_2).$$

Do $L_2 = R_1$ nên $R_1 = 0$. Tương tự $R'_1 = L'_2 = 0$. Điều đáng chú ý là $R_1 \oplus R'_1 = 0$, nghĩa là vi sai input bằng 0, nên vi sai output $F(R_1, K_2) \oplus F(R'_1, K_2)$ chắc chắn bằng 0 (xác suất bằng 1).

Ở vòng 2, xác suất để vi sai input bằng 0 và vi sai output bằng 0 là 1.

Ta lại có

$$R_2 \oplus R'_2 = 3 = L_1 \oplus F(R_1, K_2) \oplus L'_1 \oplus F(R'_1, K_2) = L_1 \oplus L'_1,$$

do vi sai output hàm F chắc chắn bằng 0. Do đó $L_1 \oplus L'_1 = 3$.

Tuy nhiên, $L_1 = R_0$ và $L'_1 = R'_0$ nên $L_1 \oplus L'_1 = R_0 \oplus R'_0 = 3$. Do đó vi sai output của hàm F là $F(R_0, K_1) \oplus F(R'_0, K_1) = 8$ có xác suất $1/4$.

Ở vòng 1, xác suất để vi sai input bằng 3 và vi sai output bằng 8 là $1/4$.

Cuối cùng,

$$R_1 \oplus R'_1 = L_0 \oplus F(R_0, K_1) \oplus L'_0 \oplus F(R'_0, K_1) = L_0 \oplus L'_0 \oplus 8,$$

mà ta nhớ lại ở trên $R_1 \oplus R'_1 = 0$ nên suy ra $L_0 \oplus L'_0 = 8$.

Tổng kết lại, ta chọn vi sai input $(L, R) = (8, 3)$ thì xác suất để vi sai output bằng $(3, 8)$ là $(1/4) \times 1 \times (1/4) = 1/16$. Đây là xác suất cao nhất có thể sau khi TinyDES chạy đủ 3 vòng.

Chosen plaintext phần hai

Tương tự, chúng ta xét trường hợp 2 ở trên, khi vi sai input của F là 1 thì vi sai output của F là 11 với xác suất $7/32$. Ta cũng mong muốn sau 3 vòng của TinyDES sẽ tận dụng được càng nhiều càng tốt.

Chúng ta lại đi dưới lên. Đặt

$$L_3 = R_2, \quad R_3 = L_2 \oplus F(R_2, K_3)$$

và

$$L'_3 = R'_2, \quad R'_3 = L'_2 \oplus F(R'_2, K_3).$$

Chọn $R_2 \oplus R'_2 = 1$ thì $F(R_2, K_3) \oplus F(R'_2, K_3) = 11$ với xác suất $7/32$. Vì là bước cuối cùng nên ta cũng hy vọng ciphertext sẽ càng ít phức tạp càng tốt. Do đó ta chọn $L_2 = L'_2 = 0$.

Ở vòng 3, xác suất để vi sai input bằng 1 và vi sai output bằng 11 là $7/32$.

Tiếp theo, đặt

$$L_2 = R_1, \quad R_2 = L_1 \oplus F(R_1, K_2),$$

và

$$L'_2 = R'_1, \quad R'_2 = L'_1 \oplus F(R'_1, K_2).$$

Do $L_2 = R_1$ nên $R_1 = 0$. Tương tự $R'_1 = 0$. Suy ra $R_1 \oplus R'_1 = 0$ và vi sai output $F(R_1, K_2) \oplus F(R'_1, K_2) = 0$ với xác suất bằng 1 (chắc chắn xảy ra).

Ở vòng 2, xác suất để vi sai input bằng 0 và vi sai output bằng 0 là 1.

Ta lại có:

$$R_2 \oplus R'_2 = L_1 \oplus F(R_1, K_2) \oplus L'_1 \oplus F(R'_1, K_2) = L_1 \oplus L'_1,$$

do vi sai output của hàm F chắc chắn bằng 0. Do đó $L_1 \oplus L'_1 = 1$.

Tuy nhiên $L_1 = R_0$ và $L'_1 = R'_0$ nên $R_0 \oplus R'_0 = L_1 \oplus L'_1 = 1$. Do đó vi sai output của hàm F ở vòng 1 là $F(R_0, K_1) \oplus F(R'_0, K_1) = 11$ với xác suất $7/32$.

Ở vòng 1, xác suất để vi sai input bằng 1 và vi sai output bằng 11 là $7/32$.

Cuối cùng, do

$$R_1 \oplus R'_1 = L_0 \oplus F(R_0, K_1 \oplus L'_0 \oplus F(R'_0, K_1)),$$

mà $R_1 \oplus R'_1 = 0$ và $F(R_0, K_1) \oplus F(R'_0, K_1) = 11$ nên $L_0 \oplus L'_0 = 11$.

Tổng kết lại, ta chọn vi sai input $(L, R) = (11, 1)$ thì xác suất để vi sai output bằng $(1, 11)$ là $(7/32) \times 1 \times (7/32) \approx 0.048$. Đây cũng là xác suất cao nhất có thể sau khi TinyDES chạy đủ 3 vòng.

Final attack

Như vậy, đối với TinyDES chúng ta phá mã vi sai như sau:

1. Tìm một số lượng cặp plaintext, ciphertext $(P_1, C_1), (P_2, C_2), \dots$ cho tới khi tìm được $P_i \oplus P_j = 0x83$ và $C_i \oplus C_j = 0x38$.
2. Tìm một số lượng cặp plaintext, ciphertext $(P'_1, C'_1), (P'_2, C'_2), \dots$ cho tới khi tìm được $P'_i \oplus P'_j = 0xB1$ và $C'_i \oplus C'_j = 0x1B$.

Sau khi đã tìm được một số lượng cặp plaintext, ciphertext thỏa vi sai trên, nhớ lại hàm F ở vòng 3, do

$$L_3 = R_2, \text{ và } R_3 = L_2 \oplus F(R_2, K_3) = L_2 \oplus F(L_3, K_3) = F(L_3, K_3),$$

theo cách chọn $L_2 = 0$ ở trên, dễ thấy rằng chúng ta có thể tìm được các K_3 thỏa mãn hàm F ở vòng 3.

Để làm điều đó thì ta tính $O = \text{Expand}(L_3)$, và do $\text{SBox}(O \oplus K_3) = \text{PBox}^{-1}(R_3)$ cũng tính được nên có thể tìm các giá trị $O \oplus K_3$ mà khi đi qua SBox cho kết quả bằng $\text{PBox}^{-1}(R_3)$. Sau đó XOR lại cho O thì sẽ tìm được các giá trị có thể của K_3 . Lưu ý rằng SBox làm giảm 6 bits còn 4 bits nên sẽ có nhiều giá trị khác nhau cho cùng giá trị SBox .

Thực hiện trên hai trường hợp vi sai input-output là $(0x83, 0x38)$ và $(0xB1, 0x1B)$ ta có tập các giá trị có thể xảy ra của K_3 .

Theo thuật toán sinh khóa con thì với khóa K 8 bits ban đầu, đặt là $k_0 k_1 k_2 k_3 k_4 k_5 k_6 k_7$, thì khóa con K_3 là $k_5 k_1 k_3 k_2 k_7 k_0$. Trong K_3 không có k_4 và k_6 nên chúng ta sẽ bruteforce hai bit này tới khi tìm được đúng khóa K mà tương ứng với cặp (P_i, C_i) .

find_key.py

```
# find_key.py

import tinydes
from itertools import product

def int_to_vec6(n: int) -> list[int]:
```

```

    return list(map(int, format(n, "06b")))

def int_to_vec8(n: int) -> list[int]:
    return list(map(int, format(n, "08b")))

def vec_to_int(v: list[int]) -> int:
    return int("".join(map(str, v)), 2)

def recover_key(k: list[int]) -> list[int]:
    return [k[5], k[1], k[3], k[2], 0, k[0], 0, k[4]]

key = [1, 0, 0, 1, 1, 0, 1, 0]

ptx = []
ctx = []

pt, ct = [0, 1, 0, 1, 1, 1, 0, 0], [1, 0, 0, 1, 1, 0, 1, 0]

candidates = []
K3 = []

for _ in range(24):
    pt = int_to_vec8(_)
    ct = tinydes.encrypt_block(pt, key)
    pt_ = tinydes.Xor(int_to_vec8(0x83), pt)
    ct_ = tinydes.encrypt_block(pt_, key)
    if tinydes.Xor(ct_, ct) == list(map(int, format(0x38, "08b"))):
        ptx.append(pt_)
        ctx.append(ct_)
        candidates.append((pt, pt_))
        break

for pt1, pt2 in candidates:
    o1, o2 = tinydes.PBox_inv(pt1[4:]), tinydes.PBox_inv(pt2[4:])
    q1, q2 = tinydes.Expand(pt1[:4]), tinydes.Expand(pt2[:4])
    for i in range(len(tinydes.sbox)):
        if tinydes.sbox[i] == vec_to_int(o1):
            row, col = i // 16, i % 16
            idx = [row // 2] + list(map(int, format(col, "04b"))) + [row % 2]
            K3.append(tinydes.Xor(q1, idx))
        if tinydes.sbox[i] == vec_to_int(o2):
            row, col = i // 16, i % 16
            idx = [row // 2] + list(map(int, format(col, "04b"))) + [row % 2]
            K3.append(tinydes.Xor(q2, idx))

candidates = []

for _ in range(24):
    pt = int_to_vec8(_)

```

```

ct = tinydes.encrypt_block(pt, key)
pt_ = tinydes.Xor(int_to_vec8(0xb1), pt)
ct_ = tinydes.encrypt_block(pt_, key)
if tinydes.Xor(ct_, ct) == list(map(int, format(0x1b, "08b"))):
    ptx.append(pt_)
    ctx.append(ct_)
    candidates.append((pt, pt_))
    break

for pt1, pt2 in candidates:
    o1, o2 = tinydes.PBox_inv(pt1[4:]), tinydes.PBox_inv(pt2[4:])
    q1, q2 = tinydes.Expand(pt1[:4]), tinydes.Expand(pt2[:4])
    for i in range(len(tinydes.sbox)):
        if tinydes.sbox[i] == vec_to_int(o1):
            row, col = i // 16, i % 16
            idx = [row // 2] + list(map(int, format(col, "04b"))) + [row % 2]
            K3.append(tinydes.Xor(q1, idx))
        if tinydes.sbox[i] == vec_to_int(o2):
            row, col = i // 16, i % 16
            idx = [row // 2] + list(map(int, format(col, "04b"))) + [row % 2]
            K3.append(tinydes.Xor(q2, idx))

for k3 in set([vec_to_int(k) for k in K3]):
    k = recover_key(int_to_vec6(k3))
    for k4, k6 in product(range(2), repeat=2):
        k[4], k[6] = k4, k6
        if tinydes.encrypt_block(pt, k) == ct:
            print(f'Recover key: {k}')

```

Difference và differential

Ở phần này, ký hiệu \boxplus là phép cộng modulo 2^n , \boxminus là phép trừ modulo 2^n , và \oplus là toán tử bitwise-XOR.

Nếu số nguyên a có n bit và biểu diễn dưới dạng

$$a = a_0 + 2a_1 + 2^2a_2 + \cdots + 2^{n-1}a_{n-1}$$

thì số nguyên a tương đương với vector

$$(a_0, a_1, a_2, \dots, a_{n-1}) \in \mathbb{F}_2^n.$$

Như vậy, phần tử $a \in \mathbb{Z}_{2^n}$ tương đương phần tử $(a_0, a_1, \dots, a_{n-1}) \in \mathbb{F}_2^n$ và các bạn cần lưu ý rằng phép \boxplus và \boxminus thực hiện trên \mathbb{Z}_{2^n} , còn phép XOR thực hiện trên \mathbb{F}_2^n .

Cụ thể hơn, giả sử

$$\begin{aligned} a &= a_0 + 2a_1 + 2^2a_2 + \cdots + 2^{n-1}a_{n-1}, \\ b &= b_0 + 2b_1 + 2^2b_2 + \cdots + 2^{n-1}b_{n-1}, \end{aligned}$$

thì ta có

$$\begin{aligned} a \boxplus b &= a + b \text{ mod } 2^n, & a \boxminus b &= a - b \text{ mod } 2^n, \\ a \oplus b &= c = c_0 + 2c_1 + \cdots + 2^{n-1}c_{n-1}, \text{ với } c_i = a_i \oplus b_i. \end{aligned}$$

Ví dụ, với $n = 4$, xét các số nguyên 4 bit là $a = 9$ và $b = 11$. Khi đó

$$a \boxplus b = 4, \quad a \boxminus b = 14, \quad a \oplus b = 2.$$

Difference (hiệu)

Giả sử ta có hàm $f : \mathbb{Z}_{2^n} \rightarrow \mathbb{Z}_{2^m}$, khi đó với hai phần tử $a, b \in \mathbb{Z}_{2^n}$ thì ta nói

- $b \boxminus a$ là hiệu đầu vào;
- $f(b) \boxminus f(a)$ là hiệu đầu ra.

Dưới góc độ phá mã, với $\delta \in \mathbb{Z}_{2^n}$ và $\Delta \in \mathbb{Z}_{2^m}$ cố định, ta xem xét có bao nhiêu cặp $a, b \in \mathbb{Z}_{2^n}$ mà

$$b \boxminus a = \delta, \quad f(b) \boxminus f(a) = \Delta.$$

Chuyển về ta có các đẳng thức trên tương với

$$b = a \boxplus \delta \implies f(b) = f(a \boxplus \delta) = f(a) \boxplus \Delta.$$

Nói cách khác, ta xem xét xác suất

$$\text{adp}^f(\delta \mapsto \Delta) = \Pr[f(a \boxplus \delta) = f(a) \boxplus \Delta]$$

với mọi $a \in \mathbb{Z}_{2^n}$.

Tổng quát, nếu ta xét k hiệu đầu vào $\alpha_0, \alpha_1, \dots, \alpha_{k-1}$ và hiệu đầu ra α_k thì ta quan tâm xác suất

$$\text{adp}^f(\alpha_0, \alpha_1, \dots, \alpha_{k-1} \mapsto \alpha_k) = \Pr[f(x_0 \boxplus \alpha_0, x_1 \boxplus \alpha_1, \dots, x_{k-1} \boxplus \alpha_{k-1}) = f(x_0, x_1, \dots, x_{k-1}) \boxplus \alpha_k].$$

Example 1.33

Xét hàm $f(x, y) = x \oplus y$ với $x, y \in \mathbb{Z}_{2^n}$. Với α, β và γ thuộc \mathbb{Z}_{2^n} ta xem xét

$$\text{adp}^f(\alpha, \beta \mapsto \gamma) = \Pr[(x \boxplus \alpha) \oplus (y \boxplus \beta) = (x \oplus y) \boxplus \gamma].$$

Differential (vi sai)

Giả sử ta có hàm $f : \mathbb{Z}_{2^n} \rightarrow \mathbb{Z}_{2^m}$, khi đó với hai phần tử $a, b \in \mathbb{Z}_{2^n}$ thì ta nói

- $b \oplus a$ là vi sai đầu vào;
- $f(b) \oplus f(a)$ là vi sai đầu ra.

Ở đây cần lưu ý rằng, vi sai bản chất là phép trừ, nhưng trên \mathbb{F}_2^n thì phép trừ cũng chính là phép cộng \oplus .

Tương tự, dưới góc độ phá mã, với $\delta \in \mathbb{Z}_{2^n}$ và $\Delta \in \mathbb{Z}_{2^m}$ cố định, ta xem xét có bao nhiêu cặp $a, b \in \mathbb{Z}_{2^n}$ mà

$$b \oplus a = \delta, \quad f(b) \oplus f(a) = \Delta.$$

Chuyển về ta có các đẳng thức trên tương với

$$b = a \oplus \delta \implies f(b) = f(a \oplus \delta) = f(a) \oplus \Delta.$$

Nói cách khác, ta xem xét xác suất

$$\text{xdp}^f(\delta \mapsto \Delta) = \Pr[f(a \oplus \delta) = f(a) \oplus \Delta]$$

với mọi $a \in \mathbb{Z}_{2^n}$.

Tổng quát, nếu ta xét k hiệu đầu vào $\alpha_0, \alpha_1, \dots, \alpha_{k-1}$ và hiệu đầu ra α_k thì ta quan tâm xác suất

$$\text{adp}^f(\alpha_0, \alpha_1, \dots, \alpha_{k-1} \mapsto \alpha_k) = \Pr [f(x_0 \oplus \alpha_0, x_1 \oplus \alpha_1, \dots, x_{k-1} \oplus \alpha_{k-1}) = f(x_0, x_1, \dots, x_{k-1}) \oplus \alpha_k].$$

Example 1.34

Xét hàm $f(x, y) = x \boxplus y$ với $x, y \in \mathbb{Z}_{2^n}$. Với α, β và γ thuộc \mathbb{Z}_{2^n} ta xem xét

$$\text{xdp}^f(\alpha, \beta \mapsto \gamma) = \Pr [(x \oplus \alpha) \boxplus (y \oplus \beta) = (x \boxplus y) \oplus \gamma].$$

Một số lưu ý và nhận xét

Một số nguyên n bit cũng có thể xem xét dưới dạng vector n phần tử, do đó cho phép chúng ta xem xét hai dạng vi sai theo phép XOR và theo phép cộng modulo 2^n .

Ở đây, adp là viết tắt của *addition differential probability* (xác suất vi sai theo phép cộng) và xdp là viết tắt của *xor differential probability* (xác suất vi sai theo phép xor).

Trong nhiều thuật toán mã khối, các phép biến đổi có thể sử dụng phép XOR lẫn phép cộng modulo 2^n , do đó việc nghiên cứu mối liên hệ giữa các toán tử trên nhằm đưa ra đánh giá vi sai nào đạt được xác suất mong muốn là việc quan trọng. Tuy nhiên hiện chưa có nhiều nghiên cứu cho việc này, ví dụ như phép cộng modulo 2^n sẽ có xác suất như thế nào đối với vi sai là phép XOR, và ngược lại.

Lý do đơn giản nhất của ứng dụng vi sai là loại bỏ sự có mặt của khóa trong mỗi vòng.

Ví dụ thứ nhất, rất phổ biến, là hệ mã DES. Nếu ta xét một S-box vòng thì biến đổi trên nửa phải có dạng

$$F(R, K) = \text{SBox}(R \oplus K)$$

với R là nửa phải đầu vào và K là khóa ở vòng hiện tại, thì ta có vi sai đầu vào là

$$\delta = R \oplus R' = (R \oplus K) \oplus (R' \oplus K)$$

và vi sai đầu ra là

$$\Delta = \text{SBox}(R \oplus K) \oplus \text{SBox}(R' \oplus K).$$

Ở đây, vi sai đầu vào không phụ thuộc vào khóa K .

Ví dụ thứ hai là round function của hệ mã Magma. Nếu ta lấy S-box của Magma có dạng

$$F(R, K) = \text{SBox}(R \boxplus K)$$

thì vi sai đầu vào là

$$\delta = R' \boxminus R = (R' \boxplus K) \boxminus (R \boxplus K)$$

và vi sai đầu ra là

$$\Delta = \text{SBox}(R' \boxplus K) \boxminus \text{SBox}(R \boxplus K).$$

Ở đây ta cũng có vi sai đầu vào không phụ thuộc vào khóa K nhưng đối với toán tử hiệu \boxminus , không phải hiệu \oplus .

3.2.2 Phá mã tuyến tính

Phá mã tuyến tính (linear cryptanalysis) đã làm chuẩn mã hóa DES không còn đủ an toàn.

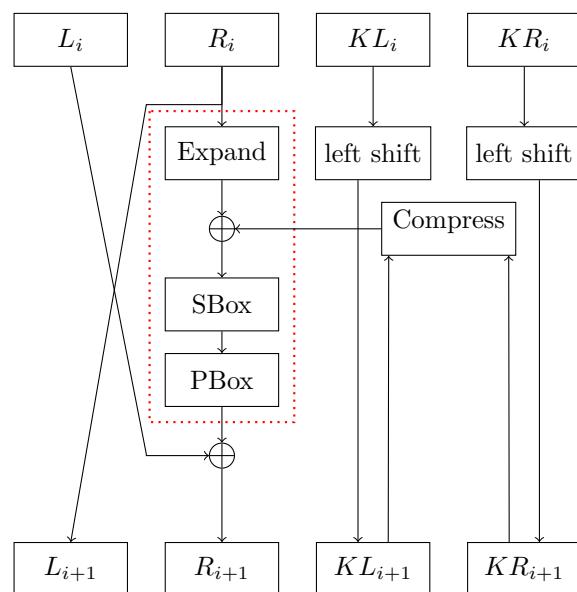
Trong các chuẩn mã hóa hiện đại về sau, khả năng kháng phá mã vi sai và phá mã tuyến tính trở thành tiêu chuẩn đánh giá độ an toàn của thuật toán mã hóa.

Phá mã tuyến tính trên TinyDES

Mô tả TinyDES

Trong phần mô tả phá mã vi sai mình đã sử dụng TinyDES để làm ví dụ. Ở đây mình tiếp tục sử dụng TinyDES để ví dụ cho phá mã tuyến tính. Nhằm gợi nhớ cấu trúc của TinyDES thì mình xin chép lại.

TinyDES là một phiên bản thu nhỏ của chuẩn mã hóa DES. TinyDES là mã hóa khối theo mô hình Feistel, kích thước khối là 8 bit, kích thước khóa cũng là 8 bit. Mỗi vòng khóa con có độ dài 6 bit.



Hình 3.32: Một vòng TinyDES

Mã TinyDES khá đơn giản. Theo mô hình Feistel, khối đầu vào 8 bit được chia thành hai nửa trái phải 4 bit. Nửa phải sẽ đi qua các hàm Expand, SBox và PBox, sau đó XOR với nửa trái để được nửa phải mới. Còn nửa trái mới là nửa phải cũ. Tóm lại công thức mô hình Feistel là:

$$L_{i+1} = R_i, \quad R_{i+1} = L_i \oplus F(R_i, K_{i+1})$$

với $i = 1, 2, 3$ tương ứng 3 vòng với đầu vào của khối là (L_0, R_0) .

Chúng ta cần các động tác sau:

1. Expand: mở rộng và hoán vị R_i từ 4 bits lên 6 bits. Giả sử 4 bits của R_i là $b_0 b_1 b_2 b_3$ thì kết quả sau khi Expand là $b_2 b_3 b_1 b_2 b_1 b_0$.
2. SBox: gọi 6 bits đầu vào là $b_0 b_1 b_2 b_3 b_4 b_5$. Khi đó ta tra cứu theo bảng SBox với $b_0 b_5$ chỉ số **hàng**, $b_1 b_2 b_3 b_4$ chỉ số **cột**. Nói cách khác bảng SBox có 4 hàng, 16 cột. Kết quả của SBox là một số 4 bit.
3. PBox: là hàm hoán vị 4 bits $b_0 b_1 b_2 b_3$ thành $b_2 b_0 b_3 b_1$.

Như vậy, hàm F của mô hình Feistel đối với mã TinyDES là:

$$F(R_i, K_i) = \text{PBox}(\text{SBox}(\text{Expand}(R_i) \oplus K_{i+1})).$$

Để sinh khóa con cho 3 vòng, khóa ban đầu được chia thành hai nửa trái phải lần lượt là KL_0 và KR_0 . TinyDES thực hiện như sau:

1. Vòng 1: KL_0 và KR_0 được dịch vòng trái 1 bit để được KL_1 và KR_1 ;
2. Vòng 2: KL_1 và KR_1 được dịch vòng trái 2 bit để được KL_2 và KR_2 ;
3. Vòng 3: KL_2 và KR_2 được dịch vòng trái 1 bit để được KL_3 và KR_3 .

Khi đó, khóa K_i ở vòng thứ i (với $i = 1, 2, 3$) là hoán vị và nén 8 bits của KL_i và KR_i lại thành 6 bits.

Đặt 8 bits khi ghép $KL_i \parallel KR_i$ là $k_0 k_1 k_2 k_3 k_4 k_5 k_6 k_7$, kết quả là 6 bits $k_5 k_1 k_3 k_2 k_7 k_0$.

❶ tinydes.py

```
# tindes.py

sbox = [
    0xE, 0x4, 0xD, 0x1, 0x2, 0xF, 0xB, 0x8, 0x3, 0xA, 0x6, 0xC, 0x5, 0x9, 0x0, 0x7,
    0x0, 0xF, 0x7, 0x4, 0xE, 0x2, 0xD, 0x1, 0xA, 0x6, 0xC, 0xB, 0x9, 0x5, 0x3, 0x8,
    0x4, 0x1, 0xE, 0x8, 0xD, 0x6, 0x2, 0xB, 0xF, 0xC, 0x9, 0x7, 0x3, 0xA, 0x5, 0x0,
    0xF, 0xC, 0x8, 0x2, 0x4, 0x9, 0x1, 0x7, 0x5, 0xB, 0x3, 0xE, 0xA, 0x0, 0x6, 0xD
]

def Xor(a: list[int], b: list[int]) -> list[int]:
    return [x^y for x, y in zip(a, b)]

def Expand(R: list[int]) -> list[int]:
    return [R[2], R[3], R[1], R[2], R[1], R[0]]

def SBox(R: list[int]) -> list[int]:
    row = int("".join(map(str, [R[0], R[5]])), 2)
    col = int("".join(map(str, R[1:5])), 2)

    return list(map(int, format(sbox[row*16 + col], "04b")))

def PBox(R: list[int]) -> list[int]:
    return [R[2], R[0], R[3], R[1]]

def PBox_inv(R: list[int]) -> list[int]:
    return [R[1], R[3], R[0], R[2]]

def Compress(K: list[int], round: int) -> list[int]:
    left, right = K[:4], K[4:]
    if round == 0 or round == 2:
        left = left[1:] + left[:1]
    else:
        left = left[1:-1] + left[-1] + left[:-1]
```

```

        right = right[1:] + right[:1]
    elif round == 1:
        left = left[2:] + left[:2]
        right = right[2:] + right[:2]

    Ki = left + right
    return left, right, [Ki[5], Ki[1], Ki[3], Ki[2], Ki[7], Ki[0]]


def encrypt_block(plaintext: list[int], key: list[int]) -> list[int]:
    keys = [key]
    left, right = key[:4], key[4:]
    for i in range(3):
        left, right, key = Compress(left + right, i)
        keys.append(key)

    left, right = plaintext[:4], plaintext[4:]
    for i in range(3):
        left, right = right, Xor(left, PBox(SBox(Xor(Expand(right), keys[i+1])))))

    return left + right

#print(encrypt_block([0, 1, 0, 1, 1, 1, 0, 0], [1, 0, 0, 1, 1, 0, 1, 0]))

```

Phá mã tuyến tính trên TinyDES

Trong các phép biến đổi trên TinyDES thì chỉ có SBox là không tuyến tính. Tuy nhiên nếu chỉ xét một số bit nhất định giữa đầu vào và đầu ra thì ta có quan hệ tuyến tính.

Nhắc lại, một biến đổi $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ gọi là tuyến tính nếu với mọi $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{F}_2^n$ ta đều có

$$f(\mathbf{x}_1 \oplus \mathbf{x}_2) = f(\mathbf{x}_1) \oplus f(\mathbf{x}_2).$$

Ta sẽ xét các phép biến đổi trong TinyDES.

Phép XOR key

Gọi K là khóa con ở vòng nào đó trong thuật toán. Khi đó nếu đặt $Y_1 = X_1 \oplus K$ và $Y_2 = X_2 \oplus K$ thì ta có $Y_1 \oplus Y_2 = X_1 \oplus X_2$. Như vậy phép XOR là biến đổi tuyến tính.

Phép PBox

Phép PBox bảo toàn số bit (hoán vị 4 bits thành 4 bits) và cách xây dựng hoán vị là một biến đổi tuyến tính. Việc hoán vị 4 bits $b_0 b_1 b_2 b_3$ thành $b_2 b_0 b_3 b_1$ tương đương với phép nhân ma trận

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} b_2 \\ b_0 \\ b_3 \\ b_1 \end{pmatrix}.$$

Do đó nếu đặt $Y_1 = \text{PBox}(X_1)$ và $Y_2 = \text{PBox}(X_2)$ thì

$$Y_1 \oplus Y_2 = \text{PBox}(X_1) \oplus \text{PBox}(X_2) = \text{PBox}(X_1 \oplus X_2).$$

Phép Expand

Tương tự, phép Expand cũng là biến đổi tuyến tính và nếu đặt $Y_1 = \text{Expand}(X_1)$ và $Y_2 = \text{Expand}(X_2)$ thì

$$Y_1 \oplus Y_2 = \text{Expand}(X_1) \oplus \text{Expand}(X_2) = \text{Expand}(X_1 \oplus X_2).$$

Phép SBox

Phép SBox là một biến đổi không tuyến tính với input 6 bits và output 4 bits.

Đặt $\mathbf{y} = \text{SBox}(\mathbf{x})$ với $\mathbf{x} = (x_0, x_1, x_2, x_3, x_4, x_5) \in \mathbb{F}_2^6$ và $\mathbf{y} = (y_0, y_1, y_2, y_3) \in \mathbb{F}_2^4$.

Gọi $\mathbf{a} = (a_0, a_1, a_2, a_3, a_4, a_5) \in \mathbb{F}_2^6$ với biểu diễn thập phân là các số từ 1 tới 63, nghĩa là trừ vector không.

Tương tự, gọi $\mathbf{b} = (b_0, b_1, b_2, b_3) \in \mathbb{F}_2^4$ với biểu diễn thập phân là các số từ 1 tới 15, ta cũng không xét vector không.

Tích vô hướng là một biểu diễn tuyến tính giữa \mathbf{a} và \mathbf{x} :

$$\langle \mathbf{a}, \mathbf{x} \rangle = a_0x_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus a_4x_4 \oplus a_5x_5 \oplus a_6x_6.$$

Tương tự, quan hệ tuyến tính giữa \mathbf{b} và \mathbf{y} là

$$\langle \mathbf{b}, \mathbf{y} \rangle = b_0y_0 \oplus b_1y_1 \oplus b_2y_2 \oplus b_3y_3.$$

Lúc này ta sẽ quan tâm xem với các vector \mathbf{a} và \mathbf{b} nào sẽ khiến nhiều bit của \mathbf{y} phụ thuộc tuyến tính vào các bit của \mathbf{x} , cụ thể là khi $\langle \mathbf{x}, \mathbf{a} \rangle = \langle \mathbf{y}, \mathbf{b} \rangle$.

Với hai vector $\mathbf{a} \in \mathbb{F}_2^6$ và $\mathbf{b} \in \mathbb{F}_2^4$, gọi $S(\mathbf{a}, \mathbf{b})$ là số lượng cặp vector (\mathbf{x}, \mathbf{y}) sao cho

$$\langle \mathbf{x}, \mathbf{a} \rangle = \langle \mathbf{y}, \mathbf{b} \rangle.$$

Bảng dưới liệt kê các giá trị $S(\mathbf{a}, \mathbf{b}) - 32$ với hàng đầu là các vector \mathbf{b} từ 1 tới 15, và cột đầu là các vector \mathbf{a} từ 1 tới 32.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	-2	6	2	2	4	-4	2	6	0	-4	0	-4	-6	-18
3	0	-2	6	-2	6	0	0	-2	2	4	0	0	4	2	-2
4	-4	-6	2	-2	2	0	0	4	-4	-6	-2	6	-2	-4	0
5	4	-2	-2	-2	2	-4	12	4	4	-2	-6	-2	6	0	4
6	4	0	0	8	4	4	-4	2	-2	6	-2	2	6	2	2
7	4	-4	-4	4	0	4	-4	-10	2	-2	6	2	6	-2	-2
8	-2	-2	0	-2	8	-4	-6	2	4	0	-2	-4	2	-6	12
9	2	2	0	-6	0	4	-10	2	0	4	6	-8	2	2	0
10	2	-8	6	0	-2	4	-2	4	6	-4	2	4	2	0	2
11	-2	4	6	-8	2	0	-2	8	-2	4	6	8	-6	0	-2
12	2	0	2	0	6	0	6	-2	0	2	-4	6	-4	2	0
13	-2	8	-2	-4	-2	4	-2	6	-4	2	-8	2	12	6	0
14	-2	2	0	2	4	0	2	-4	-2	-6	4	2	0	-4	2
15	-6	-6	-4	10	0	0	-2	0	6	-2	4	-2	-8	8	2
16	2	-2	4	-2	0	-4	-6	-2	0	0	-2	-4	6	6	4
17	-2	2	4	-2	-4	0	10	2	0	0	10	0	6	6	0
18	-6	-4	2	0	2	0	6	4	2	4	6	0	6	4	-10
19	-2	0	-6	-4	-6	0	2	-4	-2	0	6	-4	-2	4	2

continues on next page

Bảng 3.3 – continued from previous page

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
20	-2	0	-2	0	-2	8	-2	-2	0	6	0	2	4	2	4
21	2	0	2	0	-6	0	2	2	-8	2	0	-2	-4	-2	-4
22	-2	-2	-4	-6	0	4	2	0	-2	-2	4	2	0	4	2
23	2	6	8	6	0	0	2	0	2	6	0	-2	0	0	2
24	0	0	0	0	4	0	-4	0	-4	4	0	4	4	0	-8
25	0	8	0	4	0	4	0	4	-8	-8	4	-4	-4	0	0
26	4	2	-2	2	2	0	0	6	2	-4	0	0	0	2	2
27	4	2	6	2	2	8	0	6	-6	-4	0	-8	0	2	2
28	4	2	2	-2	6	0	-4	0	4	2	2	-2	-2	0	4
29	-4	6	6	2	2	-8	4	-4	0	-6	2	6	6	4	0
30	0	4	0	0	-4	0	0	2	6	-2	-2	-2	-2	-2	2
31	0	-8	-4	0	4	4	4	2	-2	2	2	-2	-2	2	-2
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

❶ check_sbox.py

```
# check_sbox.py

from tinydes import SBox

S = [[0 for _ in range(15)] for __ in range(63)]

for a in range(1, 64):
    va = list(map(int, f"{a:06b}"))[::-1]
    for b in range(1, 16):
        vb = list(map(int, f"{b:04b}"))[::-1]
        for x in range(64):
            vx = list(map(int, f"{x:06b}"))[::-1]
            vy = SBox(vx)
            u = sum(i * j for i, j in zip(va, vx)) % 2
            v = sum(i * j for i, j in zip(vb, vy)) % 2
            if u == v:
                S[a - 1][b - 1] += 1

print(S)
```

Sau khi xem bảng phân phối $S(a, b)$ thì chúng ta quan tâm một số giá trị.

Xét $S(16, 15) = 14$, tương ứng với vector $\mathbf{a} = (0, 1, 0, 0, 0, 0)$ và $\mathbf{b} = (1, 1, 1, 1)$, thì

$$x_1 = y_0 \oplus y_1 \oplus y_2 \oplus y_3$$

với xác suất 14/64.

Ngược lại ta cũng có

$$x_1 \neq y_0 \oplus y_1 \oplus y_2 \oplus y_3$$

với xác suất 1 – 14/64.

Ta kí hiệu mỗi quan hệ này là

$$\mathbf{y}[0, 1, 2, 3] = \mathbf{x}[1].$$

Hàm F

Xét $\mathbf{y} = F(\mathbf{x}, \mathbf{k})$ là round function của TinyDES, trong đó

- $\mathbf{x} = (x_0, x_1, x_2, x_3) \in \mathbb{F}_2^4$ là đầu vào cho round function (nửa phải);
- $\mathbf{k} = (k_0, k_1, k_2, k_3, k_4, k_5, k_6) \in \mathbb{F}_2^6$ là khóa con ở vòng nào đó;
- $\mathbf{y} = (y_0, y_1, y_2, y_3) \in \mathbb{F}_2^4$ là đầu ra của round function.

Ta có các động tác biến đổi sau.

Hàm Expand:

$$(x_0, x_1, x_2, x_3) \xrightarrow{\text{Expand}} (x_2, x_3, x_1, x_2, x_1, x_0).$$

Hàm XOR key:

$$(x_2, x_3, x_1, x_2, x_1, x_0) \oplus (k_0, k_1, k_2, k_3, k_4, k_5) \rightarrow (x'_0, x'_1, x'_2, x'_3, x'_4, x'_5).$$

Hàm SBox:

$$\mathbf{x}' = (x'_0, x'_1, x'_2, x'_3, x'_4, x'_5) \xrightarrow{\text{SBox}} \mathbf{y}' = (y'_0, y'_1, y'_2, y'_3).$$

Hàm PBox:

$$(y'_0, y'_1, y'_2, y'_3) \xrightarrow{\text{PBox}} (y'_2, y'_0, y'_3, y'_1) \equiv (y_0, y_1, y_2, y_3).$$

Theo phân tích tuyến tính ở trên ta tập trung vào phần SBox, như vậy

$$\mathbf{y}'[0, 1, 2, 3] = \mathbf{x}'[1].$$

Từ PBox suy ra $y_0 = y'_2$, $y_1 = y'_0$, $y_2 = y'_3$ và $y_3 = y'_1$, nên suy ra

$$\mathbf{y}'[0, 1, 2, 3] = \mathbf{y}[0, 1, 2, 3].$$

Điều này có vẻ khá rõ ràng vì tuyến tính $y_0 \oplus y_1 \oplus y_2 \oplus y_3$ có mặt ở mọi bit nên \mathbf{y}' hay \mathbf{y} đều như nhau. Tuy nhiên nếu trong các trường hợp tuyến tính không có đủ tất cả bit là 1 như $b \neq 15$ thì chúng ta cần chú ý sự biến đổi của PBox.

Tiếp theo, do $\mathbf{x}'[1] = x'_1 = x_3 \oplus k_1$ nên có thể suy ra quan hệ tuyến tính giữa đầu vào \mathbf{x} và \mathbf{y} là

$$\mathbf{y}[0, 1, 2, 3] = \mathbf{x}[3] \oplus \mathbf{k}[1].$$

Known-plaintext

Linear attack là một dạng known-plaintext, trong đó chúng ta tận dụng các xác suất ở trên.

Phụ thuộc tuyến tính giữa các vòng

Gọi $P = (L_0, R_0)$ là plaintext ban đầu với L_0 và R_0 là hai nửa trái phải.

Ở vòng 1 ta có

$$L_1 = R_0, \quad R_1 = L_0 \oplus F(R_0, K_1),$$

suy ra

$$F(R_0, K_1) = R_1[0, 1, 2, 3] \oplus L_0[0, 1, 2, 3] = R_0[3] \oplus K_1[1] \quad (3.16)$$

với xác suất $14/64$. Ngược lại ta có

$$F(R_0, K_1) = R_1[0, 1, 2, 3] \oplus L_0[0, 1, 2, 3] = R_0[3] \oplus K_1[1] \oplus 1 \quad (3.17)$$

với xác suất $1 - 14/64$.

Ở vòng 2 ta có

$$L_2 = R_1, \quad R_2 = L_1 \oplus F(R_1, K_2).$$

Ở vòng 3 ta có

$$L_3 = R_2, \quad R_3 = L_2 \oplus F(R_2, K_3),$$

suy ra

$$F(R_2, K_3) = R_3[0, 1, 2, 3] \oplus L_2[0, 1, 2, 3] = R_2[3] \oplus K_3[1] \quad (3.18)$$

với xác suất $14/64$. Tương tự ta cũng có

$$F(R_2, K_3) = R_3[0, 1, 2, 3] \oplus L_2[0, 1, 2, 3] = R_2[3] \oplus K_3[1] \oplus 1 \quad (3.19)$$

với xác suất $1 - 14/64$.

Ta lại có $L_2 = R_1$, kết hợp thêm vòng 1 ta có phương trình

$$\begin{aligned} F(R_2, K_3) &= R_3[0, 1, 2, 3] \oplus R_1[0, 1, 2, 3] \\ &= R_3[0, 1, 2, 3] \oplus L_0[0, 1, 2, 3] \oplus R_0[3] \oplus K_1[1] \\ &= R_2[3] \oplus K_3[1] = L_3[3] \oplus K_3[1], \end{aligned}$$

tương đương với

$$K_1[1] \oplus K_3[1] = R_3[0, 1, 2, 3] \oplus L_0[0, 1, 2, 3] \oplus R_0[3] \oplus L_3[3].$$

Phương trình xảy ra:

- với xác suất $(14/64)^2$ khi là xảy ra hai phương trình (3.16) và (3.18);
- với xác suất $(1 - 14/64)^2$ khi xảy ra hai phương trình (3.17) và (3.19).

Như vậy tổng xác suất là $(14/64)^2 + (1 - 14/64)^2$ xấp xỉ $0,66$, khoảng $2/3$.

Tính toán khóa con

Giả sử khóa ban đầu gồm 8 bit là

$$KL_0 = (k_0^{(0)}, k_1^{(0)}, k_2^{(0)}, k_3^{(0)}), \quad KR_0 = (k_4^{(0)}, k_5^{(0)}, k_6^{(0)}, k_7^{(0)}).$$

Dịch vòng trái 1 bit KL_0 và KR_0 ta được KL_1 và KR_1 lần lượt là

$$\begin{aligned} KL_0 &= (k_0^{(0)}, k_1^{(0)}, k_2^{(0)}, k_3^{(0)}) \xrightarrow{\ll 1} KL_1 = (k_1^{(0)}, k_2^{(0)}, k_3^{(0)}, k_0^{(0)}) = (k_0^{(1)}, k_1^{(1)}, k_2^{(1)}, k_3^{(1)}) \\ KR_0 &= (k_4^{(0)}, k_5^{(0)}, k_6^{(0)}, k_7^{(0)}) \xrightarrow{\ll 1} KR_1 = (k_5^{(0)}, k_6^{(0)}, k_7^{(0)}, k_4^{(0)}) = (k_4^{(1)}, k_5^{(1)}, k_6^{(1)}, k_7^{(1)}) \end{aligned}$$

nên suy ra

$$K_1 = (k_5^{(1)}, k_1^{(1)}, k_3^{(1)}, k_2^{(1)}, k_7^{(1)}, k_0^{(1)}).$$

Ở đây, $K_1[1] = k_1^{(1)} = k_2^{(0)}$.

Thực hiện tiếp quá trình này sẽ dẫn chúng ta tới $K_3[1] = k_1^{(0)}$.

Như vậy chúng ta có một mối phụ thuộc giữa hai bit khóa ban đầu $k_1^{(0)}$ và $k_2^{(0)}$.

Giả sử ta phá mã known-plaintext với 100 cặp plaintext-ciphertext và có được kết quả sau của biểu thức

$$R_3[0, 1, 2, 3] \oplus L_0[0, 1, 2, 3,] \oplus R_0[3] \oplus L_3[3]$$

bằng 1 ở 66 lần và bằng 0 ở 34 lần. Như vậy theo phân tích xác suất ở phần phụ thuộc tuyến tính ở trên có thể kết luận $k_1^{(0)} \oplus k_2^{(0)} = 1$. Điều này cho chúng ta hai trường hợp về hai bit của khóa, và nếu ta vét cạn 6 bits còn lại thì tổng cộng cần $2 \cdot 2^6 = 128$ trường hợp. Như vậy chúng ta không phải vét cạn 8 bits, tốn $2^8 = 256$ trường hợp. Hiện tại chúng ta chỉ mới xét một liên hệ giữa $k_1^{(0)}$ và $k_2^{(0)}$ nên độ phức tạp chỉ giảm một nửa. Nếu xét thêm các liên hệ khác thì sẽ có thể giảm thêm.

solve.py

```
# solve.py

from tinydes import encrypt_block, SBox
from functools import reduce
import random

random.seed(4)

secret_key = [1, 1, 0, 1, 0, 0, 1, 0]
count = 0
plaintext = [random.randint(0, 1) for __ in range(8)]
ciphertext = encrypt_block(plaintext, secret_key)

for _ in range(100):
    pt = [random.randint(0, 1) for __ in range(8)]
    ct = encrypt_block(pt, secret_key)
    L0, R0 = pt[:4], pt[4:]
    L3, R3 = ct[:4], ct[4:]
    S = reduce(lambda x, y: x ^ y, R3) ^ reduce(lambda x, y: x ^ y, L0) ^ R0[3] ^ L3[3]
    if S == 1:
        count += 1

if count > 100 - count:
    for k1 in range(2):
        k2 = k1 ^ 1
        for k0 in range(2):          # Bruteforce k_0
            for k in range(2**5):    # Bruteforce k_3 to k_7
                K = [k0, k1, k2] + list(map(int, f"[k:05b]"))
                if encrypt_block(plaintext, K) == ciphertext:
                    print(f"Found key: {K}")
```

3.2.3 Slide attack

Phá mã vi sai (differential cryptanalysis) và phá mã tuyến tính (linear cryptanalysis) dựa trên các phân bố xác suất khi sử dụng S-box. Một ý tưởng đơn giản để chống lại phân tích xác suất là tăng số vòng lén, khi đó chúng ta cần nhiều cặp bản rõ-bản mã hơn để tìm liên hệ giữa các bit của khóa. Rõ ràng nếu số lượng cặp bản rõ-bản mã quá nhiều thì rất khó để tính toán và lưu trữ nên có thể nói cách tiếp cận này hợp lý.

Tuy nhiên slide attack ra đời và đã chứng minh được rằng số vòng nhiều không đồng nghĩa với an toàn hơn. Tương tự với hai phần trước, mình vẫn dùng TinyDES làm ví dụ cho slide attack.

Slide attack

Slide attack là kỹ thuật tấn công mã khôi dãng known-plaintext hoặc chosen-plaintext.

Gọi F là một hoặc hợp của nhiều phép biến đổi trong thuật toán. Giả sử bản rõ ban đầu là $P = P_0$, sau khi đi qua hàm F sẽ trở thành $P_1 = F(P_0)$. Tương tự, P_1 đi qua hàm F sẽ trở thành $P_2 = F(P_1)$. Cứ như vậy tới khi nhận được bản rõ ở cuối thuật toán, giả sử là sau n lần biến đổi, $C = P_n$.

Thông thường, mỗi lần thực hiện phép biến đổi F cũng sẽ đi kèm một hoặc nhiều khóa con. Khi khóa con được sử dụng lặp lại, gọi là K , thì ta có sơ đồ

$$P = P_0 \xrightarrow{F_K} P_1 \xrightarrow{F_K} P_2 \xrightarrow{F_K} \dots \xrightarrow{F_K} P_n = C.$$

Mục tiêu của slide attack là tìm một cặp bản rõ-bản mã (P, C) và (P', C') mà chúng ta gọi là **slid pair**.

Definition (Slid pair)

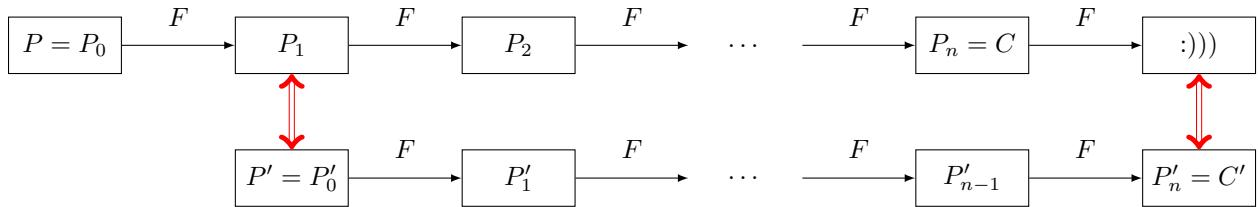
Xét một phép biến đổi F_K với K là khóa được sử dụng lặp lại cho mỗi lần thực hiện hàm F . Để mã hóa bản rõ P thành bản mã C giả sử ta thực hiện theo thứ tự

$$P = P_0 \xrightarrow{F_K} P_1 \xrightarrow{F_K} P_2 \xrightarrow{F_K} \dots \xrightarrow{F_K} P_n = C.$$

Tương tự, để mã hóa bản rõ P' thành bản mã C' giả sử ta thực hiện theo thứ tự

$$P' = P'_0 \xrightarrow{F_K} P'_1 \xrightarrow{F_K} P'_2 \xrightarrow{F_K} \dots \xrightarrow{F_K} P'_n = C'.$$

Khi đó, cặp bản rõ-bản mã (P, C) và (P', C') được gọi là **slid pair** nếu $F_K(P) = P'$ và $F_K(C) = C'$.



Hình 3.33: Sơ đồ mô tả slid pair

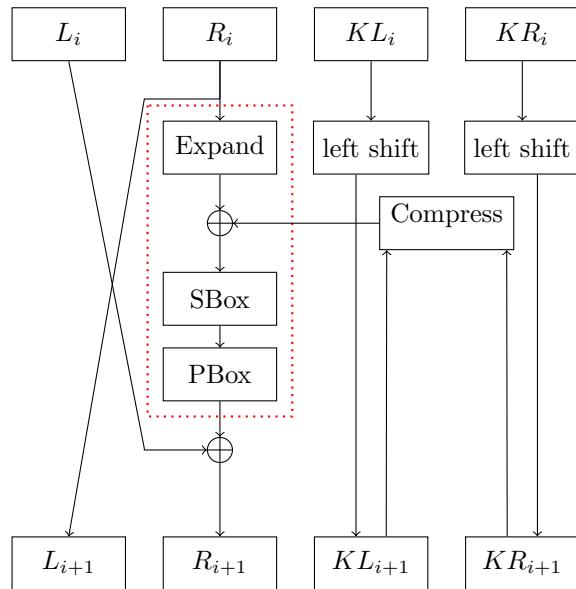
Nếu chúng ta có một cặp bản rõ-bản mã là slid pair thì chúng ta có thể trích xuất khóa K từ hai phương trình. Điểm quan trọng của slide attack là chúng ta chỉ quan tâm hai điều kiện $F_K(P) = P'$ và $F_K(C) = C'$, còn việc hàm F thực hiện bao nhiêu lần không quan trọng. Đây chính là ý nghĩa mình nói ở đầu bài, tăng số vòng không đồng nghĩa với an toàn hơn.

Sau đây mình sẽ ví dụ đơn giản về slide attack. Giống như hai bài trước, mình vẫn dùng TinyDES nhưng ở đây sẽ có hai thay đổi nhỏ.

Slide attack trên mô hình Feistel

Slide attack với TinyDES

TinyDES là một phiên bản thu nhỏ của chuẩn mã hóa DES. TinyDES là mã hóa khối theo mô hình Feistel, kích thước khối là 8 bit, kích thước khóa cũng là 8 bit. Mỗi vòng khóa con có độ dài 6 bit. Trong phần slide attack này chúng ta sẽ thay đổi một vài điểm so với TinyDES ở hai bài trước.



Hình 3.34: Một vòng TinyDES

Theo mô hình Feistel, khối đầu vào 8 bit được chia thành hai nửa trái phải 4 bit. Nửa phải sẽ đi qua các hàm Expand, SBox và PBox, sau đó XOR với nửa trái để được nửa phải mới. Còn nửa trái mới là nửa phải cũ. Tóm lại công thức mô hình Feistel là:

$$L_{i+1} = R_i, \quad R_{i+1} = L_i \oplus F(R_i, K)$$

với $i = 1, 2, \dots, 100$ với đầu vào của khối là (L_0, R_0) . Ở đây chúng ta lưu ý **hai** thay đổi so với TinyDES ở hai bài trước:

- hiện tại chúng ta sử dụng 100 vòng thay vì 3 như hai bài trước;
- cả 100 vòng sử dụng duy nhất một khóa con là K .

Chúng ta vẫn dùng các động tác sau:

1. Expand: mở rộng và hoán vị R_i từ 4 bits lên 6 bits. Giả sử 4 bits của R_i là $b_0 b_1 b_2 b_3$ thì kết quả sau khi Expand là $b_2 b_3 b_1 b_2 b_1 b_0$.
2. SBox: gọi 6 bits đầu vào là $b_0 b_1 b_2 b_3 b_4 b_5$. Khi đó ta tra cứu theo bảng SBox với $b_0 b_5$ chỉ số **hàng**, $b_1 b_2 b_3 b_4$ chỉ số **cột**. Nói cách khác bảng SBox có 4 hàng, 16 cột. Kết quả của SBox là một số 4 bits.
3. PBox: là hàm hoán vị 4 bit $b_0 b_1 b_2 b_3$ thành $b_2 b_0 b_3 b_1$.

Như vậy, hàm F của mô hình Feistel đối với mã TinyDES là:

$$F(R_i, K) = \text{PBox}(\text{SBox}(\text{Expand}(R_i) \oplus K)).$$

Để sinh khóa con cho 100 vòng, khóa ban đầu được chia thành hai nửa trái phải lần lượt là KL_0 và KR_0 . TinyDES thực hiện như sau:

- KL_0 và KR_0 được dịch vòng trái 1 bit để được KL_1 và KR_1 ;
- khóa K dùng chung cho 100 vòng là hoán vị và nén 8 bits của $KL_1 \parallel KR_1$. Đặt 8 bits khi ghép $KL_1 \parallel KR_1$ là $k_0k_1k_2k_3k_4k_5k_6k_7$. Khi đó khóa K là 6 bits $k_5k_1k_3k_2k_7k_0$.

tinydes.py

```
# tindexs.py

sbox = [
    0xE, 0x4, 0xD, 0x1, 0x2, 0xF, 0xB, 0x8, 0x3, 0xA, 0x6, 0xC, 0x5, 0x9, 0x0, 0x7,
    0x0, 0xF, 0x7, 0x4, 0xE, 0x2, 0xD, 0x1, 0xA, 0x6, 0xC, 0xB, 0x9, 0x5, 0x3, 0x8,
    0x4, 0x1, 0xE, 0x8, 0xD, 0x6, 0x2, 0xB, 0xF, 0xC, 0x9, 0x7, 0x3, 0xA, 0x5, 0x0,
    0xF, 0xC, 0x8, 0x2, 0x4, 0x9, 0x1, 0x7, 0x5, 0xB, 0x3, 0xE, 0xA, 0x0, 0x6, 0xD
]

def Xor(a: list[int], b: list[int]) -> list[int]:
    return [x^y for x, y in zip(a, b)]

def Expand(R: list[int]) -> list[int]:
    return [R[2], R[3], R[1], R[2], R[1], R[0]]

def SBox(R: list[int]) -> list[int]:
    row = int("".join(map(str, [R[0], R[5]])), 2)
    col = int("".join(map(str, R[1:5])), 2)

    return list(map(int, format(sbox[row*16 + col], "04b")))

def PBox(R: list[int]) -> list[int]:
    return [R[2], R[0], R[3], R[1]]

def PBox_inv(R: list[int]) -> list[int]:
    return [R[1], R[3], R[0], R[2]]

def Compress(K: list[int], round: int) -> list[int]:
    left, right = K[:4], K[4:]
    if round == 0 or round == 2:
        left = left[1:] + left[:1]
        right = right[1:] + right[:1]
    elif round == 1:
        left = left[2:] + left[:2]
        right = right[2:] + right[:2]

    Ki = left + right
    return left, right, [Ki[5], Ki[1], Ki[3], Ki[2], Ki[7], Ki[0]]

def encrypt_block(plaintext: list[int], key: list[int]) -> list[int]:
```

```

keys = [key]
left, right = key[:4], key[4:]
for i in range(3):
    left, right, key = Compress(left + right, i)
    keys.append(key)

left, right = plaintext[:4], plaintext[4:]
for i in range(100):      # 100 vòng
    # chỉ sử dụng mỗi K_1 trong TinyDES gốc
    left, right = right, Xor(left, PBox(SBox(Xor(Expand(right), keys[1]))))

return left + right

# print(encrypt_block([0, 1, 0, 1, 1, 1, 0, 0], [1, 0, 0, 1, 1, 0, 1, 0]))

```

Ở đây chúng ta thấy mô hình mã hóa sẽ diễn ra như sau. Gọi (L_0, R_0) là hai nửa trái phải của bản rõ ban đầu P . Khi đó, ở mỗi vòng biến đổi sẽ sử dụng chung khóa con K theo mô hình

$$P = (L_0, R_0) \xrightarrow{F_K} (L_1, R_1) \xrightarrow{F_K} (L_2, R_2) \xrightarrow{F_K} (L_3, R_3) = C.$$

Theo mô hình Feistel thì

$$L_1 = R_0, \quad R_1 = L_0 \oplus f(R_0, K)$$

với f là round function tương ứng với thuật toán TinyDES. Nói cách khác thì F_k là

$$F_K(L_i, R_i) = (R_i, L_i \oplus f(R_i, K)).$$

Lúc này slid pair có dạng

$$\begin{cases} F_K(P) = P' \\ F_K(C) = C' \end{cases} \iff \begin{cases} F_K(L_0, R_0) = (L'_0, R'_0) \\ F_K(L_3, R_3) = (L'_3, R'_3) \end{cases}$$

hay tương đương với

$$\begin{cases} (R_0, L_0 \oplus f(R_0, K)) = (L'_0, R'_0) \\ (R_3, L_3 \oplus f(R_3, K)) = (L'_3, R'_3) \end{cases} \iff \begin{cases} R_0 = L'_0 \\ L_0 \oplus f(R_0, K) = L'_0 \\ R_3 = L'_3 \\ L_3 \oplus f(R_3, K) = R'_3. \end{cases} \quad (3.20)$$

Như vậy:

nếu chúng ta có (P, C) và (P', C') thỏa các điều kiện ở (3.20) thì chúng ta có slid pair.

Câu hỏi đặt ra là nếu chúng ta không biết khóa con K thì làm sao kiểm tra được các điều kiện trên?

Câu trả lời (mà cũng là cách chúng ta thực hiện trên thực tế) là chúng ta giả sử đã tìm được slid pair. Như vậy điều kiện đầu và điều kiện thứ ba phải thỏa mãn trước. Sau đó từ điều kiện thứ hai và thứ tư chúng ta tìm ngược lại K . Cuối cùng chúng ta thử mã hóa P với K đã tìm được. Nếu chúng ta thu được chính xác C thì K là khóa con cần tìm, ngược lại thì chúng ta thử với slid pair khác.

 solve.py

```

# solve.py

from tinydes import encrypt_block, PBox, SBox, Expand, Xor
import random


def fault_encrypt_block(plaintext: list[int], key: list[int]) -> list[int]:
    left, right = plaintext[:4], plaintext[4:]
    for _ in range(100):
        left, right = right, Xor(left, PBox(SBox(Xor(Expand(right), key))))
    return left + right


pts = []
cts = []
secret_key = [1, 0, 0, 1, 1, 1, 1, 0]

L = 2**4

for _ in range(L):
    pt = [random.randint(0, 1) for __ in range(8)]
    ct = encrypt_block(pt, secret_key)
    pts.append(pt)
    cts.append(ct)

for i in range(L):
    L0, R0 = pts[i][:4], pts[i][4:]
    L3, R3 = cts[i][:4], cts[i][4:]
    for j in range(i + 1, L):
        l0, r0 = pts[j][:4], pts[j][4:]
        l3, r3 = cts[j][:4], cts[j][4:]
        # Lazy bruteforce for K
        for k in range(2**6):
            key = list(map(int, f"{k:06b}"))
            # Check slid pair
            if l0 == R0 and r0 == Xor(L0, PBox(SBox(Xor(Expand(R0), key)))):
                if l3 == R3 and r3 == Xor(R3, PBox(SBox(Xor(Expand(R3), key)))):
                    if fault_encrypt_block(pts[i], key) == cts[i]:
                        print(key)

```

Ở đoạn code trên mình bruteforce khóa con K vì SBox của TinyDES (và cũng là của DES) nhận đầu vào 6 bit nhưng đầu ra giảm còn 4 bit (chứ không phải do mình lười đâu hihihi). Do đó có thể có nhiều trường hợp của khóa con K có thể thỏa mãn điều kiện của slid pair. Chúng ta cũng có thể tạo lookup table và thực hiện ngược lại round function để tìm các khả năng của khóa con K .

Gọi PBox^{-1} là phép biến đổi ngược của PBox . Khi đó từ điều kiện thứ hai $L_0 \oplus f(R_0, K) = L'_0$ suy ra

$$L_0 \oplus L'_0 = f(R_0, K) = \text{PBox}(\text{SBox}(\text{Expand}(R_0) \oplus K)),$$

suy ra

$$\text{PBox}^{-1}(L_0 \oplus L'_0) = \text{SBox}(\text{Expand}(R_0) \oplus K).$$

Chúng ta có thể tính được $\text{PBox}^{-1}(L_0 \oplus L'_0)$ và $\text{Expand}(R_0)$ nên việc "đoán" K không phải vấn đề khó khăn. Thêm nữa điều kiện thứ tư cũng cho chúng ta các ứng cử viên cho khóa con K . Kết hợp hai điều kiện này chúng ta có khóa con K và chúng ta sẽ thử mã hóa P thành C .

Slide attack với DES

Ở olympiad mật mã học quốc tế NSUCRYPTO 2024 có một bài slide attack trên DES là bài 4 ở round 2 "Weak key schedule for DES". Bài này được giải bởi bạn Chương (vnc) đội mình. Ở phần sau mình sẽ trình bày lời giải cho bài này. Mình sẽ sử dụng code của bạn Chương trong lời giải. Xin cảm ơn bạn Chương vì đã đóng góp :D :D

Trong bài này, thông tin ban đầu là file `Book.txt` và được mã hóa thành file `Book_Cipher.txt`.

Thuật toán được sử dụng để mã hóa là DES. Tuy nhiên trong bài này đặc biệt ở chỗ mỗi vòng đều dùng chung một khóa con (khóa con đầu tiên của thuật toán sinh khóa con).

Nhiệm vụ của chúng ta là tìm khóa con đó và giải mã thông điệp sau:

```
86991641D28259604412D6BA88A5C0A6471CA7222C52482BF2D0E841D4343DFB877DC8E0147F3D5F20FC18FF28CB5C4DA8A0F46
```

Cài đặt FAULTY_DES

Ở đây bạn Chương gọi thuật toán của đề bài là `FAULTY_DES` và bạn sẽ cài đặt thuật toán này cùng với một số hàm hỗ trợ cho việc giải bài.

❶ faulty_des.py

```
# faulty_des.py

from typing import List, Tuple

class DES_CONST:
    PC1 = (
        57, 49, 41, 33, 25, 17, 9,
        1, 58, 50, 42, 34, 26, 18,
        10, 2, 59, 51, 43, 35, 27,
        19, 11, 3, 60, 52, 44, 36,
        63, 55, 47, 39, 31, 23, 15,
        7, 62, 54, 46, 38, 30, 22,
        14, 6, 61, 53, 45, 37, 29,
        21, 13, 5, 28, 20, 12, 4,
    )

    PC2 = (
        14, 17, 11, 24, 1, 5,
        3, 28, 15, 6, 21, 10,
        23, 19, 12, 4, 26, 8,
        16, 7, 27, 20, 13, 2,
        41, 52, 31, 37, 47, 55,
        30, 40, 51, 45, 33, 48,
        44, 49, 39, 56, 34, 53,
        46, 42, 50, 36, 29, 32,
    )
```

```

KEY_ROTATION = (
    1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1,
)

INITIAL_PERMUTATION = (
    58, 50, 42, 34, 26, 18, 10, 2,
    60, 52, 44, 36, 28, 20, 12, 4,
    62, 54, 46, 38, 30, 22, 14, 6,
    64, 56, 48, 40, 32, 24, 16, 8,
    57, 49, 41, 33, 25, 17, 9, 1,
    59, 51, 43, 35, 27, 19, 11, 3,
    61, 53, 45, 37, 29, 21, 13, 5,
    63, 55, 47, 39, 31, 23, 15, 7,
)

ROUND_PERMUTATION = (
    16, 7, 20, 21,
    29, 12, 28, 17,
    1, 15, 23, 26,
    5, 18, 31, 10,
    2, 8, 24, 14,
    32, 27, 3, 9,
    19, 13, 30, 6,
    22, 11, 4, 25,
)

INV_ROUND_PERMUTATION = (
    9, 17, 23, 31,
    13, 28, 2, 18,
    24, 16, 30, 6,
    26, 20, 10, 1,
    8, 14, 25, 3,
    4, 29, 11, 19,
    32, 12, 22, 7,
    5, 27, 15, 21,
)

FINAL_PERMUTATION = (
    40, 8, 48, 16, 56, 24, 64, 32,
    39, 7, 47, 15, 55, 23, 63, 31,
    38, 6, 46, 14, 54, 22, 62, 30,
    37, 5, 45, 13, 53, 21, 61, 29,
    36, 4, 44, 12, 52, 20, 60, 28,
    35, 3, 43, 11, 51, 19, 59, 27,
    34, 2, 42, 10, 50, 18, 58, 26,
    33, 1, 41, 9, 49, 17, 57, 25,
)

EXPANSION = (
    32, 1, 2, 3, 4, 5,
    4, 5, 6, 7, 8, 9,
    8, 9, 10, 11, 12, 13,
    12, 13, 14, 15, 16, 17,
)

```

```

16, 17, 18, 19, 20, 21,
20, 21, 22, 23, 24, 25,
24, 25, 26, 27, 28, 29,
28, 29, 30, 31, 32, 1 ,
)

SBOX = [
(
  14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,
  0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,
  4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,
  15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13,
),
(
  15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10,
  3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5,
  0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15,
  13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9,
),
(
  10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,
  13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1,
  13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7,
  1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12,
),
(
  7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,
  13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,
  10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,
  3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14,
),
(
  2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9,
  14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6,
  4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14,
  11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3,
),
(
  12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11,
  10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8,
  9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6,
  4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13,
),
(
  4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1,
  13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6,
  1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2,
  6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12,
),
(
  13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,
  1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,
  7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,
)
]

```

```

        2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11,
    ),
]

class HELP:
    @classmethod
    def INT_TO_BITS(cls, value: int, size: int) -> List[int]:
        bits = []
        for i in reversed(range(size)):
            bits.append((value >> i) & 1)
        return bits

    @classmethod
    def BITS_TO_INT(cls, bits: List[int]) -> int:
        value = 0
        for bit in bits:
            value <= 1
            value |= bit
        return value

    @classmethod
    def BLOCK_TO_BITS(cls, block: bytes) -> List[int]:
        bits = []
        for byte in block:
            bits.extend(cls.INT_TO_BITS(byte, 8))
        return bits

    @classmethod
    def BITS_TO_BLOCK(cls, bits: List[int], size: int) -> bytes:
        block_bytes = []
        for i in range(size):
            byte = cls.BITS_TO_INT(bits[i * 8 : (i + 1) * 8])
            block_bytes.append(byte)
        return bytes(block_bytes)

    @classmethod
    def PERMUTATE(cls, bits: List[int], table: List[int]) -> List[int]:
        return [
            bits[table[i] - 1] for i in range(len(table))
        ]

    @classmethod
    def XOR(cls, bits1: List[int], bits2: List[int]) -> List[int]:
        return [x ^ y for x, y in zip(bits1, bits2)]

    @classmethod
    def EXPAND(cls, bits: List[int], table: List[Tuple]) -> List[int]:
        return [
            bits[table[i] - 1]
            for i in range(len(table))
        ]
    @classmethod
    def SUBSTITUTE(cls, bits: List[int], mapping: List[Tuple]) -> List[int]:

```

```

pieces = []

for i in range(8):
    piece_bits = bits[i * 6 : (i + 1) * 6]

    piece = HELP.BITS_TO_INT(piece_bits)
    pieces.append(piece)

values = []

for i, piece in enumerate(pieces):
    row = (piece & 1) | ((piece >> 4) & 0b10)
    column = (piece & 0b011110) >> 1

    values.append(mapping[i][row * 16 + column])

result_bits = []

for value in values:
    result_bits.extend(HELP.INT_TO_BITS(value, 4))

return result_bits

class FAULTY_DES:
    def __init__(self, rk: bytes, rounds: int = 16):
        self.rounds = rounds
        self.round_keys = [HELP.BLOCK_TO_BITS(rk) for _ in range(rounds)]
        self.reversed_round_keys = self.round_keys[::-1]

    def encrypt(self, plaintext: bytes) -> bytes:
        assert len(plaintext)%8 == 0
        blocks = [plaintext[i:i+8] for i in range(0, len(plaintext), 8)]
        return b"".join([self._process_block(block, self.round_keys) for block in blocks])

    def decrypt(self, ciphertext: bytes) -> bytes:
        assert len(ciphertext)%8 == 0
        blocks = [ciphertext[i:i+8] for i in range(0, len(ciphertext), 8)]
        return b"".join([self._process_block(block, self.reversed_round_keys) for block in blocks])

    def _process_block(self, block: bytes, schedule) -> bytes:
        bits = HELP.BLOCK_TO_BITS(block)

        bits = HELP.PERMUTATE(bits, DES_CONST.INITIAL_PERMUTATION)
        left, right = bits[:32], bits[32:]

        for i in range(self.rounds):
            new_right = self._function(right, schedule[i])
            left, right = right, HELP.XOR(new_right, left)

        result_bits = right + left
        result_bits = HELP.PERMUTATE(result_bits, DES_CONST.FINAL_PERMUTATION)

```

```

    return HELP.BITS_TO_BLOCK(result_bits, 8)

def _function(self, bits: List[int], key_bits: List[int]) -> List[int]:
    bits = HELP.EXPAND(bits, DES_CONST.EXPANSION)
    bits = HELP.XOR(bits, key_bits)
    bits = HELP.SUBSTITUTE(bits, DES_CONST.SBOX)
    bits = HELP.PERMUTATE(bits, DES_CONST.ROUND_PERMUTATION)

    return bits

```

Thực hiện solve.py

Phần này mình sẽ viết `solve.py` để giải bài này từ jupyter notebook của bạn Chương.

Đầu tiên chúng ta gọi một số thư viện.

```

from faulty_des import DES_CONST, HELP, FAULTY_DES
from collections import defaultdict
from typing import List, Tuple
from itertools import product
from tqdm import tqdm

```

Tiếp theo, chúng ta cần thống nhất rằng mỗi khối trong thuật toán FAULTY_DES sẽ được biểu diễn bởi mảng gồm 8 phần tử (`list[int]`). Khi đó nửa trái (left-half) là 4 phần tử đầu của mảng và nửa phải (right-half) là 4 phần tử sau. Do đó chúng ta cần một số lambda để lấy nửa trái/phải. Sau bước cuối chúng ta ghép nửa phải cuối cùng với nửa trái cuối cùng nên cần thêm hàm `Swap_f`.

```

Left_f = lambda block: block[:4]
Right_f = lambda block: block[4:]
Swap_f = lambda block: Right_f(block) + Left_f(block)

```

Sau đó đọc bản rõ và bản mã từ file đè cho rồi tách chúng thành các khối 8 bytes.

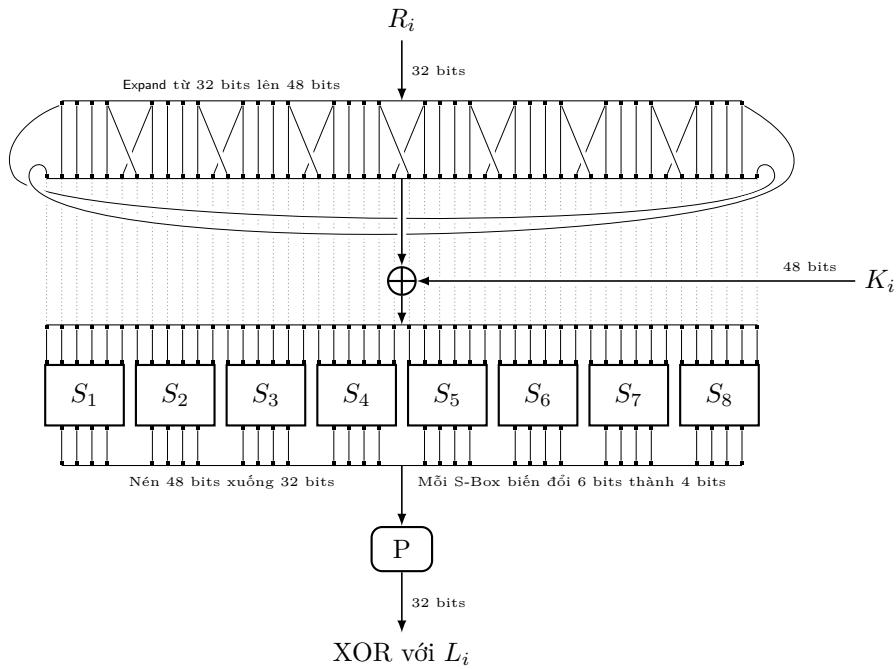
```

# Known-Plaintext-Ciphertext Pairs
plaintext = open("Book.txt", "rb").read()[:-1]      # độ dài bản rõ chia hết cho 8
ciphertext = open("Book_cipher.txt", "rb").read()

# Divide them into blocks
pts = [plaintext[i:i+8] for i in range(0, len(plaintext), 8)]
cts = [ciphertext[i:i+8] for i in range(0, len(ciphertext), 8)]

```

Tiếp theo chúng ta tìm các slid pair. Ở đây chúng ta cần xem lại cách hoạt động của thuật toán DES. Hình sau mình lấy từ [30] và chỉnh sửa lại.



Hình 3.35: Round function của DES

Tóm tắt cách hoạt động thì DES cũng theo mô hình

$$L_{i+1} = R_i, \quad R_{i+1} = L_i \oplus \text{PBox}(\text{SBox}(\text{Expand}(R_i) \oplus K_{i+1})),$$

trong đó

- Expand mở rộng nửa khối từ 32 bits lên 48 bits;
- SBox: 48 bits sẽ được chia thành 8 đoạn, mỗi đoạn có 6 bits. Sau đó mỗi đoạn sẽ đi qua các S-Box và giảm từ 6 bits xuống 4 bits. Các kết quả được nối lại với nhau nên kết quả sau SBox là $4 \cdot 8 = 32$ bits;
- PBox: thực hiện hoán vị 32 bit sau SBox.

Ở đây slid pair cũng tương tự TinyDES ở trên. Giả sử slid pair là cặp (P, C) và (P', C') . Khi đó

$$F_K(P) = P', \quad F_K(C) = C'$$

với F_K là round function

$$(L_i, R_i) \xrightarrow{F_K} (L_{i+1}, R_{i+1}) \equiv (R_i, L_i \oplus \text{PBox}(\text{SBox}(\text{Expand}(R_i) \oplus K_{i+1}))),$$

như trên.

```
# Before and After DES' rounds operations, there is a Block Permutation step!
def BlockPermutate(block: bytes, table: list):
    assert len(block) == 8
    block_bits = HELP.BLOCK_TO_BITS(block)
    block_bits = HELP.PERMITATE(block_bits, table)
    block      = HELP.BITS_TO_BLOCK(block_bits, 8)

    return block
```

(continues on next page)

(continued from previous page)

```

# Create Lookup Table
lookup = defaultdict(list)

for pi, ci in tqdm(zip(pts, cts), desc="[+] Creating Lookup Table..."):
    pi_initial = BlockPermute(pi, DES_CONST.INITIAL_PERMUTATION)
    ci_preout = BlockPermute(ci, DES_CONST.INITIAL_PERMUTATION)
    # Why Left_f(Ci) -> Remember that the final round doesn't swap two-halves
    lookup[Right_f(pi_initial) + Left_f(ci_preout)].append((pi_initial, ci_preout))

# Finding "slid pairs"
slid_pairs = []

for pj, cj in tqdm(zip(pts, cts), desc="[+] Finding slid pairs..."):
    pj_initial = BlockPermute(pj, DES_CONST.INITIAL_PERMUTATION)
    cj_preout = BlockPermute(cj, DES_CONST.INITIAL_PERMUTATION)

    try:
        # Why Right_f(Cj) -> Remember that the final round doesn't swap two-halves
        for pi_initial, ci_preout in lookup[Left_f(pj_initial) + Right_f(cj_preout)]:
            slid_pairs.append([
                # Now we swap to ensure that (P,C) and (P',C') is slid pair
                # <=> F(P) = P' and F(C) = C'
                (pi_initial, Swap_f(ci_preout)),
                (pj_initial, Swap_f(cj_preout)),
            ])
    except:
        continue

print(f"[!] Found {len(slid_pairs)} possible slid pairs!")

```

Các phép biến đổi ngược của các phép biến đổi trong DES nhằm tìm các "ứng cử viên" cho khóa. Quan trọng là S-Box vì chúng ta đã biết mỗi S-Box biến đổi 6 bits thành 4 bits nên với nhiều đầu vào cho cùng đầu ra. Khi đi ngược lại để tìm "ứng cử viên" thì ta phải xét tất cả trường hợp đầu vào S-Box cho cùng đầu ra mà ta đang có.

```

# Enumerate all possible candidates
def RevSubstitute(out: List[int], mapping: List[Tuple]):
    cands = [[] for _ in range(8)]
    out = [out[i:i+4] for i in range(0, len(out), 4)]

    for idx in range(8):
        for piece in range(2**6):
            row = (piece & 1) | ((piece >> 4) & 0b10)
            column = (piece & 0b011110) >> 1
            if HELP.INT_TO_BITS(mapping[idx][row * 16 + column], 4) == out[idx]:
                cands[idx].append(HELP.INT_TO_BITS(piece, 6))

    for rk in product(*cands):
        yield sum(rk, [])

def RevFeistel(inp, out):

```

(continues on next page)

(continued from previous page)

```

out = HELP.PERMUTATE(out, DES_CONST.INV_ROUND_PERMUTATION)
inp = HELP.EXPAND(inp, DES_CONST.EXPANSION)
for rev_out in RevSubstitute(out, DES_CONST.SBOX):
    yield HELP.BITS_TO_BLOCK(HELP.XOR(rev_out, inp), 6)

def RevRound(state0: bytes, state1: bytes):
    L0, R0 = Left_f(state0), Right_f(state0)
    L1, R1 = Left_f(state1), Right_f(state1)
    return set(RevFeistel(
        inp=HELP.BLOCK_TO_BITS(R0),
        out=HELP.XOR(HELP.BLOCK_TO_BITS(R1), HELP.BLOCK_TO_BITS(L0))
    ))

```

Tìm các khóa con có thể và thử giải mã thông điệp đè cho với khóa con đó.

```

rk_cands = set()

for (pi, ci), (pj, cj) in tqdm(slid_pairs, desc="[+] Recovering Possible RoundKey..."):
    rks_1 = RevRound(pi, pj)
    rks_2 = RevRound(ci, cj)
    rk_cands = rk_cands.union(rks_1.intersection(rks_2))

print(f"[!] Found {len(rk_cands)} possible RoundKeys!")

# Try to decrypt intercepted ciphertext
intercepted_ciphertext = b"".join([
    bytes.fromhex("86991641D28259604412D6BA88A5C0A6471CA722"),
    bytes.fromhex("2C52482BF2D0E841D4343DFB877DC8E0147F3D5F"),
    bytes.fromhex("20FC18FF28CB5C4DA8A0F4694861AB5E98F37ADB"),
    bytes.fromhex("C2D69B35779D9001BB4B648518FE6EBC00B2AB10")
])

for rk in tqdm(rk_cands, desc="[+] Try to decrypt intercepted ciphertext..."):
    cipher = FAULTY_DES(rk)
    try:
        print("> Readable Message:", cipher.decrypt(intercepted_ciphertext).decode())
        print("> WRT RoundKey:", rk.hex(), end="\n\n")
    except:
        continue

# Verify again?
assert FAULTY_DES(bytes.fromhex("0c74fa6a642a")).encrypt(plaintext) == ciphertext

```

3.2.4 Algebraic cryptanalysis

Theo [31] thì algebraic cryptanalysis (mình tạm dịch là **phá mã đại số**) là kỹ thuật phá mã dựa trên việc giải một hệ phương trình các đa thức trên trường \mathbb{F}_2 hoặc đôi khi là vành khác.

Algebraic cryptanalysis gồm hai giai đoạn:

1. Tìm sự liên kết giữa bản rõ và bản mã dưới dạng các hệ phương trình đa thức. Do đó algebraic cryptanalysis thuộc loại tấn công known-plaintext hoặc chosen-plaintext.
2. Giải hệ phương trình đại số. Ở bước này chúng ta có thể giải tay hoặc sử dụng những phần mềm

chuyên dụng gọi là SAT-solver vì thông thường các loại mã hóa rất phức tạp.

Interpolation Attack

Interpolation, hay tiếng Việt gọi là nội suy (ví dụ Lagrange's interpolation là nội suy Lagrange) được giới thiệu vào 1997 [32].

Mình xin phép nhắc lại ý tưởng của nội suy Lagrange như sau:

- chúng ta không biết hệ số của đa thức bậc n , giả sử là

$$f(x) = a_0 + a_1x + \cdots + a_nx^n$$

với $a_n \neq 0$;

- chúng ta biết $n+1$ điểm thuộc đồ thị của đa thức là

$$(x_0, f(x_0)), (x_1, f(x_1)), \dots, (x_n, f(x_n));$$

- khi đó chúng ta hoàn toàn có thể tìm lại được đa thức $f(x)$ ban đầu.

Ở đây, interpolation attack sử dụng ý tưởng tương tự.

1. Chúng ta cố gắng xây dựng một đa thức liên hệ giữa bản rõ p và bản mã c , nghĩa là $f(p) = c$ với mọi cặp bản rõ p và bản mã c . Ở đây chúng ta không biết khóa.
2. Với một lượng cặp bản rõ-bản mã nhất định và bậc của đa thức f thấp chúng ta hy vọng tìm lại được đa thức bằng nội suy.
3. Sử dụng đa thức tìm được để khôi phục khóa.

Ở [32], các tác giả xây dựng một toy cipher gọi là \mathcal{PURE} nhằm demo cho phương pháp tấn công này.

Giới thiệu \mathcal{PURE} cipher

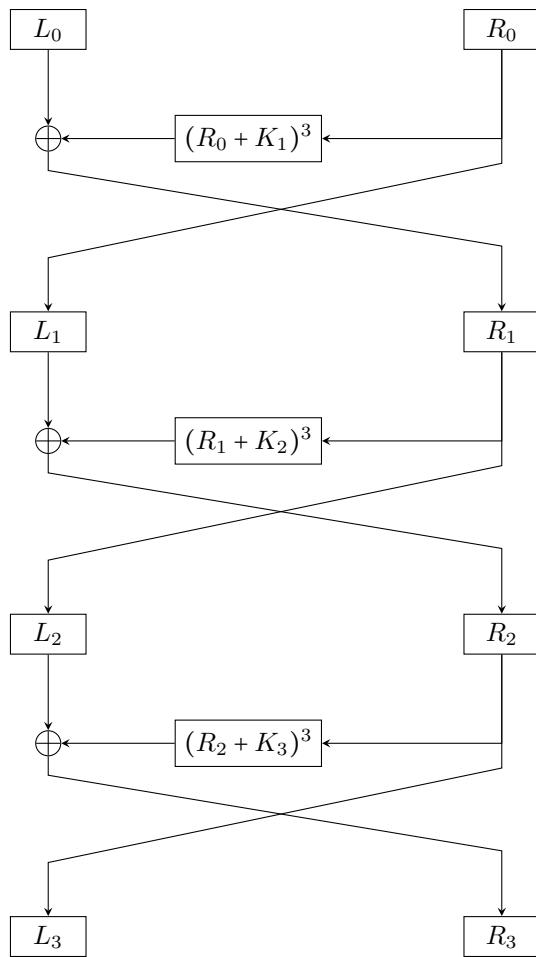
Thuật toán dựa trên mô hình Feistel với độ dài khối là 64 bits và độ dài khóa là 192 bits. Khóa K gồm 6 khóa con cho 6 vòng là k_1, k_2, \dots, k_6 với $k_i \in \mathbb{F}_{2^{32}}$, nghĩa là mỗi khóa có 32 bits (bằng một nửa độ dài khối, tương ứng với mô hình Feistel).

Ở vòng thứ i , giả sử nửa trái là L_i và nửa phải là R_i thì round function của mô hình Feistel là

$$L_{i+1} = R_i, \quad R_{i+1} = L_i \oplus (R_i \oplus K_{i+1})^3.$$

Ở đây, phép tính z^3 được tính trong trường $\mathbb{F}_{2^{32}}$. Chúng ta có thể sử dụng bất kì đa thức tối giản nào làm modulo cho $\mathbb{F}_{2^{32}}$.

Để demo cho interpolation attack chúng ta sẽ xét \mathcal{PURE} cipher với 3 vòng như hình sau.



Hình 3.36: PURÉ cipher với 3 vòng

Thiết lập hệ phương trình giữa bản rõ và bản mã

Tiếp theo, chúng ta ... thay từng biểu thức vào để biểu diễn bản mã theo bản rõ.

Đầu tiên chúng ta cần chú ý một điều là do trường $\mathbb{F}_{2^{32}}$ có đặc số (characteristic) là 2 nên $3 \equiv 1$.

Ở vòng 1:

$$\begin{aligned} L_1 &= R_0, \\ R_1 &= L_0 \oplus (R_0 \oplus K_1)^3 \\ &= L_0 \oplus R_0^3 \oplus 3R_0^2K_1 \oplus 3R_0K_1^2 \oplus K_1^3 \\ &= L_0 \oplus R_0^3 \oplus R_0^2K_1 \oplus R_0K_1^2 \oplus K_1^3. \end{aligned}$$

Chúng ta có thể nói rằng L_1 và R_1 phụ thuộc vào L_0 và R_0 , nghĩa là có ánh xạ

$$\begin{cases} L_1 = \alpha_1(L_0, R_0) \\ R_1 = \beta_1(L_0, R_0). \end{cases}$$

Ở vòng 2:

$$\begin{aligned} L_2 &= R_1 = L_0 \oplus R_0^3 \oplus R_0^2K_1 \oplus R_0K_1^2 \oplus K_1^3, \\ R_2 &= L_1 \oplus (R_1 \oplus K_2)^3 = R_0 \oplus R_1^3 \oplus R_1^2K_2 \oplus R_1K_2^2 \oplus K_2^3. \end{aligned}$$

Lúc này ta có L_2 và R_2 được biểu diễn theo L_1 và R_1 , giả sử chúng ta có ánh xạ

$$\begin{cases} L_2 = \alpha_2(L_1, R_1) = \alpha_2(\alpha_1(L_0, R_0), \beta_1(L_0, R_0)) \\ R_2 = \beta_2(L_1, R_1) = \beta_2(\alpha_1(L_0, R_0), \beta_1(L_0, R_0)). \end{cases}$$

Tương tự ở vòng 3:

$$\begin{aligned} L_3 &= R_2 = R_0 \oplus R_1^3 \oplus R_1^2 K_2 \oplus R_1 K_2^2 \oplus K_2^3, \\ R_3 &= L_2 \oplus (R_2 \oplus K_3)^3 = R_1 \oplus R_2^3 \oplus R_2^2 K_2 \oplus R_2 K_2^2 \oplus K_3^3. \end{aligned}$$

Chúng ta thực hiện tương tự như hai vòng trước và nhanh chóng nhận ra rằng việc thay thế này rất cồng kềnh. Do đó mình sử dụng SageMath để thực hiện khai triển giúp mình. Các bạn có thể xem đoạn code sau nếu hứng thú.

❶ Code tính L_3 và R_3 theo L_0, R_0 , và khóa con

```
from sage.all import *

F = GF(2)['x']; x = F.gen()
F32 = GF(2**3, name='x', modulus='random')
Pr = PolynomialRing(F32, names='l0, r0, k1, k2, k3')
l0, r0, k1, k2, k3 = Pr.gens()

l1 = r0
r1 = l0 + (r0 + k1)**3

l2 = r1
r2 = l1 + (r1 + k2)**3

l3 = r2
r3 = l2 + (r2 + k3)**3

# Format for LaTeX
f = str(l3) \
.replace('r0', 'R_0') \
.replace('l0', 'L_0') \
.replace('k1', r'\color{red}{K_1}') \
.replace('k2', r'\color{red}{K_2}') \
.replace('k3', r'\color{red}{K_3}') \
.replace('+', r'\oplus') \
.replace('*', ' ')
print(f)
```

Như vậy chúng ta biểu diễn được L_3 và R_3 theo L_0, R_0 và các khóa con như sau.

$$\begin{aligned} R_3 &= R_0^{27} + \dots \\ L_3 &= R_0^9 \oplus R_0^8 K_1 \oplus R_0 K_1^8 \oplus K_1^9 \oplus L_0 R_0^6 \\ &\quad \oplus L_0 R_0^4 K_1^2 \oplus L_0 R_0^2 K_1^4 \oplus L_0 K_1^6 \oplus R_0^6 K_2 \oplus R_0^4 K_1^2 K_2 \\ &\quad \oplus R_0^2 K_1^4 K_2 \oplus K_1^6 K_2 \oplus L_0^2 R_0^3 \oplus L_0^2 R_0^2 K_1 \oplus L_0^2 R_0 K_1^2 \\ &\quad \oplus L_0^2 K_1^3 \oplus R_0^3 K_2^2 \oplus R_0^2 K_1 K_2^2 \oplus R_0 K_1^2 K_2^2 \oplus K_1^3 K_2^2 \\ &\quad \oplus L_0^3 \oplus L_0^2 K_2 \oplus L_0 K_2^2 \oplus K_2^3 \oplus R_0. \end{aligned}$$

Ở đây R_3 là đa thức bậc 27 theo R_0 và L_0 , có lẽ chúng ta không nên dây dưa với anh bạn này.

Nhìn L_3 cũng khá r้อม nhưng vẫn có thể xử lý được (hoặc mình tin vậy). Hơn nữa trong biểu thức của L_3 không có sự xuất hiện của K_3 . Tuy nhiên chỉ với một cặp bản rõ $P = L_0 \parallel R_0$ và bản mã $C = L_3 \parallel R_3$ thì không đủ để chúng ta tìm lại khóa.

Ý tưởng của interpolation attack cũng như nội suy Lagrange là dựa trên nhiều cặp bản rõ-bản mã. Trong biểu thức của L_3 nếu chúng ta nhóm hạng tử theo bậc của R_0 lại thì ta có các hệ số:

- trước R_0^9 là 1;
- trước R_0^8 là K_1 ;
- trước R_0^6 là $L_0 \oplus K_2$;
- trước R_0^4 là $L_0 K_1^2 \oplus K_1^2 K_2$;
- trước R_0^3 là $L_0^2 \oplus K_2^2$;
- trước R_0^2 là $K_1^4 K_2 \oplus L_0^2 K_1^2 \oplus K_1 K_2^2$;
- trước R_0 là $L_0^2 K_1^2 \oplus K_1^2 K_2^2$;
- hệ số tự do là các đơn thức còn lại không chứa R_0 .

Như vậy chúng ta sẽ có biểu diễn L_3 theo R_0 là đa thức dạng

$$L_3 = R_0^9 \oplus a_8 R_0^8 \oplus a_6 R_0^6 \oplus a_4 R_0^4 \oplus a_3 R_0^3 \oplus a_2 R_0^2 \oplus a_1 R_0 \oplus a_0$$

với a_i có thể xem là hàm của theo các khóa con và L_0 .

Nếu chúng ta cố định L_0 và thay đổi R_0 thì với 7 cặp bản rõ-bản mã là đủ để khôi phục đa thức trên nếu chúng ta giải hệ phương trình tuyến tính. Trong trường hợp PURE với 6 vòng như bản gốc thì chúng ta cần nhiều cặp bản rõ-bản mã hơn nhiều, theo đó việc tính toán đa thức sẽ tốn công sức hơn.

Khi đã khôi phục được đa thức, tức là đã tìm được các hệ số, thì chúng ta có thể mã hóa bất kì thông điệp nào mà không cần quan tâm khóa. Vậy nếu chúng ta muốn tìm khóa thì sao?

Như đã nói, các hệ số của đa thức có thể được xem như các hàm theo các khóa con và nửa trái ban đầu L_0 . Lúc này ta có một hệ phương trình không tuyến tính và việc giải quyết khá khó khăn. Trong phiên bản đơn giản với 3 vòng như trên, để ý rằng hệ số trước R_0^8 chính là K_1 . Từ đây, kết hợp với các phương trình khác có thể tìm được K_2 . Tuy nhiên với K_3 thì chúng ta không còn cách nào khác ngoài vét cạn. Như vậy có thể kết luận:

1. Với phiên bản PURE cipher đơn giản với 3 vòng cần 7 cặp bản rõ-bản mã (để tìm K_1 và K_2) và vét cạn 2^{32} trường hợp khóa K_3 .
2. Với PURE cipher gốc 6 vòng thì ta cần nhiều cặp bản rõ-bản mã hơn để tìm 5 khóa con đầu và vét cạn 2^{32} trường hợp khóa K_6 .

Ở đây là đoạn code demo cho PURE cipher với 3 vòng của mình.

➊ Demo tấn công PURE cipher 3 vòng

```
from sage.all import GF, matrix, vector
import random

# Define fields
F = GF(2)[x] # GF(2)
x = F.gen()
F32 = GF(2**32, name='x', modulus='random') # GF(2^{32})
```

```

K = [F32.random_element() for _ in range(3)] # round keys

for i in range(3):
    print(f"K_{i + 1} =", K[i].to_integer())

def encrypt(pt: bytes):
    # encrypt with PURE cipher
    pL, pR = F32.from_bytes(pt[:4]), F32.from_bytes(pt[4:])
    for i in range(3):
        pL, pR = pR, pL + (pR + K[i])**3
    return pL.to_bytes()[1:] + pR.to_bytes()[1:]

pts = []
cts = []
M = []
v = []

for _ in range(7):
    # Chosen-plaintext with fixed left-half
    pt = bytes(range(12, 16)) + random.randbytes(4)
    ct = encrypt(pt)
    pR = F32.from_bytes(pt[4:])
    cL = F32.from_bytes(ct[:4])
    m = [pR**i for i in [0, 1, 2, 3, 4, 6, 8]]
    M.append(m)
    v.append(cL - pR**9)

M = matrix(F32, M)
v = vector(F32, v)
sol = M.solve_right(v)
print(sol[-1].to_integer())

```

3.3 Mật mã học

3.3.1 Lattice-based cryptography

Nhập môn mật mã dựa trên lattice

Kí hiệu. Vector hàng được kí hiệu bởi chữ thường in đậm, ví dụ x, y, v . Ma trận được kí hiệu bởi chữ hoa in đậm, ví dụ A, B .

➊ Definition 1.64 (Lattice)

Xét tập hợp các vector độc lập tuyến tính v_1, v_2, \dots, v_d trên \mathbb{R}^n . Ta nói **lattice** (hay **lưới**) $\mathcal{L} \subset \mathbb{R}^n$ được sinh bởi các vector v_1, v_2, \dots, v_d nếu

$$\mathcal{L} = \{a_1 v_1 + a_2 v_2 + \dots + a_d v_d : a_i \in \mathbb{Z}\}.$$

Nói cách khác, lattice là không gian vector được sinh bởi tổ hợp tuyến tính với hệ số nguyên.

Tập các vector

$$\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d\}$$

được gọi là **tập sinh** hay **cơ sở** (basis, базис) của lattice \mathcal{L} .

Số lượng vector trong cơ sở được gọi là **số chiều** của lattice và kí hiệu là $d = \dim(\mathcal{L})$.

Lattice được gọi là **full-rank** nếu $\dim(\mathcal{L}) = n$.

Xét ma trận \mathbf{V} có các hàng là các vector $\mathbf{v}_1, \dots, \mathbf{v}_d$, nghĩa là $\mathbf{V} \in \mathbb{R}^{d \times n}$.

i Definition 1.65 (Định thức của lattice)

Định thức của lattice \mathcal{L} được xác định bởi công thức $\det(\mathcal{L}) = \sqrt{|\det(\mathbf{V}\mathbf{V}^\top)|}$.

Nếu lattice full-rank thì $\det(\mathcal{L}) = \det(\mathbf{V})$.

i Remark 1.16

Cơ sở của một lưới không là duy nhất nhưng số lượng vector trong mỗi cơ sở là như nhau và bằng số chiều của lattice.

Nếu \mathbf{V} và \mathbf{W} là hai ma trận cơ sở của cùng lattice \mathcal{L} thì tồn tại ma trận \mathbf{A} với hệ số nguyên (tức $\mathbf{A} \in \mathbb{Z}^{d \times d}$) có định thức bằng 1 sao cho $\mathbf{W} = \mathbf{A} \cdot \mathbf{V}$. Chúng ta có thể dễ dàng chứng minh điều này khi biểu diễn các vector trong \mathbf{W} thành tổ hợp tuyến tính các vector trong \mathbf{V} .

Giả sử $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d\}$ là một cơ sở của \mathcal{L} . Tương tự, $\{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_d\}$ là một cơ sở khác của \mathcal{L} .

i Chứng minh

Ta có thể viết mỗi \mathbf{w}_i là tổ hợp tuyến tính của các vector \mathbf{v} như sau

$$\begin{aligned} \mathbf{w}_1 &= a_{11}\mathbf{v}_1 + a_{12}\mathbf{v}_2 + \dots + a_{1d}\mathbf{v}_d \\ \mathbf{w}_2 &= a_{21}\mathbf{v}_1 + a_{22}\mathbf{v}_2 + \dots + a_{2d}\mathbf{v}_d \\ &\vdots \\ \mathbf{w}_d &= a_{d1}\mathbf{v}_1 + a_{d2}\mathbf{v}_2 + \dots + a_{dd}\mathbf{v}_d \end{aligned}$$

Khi đó, nếu viết các vector \mathbf{w}_i thành hàng của ma trận \mathbf{W} và \mathbf{v}_j thành hàng của ma trận \mathbf{V} thì biểu diễn trên tương đương với:

$$\mathbf{W} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1d} \\ a_{21} & a_{22} & \cdots & a_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ a_{d1} & a_{d2} & \cdots & a_{dd} \end{pmatrix} \cdot \mathbf{V}.$$

Đặt

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1d} \\ a_{21} & a_{22} & \cdots & a_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ a_{d1} & a_{d2} & \cdots & a_{dd} \end{pmatrix}.$$

Do \mathbf{W} và \mathbf{V} là các cơ sở của \mathcal{L} nên nếu các vector \mathbf{w}_i có thể biểu diễn qua các vector \mathbf{v}_j thì ngược lại, các vector \mathbf{v}_j cũng có thể được biểu diễn qua các vector \mathbf{w}_i . Suy ra ma trận \mathbf{A} là ma trận khả nghịch. Do $a_{ij} \in \mathbb{Z}$ theo định nghĩa lattice, $\det(\mathbf{A}) \in \mathbb{Z}$.

Hơn nữa, vì:

$$I = \mathbf{A} \cdot \mathbf{A}^{-1} \Rightarrow 1 = \det(\mathbf{A}) \cdot \det(\mathbf{A}^{-1})$$

nên $\det(\mathbf{A}) = \pm 1$.

Mỗi lattice \mathcal{L} có một lattice đối ngẫu (dual lattice), kí hiệu là

$$\mathcal{L}^* = \{\mathbf{w} \in \mathbb{R}^n : \langle \mathbf{w}, \mathbf{x} \rangle \in \mathbb{Z} \text{ với mọi } \mathbf{x} \in \mathcal{L}\}.$$

➊ Definition 1.66 (Fundamental domain)

Cho lattice \mathcal{L} có số chiều là d với cơ sở gồm các vector $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d\}$. Ta gọi **fundamental domain** (hay **fundamental parallelepiped**) của \mathcal{L} ứng với cơ sở trên là tập

$$\mathcal{F}(\mathbf{v}_1, \dots, \mathbf{v}_d) = \{t_1 \mathbf{v}_1 + \dots + t_d \mathbf{v}_d : 0 \leq t_i < 1\}.$$

Trong mặt phẳng Oxy với hai vector \mathbf{u} và \mathbf{v} không cùng phương thì fundamental domain là hình bình hành tạo bởi hai vector đó.

➋ Remark 1.17

Gọi $\mathcal{L} \subset \mathbb{R}^n$ là lattice với số chiều là n và gọi \mathcal{F} là fundamental domain của \mathcal{L} . Khi đó mọi vector $\mathbf{w} \in \mathbb{R}^n$ đều có thể viết dưới dạng

$$\mathbf{w} = \mathbf{t} + \mathbf{v}$$

với \mathbf{t} duy nhất thuộc \mathcal{F} và \mathbf{v} duy nhất thuộc \mathcal{L} .

Một cách tương đương, hợp của các fundamental domains

$$\mathcal{F} + \mathbf{v} = \{\mathbf{t} + \mathbf{v} : \mathbf{t} \in \mathcal{F}\}$$

với \mathbf{v} là các vector trong \mathcal{L} , sẽ bao phủ hết \mathbb{R}^n .

➌ Chứng minh

Để chứng minh nhận xét trên, giả sử $\{\mathbf{v}_i : 1 \leq i \leq n\}$ là cơ sở của \mathcal{L} . Khi đó các \mathbf{v}_i độc lập tuyến tính nên cũng là cơ sở của \mathbb{R}^n .

Ta viết các vector $\mathbf{w} \in \mathbb{R}^n$ dưới dạng tổ hợp tuyến tính của \mathbf{v}_i và tách hệ số trước mỗi vector thành phần nguyên và phần lẻ. Phần nguyên cho vector trong \mathcal{L} và phần lẻ cho vector trong \mathcal{F} .

Để chứng minh tính duy nhất của tổ hợp, xét hai cách biểu diễn khác nhau của \mathbf{w} và chứng minh hai cách đó là một.

➊ Theorem 1.17 (Bất đẳng thức Hadamard)

Cho lattice \mathcal{L} , lấy cơ sở bất kỳ của \mathcal{L} là các vector $\mathbf{v}_1, \dots, \mathbf{v}_n$ và gọi \mathcal{F} là fundamental domain cho \mathcal{L} . Khi đó

$$\det L = \text{Vol}(\mathcal{F}) \leq \|\mathbf{v}_1\| \cdot \|\mathbf{v}_2\| \cdots \|\mathbf{v}_n\|.$$

Cơ sở càng gần với trực giao thì bất đẳng thức Hadamard trên càng trở thành đẳng thức.

➋ Definition 1.67 (Đa thức cyclotomic)

Với mỗi số nguyên dương N , đa thức cyclotomic thứ N là đa thức tối giản Φ_N duy nhất trong $\mathbb{Z}[x]$ sao cho $x^N - 1$ chia hết cho Φ_N nhưng $x^k - 1$ không chia hết cho Φ_N với mọi $k < N$.

➌ Example 1.35

Ví dụ, xét $x^3 - 1 = (x - 1)(x^2 + x + 1)$:

- với $k = 1$, ta có $(x^2 + x + 1) \nmid (x - 1)$;
- với $k = 2$, ta có $(x^2 + x + 1) \nmid (x^2 - 1)$;
- với $k = 3$, theo phân tích nhân tử trên thì $(x^2 + x + 1) \mid (x^3 - 1)$.

Như vậy $\Phi_3 = x^2 + x + 1$.

➍ Remark 1.18

Nếu d là số nguyên tố thì $\Phi_d = 1 + x + x^2 + \dots + x^{d-1}$.

➎ Remark 1.19

Các đa thức tối giản không chỉ đối với \mathbb{Z} mà còn đối với \mathbb{Q} . Ta cũng có thể chứng minh được rằng:

$$x^N - 1 = \prod_{d|N} \Phi_d(x).$$

➏ Definition 1.68

Với $i = 1, \dots, n$, định nghĩa $\lambda_i(\mathcal{L})$ là λ nhỏ nhất sao cho \mathcal{L} chứa ít nhất i vector độc lập của chuẩn Euclid (Euclidean norm) tại hầu hết λ . Cụ thể $\lambda_1(\mathcal{L})$ là độ dài vector khác không ngắn nhất trong \mathcal{L} .

Congruential public key cryptosystem

Thuật toán này mình lấy từ [33].

Tạo khóa

Trong thuật toán này, ta chọn số nguyên tố q làm public parameter.

Sau đó chọn hai số f và g làm private key. Hai số này phải thỏa mãn các điều kiện

$$f < \sqrt{q/2}, \quad \sqrt{q/4} < g < \sqrt{q/2}, \quad \gcd(f, qg) = 1.$$

Tính $h \equiv f^{-1}g \pmod{q}$. Khi đó public key là h .

Mã hóa

Để mã hóa thông điệp m , ta chọn số một số ngẫu nhiên r thỏa mãn

$$0 < m < \sqrt{q/4}, \quad 0 < r < \sqrt{q/2}.$$

Tiếp theo, ta tính $e \equiv rh + m \pmod{q}$.

Bản mã e thỏa mãn $0 < e < q$.

Giải mã

Từ bản mã e ta giải mã bằng cách tính

$$a \equiv fe \pmod{q}, \quad b \equiv f^{-1}a \pmod{g}.$$

Lưu ý f^{-1} là nghịch đảo modulo g . Khi đó $b \equiv m$ là message ban đầu.

Chứng minh

Để chứng minh rằng số b sau khi tính toán bằng chính xác m ban đầu ta cần xem xét điều kiện của private key và public key.

Dầu tiên ta có

$$a \equiv fe \equiv f(rh + m) \equiv f(rf^{-1}g + m) \equiv rg + fm \pmod{q}.$$

Từ điều kiện của f, g, r và m ta có

$$rg + fm < \sqrt{\frac{q}{2}} \cdot \sqrt{\frac{q}{2}} + \sqrt{\frac{q}{2}} \cdot \sqrt{\frac{q}{4}} < q.$$

Nói cách khác $rg + fm$ giữ nguyên giá trị trong phép modulo q , hay $a \equiv rg + fm$.

Ta suy ra

$$b \equiv f^{-1}a \equiv f^{-1}(rg + fm) \equiv m \pmod{g},$$

ở đây giá trị a không thay đổi khi chuyển từ modulo q sang modulo g .

Do $0 < m < \sqrt{q/4}$ và $\sqrt{q/4} < g < \sqrt{q/2}$ nên $m < g$. Nói cách khác b bằng đúng m ban đầu.

Ví dụ mã hóa và giải mã

Ta chọn số nguyên tố $q = 3973659461$ là public parameter.

Ta chọn $f = 36624$ và $g = 33577$ làm private key. Ở đây có thể kiểm tra điều kiện

$$f < \sqrt{q/2}, \quad \sqrt{q/4} < g < \sqrt{q/2}, \quad \gcd(f, qg) = 1.$$

Lúc này public key là

$$h \equiv f^{-1}g \equiv 3540857813 \pmod{q}.$$

Giả sử ta muốn mã hóa thông điệp $m = 1024$.

Ta chọn (ngẫu nhiên) giá trị $r = 21542$. Ta tính được bản mã là

$$e \equiv rh + m \equiv 2765654775 \pmod{q}.$$

Để giải mã bản mã $e = 2765654775$ với private key $f = 36624$ và $g = 33577$, đầu tiên ta tính

$$a \equiv fe \equiv 760818710 \pmod{q}.$$

Tiếp theo tính

$$b \equiv f^{-1}a \equiv 1024 \pmod{g}.$$

Như vậy b bằng chính xác bản rõ $m = 1024$ ban đầu.

Phá mã

Để tấn công hệ mật mã này ta xây dựng lattice. Để ý rằng $h = f^{-1}g \pmod{q}$, hay $fh + kq = g$ với $k \in \mathbb{Z}$.

Ta thấy rằng $f \cdot (h, 1) + k \cdot (q, 0) = (g, f)$. Như vậy cơ sở của lattice gồm hai vector $(h, 1)$ và $(q, 0)$. Thuật toán rút gọn lattice Gauss sẽ hoạt động trong trường hợp này (số chiều bằng 2).

NTRU

Đối với NTRU-HRSS mình sử dụng bài thuyết trình ở dự án PQC của NIST¹.

Đối với NTRUEncrypt thì mình tham khảo [33] (chương 7 về lattice). Mình sẽ sử dụng kí hiệu từ tài liệu này làm chuẩn cho cả NTRUEncrypt, NTRU-HRSS và NTRU.

Tham số cho NTRU

Các thuật toán NTRU hoạt động dựa trên các vành thương (quotient ring) sau:

$$R = \frac{\mathbb{Z}[x]}{x^n - 1}, \quad R_p = \frac{(\mathbb{Z}/p\mathbb{Z})[x]}{x^n - 1}, \quad R_q = \frac{(\mathbb{Z}/q\mathbb{Z})[x]}{x^n - 1},$$

trong đó p, q và n là các số nguyên tố khác nhau.

Ta cần định nghĩa một phép biến đổi gọi là *center-lift*. Trong các tài liệu khác thì gọi là phép *modular reduction*.

Với số nguyên n cố định đóng vai trò modulo, ta xét phần tử

$$r \in \{0, 1, \dots, n-1\}.$$

Khi đó center-lift của r là số $r' \in \mathbb{Z}$ sao cho $-\frac{n}{2} < r' \leq \frac{n}{2}$ và $r \equiv r' \pmod{n}$.

Ví dụ, khi $n = 8$:

¹ <https://csrc.nist.gov/presentations/2018/ntru-hrss-kem>

- với $r = 3$ thì $r' = 3$;
- với $r = 6$ thì $r' = -2$.

Ta cần thêm tập

$$\mathcal{T}(d_1, d_2) = \left\{ a(x) \in R \begin{array}{l} : a(x) \text{ có đúng } d_1 \text{ hệ số bằng } 1 \\ : a(x) \text{ có đúng } d_2 \text{ hệ số bằng } -1 \\ : a(x) \text{ có các hệ số còn lại là } 0. \end{array} \right\}.$$

NTRUEncrypt

Tạo khóa

Chọn $f(x) \in \mathcal{T}(d+1, d)$ và $g(x) \in \mathcal{T}(d, d)$ ngẫu nhiên.

Ta tính

$$\begin{aligned} F_q(x) &= f(x)^{-1} \in R_q, \\ F_p(x) &= f(x)^{-1} \in R_p. \end{aligned}$$

Tiếp theo, tính

$$h(x) = F_q(x) \cdot g(x) \in R_q.$$

Khi đó, private key là cặp $(f(x), F_p(x))$ và public key là $h(x)$.

Mã hóa

Giả sử bản rõ là đa thức $m(x) \in R$ sao cho hệ số m_i thỏa $-\frac{p}{2} < m_i \leq \frac{p}{2}$ (center-lift hệ số).

Chọn ngẫu nhiên đa thức $r(x) \in \mathcal{T}(d, d)$ và tính

$$e(x) = p \cdot h(x) \cdot r(x) + m(x) \pmod{q}.$$

Khi đó bản mã là $e(x) \in R_q$.

Giải mã

Để giải mã, ta tính

$$a(x) = f(x) \cdot e(x) \pmod{q}.$$

Sau đó ta center-lift các hệ số của $a(x)$ thành đa thức thuộc R rồi tính trong modulo p :

$$b(x) = F_p(x) \cdot a(x) \pmod{p}.$$

Khi đó $b(x)$ chính là bản rõ $m(x)$ ban đầu.

Ở đây, điều kiện của các tham số n, p, q và d để NTRUEncrypt hoạt động đúng là

$$q > (6d+1)p.$$

Tính đúng đắn của quá trình giải mã

Ta có

$$\begin{aligned} a(x) &\equiv f(x) \cdot e(x) \pmod{q} \\ &\equiv f(x)(p \cdot h(x) \cdot r(x) + m(x)) \pmod{q} \\ &\equiv p \cdot f(x) \cdot h(x) \cdot r(x) + f(x) \cdot m(x) \pmod{q} \\ &\equiv p \cdot f(x) \cdot r(x) \cdot F_q(x) \cdot g(x) + f(x) \cdot m(x) \pmod{q} \\ &\equiv p \cdot g(x) \cdot r(x) + f(x) \cdot m(x) \pmod{p} \end{aligned}$$

vì $f(x) \cdot F_q(x) \equiv 1 \pmod{q}$.

Xét đa thức $p \cdot g(x) \cdot r(x) + f(x) \cdot m(x)$ trong R (center-lift). Ta có

$$\begin{aligned} b(x) &\equiv F_p(x) \cdot a(x) \pmod{p} \\ &\equiv \underbrace{F_p(x) \cdot p \cdot g(x) \cdot r(x)}_0 + \underbrace{F_p(x) \cdot f(x) \cdot m(x)}_1 \pmod{p} \\ &\equiv m(x) \pmod{p}. \end{aligned}$$

NTRU lattice

Đặt public key

$$h(x) = h_0 + h_1(x) + \cdots + h_{n-1}x^{n-1},$$

các hệ số đa thức tương ứng với vector

$$\mathbf{h} = (h_0, h_1, \dots, h_{n-1}).$$

Đặt

$$M_{\mathbf{h}}^{\text{NTRU}} = \left(\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{n-1} \\ 0 & 1 & \cdots & 0 & h_{n-1} & h_0 & \cdots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & h_1 & h_2 & \cdots & h_0 \\ 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array} \right) = \left(\begin{array}{c|c} \mathbf{I}_n & \mathbf{h} \\ \mathbf{O}_n & q\mathbf{I}_n \end{array} \right),$$

ở đây \mathbf{I}_n là ma trận đơ đơn vị cấp n , \mathbf{O}_n là ma trận không cấp n .

Giả sử

$$\begin{aligned} a(x) &= a_0 + a_1x + \cdots + a_{n-1}x^{n-1}, \\ b(x) &= b_0 + b_1x + \cdots + b_{n-1}x^{n-1}, \end{aligned}$$

tương ứng với các vector

$$\mathbf{a} = (a_0, a_1, \dots, a_{n-1}), \quad \mathbf{b} = (b_0, b_1, \dots, b_{n-1}).$$

Ta kí hiệu

$$(\mathbf{a}, \mathbf{b}) = (a_0, a_1, \dots, a_{n-1}, b_0, b_1, \dots, b_{n-1}) \in \mathbb{Z}^{2n}.$$

Giả sử $f(x) \cdot h(x) \equiv g(x) \pmod{q}$, đặt $u(x) \in R$ là đa thức thỏa

$$f(x) \cdot g(x) = g(x) + q \cdot u(x).$$

Khi đó

$$(\mathbf{f}, -\mathbf{u}) \cdot M_h^{\text{NTRU}} = (\mathbf{f}, \mathbf{g})$$

nên vector (\mathbf{f}, \mathbf{g}) thuộc lattice L_h^{NTRU} .

NTRU-HRSS

Hệ mật mã NTRU-HRSS được nộp vòng 1 của dự án NIST PQC. Ở vòng 2, NTRU-HRSS hợp nhất với NTRUEncrypt trở thành NTRU.

Tạo khóa

Đối với NTRU-HRSS cần thêm các vành con của R_p và R_q xác định bằng đa thức cyclotomic. Ta kí hiệu đa thức cyclotomic thứ d là Φ_d . Nhận xét bên trên:

- $\Phi_1(x) = x - 1$;
- nếu d là số nguyên tố thì

$$\Phi_d(x) = 1 + x + \dots + x^{d-1}.$$

Đặt $S_p = \frac{(\mathbb{Z}/p\mathbb{Z})[x]}{\Phi_n(x)}$ và $S_q = \frac{(\mathbb{Z}/q\mathbb{Z})[x]}{\Phi_n(x)}$ với p, q và n là các số nguyên tố như trên.

Hơn nữa, Khi n là số nguyên tố thì $x^n - 1 = \Phi_1(x)^n$ và do đó

$$R_p \cong \frac{(\mathbb{Z}/p\mathbb{Z})[x]}{\Phi_1(x)} \times S_p, \quad R_q \cong \frac{(\mathbb{Z}/q\mathbb{Z})[x]}{\Phi_1(x)} \times S_q.$$

Thông thường ta chọn $p = 3$ và $n = 701$. Việc chọn n như vậy được (nhiều) người khẳng định là an toàn.

Đối với NTRU-HRSS-PKE, private key là một phần tử khác không $f \in S_p$.

Từ f ta tính public key $h = \Phi_1(x) \cdot g \cdot f^{-1} \in R_q$ với một phần tử $g \in S_p$. Điều này đòi hỏi f khả nghịch trong R_q .

Để hỗ trợ giải mã ta cần nghịch đảo của f trong S_p . Ta viết nghịch đảo của f trong R_q và S_p tương ứng là f_q^{-1} và f_p^{-1} .

Theo cấu trúc thì f khả nghịch trong cả S_p và S_q .

Thay vì lấy f và g trực tiếp từ S_p thì ta lấy từ tập con:

$$\mathcal{T}^+ = \{v \in S_p : \langle xv, v \rangle \geq 0\}.$$

Thuật toán sinh khóa gồm các bước:

1. $f \leftarrow \mathcal{T}^+$.
2. $f_p^{-1} \leftarrow f^{-1} \in S_p$.
3. $f_q^{-1} \leftarrow f^{-1} \in S_q$.
4. $g \leftarrow \mathcal{T}^+$.
5. $h = \Phi_1(x) \cdot g \cdot f^{-1} \in R_q$.
6. Trả về public key là $\text{pk} = h$ và private key là $\text{sk} = (f, f_p^{-1})$.

Mã hóa

Gọi $m \in S_p$ là encode của bản rõ và $r \in S_p$ được chọn ngẫu nhiên.

Bản rõ e được tính:

$$e \leftarrow p \cdot r \cdot h + [m]_3 \in R_q$$

Ở đây $p = 3$ là số nguyên tố bên trên.

Giải mã

Input: $(e, (f, f_3^{-1}))$ và hai vành (R_q, S_p) .

Output: plaintext m' :

$$m' \leftarrow [e \cdot f]_q \cdot f_3^{-1} \in S_p.$$

Ở đây kí hiệu $[e \cdot f]_q$ nghĩa là phép tính diễn ra trong vành R_q . Tương tự ở trên $[m]_3$ là encode của thông điệp m trong S_p .

FALCON

[TODO] Viết lại.

Falcon làm việc với các phần tử thuộc vành $\frac{\mathbb{Q}[x]}{\phi(x)}$, ở đây $\phi(x) = x^n + 1$ và $n = 2^k$.

Ở đây $\phi(x)$ là đa thức cyclotomic nên ta có

$$\phi(x) = \prod_{k \in \mathbb{Z}_m^\times} (x - \xi^k)$$

với $m = 2n$ và ξ là nghiệm bậc m của đơn vị.

\mathbb{Z}_m^\times là nhóm các phần tử khả nghịch của \mathbb{Z}_m đối với phép nhân.

Ta có quan hệ

$$\mathbb{Q} \subseteq \mathbb{Q}[x]/(x^2 + 1) \subseteq \dots \subseteq \mathbb{Q}[x]/(x^{n/2} + 1) \subseteq \mathbb{Q}[x]/(x^n + 1)$$

và chuỗi đẳng cấu

$$\mathbb{Q}^n \cong (\mathbb{Q}[x]/(x^2 + 1))^{n/2} \cong \dots \cong (\mathbb{Q}[x]/(x^{n/2} + 1))^2 \cong \mathbb{Q}[x]/(x^n + 1).$$

Đặt

$$a(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1} \in \mathcal{Q}, b(x) = b_0 + b_1x + \dots + b_{n-1}x^{n-1} \in \mathcal{Q}$$

với \mathcal{Q} là vành $\mathbb{Q}[x]/(x^n + 1)$.

Kí hiệu $a^*(\xi)$ và gọi là Hermitian adjoint của a , là phần tử duy nhất của \mathcal{Q} sao cho mọi nghiệm ξ của $\phi(x)$ ta đều có $a^*(\xi) = \overline{a(\xi)}$, trong đó $\bar{\cdot}$ là liên hợp (conjunction) trên \mathbb{C} .

Với $\phi(x) = x^n + 1$ thì Hermitian adjoint là $a^* = a_0 - (a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1})$.

Ta mở rộng định nghĩa (Hermitian) adjoint lên vector và ma trận.

Adjoin B^* của ma trận $B \in \mathcal{Q}^{n \times m}$ (tương tự với vector) là adjoint của từng phần tử (component-wise):

$$B = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \Rightarrow B^* = \begin{pmatrix} a^* & b^* \\ c^* & d^* \end{pmatrix}.$$

Tích vô hướng (inner product) của hai đa thức $a(x)$ và $b(x)$ là

$$\langle a, b \rangle = \frac{1}{\deg f(x)} \cdot \sum_{\phi(\xi)=0} a(\xi) \cdot \overline{b(\xi)}$$

đây gọi là cách biểu diễn bằng fast fourier transform.

Ta mở rộng lên vector, với $\bar{u} = (u_i)_i$ và $\bar{v} = (v_i)_i$ thuộc \mathcal{Q}^m thì

$$\langle \bar{u}, \bar{v} \rangle = \sum_i \langle u_i, v_i \rangle.$$

Cách chọn $\phi(x)$ ở trên cho chúng ta tích vô hướng giống thông thường

$$\langle a, b \rangle = \sum_{i=0}^{n-1} a_i b_i$$

đây là cách biểu diễn bằng hệ số.

Ring lattice: với vành $\mathcal{Q} = \mathbb{Q}[x]/\phi(x)$ và $\mathcal{Z} = \mathbb{Z}[x]/\phi(x)$, số nguyên dương $m \geq n$ và ma trận full-rank $B \in \mathcal{Q}^{n \times m}$, ta kí hiệu $\Lambda(B)$ và gọi lattice sinh bởi B là tập \mathcal{Z}^n . Khi đó

$$B = \{\bar{z} \cdot B : z \in \mathcal{Z}^n\}$$

Mở rộng ra, tập Λ là lattice nếu tồn tại ma trận B sao cho $\Lambda = \Lambda(B)$.

Ta có thể nói $\Lambda \subseteq \mathcal{Z}^m$ là lattice q -phân nếu $q\mathcal{Z}^m \subseteq \Lambda$.

Discrete Gaussian:

Với $\sigma, mu \in \mathbb{R}$ mà $\sigma > 0$, ta định nghĩa hàm Gauss $\rho_{\sigma, \mu}$ là

$$\rho_{\sigma, \mu} = \exp(-(x - \mu)^2 / (2\sigma^2)),$$

và phân phối Gauss rời rạc $D_{\mathbb{Z}, \sigma, \mu}$ trên vành số nguyên là

$$D_{\mathbb{Z}, \sigma, \mu}(x) = \frac{\rho_{\sigma, \mu}(x)}{\sum_{z \in \mathbb{Z}} \rho_{\sigma, \mu}(z)}.$$

Trục giao hóa Gram-Schmidt.

Mỗi ma trận $B \in \mathcal{Q}^{n \times m}$ có thể phân tích thành $B = L \times \tilde{B}$ với L là ma trận tam giác dưới (lower triangle) với các phần tử trên đường chéo chính bằng 1.

Các hàng \tilde{b}_i của \tilde{B} kiểm tra $\langle b_i, b_j \rangle = 0$ với $i \neq j$. Khi B full-rank thì phân tích này là duy nhất và được gọi là trực giao hóa Gram-Schmidt (Gram-Schmidt orthogonalization GSO).

Ta gọi chuẩn Gram-Schmidt (norm) của B là giá trị

$$\|B\|_{GS} = \max_{\tilde{b}_i \in \tilde{B}} \|\tilde{b}_i\|$$

LDL:math: ^* decomposition: LDL:math: ^* decomposition viết mỗi ma trận Gram full-rank thành tích LDL:math: ^* với:

1. $L \in \mathcal{Q}^{n \times n}$ là ma trận tam giác dưới với các phần tử 1 trên đường chéo chính.
2. $D \in \mathcal{Q}^{n \times n}$ là ma trận chéo.

Nếu tồn tại GSO duy nhất $B = L \cdot \tilde{B}$ và với ma trận Gram G full-rank thì tồn tại LDL:math: ${}^* \text{decomp}$ duy nhất $G = LDL^*$.

Nếu $G = B \cdot B^*$ thì $G = L \cdot (\tilde{B} \tilde{B}^*) \cdot L^*$ là LDL:math: ${}^* \text{decomp}$ hợp lệ.

Khi đó, $L \cdot \tilde{B}$ là GSO của B khi và chỉ khi $L \cdot (\tilde{B} \cdot \tilde{B}^*) \cdot L^*$ là LDL:math: ${}^* \text{decomp}$. của $B \cdot B^*$.

Keys.

Public params.

1. Đa thức cyclotomic $\phi(x) = x^n + 1$ với $n = 2^k$.
2. Modulus là $q \in \mathbb{N}^*$ là số nguyên tố, $\phi(x) \bmod q$ sẽ split (phân rã) trên $\mathbb{Z}_q[x]$.
3. Bound (chặn) $\lfloor \beta^2 \rfloor > 0$.
4. Độ lệch chuẩn σ và $\sigma_{\min} < \sigma_{\max}$.
5. Chữ ký với độ dài (theo byte) là `sbytelen`.

Private key

Private key gồm 4 đa thức f, g, F, G thuộc $\mathbb{Z}[x]/\phi(x)$ với hệ số ngắn, thỏa phương trình

$$f \cdot G - g \cdot F = q \bmod \phi(x).$$

Đa thức f cũng phải khả nghịch trong $\mathbb{Z}_q[x]/\phi(x)$.

Cho trước f và g , ta có thể tính được F và G nhưng sẽ rất tốn sức. Do đó ta cần lưu thêm F và từ f, g và F ta sẽ tính lại G .

FFT representation (biểu diễn FFT) của f, g, F và G là dạng ma trận

$$\hat{B} = \begin{pmatrix} FFT(g) & -FFT(f) \\ FFT(G) & -FFT(F) \end{pmatrix}.$$

Falcon trees. Falcon trees là cây nhị phân được định nghĩa đệ quy như sau:

1. Falcon tree với độ cao 0 gồm một nút đơn với giá trị là $\sigma > 0$.
2. Falcon tree với độ cao k có tính chất:
 - giá trị tại gốc, `T.value`, là đa thức $l \in \mathbb{Q}[x]/(x^n + 1)$ với $n = 2^k$;
 - nút là trái và phải, `T.leftChild` và `T.rightChild`, là Falcon tree với độ cao $k - 1$.

Nội dung của Falcon tree T được tính từ các thành phần f, g, F, G của private key và được mô tả bởi thuật toán ở sau.

Public key.

Falcon public key pk tương ứng với private key $sk = (f, g, F, G)$ là đa thức $h \in \mathbb{Z}_q[x]/\phi(x)$ thỏa

$$h = gf^{-1} \pmod{\phi(x), q}.$$

Thuật toán rút gọn lattice

Thuật toán rút gọn lattice Gauss

Gọi $L \subset \mathbb{R}^2$ là lattice 2 chiều với các vector cơ sở là v_1 và v_2 . Thuật toán rút gọn lattice Gauss (Gaussian Lattice Reduction) hoạt động như sau.

❶ Algorithm 1.2 (Thuật toán rút gọn lattice Gauss)

1. Loop

1. Nếu $\|\mathbf{v}_2\| < \|\mathbf{v}_1\|$ thì đổi chỗ \mathbf{v}_1 và \mathbf{v}_2 (swap).
2. Tính $m = \left\lfloor \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|^2} \right\rfloor$.
3. Nếu $m = 0$ thì trả về cơ sở mới gồm các vector \mathbf{v}_1 và \mathbf{v}_2 .
4. Thay \mathbf{v}_2 thành $\mathbf{v}_2 - m\mathbf{v}_1$.
2. Tiếp tục loop :)))

Sau khi thuật toán kết thúc thì \mathbf{v}_1 là vector khác không ngắn nhất trong L , và do đó bài toán SVP được giải quyết. Hơn nữa góc θ giữa \mathbf{v}_1 và \mathbf{v}_2 thỏa mãn

$$|\cos \theta| \leq \|\mathbf{v}_1\|/2\|\mathbf{v}_2\|,$$

cụ thể là $\frac{\pi}{3} \leq \theta \leq \frac{2\pi}{3}$.

❷ Chứng minh

Dầu tiên ta chứng minh \mathbf{v}_1 là vector khác không ngắn nhất trong L .

Giả sử thuật toán kết thúc và trả về hai vector \mathbf{v}_1 và \mathbf{v}_2 . Bước 1 của vòng lặp đảm bảo rằng $\|\mathbf{v}_2\| \geq \|\mathbf{v}_1\|$ và

$$\frac{|\mathbf{v}_1 \cdot \mathbf{v}_2|}{\|\mathbf{v}_1\|^2} \leq \frac{1}{2} \quad (3.21)$$

vì đây là điều kiện để làm tròn $\frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|^2}$ thành 0 và thỏa bước 3.

Mỗi vector \mathbf{v} khác không trong L đều biểu diễn được dưới dạng

$$\mathbf{v} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2, \quad a_1, a_2 \in \mathbb{Z}.$$

Sử dụng (3.21) và bất đẳng thức quen thuộc $|x| \geq -x$ với mọi $x \in \mathbb{R}$ ta có

$$\begin{aligned} \|\mathbf{v}\|^2 &= \|a_1\mathbf{v}_1 + a_2\mathbf{v}_2\|^2 \\ &= a_1^2\|\mathbf{v}_1\|^2 + 2a_1a_2 \cdot \mathbf{v}_1 \cdot \mathbf{v}_2 + a_2^2\|\mathbf{v}_2\|^2 \\ &\geq a_1^2\|\mathbf{v}_1\|^2 - 2|a_1a_2| \cdot \mathbf{v}_1 \cdot \mathbf{v}_2 + a_2^2\|\mathbf{v}_2\|^2 \\ &\geq a_1^2\|\mathbf{v}_1\|^2 - 2|a_1a_2| \cdot \|\mathbf{v}_1\|^2 + a_2^2\|\mathbf{v}_2\|^2 \\ &\geq a_1^2\|\mathbf{v}_1\|^2 - 2|a_1a_2| \cdot \|\mathbf{v}_1\|^2 + a_2^2\|\mathbf{v}_1\|^2 \quad (\text{vì } \|\mathbf{v}_2\| \geq \|\mathbf{v}_1\|) \\ &= (a_1^2 - 2|a_1| \cdot |a_2| + a_2^2)\|\mathbf{v}_1\|^2. \end{aligned}$$

Ta sử dụng bất đẳng thức quen thuộc: với mọi $t_1, t_2 \in \mathbb{R}$ thì

$$t_1^2 - 2t_1t_2 + t_2^2 = \left(t_1 - \frac{t_2}{2}\right)^2 + \frac{3}{4} \cdot t_2^2 = \frac{3}{4} \cdot t_1^2 + \left(\frac{t_1}{2} - t_2\right)^2 \geq 0.$$

Biểu thức bằng 0 khi và chỉ khi $t_1 = t_2 = 0$. Vì a_1 và a_2 là các số nguyên không đồng thời bằng 0 nên có thể suy ra $\|\mathbf{v}\|^2 \geq \|\mathbf{v}_1\|^2$ với mọi vector \mathbf{v} khác không trong L . Nói cách khác, \mathbf{v}_1 là vector khác không ngắn nhất trong L và bài toán SVP được giải xong.

Tiếp theo, với $\cos \theta$, ta có

$$|\cos \theta| = \left| \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\| \cdot \|\mathbf{v}_2\|} \right| = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|^2} \cdot \frac{\|\mathbf{v}_1\|}{\|\mathbf{v}_2\|} \leq \frac{1}{2} \cdot \frac{\|\mathbf{v}_1\|}{\|\mathbf{v}_2\|}.$$

Hơn nữa $\|\mathbf{v}_2\| \geq \|\mathbf{v}_1\|$ nên suy ra

$$|\cos \theta| \leq \frac{1}{2} \cdot \frac{\|\mathbf{v}_1\|}{\|\mathbf{v}_2\|} \leq \frac{1}{2} \iff -\frac{1}{2} \leq \cos \theta \leq \frac{1}{2} \iff -\frac{\pi}{3} \leq \theta \leq \frac{2\pi}{3}.$$

Mình sẽ ví dụ thuật toán trên qua việc phá mã congruent public key cryptosystem với số liệu đã trình bày trong bài viết.

Số nguyên tố $q = 3973659461$ là public parameter.

Ta đã chọn $f = 36624$ và $g = 33577$ làm private key. Ở đây f và g điều kiện

$$f < \sqrt{q/2}, \quad \sqrt{q/4} < g < \sqrt{q/2}, \quad \gcd(f, qg) = 1.$$

Lúc này public key là

$$h \equiv f^{-1}g \equiv 3540857813 \pmod{q}.$$

Để ý rằng $h = f^{-1}g \pmod{q}$, tương đương $fh + kq = g$ với $k \in \mathbb{Z}$.

Ta thấy rằng $f \cdot (h, 1) + k \cdot (q, 0) = (g, f)$. Như vậy cơ sở của lattice gồm hai vector

$$\{(h, 1), (q, 0)\}.$$

Điều kiện của f và g cho ta tính chất quan trọng: vector (g, f) ngắn trong lattice. Do đó thuật toán rút gọn lattice Gauss sẽ hoạt động trong trường hợp này (số chiều bằng 2).

Đặt $\mathbf{v}_1 = (h, 1) = (3540857813, 1)$ và $\mathbf{v}_2 = (q, 0) = (3973659461, 0)$.

Bước 1, ta có

$$\|\mathbf{v}_1\| = \sqrt{12537674051883142970} > \|\mathbf{v}_2\| = 3973659461$$

nên ta giữ nguyên \mathbf{v}_1 và \mathbf{v}_2 .

Tiếp theo ta tính

$$m = \left\lfloor \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|^2} \right\rfloor = \left\lfloor \frac{14070163148683218793}{12537674051883142970} \right\rfloor = 1 \neq 0$$

nên ta thay \mathbf{v}_2 bởi $\mathbf{v}_2 - m\mathbf{v}_1$ thì

$$\mathbf{v}_2 = (432801648, -1).$$

Bước 2, hiện tại

$$\mathbf{v}_1 = (3540857813, 1), \quad \mathbf{v}_2 = (432801648, -1)$$

nên

$$\|\mathbf{v}_1\| < \|\mathbf{v}_2\|,$$

đổi chỗ \mathbf{v}_1 và \mathbf{v}_2 ta được

$$\mathbf{v}_1 = (432801648, -1), \quad \mathbf{v}_2 = (3540857813, 1).$$

Tiếp theo ta tính

$$m = \left\lfloor \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|^2} \right\rfloor = \left\lfloor \frac{1532489096800075823}{187317266511515905} \right\rfloor = 8 \neq 0$$

nên ta thay \mathbf{v}_2 bởi $\mathbf{v}_2 - m\mathbf{v}_1$ thì

$$\mathbf{v}_2 = (78444629, 9).$$

Cứ tiếp tục như vậy, vector \mathbf{v}_1 , \mathbf{v}_2 và giá trị m sau bước 3 ở mỗi vòng lặp sẽ được thể hiện ở bảng sau.

Loop	\mathbf{v}_1	\mathbf{v}_2	m
1	(3540857813, 1)	(3973659461, 0)	1
2	(432801648, -1)	(3540857813, 1)	8
3	(78444629, 9)	(432801648, -1)	6
4	(-37866126, -55)	(78444629, 9)	-2
5	(2712377, -101)	(-37866126, -55)	-14
6	(107152, -1469)	(2712377, -101)	25
7	(33577, 36624)	(107152, -1469)	1
8	(33577, 36624)	(73575, -38093)	0

Vector \mathbf{v}_1 ở bước cuối chính xác là (g, f) bên trên.

Phương pháp Coppersmith

Phương pháp Coppersmith được dùng để tìm nghiệm nhỏ của đa thức trên modulo. Phần này mình tham khảo chính từ [34].

Ý tưởng

Giả sử ta có phương trình $F(x) \equiv 0 \pmod{M}$. Với số X cố định cho trước, phương pháp Coppersmith cho phép tìm nghiệm x_0 nhỏ thỏa mãn $|x_0| \leq X$.

Ý tưởng của phương pháp này là thay vì tìm nghiệm x_0 của $F(x)$ trên modulo M , chúng ta sẽ mở rộng lên, tìm một hàm $G(x)$ nào đó mà có nghiệm x_0 trên \mathbb{Z} .

Đơn giản nhất là

$$G(x) = k \cdot F(x) + M \cdot g(x), \quad k \in \mathbb{Z}$$

và $\deg g(x) \leq \deg F(x)$. Rõ ràng khi modulo hai vế cho M thì $G(x_0) = F(x_0) = 0 \pmod{M}$.

Phương pháp này giúp tìm nghiệm của một đa thức bậc nhỏ modulo M . Do đó giả sử đặt:

$$F(x) = a_0 + a_1x + \cdots + a_dx^d$$

với $a_i \in \mathbb{Z}$.

Lúc này chúng ta sẽ tìm hàm $G(x)$ trên với hệ số nhỏ.

Giả sử

$$g(x) = b_0 + b_1x + \cdots + b_dx^d$$

với $b_i \in \mathbb{Z}$.

Khi đó

$$G(x) = (k \cdot a_0 + M \cdot b_0) + (k \cdot a_1 + M \cdot b_1)x + \cdots + (k \cdot a_d + M \cdot b_d)x^d.$$

Ta mong muốn các hệ số $k \cdot a_0 + M \cdot b_0, k \cdot a_1 + M \cdot b_1, \dots, k \cdot a_d + M \cdot b_d$ nhỏ so với M .

Do đó với số X cho trước, nếu ta xây lattice ($d+2$ vector) sau:

$$\begin{array}{ccccccccc} \mathbf{v}_0 & \Leftarrow & M & 0 & 0 & \cdots & 0 & 0 \\ \mathbf{v}_1 & \Leftarrow & 0 & MX & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{v}_d & \Leftarrow & 0 & 0 & 0 & \cdots & 0 & MX^d \\ \mathbf{v}_{d+1} & \Leftarrow & a_0 & a_1X & a_2X^2 & \cdots & a_{d-1}X^{d-1} & a_dX^d \end{array}$$

Khi đó hệ số của mỗi dòng từ \mathbf{v}_0 tới \mathbf{v}_d là b_0 tới b_d . Còn hệ số của \mathbf{v}_{d+1} là k .

Tuy nhiên chúng ta thường sẽ biến đổi để đa thức trở thành monic ($a_d = 1$). Khi đó chúng ta bỏ đi v_d và còn $d+1$ vector trong lattice.

Cải tiến thuật toán

Dạng đơn giản

Giả sử $k(x)$ có bậc là h . Đặt $k(x) = c_0 + c_1x + \cdots + c_hx^h$.

Khi đó:

$$\begin{aligned} G(x) &= k(x) \cdot F(x) + M \cdot g(x) \\ &= (c_0 + c_1x + \cdots + c_hx^h) \cdot (a_0 + a_1x + \cdots + x^d) + M \cdot (b_0 + b_1x + \cdots + b_dx^d) \\ &= c_0 \cdot F(x) + c_1 \cdot xF(x) + \cdots + c_h \cdot x^hF(x) + M \cdot (b_0 + b_1x + \cdots + b_dx^d) \end{aligned}$$

Lúc này mỗi đại lượng $F(x), xF(x), \dots, x^hF(x)$ sẽ khiến hệ số của $F(x)$ ban đầu tăng bậc. Nói cách khác hệ số a_i của x^i trong $F(x)$ sẽ là hệ số của x^{i+j} trong $x^jF(x)$.

Sách giáo khoa nói rằng nếu mình chọn $h = d - 1$ thì phương pháp Coppersmith sẽ cho ra kết quả nếu X được chọn thỏa $X \approx M^{1/(2d-1)}$.

Tương tự, mình sẽ có các vector trong lattice như sau:

$$\begin{array}{ccccccccc} \mathbf{v}_0 & \Leftarrow & M & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots \\ \mathbf{v}_1 & \Leftarrow & 0 & MX & 0 & \cdots & 0 & 0 & 0 & \cdots \\ \vdots & \vdots \\ \mathbf{v}_{d-1} & \Leftarrow & 0 & 0 & 0 & \cdots & MX^{d-1} & 0 & 0 & \cdots \\ \mathbf{v}_d & \Leftarrow & a_0 & a_1X & a_2X^2 & \cdots & a_{d-1}X^{d-1} & X^d & 0 & \cdots \end{array}$$

Sau đó thêm các vector khi shift vào, tương ứng với $xF(x), x^2F(x), \dots$

$$\begin{array}{ccccccccc} \mathbf{v}_{d+1} & \Leftarrow & 0 & a_0X & a_1X^2 & a_2X^3 & \cdots & a_{d-1}X^d & a_dX^{d+1} & 0 & 0 & \cdots \\ \mathbf{v}_{d+2} & \Leftarrow & 0 & 0 & a_0X^2 & a_1X^3 & \cdots & a_{d-2}X^d & a_{d-1}X^{d+1} & a_dX^{d+2} & 0 & \cdots \end{array}$$

Tuy nhiên vấn đề ở đây là bound của nghiệm bị thu hẹp lại. Ban nãy mình nói rằng nếu chọn $h = d - 1$ thì nghiệm cho kết quả nếu $X \approx M^{1/(2d-1)}$. Ở đây $M = 10001$ nên $X \approx 4$. Trong khi ở bài viết trước thì $X = 10$. Do đó có thể thấy việc mở rộng này đòi hỏi kém hiệu quả tùy thuộc vào h .

Dạng nâng cao

Coppersmith đã đề xuất ý tưởng như sau:

❶ Lemma 1.1 (Coppersmith)

Với $0 < \epsilon < \min\{0, 18; 1/d\}$. Đặt $F(x)$ là đa thức monic bậc d có một hoặc nhiều nghiệm x_0 trên modulo M sao cho $|x_0| \leq \frac{1}{2}M^{1/d-\epsilon}$. Khi đó x_0 có thể tính với thời gian đa thức giới hạn bởi $d, 1/\epsilon$ và $\log(M)$.

Để chứng minh bở đê này thì Coppersmith đã xây dựng một hệ lattice để tính toán và cũng là cách xây dựng lattice sẽ đê cập tới đây.

Chúng ta biết rằng x_0 là nghiệm của đa thức $F(x) \equiv 0 \pmod{M}$. Do đó ta suy ra $F(x_0)^k \equiv 0 \pmod{M^k}$.

Từ nhận xét này, mình mở rộng phần cơ bản lên tìm nghiệm trong modulo M^{h-1} với h là một số được chọn trước.

Với $k(x) = c_0 + c_1x + \dots + c_{d-1}x^{d-1}$ biểu diễn việc shift thành $F(x), xF(x), x^2F(x), \dots, x^{d-1}F(x)$.

Ta xét h đa thức sau:

$$\begin{aligned} M^{h-1}F(x)^0 k(x) &\equiv 0 \pmod{M^{h-1}} \\ M^{h-2}F(x)^1 k(x) &\equiv 0 \pmod{M^{h-1}} \\ &\dots && \dots && \dots \\ M^0 F(x)^{h-1} k(x) &\equiv 0 \pmod{M^{h-1}} \end{aligned}$$

Với mỗi đa thức $M^{h-1-j}F(x)^j$ chúng ta có d lần shift tương ứng với từng hệ số của $k(x)$. Cụ thể là $M^{h-1-j}F(x)^j, M^{h-1-j}F(x)^jx, M^{h-1-j}F(x)^jx^2, \dots, M^{h-1-j}F(x)^jx^{d-1}$.

Do đó có tất cả dh vector trong lattice. Số h thường được chọn sao cho $dh \approx \epsilon$. Và chẵn nghiệm X có thể chọn là $\frac{1}{2}M^{1/d-\epsilon}$ theo như bở đê.

Như vậy mình xây lattice như sau:

- Bước 1. Với $M^{h-1}F(x)^0$ thì các vector sau lần lượt tương ứng với

$$M^{h-1}F(x)^0, M^{h-1}F(x)^0x, \dots, M^{h-1}F(x)^0x^{d-1}$$

Nói cách khác thì:

$$\begin{array}{rcl} \mathbf{v}_0 & \Leftarrow & M^{h-1} & 0 & 0 & \cdots & 0 & 0 & \cdots \\ \mathbf{v}_1 & \Leftarrow & 0 & M^{h-1}X & 0 & \cdots & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{v}_{d-1} & \Leftarrow & 0 & 0 & 0 & \cdots & M^{h-1}X^{d-1} & 0 & \cdots \end{array}$$

- Bước 2. Với $M^{h-2}F(x)^1$ thì các vector sau lần lượt tương ứng với

$$M^{h-2}F(x)^1, M^{h-2}F(x)^1x, \dots, M^{h-2}F(x)^1x^{d-1}.$$

Nói cách khác thì:

$$\begin{array}{rcl} \mathbf{v}_d & \Leftarrow & Ma_0 & Ma_1X & Ma_2X^2 & \cdots & Ma_{d-1}X^{d-1} & MX^d & \cdots & \cdots \\ \mathbf{v}_{d+1} & \Leftarrow & 0 & Ma_0X & Ma_1X^2 & \cdots & Ma_{d-2}X^{d-1} & Ma_{d-1}X^d & MX^{d+1} & \cdots \\ \vdots & \vdots \\ \mathbf{v}_{2d-1} & \Leftarrow & 0 & 0 & 0 & \cdots & Ma_0X^{d-1} & Ma_1X^d & \cdots & \cdots \end{array}$$

- Bước thứ j . Cứ như vây với $M^{h-1-j}F(x)^j$.

Chạy LLL trên lattice trên sẽ cho kết quả ^^.

Code thử nghiệm**So sánh ví dụ đơn giản với small_roots của SageMath**

```

from sage.all import *

def small_roots(f, M, X):
    d = f.degree()

    P = f.base_ring()
    B = []
    for i in range(d):
        b = [0] * (d + 1)
        b[i] = M * X**i
        B.append(b)
    B.append([j*X**i for i,j in enumerate(f.coefficients())])
    B = Matrix(ZZ, B)
    B = B.LLL()
    bf = B[0]
    g = sum((j//(X**i)) * x**i for i,j in enumerate(bf))
    roots = g.roots()
    return [root[0] for root in roots]

M = 10001
X = 10
P = PolynomialRing(ZZ, 'x')
Q = PolynomialRing(Zmod(M), 'xn')
x = P.gen()
xn = Q.gen()
f = x**3 + 10*x**2 + 5000*x - 222
g = f.change_ring(Q).subs(x=xn)
print(small_roots(f, M, X))
print(g.small_roots(X=10))

```

Cải tiến dạng đơn giản

```

from sage.all import *

def small_roots(f, M, X):
    d = f.degree()
    B = []
    for i in range(d):
        b = [0] * (2*d)
        b[i] = M * X**i
        B.append(b)
    for i in range(d):
        g = x**i * f
        b = [v * X**u for u, v in enumerate(g.coefficients(sparse=False))]
        b = b + [0] * (2*d - len(b))
        B.append(b)
    B = Matrix(ZZ, B)
    B = B.LLL()

```

(continues on next page)

(continued from previous page)

```

bf = B[0]
g = sum((j // (X**i)) * x**i for i, j in enumerate(bf))
roots = g.roots()
return [root[0] for root in roots]

M = 10001
X = 10
P = PolynomialRing(ZZ, 'x')
x = P.gen()
f = x**3 + 10*x**2 + 5000*x - 222
print(small_roots(f, M, X))

```

Cải tiến dạng nâng cao

```

from sage.all import *

def small_roots(f, M, h = None, epsilon = None, X = None):
    d = f.degree()
    if not h:
        h = d
    if not epsilon:
        epsilon = 1/(d*h)
    if not X:
        X = round(0.5*M**(1/d-epsilon))
    B = []
    for j in range(h):
        g = M**(h-1-j) * f**j
        for i in range(d):
            k = g * x**i
            b = [v * X**u for u, v in enumerate(k.coefficients(sparse=False))]
            b = b + [0] * (d*h - len(b))
            B.append(b)

    B = Matrix(ZZ, B)
    B = B.LLL()
    bf = B[0]
    g = sum((j // (X**i)) * x**i for i, j in enumerate(bf))
    roots = g.roots()
    return [root[0] for root in roots]

# Test theo sách giáo khoa

M = (2**30 + 3)*(2**32 + 15)
P = PolynomialRing(ZZ, 'x')
x = P.gen()
f = 1942528644709637042 + 1234567890123456789*x + 987654321987654321*x**2 + x**3
print(small_roots(f, M))

# Tự test

M = (2**20 + 7)*(2**21 + 17)

```

(continues on next page)

(continued from previous page)

```
P = PolynomialRing(ZZ, 'x')
x = P.gen()
f = x**3 + (2**25 - 2883584)*x**2 + 46976195*x + 227
print(small_roots(f, M, X = 2**9))
```

3.3.2 Code-based cryptography

Introduction

➊ Definition 2.37 (Block code)

Block code $(n)_q$ là tập hợp \mathcal{C} bất kì chứa các vector độ dài n trên trường \mathbb{F}_q .

➋ Definition 2.38 (Codeword)

Codeword (hay **tử mā**) là bất kì vector nào trong block code \mathcal{C} .

➌ Definition 2.39 (Lực lượng của code)

Lực lượng (hay **cardinality, мощность**) M của block code \mathcal{C} là số lượng codeword trong \mathcal{C} .

Kí hiệu $M = |\mathcal{C}|$.

➍ Definition 2.40

Block code \mathcal{C} trên trường \mathbb{F}_q có độ dài n và có lực lượng M được gọi là $(n, M)_q$ code.

➎ Definition 2.41 (Độ dài của code)

Độ dài của $(n)_q$ code \mathcal{C} là số n chỉ độ dài mỗi codeword (độ dài mỗi vector).

➏ Definition 2.42 (Khoảng cách tối thiểu)

Khoảng cách tối thiểu (hay **minimum distance**) của code \mathcal{C} là số d chỉ khoảng cách Hamming nhỏ nhất giữa hai codeword trong code \mathcal{C} :

$$d = \min_{\mathbf{u}, \mathbf{v} \in \mathcal{C}, \mathbf{u} \neq \mathbf{v}} \rho(\mathbf{u}, \mathbf{v}) = \min_{\mathbf{u}, \mathbf{v} \in \mathcal{C}, \mathbf{u} \neq \mathbf{v}} \text{wt}(\mathbf{u} - \mathbf{v}).$$

Nhắc lại, khoảng cách Hamming của một vector là số phần tử khác 0 trong vector đó:

$$\text{wt}(x_1, x_2, \dots, x_n) = |\{x_i : x_i \neq 0\}|.$$

❶ Definition 2.43

Block code \mathcal{C} trên trường \mathbb{F}_q với độ dài n , lực lượng M và có khoảng cách tối thiểu d được gọi là $(n, M, d)_q$ code.

Thông tin ban đầu có thể được chia thành các đoạn độ dài k và mỗi đoạn như vậy được encode thành một đoạn độ dài n .

Cụ thể hơn, **hàm encode** biến đổi một thông điệp là vector \mathbf{a} độ dài k thành codeword $\mathbf{c} \in \mathcal{C}$ với độ dài n :

$$\varphi : V_k(q) \rightarrow \mathcal{C}, \quad \varphi(\mathbf{a}) = \mathbf{c}.$$

Ánh xạ φ cần là đơn ánh (one-to-one) vì chúng ta không muốn hai thông điệp độ dài k cùng encode ra một codeword độ dài n (khi decode chúng ta không biết sẽ ra thông điệp nào).

❶ Lemma 2.1

Để $(n, M)_q$ code có thể encode được các thông điệp độ dài k thì điều kiện cần và đủ để tồn tại ánh xạ φ như trên là $k \leq \lfloor \log_q M \rfloor$.

❶ Definition 2.44 (Спектр весов)

Phổ trọng số (hay **спектр весов**) của $(n)_q$ code \mathcal{C} là một vector các số nguyên không âm $(A_0, A_1, \dots, A_n) \in \mathbb{Z}_{\geq 0}^{n+1}$ với A_i là số lượng vector có trọng số Hamming bằng i trong code.

Bài toán decode tổng quát

Khi vector \mathbf{x} được truyền qua kênh truyền, không gì đảm bảo chúng ta sẽ nhận được chính xác \mathbf{x} ở bên nhận, mà có thể là \mathbf{y} nào đó.

Các codeword nằm trong một block code nào đó là tập con của $V_n(q)$, trong khi ở bên nhận có thể là một vector bất kỳ trong $V_n(q)$. Bài toán decode đặt ra câu hỏi, làm sao biết được vector \mathbf{y} sẽ được decode thành vector nào trong block code.

❶ Definition 2.45 (Bài toán decode)

Bài toán decode của $(n)_q$ code \mathcal{C} là bài toán cho trước vector $\mathbf{y} \in V_n(q)$, ta tìm codeword $\mathbf{c} \in \mathcal{C}$ của code $(n)_q$ sao cho $\rho_H(\mathbf{c}, \mathbf{y}) = \min_{\mathbf{c}' \in \mathcal{C}} \rho_H(\mathbf{c}', \mathbf{y})$ với ρ_H là khoảng cách Hamming.

Input:

- vector $\mathbf{y} \in V_n(q)$ nhận được từ kênh truyền;
- \mathcal{C} là $(n)_q$ code.

Output:

- vector $\mathbf{c} \in \mathcal{C}$ là codeword sao cho $\rho_H(\mathbf{c}, \mathbf{y}) = \min_{\mathbf{c}' \in \mathcal{C}} \rho_H(\mathbf{c}', \mathbf{y})$.

Definition 2.46 (Bài toán decode)

Bài toán decode của $(n)_q$ code \mathcal{C} là bài toán cho trước vector $\mathbf{y} \in V_n(q)$, ta tìm codeword $\mathbf{c} \in \mathcal{C}$ của code $(n)_q$ sao cho $\text{wt}(\mathbf{y} - \mathbf{c}) = \min_{\mathbf{c}' \in \mathcal{C}} \text{wt}(\mathbf{y} - \mathbf{c}')$ với wt là trọng số Hamming.

Input:

- vector $\mathbf{y} \in V_n(q)$ là vector nhận được từ kênh truyền;
- \mathcal{C} là $(n)_q$ code.

Output:

- vector $\mathbf{c} \in \mathcal{C}$ là codeword sao cho $\text{wt}(\mathbf{y} - \mathbf{c}) = \min_{\mathbf{c}' \in \mathcal{C}} \text{wt}(\mathbf{y} - \mathbf{c}')$.

Definition 2.47 (Bài toán decode)

Bài toán decode của $(n)_q$ code \mathcal{C} là bài toán cho trước vector $\mathbf{y} \in V_n(q)$, ta tìm vector $\mathbf{e} \in V_n(q)$ sao cho $\mathbf{y} - \mathbf{e} \in \mathcal{C}$, nghĩa là $\mathbf{y} - \mathbf{e}$ là codeword, và vector \mathbf{e} có trọng số Hamming nhỏ nhất $\text{wt}(\mathbf{e}) \rightarrow \min$.

Input:

- vector $\mathbf{y} \in V_n(q)$ là vector nhận được từ kênh truyền;
- \mathcal{C} là $(n)_q$ code.

Output:

- vector $\mathbf{e} \in V_n(q)$ sao cho $\mathbf{y} - \mathbf{e} \in \mathcal{C}$, nghĩa là $\mathbf{y} - \mathbf{e}$ là codeword;
- và vector \mathbf{e} có trọng số Hamming nhỏ nhất $\text{wt}(\mathbf{e}) \rightarrow \min$.

Remark 2.20

Các định nghĩa về bài toán decode ở trên tương đương nhau.

Definition 2.48 (Decoder)

Decoder của code $(n)_q$ \mathcal{C} là thuật toán mà với vector $\mathbf{y} \in V_n(q)$ cho trước, tìm được codeword $\mathbf{c} \in \mathcal{C}$ sao cho vector $\mathbf{e} = \mathbf{y} - \mathbf{c}$ có trọng số Hamming nhỏ nhất có thể, nghĩa là $\mathbf{c} = \underset{\mathbf{u} \in \mathcal{C}}{\operatorname{argmin}} \text{wt}(\mathbf{y} - \mathbf{u})$.

Decoder của code \mathcal{C} sẽ giải bài toán decode trên code \mathcal{C} .

Linear code**Definition (Linear code)**

Block code $(n)_q$ \mathcal{C} được gọi là **tuyến tính** (hay **linear**) nếu với mọi $a, b \in \mathbb{F}_2$ và với mọi codeword $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ thì vector $a \cdot \mathbf{x} + b \cdot \mathbf{y}$ cũng là codeword thuộc \mathcal{C} .

Ta kí hiệu linear code (mã tuyến tính) là $[n]_q$. Có thể thấy block code $[n]_q$ là không gian vector con của $V_n(q)$.

➊ Definition

Đối với code $[n]_q$ thì số n được gọi là **độ dài** của code.

➊ Definition

Số chiều của code $[n]_q$ là số k bằng với số chiều của không gian vector con \mathcal{C} của $V_n(q)$.

➊ Definition

Linear code \mathcal{C} có độ dài n và số chiều k được gọi là $[n, k]_q$ code.

➊ Definition

Tốc độ truyền của $[n, k]_q$ code là số $R = \frac{k}{n}$.

➊ Definition

Redundancy (hay *избыточность*) của $[n, k]_q$ code là số $r = n - k$.

➊ Definition

Khoảng cách nhỏ nhất của code \mathcal{C} là số d bằng với trọng số Hamming nhỏ nhất của các codeword trong \mathcal{C}

$$d = \min_{\mathbf{u} \in \mathcal{C}, \mathbf{u} \neq \mathbf{0}} \text{wt}(\mathbf{u}).$$

➊ Definition

Linear code $[n, k]_q$ với khoảng cách nhỏ nhất d được gọi là $[n, k, d]_q$ code.

Ma trận sinh

➊ Definition

Ma trận G được gọi là **ma trận sinh** (hay *порождающая матрица*) của code C nếu nó chứa các vector trong cơ sở của không gian vector con \mathcal{C} .

Nếu \mathcal{C} là $[n, k]_q$ code thì G là ma trận kích thước $k \times n$.

Remark

Nếu G là ma trận sinh của $[n, k]_q$ code thì

$$\mathcal{C} = \{\mathbf{a} \cdot \mathbf{G} : \mathbf{a} \in V_k(q)\}.$$

Ở đây ta nói ma trận \mathbf{G} sinh ra code \mathcal{C} .

Theo kiến thức đại số tuyến tính, nếu \mathbf{G}_1 và \mathbf{G}_2 kích thước $k \times k$ cùng sinh ra một code \mathcal{C} thì tồn tại ma trận khả nghịch \mathbf{A} kích thước $k \times k$ trên \mathbb{F}_q sao cho $\mathbf{A}\mathbf{G}_1 = \mathbf{G}_2$.

Coder

Definition

Coder $\varphi : V_k(q) \rightarrow V_n(q)$ cho $[n, k]_q$ code xác định bởi ma trận sinh \mathbf{G} theo nghĩa:

$$\varphi((a_1, \dots, a_k)) = (a_1, \dots, a_k) \cdot \mathbf{G}.$$

Definition

Ma trận \mathbf{G} kích thước $k \times n$ được gọi là **systematic** nếu nó có dạng:

$$\mathbf{G} = (\mathbf{I}_k \parallel \mathbf{G}_0),$$

với \mathbf{I}_k là ma trận đơn vị $k \times k$ và \mathbf{G}_0 là ma trận cỡ $k \times (n - k)$ nào đó.

Definition

Code được gọi là **systematic** nếu nó có ma trận sinh \mathbf{G} là systematic.

Definition

Coder $\varphi_{\mathbf{G}}$ của systematic code $[n, k]_q$, xác định bởi ma trận sinh systematic $\mathbf{G} = (\mathbf{I}_k \parallel \mathbf{G}_0)$, được gọi là **systematic**.

Khi đó với mọi thông điệp $\mathbf{a} = (a_1, \dots, a_k) \in V_k(q)$ thì coder thực hiện:

$$\mathbf{a} = (a_1, \dots, a_n) \xrightarrow{\varphi} (a_1, \dots, a_k, \mathbf{a} \cdot \mathbf{G}_0).$$

Lưu ý rằng không phải code nào cũng là systematic và do đó không chắc chắn tồn tại systematic coder.

Definition

Ta đánh số tọa độ các codeword trong $[n, k]_q$ code từ 1 tới n .

Tập thông tin (hay **information set**, **информационное множество**) \mathcal{I} của code $[n, k]_q$ là tập con của tập đánh số tọa độ, nghĩa là $\mathcal{I} \subseteq \{1, \dots, n\}$, sao cho $|\mathcal{I}| = k$ và ma trận con \mathbf{G}_k kích thước $k \times k$ của ma trận sinh \mathbf{G} khả nghịch.

Ma trận kiểm tra

Definition

Nếu \mathcal{C} là hạt nhân của ma trận \mathbf{H} , tức là

$$\mathcal{C} = \ker(\mathbf{H}) = \{\mathbf{v} \in \mathcal{C} : \mathbf{H}\mathbf{v}^\top = \mathbf{0}\},$$

thì ta nói \mathbf{H} là **ma trận kiểm tra chẵn lẻ** (hay **parity-check matrix**).

Ta có thể thấy $\mathbf{G}\mathbf{H}^\top = \mathbf{0}$.

Definition

Syndrome $S_{\mathbf{H}}(\mathbf{y})$ của vector $\mathbf{y} \in V_n(q)$ tương ứng với ma trận parity-check \mathbf{H} của $[n, k]_q$ code \mathcal{C} là vector cột $S_{\mathbf{H}}(\mathbf{y}) = \mathbf{H} \cdot \mathbf{y}^\top$.

Khi đó $S_{\mathbf{H}}(\mathbf{y})$ có độ dài $r = n - k$ -- chính là redundancy ở trên.

Remark

Nếu \mathbf{H}_1 và \mathbf{H}_2 là hai ma trận parity-check của code \mathcal{C} thì $S_{\mathbf{H}_1}(\mathbf{y}) = \mathbf{A} \cdot S_{\mathbf{H}_2}(\mathbf{y})$ với \mathbf{A} là ma trận khả nghịch thỏa mãn $\mathbf{H}_2 = \mathbf{A} \cdot \mathbf{H}_1$.

Linear code

Do $[n]_q$ code cũng là $(n)_q$ code nên ta cũng có định nghĩa phổ biến như $(n)_q$:

Definition

Спектр весов của $[n]_q$ code \mathcal{C} là vector $(A_0, A_1, \dots, A_n) \in \mathbb{Z}_{\geq 0}^{n+1}$ với $A_i \geq 0$ là số lượng vector có trọng số bằng i trong code.

Dual code (mã đối ngẫu) và kiểm tra chẵn lẻ (check-parity)

Definition (Dual code)

Dual code (hay **mã đối ngẫu**, **дуальный код**) của linear code \mathcal{C} là code \mathcal{C}^\top được sinh bởi parity-check matrix \mathbf{H} của code \mathcal{C} .

➊ Remark

Dual code với $[n, k]_q$ code là $[n, n - k]_q$ code.

➊ Remark

Với mọi code \mathcal{C} ta có $(\mathcal{C}^\top)^\top = \mathcal{C}$.

➊ Remark

Với mọi $[n]_q$ code \mathcal{C} và \mathcal{B} thì ta có đẳng thức:

$$(\mathcal{C} + \mathcal{B})^\top = \mathcal{C}^\top \cap \mathcal{B}^\top.$$

Trong đó $\mathcal{C} + \mathcal{B} = \{\mathbf{u} + \mathbf{v} : \mathbf{u} \in \mathcal{C}, \mathbf{v} \in \mathcal{B}\}$ là tổng của hai code \mathcal{C} và \mathcal{B} .

➊ Definition

Với các vector $\mathbf{x} = (x_1, \dots, x_n)$ và $\mathbf{y} = (y_1, \dots, y_n)$ trong $V_n(q)$ ta định nghĩa **tích vô hướng** (hay **inner product, dot product, скалярное произведение**) của hai vector là:

$$\langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 + \dots + x_n y_n \in \mathbb{F}_q.$$

➊ Definition

Vector $\mathbf{h} \in V_n(q)$ được gọi là **parity-check** (hay **проверка на чётность**) của $[n]_q$ code nếu với mọi $\mathbf{c} \in \mathcal{C}$ ta có $\langle \mathbf{c}, \mathbf{h} \rangle = 0$, nói cách khác vector \mathbf{h} trực giao với mọi vector $\mathbf{c} \in \mathcal{C}$.

➊ Remark

Dual code \mathcal{C}^\top trùng với tập tất cả parity-check vector của code \mathcal{C} .

Bài toán phổ trọng số của mã tuyến tính**➊ Definition 2.69**

Bài toán về phổ trọng số của linear code (hay **Задача о спектре весов линейного кода**) $WS(\mathbf{H}, t)$ là bài toán nhận dạng trong code, với ma trận parity-check \mathbf{H} , có tồn tại hay không vector $\mathbf{c} \in V_n(q)$ có trọng số Hamming bằng t và thỏa mãn $\mathbf{H}\mathbf{c}^\top = \mathbf{0}$.

➊ Theorem 2.7 (Định lí từ [35])

Bài toán về phỏ trọng số là NP-complete.

Bài toán decode trên linear code

Definition 2.70

Bài toán decode trên linear code $[n, k]_q$ với ma trận sinh \mathbf{G} trong điều kiện kênh truyền có t lỗi là bài toán cho trước vector $\mathbf{y} \in V_n(q)$, ta tìm vector $\mathbf{m} \in V_k(q)$ sao cho vector $\mathbf{e} = \mathbf{y} - \mathbf{m}\mathbf{G}$ có trọng số Hamming bằng t .

Input:

- vector $\mathbf{y} \in V_n(q)$;
- ma trận sinh \mathbf{G} cỡ $k \times n$;
- số tự nhiên $t \in \mathbb{N}$.

Output:

- vector $\mathbf{m} \in V_k(q)$ sao cho vector $\mathbf{e} = \mathbf{y} - \mathbf{m}\mathbf{G}$ có trọng số Hamming bằng t .

Definition 2.71

Bài toán decode trên linear code $[n, k]_q$ với ma trận parity-check \mathbf{H} trong điều kiện kênh truyền có t lỗi là bài toán cho trước vector $\mathbf{y} \in V_n(q)$, ta tìm vector $\mathbf{e} \in V_n(q)$ sao cho $\mathbf{e}\mathbf{H}^\top = \mathbf{y}\mathbf{H}^\top$ và vector \mathbf{e} có trọng số Hamming bằng t , hay $\text{wt}(\mathbf{e}) = t$.

Input:

- vector $\mathbf{y} \in V_n(q)$, ma trận parity-check \mathbf{H} cỡ $(n - k) \times n$ và số tự nhiên $t \in \mathbb{N}$.

Output:

- vector $\mathbf{e} \in V_n(q)$ có trọng số Hamming bằng t , hay $\text{wt}(\mathbf{e}) = t$, sao cho $\mathbf{e}\mathbf{H}^\top = \mathbf{y}\mathbf{H}^\top$.

Từ các bài toán decode linear code liên hệ với bài toán decode syndrome code.

Definition 2.72

Bài toán tối ưu (hay **bài toán tổng quát**) của syndrome decode $[n, k]_q$ với ma trận parity-check \mathbf{H} là bài toán cho trước vector $\mathbf{s} \in V_n(q)$, ta tìm vector $\mathbf{e} \in V_n(q)$ sao cho $\mathbf{H}\mathbf{e}^\top = \mathbf{s}^\top$ và vector \mathbf{e} có trọng số Hamming nhỏ nhất có thể, hay $\text{wt}(\mathbf{e}) \rightarrow \min$.

Input:

- $\mathbf{s} \in V_n(q)$ là syndrome;
- \mathbf{H} là ma trận parity-check cỡ $(n - k) \times n$ của $[n, k]_q$ code \mathcal{C}

Output:

- vector $\mathbf{e} \in V_n(q)$ thỏa mãn $\mathbf{H}\mathbf{e}^\top = \mathbf{s}^\top$ và vector \mathbf{e} có trọng số Hamming nhỏ nhất có thể, hay $\text{wt}(\mathbf{e}) \rightarrow \min$.

❶ Definition 2.73

Bài toán tìm kiếm syndrome decode $sSD(\mathbf{H}, \mathbf{s}, t)$ là bài toán tìm kiếm theo vector $\mathbf{s} \in V_r(q)$ vector $\mathbf{e} \in V_n(q)$ có trọng số Hamming bằng t , hay $\text{wt}(\mathbf{e}) = t$ và $\mathbf{H}\mathbf{e}^\top = \mathbf{s}^\top$.

Input:

- \mathbf{H} là ma trận cỡ $r \times n$ trên \mathbb{F}_q ;
- $\mathbf{s} \in V_r(q)$ là syndrome;
- $t \in \mathbb{N}$ là trọng số Hamming

Output:

- vector $\mathbf{e} \in V_n(q)$ có trọng số Hamming bằng t , hay $\text{wt}(\mathbf{e}) = t$, và $\mathbf{H}\mathbf{e}^\top = \mathbf{s}^\top$.

❶ Definition 2.74

Bài toán nhận dạng syndrome decode (hay **распознавательная задача синдромного декодирования**) hay đơn giản là **bài toán syndrome decode** (hay **задача синдромного декодирования**) $SD(\mathbf{H}, \mathbf{s}, t)$ là bài toán xác định theo syndrome $\mathbf{s} \in V_r(q)$ xem có tồn tại hay không vector $\mathbf{e} \in V_n(q)$ có trọng số Hamming bằng t và thỏa mãn $\mathbf{H}\mathbf{e}^\top = \mathbf{s}^\top$.

Input:

- \mathbf{H} là ma trận cỡ $r \times n$ trên \mathbb{F}_q
- $\mathbf{s} \in V_r(q)$ là syndrome;
- $t \in \mathbb{N}$ là trọng số Hamming.

Output: tồn tại hay không vector $\mathbf{e} \in V_n(q)$ có trọng số Hamming $\text{wt}(\mathbf{e}) = t$ và thỏa $\mathbf{H}\mathbf{e}^\top = \mathbf{s}^\top$.

❶ Theorem 2.8 (Định lí từ [35])

Bài toán nhận dạng syndrome code $SD(\mathbf{H}, \mathbf{s}, t)$ là NP-complete.

Bài toán tương đương các linear code

Hoán vị và tác động của chúng lên tập hợp

❶ Definition 2.75 (Hoán vị)

Xét N là tập hợp có n phần tử. Khi đó một **hoán vị** (hay **permutation, постановка**) $\sigma : N \rightarrow N$ là một song ánh từ N tới N .

Tập hợp tất cả hoán vị trên tập N được kí hiệu là \mathcal{S}_N . Nếu $N = \{1, 2, \dots, n\}$ thì ta có thể viết $\mathcal{S}_N = \mathcal{S}_n$.

Đặt $\sigma \in \mathcal{S}_n$ là một hoán vị trên tập $\{1, \dots, n\}$. Khi đó ta định nghĩa tác động của hoán vị σ lên vector $\mathbf{x} \in V_n(q)$. Giả sử vector $\mathbf{x} = (x_1, \dots, x_n) \in V_n(q)$, ta định nghĩa tác động của hoán vị σ lên vector \mathbf{x} như sau:

$$\mathbf{x}^\sigma = (x_{\sigma(1)}, x_{\sigma(2)}, \dots, x_{\sigma(n)}).$$

Có thể thấy tác động của σ lên $V_n(q)$ là tuyến tính, nghĩa là với mọi $\alpha, \beta \in \mathbb{F}_q$ và với mọi vector $\mathbf{x}, \mathbf{y} \in V_n(q)$ ta có:

$$(\alpha \cdot \mathbf{x} + \beta \cdot \mathbf{y})^\sigma = \alpha \cdot \mathbf{x}^\sigma + \beta \cdot \mathbf{y}^\sigma.$$

Nếu $U \subseteq V_n(q)$ thì kí hiệu U^σ là tác động của hoán vị σ lên mỗi vector trong U , nghĩa là $U^\sigma = \{\mathbf{x}^\sigma : \mathbf{x} \in U\}$.

Do tính tuyến tính của tác động từ \mathcal{S}_n lên $V_n(q)$, nếu \mathcal{C} là $[n, k, d]_q$ code thì \mathcal{C}^σ cũng là $[n, k, d]_q$ code.

Mỗi hoán vị $\sigma \in \mathcal{S}_n$ có thể được viết thành dạng ma trận $n \times n$ là $\mathbf{P}_\sigma = (p_{ij})$. Khi đó phần tử $p_{ij} = 1$ nếu $\sigma(j) = i$, các vị trí còn lại bằng 0.

Example 2.21

Với hoán vị $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 1 & 5 & 3 & 2 \end{pmatrix}$ thì ma trận tương ứng là:

$$\mathbf{P}_\sigma = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

Khi đó tác động của σ lên vector \mathbf{x} là $\mathbf{x}^\sigma = \mathbf{x} \cdot \mathbf{P}_\sigma$.

Như vậy, với mọi vector $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$, dưới tác động của σ bên trên ta có:

$$\mathbf{x}^\sigma = (x_{\sigma(1)}, x_{\sigma(2)}, x_{\sigma(3)}, x_{\sigma(4)}, x_{\sigma(5)}) = (x_4, x_1, x_5, x_3, x_2).$$

Ta cũng thấy phép nhân vector \mathbf{x} với ma trận \mathbf{P}_σ cho kết quả:

$$\mathbf{x} \cdot \mathbf{P}_\sigma = (x_4, x_1, x_5, x_3, x_2).$$

Mã tương đương và nhóm các tự đẳng cấu của linear code

Definition 2.76

Hai $[n]_q$ code \mathcal{C}_1 và \mathcal{C}_2 được gọi là **tương đương** (hay **tương đương hoán vị**, **перестановочно эквивалентные**) nếu tồn tại một hoán vị $\sigma \in \mathcal{S}_n$ sao cho $\mathcal{C}_1 = \mathcal{C}_2^\sigma$.

Definition 2.77

Hai ma trận \mathbf{G}_1 và \mathbf{G}_2 kích thước $k \times n$ gọi là **tương đương hoán vị** nếu tồn tại ma trận \mathbf{M} cỡ $k \times k$ và hoán vị $\sigma \in \mathcal{S}_n$ sao cho $\mathbf{M} \cdot \mathbf{G}_1 = \mathbf{G}_2 \cdot \sigma$.

Nhóm các tự đẳng cấu của code

Definition 2.78

Tự đẳng cấu của $(n)_q$ code \mathcal{C} là hoán vị σ sao cho $\mathcal{C} = \mathcal{C}^\sigma$. Tập hợp tất cả tự đẳng cấu của code \mathcal{C}

được kí hiệu là nhóm $\text{PAut}(\mathcal{C})$ và được gọi là **nhóm tự đẳng cấu hoán vị** hoặc đơn giản hơn là **nhóm tự đẳng cấu** của code \mathcal{C} .

Bài toán về tính tương đương giữa các linear code nhị phân

❶ Definition 2.79

Bài toán về tính tương đương giữa các linear code là bài toán nhận dạng tính chất tương đương giữa hai linear code $[n]_q \mathcal{C}_1$ và \mathcal{C}_2 với ma trận sinh tương ứng là \mathbf{G}_1 và \mathbf{G}_2 .

Input: hai ma trận \mathbf{G}_1 và \mathbf{G}_2 cỡ $k \times n$.

Output: hai ma trận \mathbf{G}_1 và \mathbf{G}_2 có tương đương hoán vị hay không.

Sau đây chúng ta xem xét code nhị phân, tức $q = 2$.

Định lí về polynomial-time reduction của bài toán đẳng cấu đồ thị đến bài toán sự tương đương của code

❶ Definition 2.80

Ma trận kè (hay **матрица инцидентности**) là ma trận nhị phân $\mathbf{D} = (d_{ev})$ cỡ $|E| \times |V|$, với $d_{ev} = 1$ nếu $e = (u, v)$ với v nào đó thuộc V , nghĩa là $d_{ev} = 1$ khi và chỉ khi trên đồ thị có cạnh e xuất phát từ đỉnh v .

Định nghĩa về sự đẳng cấu của hai đồ thị đã được giới thiệu ở *Định nghĩa 29*, ở đây mình giới thiệu lại.

❶ Quan trọng

Đồ thị đẳng cấu

Hai đồ thị (E_1, V_1) và (E_2, V_2) được gọi là **đẳng cấu** (hay **isomorphism**) nếu tồn tại song ánh $\varphi : V_1 \rightarrow V_2$ sao cho với mọi cặp đỉnh $(u, v) \in E_1$ thì cặp $(\varphi(u), \varphi(v)) \in E_2$.

❶ Definition 2.81

Bài toán xác định hai đồ thị, xác định bởi ma trận kè, có đẳng cấu với nhau hay không.

Input: hai ma trận nhị phân \mathbf{D}_1 và \mathbf{D}_2 cỡ $k \times n$.

Output: có tồn tại hay không các hoán vị $\gamma \in \mathcal{S}_k$ và $\sigma \in \mathcal{S}_n$ sao cho $\mathbf{D}_1 = \gamma \cdot \mathbf{D}_2 \cdot \sigma$.

Ý tưởng xây dựng các hệ mật mã dựa trên mã sửa sai

Đặt \mathbf{G} là ma trận sinh của linear code $[n, k]_q \mathcal{C}$ có thể sửa t lỗi.

Nếu code với ma trận sinh \mathbf{G} không có các cấu trúc đại số hoặc tổ hợp thì theo định lí trên về độ khó NP của bài toán nhận dạng syndrome decode, bài toán decode linear code $[n, k]_q$ là rất khó.

Khi đó nếu mã hóa thông điệp m có thể thực hiện qua việc: $\mathbf{m} \rightarrow \mathbf{c} = \mathbf{m}\mathbf{G} + \mathbf{e}$, $\text{wt}(\mathbf{e}) = t$, $\mathbf{e} \in V_n(q)$.

Khi đó, cho trước vector \mathbf{c} , việc tìm một vector \mathbf{m} là bài toán decode trên linear code $[n, k]_q$ với ma trận sinh \mathbf{G} , không có các cấu trúc đại số hoặc tổ hợp. Suy ra việc phá mã rất khó.

Vấn đề là ở phía bên việc tìm lại \mathbf{m} từ \mathbf{c} rất khó nên cần một phương án để giải mã lại thông điệp ban đầu.

Các hệ mật mã khóa công khai dựa trên lý thuyết code

Hệ mật mã McEliece

Hệ mật mã McEliece được phát triển bởi R.J. McEliece vào năm 1978 trong [36].

Ý tưởng xây dựng

Đặt \mathbf{G} là ma trận sinh của linear code $[n, k]_q \mathcal{C}$ nào đó, sửa được t lỗi.

Tham số

- $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \Lambda}$ là một họ các linear code trên trường \mathbb{F}_q sao cho với mỗi code $\mathbf{C} \in \mathcal{C}$ có một thuật toán decode hiệu quả là Decode sửa được t lỗi. Ở đây $\Lambda \subseteq \{0, 1\}^*$ là tập các giá trị của tham số code trong họ.
- Sample(1^λ) là thuật toán xác suất hiệu quả và детерминированный, sao cho với giá trị tham số $\lambda \in \Lambda$ cho ra ma trận sinh \mathbf{G} của code \mathcal{C}_λ , số lượng lỗi t và thuật toán Decode, giải được bài toán decode trên linear code $[n, k]_q$ với ma trận sinh \mathbf{G} và kênh truyền có t lỗi. Kí hiệu $\mathcal{G} = \{\mathbf{G}_\lambda\}_{\lambda \in \Lambda}$ là tập tất cả ma trận sinh cho ra thuật toán Sample(1^λ) cho mọi trường hợp tham số $\lambda \in \Lambda$.

Thuật toán sinh khóa (Gen)

Private key gồm:

1. Ma trận $\mathbf{M} \in \text{GL}_q(k)$ không suy biến cỡ $k \times k$ trên \mathbb{F}_q .
2. $\mathbf{G} \in V_{k \times n}(q)$ là ma trận sinh của $[n, k]_q$ code \mathcal{C}_λ .
3. $\mathbf{P} \in \mathcal{S}_n$ là hoán vị trên tập $\{1, \dots, n\}$.
4. Decode là thuật toán decode trên mã \mathcal{C}_λ .

Public key gồm:

1. $\mathbf{G}_{pub} = \mathbf{M} \cdot \mathbf{G} \cdot \mathbf{P} \in V_{k \times n}(q)$ là ma trận sinh của code tương đương với code \mathcal{C}_λ .
2. $t \in \mathbb{N}$ là số lỗi được decode bởi Decode.

Thuật toán mã hóa (Enc)

Thuật toán mã hóa nhận đầu vào là thông điệp $\mathbf{m} \in V_k(q)$ và trả về ciphertext $\mathbf{c} \in V_n(q)$.

Để mã hóa, chọn ngẫu nhiên vector $\mathbf{e} \in V_n(q)$ độ dài n và có trọng số Hamming bằng t . Khi đó ta tính ciphertext:

$$\mathbf{c} = \mathbf{m} \cdot \mathbf{G}_{pub} + \mathbf{e}.$$

Thuật toán giải mã (Dec)

1. Tính $\mathbf{b} \leftarrow \mathbf{c} \cdot \mathbf{P}^{-1}$.
2. Tính $\mathbf{u} \leftarrow \text{Decode}(\mathbf{b})$.
3. Tính $\mathbf{m} \leftarrow \mathbf{u} \cdot \mathbf{M}^{-1}$.

Khi đó \mathbf{m} là plaintext ban đầu.

Hệ mật mã Niederreiter

Tham số

- $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \Lambda}$ là một họ các linear code trên trường \mathbb{F}_q sao cho với mỗi code $C \in \mathcal{C}$ có một thuật toán decode hiệu quả là Decode sửa được t lỗi. Ở đây $\Lambda \subseteq \{0, 1\}^*$ là tập các giá trị của tham số code trong họ.
- Sample(1^λ) là thuật toán xác suất hiệu quả và детерминированный, sao cho với giá trị tham số $\lambda \in \Lambda$ cho ra ma trận parity-check \mathbf{H} của code \mathcal{C}_λ , số lượng lỗi t và thuật toán SDecode, giải được bài toán syndrome decode trên linear code $[n, k]_q$ với ma trận parity-check \mathbf{H} và kênh truyền có t lỗi. Kí hiệu $\mathcal{H} = \{\mathbf{H}_\lambda\}_{\lambda \in \Lambda}$ là tập tất cả ma trận parity-check cho ra thuật toán Sample(1^λ) cho mọi trường hợp tham số $\lambda \in \Lambda$.

Thuật toán sinh khóa (Gen)

Private key gồm:

1. Ma trận $\mathbf{M} \in \text{GL}_q(r)$ cỡ $r \times r$ trên \mathbb{F}_q , với $r = n - k$ là số kí tự kiểm tra của $[n, k]_q$ code \mathcal{C}_λ .
2. $\mathbf{H} \in V_{r \times n}(q)$ là ma trận parity-check của $[n, k]_q$ code \mathcal{C}_λ .
3. $\mathbf{P} \in \mathcal{S}_n$ là hoán vị trên $\{1, \dots, n\}$.
4. SDecode là thuật toán syndrome decode trên code \mathcal{C}_λ .

Public key gồm:

1. $\mathbf{H}_{pub} = \mathbf{M} \cdot \mathbf{H} \cdot \mathbf{P} \in V_{k \times n}(q)$ là ma trận parity-check của code mới tương đương với code \mathcal{C}_λ .
2. $t \in \mathbb{N}$ là số lượng lỗi có thể sửa bởi SDecode.

Thuật toán mã hóa (Enc)

Plaintext được biểu diễn thành vector $\mathbf{m} \in V_l(q)$ với $l = \lfloor \log_q((q-1)^t \binom{n}{t}) \rfloor$. Ciphertext là vector $\mathbf{c} \in V_r(q)$.

Ta cần ánh xạ $\varphi_{n,t,q} : V_l(q) \rightarrow V_n(q)$ là đơn ánh, với mỗi $\mathbf{m} \in V_l(q)$ cho ra vector $\mathbf{e} \in V_n(q)$ có trọng số Hamming là t .

1. $\mathbf{e} \leftarrow \varphi_{n,t,q}(\mathbf{m})$
2. $\mathbf{c} \leftarrow \mathbf{e} \cdot \mathbf{H}_{pub}^\top$

Thuật toán giải mã (Dec)

Để giải mã ta cần ánh xạ $\varphi_{n,t,q}^{-1}$ là ánh xạ ngược của $\varphi_{n,t,q}$.

1. $\mathbf{s} \leftarrow \mathbf{M}^{-1} \mathbf{c}^\top$.
2. $\mathbf{e} \leftarrow \text{SDecode}(\mathbf{c})$.
3. $\mathbf{e} \leftarrow \mathbf{e} \cdot \mathbf{P}$.
4. $\mathbf{m} \leftarrow \varphi_{n,t,q}^{-1}(\mathbf{e})$.

Goppa Code

Goppa code được sử dụng trong thuật toán Classic McEliece, là một thuật toán mã hóa khóa công khai thuộc post-quantum cryptography. Phần này mình sử dụng slide bài giảng¹.

¹ https://crypto-kantiana.com/elenakirshanova/talks/Talk_McEliece.pdf

Thiết lập Goppa Code

Đầu tiên ta chọn số nguyên $m \geq 1$ và số nguyên tố $q \geq 2$ nhằm xác định trường \mathbb{F}_{q^m} .

Chọn tập

$$L = \{\alpha_1, \dots, \alpha_n\}$$

sao cho $\alpha_i \in \mathbb{F}_{q^m}$ đôi một khác nhau và $n \leq q^m$.

Ta chọn đa thức $g(x)$ bậc không quá t với hệ số trong \mathbb{F}_{q^m} , sao cho $g(x)$ không có nghiệm bội và $g(\alpha_i) \neq 0$ với mọi $i = 1, \dots, n$.

Khi đó Goppa code \mathcal{C} với độ dài n là:

$$\mathcal{C} = \Gamma(L, g) = \left\{ \mathbf{c} = (c_1, \dots, c_n) \in \mathbb{F}_q^n : \sum_{i=1}^n \frac{c_i}{x - \alpha_i} = 0 \pmod{g(x)} \right\},$$

nghĩa là Goppa code \mathcal{C} gồm các vector $\mathbf{c} \in \mathbb{F}_q^n$ sao cho tổng

$$\frac{c_1}{x - \alpha_1} + \frac{c_2}{x - \alpha_2} + \dots + \frac{c_n}{x - \alpha_n} = 0 \pmod{g(x)}.$$

Dễ thấy khi biến đổi tương đương ta có

$$\frac{1}{x - \alpha_i} = -\frac{g(x) - g(\alpha_i)}{x - \alpha_i} \cdot g^{-1}(\alpha_i) \pmod{g(x)}.$$

Trong Classic McEliece thì $q = 2$ và $g(x)$ là đa thức monic và tối giản.

Tìm ma trận parity-check

Giả sử

$$g(x) = g_0 + g_1x + \dots + g_tx^t = \sum_{i=0}^t g_i x^i$$

với $g_i \in \mathbb{F}_{q^m}$.

Ta có

$$\begin{aligned} \frac{g(x) - g(\alpha_i)}{x - \alpha_i} &= \frac{g_t(x^t - \alpha_i^t) + \dots + g_1(x - \alpha_i) + g_0 \cdot 0}{x - \alpha_i} \\ &= g_t(x^{t-1} + x^{t-2}\alpha_i + x^{t-3}\alpha_i^2 + \dots + \alpha_i^{t-1}) \\ &\quad + g_{t-1}(x^{t-2} + x^{t-3}\alpha_i + x^{t-4}\alpha_i^2 + \dots + \alpha_i^{t-2}) \\ &\quad + \dots + g_2(x + \alpha_i) + g_1 \\ &= g_t x^{t-1} + (g_t \alpha_i + g_{t-1}) x^{t-2} \\ &\quad + (g_t \alpha_i^2 + g_{t-1} \alpha_i + g_{t-2}) x^{t-3} \\ &\quad + \dots \\ &\quad + (g_t \alpha_i^{t-1} + g_{t-1} \alpha_i^{t-2} + \dots + g_2 \alpha_i + g_1). \end{aligned}$$

Như vậy, hệ số trước x^j của codeword

$$\sum_{i=1}^n c_i \cdot \frac{g(x) - g(\alpha_i)}{x - \alpha_i} \cdot g^{-1}(\alpha_i)$$

lần lượt là

$$\begin{aligned} x^{t-1} & : g_t g^{-1}(\alpha_1) c_1 + \cdots + g_t g^{-1}(\alpha_n) c_n \\ x^{t-2} & : (g_t \alpha_1 + g_{t-1}) g^{-1}(\alpha_1) c_1 + \cdots + (g_t \alpha_n + g_{t-1}) g^{-1}(\alpha_n) c_n \\ & \vdots \\ x^0 & : (g_t \alpha_1^{t-1} + g_{t-1} \alpha_1^{t-2} + \cdots + g_2 \alpha_1 + g_1) c_1 + (g_t \alpha_2^{t-1} + g_{t-1} \alpha_2^{t-2} + \cdots + g_2 \alpha_2 + g_1) c_2 \\ & \quad + \cdots + (g_t \alpha_n^{t-1} + g_{t-1} \alpha_n^{t-2} + \cdots + g_2 \alpha_n + g_1) c_n \end{aligned}$$

Khi đó, $\mathbf{c} \in \Gamma(L, g)$ khi và chỉ khi tất cả hệ số trước x^j bằng 0. Điều này tương đương với $\bar{\mathbf{H}}\mathbf{c}^\top = 0$ với $\bar{\mathbf{H}} \in \mathbb{F}_{q^m}^{t \times n}$.

Ở đây

$$\begin{aligned} \bar{\mathbf{H}} &= \begin{pmatrix} g_t \cdot g^{-1}(\alpha_1) & \cdots & g_t \cdot g^{-1}(\alpha_n) \\ (g_t \alpha_1 + g_{t-1}) \cdot g^{-1}(\alpha_1) & \cdots & (g_t \alpha_n + g_{t-1}) \cdot g^{-1}(\alpha_n) \\ \vdots & \ddots & \vdots \\ (g_t \alpha_1^{t-1} + g_{t-1} \alpha_1^{t-2} + \cdots + g_2 \alpha_1 + g_1) \cdot g^{-1}(\alpha_1) & \cdots & (g_t \alpha_n^{t-1} + g_{t-1} \alpha_n^{t-2} + \cdots + g_2 \alpha_n + g_1) \cdot g^{-1}(\alpha_n) \end{pmatrix} \\ &= \underbrace{\begin{pmatrix} g_t & 0 & \cdots & 0 \\ g_{t-1} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & \cdots & g_t \end{pmatrix}}_G \cdot \underbrace{\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \cdots & \alpha_n^{t-1} \end{pmatrix}}_X \cdot \underbrace{\begin{pmatrix} g^{-1}(\alpha_1) & 0 & \cdots & 0 \\ 0 & g^{-1}(\alpha_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & g^{-1}(\alpha_n) \end{pmatrix}}_Y. \end{aligned}$$

Ma trận \mathbf{G} khả nghịch, đặt

$$\mathbf{H} = \mathbf{G}^{-1} \bar{\mathbf{H}} = \begin{pmatrix} g^{-1}(\alpha_1) & \cdots & g^{-1}(\alpha_n) \\ \alpha_1 g^{-1}(\alpha_1) & \cdots & \alpha_n g^{-1}(\alpha_n) \\ \vdots & \ddots & \vdots \\ \alpha_1^{t-1} g^{-1}(\alpha_1) & \cdots & \alpha_n^{t-1} g^{-1}(\alpha_n) \end{pmatrix} \in \mathbb{F}_{q^m}^{t \times n}.$$

Ta sẽ thu được ma trận trên $\mathbb{F}_{q^m}^{tm \times n}$ bằng việc xét song ánh $\mathbb{F}_{q^m}^{t \times n} \rightarrow \mathbb{F}_{q^m}^{tm \times n}$ với một cơ sở cố định.

Decode với Goppa code

Khoảng cách tối thiểu (minimal distance) của $\Gamma(L, g)$ là $d \geq t + 1$. Đặc biệt, khi $q = 2$ và g là separable (tức là g có đủ t nghiệm phân biệt trên một trường mở rộng nào đó của \mathbb{F}_{q^m}) thì $d \geq 2t + 1$ (bài tập).

Giả sử ta nhận được vector qua kênh truyền là

$$\mathbf{y} = (y_1, \dots, y_n) = \underbrace{(c_1, \dots, c_n)}_e + \underbrace{(e_1, \dots, e_n)}_e,$$

trong đó e là lỗi. Khi đó $\text{wt}(e) \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ với $\text{wt}(e)$ là trọng số của vector e .

Đặt $\mathcal{B} = \{i : e_i \neq 0\}$ là tập các vị trí xảy ra lỗi. Khi đó $|\mathcal{B}| = \text{wt}(e)$.

Đặt

$$s(x) = \sum_{i=1}^n \frac{y_i}{x - \alpha_i} = \sum_{i=1}^n \frac{e_i}{x - \alpha_i} \bmod g(x)$$

thì đây là syndrome của \mathbf{y} và chúng ta sẽ decode dựa trên $s(x)$.

Ta cần hai đa thức bổ trợ nữa là

$$\sigma(x) = \prod_{i \in \mathcal{B}} (x - \alpha_i)$$

gọi là đa thức định vị lỗi (error locator polynomial), và đa thức

$$w(x) = \prod_{i \in \mathcal{B}} e_i \prod_{j \in \mathcal{B}, j \neq i} (x - \alpha_j).$$

Khi đó ta suy ra

$$e_k = \frac{w(\alpha_k)}{\sigma(\alpha_k)}$$

với mọi $k \in \mathcal{B}$.

Ta cũng suy ra

$$\sigma(x) \cdot s(x) = w(x) \bmod g(x).$$

Khi đó

$$\deg \sigma(x) = |\mathcal{B}|, \quad \deg w(x) = \text{wt}(e) - 1.$$

Lúc này ta có $\deg g = t$ phương trình, trong đó có $2\text{wt}(e) - 1$ phương trình chưa biết.

Vì $\text{wt}(e) < (t - 1)/2$ nên ta có thể giải hệ và tìm lại được các e_k .

3.3.3 Đường cong elliptic

Đường cong elliptic (elliptic curve) rất nổi tiếng trong toán học. Đây là công cụ giúp các nhà toán học giải quyết bài toán lớn **Định lý cuối cùng của Fermat**.

Trong mật mã học, đường cong elliptic là một trong những tiêu chuẩn bảo mật về mã hóa và chữ ký điện tử. Chương này khảo sát những đặc trưng cơ bản đường cong elliptic và ứng dụng trong mật mã học.

Mở đầu về đường cong elliptic

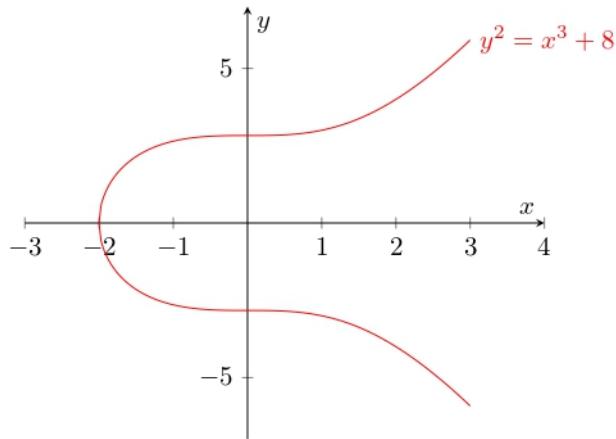
Đường cong elliptic là tập hợp các điểm (x, y) trên mặt phẳng Oxy thỏa mãn phương trình

$$y^2 = x^3 + ax + b,$$

với $a, b \in \mathbb{R}$ và $4a^3 + 27b^2 \neq 0$.

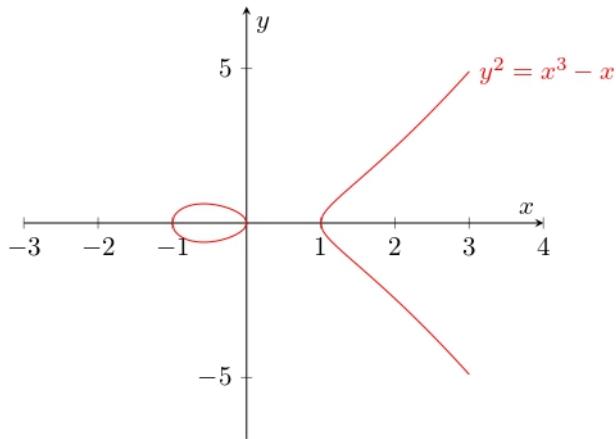
Ví dụ với phương trình $y^2 = x^3 + 8$, đồ thị được biểu diễn ở hình 3.37.

Ta thấy rằng, đường cong elliptic đối xứng qua trục hoành.



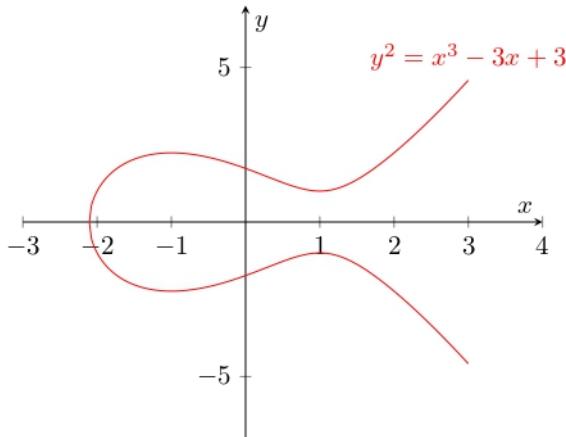
Hình 3.37: Elliptic $y^2 = x^3 + 8$

Ví dụ với phương trình $y^2 = x^3 - x$, đồ thị được biểu diễn ở hình 3.38.



Hình 3.38: Elliptic $y^2 = x^3 - x$

Hoặc đối với phương trình $y^2 = x^3 - 3x + 3$ thì đồ thị được biểu diễn ở hình 3.39.



Hình 3.39: Elliptic $y^2 = x^3 - 3x + 3$

Phép cộng các điểm trên elliptic

Phương trình và đồ thị của đường cong elliptic đã được trình bày ở trên. Tuy nhiên chúng ta quan tâm tới mối liên hệ giữa các điểm trên elliptic, cụ thể là **phép cộng** hai điểm.

Ta thêm một điểm trung điểm vào tập hợp các điểm trên đường cong elliptic và gọi là **điểm vô cực**. Điểm vô cực được kí hiệu là \mathcal{O} . Khi đó mọi điểm P thuộc đường cong elliptic sẽ có tính chất

$$P + \mathcal{O} = \mathcal{O} + P = P.$$

Khi đó, với điểm $P = (x, y)$ bất kì trên elliptic, điểm đối xứng của nó qua trực hoành là $P' = (x, -y)$, và ta định nghĩa $P + P' = \mathcal{O}$.

Tiếp theo ta định nghĩa phép cộng hai điểm.

Giả sử $P = (x_P, y_P)$ và $Q = (x_Q, y_Q)$ là hai điểm trên elliptic. Ta có hai trường hợp:

- Nếu $P \neq Q$, ta vẽ đường thẳng đi qua P và Q . Đường thẳng này cắt elliptic tại điểm thứ ba là S . Ta lấy R đối xứng với S qua trực hoành. Khi đó R cũng nằm trên elliptic và $P + Q = R$;

2. Nếu $P \equiv Q$, ta vẽ tiếp tuyến với elliptic tại điểm P . Tiếp tuyến này cắt elliptic tại điểm thứ hai là S .
Tương tự ta lấy R đối xứng với S qua trục hoành. Khi đó $P + Q = 2P = R$.

Khi đó, tập hợp các điểm trên elliptic cùng với điểm vô cực, và phép cộng hai điểm được định nghĩa như trên tạo thành một nhóm.

Để chứng minh đây là nhóm, ta cần chuyển các khái niệm hình học kia sang đại số để tính toán và chứng minh.

Phép cộng hai điểm khác nhau

Đầu tiên ta thiết lập công thức phép cộng giữa hai điểm cho trường hợp $P \neq Q$. Giả sử $P = (x_P, y_P)$ và $Q = (x_Q, y_Q)$.

Phương trình đường thẳng đi qua P và Q là

$$y = \frac{y_Q - y_P}{x_Q - x_P}(x - x_P) + y_P. \quad (3.22)$$

Thay y vào phương trình đường cong elliptic ta có

$$\left[\frac{y_Q - y_P}{x_Q - x_P}(x - x_P) + y_P \right]^2 = x^3 + ax + b \quad (3.23)$$

Đặt $k = \frac{y_Q - y_P}{x_Q - x_P}$. Khi đó phương trình tương đương với

$$(kx - kx_P + y_P)^2 = x^3 + ax + b.$$

Khai triển và chuyển vế ta có

$$x^3 - k^2x^2 + \dots = 0.$$

Ta chỉ cần quan tâm hệ số trước x^2 . Bởi vì ta biết rằng đường thẳng PQ cắt elliptic tại ba điểm P, Q, S , nên phương trình bậc 3 này có 3 nghiệm phân biệt là x_P, x_Q và x_S . Theo định lý Viete ta có

$$x_P + x_Q + x_S = k^2.$$

Như vậy ta có hoành độ điểm S

$$x_S = k^2 - x_P - x_Q,$$

thay x_S vào (3.22), ta có tung độ điểm S

$$y_S = k(x_S - x_P) + y_P,$$

mà R đối xứng với S qua trục hoành, như vậy $x_R = x_S$ và $y_R = -y_S$.

Như vậy kết quả của phép cộng là

$$\begin{aligned} x_R &= k^2 - x_P - x_Q \\ y_R &= k(x_P - x_R) - y_P \end{aligned}$$

với $k = \frac{y_Q - y_P}{x_Q - x_P}$.

Phép cộng hai điểm giống nhau

Trong trường hợp hai điểm giống nhau, ta vẽ tiếp tuyến tiếp xúc với elliptic đi qua điểm đó.

Giả sử ta muốn vẽ tiếp tuyến tại điểm $P = (x_P, y_P)$, khi đó từ phương trình elliptic $y^2 = x^3 + ax + b$ ta vi phân hai vê thu được

$$2y \, dy = (3x^2 + a) \, dx.$$

Ta biết rằng hệ số góc của đường tiếp tuyến là đạo hàm hàm số tại điểm đó, hay nói cách khác là dy/dx . Như vậy hệ số góc tiếp tuyến tại điểm P là

$$k = \frac{dy}{dx} = \frac{3x_P^2 + a}{2y_P}$$

và như vậy phương trình đường tiếp tuyến là

$$y = k(x - x_P) + y_P.$$

Thực hiện tương tự như bên trên, ta có đường tiếp tuyến cắt elliptic tại hai điểm phân biệt, trong đó có một điểm tiếp xúc nên trong phương trình hoành độ giao điểm điểm tiếp xúc là nghiệm bội hai. Nói cách khác, theo định lý Viete thì

$$x_P + x_S + x_R = k^2,$$

suy ra hoành độ điểm S là

$$x_S = k^2 - 2x_P,$$

và tung độ điểm S là

$$y_S = k(x_S - x_P) + y_P$$

Cuối cùng, tọa độ điểm $R = P + P$ là

$$x_R = k^2 - 2x_P, \quad y_R = k(x_P - x_S) - y_P.$$

Tổng kết

Để cộng hai điểm $P = (x_P, y_P)$ và $Q = (x_Q, y_Q)$ ta có ba trường hợp sau:

1. Nếu $x_Q = x_P$ và $y_Q = -y_P$, nói cách khác là đối xứng qua trục hoành, thì ta có $P + Q = \mathcal{O}$.

2. Nếu $x_P \neq x_Q$, đặt $k = \frac{y_Q - y_P}{x_Q - x_P}$ thì tọa độ điểm $R = P + Q$ là

$$x_R = k^2 - x_P - x_Q, \quad y_R = k(x_P - x_R) - y_P.$$

3. Nếu $x_P = x_Q$ và $y_P = y_Q$, khi hai điểm trùng nhau, đặt $k = \frac{3x_P^2 + a}{2y_P}$, thì tọa độ điểm $R = 2P$ là

$$x_R = k^2 - 2x_P, \quad y_R = k(x_P - x_R) - y_P.$$

Đường cong elliptic trên trường hữu hạn \mathbb{F}_p

Trong mật mã học đường cong elliptic được sử dụng nhiều nhất là trên trường hữu hạn \mathbb{F}_p với p là số nguyên tố.

Đường cong elliptic lúc này là điểm vô cực \mathcal{O} và tập hợp các điểm $(x, y) \in \mathbb{F}_p^2$ thỏa mãn

$$y^2 \equiv x^3 + ax + b \pmod{p},$$

với $4a^3 + 27b^2 \neq 0 \pmod{p}$.

Việc thực hiện phép cộng hai điểm cũng tương tự như trên \mathbb{R} ở phần trên nhưng các phép tính được thực hiện trong modulo p .

Như vậy để cộng hai điểm $P = (x_P, y_P)$ và $Q = (x_Q, y_Q)$ thì:

1. Nếu $x_Q \equiv x_P \pmod{p}$ và $y_Q \equiv -y_P \pmod{p}$ thì $P + Q = \mathcal{O}$.
2. Nếu $x_Q \not\equiv x_P \pmod{p}$, đặt $k \equiv \frac{y_Q - y_P}{x_Q - x_P} \pmod{p}$ thì tọa độ điểm $R = P + Q$ là

$$x_R \equiv k^2 - x_P - x_Q \pmod{p}, \quad y_R = k(x_P - x_R) - y_P \pmod{p}.$$
3. Nếu $x_Q \equiv x_P$ và $y_Q \equiv y_P$, nói cách khác là hai điểm trùng nhau, đặt $k = \frac{3x_P^2 + a}{2y_P} \pmod{p}$ thì tọa độ điểm $R = 2P$ là

$$x_R = k^2 - 2x_P \pmod{p}, \quad y_R = k(x_P - x_R) - y_P \pmod{p}.$$

Một điểm đáng chú ý là \mathbb{F}_p có hữu hạn phần tử, do đó số phần tử của đường cong elliptic với tọa độ trong \mathbb{F}_p cũng là hữu hạn. Do tính chất này mà mật mã học có thể sử dụng đường cong elliptic.

Dạng Weierstrass tổng quát

Ở công thức cộng hai điểm giống nhau xuất hiện hai hằng số là 2 và 3. Một vấn đề đặt ra là nếu trường có đặc số bằng 2, nghĩa là $2 \equiv 0$, khi đó hàm hai biến

$$f(x, y) = y^2 - x^3 - ax - b$$

sẽ có đạo hàm theo y luôn bằng 0 vì $f_y = 2y \equiv 0$. Ta không thể tính tiếp tuyến như trên \mathbb{R} để đưa ra công thức cho phép cộng hai điểm giống nhau. Tương tự đối với trường có đặc số bằng 3.

Trong trường hợp các trường K có đặc số bằng 2 thì chúng ta sử dụng dạng Weierstrass tổng quát (generalized Weierstrass) là

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

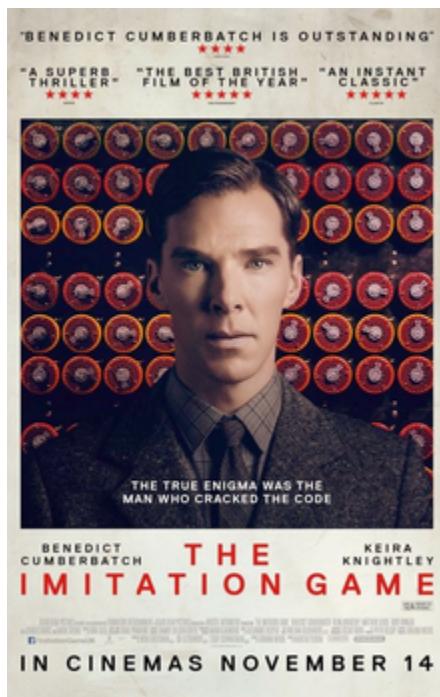
với $a_1, a_2, a_3, a_4, a_6 \in K$.

[TODO] Phép cộng hai điểm trên đường cong elliptic dạng Weierstrass tổng quát.

3.3.4 Nguyên tắc Kerckhoffs và mật mã học

Khi mình kể với mọi người rằng mình đang học mật mã thì mình nhận ra một điều thú vị là rất nhiều người hiểu lầm về mật mã. Trong quan điểm của đa số thì mật mã là một thứ gì đó cực kì bí mật và chỉ sử dụng trong quân sự. Điều này đúng từ nửa đầu thế kỷ 20 trở về trước, nhưng hiện tại thì mật mã được sử dụng rộng rãi trong dân sự, thậm chí là bắt buộc để đảm bảo an toàn thông tin trên Internet. Bài viết này sẽ giải thích một số khía cạnh của mật mã được sử dụng trong việc truyền thông tin.

Các bạn có thể xem phim "The Imitation Game" (năm 2014) về đội giải mã Enigma và nhà toán học Alan Turing để có cái nhìn tổng quan hơn những điều mình sắp nói.



Hình 3.40: Nguồn: Wikipedia

Nguyên lý Kerckhoffs được phát biểu như sau:

Độ an toàn của hệ thống mật mã không phụ thuộc vào việc giữ bí mật thuật toán mã hóa mà phụ thuộc vào việc giữ bí mật khóa.

Các bạn cũng có thể đọc hai bài viết sau:

- Nguyên lý Kerckhoffs - công trình khoa học mật mã uyên bác từ thế kỷ XIX.
- Mật mã hiện đại (1).

Vấn đề truyền tin trên kênh mở

Thông tin có thể được truyền đi dưới nhiều dạng:

- tiếng nói;
- sóng điện từ (sóng radio);
- tín hiệu điện tử (qua cáp đồng);
- tín hiệu ánh sáng (qua cáp quang);
- vân vân và mây mây.

Thông thường chúng ta sử dụng các loại sóng vô tuyến, cáp quang để truyền tín hiệu đi xa. Vấn đề là các tín hiệu đó có thể bị bắt (chặn) lại bằng nhiều dụng cụ vật lý.

Trong phim "The Imitation Game", bộ phận thu sóng của quân Đồng minh chặn được hàng tá thông điệp của quân Phát xít gửi bằng radio mỗi ngày (từ bộ chỉ huy gửi tới đơn vị tác chiến) bằng các thiết bị chuyên dụng. Như vậy các kênh truyền trên trở thành các kênh mở, không chỉ người gửi và người nhận có thể biết thông điệp (tín hiệu) mà những bên có các thiết bị chuyên dụng cũng có thể đọc được.

Mã Caesar - nguồn gốc mật mã học

Thời xa xưa, cụ thể là thế kỉ 3 Sau Công Nguyên, hệ mã hóa đầu tiên là Caesar ra đời nhằm phục vụ chiến tranh. Các chỉ thị từ bộ chỉ huy được gửi tới các trạm, nhưng nếu gửi văn bản bình thường sẽ rất nguy hiểm nếu người đưa tin bị địch tóm. Như vậy mật mã sẽ giải quyết vấn đề này.

Mật mã gồm hai quá trình ngược nhau là **mã hóa** (hay **encrypt**) và **giải mã** (hay **decrypt**). Câu chuyện về mật mã Caesar như sau.

Giả sử ông Caesar muốn gửi thông điệp "It is rainy today" (thông tin về thời tiết). Ông Caesar sẽ không viết thông điệp này lên tờ giấy, xếp gọn lại, rồi đưa cho anh shipper gửi tới tiền tuyến. Thay vào đó, ông ấy viết từng ký tự trong thông điệp bởi chữ cái đứng sau nó ba ký tự trong bảng chữ cái. Nghĩa là "I" thành "L", "t" thành "w", vân vân và mây mây. Như vậy thông điệp sau khi biến đổi là "Lw lv udlqb wrgdb". Đây là quá trình **mã hóa**, biến đổi thông điệp gốc thành thông điệp mới khác ban đầu (và đa phần không có ý nghĩa).

Ghi chú

Thông điệp ban đầu được gọi là **bản rõ** (hay **plaintext**).

Thông điệp đã bị mã hóa được gọi là **bản mã** (hay **ciphertext**).

Ở ví dụ trên:

- bản rõ "It is rainy today";
- bản mã là "Lw lv udlqb wrgdb".

Sau đó anh shipper mang tờ giấy có thông điệp này tới tiền tuyến. Các chú lính cát ở tiền tuyến sẽ làm công việc ngược lại, thay từng ký tự ở thông điệp mới bởi ký tự trước đó ba vị trí trong bảng chữ cái. Nghĩa là "L" thành "I", "w" thành "t", và tương tự vậy. Đây là quá trình **giải mã**, biến đổi thông điệp mới quay về thông điệp gốc ban đầu.

Công việc mã hóa và giải mã sử dụng một quy tắc chung cho tất cả ký tự của thông điệp, do đó chúng ta cần một công thức toán học biểu thị quá trình này.

Đầu tiên ta đánh số các ký tự của bảng chữ cái tiếng Anh bắt đầu từ 0.

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Giả sử bản rõ là dãy các ký tự p_1, p_2, \dots, p_n với p_i là các ký tự thuộc bảng chữ cái tiếng Anh. Như vậy $0 \leq p_i \leq 25$.

Nếu gọi bản mã là dãy các ký tự tương ứng c_1, c_2, \dots, c_n thì việc dịch chuyển ba ký tự sang trái tương đương với công thức

$$c_i = (p_i + 3) \bmod 26.$$

Ở đây modulo 26 giúp việc tính toán không vượt khỏi bảng chữ cái, mang ý nghĩa là nếu đi tới cuối bảng chữ cái thì ta bắt đầu lại từ đầu (ví dụ "Y" sẽ thành "B").

Thay vì sử dụng số 3, chúng ta có thể sử dụng một số bất kì khác và giữ bí mật số này, gọi là **khóa bí mật**. Công thức chung lúc này sẽ là

$$c_i = (p_i + k) \bmod 26,$$

với k là khóa bí mật.

Reverse Engineering và mật mã học

Nếu chỉ có người gửi và người nhận biết nguyên lý mã hóa và khóa bí mật thì gần như không thể phục hồi bản rõ. Tuy nhiên nếu kẻ địch biết nguyên lý mã hóa thì sao? Với mã Caesar ở trên, nếu đã biết $c_i = (p_i + k) \bmod 26$ thì rõ ràng chúng ta có thể thử giải mã với từng khóa $k = 1, 2, \dots, 25$ và xem thông điệp gốc nào có ý nghĩa.

Chúng ta thử giải mã thông điệp "Jr jvyy zrrg ng fgngvba". Với từng khóa k ta giải mã theo công thức ngược lại là

$$p_i = (c_i - k) \bmod 26$$

và nhận được bản rõ tương ứng theo bảng sau:

Khóa k	Bản rõ tương ứng
1	Iq iuxx yqqf mf efmfuaz
2	Hp htww xppe le deletzy
3	Go gsvv wood kd cdkdsyx
4	Fn fruu vnnc jc bcjcrxw
5	Em eqtt ummb ib abibqwv
6	Dl dpss tlla ha zahapvu
7	Ck corr skkz gz yzgzout
8	Bj bnqq rjyf fy xyfynts
9	Ai ampp qiiex wxexmsr
10	Zh zloo phhw dw vwdwlrq
11	Yg yknn oggv cv uvcvkqp
12	Xf xjmm nffu bu tubujpo
13	We will meet at station
14	Vd vhkk ldds zs rszshnm
15	Uc ugjj kccr yr qryrgml
16	Tb tfii jbbq xq pqxqflk
17	Sa sehh iaap wp opwpekj
18	Rz rdgg hzzo vo novodji
19	Qy qcif gyyn un mnuncih
20	Px pbee fxxm tm lmtmbhg
21	Ow oadd ewwl sl klslagf
22	Nv nzcc dvvk rk jkrkzfe
23	Mu mybb cuuj qj ijqjyed
24	Lt lxaa btti pi hipixdc
25	Ks kwzz assz oh ghohwcb

Ta thấy rằng $k = 13$ cho bản rõ là một thông điệp có ý nghĩa. Như vậy mã Caesar không an toàn trước phương pháp thử tất cả khóa có thể (vét cạn, bruteforce).

Vấn đề ở đây là khi đối thủ cũng biết nguyên lý mã hóa thì đối thủ có thể đưa ra phương pháp tấn công thuật toán mật mã. Khi thời đại phát triển với sự ra đời các máy cơ học, máy tính (phần cứng, phần mềm) thì các thuật toán mật mã được "giáu" kỹ bằng các kỹ thuật gây rối khác nhau. Tuy nhiên sau một thời gian thì các bên tấn công cũng hiểu được toàn bộ cơ chế hoạt động của phần mềm, phần cứng và sau đó là thuật toán. Quy trình tìm hiểu cơ chế hoạt động của một thiết bị, thuật toán nào đó là **reverse engineering** (dịch ngược). Khi đã biết cơ chế mã hóa thì việc phá mã trở nên dễ dàng, như mã RC4 được sử dụng để mã hóa tín hiệu WiFi.

Lúc này, mật mã học đi theo một cách tiếp cận khác. Nếu mã hóa là một quy trình với hai đầu vào là bản rõ p và khóa k , đầu ra là bản mã c , thì quy trình này có thể được viết dưới dạng hàm số

$$c = \text{Enc}(p, k)$$

với Enc là thuật toán mã hóa. Lúc này, thuật toán Enc không cần thiết được giữ bí mật, mà chỉ cần giữ bí mật khóa k . Như chúng ta đã thấy ở mã Caesar, từ bản mã ban đầu c , với mỗi khóa k chúng ta lại có một bản rõ p tương ứng. Vậy bản rõ p tương ứng khóa k nào mới là thông điệp đúng, mang ý nghĩa?

Nguyên lý Kerckhoffs và mật mã hiện đại

Từ phần trên chúng ta có thể rút ra một số yêu cầu chính đối với mật mã hiện đại.

Yêu cầu thứ nhất đối với mật mã an toàn là **tính không bí mật** của thuật toán mã hóa. Trong phim "The Imitation Game", quân Đồng Minh biết rõ quân Đức sử dụng máy Enigma để mã hóa các thông điệp gửi từ bộ chỉ huy tới đơn vị tác chiến. Họ (quân Đồng minh) thậm chí còn "thó" được một máy Enigma và mở xé nó, biết rõ cách hoạt động của nó. Nhưng thiết lập của máy Enigma thay đổi mỗi ngày, đó chính là khóa bí mật. Dù biết cách máy Enigma hoạt động, nhưng không biết thiết lập (không biết khóa) thì cũng không thể giải mã được các thông điệp được gửi đi. Như mình đã nói, quân Đồng minh dư sức đọc các tín hiệu điện tử được gửi đi bởi quân Đức, nhưng những gì họ nhận được chỉ là những văn bản vô nghĩa, chính là bản mã. Enigma trở thành pháo đài vững chãi không thể bị phá thời đó.

Ngày nay, các thuật toán mật mã được sử dụng trên Internet được định nghĩa đầy đủ trong các tiêu chuẩn quốc gia, quốc tế (NIST, GOST, RFC, ...). Các tài liệu này tất nhiên là được công bố rộng rãi và được nhiều nhà nghiên cứu phân tích, đánh giá độ an toàn của các thuật toán mật mã. Ngoài ra, các giao thức mạng về mật mã (SSL/TLS) cũng chỉ rõ thuật toán mã hóa nào được sử dụng, yêu cầu với các tham số đầu vào, vân vân.

Trên đây chính là nội dung của nguyên lý Kerckhoffs. Ông viết nguyên lý này trong một tài liệu tên là "Mật mã trong quân sự" (tên gốc là "La Cryptographie Militaire"). Tuy nhiên ý nghĩa của nó không chỉ dừng lại ở quân sự mà còn cả dân sự, là môi trường Internet chúng ta đang sử dụng hằng ngày.

Yêu cầu thứ hai đối với mật mã an toàn là **không gian khóa phải đủ lớn**. Ở mã Caesar, không gian khóa có 25 phần tử (vì $k = 0$ cho bản mã y hệt bản rõ nên không có ý nghĩa che giấu thông tin) nên rất dễ phá, thậm chí có thể thử bằng giấy và bút. Ngày nay các thuật toán mã hóa khối có không gian khóa rất lớn với các khóa có độ dài 128 bit hoặc 256 bit. Khi đó có 2^{128} trường hợp khóa nếu khóa có độ dài 128 bit, cụ thể thì

$$2^{128} = 340282366920938463463374607431768211456,$$

một con số khổng lồ nếu chúng ta thử từng trường hợp, kể cả có sử dụng các máy tính mạnh nhất hiện tại. Ví dụ như thuật toán AES với trường hợp khóa 256 bit được đánh giá là an toàn nhất hiện nay và sẽ còn an toàn trong nhiều năm tới.

Trên đây là hai yêu cầu đơn giản đối với mật mã an toàn hiện nay. Theo sự phát triển của khoa học và công nghệ thì nhiều phương pháp phá mã phức tạp (về mặt toán học) đã ra đời nên cũng có nhiều tiêu chí khác đánh giá độ an toàn của mật mã. Tuy nhiên các tiêu chí kia rất phức tạp nên mình sẽ không đề cập ở đây.

Cám ơn các bạn đã đọc bài viết của mình.

Moscow, ngày 23 tháng 1 năm 2025.

3.3.5 Câu hỏi ôn thi mật mã học

Môn "Các phương pháp bảo vệ thông tin bằng mật mã"

Задачи криптографической защиты информации и средства их решения

В криптографических исследованиях разрабатываются средства и методы решения следующих задач:

1. **Конфиденциальность.** Защита от ознакомления с содержанием информации (сообщения) лицами, не обладающими правом доступа. При этом:
 - не скрывается сам факт передачи сообщения;
 - зашифрованное сообщение передается по открытому каналу связи.
2. **Целостность.** Обеспечение невозможности несанкционированного изменения исходной информации.
3. **Аутентификации.** Доказательное подтверждение подлинности сторон и передаваемой информации в процессе информационного взаимодействия.
4. **Невозможность отказа от авторства.** Разработка методов предотвращения возможности отказа от ранее совершенных действий.

Симметричные, асимметричные и комбинированные криптосистемы

Симметричные криптосистемы - данные криптосистемы построены на основе сохранения в тайне ключа шифрования. Процессы зашифрования и расшифрования используют **один и тот же ключ**. Секретность ключа является постулатом. **Основная проблема** при применении симметричных криптосистем для связи заключается в сложности **передачи** обоим сторонам **секретного ключа**.

Однако даны раскрытие ключа злоумышленником грозит раскрытием только той информации, что была зашифрована на этом ключе. Эти системы обладают высоким быстродействием.

Асимметричные криптосистемы - смысл данных криптосистем состоит в том, что для зашифрования и расшифрования используются разные преобразования. Одно из них - зашифрование - является абсолютно открытым для всех. Другое же - расшифрование - остается секретным. Таким образом, любой, кто хочет что-либо зашифровать, пользуется открытым преобразованием. Но расшифровать и прочитать это сможет лишь тот, кто владеет секретным преобразованием.

Комбинированное - совместное использование этих криптосистем позволяет эффективно реализовывать такую базовую функцию защиты, как криптографическое закрытие передаваемой информации с целью обеспечения ее конфиденциальности.

Комбинированное применение симметричного и асимметричного шифрования устраняет основные недостатки, присущие обоим методам, и позволяет сочетать преимущества высокой секретности, предоставляемые асимметричными криптосистемами с открытым ключом, с преимуществами высокой скорости работы, присущими симметричным криптосистемам с секретным ключом.

Метод комбинированного использования симметричного и асимметричного шифрования заключается в следующем.

Симметричную криптосистему применяют для шифрования исходного открытого текста, а асимметричную криптосистему с открытым ключом применяют только для шифрования секретного ключа симметричной криптосистемы. В результате асимметричная криптосистема с открытым ключом не заменяет, а лишь дополняет симметричную криптосистему с секретным ключом, позволяя повысить в целом защищенность передаваемой информации. Такой подход иногда называют схемой электронного "цифрового конверта".

Шифры, алгебраическая модель шифра, примеры

Пусть:

1. \mathcal{X} -- пространство открытых текстов;
2. \mathcal{Y} -- пространство шифротекстов;
3. \mathcal{K} -- пространство ключей;

4. функцию зашифрования $y = E(x, k)$, переводящую открытый текст $x \in \mathcal{X}$ на ключе шифрования $k \in \mathcal{K}$ в шифротекст $y \in \mathcal{Y}$;
5. функцию расшифрования $x = D(y, k)$.

Тройка множеств, $\mathcal{X}, \mathcal{K}, \mathcal{Y}$ с введенными функциями называется шифром по Шеннону, если каждый шифротекст есть результат шифрования одного из открытых текстов(т.е. функция $y = E(x, k)$ сюръективна), а разным открытым текстам отвечают разные шифротексты(т.е. функция $y = E(x, k)$ инъективна). Алгебраическая модель предложена Шенном.

Шифры, вероятностная модель шифра, примеры

Вероятностной моделью шифра называется его алгебраическая модель, дополненная известными независимыми распределениями вероятностей $P(\mathcal{X})$ и $P(\mathcal{K})$.

Вероятность появления шифротекста y равна:

$$P(y) = \sum_{E(x,k)=y} p(x) \cdot p(k)$$

В тех случаях, когда требуется знание распределений $P(\mathcal{X})$ и $P(\mathcal{K})$, мы будем пользоваться вероятностной моделью Σ_B , состоящей из пяти множеств, связанных условиями 1 и 2 предыдущего определения алгебраической модели шифра, и двух вероятностных распределений:

$$\Sigma_B = (\mathcal{X}, \mathcal{K}, \mathcal{Y}, E, D, P(\mathcal{X}), P(\mathcal{K}))$$

Модели и критерии распознавания открытых текстов

Криптоанализ классических шифров. Дешифрование шифра Виженера

Классификации шифров

По особенностям алгоритмы шифрования

1. Симметричные
2. Асимметричные
3. Комбинированные

По количеству символов сообщения шифруемых или расшифровываемых по однотипной процедуре преобразования

1. Потоковые
2. Блочные

Теоретическая и практическая стойкость шифров

Совершенные шифры

Пусть $\Sigma_B = (\mathcal{X}, \mathcal{Y}, \mathcal{K}, E, D, P(\mathcal{X}), P(\mathcal{Y}))$ - вероятностная модель шифра. Шифр Σ_B называется **совершенным**, если для любого x из \mathcal{X} и любого y из \mathcal{Y} выполняется равенство

$$p(x) = p(x|y)$$

Назовем шифр Σ_B совершенным, если для любых $x \in \mathcal{X}$, $y \in \mathcal{Y}$ выполняется равенство $p(x|y) = p(x)$.

Шифры замены и их криптоанализ

Шифр простой замены — класс методов шифрования, которые сводятся к созданию по определённому алгоритму таблицы шифрования, в которой для каждой буквы открытого текста существует единственная сопоставленная ей буква шифр-текста.

Шифры перестановки и их криптоанализ

Шифр перестановки — это метод симметричного шифрования, в котором элементы исходного открытого текста меняют местами.

Шифрование методом гаммирования и его криптоанализ

Это метод симметричного шифрования, заключающийся в «наложении» последовательности, состоящей из случайных чисел, на открытый текст. Последовательность случайных чисел называется гаммапоследовательностью и используется для зашифровывания и расшифровывания данных. Суммирование обычно выполняется в каком-либо конечном поле.

Например, в поле Галуа GF(2) суммирование принимает вид операции «исключающее ИЛИ (XOR)». Взлом шифра невозможен т.к. по Шеннону он является совершенным.

Криптоаналитические атаки и их классификация

Криптоаналитическая атака при наличии только известного шифртекста (known-plaintext). Криптоаналитик имеет только шифртексты C_1, C_2, \dots, C_i нескольких сообщений, причем все они зашифрованы с использованием одного и того же алгоритма шифрования E . Работа криптоаналитика заключается в том, чтобы раскрыть исходные тексты M_1, M_2, \dots, M_i .

Криптоаналитическая атака при наличии известного открытого текста. Криптоаналитик имеет доступ не только к шифртекстам C_1, C_2, \dots, C_i и нескольких сообщений, но также к открытым текстам M_1, M_2, \dots, M_i этих сообщений. Его работа заключается в нахождении ключа k , используемого при шифровании этих сообщений, или алгоритма расшифрования любых новых сообщений, зашифрованных тем же ключом.

Криптоаналитическая атака при возможности выбора открытого текста (chosen-plaintext). Крипто-аналитик не только имеет доступ к шифртекстам C_1, C_2, \dots, C_i и связанным с ними открытым текстам M_1, M_2, \dots, M_i этих сообщений, но и может по желанию выбирать открытые тексты, которые затем получает в зашифрованном виде.

Криптоаналитическая атака методом полного перебора всех возможных ключей (bruteforce).

Блочные шифры. Принципы построения симметричных блочных шифров

Блочный шифр — разновидность симметричного шифра, оперирующего группами бит фиксированной длины — блоками.

Если исходный текст (или его остаток) меньше размера блока, перед шифрованием его дополняют. Фактически, блочный шифр представляет собой подстановку на алфавите блоков, которая, как следствие, может быть моно- или полиалфавитной.

Принципы построения:

1. SP-сети: AES, Кузнецик
2. Сеть Фейстеля: DES, Магма

Режимы работы блочных шифров и их сравнение

В ГОСТ 28147—89 этот режим называется режимом простой замены.

Ghi chú

Простая замена (thay thế đơn giản) là cách gọi mode ECB trong các tài liệu tiếng Nga.

Сообщение делится на блоки одинакового размера.

Размер (длина) блока равен n и измеряется в битах. В результате получается последовательность блоков P_1, P_2, \dots, P_m .

Последний блок при необходимости дополняется до длины n . Каждый блок P_i шифруется алгоритмом шифрования E_k с использованием ключа k :

$$C_i = E_k(P_i, k).$$

Методы анализа алгоритмов блочного шифрования

1. Атака полным перебором
2. Дифференциальный криптоанализ
3. Линейный криптоанализ (поиск аффинных приближений)

Стандарт шифрования данных DES

DES (англ. Data Encryption Standard) — алгоритм для симметричного шифрования, утвержденный правительством США в 1977 году как официальный стандарт (FIPS 46-3).

Размер блока для DES равен 64 битам. В основе алгоритма лежит сеть Фейстеля с 16 циклами (раундами) и ключом, имеющим длину 56 бит. Алгоритм использует комбинацию нелинейных (S-блоки) и линейных преобразований.

Развертывание раундовых ключей в DES

Ключи k_i получаются из начального ключа k (56 бит) следующим образом.

Добавляются биты в позиции 8, 16, 24, 32, 40, 48, 56, 64 ключа k таким образом, чтобы каждый байт содержал нечетное число единиц. Это используется для обнаружения ошибок при обмене и хранении ключей.

Затем делают перестановку для расширенного ключа (кроме добавляемых битов 8, 16, 24, 32, 40, 48, 56, 64)

Эта перестановка определяется двумя блоками C_0 и D_0 по 28 бит каждый. Первые 3 бита C_0 есть биты 57, 49, 41 расширенного ключа. А первые три бита D_0 есть биты 63, 55, 47 расширенного ключа $C_i, D_i, i = 1, 2, 3, \dots$ получаются из C_{i-1}, D_{i-1} одним или двумя левыми циклическими сдвигами.

Ключ $k_i, i = 1, \dots, 16$ состоит из 48 бит, выбранных из битов вектора $C_i \| D_i$ (56 бит) согласно таблице. Пример, первый и второй биты k_i есть биты 14, 17 вектора $C_i \| D_i$.

Режим сцепления блоков шифра (CBC) на примере DES

Режим обратной связи по выходу (OFB) на примере DES

Режим сцепления блоков шифра (CBC) на примере DES

Имитостойкость шифров

Имитостойкость шифра определим как его способность противостоять попыткам противника по имитации или подмене.

Имитозащита — защита системы шифровальной связи или другой крипtosистемы от навязывания ложных данных. Защита данных от внесения в них несанкционированных изменений, другими словами, защита целостности сообщения.

Реализуется с помощью добавления к сообщению дополнительного кода (добавление битов четности, контрольной суммы), имитовставки, зависящей от содержания сообщения и секретного элемента, известного только отправителю и получателю (ключа).

Закладка избыточности позволяет обнаружить внесённые в сообщение несанкционированные изменения.

Стандарт шифрования данных AES

Xem bài viết về AES.

Развертывание раундовых ключей в AES

Xem bài viết về AES.

Российский стандарт шифрования данных МАГМА (ГОСТ Р 34.12-2015)

Xem bài viết về Magma.

Развертывание раундовых ключей в стандарте МАГМА, количество слабых и 2-слабых ключей

Российский стандарт шифрования данных КУЗНЕЧИК (ГОСТ Р 34.12-2015)

Xem bài viết về Kuznyechik.

Развертывание раундовых ключей в стандарте КУЗНЕЧИК

Xem bài viết về Kuznyechik.

Поточные шифры. Принципы их построения

Методы генерации и анализа псевдослучайных последовательностей

Регистры сдвига, критерий регулярности

Регистры сдвига максимального периода

Криптоанализ поточных шифров

Системы шифрования с открытыми ключами. Принципы их построения

Анализ асимметричных криптосистем

Атаки на асимметричные криптосистемы

Системы шифрования с открытыми ключами Криптосистема RSA

Системы шифрования с открытыми ключами. Криптосистема Эль-Гамаля

Управление ключами. Открытое распределение ключей Диффи - Хеллмана

Электронная подпись. Принципы ее формирования

Электронная подпись на базе криптосистемы RSA

Электронная подпись на базе криптосистемы Эль Гамаля

Российский стандарт электронной подписи ГОСТ Р 34.10-2012

Хэш-функции, требования к ним

Методы построения функций хэширования

Российский стандарт хэш-функции ГОСТ Р 34.11-2012

Криптографические протоколы и их классификация

Системы аутентификации

Алгоритмы «облегченной» (lightweight) криптографии и их предназначение

Криптографические средства защиты информации в ОС Windows

Криптографические средства защиты информации в MSDN

Реализация операций над байтами в стандарте AES

Реализация преобразования SubBytes в стандарте AES

Реализация нелинейного узла замены в стандарте DES

Вычисления в группе точек эллиптических кривых

Криптосистемы на эллиптических кривых. Принципы их построения

Распределение ключей с использованием эллиптических кривых. Протокол Диффи-Хеллмана

Криптосистема Эль-Гамаля на эллиптических кривых

Электронная подпись Эль-Гамаля на эллиптических кривых

Квантовая криптография, протоколы открытого распределения ключей

НЕТ.

3.3.6 Quantum computing

Qubit và toán tử quantum

Trên máy tính hiện nay, đơn vị xử lý cơ bản là bit (0 hoặc 1). Trong máy tính lượng tử, đơn vị tính toán là qubit (quantum bit).

Qubit

Mỗi qubit $|\psi\rangle$ được biểu diễn dưới dạng tổ hợp tuyến tính của cơ sở gồm $|0\rangle = (1, 0)$ và $|1\rangle = (0, 1)$. Khi đó qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Ở đây $\alpha, \beta \in \mathbb{C}$ (tập số phức).

Tích của n qubit là các tổ hợp $|0, 0, \dots, 0\rangle$, $|0, 0, \dots, 1\rangle$, ..., $|1, 1, \dots, 1\rangle$. Ta cũng kí hiệu $|0\rangle \otimes |1\rangle = |01\rangle$.

Example 6.9

Nếu $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ và $|\Psi\rangle = \gamma|0\rangle + \delta|1\rangle$ thì

$$|\psi\rangle \otimes |\Psi\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle) = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle$$

Tiếp theo là **toán tử quantum** và tương ứng với nó là các **cổng** (gate) trên mạch.

Toán tử quantum tác động lên một qubit hoặc tích của nhiều qubit.

Qubit có dạng $|\psi\rangle = a|0\rangle + b|1\rangle$. Ta có thể viết hệ số dưới dạng vector cột $\begin{pmatrix} a \\ b \end{pmatrix}$. Khi đó, toán tử quantum sẽ là một ma trận 2×2 biến đổi vector trên thành vector mới $\begin{pmatrix} c \\ d \end{pmatrix}$ tương ứng với qubit $|\Psi\rangle = c|0\rangle + d|1\rangle$.

Nói cách khác, đặt toán tử quantum là ma trận $\mathcal{U} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$ thì ta có

$$|\psi\rangle \rightarrow |\Psi\rangle = \mathcal{U}|\psi\rangle, \quad \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix}$$

Các toán tử quantum thường gặp**Definition 6.4 (Toán tử đồng nhất)**

Toán tử đồng nhất identity giữ nguyên qubit đầu vào. Ma trận tương ứng là ma trận đơn vị $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

Definition 6.5 (Toán tử NOT)

Toán tử NOT có ma trận tương ứng là $\text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Khi đó $\text{NOT}|\psi\rangle = b|0\rangle + a|1\rangle$ với $x \in \{0, 1\}$.

Khi qubit là $|0\rangle$ hoặc $|1\rangle$, tác dụng của toán tử NOT là phép XOR nên ta có $\text{NOT}|x\rangle = |x \oplus 1\rangle$.

Definition 6.6 (Toán tử Hadamard)

Ma trận của toán tử Hadamard là $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

❶ Example 6.10

Xét qubit $|\psi\rangle = a|0\rangle + b|1\rangle$, toán tử Hadamard tương ứng với phép nhân ma trận

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}}(a+b) \\ \frac{1}{\sqrt{2}}(a-b) \end{pmatrix}$$

Ta chuyển cột kết quả về lại dạng tổ hợp tuyến tính thì cổng Hadamard hoạt động trên qubit $|\psi\rangle = a|0\rangle + b|1\rangle$ cho kết quả là

$$H|\psi\rangle = H(a|0\rangle + b|1\rangle) = \frac{1}{\sqrt{2}}(a+b)|0\rangle + \frac{1}{\sqrt{2}}(a-b)|1\rangle$$

Nếu $|\psi\rangle \equiv |0\rangle$ thì tương đương với $a = 1, b = 0$. Ta có $H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$.

Nếu $|\psi\rangle \equiv |1\rangle$ thì tương đương với $a = 0, b = 1$. Ta có $H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$.

Tổng quát ta nhận thấy, với $x \in \{0, 1\}$ thì $H|x\rangle = \frac{|0\rangle + (-1)^x|1\rangle}{\sqrt{2}}$.

Ta thấy rằng toán tử ngược của toán tử Hadamard là chính nó.

Tiếp theo là toán tử thường được dùng nhất khi tính toán trên tích của nhiều qubit: toán tử control.

Như đã xem xét ở trên, tích của n qubit sẽ có 2^n phần tử tương ứng các bộ $|0, 0, \dots, 0, 0\rangle, |0, 0, \dots, 0, 1\rangle, \dots$. Do đó toán tử control sẽ là ma trận kích thước $2^n \times 2^n$.

❷ Definition 6.7 (Toán tử control)

Gọi $\mathcal{U} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$ là toán tử tác động lên một qubit. Xét hai qubit là $|x\rangle = a|0\rangle + b|1\rangle$ và $|y\rangle = c|0\rangle + d|1\rangle$. Ta có tích

$$|x\rangle \otimes |y\rangle = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$$

Khi đó toán tử control có dạng ma trận là

$$c\mathcal{U} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & c_{11} & c_{12} \\ 0 & 0 & c_{21} & c_{22} \end{pmatrix}$$

Hay viết dưới dạng ma trận khối là $c\mathcal{U} = \begin{pmatrix} I & \mathcal{O} \\ \mathcal{O} & \mathcal{U} \end{pmatrix}$.

Ta cũng viết tích $|x\rangle \otimes |y\rangle$ dưới dạng vector cột (4 phần tử). Khi đó

$$\mathcal{U}(|x\rangle \otimes |y\rangle) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & c_{11} & c_{12} \\ 0 & 0 & c_{21} & c_{22} \end{pmatrix} \cdot \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ c_{11} \cdot bc + c_{12} \cdot bd \\ c_{21} \cdot bc + c_{22} \cdot bd \end{pmatrix}$$

Hai phần tử đầu của vector kết quả không thay đổi, còn phần sau có "một phần" là $\mathcal{U}|y\rangle$. Khi viết lại kết quả dưới dạng qubit thì

$$ac|00\rangle + ad|01\rangle + (c_{11} \cdot bc + c_{12} \cdot bd)|10\rangle + (c_{21} \cdot bc + c_{22} \cdot bd)|11\rangle$$

Ta có một số nhận xét sau đây.

➊ Remark 6.8

Nếu $|x\rangle \equiv |0\rangle$, tức là $a = 1, b = 0$ thì tích trên tương ứng với

$$c|00\rangle + d|01\rangle + 0|10\rangle + 0|11\rangle = |0\rangle \otimes (c|0\rangle + d|1\rangle) = |x\rangle \otimes |y\rangle.$$

Nếu $|x\rangle \equiv |1\rangle$, tức là $a = 0, b = 1$ thì tích trên tương ứng với

$$0|00\rangle + 0|01\rangle + (c_{11}c + c_{12}d)|10\rangle + (c_{21}c + c_{22}d)|11\rangle = |1\rangle \otimes ((c_{11}c + c_{12}d)|0\rangle + (c_{21}c + c_{22}d)|1\rangle) = |1\rangle \otimes \mathcal{U}|y\rangle = |x\rangle \otimes \mathcal{U}|y\rangle.$$

Tổng kết lại, với $x \in \{0, 1\}$ thì

- nếu $x = 0$ thì $|x\rangle \otimes |y\rangle \rightarrow |x\rangle \otimes |y\rangle$.
- nếu $x = 1$ thì $|x\rangle \otimes |y\rangle \rightarrow |x\rangle \otimes \mathcal{U}|y\rangle$.

Tùy vào x là 0 hay 1 mà toán tử quantum \mathcal{U} sẽ bị bỏ qua hoặc xem xét. Ở đây qubit $|x\rangle$ đóng vai trò điều khiển nên đây được gọi là toán tử control.

➊ Definition 6.8 (Toán tử control CNOT, Control NOT)

Toán tử quantum CNOT có ma trận là

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} I & \mathcal{O} \\ \mathcal{O} & \text{NOT} \end{pmatrix}$$

Qubit $|x\rangle$ với $x \in \{0, 1\}$ đóng vai trò control cho qubit $|y\rangle$. Khi $x \equiv 0$ thì y giữ nguyên, hay $|y \oplus 0\rangle = |y\rangle$. Khi $x \equiv 1$ thì áp dụng cổng NOT bên trên, khi đó y biến đổi thành $y \oplus 1 = y \oplus x$.

3.3.7 Zero Knowledge Proof

Giới thiệu

Zero Knowledge Proof (ZKP) là một protocol mật mã cho phép một bên thuyết phục bên còn lại rằng họ sở hữu những thông tin quan trọng mà không để lộ bất cứ gì về chính thông tin quan trọng đó.

Khi đó, bên thuyết phục được gọi là **prover**, bên đưa ra thử thách để prover chứng minh bản thân gọi là **challenger** hoặc **verifier**.

➊ Tính chất của zero knowledge proof

Mỗi protocol zero-knowledge phải đảm bảo ba tính chất sau:

1. Completeness (tính đầy đủ): nếu mệnh đề đúng thì verifier có thể xác nhận và bị thuyết phục bởi prover.

2. Soundness: nếu mệnh đề sai thì prover không thể thuyết phục verifier rằng nó đúng.
3. Zero-knowledge: verifier không biết gì về tính đúng sai của mệnh đề.

Lấy một ví dụ đơn giản để xem cách hoạt động của ZKP là protocol QR¹.

Protocol PQ

Mệnh đề cần kiểm tra: x là một thăng dư chính phương modulo n . Prover muốn chứng minh với verifier rằng mình biết căn bậc hai của x trong modulo n .

Public input

Số x và modulo n , với $0 \leq x < n$.

Quá trình tạo proof

Quá trình tạo proof (thuyết phục) diễn ra như sau:

Prover (Alice)	Verifier (Bob)
Public input: $x \pmod{n}$	
Private input: $w \pmod{n}$ thỏa $x = w^2 \pmod{n}$	
Chọn ngẫu nhiên số u từ \mathbb{Z}_n^*	
Gửi Bob $y = u^2 \pmod{n}$	Chọn ngẫu nhiên $b \in \{0, 1\}$
	Gửi b cho Alice
Gửi $z = w^b \cdot u$ cho Bob	

Quá trình verification

Gọi z là số được gửi bởi Alice. Bob kiểm tra

$$z^2 \stackrel{?}{=} x^b \cdot y.$$

Như vậy:

1. Nếu $b = 0$ thì $z^2 = y = u^2 \pmod{n}$.
2. Nếu $b = 1$ thì $z^2 = xy = (wu)^2 \pmod{n}$.

Nguyên lý hoạt động

Bob chỉ biết x và n trong khi Alice biết căn bậc hai của x modulo n .

Ở đây Alice muốn thuyết phục Bob rằng mình biết căn bậc hai của x trong modulo n .

Nếu Alice thật sự biết căn bậc hai của x là w , hay $x = w^2 \pmod{n}$, thì Alice cần chứng minh cho Bob thấy.

1. Alice chọn số random $u \in \mathbb{Z}_n^*$ và gửi $y = u^2 \pmod{n}$ cho Bob.
2. Bob chọn ngẫu nhiên $b \in \{0, 1\}$ và gửi cho Alice.

Với mỗi trường hợp của b :

¹ <https://www.cs.princeton.edu/courses/archive/fall07/cos433/lec15.pdf>

- nếu $b = 0$ thì Alice cần tính căn bậc hai của y modulo n và gửi cho Bob. Đó chính là u ;
- nếu $b = 1$ thì Alice cần tính căn bậc hai của xy (x public và y được gửi trước đó). Ta có $xy = w^2u^2 \pmod{n}$ nên Alice cần gửi wu . Nếu Alice thật sự biết căn bậc hai của x thì có thể tính được wu ;

Bob có thể kiểm tra số z được gửi tới có thỏa mãn $z^2 = y \pmod{n}$ (nếu $b = 0$) hoặc thỏa mãn $z^2 = xy \pmod{n}$ (nếu $b = 1$) hay không.

Trong ví dụ trên, ta thấy các tính chất của ZKP:

1. Completeness: khi x thực sự là số chính phương modulo n và Alice có thể đưa số w sao cho $x = w^2 \pmod{n}$ thì Bob sẽ chấp nhận với xác suất bằng 1. Điều này khá dễ thấy;
2. Soundness: nếu x không là số chính phương modulo n thì Bob có thể bác bỏ chứng minh của Alice với xác suất ít nhất $1/2$ (trong trường hợp $b = 1$). Trong khi đó $b = 0$ thì vẫn có "cơ may" đúng;
3. Zero knowledge: Bob không biết bất cứ thông tin nào liên quan đến Alice nhưng Alice có thể thuyết phục Bob tin rằng mình biết căn bậc hai của x . Zero knowledge có nghĩa là trong suốt quá trình Bob không biết thêm thông tin gì hơn từ Alice.

Mô hình Zero Knowledge Proof

Phần tiếp theo được tổng hợp từ một số nguồn dành cho newbie².

ZKP bao gồm ba bước là: key generation, proof generation và verification.

1. Key generation

Bước này tạo các tham số để một bên proof và để bên còn lại verification.

Thông thường, ở bước này có một hàm $C(x, w)$ nhận hai tham số là x (public input) và w (private input, witness).

Một tham số private là λ giúp việc sinh ra các tham số gồm public key pk và private key vk .

Ký hiệu: $\text{Setup}(C, \lambda) \rightarrow (pk, vk)$.

Trong đó pk được gửi cho prover để họ chứng minh bản thân (thuyết phục verifier rằng họ sở hữu thông tin).

Ngược lại, vk được gửi cho verifier để họ kiểm tra mệnh đề từ prover.

2. Proof generation

Bước này thực hiện bởi prover để sinh ra giá trị gửi cho bên verifier kiểm tra.

Với đầu vào gồm private input là w , public input là x và public key là pk , prover sẽ tính toán prf rồi gửi cho verifier.

Ký hiệu: $\text{Prove}(w, x, pk) \rightarrow \text{prf}$.

3. Verification

Bước này thực hiện bởi verifier để kiểm tra mệnh đề prf được gửi từ prover.

Với đầu vào gồm public input x , proof là prf và private key là vk , verifier kiểm tra tính đúng đắn của prf .

Ký hiệu: $\text{Verify}(x, \text{prf}, vk) \rightarrow \text{đúng} (\text{nếu hợp lệ}) \text{ hoặc sai} (\text{ngược lại})$.

² What is a zk-SNARK? URL - <https://blog.thirdweb.com/what-is-a-zk-snark/>

Plonk Protocol

[TODO] Viết lại.

Elliptic curve và pairing

Elliptic curve

Đường cong elliptic trên trường \mathbb{F} có dạng $y^2 = x^3 + ax + b$ với x, y, a, b thuộc \mathbb{F} và a, b là các phần tử cho trước.

Khi đó đường cong elliptic là tập hợp các điểm (x, y) thỏa mãn phương trình trên và điểm vô cực \mathcal{O} .

Để lấy ví dụ, xét \mathbb{F}_{101} là trường modulo 101 và đường cong elliptic với phương trình $y^2 = x^3 + 3 \pmod{101}$.

Pairing

Gọi $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ là các nhóm có cùng order bằng r và e là một pairing. Khi đó e là ánh xạ sao cho

$$\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t : e(g_1, g_2) = g_t.$$

Embedding degree

Order của đường cong elliptic trên là $102 = 2 \cdot 3 \cdot 17$. Như vậy tồn tại các điểm trên đường cong có order là 17. Tuy nhiên, chung ta cần nhiều điểm khác cũng có order là 17 nhưng không nằm trên đường cong \mathbb{F}_{101} . Lúc này chúng ta sẽ mở rộng trường lên \mathbb{F}_{101^k} .

Bậc thấp nhất của mở rộng trường (với characteristic p) chứa tất cả điểm với order r được gọi là **embedding degree**. Nói đơn giản, embedding degree là số k nhỏ nhất mà $r \mid (p^k - 1)$.

Trong ví dụ này thì $r = 17$ là order của điểm và $k = 2$.

Khi đó trường mở rộng \mathbb{F}_{101^2} với đa thức tối giản $x^2 + 2$ cho đường cong với các điểm order 17.

Mỗi phần tử trong \mathbb{F}_{101^2} có dạng $a + bx$ với $a, b \in \mathbb{F}_{101}$.

Mạch Plonk

Trusted setup

Chúng ta cần một trusted setup gọi là **structured reference string** (SRS, chuỗi liên kết cấu trúc).

SRS là một danh sách các điểm trên đường cong elliptic được tính toán từ một số bí mật sinh ngẫu nhiên s . Theo paper PLONK thì một mạch n cổng sẽ cần $n + 5$ điểm sau:

$$\begin{aligned} \text{SRS : } & 1 \cdot G_1, s \cdot G_1, \dots, s^{n+2} \cdot G_1, \\ & 1 \cdot G_2, s \cdot G_2. \end{aligned}$$

Trong ví dụ này, s giúp sinh ra các điểm trong subgroup với generator là G_1 . Do G_1 có order là 17, s không được vượt quá 17.

Chọn $G_1 = (1, 2)$ và $G_2 = (36, 31x)$ là hai generator cho hai subgroup đều có order là 17.

Xét $s = 2$ (cho dễ tính) và $n = 4$ (có 4 cổng). Khi đó $1 \cdot G_1, s \cdot G_1, \dots, s^{n+2} \cdot G_1$ là các điểm sau:

$$\left| \begin{array}{l} 1 \cdot G_1 = (1, 2) \\ s^4 \cdot G_1 = (1, 99) \end{array} \right| \left| \begin{array}{l} s \cdot G_1 = (68, 74) \\ s^5 \cdot G_1 = (68, 27) \end{array} \right| \left| \begin{array}{l} s^2 \cdot G_1 = (65, 98) \\ s^6 \cdot G_1 = (65, 3) \end{array} \right| \left| \begin{array}{l} s^3 \cdot G_1 = (18, 49) \end{array} \right|$$

Tương tự, $1 \cdot G_2$ và $s \cdot G_2$ là các điểm $(36, 31x)$ và $(90, 82x)$.

Như vậy,

$$\begin{aligned} \text{SRS : } & (1, 2), (68, 74), (65, 98), (18, 49), (1, 99), (68, 27), (65, 3) \\ & (36, 31x), (90, 82x). \end{aligned}$$

Mạch PLONK

Plonk circuit là mạch được biểu diễn ở dạng đa thức (gồm phép cộng và nhân). Trong đó mỗi cỗng, hay constrain, thực hiện một thao tác (cộng hoặc nhân).

Ví dụ với định lý Pythagoras

Giả sử chúng ta có bộ ba $(3, 4, 5)$ và chúng ta muốn kiểm tra xem chúng có thỏa mãn phương trình Pythagoras $a^2 + b^2 = c^2$ hay không.

Giả sử bộ ba $(3, 4, 5)$ sẽ đi vào ba đầu là x_1, x_3, x_5 . Khi đó mạch tính

$$\begin{aligned} x_1 \cdot x_1 &= x_2 \\ x_3 \cdot x_3 &= x_4 \\ x_5 \cdot x_5 &= x_6 \\ x_2 + x_4 &= x_6 \end{aligned}$$

Phương trình cuối cho kết quả $x_1^2 + x_3^2 = x_5^2$ là điều kiện thỏa mãn phương trình Pythagoras. Mỗi x_i được gọi là "wire" (dây). Mạch này đi qua ba cỗng nhân và một cỗng cộng. Ta có thể viết tất cả x_i thành vector $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6)$. Đối với ví dụ bộ ba $(3, 4, 5)$ thì

$$\mathbf{x} = (3, 9, 4, 16, 5, 25).$$

Tương tự, đối với bộ ba $(5, 12, 13)$ thì

$$\mathbf{x} = (5, 25, 12, 144, 13, 169).$$

Mỗi cỗng plonk sẽ gồm hai dây input trái và phải, và một dây output. Ta gọi dây input trái và phải lần lượt là a và b , còn dây output là c .

Cỗng plonk phức tạp hơn cỗng cộng hoặc nhân để kiểm tra bộ ba Pythagoras ở trên. Cỗng plonk đầy đủ có dạng

$$(q_L) \cdot a + (q_R) \cdot b + (q_O) \cdot c + (q_M) \cdot ab + q_C = 0.$$

Trong đó a, b là hai dây input trái và phải, c là dây output, còn lại là các hệ số cho trước.

Ví dụ. Đối với đẳng thức $a + b = c$ thì ta cho $q_L = q_R = 1$, $q_O = -1$ và $q_M = q_C = 0$.

Mạch kiểm tra bộ ba Pythagoras bên trên có thể được viết lại dưới dạng mạch với các cỗng plonk như sau

$$\begin{array}{ccccccccc} 0 \cdot a_1 & + & 0 \cdot b_1 & + & (-1) \cdot c_1 & + & 1 \cdot a_1 b_1 & + & 0 = 0 \\ 0 \cdot a_2 & + & 0 \cdot b_2 & + & (-1) \cdot c_2 & + & 1 \cdot a_2 b_2 & + & 0 = 0 \\ 0 \cdot a_3 & + & 0 \cdot b_3 & + & (-1) \cdot c_3 & + & 1 \cdot a_3 b_3 & + & 0 = 0 \\ 1 \cdot a_4 & + & 1 \cdot b_4 & + & (-1) \cdot c_4 & + & 0 \cdot a_3 b_3 & + & 0 = 0 \end{array}$$

Bốn cỗng plonk này tương đương bốn cỗng kiểm tra bên trên. Tương tự bên trên, nếu gọi \mathbf{a} là vector các dây input trái, \mathbf{b} là vector các dây input phải và \mathbf{c} là các dây output thì

$$\mathbf{a} = (3, 4, 5, 9), \mathbf{b} = (3, 4, 5, 16), \mathbf{c} = (9, 16, 25, 25).$$

Ta cũng viết hệ số cổng plonk dưới dạng vector, khi đó

$$\begin{aligned}\mathbf{q_L} &= (0, 0, 0, 1), \\ \mathbf{q_R} &= (0, 0, 0, 1), \\ \mathbf{q_O} &= (-1, -1, -1, -1), \\ \mathbf{q_M} &= (1, 1, 1, 0), \\ \mathbf{q_C} &= (0, 0, 0, 0)\end{aligned}$$

Lúc này, các vector \mathbf{q} được gọi là **selector** và sẽ giúp xây dựng nên cấu trúc của mạch. Các vector $\mathbf{a}, \mathbf{b}, \mathbf{c}$ thể hiện các biến của mạch và là private input của Prover (bên chứng minh ZKP).

Proof

Roots of unity

Trong trường \mathbb{F} có phần tử đơn vị là 1. Nghiệm bậc n của 1 là các nghiệm x thỏa mãn phương trình $x^n = 1$.

Ở ví dụ trên, mỗi subgroup đều có order là 17. Bây giờ các tính toán của chúng ta sẽ nằm trong \mathbb{F}_{17} .

Nhắc lại, các vector $\mathbf{a}, \mathbf{b}, \mathbf{c}$ cũng như các vector hệ số $\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C}$ đều có bốn phần tử. Chúng ta có cách xây dựng đa thức $f(x)$ từ các cặp $(x_i, f(x_i))$ là **đa thức nội suy**.

Khi đó, mỗi vector ở trên sẽ tương ứng $f(x_i)$ với $1 \leq i \leq 4$. Đa thức sinh ra là đa thức bậc 3.

Üng với mỗi vector $\mathbf{a}, \mathbf{b}, \mathbf{c}$ ta xây dựng một đa thức bậc 3. Do có $3 \cdot 4 = 12$ giá trị x_i nên ta cần một "cơ chế" để tách trường \mathbb{F}_{17} thành 3 tập rời nhau, mỗi tập 4 phần tử.

Đầu tiên là cần một tập con của \mathbb{F}_{17} có 4 phần tử. Tập con này là tập nghiệm của $x^4 = 1 \pmod{17}$. Phương trình này có nghiệm và may thay là có đủ 4 nghiệm. Gọi H là tập các nghiệm của phương trình $x^4 = 1 \pmod{17}$ thì $H = \{1, 4, 16, 13\}$.

Ta cần tìm thêm hai tập con khác của \mathbb{F}_{17} , cũng chứa 4 phần tử và không giao nhau với H . Chúng ta sẽ dùng coset vì hai coset bất kì của nhóm hoặc là không giao nhau, hoặc là trùng nhau.

Chọn $k_1 = 2$ và $k_2 = 3$ thì ta có các coset

$$1H = \{1, 4, 16, 13\}, \quad k_1 H = \{2, 8, 15, 9\}, \quad k_2 H = \{3, 12, 14, 5\}.$$

Với vector $\mathbf{a} = \{a_1, a_2, a_3, a_4\}$, ta mong muốn $f_{\mathbf{a}}(h_i) = a_i$ với $h_i \in H$, nghĩa là $f_{\mathbf{a}}(1) = 3, f_{\mathbf{a}}(4) = 4, f_{\mathbf{a}}(16) = 5$ và $f_{\mathbf{a}}(13) = 9$. Sử dụng đa thức nội suy Lagrange có thể tìm được hàm $f_{\mathbf{a}}(x) = 1 + 13x + 3x^2 + 3x^3$.

Tương tự ta cũng tìm được đa thức nội suy cho các vector còn lại (tất cả đều dùng H).

$$\begin{aligned}f_{\mathbf{a}}(x) &= 1 + 13x + 3x^2 + 3x^3 \\ f_{\mathbf{b}}(x) &= 7 + 3x + 14x^2 + 13x^3 \\ f_{\mathbf{c}}(x) &= 6 + 5x + 11x^2 + 4x^3 \\ f_{\mathbf{q_L}}(x) &= 13 + x + 4x^2 + 16x^3 \\ f_{\mathbf{q_R}}(x) &= 13 + x + 4x^2 + 16x^3 \\ f_{\mathbf{q_O}}(x) &= 16 \\ f_{\mathbf{q_M}}(x) &= 5 + 16x + 13x^3 + x^3 \\ f_{\mathbf{q_C}}(x) &= 0\end{aligned}$$

Một vấn đề xảy ra là, bốn cổng plonk ở trên là rời nhau và không liên quan gì nhau. Do đó chúng ta cần một phương án để "nối" output từ cổng này thành input của cổng kia.

Để làm việc này, ta đánh nhãn vector \mathbf{a} bởi H , vector \mathbf{b} bởi k_1H và vector \mathbf{c} bởi k_2H . Cụ thể ta có ánh xạ

$a_1 = b_1$	$1 \rightarrow 2$
$a_2 = b_2$	$4 \rightarrow 8$
$a_3 = b_3$	$16 \rightarrow 15$
$a_4 = c_1$	$13 \rightarrow 3$
$b_1 = a_1$	$2 \rightarrow 1$
$b_2 = a_2$	$8 \rightarrow 4$
$b_3 = a_3$	$15 \rightarrow 16$
$b_4 = c_2$	$9 \rightarrow 12$
$c_1 = a_4$	$3 \rightarrow 13$
$c_2 = b_4$	$12 \rightarrow 8$
$c_3 = c_4$	$14 \rightarrow 5$
$c_4 = c_3$	$5 \rightarrow 14$

Gọi $\sigma_1, \sigma_2, \sigma_3$ lần lượt là output khi ánh xạ tác động lên H, k_1H, k_2H . Theo bảng trên thì $\sigma_1 = (2, 8, 15, 3), \sigma_2 = (1, 4, 16, 12), \sigma_3 = (13, 8, 5, 14)$.

Mỗi tác động như vậy thực chất tương ứng với một đa thức bậc 3.

Gọi S_{σ_1} là đa thức từ H tới σ_1 . Sử dụng nội suy Lagrange ta tính được $S_{\sigma_1} = 7 + 13x + 10x^2 + 6x^3$.

Tương tự, gọi S_{σ_2} là đa thức từ k_1H tới σ_2 và S_{σ_3} là đa thức từ k_2H tới σ_3 . Ta tính được $S_{\sigma_2} = 4 + 13x^2 + x^3$ và $S_{\sigma_3} = 6 + 7x + 3x^2 + 14x^3$.

Đến đây ta có đủ thành phần để tạo proof, gồm 5 vòng.

Round 1

Round 1 bao gồm các bước:

- tạo random các phần tử $b_1, \dots, b_6 \in \mathbb{F}_{17}$;
- tính các đa thức $a(x), b(x), c(x)$ theo công thức

$$\begin{aligned} a(x) &= (b_1x + b_2) \cdot Z_H + f_a(x) \\ b(x) &= (b_3x + b_4) \cdot Z_H + f_b(x) \\ c(x) &= (b_5x + b_6) \cdot Z_H + f_c(x). \end{aligned}$$

Output là $[a(s)], [b(s)], [c(s)]$, trong đó Z_H là đa thức sao cho mọi phần tử của subgroup H là nghiệm của Z_H . Do các phần tử trong H là nghiệm của $x^4 = 1 \pmod{17}$ nên $Z_H = x^4 - 1$.

Giả sử ta chọn (ngẫu nhiên) các phần tử $b_1 = 7, b_2 = 4, b_3 = 11, b_4 = 12, b_5 = 16, b_6 = 2$. Khi đó

$$\begin{aligned} Z_H(x) &= x^4 - 1 \\ a(x) &= \dots = 14 + 6x + 3x^2 + 3x^3 + 4x^4 + 7x^5 \\ b(x) &= \dots = 12 + 9x + 14x^2 + 13x^3 + 12x^4 + 11x^5 \\ c(x) &= \dots = 4 + 6x + 11x^2 + 4x^3 + 2x^4 + 16x^5. \end{aligned}$$

Tiếp theo ta tính $[a(s)], [b(s)], [c(s)]$ là các điểm trên đường cong sử dụng SRS ban đầu và hệ số của các đa thức $a(x), b(x), c(x)$. Cụ thể prover sẽ tính

$$\begin{aligned} [a(s)] &= 14 \cdot (1, 2) + 6 \cdot (68, 74) + 3 \cdot (65, 98) \\ &\quad + 3 \cdot (18, 49) + 4 \cdot (1, 99) + 7 \cdot (68, 27) = (91, 66). \end{aligned}$$

Tương tự cho $[b(s)]$ và $[c(s)]$. Như vậy ta có

$$[a(s)] = (91, 66), [b(s)] = (26, 45), [c(s)] = (91, 35).$$

Round 2

Round 2 được thực hiện qua các bước:

- random các phần tử $b_7, b_8, b_9 \in \mathbb{F}_{17}$;
- nhận challenge từ verifier là $\beta, \gamma \in \mathbb{F}_{17}$;
- tính $z(x)$ theo công thức:

$$z(x) = (b_7x^2 + b_8x + b_9) \cdot Z_H(x) + \text{acc}(x).$$

Ở round này, chúng ta *commit* một đa thức $z(x)$ để encode cho việc copy contrains (nối các dây của cỗng) bên trên. Đa thức $\text{acc}(x)$ sẽ được đề cập sau.

Các giá trị challenge β, γ phụ thuộc vào protocol là interactive (tương tác) hay non-interactive.

- trong protocol interactive, verifier chọn các giá trị này và gửi cho prover để prover tạo ra proof;
- trong protocol non-interactive, các giá trị này sinh ra từ quá trình tính toán của prover và đi qua một hàm hash mật mã.

Trong cả hai trường hợp, giá trị này là không thể đoán trước và cần được tính giống nhau (về hàm hash, về trường) ở cả hai phía. Do đó quá trình này còn được gọi là **biến đổi Fiat-Shamir** có thể biến interactive thành non-interactive protocol.

Trước tiên ta cần tìm hiểu cơ chế interactive, ở đó verifier gửi các challenge cho prover.

Giả sử $\beta = 12, \gamma = 13$.

Hàm $z(x)$ ở trên được xây dựng từ các **accumulator** vector, định nghĩa như sau:

$$\begin{aligned} \text{acc}_0 &= 1 \\ \text{acc}_i &= \text{acc}_{i-1} \cdot \left(\frac{(a_i + \beta \cdot w^{i-1} + \gamma) \cdot (b_i + \beta k_1 w^{i-1} + \gamma) \cdot (c_i + \beta k_2 w^{i-1} + \gamma)}{(a_i + \beta S_{\sigma_1}(w^{i-1}) + \gamma) \cdot (b_i + \beta k_1 S_{\sigma_2}(w^{i-1}) + \gamma) \cdot (c_i + \beta k_2 S_{\sigma_3}(w^{i-1}) + \gamma)} \right) \end{aligned}$$

Do cách định nghĩa S_{σ} ở trên mà sẽ có những phần ở tử và mẫu giản lược nhau, và tất nhiên là một số thì không.

Bằng tính toán trực tiếp thu được

$$\text{acc}_0 = 1, \text{ acc}_1 = 3, \text{ acc}_2 = 9, \text{ acc}_3 = 4.$$

Khi đó đa thức $\text{acc}(x)$ tương ứng với đa thức nội suy từ nguồn là subgroup $H = \{1, 4, 16, 13\}$ và ảnh tương ứng là vector $\text{acc} = (1, 3, 9, 4)$.

Nói cách khác là $\text{acc}(1) = 1, \text{ acc}(4) = 3, \text{ acc}(16) = 9$ và $\text{acc}(13) = 4$. Từ đó ta có

$$\text{acc}(x) = 16x + 5x^2 + 14x^3.$$

Thay vào công thức tính $z(x)$ ta có

$$\begin{aligned} z(x) &= (14x^2 + 11x + 7)(x^4 - 1) + 16x + 5x^2 + 14x^3 \\ &= 10 + 5x + 8x^2 + 14x^3 + 7x^4 + 11x^5 + 14x^6. \end{aligned}$$

Cuối cùng thay giá trị secret s vào $z(x)$ ta có $z(s) = z(2) = 11$ và tính $[z(s)] = 11(1, 2) = (32, 59)$.

Round 3

Round này chứa một lượng lớn tính toán. Chúng ta sẽ tính đa thức $t(x)$ bậc $3n + 5$ với n là số lượng cỗng. Đa thức $t(x)$ sẽ chứa mạch và tắt cả assignment cùng lúc.

Quá trình ở round 3 gồm:

- tính quotient challenge $\alpha \in \mathbb{F}_{17}$;
- tính quotient polynomial $t(x)$

$$\begin{aligned} t(x) = & (a(x)b(x)q_M(x) + a(x)q_L(x) + b(x)q_R(x) + c(x)q_O(x) + \text{PI}(x) + q_C(x)) \frac{1}{Z_H(x)} \\ & + ((a(x) + \beta x + \gamma)(b(x) + \beta k_1 x + \gamma)(c(x) + \beta k_2 x + \gamma)z(x)) \frac{\alpha}{Z_H(x)} \\ & - ((a(x) + \beta S_{\sigma_1}(x) + \gamma)(b(x) + \beta S_{\sigma_2}(x) + \gamma)(c(x) + \beta S_{\sigma_3}(x) + \gamma)z(xw)) \frac{\alpha}{Z_H(x)} \\ & + (z(x) - 1)L_1(x) \frac{\alpha^2}{Z_H(x)} \end{aligned}$$

- tách $t(x)$ thành các đa thức bậc nhỏ hơn $n + 2$ là $t_{10}(x)$, $t_{\text{mid}}(x)$ và $t_{\text{ni}}(x)$ sao cho

$$t(x) = t_{10}(x) + x^{n+2}t_{\text{mid}}(x) + x^{2n+4}t_{\text{ni}}(x).$$

Đầu ra của round 3 là $[t_{10}(s)]$, $[t_{\text{mid}}(x)]$ và $[t_{\text{ni}}(x)]$.

Dòng cuối có $L_1(x)$ sẽ được định nghĩa dưới đây.

$L_1(x)$ là cơ sở của đa thức nội suy từ H . Với $L_1(1) = 1$, các giá trị còn lại cho ảnh bằng 0. Khi đó $L_1(x)$ tương ứng vector $(1, 0, 0, 0)$ nên $L_1(x) = 13 + 13x + 13x^3 + 13x^7$.

Verifier chọn số α (ngẫu nhiên) và gửi cho prover.

Round 4

- verifier chọn một số \mathfrak{z} và gửi cho prover;
- prover khi đó cần tính

$$\begin{aligned} \bar{a} &= a(\mathfrak{z}) & \bar{b} &= b(\mathfrak{z}) & \bar{c} &= c(\mathfrak{z}) \\ \bar{S}_{\sigma_1} &= S_{\sigma_1}(\mathfrak{z}) & \bar{S}_{\sigma_2} &= S_{\sigma_2}(\mathfrak{z}) \\ \bar{t} &= t(\mathfrak{z}) \\ \bar{z}_w &= z(\mathfrak{z}w) \end{aligned}$$

- tính đa thức linearization

$$\begin{aligned} r(x) = & \bar{a}\bar{b}q_M(x) + \bar{a}q_L(x) + \bar{b}q_R(x) + \bar{c}q_O(x) + q_C(x) \\ & + ((\bar{a} + \beta \mathfrak{z} + \gamma)(\bar{b} + \beta k_1 \mathfrak{z} + \gamma)(\bar{c} + \beta k_2 \mathfrak{z} + \gamma) \cdot z(x))\alpha \\ & - ((\bar{a} + \beta \bar{S}_{\sigma_1} + \gamma)(\bar{b} + \beta \bar{S}_{\sigma_2} + \gamma)\beta \bar{z}_w \cdot S_{\sigma_3}(s))\alpha \\ & + z(x)L_1(\mathfrak{z})\alpha^2 \end{aligned}$$

- tính linearization evaluation $\bar{r} = r(\mathfrak{z})$.

Output của round 5 là các giá trị

$$(\bar{a}, \bar{b}, \bar{c}, \bar{S}_{\sigma_1}, \bar{S}_{\sigma_2}, \bar{z}_w, \bar{t}, \bar{z}).$$

Round 5

Verifier chọn số ngẫu nhiên $v \in \mathbb{F}_{17}$ và gửi cho prover.

- tính đa thức đầu cho proof là $W_{\mathfrak{z}}(x)$ như sau

$$W_{\mathfrak{z}}(x) = \frac{1}{x - \mathfrak{z}} \begin{pmatrix} t_{10}(x) + \mathfrak{z}^{n+2}t_{\text{mid}}(x) + \mathfrak{z}^{2n+4}t_{\text{ni}}(x) - \bar{t} \\ +v(r(x) - \bar{r}) \\ +v^2(a(x) - \bar{a}) \\ +v^3(b(x) - \bar{b}) \\ +v^4(c(x) - \bar{c}) \\ +v^5(S_{\sigma_1}(x) - \bar{S}_{\sigma_1}) \\ +v^6(S_{\sigma_2}(x) - \bar{S}_{\sigma_2}) \end{pmatrix}$$

- tính đa thức tiếp theo cho proof là $W_{\mathfrak{z}w}(x)$:

$$W_{\mathfrak{z}w}(x) = \frac{z(x) - \bar{z}_w}{x - \mathfrak{z}w}.$$

Output của round 5 là $[w_{\mathfrak{z}}], [w_{\mathfrak{z}w}]$.

Proof

Proof thu được là vector π_{SNARK} như sau

$$\pi_{\text{SNARK}} = ([a], [b], [c], [z], [t_{10}], [t_{\text{mid}}], [t_{\text{ni}}], [W_{\mathfrak{z}}], [W_{\mathfrak{z}w}], \bar{a}, \bar{b}, \bar{c}, \bar{S}_{\sigma_1}, \bar{S}_{\sigma_2}, \bar{r}, \bar{z}_w).$$

Code mẫu với SageMath

```
from sage.all import *

F101 = GF(101)['xn']; xn = F101.gen()
F101_2 = GF(101**2, name='yn', modulus=xn**2 + 2)

Ec = EllipticCurve(F101_2, [0, 3])

G1 = Ec(1, 2)
G2 = Ec(36, 31*xn)

s = 2
n = 4
SRS = []
for i in range(n+3):
    SRS.append(s**i * G1)
for i in range(2):
    SRS.append(s**i * G2)

print([srs.xy() for srs in SRS])

F17 = GF(17)
Pol = PolynomialRing(F17, 'x')
x = Pol.gen()
```

(continues on next page)

(continued from previous page)

```

Z_H = x**4 - 1

k1, k2 = 2, 3
H = [F17(i) for i in [1, 4, 16, 13]]
assert set(H) == set([z[0] for z in Z_H.roots()])
k1_H = [F17(k1 * h) for h in H]
k2_H = [F17(k2 * h) for h in H]

print(f"k1 H = {k1_H}, k2 H = {k2_H}")

ai = [3, 4, 5, 9]
bi = [3, 4, 5, 16]
ci = [9, 16, 25, 25]
qLi = [0, 0, 0, 1]
qRi = [0, 0, 0, 1]
qQi = [-1, -1, -1, -1]
qMi = [1, 1, 1, 0]
qCi = [0, 0, 0, 0]

fa = Pol.lagrange_polynomial(list(zip(H, ai)))
fb = Pol.lagrange_polynomial(list(zip(H, bi)))
fc = Pol.lagrange_polynomial(list(zip(H, ci)))
fQL = Pol.lagrange_polynomial(list(zip(H, qLi)))
fQR = Pol.lagrange_polynomial(list(zip(H, qRi)))
fQ0 = Pol.lagrange_polynomial(list(zip(H, qQi)))
fQM = Pol.lagrange_polynomial(list(zip(H, qMi)))
fQC = Pol.lagrange_polynomial(list(zip(H, qCi)))

sigma_1 = list(map(F17, [2, 8, 15, 3]))
sigma_2 = list(map(F17, [1, 4, 16, 12]))
sigma_3 = list(map(F17, [13, 8, 5, 14]))

S_sig_1 = Pol.lagrange_polynomial(list(zip(H, sigma_1)))
S_sig_2 = Pol.lagrange_polynomial(list(zip(k1_H, sigma_2)))
S_sig_3 = Pol.lagrange_polynomial(list(zip(k2_H, sigma_3)))

# Dưới đây là kết quả tính theo ví dụ, khác phần trên
S_sig_2 = x**3 + 13*x**2 + 4
S_sig_3 = 14*x**3 + 3*x**2 + 7*x + 6

# Round 1

b_coeffs = [7, 4, 11, 12, 16, 2]

ax = (b_coeffs[0] * x + b_coeffs[1]) * Z_H + fa
bx = (b_coeffs[2] * x + b_coeffs[3]) * Z_H + fb
cx = (b_coeffs[4] * x + b_coeffs[5]) * Z_H + fc

ass = sum(i * j for i, j in zip(ax.coefficients(), SRS))
bss = sum(i * j for i, j in zip(bx.coefficients(), SRS))
css = sum(i * j for i, j in zip(cx.coefficients(), SRS))

```

(continues on next page)

(continued from previous page)

```

print(ass.xy(), bss.xy(), css.xy())

# Round 2

b_coeffs.extend([14, 11, 7])

beta_, gamma_ = 12, 13

w = 4 # Choose from H
acc = [1]
for i in range(1, len(H)):
    numerator_ = (ai[i-1] + beta_* w***(i-1) + gamma_)
    numerator_ *= (bi[i-1] + beta_* k1 * w***(i-1) + gamma_)
    numerator_ *= (ci[i-1] + beta_* k2 * w***(i-1) + gamma_)

    denominator_ = (ai[i-1] + beta_* S_sig_1(w***(i-1)) + gamma_)
    denominator_ *= (bi[i-1] + beta_* S_sig_2(w***(i-1)) + gamma_)
    denominator_ *= (ci[i-1] + beta_* S_sig_3(w***(i-1)) + gamma_)

    acc.append(acc[-1] * F17(numerator_) / F17(denominator_))

accx = Pol.lagrange_polynomial(list(zip(H, acc)))
print(accx)

zx = (b_coeffs[6] * x**2 + b_coeffs[7] * x + b_coeffs[8]) * Z_H + accx
zs = zx(s) * G1
print(f"z(s) = {zs.xy()}")


alpha = 15

L1x = Pol.lagrange_polynomial(list(zip(H, [1, 0, 0, 0])))

tx = (ax * bx * fqM + ax * fqL + bx * fqR + cx * fqO + 0 + fqC)
tx += (ax + beta_* x + gamma_) * (bx + beta_* k1 * x + gamma_) * (cx + beta_* k2 * x
    ↪+ gamma_) * zx * alpha
tx -= (ax + beta_* S_sig_1 + gamma_) * (bx + beta_* S_sig_2 + gamma_) * (cx + beta_* S_sig_3 + gamma_) * zx(w * x) * alpha
tx += (zx - 1) * L1x * alpha * alpha

#print(tx.coefficients(sparse=False))

print(tx % Z_H)

tx /= (x**4 - 1)

tx_c = tx.coefficients(sparse=False)

#print(tx_c)

t10 = Pol(tx_c[0:6])
tmid = Pol(tx_c[6:12])
tni = Pol(tx_c[12:18])

```

(continues on next page)

(continued from previous page)

```

t10s = t10(s) * G1
tmids = tmid(s) * G1
tnis = tni(s) * G1

zz = 5

abar = ax(zz)
bbar = bx(zz)
cbar = cx(zz)
S_sig_1_bar = S_sig_1(zz)
S_sig_2_bar = S_sig_2(zz)
tbar = tx(zz)
zwbar = zx(w * zz)

rx = abar * bbar * fqM + abar * fqL + bbar * fqR + cbar * fq0 + fqC
rx += (abar + beta_ * zz + gamma_) * (bbar + beta_ * k1 * zz + gamma_) * (cbar + beta_ * k2 * zz + gamma_) * zx * alpha
rx -= (abar + beta_ * S_sig_1_bar + gamma_) * (bbar + beta_ * S_sig_2_bar + gamma_) * beta_ * zwbar * S_sig_3 * alpha
rx += zx * L1x(zz) * alpha**2

rbar = rx(zz)

print(abar, bbar, cbar, S_sig_1_bar, S_sig_2_bar, zwbar, tbar, rbar)

v = F17(12)

wzx = t10 + zz**(n+2) * tmid + zz**(2*n+4) * tni - tbar
wzx += v * (rx - rbar)
wzx += v**2 * (ax - abar)
wzx += v**3 * (bx - bbar)
wzx += v**4 * (cx - cbar)
wzx += v**5 * (S_sig_1 - S_sig_1_bar)
wzx += v**6 * (S_sig_2 - S_sig_2_bar)
wzx //=(x - zz)

wzwx = (zx - zwbar) // (x - zz * w)

f__ = lambda fx, srs: sum(i * j for i, j in zip(fx.coefficients(sparse=False), srs))

a__ = f__(ax, SRS)
b__ = f__(bx, SRS)
c__ = f__(cx, SRS)
z__ = f__(zx, SRS)
t10__ = f__(t10, SRS)
tmid__ = f__(tmid, SRS)
tni__ = f__(tni, SRS)
wz__ = f__(wzx, SRS)
wzwx__ = f__(wzwx, SRS)

```

(continues on next page)

(continued from previous page)

```
pi_snark = [a__, b__, c__, z__, t10__, tmid__, tni__, wz__, wzw__, abar, bbar, cbar, S_
↪sig_1_bar, S_sig_2_bar, rbar, zwbar]
print(pi_snark)
```

Tài liệu tham khảo

- [1] *Under the hood of zkSNARKs — PLONK protocol: Part 1*, URL - <https://medium.com/coinmonks/under-the-hood-of-zksnarks-plonk-protocol-part-1-34bc406d8303>
- [2] *Under the hood of zkSNARKs — PLONK protocol: Part 2*, URL - <https://medium.com/coinmonks/under-the-hood-of-zksnarks-plonk-protocol-part-2-ee00d6accb4d>
- [3] *Under the hood of zkSNARKs — PLONK protocol: Part 3*, URL - <https://medium.com/@cryptofairy/under-the-hood-of-zksnarks-plonk-protocol-part3-821855e49ce6>
- [4] <https://github.com/tarassh/zkSNARK-under-the-hood>

Lời giải cho những vấn đề

4.1 Lời giải các bài tập trong sách tham khảo

4.1.1 Abstract Algebra: Theory and Applications

Lời giải cho quyển sách **Abstract Algebra: Theory and Applications** của Thomas W. Judson [7].

Chương 3. Groups

1 Exercise (Bài 7)

Đặt $S = \mathbb{R} \setminus \{-1\}$ và định nghĩa toán tử hai ngôi trên S là $a \star b = a + b + ab$. Chứng minh rằng (S, \star) là nhóm Abel.

Để chứng minh (S, \star) là nhóm ta chứng minh ba tiên đề của nhóm.

- Giả sử tồn tại phần tử đơn vị e , khi đó $e \star s = s \star e = s$ với mọi $s \in S$. Nghĩa là $e + s + es = s + e + se = s$. Vậy $e + se = 0$ mà $s \neq -1$ nên $e = 0$
- Với $e = 0$, giả sử với mọi $s \in S$ có nghịch đảo s' . Do $s \star s' = s' \star s = e$ nên $s + s' + ss' = s' + s + s's = e = 0$, tức là $s'(1 + s) = -s$. Vậy $s' = \frac{-s}{1 + s}$
- Với mọi $a, b, c \in S$,

$$\begin{aligned} a \star (b \star c) &= a \star (b + c + bc) = a + (b + c + bc) + a(b + c + bc) \\ &= a + b + c + ab + bc + ca + abc \end{aligned}$$

và

$$\begin{aligned} (a \star b) \star c &= (a + b + ab) \star c = a + b + ab + c + c(a + b + bc) \\ &= a + b + c + ab + bc + ca + abc. \end{aligned}$$

Như vậy $a \star (b \star c) = (a \star b) \star c$, do đó \star có tính kết hợp.

Vậy (S, \star) là nhóm.

1 Exercise (Bài 39)

Gọi G là tập các ma trận 2×2 với dạng

$$\begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

với $\theta \in \mathbb{R}$. Chứng minh rằng G là subgroup của $SL_2(\mathbb{R})$.

Với $\theta_1, \theta_2 \in \mathbb{R}$, ta có

$$\begin{aligned} & \begin{pmatrix} \cos \theta_1 & -\sin \theta_1 \\ \sin \theta_1 & \cos \theta_1 \end{pmatrix} \begin{pmatrix} \cos \theta_2 & -\sin \theta_2 \\ \sin \theta_2 & \cos \theta_2 \end{pmatrix} \\ &= \begin{pmatrix} \cos \theta_1 \cos \theta_2 - \sin \theta_1 \sin \theta_2 & -\cos \theta_1 \sin \theta_2 - \sin \theta_1 \cos \theta_2 \\ \sin \theta_1 \cos \theta_2 + \cos \theta_1 \sin \theta_2 & -\sin \theta_1 \sin \theta_2 + \cos \theta_1 \cos \theta_2 \end{pmatrix} \\ &= \begin{pmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) \end{pmatrix}, \end{aligned}$$

suy ra định thức của tích hai ma trận là

$$\det \left(\begin{pmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) \end{pmatrix} \right) = 1 \cdot 1 = 1.$$

Như vậy phép nhân hai ma trận có dạng trên đóng trên $SL_2(\mathbb{R})$.

Phần tử đơn vị là $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ tương ứng với $\theta = 0$.

Phần tử nghịch đảo là $\begin{pmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{pmatrix}$ suy ra từ công thức định thức ban nãy.

Cuối cùng, phép nhân ma trận có tính kết hợp. Như vậy G là subgroup của $SL_2(\mathbb{R})$.

1 Exercise (Bài 47)

Đặt G là nhóm và $g \in G$. Chứng minh rằng

$$Z(G) = \{x \in G : gx = xg \ \forall g \in G\}$$

là subgroup của G . Subgroup này gọi là **center** của G .

Giả sử trong G có hai phần tử là x_1 và x_2 thuộc $Z(G)$. Khi đó

$$x_1g = gx_1 \text{ và } x_2g = gx_2 \text{ với mọi } g \in G.$$

Xét phần tử x_1x_2 , ta có

$$(x_1x_2)g = x_1(x_2g) = x_1(gx_2) = (gx_1)x_2 = g(x_1x_2)$$

với mọi $g \in G$. Do đó $x_1x_2 \in Z(G)$ nên $Z(G)$ là subgroup.

1 Exercise (Bài 49)

Cho ví dụ về nhóm vô hạn mà mọi nhómc con không tầm thường của nó đều vô hạn.

Ví dụ tập \mathbb{Z} và phép cộng số nguyên. Khi đó mọi nhóm con của \mathbb{Z} có dạng $n\mathbb{Z}$ với $n \in \mathbb{Z}$. Ví dụ

- $2\mathbb{Z} = \{\dots, -4, -2, 0, 2, 4, \dots\}$ với phần tử sinh là 2;
- $n\mathbb{Z} = \{\dots, -2n, -n, 0, n, 2n, \dots\}$ với phần tử sinh là n .

Exercise (Bài 54)

Cho H là subgroup của G và

$$C(H) = \{g \in G : gh = hg \ \forall h \in H\}.$$

Chứng minh rằng $C(H)$ là subgroup của G . Subgroup này được gọi là **centralizer** của H trong G .

Gọi g_1 và g_2 thuộc $C(H)$. Khi đó $g_1h = hg_1$ và $g_2h = hg_2$ với mọi $h \in H$.

Xét phần tử g_1g_2 , với mọi $h \in H$ ta có

$$(g_1g_2)h = g_1(g_2h) = g_1(hg_2) = (g_1h)g_2 = (hg_1)g_2 = h(g_1g_2).$$

Như vậy $g_1g_2 \in C(H)$, từ đó $C(H)$ là subgroup của G

Chương 5. Permutation Groups

Exercise (Bài 13)

Đặt $\sigma = \sigma_1 \cdots \sigma_m \in S_n$ là tích của các cycle độc lập. Chứng minh rằng order của σ là LCM của độ dài các cycle $\sigma_1, \dots, \sigma_m$.

Đặt l_i là độ dài cycle σ_i ($i = 1, \dots, m$). Khi đó $\sigma_i^{k_i l_i}$ sẽ ở dạng các cycle độ dài 1 ($k_i \in \mathbb{Z}$).

Từ đó, $\sigma^l = \sigma_1^l \cdots \sigma_m^l = (1) \cdots (n)$ nếu $l = k_1 l_1 = \cdots k_m l_m$. Số l nhỏ nhất thỏa mãn điều kiện này là $\text{lcm}(l_1, \dots, l_m)$ (đpcm).

Exercise (Bài 23)

Nếu σ là chu trình với độ dài lẻ, chứng minh rằng σ^2 cũng là chu trình.

Giả sử $\sigma = (g_1, g_2, \dots, g_{n-1}, g_n)$ với n lẻ.

Khi đó

$$\sigma^2 = (g_1, g_3, \dots, g_n, g_2, g_4, \dots, g_{n-1})$$

cũng là chu trình.

Exercise (Bài 30)

Cho $\tau = (a_1, a_2, \dots, a_k)$ là chu trình độ dài k .

1. Chứng minh rằng với mọi hoán vị σ thì

$$\sigma \tau \sigma^{-1} = (\sigma(a_1), \sigma(a_2), \dots, \sigma(a_k))$$

là chu trình độ dài k .

2. Gọi μ là chu trình độ dài k . Chứng minh rằng tồn tại hoán vị σ sao cho $\sigma\tau\sigma^{-1} = \mu$.

Để chứng minh hai mệnh đề trên ta cần chú ý một số điều.

- Ta thấy rằng bất kì phần tử nào khác a_1, a_2, \dots, a_k thì khi qua τ không đổi, do đó khi đi qua $\sigma\tau\sigma^{-1}$ thì chỉ đi qua $\sigma\sigma^{-1}$ và cũng không đổi. Nói cách khác các phần tử a_1, a_2, \dots, a_k vẫn nằm trong chu trình nên ta có đpcm.
- Từ câu (a), với $\mu = (b_1, b_2, \dots, b_k)$ thì ta chọn σ sao cho $b_i = \sigma(a_i)$.

Chương 6. Cosets

Exercise (Bài 11)

Gọi H là subgroup của nhóm G và giả sử $g_1, g_2 \in G$. Chứng minh các mệnh đề sau là tương đương:

- $g_1H = g_2H$
- $Hg_1^{-1} = Hg_2^{-1}$
- $g_1H \subseteq g_2H$
- $g_2 \in g_1H$
- $g_1^{-1}g_2 \in H$

Từ (1) ra (2): Ta đã biết các coset là rời nhau hoặc trùng nhau, do đó với mọi $g_1h \in g_1H$, tồn tại $g_2h' \in g_2H$ mà $g_1h = g_2h'$, suy ra $(g_1h)^{-1} = (g_2h')^{-1}$ hay $h^{-1}g_1^{-1} = h'^{-1}g_2^{-1}$ (đpcm).

Từ (1) ra (3): Hiển nhiên.

Từ (1) ra (4): Với mọi $g_1h \in g_1H$, tồn tại $g_2h' \in g_2H$ sao cho $g_1h = g_2h'$, hay $g_2 = g_1hh'^{-1}$, đặt $h'' = hh'^{-1}$ thì $h'' \in H$ (H là nhóm con) nên $g_1h'' \in g_1H$, suy ra $g_2 \in g_1H$.

Từ (1) ra (5): Tương tự, ta có $g_1h = g_2h'$, suy ra $hh'^{-1} = g_1^{-1}g_2 \in H$.

Exercise (Bài 16)

Nếu $ghg^{-1} \in H$ với mọi $g \in G$ và $h \in H$, chứng minh rằng right coset trùng với left coset.

Do $ghg^{-1} \in H$ nên tồn tại $h' \in H$ sao cho $ghg^{-1} = h'$. Tương đương $gh = h'g$ với mọi $h \in H$ nên $gH = Hg$. Điều này đúng với mọi $g \in G$ nên các right coset trùng left coset.

Exercise (Bài 17)

Giả sử $[G : H] = 2$. Chứng minh rằng nếu a, b không thuộc H thì $ab \in H$.

Ta biết rằng 2 coset ứng với 2 phần tử g_1, g_2 bất kì là trùng nhau hoặc rời nhau.

Do đó với $eH = H$, ta suy ra 2 coset của G là H và $G \setminus H$.

Vì $a, b \notin H$ nên coset của chúng trùng nhau. Và nghịch đảo của a cũng nằm trong $G \setminus H$ vì nếu nghịch đảo của a nằm trong H thì a cũng phải nằm trong H .

Từ đó suy ra $a^{-1}H = bH$, nghĩa là tồn tại hai phần tử $h_1, h_2 \in H$ sao cho $a^{-1}h_1 = bh_2$, tương đương $h_1h_2^{-1} = ab \in H$ (đpcm).

Exercise (Bài 21)

Gọi G là cyclic group với order n . Chứng minh rằng có đúng $\phi(n)$ phần tử sinh của G .

Gọi g là một phần tử sinh của G . Khi đó g sinh ra tất cả phần tử trong G , hay nói cách khác các phần tử trong G có dạng g^i với $0 \leq i < n$.

Như vậy một phần tử $h = g^i$ cũng là phần tử sinh của G khi và chỉ khi $\gcd(i, n) = 1$ và có $\phi(n)$ số i như vậy (đpcm).

Chương 9. Isomorphism**Exercise (Bài 18)**

Chứng minh rằng subgroup của \mathbb{Q}^* gồm các phần tử có dạng $2^m 3^n$ với $m, n \in \mathbb{Z}$ là internal direct product tới $\mathbb{Z} \times \mathbb{Z}$

Xét ánh xạ $\varphi : \mathbb{Q}^* \rightarrow \mathbb{Z} \times \mathbb{Z}$, $\varphi(2^m 3^n) = (m, n)$.

Hàm này là well-defined vì với m cố định thì mỗi phần tử $2^m 3^n$ chỉ cho ra một phần tử (m, n) . Tương tự với cố định n .

Hàm này là đơn ánh (one-to-one) vì với $m_1 = m_2$ và $n_1 = n_2$ thì $2^{m_1} 3^{n_1} = 2^{m_2} 3^{n_2}$.

Hàm này cũng là toàn ánh vì với mỗi cặp (m, n) ta đều tính được $2^m 3^n$.

Vậy hàm φ là song ánh.

Thêm nữa,

$$\begin{aligned}\varphi(2^{m_1} 3^{n_1} \cdot 2^{m_2} 3^{n_2}) &= \varphi(2^{m_1+m_2} 3^{n_1+n_2}) \\ &= (m_1 + m_2, n_1 + n_2) = (m_1, n_1) + (m_2, n_2) \\ &= \varphi(2^{m_1} 3^{n_1})\varphi(2^{m_2} 3^{n_2}).\end{aligned}$$

Vậy φ là homomorphism, và là song ánh nên là isomorphism.

Exercise (Bài 20)

Chứng minh hoặc bác bỏ: mọi nhóm Abel có order chia hết bởi 3 chứa một subgroup có order là 3.

Gọi order của nhóm Abel là $n = 3k$, và g là phần tử sinh của nhóm Abel đó. Như vậy $g^n = g^{3k} = e$.

Nếu ta chọn $h = g^k$ thì $h^3 = e$, khi đó subgroup được sinh bởi h có order 3 (đpcm).

Exercise (Bài 21)

Chứng minh hoặc bác bỏ: mọi nhóm không phải Abel có order chia hết bởi 6 chứa một subgroup có order 6.

Với S_3 có order là 6 nhưng không có nhóm con nào order 6 (nhóm con chỉ có order 1, 2 hoặc 3) (bác bỏ).

1 Exercise (Bài 22)

Gọi G là group với order 20. Nếu G có các subgroup H và K với order 4 và 5 mà $hk = kh$ với mọi $h \in H$ và $k \in K$, chứng minh rằng

Ta chứng minh $H \cap K = \{e\}$. Giả sử tồn tại phần tử $m \in H \cap K$, khi đó do $m \in H$ nên $mk = km$ với mọi $k \in K$. Tuy nhiên $m \in K$ do đó điều này xảy ra khi và chỉ khi $m = e$.

Như vậy $H \cap K = \{e\}$.

Chương 11. Homomorphism**1 Exercise (Bài 1)**

Chứng minh rằng

$$\det(AB) = \det(A) \det(B)$$

với $A, B \in GL_2(\mathbb{R})$. Điều này chứng tỏ rằng định thức là homomorphism từ $GL_2(\mathbb{R})$ tới \mathbb{R}^* .

Đặt $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ và $B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$. Khi đó

$$AB = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

Như vậy ta có

$$\begin{aligned} \det(AB) &= (a_{11}b_{11} + a_{12}b_{21})(a_{21}b_{12} + a_{22}b_{22}) \\ &\quad - (a_{11}b_{12} + a_{12}b_{22})(a_{21}b_{11} + a_{22}b_{21}) \\ &= \cancel{a_{11}a_{21}b_{11}b_{12}} + a_{11}a_{22}b_{11}b_{22} + a_{12}a_{21}b_{12}b_{21} + \cancel{a_{12}a_{22}b_{21}b_{22}} \\ &\quad - \cancel{a_{11}a_{21}b_{11}b_{12}} - a_{11}a_{22}b_{12}b_{21} - a_{12}a_{21}b_{11}b_{22} - \cancel{a_{12}a_{22}b_{21}b_{22}}. \end{aligned}$$

Tương tự,

$$\begin{aligned} \det(A) \det(B) &= (a_{11}a_{22} - a_{12}a_{21})(b_{11}b_{22} - b_{12}b_{21}) \\ &= a_{11}a_{22}b_{11}b_{22} - a_{11}a_{22}b_{12}b_{21} \\ &\quad - a_{12}a_{21}b_{11}b_{22} + a_{12}a_{21}b_{12}b_{21}. \end{aligned}$$

Như vậy $\det(AB) = \det(A) \det(B)$ và do đó ánh xạ \det từ $GL_2(\mathbb{R})$ tới \mathbb{R}^* là homomorphism.

1 Exercise (Bài 4)

Xét $\phi : \mathbb{Z} \rightarrow \mathbb{Z}$ cho bởi $\phi(n) = 7n$. Chứng minh rằng ϕ là homomorphism. Tìm hạt nhân và ảnh của ϕ .

Ta có

$$\phi(a + b) = 7(a + b) = 7a + 7b = \phi(a) + \phi(b)$$

với mọi $a, b \in \mathbb{Z}$. Do đó ϕ là homomorphism.

Hạt nhân của ϕ là tập hợp các số n để $\phi(n) = 0$, hay $7n = 0$. Như vậy $n = 0$ nên $\ker \phi = \{0\}$.

Ánh của ϕ là tập $\{\dots, -2 \cdot 7, -7, 0, 7, 2 \cdot 7, \dots\}$.

Exercise (Bài 8)

Nếu G là nhóm Abel và $n \in \mathbb{N}$, chứng minh rằng $\phi : G \rightarrow G$ xác định bởi $g \mapsto g^n$ là homomorphism.

Với mọi $g, h \in G$ thì $\phi(gh) = (gh)^n$. Do G là nhóm Abel nên ta có $(gh)^n = g^n h^n = \phi(g)\phi(h)$. Như vậy ϕ là đồng cấu nhóm.

Exercise (Bài 9)

Nếu $\phi : G \rightarrow H$ là homomorphism và G là nhóm Abel, chứng minh rằng $\phi(G)$ cũng là nhóm Abel.

Với mọi $g, h \in G$, do ϕ là homomorphism nên $\phi(gh) = \phi(g)\phi(h)$. Do G là nhóm Abel nên $gh = hg$ với mọi $g, h \in G$, suy ra $\phi(gh) = \phi(hg)$. Tương đương với $\phi(g)\phi(h) = \phi(h)\phi(g)$ nên $\phi(G)$ cũng là nhóm Abel.

Exercise (Bài 10)

Nếu $\phi : G \rightarrow H$ là homomorphism và G là nhóm cyclic, chứng minh rằng $\phi(G)$ cũng là nhóm cyclic.

Tương tự câu 9.

Exercise (Bài 20)

Cho homomorphism $\phi : G \rightarrow H$ và định nghĩa quan hệ \sim trên G theo quy tắc $a, b \in G$ có quan hệ với nhau nếu $\phi(a) = \phi(b)$ và kí hiệu là $a \sim b$. Chứng minh đây là quan hệ tương đương và mô tả cách xây dựng các lớp tương đương.

Do ϕ là ánh xạ nên $\phi(a) = \phi(a)$ với mọi $a \in G$, suy ra \sim có tính phản xạ.

Nếu $a \sim b$ thì $\phi(a) = \phi(b)$. Tương đương với $\phi(b) = \phi(a)$ nên $b \sim a$. Như vậy quan hệ trên có tính đối xứng.

Nếu $a \sim b$ thì $\phi(a) = \phi(b)$, và nếu $b \sim c$ thì $\phi(b) = \phi(c)$. Suy ra $\phi(a) = \phi(b) = \phi(c)$ nên $a \sim c$. Như vậy quan hệ có tính bắc cầu.

Kết luận: quan hệ \sim xác định như trên là quan hệ tương đương.

Để xây dựng các lớp tương đương, đặt $I = \{i_1, i_2, \dots, i_m\}$ là ánh của homomorphism ϕ . Rõ ràng $I \subset H$. Khi đó các lớp tương đương ứng với các phần tử i_1, i_2, \dots, i_m , hay

$$\bar{i}_j = \{g \in G : \phi(g) = i_j\}, \quad 1 \leq i \leq m.$$

4.1.2 Курс высшей математики: Теория вероятностей

Lời giải cho quyển sách [37]. Đây là quyển mình được học trên trường.

Xác suất không liên tục

Bài 1. Có 10 đội được chia ngẫu nhiên làm hai nhóm. Tính xác xuất để hai đội mạnh nhất vào hai nhóm khác nhau? ... cùng một nhóm? ... cùng vào nhóm thứ nhất?

Giải.

- Một đội mạnh có C_2^1 cách chọn, đội còn lại sẽ vào nhóm còn lại. Tiếp theo, chọn 4 đội cho nhóm đầu có C_8^4 cách, 4 đội cho nhóm còn lại có C_4^4 cách. Không gian mẫu là $C_{10}^5 \cdot C_5^5$. Vậy đáp án là $\frac{5}{9}$.
- Hai đội mạnh vào cùng một nhóm, có C_2^1 cách. Chọn 3 đội cho nhóm đó có C_8^3 cách. Nhóm còn lại sẽ có C_5^5 cách. Không gian mẫu là $C_{10}^5 \cdot C_5^5$. Vậy đáp án là $\frac{4}{9}$.
- Hai đội mạnh đều vào nhóm đầu nên chỉ có 1 cách chọn. Chọn 3 đội còn lại của nhóm đầu có C_8^3 cách. Chọn 5 đội cho nhóm còn lại có C_5^5 cách. Không gian mẫu là $C_{10}^5 \cdot C_5^5$. Vậy đáp án là $\frac{2}{9}$.

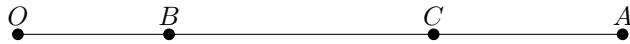
Bài 2. Một bộ bài đầy đủ có 52 lá. Lấy ngẫu nhiên ra ba lá. Tính xác suất để ba lá đó là 3, 7 và át? Tính xác suất để ba lá đó được lấy theo thứ tự trên?

Giải.

- Không gian mẫu là C_{52}^3 . Lấy một trong bốn con 3 có C_4^1 cách, tương tự cho lấy con 7 có C_4^1 cách và lấy con át có C_4^1 cách. Vậy đáp án là $\frac{4 \cdot 4 \cdot 4}{C_{52}^3} = \frac{16}{5525}$.
- Không gian mẫu là A_{52}^3 vì có xét thứ tự. Cách chọn ba lá bài vẫn như trước. Vậy đáp án là $\frac{4 \cdot 4 \cdot 4}{A_{52}^3} = \frac{8}{16575}$.

Bài 3. Trên đoạn thẳng OA độ dài L chọn ngẫu nhiên hai điểm B và C , điều kiện là C nằm bên phải so với B . Tính xác suất để độ dài BC nhỏ hơn độ dài OB .

Giải.



Gọi L là độ dài đoạn OA . Đặt x là độ dài OB và y là độ dài BC . Khi đó các độ dài này phải thỏa mãn các phương trình

$$\begin{cases} y < x \\ 0 < x + y < L \\ x > 0, y > 0 \end{cases}$$

Xác suất trên tương ứng biểu diễn hình học. Khi đó xác suất là tỉ lệ diện tích:

- Không gian mẫu là tam giác vuông nên diện tích là $\frac{L^2}{2}$.
- Vùng giới hạn bởi các đường thẳng $y = 0$, $y = x$ và $x + y = L$ là tam giác với diện tích là $\frac{L \cdot \frac{L}{2}}{2} = \frac{L^2}{4}$.

Đáp án: $\frac{1}{2}$.

Bài 4. Có 10 bilet nằm trên bàn giám thị được đánh số từ 1 tới 10. Tính xác suất để các sinh viên lấy bilet theo thứ tự từ 1 tới 10.

Giải. Không gian mẫu là $10!$ cách lấy 10 bilet theo thứ tự bất kì.

Để lấy 10 bilet theo thứ tự, chỉ có duy nhất một cách lấy lần lượt 1, 2, ...

Đáp án: $\frac{1}{10!}$.

Bài 5. Bốn người vào thang máy ở tầng 1 của tòa nhà 9 tầng. Biết rằng xác suất mỗi người rời khỏi thang máy là như nhau cho các tầng từ 2 tới 9. Tính xác suất để: a) mọi người rời thang máy ở các tầng khác nhau; b) mọi người rời thang máy cao hơn tầng 5; c) ở tầng 3 không ai rời thang máy.

Giải. Không gian mẫu là 8^4 vì mỗi người đều có 8 cách chọn ra các tầng từ 2 tới 9.

- Chọn 4 tầng mà mỗi người sẽ ra, có A_8^4 cách vì thứ tự có thể khác nhau. Đáp án là $\frac{A_8^4}{8^4} = \frac{105}{256}$.
- Tầng cao hơn 5 có 4 phương án 6, 7, 8, 9. Khi đó mỗi người có 4 cách chọn nên bốn người có 4^4 cách chọn. Đáp án là $\frac{4^4}{8^4} = \frac{1}{16}$.
- Do không ai rời thang máy tầng 3 nên mỗi người có 7 cách chọn là 2, 4, 5, 6, 7, 8 và 9. Do đó số cách chọn của bốn người là 7^4 . Đáp án là $\frac{7^4}{8^4} = \frac{2401}{4096}$.

Bài 6. Lấy ngẫu nhiên bốn chữ số và xếp chúng theo vị trí bất kì. Tính xác suất để thu được số có 4 chữ số? Tính xác suất để thu được số có 4 chữ số chia hết cho 5?

Giải. Không gian mẫu là 10^4 vì có 10 chữ số từ 0 tới 9.

- Để tạo thành số có 4 chữ số thì chữ số đầu phải khác 0 nên có 9 cách chọn. Ba vị trí còn lại mỗi vị trí 10 cách chọn. Như vậy số cách chọn là $9 \cdot 10^3$ nên đáp án là $\frac{9}{10}$.
- Tương tự, để tạo thành số có 4 chữ số thì chữ số đầu có 9 cách chọn. Vị trí cuối phải là 0 hoặc 5 để chia hết cho 5. Hai chữ số ở giữa tùy ý nên có 10^2 cách. Như vậy số cách chọn là $9 \cdot 2 \cdot 10^2$, nên đáp án là $\frac{9}{50}$.

Bài 7. Có 20 bilet exam trên bàn giám thị theo thứ tự ngẫu nhiên. 10 sinh viên lần lượt bốc bilet ngẫu nhiên. Tính xác suất để bilet số 1 và 2 không được chọn.

Giải. Không gian mẫu là A_{20}^{10} do các sinh viên bốc lần lượt nên sẽ có thứ tự.

Biến cố bilet số 1 và 2 không được chọn, như vậy sẽ còn 18 trường hợp và 10 sinh viên sẽ bốc ngẫu nhiên 10 bilet trong này. Do đó có A_{18}^{10} cách.

$$\text{Đáp án: } \frac{A_{18}^{10}}{A_{20}^{10}} = \frac{9}{38}.$$

Bài 8. Trong hộp có 6 quả bóng trắng, 4 quả bóng đen và 2 quả bóng đỏ. Lấy ngẫu nhiên 4 quả bóng. Tính xác suất để trong số 4 quả đó chỉ có bóng đen và bóng đỏ.

Giải. Có tất cả $6 + 4 + 2 = 12$ quả bóng. Không gian mẫu là C_{12}^4 .

Bóng đen và bóng đỏ có tất cả $4 + 2 = 6$ quả. Lấy 4 trong số đó có C_6^4 cách.

$$\text{Đáp án: } \frac{C_6^4}{C_{12}^4} = \frac{1}{33}.$$

Bài 9. Có 10 quyển sách, trong đó có 3 quyển màu đỏ, được xếp theo thứ tự ngẫu nhiên trên kệ. Tính xác suất để 3 quyển màu đỏ nằm liền kề nhau.

Giải. Không gian mẫu là $10!$.

Nếu xem 3 quyển đỏ là cùng một khối thì lúc này trên kệ có 8 quyển. Số cách chọn là $8!$. Bên trong khối đó, 3 quyển màu đỏ có thể xếp theo thứ tự bất kì nên có $3!$ hoán vị. Như vậy tổng số cách xếp là $8! \cdot 3!$.

$$\text{Đáp án là } \frac{8! \cdot 3!}{10!} = \frac{1}{15}.$$

Bài 10. Gieo ba con súc sắc. Tính xác suất các biến cố: A - các súc sắc cho các số khác nhau; B - các súc sắc ra cùng số.

Giải. Không gian mẫu là 6^3 .

- Các súc sắc cho số khác nhau nên có A_6^3 cách chọn vì có tính hoán vị. Đáp án là $P(A) = \frac{A_6^3}{6^3} = \frac{5}{9}$.

2. Các súc sắc ra cùng số có 6 trường hợp. Đáp án là $P(B) = \frac{6}{6^3} = \frac{1}{36}$.

Bài 11. Trong tủ có 5 đôi tất. Lấy ngẫu nhiên 2 chiếc tất. Tính xác suất để 2 chiếc tất đó cùng đeo.

Giải. Không gian mẫu C_{10}^2 .

Lấy 2 chiếc cùng đeo nghĩa là lấy một trong 5 đôi. Vậy có 5 cách chọn.

$$\text{Đáp án: } \frac{5}{C_{10}^2} = \frac{1}{9}.$$

Bài 13. Trong nhóm có 4 nam và 4 nữ được chia thành hai nhóm nhỏ, mỗi nhóm 4 người. Tính xác suất để ở mỗi nhóm nhỏ có 2 nam và 2 nữ.

Giải. Do có tất cả 8 người nên để chia thành hai nhóm nhỏ thì ta chọn 4 người cho nhóm đầu, có C_8^4 cách. Nhóm thứ hai có C_4^4 cách. Tuy nhiên có khả năng bị lặp (chọn A-B rồi C-D hoàn toàn giống chọn C-D rồi A-B). Do đó không gian mẫu cần chia $2!$, nên suy ra $|\Omega| = \frac{C_8^4 \cdot C_4^4}{2!} = 35$ cách.

Tương tự, ta chọn 2 nam cho nhóm đầu thì có C_4^2 cách và cho nhóm thứ hai có C_2^2 cách. Như vậy có 6 cách chia 4 nam vào 2 nhóm. Nữ cũng vậy, có 6 cách. Theo bên trên, hai nhóm này có thể bị lặp nên cần phải chia $2!$, suy ra số cách chia mỗi nhóm 2 nam 2 nữ là $\frac{6 \cdot 6}{2!} = 18$.

$$\text{Đáp án: } \frac{18}{35}.$$

Bài 14. Bên trong hình tròn bán kính R chọn ngẫu nhiên một điểm. Tính xác suất để điểm đó: a) nằm bên trong hình vuông nội tiếp; b) nằm bên trong tam giác đều nội tiếp.

Giải. Bài này sử dụng xác suất hình học. Khi đó không gian mẫu là diện tích hình tròn bán kính R , hay $|\Omega| = \pi R^2$.

- a) Biển cố điểm nằm trong hình vuông nội tiếp có xác suất bằng diện tích hình vuông chia diện tích hình tròn. Hình vuông nội tiếp đường tròn bán kính R có đường chéo bằng $2R$ nên cạnh là $R\sqrt{2}$. Đáp án là $\frac{2R^2}{\pi R^2} = \frac{2}{\pi}$.
- b) Tương tự, tam giác nội tiếp thì cạnh nối từ tâm tới đỉnh là R , bằng $2R/3$ độ dài đường cao nên suy ra đường cao là $3R/2$, suy ra cạnh là $R\sqrt{3}$. Diện tích tam giác nội tiếp là $\frac{R\sqrt{3} \cdot 3R/2}{2} = \frac{3\sqrt{3} \cdot R^2}{4}$ nên đáp án là $\frac{3\sqrt{3}}{4\pi}$.

Bài 15. Trong một ngày có hai chuyến tàu cập cảng để dỡ hàng. Chuyến tàu thứ nhất cần 6 tiếng để dỡ hàng, trong khi chuyến tàu thứ hai cần 8 tiếng. Hai chuyến tàu đến cảng không phụ thuộc vào nhau. Tính xác suất để không chuyến tàu nào phải chờ chuyến tàu kia dỡ hàng xong mới được cập cảng.

Giải. Gọi x là thời gian chuyến tàu đầu cập cảng. Do mất 6 giờ để dỡ hàng nên tàu phải cập cảng trước 18 giờ. Như vậy $0 \leq x \leq 18$.

Tương tự, gọi y là thời gian chuyến tàu thứ hai cập cảng, suy ra $0 \leq y \leq 16$.

Đến đây ta có hai trường hợp:

1. Chuyến tàu đầu cập cảng trước. Khi đó chuyến tàu thứ hai phải đến sau khi chuyến tàu đầu dỡ hàng xong. Do đó $y \geq x + 6$.
2. Chuyến tàu thứ hai cập cảng trước. Tương tự, chuyến tàu đầu phải cập cảng sau khi chuyến thứ hai dỡ hàng xong. Do đó $x \geq y + 8$.

Không gian mẫu là diện tích hình chữ nhật giới hạn bởi $x = 0$, $x = 18$, $y = 0$ và $y = 16$.

Không gian biển cõi nằm trong hình giới hạn bởi các đường thẳng $x = 0$, $x = 18$, $y = 0$, $y = 16$, $x + 6 = y$ và $x - 8 = y$.

Đáp án: $\frac{25}{72}$.

Quy tắc cộng và nhân xác suất

Bài 1. Xác suất bóng đèn hoạt động tốt trong khoảng thời gian xác định của mỗi phần tử là 0,8. Một hệ thống có ba phần tử như hình. Tính độ hoạt động tốt trong mỗi phần tử.

Giải. Nào vẽ hình rồi giải sau :))

Bài 2. Trong hộp có 7 quả bóng trắng và 3 quả bóng đen. Lấy ngẫu nhiên lần lượt từng quả đến khi lấy được bóng đen. Tính xác suất đến khi dừng lại lấy được 4 quả bóng nếu: a) lấy có trả lại; b) lấy không trả lại.

Giải. Không gian mẫu gồm các cách lấy ra 4 quả bóng.

- a) Khi lấy có trả lại, ở mỗi lần lấy có 10 cách chọn nên không gian mẫu $|\Omega| = 10^4$. Theo đề bài, nếu lấy có trả lại 4 quả bóng, 3 quả đầu là bóng trắng và có 7^3 cách chọn. Quả bóng cuối màu đen, có 3 cách chọn. Như vậy có $7^3 \cdot 3$ cách chọn. Đáp án là $\frac{1029}{10000}$.
- b) Khi lấy không trả lại, không gian mẫu là số cách lấy 4 quả bóng từ 10 quả bóng có thứ tự nên $|\Omega| = A_{10}^4$. Số cách chọn 4 quả sao cho 3 quả đầu màu trắng là A_7^3 , và quả cuối màu đen là A_3^1 . Đáp án là $\frac{A_7^3 \cdot A_3^1}{A_{10}^4} = \frac{1}{8}$.

Bài 3. Trong một bộ domino có 28 quân lấy ngẫu nhiên 7 quân. Tính xác suất có ít nhất một quân có nút 6.

Giải. Gọi A là biến cố ít nhất một quân có nút 6. Khi đó \bar{A} là biến cố không có quân nào có nút 6.

Không gian mẫu là A_{28}^7 .

Do \bar{A} không chứa các quân có nút 6 nên sẽ có $28 - 7 = 21$ quân (bỏ các cặp 0-6, 1-6, ..., 6-6).

Suy ra $P(\bar{A}) = \frac{A_{21}^7}{A_{28}^7} = \frac{323}{3289}$.

Đáp án: $P(A) = 1 - P(\bar{A}) = \frac{2966}{3289}$.

Bài 4. Tung 4 cục súc sắc. Tính xác suất để chúng ra các mặt khác nhau.

Giải. Đáp án: $\frac{A_6^4}{6^4} = \frac{5}{18}$.

Bài 5. Trong hộp có 5 quả bóng trắng, 7 quả bóng đỏ và 8 quả bóng xanh. Lấy ngẫu nhiên ra 2 quả. Tính xác suất để hai quả đó cùng màu.

Giải. Gọi A - biến cố lấy hai quả bóng cùng màu.

Gọi A_1, A_2, A_3 lần lượt là biến cố lấy hai quả bóng trắng, hai quả bóng đỏ và hai quả bóng xanh.

Khi đó $A = A_1 \cup A_2 \cup A_3$.

Đáp án:

$$P(A) = P(A_1) + P(A_2) + P(A_3) = \frac{C_5^2 + C_7^2 + C_8^2}{C_{5+7+8}^2} = \frac{59}{190}.$$

Bài 6. Hai cung thủ bắn tên đồng thời độc lập nhau. Xác suất mỗi người bắn trúng là 0,2. Tính xác suất để chỉ một người bắn trúng.

Giải. Chọn một trong hai người bắn trúng, có 2 cách.

Nếu một người bắn trúng thì xác suất là 0,2. Người còn lại sẽ bắn trượt với xác suất 0,8.

Đáp án: $2 \cdot 0,2 \cdot 0,8 = 0,32$.

Bài 7. 20 đội bóng tham gia giải đấu. Trong giải đấu có 4 giải thưởng sẽ được trao cho 4 đội xuất sắc nhất. Giả sử 20 đội được chia thành 4 nhóm được đánh số, mỗi nhóm 5 đội. Tính xác suất để mỗi nhóm có một đội đạt giải? Tính xác suất để nhóm đầu không có đội nào đạt giải.

Giải. Số cách chọn 5 đội cho nhóm đầu là C_{20}^5 , cho nhóm thứ hai là C_{15}^5 , cho nhóm thứ ba là C_{10}^5 và nhóm cuối là C_5^5 . Do có xét thứ tự nhóm nên vẫn tính các hoán vị. Ta suy ra

$$|\Omega| = C_{20}^5 \cdot C_{15}^5 \cdot C_{10}^5 \cdot C_5^5.$$

Trong 4 đội đạt giải, chọn một đội cho vào nhóm 1 thì có 4 cách chọn. Tiếp theo chọn 4 đội trong 16 đội không đạt giải vào nhóm 1 thì có C_{16}^4 cách. Tương tự cho các nhóm 2, 3, 4 nên đáp án là

$$P(A) = \frac{4 \cdot C_{16}^4 \cdot 3 \cdot C_{12}^4 \cdot 2 \cdot C_8^4 \cdot 1 \cdot C_4^4}{C_{20}^5 \cdot C_{15}^5 \cdot C_{10}^5 \cdot C_5^5} = \frac{125}{969}$$

Nếu nhóm đầu không có đội nào đạt giải, chọn 5 trong số 16 đội không đạt giải thì có C_{16}^5 cách. Lúc này còn lại 15 đội. Chọn 5 đội cho nhóm thứ 2, 5 đội cho nhóm thứ 3 và 5 đội cho nhóm thứ 4 có $C_{15}^5 \cdot C_{10}^5 \cdot C_5^5$ cách. Đáp án câu (b) là

$$P(B) = \frac{C_{16}^5 \cdot C_{15}^5 \cdot C_{10}^5 \cdot C_5^5}{C_{20}^5 \cdot C_{15}^5 \cdot C_{10}^5 \cdot C_5^5} = \frac{91}{323}$$

Bài 8. Trong hộp đầu có 2 quả bóng trắng và 3 quả bóng đen, trong hộp thứ hai có 1 trắng và 2 xanh, trong hộp thứ ba có 3 trắng và 1 đỏ. Từ mỗi hộp lấy ngẫu nhiên một quả bóng. Tính xác suất các biến số sau: A - { chỉ lấy ra một bóng trắng }; B - { lấy ít nhất một bóng trắng }; C - { lấy ra các màu khác nhau }.

Giải. Không gian mẫu là $|\Omega| = (2+3) \cdot (1+2) \cdot (3+1) = 60$.

Đặt A_1, A_2, A_3 lần lượt là biến cố quả bóng trắng lấy từ hộp 1, 2 và 3. Khi đó $A = A_1 \cup A_2 \cup A_3$.

- Nếu quả bóng trắng lấy từ hộp 1 thì có 2 cách chọn. Ở hộp thứ hai và thứ ba phải lấy ra quả khác màu trắng nên có $2 \cdot 1$ cách chọn. Suy ra $P(A_1) = \frac{2 \cdot 2 \cdot 1}{60} = \frac{1}{15}$.
- Nếu quả bóng trắng lấy từ hộp 2 thì có 1 cách chọn. Ở hộp đầu và hộp thứ ba phải lấy ra quả khác màu trắng nên có $1 \cdot 3 \cdot 1$ cách. Vậy $P(A_2) = \frac{1 \cdot 3 \cdot 1}{60} = \frac{1}{20}$.
- Nếu quả bóng trắng lấy từ hộp 3 thì có 3 cách chọn. Ở hộp đầu và hộp thứ hai lấy ra quả khác màu trắng có $3 \cdot 2$ cách chọn. Vậy $P(A_3) = \frac{3 \cdot 3 \cdot 2}{60} = \frac{3}{10}$.

Như vậy

$$P(A) = P(A_1) + P(A_2) + P(A_3) = \frac{1}{15} + \frac{1}{20} + \frac{3}{10} = \frac{5}{12}.$$

Do B là biến cố lấy ít nhất một bóng trắng nên \bar{B} là biến cố không lấy ra bóng trắng nào. Khi đó ở hộp 1 có 3 cách chọn (đen), hộp 2 có 2 cách chọn (xanh) và hộp 3 có 1 cách chọn (đỏ), suy ra $P(\bar{B}) = \frac{3 \cdot 2 \cdot 1}{60} = \frac{1}{10}$.

Như vậy $P(B) = 1 - P(\bar{B}) = \frac{9}{10}$.

Khi lấy ra ba quả bóng khác màu nhau có các trường hợp theo thứ tự hộp là:

- trắng-xanh-đỏ: có $2 \cdot 2 \cdot 1$ cách chọn

- đen-trắng-đỏ: có $3 \cdot 1 \cdot 1$ cách chọn
- đen-xanh-trắng: có $3 \cdot 2 \cdot 3$ cách chọn
- đen-xanh-đỏ: có $3 \cdot 2 \cdot 1$ cách chọn

Như vậy

$$P(C) = \frac{2 \cdot 2 \cdot 1 + 3 \cdot 1 \cdot 1 + 3 \cdot 2 \cdot 3 + 3 \cdot 2 \cdot 1}{60} = \frac{31}{60}.$$

Bài 9. Một bộ bài gồm 36 lá. Lấy ngẫu nhiên hai lá. Tính xác suất để hai lá đó có màu đỏ.

Giải. Không gian mẫu $|\Omega| = C_{26}^2$.

Lấy hai trong $36/2 = 18$ lá đỏ có C_{18}^2 cách.

$$\text{Đáp án: } \frac{C_{18}^2}{C_{26}^2} = \frac{17}{70}.$$

Bài 10. Trong thùng có 3 bóng đèn hỏng và 7 bóng đèn tốt. Lấy ngẫu nhiên các bóng lần lượt đến khi lấy được 2 bóng tốt thì dừng. Tính xác suất để lấy một nửa số bóng đèn.

Giải. Không gian mẫu là A_{10}^5 .

Trong số bốn bóng đèn đầu sẽ có một bóng tốt và 3 bóng hỏng. Bóng đèn thứ 5 sẽ là bóng tốt. Như vậy ta chọn vị trí cho bóng tốt trong 4 vị trí đầu, có 4 cách chọn. Chọn bóng tốt cho vị trí đó có 7 cách chọn. Chọn 3 bóng hỏng cho 3 vị trí còn lại có thứ tự, có A_3^3 cách.

Chọn bóng đèn tốt cho vị trí thứ 5 có 6 cách chọn.

$$\text{Đáp án: } \frac{4 \cdot 7 \cdot A_3^3 \cdot 6}{A_{10}^5} = \frac{1}{30}.$$

Quy tắc cộng và nhân xác suất (tiếp theo)

Bài 1. Từ một bộ bài lấy ngẫu nhiên lần lượt bốn lá. Tính xác suất để tất cả các lá khác chất nhau? Tính xác suất để tất cả các lá khác số nhau?

Giải. Bộ bài có 36 lá (bài Nga) nên có không gian mẫu $|\Omega| = A_{36}^4$.

Gọi A là biến cố tất cả các lá khác chất nhau.

- Chọn chất cho lá thứ nhất có 4 cách chọn. Chọn lá thứ nhất có 9 cách chọn (trong chất đó).
- Chọn chất cho lá thứ hai có 4 cách chọn. Chọn lá thứ hai có 9 cách chọn (trong chất đó).
- Tương tự cho lá thứ ba và thứ tư.

Như vậy $|\Omega_A| = 4! \cdot 9^4$.

$$\text{Đáp án là } P(A) = \frac{|\Omega_A|}{|\Omega|} = \frac{729}{6545}.$$

Gọi B là biến cố tất cả các lá có số khác nhau.

- Chọn số cho lá thứ nhất có 9 cách chọn. Chọn chất cho lá đó có 4 cách chọn.
- Chọn số cho lá thứ hai có 8 cách chọn. Chọn chất cho lá đó có 4 cách chọn.
- Tương tự cho lá thứ ba và thứ tư.

Như vậy $|\Omega_B| = A_9^4 \cdot 4^4$.

$$\text{Đáp án là } P(B) = \frac{|\Omega_B|}{|\Omega|} = \frac{512}{935}.$$

Bài 2. Trong rổ có 10 quả bóng tennis, 7 quả mới và 3 quả đã qua sử dụng. Lấy ngẫu nhiên 2 quả từ rổ cho trận đấu rồi trả lại. Tới trận đấu thứ hai cũng lấy ngẫu nhiên 2 quả. Tính xác suất để ở trận thứ hai lấy được 2 quả mới.

Giải. Sau khi chơi xong trận đầu thì những quả bóng mới sẽ trở thành bóng đã qua sử dụng.

Đặt B là biến cố trận thứ hai lấy được 2 quả mới.

Đặt A_1 , A_2 và A_3 lần lượt là biến cố trận đấu lấy được 2 quả bóng mới, 1 mới 1 đã qua sử dụng, và 2 quả đã qua sử dụng. Ba biến cố này hợp thành biến cố đầy đủ cho việc lấy 2 quả ở trận đấu.

Theo công thức xác suất toàn phần:

$$P(B) = P(A_1)P(B|A_1) + P(A_2)P(B|A_2) + P(A_3)P(B|A_3).$$

Nếu ở trận đấu lấy 2 quả mới, ta có $P(A_1) = \frac{C_7^2}{C_{10}^2} = \frac{7}{15}$. Sau khi trận đấu kết thúc, số bóng mới là $7 - 2 = 5$

và số bóng đã qua sử dụng là $3 + 2 = 5$ nên $P(B|A_1) = \frac{C_5^2}{C_{10}^2} = \frac{2}{9}$.

Nếu ở trận đấu lấy 1 quả mới và 1 quả đã qua sử dụng, ta có $P(A_2) = \frac{C_7^1 \cdot C_3^1}{C_{10}^2} = \frac{7}{15}$. Sau khi trận đấu kết

thúc, số bóng mới là 6 và số bóng đã qua sử dụng là 4 nên $P(B|A_2) = \frac{C_6^2}{C_{10}^2} = \frac{1}{3}$.

Nếu ở trận đấu lấy 2 quả đã qua sử dụng thì $P(A_3) = \frac{C_3^2}{C_{10}^2} = \frac{1}{15}$. Sau khi trận đấu kết thúc, số bóng mới

vẫn là 7 và số bóng đã qua sử dụng vẫn là 3 nên $P(B|A_3) = \frac{C_7^2}{C_{10}^2} = \frac{7}{15}$.

Đáp án:

$$P(B) = \frac{7}{15} \cdot \frac{2}{9} + \frac{7}{15} \cdot \frac{1}{3} + \frac{1}{15} \cdot \frac{7}{15} = \frac{196}{675} \approx 0,29.$$

Bài 3. Trên mỗi cánh máy bay có 2 động cơ. Xác suất bị lỗi của mỗi động cơ trong một chuyến bay là p . Chuyến bay sẽ thành công nếu ở mỗi cánh có ít nhất một động cơ hoạt động bình thường. Tính xác suất chuyến bay thành công.

Giải. Gọi A là biến cố ở một cánh có ít nhất một động cơ hoạt động, suy ra \bar{A} là biến cố không động cơ nào hoạt động ở cả một cánh.

Như vậy $P(\bar{A}) = p^2$ nên $P(A) = 1 - p^2$. Áp dụng cho hai bên cánh (theo quy tắc nhân) ta có đáp án là $(1 - p^2)^2$.

Bài 4. Sinh viên biết 20 trong 30 câu hỏi. Để qua bài thi cần trả lời đúng 2 câu hỏi bắt buộc hoặc trả lời đúng 1 trong 2 câu bắt buộc cộng thêm 1 câu hỏi phụ. Tính xác suất để sinh viên vượt qua bài thi?

Giải. Ở đây cần hiểu là sinh viên biết 20 câu trong số 30 câu, 10 câu kia là câu hỏi phụ. Khi đó có hai trường hợp:

Trường hợp 1 là sinh viên trả lời đúng 2 câu trong số 20 câu sinh viên biết nên xác suất là $\frac{A_{20}^2}{A_{30}^2}$.

Trường hợp 2 là sinh viên trả lời đúng 1 trong 2 câu bắt buộc (có $A_{20}^2 \cdot 2$ cách chọn) và 1 câu hỏi phụ (10 cách chọn) nên xác suất là $\frac{A_{20}^2 \cdot 2 \cdot 10}{A_{30}^3}$. Tổng số câu sinh viên trả lời là 3.

Đáp án: $\frac{A_{20}^2}{A_{30}^2} + \frac{A_{20}^2 \cdot 2 \cdot 10}{A_{30}^3} = \frac{152}{203}$.

4.1.3 An Introduction to Mathematical Cryptography

Lời giải cho quyển sách [33] của Hoffstein.

Trong phần này, bài giải mình sẽ viết tiếng Việt còn đề bài là tiếng Anh (mình lười nên chép lại từ sách :v).

Chapter 2. Discrete Logarithms and Diffie-Hellman

1 Exercise (Câu 2.3)

Let g be a primitive root of \mathbb{F}_p .

- Suppose that $x = a$ and $x = b$ are both integer solutions to the congruence $g^x \equiv h \pmod{p}$. Prove that $a \equiv b \pmod{p-1}$. Explain why this implies that the map (2.1) on page 64 is well-defined
- Prove that $\log_g(h_1h_2) = \log_g(h_1) + \log_g(h_2)$ for all $h_1, h_2 \in \mathbb{F}_p^*$
- Prove that $\log_g(h^n) = n \log_g(h)$ for all $h \in \mathbb{F}_p^*$ and $n \in \mathbb{Z}$

Các tính chất cơ bản của hàm Euler.

- Cả a and b là nghiệm của đồng dư $g^x \equiv h \pmod{p}$ nên $g^a \equiv h \pmod{p}$ và $g^b \equiv h \pmod{p}$.

Suy ra ta có $g^{-b} \equiv h^{-1} \pmod{p}$.

Ta nhân kết quả với đồng dư đầu thì được $g^a g^{-b} \equiv h h^{-1} \equiv 1 \pmod{p}$, hay $g^{a-b} \equiv 1 \pmod{p}$.

Do g là primitive root of \mathbb{F}_p tên ta có $\phi(p) \mid (a-b)$, tương đương với $(p-1) \mid (a-b)$.

Như vậy $a-b \equiv 0 \pmod{p-1}$ hay $a \equiv b \pmod{p-1}$ (đpcm).

- Giả sử $h_1 \equiv g^{x_1} \pmod{p}$ và $h_2 \equiv g^{x_2} \pmod{p}$.

Suy ra $x_1 = \log_g h_1$ và $x_2 = \log_g h_2$ (1).

Do $h_1h_2 \equiv g^{x_1+x_2} \pmod{p}$ nên $x_1 + x_2 = \log_g(h_1h_2)$ (2).

Từ (1) và (2), $\log_g h_1 + \log_g h_2 = \log_g(h_1h_2)$.

- tương tự (b).

1 Exercise (Câu 2.5)

Let p be an odd prime and let g be a primitive root modulo p .

Prove that a has a square root modulo p if and only if its discrete logarithm $\log_g(a)$ modulo $p-1$ is even.

Ta có $g^{p-1} \equiv 1 \pmod{p}$ do g là primitive root modulo p .

Điều kiện đủ. Nếu a là số chính phương modulo p thì tồn tại số b sao cho $b \equiv a^2 \pmod{p}$, suy ra

$$\log_g a = \log_g(b^2) = 2 \log_g b \pmod{p-1},$$

Như vậy $\log_g a$ chẵn.

Điều kiện cần. Nếu $\log_g a$ modulo $p-1$ chẵn.

Điều này xảy ra khi

$$\log_g a = 2 \log_g b \pmod{p-1}$$

với số $b \in \mathbb{F}_p$ nào đó, suy ra

$$\log_g a = \log_g(b^2) \pmod{p-1},$$

hay $a \equiv b^2 \pmod{p-1}$.

Như vậy a có căn bậc hai modulo $p-1$.

1 Exercise (Câu 2.10)

The exercise describes a public key cryptosystem that requires Bob and Alice to exchange several messages. We illustrate the system with an example.

Bob and Alice fix a publicly known prime $p = 32611$, and all of other numbers used are private.

Alice takes her message $m = 11111$, chooses a random exponent $a = 3589$, and sends the number $u = m^a \pmod{p} = 15950$ to Bob.

Bob chooses a random exponent $b = 4037$ and sends $v = u^b \pmod{p} = 15422$ back to Alice.

Alice then computes $w = v^{15619} \equiv 27257 \pmod{32611}$ and sends $w = 27257$ to Bob.

Finally, Bob computes $w^{31883} \pmod{32611}$ and recovers the value 11111 of Alice's message.

- (a) Explain why this algorithm works. In particular, Alice uses the numbers $a = 3589$ and 15619 as exponents. How are they related? Similarly, how are Bob's exponents $b = 4037$ and 31883 related?
- (b) Formulate a general version of this cryptosystem, i.e., using variables, and show how it works in general.
- (c) What is the disadvantage of this cryptosystem over Elgamal? (*Hint.* How many times must Alice and Bob exchange data?)
- (d) Are there any advantages of this cryptosystem over Elgamal? In particular, can Eve break it if she can solve the discrete logarithm problem? Can Eve break it if she can solve the Diffie-Hellman problem?

(a) Ta có

$$3589 \cdot 15619 \equiv 4073 \cdot 31883 \equiv 1 \pmod{p-1}.$$

(b) Alice chọn a và a' sao cho $aa' \equiv 1 \pmod{p-1}$.

Tương tự, Bob chọn b và b' sao cho $bb' \equiv 1 \pmod{p-1}$.

Do đó ta có $aa' = k(p-1) + 1$ và $bb' = l(p-1) + 1$.

$$\begin{aligned} &\Rightarrow v \equiv u^b \equiv (m^a)^b \equiv m^{ab} \pmod{p} \\ &\Rightarrow w \equiv v^{a'} \equiv (m^{ab})^{a'} \equiv m^{aa'b} \pmod{p} \\ &\Rightarrow w^{b'} \equiv m^{aa'bb'} \equiv m^{[k(p-1)+1]x[l(p-1)+1]} \equiv m^{D(p-1)+1} \equiv m \pmod{p}. \end{aligned}$$

1 Exercise (Câu 2.11)

The group S_3 consists of the following six distinct elements

$$e, \sigma, \sigma^2, \tau, \sigma\tau, \sigma^2\tau$$

where e is the identity element and multiplication is performed using the rules

$$\sigma^3 = e, \quad \tau^2 = e, \quad \tau\sigma = \sigma^2\tau$$

Compute the following values in the group S_3 :

- (a) $\tau\sigma^2$
- (b) $\tau(\sigma\tau)$
- (c) $(\sigma\tau)(\sigma\tau)$
- (d) $(\sigma\tau)(\sigma^2\tau)$

Is S_3 a commutative group?

(a)

$$\tau\sigma^2 = \tau\sigma\sigma = \sigma^2\tau\sigma = \sigma\sigma^2\tau = \sigma^3\tau = e\tau = \tau.$$

(b)

$$\tau(\sigma\tau) = (\tau\sigma)\tau = \sigma^2\tau\tau = \sigma^2\tau^2 = \sigma^2e = \sigma^2.$$

(c)

$$(\sigma\tau)(\sigma\tau) = \sigma(\tau\sigma)\tau = \sigma(\sigma^2\tau)\tau = \sigma^3\tau^2 = ee = e.$$

(d)

$$(\sigma\tau)(\sigma^2\tau) = (\sigma\tau)(\tau\sigma) = \sigma\tau^2\sigma = \sigma e \sigma = \sigma^2.$$

S_3 không là nhóm giao hoán vì:

$$\sigma\tau = \sigma\tau, \quad \tau\sigma = \sigma^2\tau,$$

đây là hai phần tử phân biệt trong S_3 .

Exercise (Câu 2.12)

Let G be a group, let $d \geq 1$ be an integer, and define a subset of G by

$$G[d] = \{g \in G : g^d = e\}$$

- (a) Prove that if g is in $G[d]$, then g^{-1} is in $G[d]$
- (b) Suppose that G is commutative. Prove that if g_1 and g_2 are in $G[d]$, then their product $g_1 \star g_2$ is in $G[d]$
- (c) Deduce that if G is commutative, then $G[d]$ is a group.
- (d) Show by an example that if G is not a commutative group, then $G[d]$ need not be a group. (Hint. Use Exercise 2.11.)

(a) Vì $g \star g^{-1} = e$ nên

$$\begin{aligned} g \star e \star g^{-1} &= e \\ \Rightarrow g \star g \star g^{-1} \star g^{-1} &= e \\ \Rightarrow g^2 \star (g^{-1})^2 &= e. \end{aligned}$$

Thực hiện thêm $d - 2$ lần nữa và ta nhận được

$$\begin{aligned} g^d \star (g^{-1})^d &= e \\ \Rightarrow e \star (g^{-1})^2 &= e \\ \Rightarrow (g^{-1})^2 &= e \\ \Rightarrow g^{-1} &\in G[d] \end{aligned}$$

(b) Ta có $g_1^d = e$ and $g_2^d = e$.

Do G là nhóm hoán vị nên $g_1^d \star g_2^d = (g_1 \star g_2)^d$, suy ra $(g_1 \star g_2)^d = e \star e = e$.

Như vậy $g_1 \star g_2 \in G[d]$.

(c) Từ câu (b), ta có với mọi $g_1, g_2 \in G[d]$, thì $g_1 \star g_2 \in G[d]$.

Do e là phần tử đơn vị của G nên cũng là phần tử đơn vị của $G[d]$.

Từ câu (a) ta chứng minh được sự tồn tại của phần tử nghịch đảo.

Với $a, b, c \in G[d]$ thì $a^d = b^d = c^d = e$.

Ta có $b^d \star c^d = (b \star c)^d$ do tính giao hoán, suy ra

$$\begin{aligned} a^d \star (b^d \star c^d) &= a^d \star (b \star c)^d = (a \star b \star c)^d \\ &= (a \star b)^d \star c^d = (a^d \star b^d) \star c^d. \end{aligned}$$

Như vậy ta chứng minh được tính kết hợp. Và từ đó $G[d]$ là nhóm.

(d) Sử dụng kết quả câu 2.11

$$S_3[2] = \{\tau, \sigma\tau, \sigma^2, \tau, e\}.$$

Vì

$$(\sigma\tau)\tau = \sigma\tau^2 = \sigma \notin S_3[2]$$

nên $S_3[2]$ không là nhóm.

Exercise (Câu 2.13)

Let G and H be groups. A function $\phi : G \rightarrow H$ is called a (*group*) *homomorphism* if it satisfies

$$\phi(g_1 \star g_2) = \phi(g_1) \star \phi(g_2) \quad \text{for all } g_1, g_2 \in G$$

(Note that the product $g_1 \star g_2$ uses the group law in the group G , while the product $\phi(g_1) \star \phi(g_2)$ uses the group law in the group H .)

(a) Let e_G be the identity element of G , let e_H be identity element of H , and the $g \in G$. Prove that

$$\phi(e_G) = e_H \quad \text{and} \quad \phi(g^{-1}) = \phi(g)^{-1}$$

(b) Let G be a commutative group. Prove that the map $\phi : G \rightarrow G$ defined by $\phi(g) = g^2$ is a homomorphism. Give an example of a noncommutative group for which this map is not a homomorphism.

(c) Same question as (b) for the map $\phi(g) = g^{-1}$.

(a) Với mọi $g \in G$ thì $g = g \star e = e \star g$. Suy ra

$$\begin{aligned}\phi(g) &= \phi(g \star e_G) = \phi(e_G \star g) \\ \iff \phi(g) &= \phi(g) \star \phi(e_G) = \phi(e_G) \star \phi(g)\end{aligned}$$

Do $\phi(g) \in H$, $\phi(e_G)$ là phần tử đơn vị của H , nói cách khác là $\phi(e_G) = e_H$.

Trong nhóm G , $g \star g^{-1} = e_G$. Suy ra

$$\begin{aligned}\phi(g \star g^{-1}) &= \phi(e_G) \\ \iff \phi(g) \star \phi(g^{-1}) &= \phi(e_G) \\ \iff \phi(g) \star \phi(g^{-1}) &= e_H \\ \iff \phi(g^{-1}) &= \phi(g)^{-1}\end{aligned}$$

(b) $\phi : G \rightarrow G$, $\phi(g) = g^2$. Với mọi $g_1, g_2 \in G$, do G là nhóm giao hoán nên

$$\phi(g_1 \star g_2) = (g_1 \star g_2)^2 = g_1^2 \star g_2^2$$

Ta có $g_1^2 \star g_2^2 = \phi(g_1) \star \phi(g_2)$ nên $\phi(g_1 \star g_2) = \phi(g_1) \star \phi(g_2)$. Như vậy G là homomorphism.

Xét nhóm ở Câu 2.11 và ánh xạ $\phi : G \rightarrow G$, $\phi(g) = g^2$.

Khi đó

$$\begin{aligned}\phi(e) &= e^2 = e, \quad \phi(\sigma) = \sigma^2, \\ \phi(\tau) &= \tau^2 = e, \quad \phi(\sigma\tau) = (\sigma\tau)^2 = e.\end{aligned}$$

Vì

$$\phi(\sigma\tau) = e \neq \sigma^2 = \phi(\sigma)\phi(\tau),$$

nên G không là homomorphism.

(c) $\phi : G \rightarrow G$, $\phi(g) = g^{-1}$.

Với mọi $g_1, g_2 \in G$ thì $g_1 g_1^{-1} = e$ và $g_2 g_2^{-1} = e$

Do đó $g_1 g_1^{-1} g_2 g_2^{-1} = e$, mà G là nhóm hoán vị nên $(g_1 g_2)(g_1^{-1} g_2^{-1}) = e$, tương đương với $g_1^{-1} g_2^{-1} = (g_1 g_2)^{-1}$. Như vậy

$$\phi(g_1 g_2) = (g_1 g_2)^{-1} = g_1^{-1} g_2^{-1} = \phi(g_1)\phi(g_2)$$

Kết luận: G là homomorphism.

Xét nhóm ở Câu 2.11 và ánh xạ $\phi : G \rightarrow G$, $\phi(g) = g^{-1}$. Ta có

$$\sigma\sigma^2 = e = \sigma^2\sigma = e, \quad \tau^2 = e, \quad (\sigma\tau)^2 = e, \quad (\sigma^2\tau)^2 = e,$$

suy ra

$$\phi(\sigma\tau) = \sigma\tau, \quad \phi(\sigma) = \sigma^2, \quad \phi(\tau) = \tau$$

Vì $\phi(\sigma\tau) = \sigma\tau \neq \sigma^2\tau = \phi(\sigma)\phi(\tau)$ nên G không là homomorphism.

1 Exercise (Câu 2.14)

Prove that each of the following maps is a group homomorphism.

(a) The map $\phi : \mathbb{Z} \rightarrow \mathbb{Z}/N\mathbb{Z}$ that sends $a \in \mathbb{Z}$ to $a \bmod N$ in $\mathbb{Z}/N\mathbb{Z}$.

(b) The map $\phi : \mathbb{R}^* \rightarrow GL_2(\mathbb{R})$ defined by $\phi(a) = \begin{pmatrix} a & 0 \\ 0 & a^{-1} \end{pmatrix}$.

(c) The discrete logarithm map $\log_g : \mathbb{F}_p^* \rightarrow \mathbb{Z}/(p-1)\mathbb{Z}$, where g is a primitive root modulo p .

(a) Với mọi $a, b \in \mathbb{Z}$,

$$\begin{aligned}\phi(ab) &= (ab) \pmod{N} \\ &= (a \pmod{N})(b \pmod{N}) \pmod{N} \\ &= \phi(a)\phi(b)\end{aligned}$$

Do đó ϕ là homomorphism.

(b) Với mọi $a, b \in \mathbb{R}^*$ thì

$$\phi(ab) = \begin{pmatrix} ab & 0 \\ 0 & (ab)^{-1} \end{pmatrix}$$

Ta có

$$\phi(a)\phi(b) = \begin{pmatrix} a & 0 \\ 0 & a^{-1} \end{pmatrix} \begin{pmatrix} b & 0 \\ 0 & b^{-1} \end{pmatrix} = \begin{pmatrix} ab & 0 \\ 0 & a^{-1}b^{-1} \end{pmatrix}$$

Trong \mathbb{R}^* ta có tính chất $(ab)^{-1} = a^{-1}b^{-1}$, do đó $\phi(ab) = \phi(a)\phi(b)$. Suy ra ϕ là homomorphism.

(c) Ta chọn ánh xạ $\phi(a) = x$ thỏa mãn $g^x \equiv a \pmod{p}$.

Khi đó, với mọi $a, b \in \mathbb{F}_p^*$, $\phi(a) = x$, hay $g^x \equiv a \pmod{p}$, và $\phi(b) = y$, hay $g^y \equiv b \pmod{p}$.

Ta có $\phi(a)\phi(b) = x + y$ vì $x, y \in \mathbb{Z}/(p-1)\mathbb{Z}$, đây là phép cộng trong modulo $p-1$.

Ta lại có $g^{x+y} \equiv ab \pmod{p}$, suy ra $\phi(ab) = x + y$. Như vậy $\phi(a)\phi(b) = \phi(ab)$ và ϕ là homomorphism.

Exercise (Câu 2.15)

- (a) Prove that $GL_2(\mathbb{F}_p)$ is a group.
- (b) Show that $GL_2(\mathbb{F}_p)$ is a noncommutative group for every prime p .
- (c) Describe $GL_2(\mathbb{F}_p)$ completely. That is, list its elements and describe the multiplication table.
- (d) How many elements are there in the group $GL_2(\mathbb{F}_p)$?
- (e) How many elements are there in the group $GL_n(\mathbb{F}_p)$?

(a) Nếu A và B là hai ma trận thuộc $GL_2(\mathbb{F}_p)$ thì AB cũng thuộc $GL_2(\mathbb{F}_p)$ do $\det(AB) = \det(A) \cdot \det(B)$ khác 0.

Phần tử đơn vị là $E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

Với mọi $A \in GL_2(\mathbb{F}_p)$, do $\det A \neq 0$ nên luôn tồn tại nghịch đảo của A trong $GL_2(\mathbb{F}_p)$.

Với mọi $A, B, C \in GL_2(\mathbb{F}_p)$ ta đều có $(AB)C = A(BC)$ vì phép nhân ma trận có tính kết hợp.

Kết luận: $GL_2(\mathbb{F}_p)$ là nhom.

Giả sử ta có $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ và $B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$ với $A, B \in GL_2(\mathbb{F}_p)$.

Phần tử ở hàng 1 và cột 1 của tích AB là $(a_{11}b_{11} + a_{12}b_{21}) \pmod{p}$.

Phần tử ở hàng 1 và cột 1 của tích BA là $(b_{11}a_{11} + b_{12}a_{21}) \pmod{p}$.

Nếu ta chọn $a_{12} \not\equiv b_{21}^{-1}a_{21}b_{21} \pmod{p}$ thì $AB \neq BA$, do đó nhom không có tính giao hoán.

(c) Ta liệt kê tất cả ma trận:

$$A_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

$$A_4 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad A_5 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad A_6 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

Bảng phép nhân sẽ như sau:

	A_1	A_2	A_3	A_4	A_5	A_6
A_1	A_3	A_5	A_1	A_6	A_2	A_4
A_2	A_4	A_6	A_2	A_5	A_1	A_3
A_3	A_1	A_2	A_3	A_4	A_5	A_6
A_4	A_2	A_1	A_4	A_3	A_6	A_5
A_5	A_6	A_4	A_5	A_2	A_3	A_1
A_6	A_5	A_3	A_6	A_1	A_4	A_2

(d) Hàng đầu \mathbf{u}_1 là vector bất kì thuộc \mathbb{F}_p^2 ngoại trừ $(0, 0)$. Do đó ta có $p^2 - 1$ cách chọn.

Hàng thứ hai \mathbf{u}_2 là vector bất kì ngoại trừ một bội của hàng đầu. Do đó ta có $p^2 - p$ cách (loại các cách chọn $i \cdot \mathbf{u}_1$ với $i = 0, 1, \dots, p - 1$).

Kết luận: có $(p^2 - 1)(p^2 - p)$ phần tử trong $GL_2(\mathbb{F}_p)$.

(e) Tương tự (d), ta chọn hàng đầu \mathbf{u}_1 là bất kì vector nào ngoại trừ $(0, 0)$. Ta có $p^n - 1$ cách chọn.

Hàng thứ hai \mathbf{u}_2 là bất kì vector nào ngoại trừ bội của hàng đầu, nghĩa là loại các tổ hợp $i \cdot \mathbf{u}_1$ với $i = 0, 1, \dots, p - 1$. Ta có $p^n - p$ cách chọn.

Hàng thứ ba \mathbf{u}_3 là bất kì vector nào ngoại trừ các tổ hợp tuyến tính của \mathbf{u}_1 và \mathbf{u}_2 . Số lượng tổ hợp tuyến tính $a_1 \cdot \mathbf{u}_1 + a_2 \cdot \mathbf{u}_2$ chính là số lượng cặp (a_1, a_2) và ta có p^2 trường hợp (vì $a_1, a_2 \in \mathbb{F}_p$). Như vậy hàng thứ ba có $p^n - p^2$ cách chọn.

Tổng quát, vector thứ n (hàng thứ n) là bất kì vector nào ngoại trừ tổ hợp tuyến tính của các vector trước đó $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{n-1}$. Như vậy ta có $p^n - p^{n-1}$ cách chọn.

Kết luận: có tất cả $(p^n - 1)(p^n - p) \cdots (p^n - p^{n-1})$ phần tử trong $GL_n(\mathbb{F}_p)$.

Exercise (Câu 2.18)

Solve each of the following simultaneous systems of congruences (or explain why no solutions exists).

- (a) $x \equiv 3 \pmod{7}$ and $x \equiv 4 \pmod{9}$
- (b) $x \equiv 137 \pmod{423}$ and $x \equiv 87 \pmod{191}$
- (d) $x \equiv 5 \pmod{9}$, $x \equiv 6 \pmod{10}$ and $x \equiv 7 \pmod{11}$
- (e) $x \equiv 37 \pmod{43}$, $x \equiv 22 \pmod{49}$ and $x \equiv 18 \pmod{71}$

(a) Vì $N = 7 \cdot 9 = 63$, đặt $T_1 = 63/7 = 9$, $T_1^{-1} \pmod{7} = 4$, $T_2 = 63/9 = 7$, $T_2^{-1} \pmod{9} = 4$.

$$\Rightarrow x \equiv 3 \cdot 9 \cdot 4 + 4 \cdot 7 \cdot 4 \equiv 220 \equiv 31 \pmod{63}$$

(b) Vì $N = 423 \cdot 191 = 90793$, đặt $T_1 = N/423 = 191$, $T_1^{-1} \pmod{423} = 392$, $T_2 = N/191 = 423$, $T_2^{-1} \pmod{191} = 14$.

$$\Rightarrow x \equiv 137 \cdot 191 \cdot 392 + 87 \cdot 423 \cdot 14 \equiv 27209 \pmod{N}$$

(c) Không thể tính vì $\gcd(451, 697) = 41 \neq 1$.

(d) Vì $N = 9 \cdot 10 \cdot 11 = 990$, đặt $T_1 = N/9 = 110$, $T_1^{-1} \pmod{9} = 5$, $T_2 = N/10 = 99$, $T_2^{-1} \pmod{10} = 9$, $T_3 = N/11 = 90$, $T_3^{-1} \pmod{11} = 6$

$$\Rightarrow x \equiv 5 \cdot 110 \cdot 5 + 6 \cdot 99 \cdot 9 + 7 \cdot 90 \cdot 6 \equiv 986 \pmod{N}$$

- (e) Vì $N = 43 \cdot 49 \cdot 71 = 149597$, đặt $T_1 = N/43 = 3479$, $T_1^{-1} \pmod{43} = 32$, $T_2 = N/49 = 3053$, $T_2^{-1} \pmod{49} = 36$, $T_3 = N/71 = 2107$, $T_3^{-1} \pmod{71} = 37$

$$\Rightarrow x \equiv 37 \cdot 3479 \cdot 32 + 22 \cdot 3053 \cdot 36 + 18 \cdot 2107 \cdot 37 \equiv 11733 \pmod{N}$$

Exercise (Câu 2.19)

Solve the 1700-year-old Chinese remainder problem from the *Sun Tzu Suan Ching* stated on page 84.

$$\begin{cases} x \equiv 2 \pmod{3} \\ x \equiv 3 \pmod{5} \\ x \equiv 2 \pmod{7} \end{cases} .$$

Đáp án: $x \equiv 23 \pmod{105}$.

Exercise (Câu 2.21.)

- (a) Let a, b, c be positive integers and suppose that

$$a | c, \quad b | c, \quad \text{and} \quad \gcd(a, b) = 1$$

Prove that $ab | c$

- (b) Let $x = c$ and $x = c'$ be two solutions to the system of simultaneous congruences in the Chinese remainder theorem. Prove that

$$c \equiv c' \pmod{m_1 m_2 \dots m_k}$$

- (a) Do $a | c$ nên tồn tại $k \in \mathbb{Z}$ sao cho $c = ka$.

Do $b | c$ nên tồn tại $l \in \mathbb{Z}$ sao cho $c = lb$.

Như vậy $ka = lb$.

Tuy nhiên do $\gcd(a, b) = 1$ nên $a | l$, hay $l = ma$ với $m \in \mathbb{Z}$

Như vậy $c = lb = lma$, hay $ab | c$.

- (b) Nếu $c \equiv c' (\equiv a_i) \pmod{m_i}$ thì $c \equiv c' \pmod{m_1 m_2 \dots m_k}$.

Exercise (Câu 2.24)

Let p be an odd prime, let a be an integer that is not divisible by p , and let b is a square root of a modulo p . This exercise investigates the square root of a modulo powers of p

- (a) Prove that for some choice of k , the number $b + kp$ is a square root of a modulo p^2 , i.e., $(b + kp)^2 \equiv a \pmod{p^2}$
- (b) The number $b = 537$ is a square root of $a = 476$ modulo the prime $p = 1291$. Use the idea in (a) to compute a square root of 476 modulo p^2
- (c) Suppose that b is a square root of a modulo p^n . Prove that for some choice of j , the number $b + jp^n$ is a square root of a modulo p^{n+1}

- (d) Explain why (c) implies the following statements: If p is an odd prime and if a has a square root modulo p , then a has a square root modulo p^n for every power of p . Is this true if $p = 2$?
- (e) Use the method in (c) to compute the square root of 3 modulo 13^3 , given that $9^2 \equiv 3 \pmod{13}$

(a) Đặt $f(b_n) = b_n^2 - a \pmod{p^n}$ với $b_1 = b$. Suy ra $f(b_1) = b^2 - a \equiv 0 \pmod{p}$.

Ta cần tìm b_2 thỏa $f(b_2) = b_2^2 - a \equiv 0 \pmod{p^2}$.

Nói cách khác, $b_2 = b_1 + kp$ nên suy ra

$$f(b_1 + kp) = (b_1 + kp)^2 - a = b_1^2 + 2b_1kp + (kp)^2 - a \equiv 0 \pmod{p^2}$$

Tương đương với

$$2b_1k \equiv -(b_1^2 - a) \pmod{p^2},$$

vì $b_1^2 - a \equiv 0 \pmod{p}$.

Do $2b_1 \not\equiv 0 \pmod{p^2}$ nên tồn tại k thỏa mãn đẳng thức.

(b) Ta có công thức

$$k = -(b^2 - a)/p \times (2b)^{-1} \pmod{p^2}$$

(c) Ta chứng minh theo quy nạp với mọi $n \geq 1$, tồn tại $b_n \in \mathbb{Z}$ sao cho

$$\begin{aligned} f(b_n) &= b_n^2 - a \equiv 0 \pmod{p^n} \\ b_n &\equiv b \pmod{p^n} \end{aligned}$$

Trường hợp $n = 1$ là điều kiện ban đầu với $b_1 = b$. Giả sử mệnh đề đúng tới n , nghĩa là:

$$\begin{aligned} f(b_n) &= b_n^2 - a \equiv 0 \pmod{p^n} \\ b_n &\equiv b \pmod{p^n} \end{aligned}$$

Xét b_{n+1}

$$f(b_{n+1}) = b_{n+1}^2 - a \equiv 0 \pmod{p^{n+1}}.$$

Ta viết $b_{n+1} = b_n + p^n t_n$.

$$\begin{aligned} \Rightarrow f(b_{n+1}) &= b_n^2 + 2b_n p^n t_n + p^{2n} t_n^2 - a \equiv 0 \pmod{p^{n+1}} \\ \Rightarrow b_n^2 + 2b_n p^n t_n - a &\equiv 0 \pmod{p^{n+1}} \quad (\text{vì } 2n \geq n+1) \\ \Rightarrow 2b_n t_n &\equiv -(b_n^2 - a)/p^n \pmod{p^{n+1}} \quad \text{từ (2)} \end{aligned}$$

Nghiệm t_n tồn tại vì ta đã giả sử $2b_n \not\equiv 0 \pmod{p^n}$. Như vậy

$$f(b_{n+1}) \equiv 0 \pmod{p^{n+1}}, \quad \text{và} \quad b_{n+1} \equiv b_n \pmod{p^n}.$$

Chứng minh này dùng cho $b + jp^n$ modulo p^n , không phải cho p^{n+1} .

(d) Sử dụng quy nạp. Đặc biệt, nếu $p = 2$ thì mọi số nguyên đều thỏa.

Exercise (Câu 2.31)

Let R and S be rings. A functions $\phi : R \rightarrow S$ is called a (*ring*) *homomorphism* if it satisfies

$$\phi(a + b) = \phi(a) + \phi(b) \quad \text{and} \quad \phi(a * b) = \phi(a) * \phi(b)$$

for all $a, b \in R$.

- (a) Let 0_R , 0_S , 1_R and 1_S denote the additive and multiplicative identities of R and S , respectively. Prove that

$$\phi(0_R) = 0_S, \phi(1_R) = 1_S, \phi(-a) = -\phi(a), \phi(a^{-1}) = \phi(a)^{-1}$$

where the last equality holds for those $a \in R$ that have a multiplicative inverse.

- (b) Let p be a prime, and let R be a ring with the property that $pa = 0$ for every $a \in R$. (Here pa means to add a to itself p times.) Prove that the map

$$\phi : R \rightarrow R, \quad \phi(a) = a^p$$

is a ring homomorphism. It is called the *Frobenius homomorphism*.

Điều kiện để bài là $\phi(a + b) = \phi(a) + \phi(b)$ và $\phi(a \star b) = \phi(a) \star \phi(b)$ với mọi $a, b \in R$.

- (a) Trong R , với mọi $a \in R$ ta có

$$a + 0_R = 0_R + a = a.$$

Suy ra

$$\phi(a) = \phi(a + 0_R) = \phi(0_R + a),$$

hay

$$\phi(a) = \phi(a) + \phi(0_R) = \phi(0_R) + \phi(a).$$

Đặt $\phi(a) = b \in S$. Khi đó

$$b = b + \phi(0_R) = \phi(0_R) + b$$

nên suy ra $\phi(0_R) = 0_S$.

Trong R , với mọi $a \in R$ thì

$$a \star 1_R = 1_R \star a = a.$$

Khi đó

$$\phi(a \star 1_R) = \phi(1_R \star a) = \phi(a)$$

nên suy ra

$$\phi(a) \star \phi(1_R) = \phi(1_R) \star \phi(a) = \phi(a),$$

hay $\phi(1_R) = 1_S$.

Với $\phi(-a) = -\phi(a)$, trong R ta có

$$a + (-a) = (-a) + a = 0_R,$$

suy ra

$$\phi(a + (-a)) = \phi((-a) + a) = \phi(0_R),$$

tương đương với

$$\phi(a) + \phi(-a) = \phi(-a) + \phi(a) = \phi(0_R) = 0_S.$$

Như vậy $\phi(-a) = -\phi(a)$.

Với $\phi(a^{-1}) = \phi(a)^{-1}$, trong R ta có

$$a \star a^{-1} = a^{-1} \star a = 1_R,$$

suy ra

$$\phi(a \star a^{-1}) = \phi(a^{-1} \star a) = \phi(1_R),$$

tương đương với

$$\phi(a) \star \phi(a^{-1}) = \phi(a^{-1}) \star \phi(a) = \phi(1_R) = 1_S.$$

Như vậy $\phi(a^{-1}) = \phi(a)^{-1}$.

(b) Ánh xạ $\phi : R \rightarrow R$ cho bởi $\phi(a) = a^p$, suy ra

$$\phi(a + b) = (a + b)^p = \sum_{i=0}^p \binom{p}{i} a^i b^{p-1}.$$

Vì $p \mid \binom{p}{i} = \frac{p!}{(p-i)! \cdot i!}$ (p là số nguyên tố) nên suy ra với mọi $1 \leq i \leq p-1$, ta có $\binom{p}{i} = 0$ (do $pa = 0$).

$$\Rightarrow \phi(a + b) = a^p + b^p = \phi(a) + \phi(b) \quad (1)$$

$$\Rightarrow \phi(ab) = (ab)^p = a^p b^p = \phi(a)\phi(b) \quad (2)$$

Từ (1) và (2) ta thu được ring homomorphism.

Exercise (Câu 2.32)

Prove Proposition 2.41

Vì $a_1 \equiv a_2 \pmod{m}$ nên $m \mid (a_1 - a_2)$.

Nói cách khác là tồn tại $k \in R$ sao cho $a_1 - a_2 = k \star m$.

Tương tự, tồn tại $l \in R$ sao cho $b_1 - b_2 = l \star m$.

Từ đó

$$a_1 - a_2 + b_1 - b_2 = (k + l) \star m,$$

tương đương với

$$m \mid (a_1 + b_1 - (a_2 + b_2)),$$

hay

$$a_1 + b_1 \equiv a_2 + b_2 \pmod{m}.$$

Tương tự cho $a_1 - b_1 \equiv a_2 - b_2 \pmod{m}$.

Đối với phép nhân, do

$$\begin{cases} a_1 = a_2 + k \star m \\ b_1 = b_2 + l \star m \end{cases}$$

ta có

$$\begin{aligned} a_1 \star b_1 &= (a_2 + k \star m)(b_2 + l \star m) \\ &= a_2 \star b_2 + a_2 \star l \star m + k \star b_2 \star m + k \star l \star m^2, \end{aligned}$$

suy ra $m \mid (a_1 \star b_1 - a_2 \star b_2)$ hay nói cách khác là

$$a_1 \star b_1 \equiv a_2 \star b_2 \pmod{m}.$$

❶ Exercise (Câu 2.33)

Prove Proposition 2.43

Theo Câu 2.32, nếu ta có $a' \in \bar{a}$ thì tương đương $a' \equiv a \pmod{m}$.

Tương tự, nếu ta có $b' \in \bar{b}$ thì tương đương $b' \equiv b \pmod{m}$.

Như vậy, theo Câu 2.32 thì

$$a' + b' \equiv a + b \pmod{m}, \quad a' \star b' \equiv a \star b \pmod{m}$$

Nói cách khác

$$a' + b' \in \overline{a + b} \quad \text{và} \quad a' \star b' \in \overline{a \star b},$$

nói cách khác phép tính cộng và nhân đóng trên R (closed).

Với mọi $a \in R$ ta có

$$\bar{a} + \bar{0} = \overline{\bar{a} + 0} = \bar{a} = \overline{0 + a} = \bar{0} + \bar{a}$$

Như vậy phần tử trung hòa của phép cộng là $\bar{0}$.

Với mọi $a \in R$ thì

$$\bar{a} + \overline{m - a} = \overline{\bar{a} + m - a} = \bar{0} = \overline{m - a} + \bar{a},$$

suy ra $\overline{m - a}$ là phần tử đối của phần tử a trong phép cộng.

Phép cộng trong modulo cũng có tính kết hợp

$$\bar{a} + (\bar{b} + \bar{c}) = \bar{a} + \overline{b + c} = \overline{\bar{a} + b + c} = \overline{\bar{a} + b} + \bar{c} = (\bar{a} + \bar{b}) + \bar{c}$$

Với mọi $a, b \in R$

$$\bar{a} + \bar{b} = \overline{\bar{a} + b} = \overline{b + a} = \bar{b} + \bar{a}$$

nên phép cộng modulo có tính giao hoán.

Vì $a \star 1 \equiv a \pmod{m}$ với mọi $a \in R$ nên

$$\bar{a} \star \bar{1} = \overline{a \star 1} = \bar{a} = \overline{1 \star a} = \bar{1} \star \bar{a}.$$

Suy ra phần tử đơn vị của phép nhân là $\bar{1}$.

Với mọi $a, b, c \in R$ ta có $a(bc) = (ab)c \pmod{m}$. Suy ra tính kết hợp của phép nhân

$$\bar{a} \star (\bar{b} \star \bar{c}) = \bar{a} \star \bar{bc} = \overline{abc} = \overline{ab} \star \bar{c} = (\bar{a} \star \bar{b}) \star \bar{c}.$$

Vì

$$\bar{a} \star \bar{b} = \overline{a \star b} = \overline{b \star a} = \bar{b} \star \bar{a}$$

ta có tính giao hoán của phép nhân.

Tính phân phối giữa phép nhân và phép cộng:

$$\bar{a} \star (\bar{b} + \bar{c}) = \bar{a} \star \overline{\bar{b} + \bar{c}} = \overline{a(\bar{b} + \bar{c})} = \overline{ab + ac} = \overline{ab} + \overline{ac} = \bar{a} \star \bar{b} + \bar{a} \star \bar{c}.$$

Như vậy, $R/(m)$ là vành (cụ thể vừa là vành với đơn vị vừa là vành giao hoán).

1 Exercise (Câu 2.34)

Let \mathbb{F} be a field and let \mathbf{a} and \mathbf{b} be nonzero polynomials in $\mathbb{F}[x]$

- (a) Prove that $\deg(\mathbf{a} \cdot \mathbf{b}) = \deg(\mathbf{a}) + \deg(\mathbf{b})$
- (b) Prove that \mathbf{a} has a multiplicative inverse in $\mathbb{F}[x]$ if and only if \mathbf{a} is in \mathbb{F} , i.e., if and only if \mathbf{a} is a constant polynomial
- (c) Prove that every nonzero element of $\mathbb{F}[x]$ can be factored into a product of irreducible polynomials.
(Hint. Use (a), (b) and induction on the degree of the polynomial.)
- (d) Let R be ring $\mathbb{Z}/6\mathbb{Z}$. Give an example to show that (a) is false for some polynomials \mathbf{a} and \mathbf{b} in $R[x]$.

(a) Đặt

$$\mathbf{a} = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0,$$

với $a_i \in \mathbb{F}$ và $\deg(\mathbf{a}) = n$.

Đặt

$$\mathbf{b} = b_m x^m + b_{m-1} x^{m-1} + \cdots + b_1 x + b_0,$$

với $b_i \in \mathbb{F}$ và $\deg(\mathbf{b}) = m$.

Hạng tử với bậc cao nhất trong phép nhân $\mathbf{a} \cdot \mathbf{b}$ là x^{n+m} , do đó

$$\deg(\mathbf{a} \cdot \mathbf{b}) = n + m = \deg(\mathbf{a}) + \deg(\mathbf{b}).$$

(b) Với

$$\mathbf{a} = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

Giả sử \mathbf{a} có nghịch đảo trong $\mathbb{F}[x]$ là đa thức

$$\mathbf{b} = b_m x^m + b_{m-1} x^{m-1} + \cdots + b_1 x + b_0.$$

Vì

$$\mathbf{a} \mathbf{b} = \sum_{i=0}^n a_i x^i \sum_{j=0}^m b_j x^j = \sum_{i=0}^n \sum_{j=0}^m a_i b_j x^{i+j} = 1.$$

Đồng nhất hệ số ta có $a_0 b_0 = 1$, các tích khác bằng 0. Như vậy \mathbf{a} là đa thức hằng.

(d) $\mathbf{a} = 2x^2 + 3x + 1$, $\mathbf{b} = 3x + 2$.

Trong $\mathbb{Z}/6\mathbb{Z}$ các hệ số được modulo cho 6 và ta có

$$\mathbf{ab} = x^2 + 3x + 2$$

Như vậy $\deg(\mathbf{ab}) = 2 < 3 = \deg(\mathbf{a}) + \deg(\mathbf{b})$

Exercise (Câu 2.37)

Prove that the polynomial $x^3 + x + 1$ is irreducible in $\mathbb{F}_2[x]$

Nếu $f(x) = x^3 + x + 1$ có nhân tử nào khác 1 và chính nó thì đa thức đó phải có bậc nhỏ hơn 3.

Các đa thức như vậy trong $\mathbb{F}_2[x]$ là

$$0, x + 1, x^2, x^2 + 1, x^2 + x, x^2 + x + 1, x.$$

Tuy nhiên $f(x)$ không chia hết cho bất kì đa thức nào kể trên nên $f(x)$ là đa thức tối giản.

Exercise (Câu 2.39)

The field $\mathbb{F}_7[x]/(x^2 + 1)$ is a field with 49 elements, which for the moment we denote by \mathbb{F}_{49}

Using example 2.58, every element in $\mathbb{F}_7[x]/(x^2 + 1)$ has form $f(x) = a + bx$, so in \mathbb{F}_{49} it has form $a + bi$ (here $i^2 = -1$)

- (a) Is $2 + 5x$ is a primitive root in \mathbb{F}_{49} ? No because $(2 + 5x)^8 = 1$
- (b) Is $2 + x$ is a primitive root in \mathbb{F}_{49} ? Yes
- (c) Is $1 + x$ is a primitive root in \mathbb{F}_{49} ? No because $(1 + x)^{24} = 1$

Chapter 3. Integer Factorization and RSA

Exercise (Câu 3.4)

Euler's phi function $\phi(N)$ is the function defined by

$$\phi(N) = |\{0 \leq k < N : \gcd(k, N) = 1\}|.$$

Chứng minh định lý Euler đã có trong bài viết về hàm Euler.

Exercise (Câu 3.5)

Properties of Euler's phi function

If p and q are distinct primes, how is $\phi(pq)$ related to $\phi(p)$ and $\phi(q)$?

Chứng minh tính chất nhân tính của hàm Euler đã có ở bài viết về hàm Euler.

❶ Exercise (Câu 3.6)

Let N , c , and e be positive integers satisfying the conditions $\gcd(N, c) = 1$ and $\gcd(e, \phi(N)) = 1$.

Explain how to solve the congruence $x^e \equiv c \pmod{N}$ assuming that you know the value of $\phi(N)$

Vì $\gcd(e, \phi(N)) = 1$, ta có thể tìm số $d \in \mathbb{Z}$ sao cho $ed \equiv 1 \pmod{\phi(N)}$ với thuật toán Euclid mở rộng.

Khi đó $ed = k\phi(N) + 1$ với $k \in \mathbb{Z}$.

Vì $\gcd(N, c) = 1$ nên $\gcd(N, x) = 1$, hay

$$c^d = (x^e)^d = x^{ed} = x^{k\phi(N)+1} = (x^k)^{\phi(N)}x,$$

và ta đã biết $(x^k)^{\phi(N)} \equiv 1 \pmod{N}$ từ Câu 3.4.

Như vậy $c^d \equiv x \pmod{N}$.

❶ Exercise (Câu 3.11)

Alice chooses two large primes p and q and she publishes $N = pq$. It is assumed that N is hard to factor. Alice also chooses three random numbers g , r_1 , and r_2 modulo N and computes

$$g_1 \equiv g^{r_1(p-1)} \pmod{N} \quad \text{and} \quad g_2 \equiv g^{r_2(q-1)} \pmod{N}$$

Her public key is the triple (N, g_1, g_2) and her private key is the pair of primes (p, q) .

Now Bob wants to send the message m to Alice, where m is a number modulo N . He chooses two random integers s_1 and s_2 modulo N and computes

$$c_1 \equiv mg_1^{s_1} \pmod{N} \quad \text{and} \quad c_2 \equiv mg_2^{s_2} \pmod{N}$$

Bob sends the ciphertext (c_1, c_2) to Alice.

Decryption is extremely fast and easy. Alice uses the Chinese remainder theorem to solve the pair of congruences

$$x \equiv c_2 \pmod{p} \quad \text{and} \quad x \equiv c_2 \pmod{q}$$

Prove that Alice's solution x is equal to Bob's plaintext m .

Ta có

$$c_1 \equiv mg_1^{s_1} \pmod{N} \equiv mg_1^{s_1} \pmod{p} \equiv m \pmod{p}$$

do $g_1^{s_1} = (g_1^{s_1 r_1})^{(p-1)} \equiv 1 \pmod{p}$.

Tương tự ta có $c_2 \equiv m \pmod{q}$.

Nghiệm của hệ đồng dư là

$$x \equiv c_1qq' + c_2pp' \pmod{N}$$

với $pp' + qq' = 1$.

$$\Rightarrow x \equiv mpp' + mqq' \equiv m(pp' + qq') \equiv m \pmod{N}$$

Ta có

$$g_1 \equiv g^{r_1(p-1)} \pmod{N} \equiv g^{r_1(p-1)} \pmod{p} \equiv 1 \pmod{p}.$$

Suy ra $p = \gcd(g_1 - 1, N)$. Tương tự, $q = \gcd(g_2 - 1, N)$.

Như vậy ta đã khôi phục lại private key.

Exercise (Câu 3.13)

Find x, y such that $xe_1 + ye_2 = 1 = \gcd(e_1, e_2)$.

$$\Rightarrow m = c_1^x c_2^y = m^{e_1 x + e_2 y} = m \pmod{N}$$

Exercise (Câu 3.14.)

[Exercise]

Because 3, 11 and 17 are primes number, $a \equiv a^3 \pmod{3}$, $a \equiv a^{11} \pmod{11}$, $a \equiv a^{17} \pmod{17}$. We have system congruence

$$\begin{aligned} a &\equiv a^3 \pmod{3} \\ a &\equiv a^{11} \pmod{11} \\ a &\equiv a^{17} \pmod{17} \end{aligned}$$

Consider that $a^3 \equiv a \pmod{3}$, $a^{3^2} \equiv a^3 \equiv a \pmod{3}$, \dots , $a^{3^i} \equiv a \pmod{3}$. And $561 = 2 \cdot 3^5 + 2 \cdot 3^3 + 2 \cdot 3^2 + 3^1$, $a^{561} \equiv a^2 \cdot a^2 \cdot a^2 \cdot a \equiv a^9 \equiv a \pmod{3}$.

Similarly, $a^{561} \equiv a \pmod{11}$, $a^{561} \equiv a \pmod{17}$. From system congruence:

$$\begin{aligned} a^{561} &\equiv a \pmod{3} \\ a^{561} &\equiv a \pmod{11} \\ a^{561} &\equiv a \pmod{17} \end{aligned}$$

Using CRT, $a^{561} = (187 \cdot 1 \cdot a + 51 \cdot 8 \cdot a + 33 \cdot 16 \cdot a) \pmod{561} = a \pmod{561}$

Suppose that n is even ($n \geq 4$), we have

$$(n-1)^{n-1} = (-1)^{n-1} = -1 \pmod{n},$$

but $a^{n-1} \equiv 1 \pmod{n}$ for all a , which is contrary. So n must be odd.

Suppose that $n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$ (p_i is odd prime). Because $a^{p^{e-1}(p-1)} \equiv 1 \pmod{p^e}$ and $a^{n-1} \equiv 1 \pmod{n}$, we have $a^{n-1} \equiv 1 \pmod{p^e}$.

$\Rightarrow p^{e-1}(p-1) \mid (n-1) \Rightarrow p^{e-1} \mid (n-1)$, but $p^{e-1} \mid n$, which is contrary if $e \geq 2$. Hence e must be 1.

Therefore $n = p_1 p_2 \cdots p_r$

Exercise (Câu 3.37)

[EXERCISE]

$$\begin{aligned} \left(a^{\frac{p-1}{2}}\right)^2 &\equiv a^{p-1} \equiv 1 \pmod{p} \\ \Rightarrow \binom{a}{p} &= \pm 1 \\ \Rightarrow \left(a^{\frac{p-1}{2}} - 1\right) \left(a^{\frac{p-1}{2}} + 1\right) &\equiv 0 \pmod{p} \\ \Rightarrow a^{\frac{p-1}{2}} &\equiv \pm 1 \pmod{p}. \end{aligned}$$

If a is quadratic residue, then $a \equiv b^2 \pmod{p}$

$$\Rightarrow a^{\frac{p-1}{2}} \equiv (b^2)^{\frac{p-1}{2}} = b^{p-1} \equiv 1 \pmod{p}$$

If $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$

Let g be generator modulo p , then $a \equiv g^m \pmod{p}$

$$\text{If } m \text{ is even } \Rightarrow a \equiv g^{2k} \pmod{p} \Rightarrow a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$$

If m is odd $\Rightarrow a = g^{2k+1} \pmod{p} \Rightarrow a^{\frac{p-1}{2}} \equiv g^{(2k+1)\frac{p-1}{2}} \equiv g^{p-1}g^{\frac{p-1}{2}} \equiv g^{\frac{p-1}{2}} \not\equiv 1 \pmod{p}$, because $p-1$ is smallest number that $g^{p-1} \equiv 1 \pmod{p}$

From (a) and (b) $\binom{-1}{p} \equiv (-1)^{\frac{p-1}{2}} \pmod{p}$, if $p = 4k+1 \Rightarrow (-1)^{\frac{p-1}{2}} \equiv (-1)^{2k} \equiv 1 \pmod{p}$ \ If $p = 4k+3 \Rightarrow (-1)^{\frac{p-1}{2}} \equiv (-1)^{2k+1} \equiv -1 \pmod{p}$

Exercise (Câu 3.38)

Prove that the three parts of the quadratic reciprocity theorem are equivalent to the following three concise formulas, where p and q are odd primes.

$$\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}}.$$

$$\text{With } p \equiv 1 \pmod{4} \Rightarrow \left(\frac{-1}{p}\right) = 1 = (-1)^{\frac{p-1}{2}} \pmod{p}.$$

$$\text{Similarly with } p \equiv 3 \pmod{4} \left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}.$$

First we need a lemma (**Gauss lemma**): suppose p is an odd prime, and $a \in \mathbb{Z}$, $\gcd(a, p) = 1$. Consider the set

$$a, 2a, 3a, \dots, \frac{p-1}{2}a.$$

If s of those residues are greater than $\frac{p}{2}$, then $\binom{a}{p} = (-1)^s$

Proof of lemma

Among smallest residues of $a, 2a, 3a, \dots, \frac{p-1}{2}a$, suppose that u_1, u_2, \dots, u_s are residues greater than $\frac{p}{2}$, and v_1, v_2, \dots, v_t are residues smaller than $\frac{p}{2}$.

Because $\gcd(ja, p) = 1$ for all j , where $1 \leq j \leq \frac{p-1}{2}$, all $u_i, v_j \neq 0$, then

$$u_i, v_j \in \{1, 2, \dots, p-1\}.$$

We will prove that, the set

$$\{p - u_1, p - u_2, \dots, p - u_s, v_1, v_2, \dots, v_t\}$$

is a permutation of $\{1, 2, \dots, \frac{p-1}{2}\}$

It is clear that there are no 2 numbers u_i or 2 numbers v_j simultaneously congruent modulo p . Because if $ma \equiv na \pmod{p}$ and $\gcd(a, p) = 1$, then $m \equiv n \pmod{p}$ and it is contrast with $m, n \leq \frac{p-1}{2}$.

Similarly, we see that there are no numbers $p - u_i$ congruent with v_j , so

$$(p - u_1)(p - u_2) \cdots (p - u_s)v_1v_2 \cdots v_t \equiv \left(\frac{p-1}{2}\right)! \pmod{p}.$$

On the other hand, $u_1, u_2, \dots, u_s, v_1, v_2, \dots, v_t$ are smallest residues of $a, 2a, 3a, \dots, \frac{p-1}{2}$, so

$$u_1u_2 \cdots u_s v_1v_2 \cdots v_t \equiv a^{\frac{p-1}{2}} \left(\frac{p-1}{2}\right)! \pmod{p}.$$

Therefore

$$(-1)^s a^{\frac{p-1}{2}} \left(\frac{p-1}{2}\right)! \equiv \left(\frac{p-1}{2}\right)! \pmod{p}$$

And because

$$\gcd(p, \left(\frac{p-1}{2}\right)!) = 1 \Rightarrow (-1)^s a^{\frac{p-1}{2}} \equiv 1 \pmod{p},$$

then

$$a^{\frac{p-1}{2}} \equiv (-1)^s \pmod{p}, \quad \binom{a}{p} = a^{\frac{p-1}{2}},$$

and $\binom{a}{p} = (-1)^s \pmod{p}$.

Return to problem: using theorem above, we need to find the number of residues, which are greater than $\frac{p}{2}$ among $1 \cdot 2, 2 \cdot 2, \dots, \frac{p-1}{2} \cdot 2$. Therefore we only need to know which numbers are greater than $\frac{p}{2}$

$$\Rightarrow \text{there are } s = \frac{p-1}{2} - \left[\frac{p}{4}\right] \Rightarrow \binom{\frac{p}{2}}{p} = (-1)^{\frac{p-1}{2} - [\frac{p}{4}]}$$

With $p \equiv 1, 3, 5, 7 \pmod{8}$, we have

$$\begin{aligned} \frac{p-1}{2} - \left[\frac{p}{4}\right] &\equiv \frac{p^2 - 1}{8} \pmod{2} \\ \Rightarrow \binom{\frac{p}{2}}{p} &= (-1)^{\frac{p^2-1}{8}} \end{aligned}$$

$$\binom{p}{q} \binom{q}{p} = (-1)^{\frac{p-1}{2} \cdot \frac{q-1}{2}}$$

We need a lemma: Suppose p is an odd prime, a is odd and $\gcd(a, p) = 1$, then $\left(\frac{a}{p}\right) = (-1)^{T(a,p)}$, with

$$T(a,p) = \sum_{j=1}^{\frac{p-1}{2}} \left[\frac{ja}{p} \right].$$

Proof of lemma

Consider smallest residues of $a, 2a, 3a, \dots, \frac{p-1}{2} \cdot a$. As Gauss's lemma, $u_1, u_2, \dots, u_s, v_1, v_2, \dots, v_t$ are residues greater and less than $\frac{p}{2}$ respectively. According to Euclidean divisor:

$$ja = p \left[\frac{ja}{p} \right] + \text{remainder},$$

remainder is u_i or v_j . We have such $\frac{p-1}{2}$ equations and add them together

$$\Rightarrow \sum_{j=1}^{\frac{p-1}{2}} ja = \sum_{j=1}^{\frac{p-1}{2}} p \left[\frac{ja}{p} \right] + \sum_{i=1}^s u_i + \sum_{j=1}^t v_j$$

As we pointed out in Gauss's lemma, the set $p - u_1, p - u_2, \dots, p - u_s, v_1, v_2, \dots, v_t$ is a permutation of the set $1, 2, \dots, \frac{p-1}{2}$

$$\begin{aligned} \sum_{j=1}^{\frac{p-1}{2}} j &= \sum_{i=1}^s (p - u_i) + \sum_{j=1}^t v_j = ps - \sum_{i=1}^s u_i + \sum_{j=1}^t v_j \\ \Rightarrow \sum_{j=1}^{\frac{p-1}{2}} ja - \sum_{j=1}^{\frac{p-1}{2}} j &= \sum_{j=1}^{\frac{p-1}{2}} p \left[\frac{ja}{p} \right] - ps + 2 \sum_{i=1}^s u_i \end{aligned}$$

From formula of $T(a,p)$, $(a-1) \sum_{j=1}^{\frac{p-1}{2}} j = pT(a,p) - ps + 2 \sum_{i=1}^s u_i$

Because a, p are odd, $T(a,p) \equiv s \pmod{2}$. From Gauss's lemma we finish.

Return to problem: Consider pairs (x,y) , where $1 \leq x \leq \frac{p-1}{2}$ and $1 \leq y \leq \frac{q-1}{2}$, there are $\frac{p-1}{2} \cdot \frac{q-1}{2}$ pairs. We divide those pairs into 2 groups, depending on the magnitude of px and qy .

Because p, q are two different primes, $px \neq qy, \forall (x,y)$.

We consider pairs with $qx > py$. With every fixed element of x ($1 \leq x \leq \frac{p-1}{2}$), exist $\left[\frac{qx}{p} \right]$ elements y satisfying $1 \leq y \leq \frac{q-1}{2}$. Therefore, there are $\sum_{i=1}^{\frac{p-1}{2}} \left[\frac{iq}{p} \right]$ pairs. When $qx < py$, similarly, there are $\sum_{j=1}^{\frac{q-1}{2}} \left[\frac{jp}{q} \right]$ pairs. Because there are $\frac{p-1}{2} \cdot \frac{q-1}{2}$ pairs, we have equation

$$\sum_{i=1}^{\frac{p-1}{2}} \left[\frac{iq}{p} \right] + \sum_{j=1}^{\frac{q-1}{2}} \left[\frac{jp}{q} \right] = \frac{p-1}{2} \cdot \frac{q-1}{2}$$

From definition of $T(p,q)$, we have result

$$(-1)^{T(p,q)+T(q,p)} = (-1)^{\frac{p-1}{2} \cdot \frac{q-1}{2}}$$

1 Exercise (Câu 3.39)

Let p be a prime satisfying $p \equiv 3 \pmod{4}$.

Let a be a quadratic residue modulo p . Prove that the number

$$b \equiv a^{\frac{p+1}{4}} \pmod{p}$$

has the property that $b^2 \equiv a \pmod{p}$. (*Hint.* Write $\frac{p+1}{2}$ as $1 + \frac{p-1}{2}$ and use Exercise 3.37.) This gives an easy way to take square roots modulo p for primes that are congruent to 3 modulo 4.

Dùng Câu 3.37, $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$ vì a là thăng dư chính phương modulo p .

Khi đó

$$b^2 \equiv a^{\frac{p+1}{2}} \equiv a^{1+\frac{p-1}{2}} \equiv a \cdot a^{\frac{p-1}{2}} \equiv a \cdot 1 \equiv 1 \pmod{p}$$

1 Exercise (Câu 3.40)

Let p be an odd prime, let $g \in \mathbb{F}_p^*$ be a primitive root, and let $h \in \mathbb{F}_p^*$. Write $p-1 = 2^s m$ with m odd and $s \geq 1$, and write the binary expansion of $\log_g(h)$ as

$$\log_g(h) = \varepsilon_0 + 2\varepsilon_1 + 4\varepsilon_2 + 8\varepsilon_3 + \dots \quad \text{with } \varepsilon_0, \varepsilon_1, \dots \in \{0, 1\}$$

Give an algorithm that generalizes Example 3.69 and allows you to rapidly compute $\varepsilon_0, \varepsilon_1, \dots, \varepsilon_{s-1}$, thereby proving that the first s bits of the discrete logarithm are insecure.

1 Algorithm 2 (Thuật toán tìm s least significant bit (LSB) của x trong $g^x \equiv h \pmod{p}$)

Input: g, h, p ($p-1 = 2^s m$)

Output: s least significant bits của x trong $g^x \equiv h \pmod{p}$

1. Ta sẽ tìm mảng $\varepsilon_0, \varepsilon_1, \dots, \varepsilon_{s-1}$
2. For $i = 0, \dots, s-1$
 1. If h là thăng dư chính phương
 1. $\varepsilon_i = 0, h = \sqrt{h} \pmod{p}$
 2. ElseIf $\varepsilon_i = 1$
 1. $h = \sqrt{g^{-1}h} \pmod{p}$
 3. EndIf
3. EndFor

1 Exercise (Câu 3.41)

Let p be a prime satisfying $p \equiv 1 \pmod{3}$. We say that a is a *cubic residue modulo* p if $p \nmid a$ and there is an integer c satisfying $a \equiv c^3 \pmod{p}$.

- (a) Let a and b be cubic residues modulo p . Prove that ab is a cubic residue modulo p .
- (b) Give an example to show that (unlike the case with quadratic residues) it is possible for none of a , b and ab to be cubic residues modulo p .
- (c) Let g be a primitive root modulo p . Prove that a is a cubic residue modulo p if and only if $3 \mid \log_g(a)$, where $\log_g(a)$ is the discrete logarithm of a to the base g .
- (d) Suppose instead that $p \equiv 2 \pmod{3}$. Prove that for every integer a there is an integer c satisfying $a \equiv c^3 \pmod{p}$. In other words, if $p \equiv 2 \pmod{3}$, show that every number is a cube modulo p .

(a) Ta có $a \equiv x^3 \pmod{p}$ và $y \equiv y^3 \pmod{p}$ với x và y nào đó thuộc \mathbb{F}_p .

Suy ra $ab \equiv x^3y^3 = (xy)^3 \pmod{p}$ cũng là thặng dư bậc ba.

(b) Gọi g là primitive root modulo p .

Ta chọn $a \equiv g^{3k+1} \pmod{p}$ và $b \equiv g^{3k'+1} \pmod{p}$.

Khi đó $ab \equiv g^{(3k+1)+(3k'+1)} \equiv g^{3(k+k')+2} \pmod{p}$ không phải thặng dư bậc ba.

(c) Quên làm.

Điều kiện đủ. Nếu a là thặng dư bậc ba modulo p , giả sử $a \equiv c^3 \pmod{3}$ và $c \equiv g^u \pmod{p}$.

Khi đó $a = g^{3u} \pmod{p}$ và theo định lý Lagrange thì $3 \mid \log_g(a)$.

Điều kiện cần. Nếu $3 \mid \log_g(a)$ thì làm ngược lại bước chứng minh điều kiện đủ.

(d) Vì $p \equiv 2 \pmod{3}$ nên $\gcd(p-1, 3) = 1$. Khi đó tồn tại phần tử d là nghịch đảo của 3 modulo $p-1$.

Suy ra phương trình $x^3 \equiv a \pmod{p}$ có nghiệm $a^d = x \pmod{p}$.

Nói cách khác mọi phần tử đều là số bậc ba modulo p .

Chapter 4. Digital Signatures

Exercise (Câu 4.1)

Dáp án: $d = 561517$, $N = 661643$, $sig = 206484$.

Exercise (Câu 4.3)

Dáp án $p = 212081$, $q = 128311$ nên

$$d = 18408628619 \Rightarrow S = D^d \pmod{N} = 22054770669.$$

Exercise (Câu 4.4)

Dáp án: with $c = m^{e_B} \pmod{N_B}$ and $s = \text{Hash}(m)^{d_A} \pmod{N_A}$

$$c^{d_B} = m^{e_B \cdot d_B} \pmod{N_B} = m, \quad s^{e_A} = \text{Hash}(m)^{d_A \cdot e_A} \pmod{N_A} = \text{Hash}(m).$$

Hence this method works.

Exercise (Câu 4.5)

Dáp án: với $A = g^a \pmod{p} = 2065$, ta tính

$$S_1 = g^k \pmod{p} = 3534, S_2 = (D - a \cdot S_1)k^{-1} \pmod{p-1} = 5888.$$

Hence signature is $(S_1, S_2) = (3534, 5888)$

Exercise (Câu 4.6)

Dáp án: $A^{S_1} \cdot S_1^{S_2} \equiv g^D \pmod{p}$, so (S_1'', S_2'') is valid signature.

Exercise (Câu 4.8)

Dáp án: $S_1 = S'_1 = g^k \pmod{p}$, from here Eve can know at first glance that the same random element k is used

With $S_2 = (D - aS_1)k^{-1} \pmod{p-1}$, $S'_2 = (D' - aS'_1)k^{-1} \pmod{p-1}$, then

$$\begin{aligned} S_2 - S'_2 &\equiv (D - D')k^{-1} \pmod{p-1} \\ k &= (D - D')(S_2 - S'_2)^{-1} \pmod{p-1}. \end{aligned}$$

Here we get $D - aS_1 = S_2k \pmod{p-1}$

$$\Rightarrow \begin{cases} a = (D - S_2k)S_1^{-1} \pmod{p-1} \\ a = (D' - S'_2k)S_1^{-1} \pmod{p-1} \end{cases}.$$

Exercise (Câu 4.9)

Dáp án: $p \equiv 1 \pmod{q}$, $1 \leq a \leq q-1$, $A = g^a \pmod{p}$, $S_1 = (g^k \pmod{p}) \pmod{q}$, $S_2 = (D + aS_1)k^{-1} \pmod{q}$

Verify: $V_1 = D \cdot S_2^{-1} \pmod{q}$, $V_2 = S_1 S_2^{-1} \pmod{q}$. We need to prove that $(g^{V_1} \cdot A^{V_2} \pmod{p}) \pmod{q} = S_1$

Here we have

$$\begin{aligned} g^{V_1} \cdot A^{V_2} &\equiv g^{D \cdot S_2^{-1}} \cdot g^{aS_1 S_2^{-1}} \pmod{p} \\ &\equiv g^{(D+aS_1)S_2^{-1}} \pmod{p} \\ &\equiv g^k \pmod{p} \end{aligned}$$

Hence $(g^{V_1} \cdot A^{V_2} \pmod{p}) \pmod{q} = S_1$.

Exercise (Câu 4.10)

Dáp án: $(p, q, g) = (22531, 751, 4488)$. Public key $A = 22476$ is not valid.

1 Exercise (Câu 4.11)

Dáp án: $A = g^a \pmod{p}$. $A = 31377$, $g = 21947$, $p = 103687$, then

$$a = 602, S_1 = (g^k \pmod{p}) \pmod{q} = 439, S_2 = (D + aS_1)k^{-1} \pmod{q} = 1259.$$

Chapter 7. Lattices and Cryptography**1 Exercise (Câu 7.43)**

Dáp án: $t = \frac{\mathbf{b}_1 \cdot \mathbf{b}_2}{\|\mathbf{b}_1\|^2}$ và $\mathbf{b}_2^* = \mathbf{b}_2 - t\mathbf{b}_1$ nên suy ra

$$\mathbf{b}_2^* \cdot \mathbf{b}_1 = \mathbf{b}_1(\mathbf{b}_2 - t\mathbf{b}_1) = \mathbf{b}_1 \cdot \mathbf{b}_2 - t\|\mathbf{b}_1\|^2 = \mathbf{b}_1 \cdot \mathbf{b}_2 - \frac{\mathbf{b}_1 \cdot \mathbf{b}_2}{\|\mathbf{b}_1\|^2} \cdot \|\mathbf{b}_1\|^2 = 0$$

Do đó $\mathbf{b}_2^* \perp \mathbf{b}_1$ và \mathbf{b}_2^* là hình chiếu của \mathbf{b}_2 lên orthogonal complement của \mathbf{b}_1 .

1 Exercise (Câu 7.44)

Dáp án:

$$\|\mathbf{a} - t\mathbf{b}\|^2 = (\mathbf{a} - t\mathbf{b})^2 = \mathbf{a}^2 - 2t\mathbf{a} \cdot \mathbf{b} + t^2\mathbf{b}^2 = \|\mathbf{a}\|^2 + t^2\|\mathbf{b}\|^2 - 2t\mathbf{a} \cdot \mathbf{b} \geq 0$$

với mọi $t \in \mathbb{R}$.

Cho $\mathbf{a} - t\mathbf{b} = 0$ ta có $t = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{b}\|^2}$.

Từ đó ta có

$$(\mathbf{a} - t\mathbf{b}) \cdot \mathbf{b} = \mathbf{a} \cdot \mathbf{b} - t\|\mathbf{b}\|^2 = \mathbf{a} \cdot \mathbf{b} - \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{b}\|^2} \cdot \|\mathbf{b}\|^2 = 0.$$

Vì vậy $\mathbf{a} - t\mathbf{b}$ là hình chiếu của \mathbf{a} lên orthogonal complement của \mathbf{b} (tương tự 7.43).

1 Exercise (Câu 7.45)

Dáp án ở dưới.

Thuật toán Gauss's lattice reduction.

1 Algorithm 3 (Thuật toán Gauss's lattice reduction)

1. While True
 1. If $\|\mathbf{v}_2\| < \|\mathbf{v}_1\|$
 1. swap \mathbf{v}_1 and \mathbf{v}_2
 2. $m \leftarrow \lfloor \mathbf{v}_1 \cdot \mathbf{v}_2 / \|\mathbf{v}_1\|^2 \rfloor$

2. EndIf
3. If $m = 0$
 1. return $(\mathbf{v}_1, \mathbf{v}_2)$
4. EndIf
5. Replace \mathbf{v}_2 with $\mathbf{v}_2 - m\mathbf{v}_1$
2. EndWhile

$\mathbf{v}_1 = (14, -47)$, $\mathbf{v}_2 = (-362, -131)$, 6 steps.

$\mathbf{v}_1 = (14, -47)$, $\mathbf{v}_2 = (-362, -131)$, 6 steps.

$\mathbf{v}_1 = (147, 330)$, $\mathbf{v}_2 = (690, -207)$, 7 steps.

Exercise (Câu 7.46)

Đáp án ở dưới.

Do W^\perp là orthogonal complement của W trong V nên nếu $\mathbf{z} \in W^\perp$ thì $\mathbf{z} \cdot \mathbf{y} = 0$, với mọi $\mathbf{y} \in W$.

Với hai vector $\mathbf{z}_1, \mathbf{z}_2 \in W^\perp$ ta có $\mathbf{z}_1 \cdot \mathbf{y} = \mathbf{z}_2 \cdot \mathbf{y} = 0$, với mọi $\mathbf{y} \in W$.

Như vậy $(\mathbf{z}_1 + \mathbf{z}_2) \cdot \mathbf{y} = 0 \Rightarrow \mathbf{z}_1 + \mathbf{z}_2 \in W^\perp$.

Ta lại có $\alpha \mathbf{z}_1 \cdot \mathbf{y} = \alpha \cdot 0 = 0 \Rightarrow \alpha \mathbf{z}_1 \in W^\perp$ với mọi $\alpha \in \mathbb{R}$.

Tới đây ta có hai cách giải.

Cách 1. Ta có $W \cup W^\perp = \{\mathbf{0}\}$. Nếu \mathbf{u} thuộc cả hai tập W và W^\perp thì $\mathbf{u} \cdot \mathbf{u} = 0 \Rightarrow \mathbf{u} = \mathbf{0}$.

Kí hiệu $U = W + W^\perp$, ta chứng minh $W = V$.

Ta có thể chọn một cơ sở trực chuẩn (orthonormal basis) trong U và mở rộng nó thành cơ sở trực chuẩn trong V .

Khi đó, nếu $U \neq V$ thì có một phần tử \mathbf{e} trong cơ sở của V vuông góc với U . Do U chứa W và \mathbf{e} vuông góc với U nên $\mathbf{e} \in W^\perp$.

Phần sau là không gian con của W , do đó \mathbf{e} thuộc W ,矛盾 thuẫn.

Cách 2. Đặt $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k\}$ là cơ sở trực chuẩn của không gian con W . Với mỗi $\mathbf{v} \in V$, đặt

$$P(\mathbf{v}) = \sum_{j=1}^k (\mathbf{v} \cdot \mathbf{e}_j) \cdot \mathbf{e}_j$$

Khi đó với mọi $\mathbf{v} \in V$ thì $\mathbf{v} = \underbrace{P(\mathbf{v})}_{\in W} + \underbrace{(\mathbf{v} - P(\mathbf{v}))}_{\in W^\perp}$.

Ở đây $\mathbf{v} - P(\mathbf{v}) \in W^\perp$ là vì nếu $j \in \{1, 2, \dots, k\}$ thì

$$\begin{aligned} (\mathbf{v} - P(\mathbf{v})) \cdot \mathbf{e}_j &= \left(\mathbf{v} - \sum_{l=1}^k (\mathbf{v} \cdot \mathbf{e}_l) \cdot \mathbf{e}_l \right) \cdot \mathbf{e}_j \\ &= \mathbf{v} \cdot \mathbf{e}_j - \mathbf{v} \cdot \mathbf{e}_j = 0. \end{aligned}$$

Do $\{\mathbf{e}_1, \dots, \mathbf{e}_k\}$ là cơ sở của W , điều này cho ta $\mathbf{v} - P(\mathbf{v}) \in W^\perp$.

Như vậy

$$\begin{aligned}\|\mathbf{v}\|^2 &= (a\mathbf{w} + b\mathbf{w}')^2 = a^2\mathbf{w}^2 + 2ab\mathbf{w}\mathbf{w}' + b^2\mathbf{w}'^2 \\ &= a^2\|\mathbf{w}\|^2 + 0 + b^2\|\mathbf{w}'\|^2 = a^2\|\mathbf{w}\|^2 + b^2\|\mathbf{w}'\|^2.\end{aligned}$$

4.1.4 Các bài toán sâu tâm

Олимпиада Якут

Exercise 1

Tính $\lim_{t \rightarrow +\infty} t \sum_{k=1}^{+\infty} \frac{1}{k^2 + t^2}$.

Ta có công thức thông dụng

$$\int \frac{dx}{x^2 + t^2} = \frac{1}{t} \arctan \frac{x}{t}.$$

Ta có chẩn

$$\frac{1}{(k+1)^2 + t^2} \leq \int_k^{k+1} \frac{dx}{x^2 + t^2} \leq \frac{1}{k^2 + t^2},$$

suy ra

$$\frac{1}{k^2 + t^2} \leq \int_{k-1}^k \frac{dx}{x^2 + t^2}.$$

Cộng tất cả phương trình trên với $k = 1, 2, \dots$ thì

$$\sum_{k=1}^{\infty} \frac{1}{k^2 + t^2} \leq \int_0^{+\infty} \frac{dx}{x^2 + t^2} = \frac{1}{t} \cdot \arctan \frac{x}{t} \Big|_0^{+\infty} = \frac{1}{t} \cdot \frac{\pi}{2}.$$

Tương tự

$$\begin{aligned}\int_k^{k+1} \frac{dx}{x^2 + t^2} &\leq \frac{1}{k^2 + t^2} \\ \Rightarrow \int_1^{\infty} \frac{dx}{x^2 + t^2} &\leq \sum_{k=1}^{\infty} \frac{1}{k^2 + t^2}\end{aligned}$$

Do

$$\int_1^{\infty} \frac{dx}{x^2 + t^2} = \frac{1}{t} \arctan \frac{x}{t} \Big|_1^{\infty} = \frac{1}{t} \left(\frac{\pi}{2} - \arctan \frac{1}{t} \right)$$

nên

$$\left(\frac{\pi}{2} - \arctan \frac{1}{t} \right) \leq t \sum_{k=1}^{\infty} \frac{1}{k^2 + t^2} \leq \frac{\pi}{2}.$$

Như vậy

$$\lim \left(\frac{\pi}{2} - \arctan \frac{1}{t} \right) = \lim \frac{\pi}{2} = \frac{\pi}{2}$$

khi $t \rightarrow \infty$.

1 Exercise 2

Giải phương trình

$$19^x - 13^x = 9^x - 3^x.$$

Dễ thấy $x = 0$ là một nghiệm của phương trình.

Giả sử phương trình có nghiệm khác 0 là x .

Cố định x , đặt $g(t) = t^x$.

Theo định lí Lagrange, tồn tại $\xi \in (a, b)$ để $g(a) - g(b) = g'(\xi) \cdot (a - b)$.

Như vậy

$$g'(\xi) \cdot (19 - 13) = g'(\eta) \cdot (9 - 6)$$

với $\xi \in (13, 19)$ và $\eta \in (6, 9)$. Suy ra $g'(\xi) = g'(\eta)$, nói cách khác là

$$x \cdot \xi^{x-1} = x \cdot \eta^{x-1}$$

mà $x \neq 0$ nên $\left(\frac{\xi}{\eta} \right)^{x-1} = 1$. Điều này chỉ xảy ra khi $x - 1 = 0$, hay $x = 1$.

Kết luận: phương trình có hai nghiệm là $x = 0$ và $x = 1$.

4.1.5 Wargame chill chill

Để tránh vấn đề bản quyền và bị soi bởi cộng đồng toxic nào đó thì mình sẽ không để ở đây.

Đã giải:

- Bounded Noise (đã viết lời giải)
- Forbidden Fruit
- Noise Free
- Too Many Errors
- Pad-Thai
- Paper Plane

Bound noise

Đây là một bài LWE cơ bản.

Giả sử flag là một số nguyên M . Chọn số nguyên tố $q = 65537$ và biểu diễn M ở dạng cơ số q thì ta được vector

$$M = s_0 + s_1 q + s_2 q^2 + \cdots + s_{n-1} q^n \mapsto (s_0, s_1, \dots, s_{n-1}).$$

Secret sẽ là vector (s_0, \dots, s_{n-1}) và mình kí hiệu ngắn gọn là s .

Chọn ngẫu nhiên ma trận A kích thước $m \times n$ với các phần tử thuộc \mathbb{F}_q , trong đó n là độ dài vector s và $m = n^2$.

Sau đó chọn ngẫu nhiên vector e độ dài m với các phần tử thuộc $\{0, 1\}$. Tính vector $b = A \cdot s + e$.

Khi đó public key là ma trận A và vector b . Ta cần tìm flag là secret key.

[TODO] Cần hiểu cách thực hiện của đoạn code kia.

Forbidden fruit

Noise free

Too Many Errors

Pad Thai

Paper Plane

4.2 Các cuộc thi năm 2020

4.2.1 Midnight Sun CTF 2020

rsa_yay

Nếu biết k bit cao nhất của p và q , gọi là ph và qh thì ta có chẵn

$$ph \cdot qh \cdot 2^{1024-2k} \leq n < (ph+1) \cdot (qh+1) \cdot 2^{1024-2k}.$$

Khi đó, ta brute 12 bit thấp nhất của p và tính nghịch đảo của từng trường hợp trong modulo 2^{12} . Nghịch đảo này chính là 12 bit thấp nhất của q và suy ra được 12 bit cao nhất của p và q .

4.2.2 NSUCRYPTO 2020

Problem 1 (R1). 2020

Đề bài

Cho dãy S , cipher machine có thể thêm vào S hoặc xóa khỏi S dãy con với dạng 11, 101, 1001, 10...01. Machine cũng có thể xóa khỏi S hoặc thêm vào S một dãy liên tục các số 0.

Smith biểu diễn số 2020 dưới dạng nhị phân và sử dụng hai cipher machine để biến đổi. Máy đầu tiên trả về dạng nhị phân của số 1984, máy thứ hai trả về dạng nhị phân của số 2021.

Smith chắc chắn rằng một trong hai máy đã bị lỗi. Làm sao anh ấy biết được?

Giải

Ta viết các số 2020, 1984 và 2021 dưới dạng nhị phân.

Như vậy

$$2020 = 11111100100, \quad 1984 = 11111000000, \quad 2021 = 11111100101.$$

Đặt $2020 = 11111 \underbrace{1001}_{A} 00$ thì nếu ta xóa dãy A khỏi 2020 và thêm vào vị trí đó bốn chữ số 0 thì ta có 1984.

Như vậy máy đầu tiên hoạt động bình thường.

Việc thêm vào hoặc xóa đi 11, 101, 1001, ... thì số lượng bit 1 tăng giảm đi một lượng chẵn còn thêm bớt 0 không ảnh hưởng. Do 2020 và 2021 có số lượng bit 1 khác tính chẵn lẻ nên không thể biến đổi từ 2020 thành 2021.

Như vậy máy thứ hai bị hỏng.

Problem 4 (R1). RGB

Đề bài

Trong một server có hai biến a và b . Khi server nhận query dạng "RED", "GREEN" hoặc "BLUE" thì giá trị của chúng sẽ thay đổi theo query nhận được.

Cụ thể hơn, cặp (a, b) biến thành $(a + 18b, 18a - b)$ nếu query là "RED", biến thành $(17a + 6b, -6a + 17b)$ nếu query là "GREEN" và biến thành $(-10a - 15b, 15a - 10b)$ nếu là "BLUE".

Khi a hoặc b trở thành bội của 324 thì nó được reset thành 0. Khi $(a, b) = (0, 0)$ thì server crash.

Ban đầu, (a, b) được khai báo là $(20, 20)$. Chứng minh rằng server sẽ không bao giờ crash dù có bao nhiêu query được gửi tới.

Giải

Phép biến đổi tương đương với phép nhân ma trận $(a, b) \rightarrow (a, b)\mathbf{A}$ với \mathbf{A} là ma trận xác định bởi

1. $\mathbf{A} = \begin{pmatrix} 1 & 18 \\ 18 & -1 \end{pmatrix}$ khi query là "RED".
2. $\mathbf{A} = \begin{pmatrix} 17 & -6 \\ 6 & 17 \end{pmatrix}$ khi query là "GREEN".
3. $\mathbf{A} = \begin{pmatrix} -10 & 15 \\ -15 & -10 \end{pmatrix}$ khi query là "BLUE".

Kết quả cuối là việc thực hiện liên tiếp các phép nhân có dạng

$$(a, b) \cdot \mathbf{A}_1 \mathbf{A}_2 \cdots \mathbf{A}_r$$

với \mathbf{A}_i là một trong ba ma trận trên.

Đặt $\mathbf{A}_1 \cdots \mathbf{A}_r = B$ thì phép nhân tương đương với $(a, b) \cdot \begin{pmatrix} X & Y \\ Z & T \end{pmatrix} = (324, 324)$. Trong đó $X, Y, Z, T \in \mathbb{Z}$.

Điều này tương đương với $a \cdot X + b \cdot Z = 324$ và $a \cdot Y + b \cdot T = 324$. Suy ra $20(X + Z) = 324$ và $20(Y + T) = 324$. Nhưng điều này không thể xảy ra vì 324 không chia hết 20.

Như vậy server không bao giờ crash dù có thực hiện bao nhiêu query đi nữa.

Problem 1. Poly

Đề bài

Bob phát minh một thuật toán tên là *POLY*. Với x là plaintext thì ciphertext là $y = p(x)$, với $p(x)$ là đa thức với hệ số nguyên.

Đồng nghiệp của Bob muốn kiểm tra thuật toán. Khi encrypt 20 thì Bob nhận được 7. Sau đó anh đồng nghiệp encrypt 15 thì nhận được 5. Anh ta kết luận rằng thuật toán đã bị cài đặt sai. Chuyện gì đã xảy ra?

Giải

Giả sử đa thức là

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$

Khi đó

$$p(20) = a_n 20^n + a_{n-1} 20^{n-1} + \cdots + a_1 \cdot 20 + a_0 = 7.$$

Tương tự

$$p(15) = a_n 15^n + a_{n-1} 15^{n-1} + \cdots + a_1 \cdot 15 + a_0 = 5.$$

Suy ra $p(20) - p(15) = 2$ mà $p(20) - p(15)$ chia hết cho $20 - 15 = 5$ nên vô lý vì 2 không chia hết cho 5. Như vậy thuật toán đã bị cài đặt sai.

Problem 3. Hidden RSA**Đề bài**

Bob học về mật mã khóa công khai và bây giờ mọi người đều có thể gửi message cho anh ta. Message được biểu diễn thành số nguyên không âm x và chứa tối đa 70 chữ số trong biểu diễn thập phân. Để gửi message cho Bob, chúng ta nhập x vào trang web và kết quả được trả về ở dạng

$$\text{Encr}(x) = x^e \pmod{n},$$

với n là modulus (biết rằng n là tích của hai số nguyên tố phân biệt p và q) và e là số mũ công khai (nguyên tố cùng nhau với $p - 1$ và $q - 1$). Bob sợ bị hacker tấn công nên giấu đi n và e .

Victor bắt được một encrypted message

$$y = 7151189668132483345836139288518434493333159830863878600189212073777582178173,$$

được Alice gửi cho Bob.

Hãy giúp Victor giải mã nó.

Giải

Gọi f_2 , f_3 và f_6 là kết quả trả về khi nhập 2, 3 và 6 lên trang web. Minh nhận được

$$f_2 = 50154912289039335014669339773308393642658123228965873078737860474494117389068$$

$$f_3 = 74177167678866806519929337366689313939300015489238864541679630476008627210599$$

$$f_6 = 69732835711852253044075185248502970714729629373386336194927784886349053828079.$$

Vì

$$\begin{aligned} f_6 &\equiv 6^e \pmod{n} \\ &\equiv (2^e) \cdot (3^e) \pmod{n} \\ &\equiv f_2 \cdot f_3 \pmod{n} \end{aligned}$$

nên suy ra $n \mid (f_2 \cdot f_3 - f_6)$.

Tương tự, gọi f_5 , f_7 và f_{35} là kết quả trả về khi nhập 5, 7 và 35. Minh nhận được

$$f_5 = 66788051164865948223783605396869677445056352267867968640234839015540677264876$$

$$f_7 = 25469333231403648059708659888338792850140504272696299331424365245642760908571$$

$$f_{35} = 25850860693609575302160565920815769473222469094759983686766869114847002714718.$$

Tương tự $n \mid (f_3 \cdot f_5 - f_{35})$.

Như vậy

$$n = \gcd(f_2 \cdot f_3 - f_6, f_3 \cdot f_5 - f_{35}).$$

Mình thu được

$$n = 76200708443433250012501342992033571586971760218934756930058661627867825188509.$$

Sử dụng factordb mình tìm được

$$p = 232086664036792751646261018215123451301, \quad q = 328328681700354546732404725320581286809.$$

Với số e mình chọn $e = 65537$ là public exponent được dùng nhiều nhất trên thực tế và kiểm tra $2^e \pmod n$ có khớp với f_2 không, tương tự với $3^e \pmod n$ với f_3, \dots và hoàn toàn trùng kết quả.

Như vậy $e = 65537$ và tính

$$d = e^{-1} \pmod{(p-1) \cdot (q-1)},$$

từ đó decrypt ra được message ban đầu (theo RSA)

$$m = 202010181600.$$

Sau này đọc bài giải của anh Hiếu (ndh) thì mình mới biết ý nghĩa của plaintext là 2020-10-18-16-00 là năm, tháng, ngày, giờ bắt đầu round 2 (mùi giờ Novosibirsk, UTC+7).

4.3 Các cuộc thi năm 2021

4.3.1 Crypto CTF 2021

Chào mọi người, lại một mùa Crypto CTF nữa đã qua và lần này mình có khá khâm phục năm ngoái một chút, và sau đây là writeup các challenge mình đã làm được

Source code đề và bài giải của mình ở [đây](#).

Farm

Đề bài cho mình F tạo các đa thức trên $GF(2^6)$, `maptofarm` để lấy đa thức tương ứng với chỉ số của ký tự trong alphabet, và `encrypt` để mã hóa chuỗi base64, bằng cách là ký tự base64 m sẽ thành `ALPHABET[F.index(pkey * maptofarm(chr(m)))]`.

Ok, vì key chỉ nằm trong $GF(2^6)$, mình chỉ việc bruteforce thôi (nằm trong $GF(2^6)$ nên độ dài là 14 hay bao nhiêu cũng không quan trọng). Làm ngược lại quá trình mã hóa mình có flag.

Flag: CCTF{EnCrYp7I0n_4nD_5u8STitUtIn9_iN_Fi3Ld!}.

KeyBase

Bài này cho mình một hệ thống sử dụng AES mode CBC (key và iv giống nhau suốt một phiên kết nối) để làm hai việc:

- đưa flag đã bị mã hóa;
- mã hóa một đoạn input 32 byte bất kì và trả lại key (dạng hex) với 2 byte cuối bị ẩn, và ciphertext với 14 byte ở giữa bị ẩn.

Do mode CBC có tính chất $C_{i+1} = E_k(C_i \oplus P_{i+1})$, với $P_0 = iv$. Do đó nếu mình chọn P và P' là hai plaintext có 16 bytes đầu giống nhau còn 16 bytes cuối khác nhau thì mình có $C_1 = C'_1 = E_k(iv \oplus P_1)$ còn $C_2 = E_k(C_1 \oplus P_2)$ và $C'_2 = E_k(C'_1 \oplus P'_2)$.

Từ đây mình có $C_1 \oplus P_2 \oplus C'_1 \oplus P'_2 = D_k(C_2) \oplus D_k(C'_2)$, mà $C_1 = C'_1$ rồi nên mình cần lấy ciphertext nào mà 16 bytes cuối không bị ẩn đi để có thể bruteforce 2 byte cuối của key và decrypt, tức là mình phải có $P_2 \oplus P'_2 = D_k(C_2) \oplus D_k(C'_2)$.

Bây giờ tìm iv, mình chỉ cần đưa lên server 32 bytes \0 là xong và cũng tương tự trên, chỉ lấy ciphertext nào mà 16 bytes cuối không bị ẩn. Vì $C_2 = E_k(C_1) = E_k(E_k(iv))$ (do xor với dãy toàn 0). Decrypt mình có lại iv.

Giờ thì giải mã flag thôi.

Flag: CCTF{h0W_R3c0VER_7He_5eCrET_1V?}.

Rima

Bài này không dùng biến chữ để chạy loop và dùng dấu gạch dưới của python nên lúc đầu mình thấy hơi rắc rối.

Đầu tiên flag được chuyển sang dạng nhị phân và thêm 1 bit 0 ở đầu được dãy f . Kế tiếp với mỗi $i = 0, \dots, \text{len}(f) - 2$ thì $f_i = f_i + f_{i+1}$.

Kế tiếp hai số a và b được tạo là hai số nguyên tố kế tiếp tính từ $\text{len}(f)$ là độ dài f .

Sau đó, g và h là hai list tạo ra từ việc lặp f lần lượt với a và b lần. Như vậy độ dài của g là $a \cdot \text{len}(f)$ và độ dài của h là $b \cdot \text{len}(f)$.

Tiếp theo, c là số nguyên tố kế tiếp tính từ $\text{len}(f) \gg 2$. Thêm c bit 0 vào đầu g và thực hiện $g_i = g_i + g_{i+c}$ với $i = 0, 1, \dots, \text{len}(f) - c - 1$. Làm tương tự với h .

Cuối cùng là chuyển g và h sang số int base 5 và viết lên file dưới dạng byte. Nên đầu tiên mình sẽ làm ngược lại và tìm được g và h , sau đó mình bruteforce $\text{len}(f)$ để tìm a , b và c và xem thử bộ nào thỏa $a \cdot \text{len}(f) + c = \text{len}(g)$ và $b \cdot \text{len}(f) + c = \text{len}(h)$.

Sau đó là làm ngược lại quá trình, với $i = \text{len}(f) - c - 1, \dots, 0$ thì $g_i = g_i - g_{i+c}$. Tương tự với h . Có thể kiểm chứng cách đúng nếu đầu g có đúng c số 0. :))

Giờ thì, lấy $\text{len}(f)$ số đầu của g và tiếp tục làm ngược lại sẽ ra các bit của flag.

Flag: CCTF{_how_f1nD_7h1s_1z_s3cr3T?!}.

Maid

Ở bài này server cung cấp cho mình các chứng năng sau:

- encrypt một số bất kì không vượt quá 2^{2048-2} bits;
- decrypt một số bất kì (hàm decrypt bị giấu đi);
- lấy flag bị mã hóa.

Key là một cặp khóa công khai-bí mật ($pubkey, privkey$), trong đó $pubkey = p^2q$ còn $privkey = p^2$, với p và q là hai số nguyên tố 1024 bits và đồng dư 3 modulo 4.

Hàm `encrypt` thực hiện mã hóa số m bằng cách trả về $m^2 \pmod{pubkey}$. Còn hàm `decrypt` thực hiện giải mã chỉ cần $privkey$.

Kì cục

Thế quái nào

Mà `encrypt` cần cả p và q còn `decrypt` chỉ cần p ?

Thật ra là vì nếu $c \equiv m^2 \pmod{\text{pubkey}}$ thì $c \equiv m^2 \pmod{p^2}$, như vậy giải thích cho việc m không được vượt quá $2048 - 2$ bits và việc giải mã chỉ cần p . Như vậy cách attack của mình như sau:

1. Chọn ngẫu nhiên các ciphertext, gửi lên để server decrypt và nhận lại các plaintext tương ứng. Ta biết rằng $c \equiv m^2 \pmod{p^2}$ nên $p^2 = \gcd(m_1^2 - c_1, m_2^2 - c_2)$, từ đó lấy căn bậc hai là có p .
2. Tiếp theo, chọn ngẫu nhiên các plaintext, gửi lên server encrypt và nhận lại ciphertext tương ứng. Do

$$c \equiv m^2 \pmod{\text{pubkey}} \equiv m^2 \pmod{p^2 q},$$

- khi đó $p^2 q = \gcd(m_1^2 - c_1, m_2^2 - c_2)$. Việc này ngược lại quá trình trên vì như nãy mình đã nói, encrypt sử dụng $p^2 q$ còn decrypt thì chỉ cần p^2 ;
- và bây giờ p và q đã có đủ, ta decrypt và có flag thôi.

Flag: CCTF{__Ra8!N_H_Cryp70_5YsT3M__}.

Tuti

Ở đây x và y là nửa đầu và nửa sau của flag, k là một số cho trước thỏa mãn

$$(x^2 + 1)(y^2 + 1) - 2(x - y)(xy - 1) = 4(k + xy).$$

Biến đổi một tí mình có

$$\begin{aligned} & x^2y^2 + x^2 + y^2 + 1 - 2(x - y)xy + 2(x - y) - 4xy &= 4k \\ \Leftrightarrow & x^2y^2 + (x^2 + y^2 + 1 + 2(x - y) - 2xy) - 2(x - y + 1)xy &= 4k \\ \Leftrightarrow & x^2y^2 + (x - y + 1)^2 - 2(x - y + 1)xy &= 4k \\ \Leftrightarrow & (xy - x + y - 1)^2 &= 4k. \end{aligned}$$

Như vậy $xy - x + y - 1 = \sqrt{4k}$ mà $xy - x + y - 1 = (x + 1)(y - 1)$ nên mình chỉ cần factor số này là tìm được x và y . Do có thể có nhiều cách chọn nên mình tìm cái "hợp lí" nhất.

Flag: CCTF{S1mPL3_4Nd_N!cE_Diophantine_EqUa7I0nS!}.

Improve

Ở bài này khá có khá nhiều thứ linh tinh nhưng chung quy lại là mình cần nhập vào hai số khác nhau m_1 và m_2 khác n sao cho kết quả hàm `improve` với hai số này là giống nhau.

Các tham số sẽ là n làm chẵn trên cho hai số nhập vào (không vượt quá n^2 , và f có tính chất quan trọng là luôn chẵn, đây là tiền đề để mình giải bài này).

Hàm `improve` mình để ý rằng L được tạo ra sau vài biến đổi từ $e = m^f \pmod{n^2}$, mà mình cần hai số m cho cùng L , vậy chỉ cần cho ra cùng e là xong. Như hồi nãy mình có đề cập là f luôn chẵn, vậy chỉ cần chọn m và $n^2 - m$ là xong. :))

Flag: CCTF{Phillip_NOW_4_pr0b4b1liStiC_aSymM3Tr1C_AlGOrithM!!}.

Onlude

Bài này mình giải trước khi hết thời gian 1 tiếng và là bài cuối cùng mình giải được. Hàm `prepare` chuyển flag thành ma trận A , `key` là bộ ba ma trận R , L và U . Ma trận $S = LU$.

Việc mã hóa như sau:

- với R , L và U như trên, tính $S = LU$;
- tính $X = A + R$, $Y = SX$ và ciphertext là $E = L^{-1}Y$.

Đề cho mình bốn ma trận:

- E là ciphertext, với một chút biến đổi mình có $E = U(A + R)$;
- ma trận LUL ;
- ma trận

$$L^{-1}S^2L = L^{-1}(LU)^2L = L^{-1}LULUL = (UL)^2,$$

mình đặt là T ;

- ma trận $R^{-1}S^8$, mình đặt là W .

Lấy LUL nhân với T mình có $LUL \cdot (UL)^2 = L(UL)^3$.

Khi đó

$$T^2[L(UL)^3]^{-1} = (UL)^4 \cdot (UL)^{-3} \cdot L^{-1} = UL \cdot L^{-1} = U.$$

Vậy là mình có U rồi. :))

Quay lại $E = U(A + R)$, khi đó $R = U^{-1}E - A$, nhân bên phải của hai vế với W thì

$$R \cdot R^{-1}S^8 = (U^{-1}E - A)W = S^8 = (LU)^8.$$

Quay lại một chút,

$$(LUL) \cdot T^3 = (LUL) \cdot (L^{-1}S^6L) = (LU)^7 \cdot L,$$

suy ra $(LUL) \cdot T^3 \cdot U = (LU)^8$.

Từ đó mình dễ dàng tìm lại được $A = U^{-1}E - (LU)^8W^{-1}$.

Thực hiện tương tự hàm `prepare` mình có được flag.

Flag: CCTF{LU__D3c0mp0517Ion__4L90?}.

Writeup đến đây là hết, cảm ơn các bạn đã đọc.

4.3.2 Google CTF 2021

Pythia

Đề bài ở file `service.py`.

Ở bài này random ba password từ các ký tự in thường, mỗi password có độ dài là 3.

Nhiệm vụ của chúng ta là tìm ba password này và ghép lại (theo thứ tự) để lấy flag.

Trong 150 queries, chúng ta có thể làm một trong ba việc:

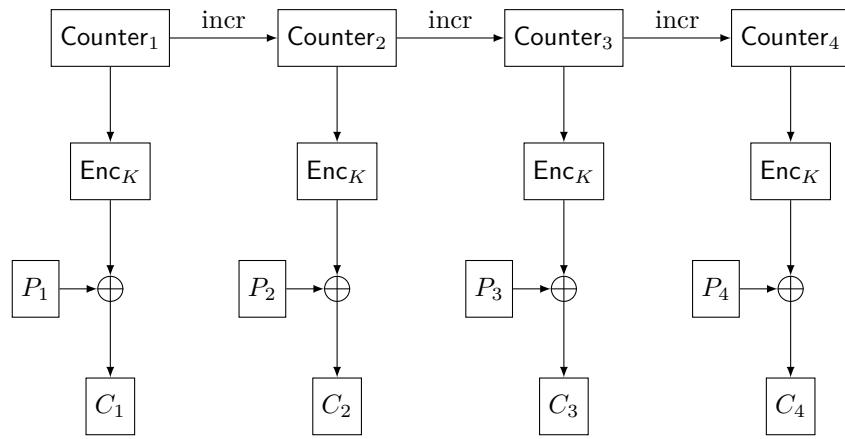
- chọn password để decrypt;
- thử password để lấy flag;
- gửi nonce và ciphertext để kiểm tra xem có thể decrypt bằng AES-GCM không.

Chúng ta sẽ cần hiểu cách AES-GCM hoạt động.

Counter mode (CTR)

Chúng ta bắt đầu với CTR trước.

Ở đây một bộ đếm (counter) được sử dụng. Chúng ta có thể dùng bộ đếm cơ bản 0, 1, 2, ... hoặc các bộ đếm phức tạp hơn. Mô hình mã hóa với khóa K là $C_i = P_i \oplus \text{Enc}_K(\text{counter}_i)$. Việc dùng bộ đếm nào cũng không quá quan trọng, qua thuật toán AES thì đều khó cǎ.



Trong AES-GCM, CTR đóng vai trò mã hóa.

Authenticated encryption (AEAD)

Một vấn đề quan trọng đối với mã hóa đối xứng (stream cipher và block cipher) là làm sao để kiểm tra thông tin có bị sửa đổi không?

Khi đó chúng ta thêm một trường dữ liệu gọi là associated data. Do đó mô hình mã hóa được gọi là authenticated encryption with associated data (AEAD). Chúng ta cần định nghĩa sau:

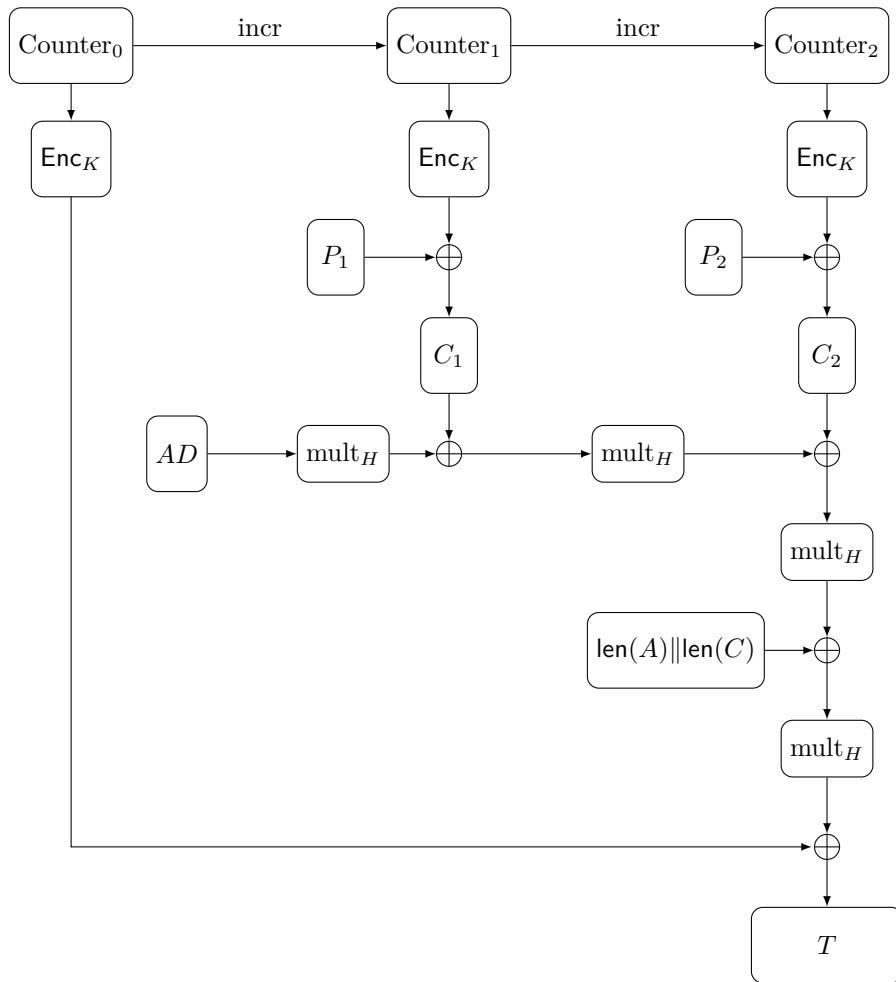
Message authentication code (MAC) là một hàm bất kì, kí hiệu là $\text{MAC}_K(N, C, A^d)$. Sử dụng secret key K trao đổi bởi Alice và Bob, hàm lấy nonce N , ciphertext C và associated data A^d nào đó để tạo ra tag T . Dựa vào tag T này Bob kiểm tra tính hợp lệ của ciphertext C do Alice gửi tới.

Chúng ta có thể hiểu nonce là một giá trị khởi đầu nào đó.

Ví dụ, giả sử Alice muốn gửi cho Bob plaintext P . Alice chọn nonce N và encrypt plaintext với $C = E_K(N, P)$. Sau đó Alice tạo tag $T = \text{MAC}_K(N, C, A^d)$ và gửi message $M = \{N, C, T\}$ tới cho Bob.

Giả sử Bob nhận được message $M' = \{N', C', T'\}$. Bob tính $\tau = \text{MAC}_K(N', C', A^d)$ và so sánh với T' . Nếu $\tau = T'$ thì ciphertext không bị sai lệch, từ đó Bob có thể decrypt $P' = D_K(N', C')$.

Galois/Counter mode (GCM)



Ở GCM:

- CTR được sử dụng để mã hóa các khối plaintext P_0, P_1, \dots, P_{n-1} thành các khối ciphertext C_0, C_1, \dots, C_{n-1} ;
- các khối ciphertext kết hợp với associated data để tạo ra tag. Tất cả việc tính toán qua hàm mult_H hay $C_i \oplus \text{mult}_H$ được thực hiện trên $\mathbb{F}_{2^{128}}$ với đa thức tối giản là $f(x) = x^{128} + x^7 + x^2 + x + 1$. Lý do của việc này là mỗi block của AES 16 byte, tương đương 128 bit, nên ta có thể chuyển đổi từ khối 16 bytes thành đa thức thuộc $\mathbb{F}_{2^{128}}$.

Giả sử mình có ciphertext C gồm n block, kí hiệu là

$$C = C_0 \| C_1 \| C_2 \| \cdots \| C_{n-1}.$$

Đối với bài CTF này thì associated data không được dùng nên mình sẽ bỏ qua.

Vậy H trong chỗ mult_H là gì? Ở đây $H = \text{AES}_K(0^{128})$, nghĩa là encrypt một dãy gồm 128 bits 0.

Quá trình tính tag diễn ra như sau.

Với lần mult_H đầu, do không có associated data nên giá trị khởi tạo là 0^{128} .

Với lần mult_H thứ hai, $(0^{128} \oplus C_0) \cdot H = C_0 H$.

Với lần mult_H thứ ba, $(C_0 H \oplus C_1) \cdot H = C_0 H^2 \oplus C_1 H$.

Cứ tiếp tục như vậy nhưng để ý hai lần mult_H cuối cùng.

Độ dài toàn bộ associated data tính theo bit được biểu diễn bởi số nguyên $\text{len}(A)$ độ dài 64 bits. Tương tự, độ dài toàn bộ ciphertext tính theo bit được biểu diễn bởi số nguyên 64 bits là $\text{len}(C)$. Vì ở đây không có associated data nên $\text{len}(A) = 0$ và $\text{len}(C) = 128 \cdot n$ với n là số lượng khối và mỗi khối có 128 bits. Như vậy chúng ta có 128 bits kết quả cho $\text{len}(A)\|\text{len}(C)$ là $0^{64}\|(128 \cdot n)$.

Với lần kế cuối ta sẽ có

$$C_0 H^{n+1} \oplus C_1 H^n \oplus \cdots \oplus L H$$

với $L = 0^{64}\|(128 \cdot n)$.

Với lần cuối ta cộng thêm encrypt của counter_0 nữa là xong.

Vậy kết quả cuối cùng của toàn bộ quá trình là tag

$$T = C_0 H^{n+1} + C_1 H^n + \cdots + C_{n-1} H^2 + L H + \text{AES}_K(J_0).$$

Theo cách chọn nonce của AES-GCM thì nonce có độ dài 12 bytes (96 bits) và

$$J_0 = IV \| 0^{31} \| 1.$$

trong đó IV là nonce.

Quay lại bài toán

Để giải bài này mình cần làm như sau:

- cố định IV và T ;
- mình chia tập hợp khóa thành hai nửa trái phải (tìm nhị phân) và kiểm tra xem key (trong bài là password) nằm ở nửa nào cho tới khi không gian key chỉ còn 1.

Tới đây, mục đích của mình là tìm các ciphertext C_0, C_1, \dots, C_{n-1} sao cho với tất cả key trong một nửa trái đều cho ra cùng một tag.

Khi mình gửi ciphertext và tag này lên server, nếu server trả về **Decryption successful** nghĩa là key cần tìm nằm trong nửa trái, nếu fail nghĩa là key nằm ở nửa phải.

Để tìm được các ciphertext như vậy mình sử dụng nội suy Lagrange (Lagrange interpolation).

Mình thấy rằng

$$T = C_0 H^{n+1} + C_1 H^n + \cdots + C_{n-1} H^2 + L H + \text{AES}_K(J_0).$$

Tương đương với

$$C_0 H^{n-1} + C_1 H^{n-2} + \cdots + C_{n-1} = (L H + \text{AES}_K(J_0) + T) \cdot H^{-2}.$$

Với mỗi K_i thuộc nửa trái mình có $H_i = \text{AES}_{K_i}(0^{128})$ và $\text{AES}_{K_i}(J_0)$ tương ứng.

Đặt $f(x) = C_0 x^{n-1} + C_1 x^{n-2} + \cdots + C_{n-1}$. Đa thức này thỏa mãn với mọi key K_i thuộc nửa trái thì

$$f(H_i) = (L H_i + \text{AES}_{K_i}(J_0) + T) \cdot H_i^{-2}.$$

Lưu ý rằng để tìm đa thức $f(x)$ bậc m thì cần $m+1$ cặp $(x_i, f(x_i))$. Do đó ở đây ta chọn $n = \text{len}(keys)$ với $keys$ là tập chứa tất cả key của nửa trái.

Từ đó với n key mình sẽ tìm được $f(x)$ (vì $f(x)$ có bậc $n-1$).

Hàm encrypt một block AES. Hàm chuyển đổi từ block 16 byte sang đa thức thuộc $\mathbb{F}_{2^{128}}$ và ngược lại

```

def aes_ecb(key, plaintext):
    return AES.new(key, AES.MODE_ECB).encrypt(plaintext)

def byte_to_pol(block):
    n = int.from_bytes(block, 'big')
    nn = list(map(int, bin(n)[2:].zfill(128)))
    pol = sum(j*x**i for i, j in enumerate(nn))
    return F(pol)

def pol_to_byte(element):
    coeff = element.polynomial().coefficients(sparse=False)
    coeff = coeff + [0] * (128 - len(coeff))
    num = int(''.join(list(map(str, coeff))), 2)
    return int.to_bytes(num, 16, 'big')

```

Hàm attack tìm ciphertext với danh sách key, nonce và tag

```

def find_poly(keys, nonce, tag):
    points = []
    L = byte_to_pol(b'\x00' * 8 + int.to_bytes(128 * len(keys), 8, 'big'))
    T = byte_to_pol(tag)
    N = nonce + b'\x00' * 3 + b'\x01'

    for key in keys:
        Hi = byte_to_pol(aes_ecb(key, bytes(16)))
        Bi = byte_to_pol(aes_ecb(key, N))
        fHi = ((L * Hi) + Bi + T) * Hi**(-2)
        points.append((Hi, fHi))

    lagrange = R.lagrange_polynomial(points)

    coeff = lagrange.coefficients(sparse=False)[::-1]

    C = b''.join([pol_to_byte(c) for c in coeff])

    return C

```

LƯU Ý 1. Do không gian key ban đầu khá lớn (2^{63}) nên việc tìm nhị phân ngay từ đầu khá là khoai (đa thức bậc $26^3/2 = 8788$) nên mình chia ra các chunk key dài 512 để tìm chunk nào chứa key (bản chất không thay đổi). Sau đó từ mỗi chunk mình mới dùng tìm nhị phân mò key.

LƯU Ý 2. Việc tính toán đa thức bậc 512 cũng tốn thời gian nên chúng ta có thể tính trước rồi lưu lại trên dict hoặc hash table hoặc bất cứ thứ gì bạn nghĩ ra :v sau đó chúng ta mới giao tiếp với server.

Phần còn lại của code là attack thôi :))))

❶ solve.py

```

from pwn import remote, process, context
from sage.all import *
from Crypto.Cipher import AES
from itertools import product
import string
from base64 import b64encode, b64decode

```

```

from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
from cryptography.hazmat.primitives.kdf.scrypt import Scrypt
from tqdm import tqdm
import os

# context.log_level = 'Debug'

F2 = GF(2)[x]
x = F2.gen()
modulus = x**128 + x**7 + x**2 + x + 1
F = GF(2**128, 'x', modulus=modulus)
R = PolynomialRing(F, 'z')
z = R.gen()

NONCE = b'\x00' * 12
TAG = b'\x00' * 16

possible_keys = []
keys = []
for a, b, c in product(string.ascii_lowercase, repeat=3):
    kdf = Scrypt(salt=b'', length=16, n=2**4, r=8, p=1, backend=default_backend())
    password = bytes(a+b+c, 'UTF-8')
    key = kdf.derive(password)
    possible_keys[key] = password
    keys.append(key)

p = process(["python3", "service.py"])

CACHE = {}
chunk_size = 512
for i in tqdm(range(0, len(keys), chunk_size)):
    test_keys = [key for key in keys[i:i+chunk_size]]
    C = find_poly(test_keys, NONCE, TAG)
    CACHE[i] = C
    aes = AES.new(test_keys[0], AES.MODE_GCM, nonce=NONCE)
    aes.decrypt_and_verify(C, TAG)

def attack(idx):
    p.sendlineafter(b">>> ", b"1")
    p.sendlineafter(b">>> ", str(idx).encode())
    # phase 1
    index = 0
    for cache in CACHE:
        C = CACHE[cache]
        payload = b64encode(NONCE) + b"," + b64encode(C + TAG)
        p.sendlineafter(b">>> ", b"3")
        p.sendlineafter(b">>> ", payload)
        p.recvline()
        if b'success' in p.recvline():
            index = cache
            break
    print(cache)

```

```

# phase 2
test_keys = [key for key in keys[cache:cache + chunk_size]]
while len(test_keys) > 1:
    C = find_poly(test_keys[:len(test_keys) // 2], NONCE, TAG)
    payload = b64encode(NONCE) + b", " + b64encode(C + TAG)
    p.sendlineafter(b">>> ", b"3")
    p.sendlineafter(b">>> ", payload)
    p.recvline()
    if b"success" in p.recvline():
        test_keys = test_keys[:len(test_keys) // 2]
    else:
        test_keys = test_keys[len(test_keys) // 2:]
return test_keys[0]

password = b""

for _ in range(3):
    password += possible_keys[attack(_)]

p.sendlineafter(b">>> ", b"2")
p.sendlineafter(b">>> ", password)
print(p.recvline())
print(p.recvline())
p.close()

```

Với mỗi password mình dùng một query để chỉ định vị trí password (0, 1, 2), $\lceil 26^3 / 512 \rceil = 35$ query cho mỗi chunk, và $\log_2(512) = 9$ cho tìm nhị phân. Như vậy mình tốn $(1 + 35 + 9) \cdot 3 = 135$ query tổng cộng.

Bài viết tới đây là hết. Cám ơn các bạn đã đọc.

4.3.3 NSUCRYPTO 2021

Elliptic curve points

Đề bài

Đặt E/\mathbb{F}_p là đường cong elliptic với dạng Weierstrass. Đường cong này có phương trình $y^2 = x^3 + ax + b$, với $a, b \in \mathbb{F}_p$ và $4a^3 + 27b^2 \neq 0$. Các điểm affine trong E và điểm vô cực \mathcal{O} tạo thành một nhóm Abel, đặt

$$E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p^2 : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

Giả sử $b = 0$. Gọi $R \in E(\mathbb{F}_p)$ là điểm với order lẻ và $R \neq \mathcal{O}$. Xét $H = \langle R \rangle$ là nhóm con sinh bởi R .

Giúp Alice chứng minh rằng nếu $(u, v) \in H$ thì u là số chính phương modulo p .

Giải

Gọi $Q = (x', y') \in H$. Do R có order lẻ nên Q cũng có order lẻ (order của phần tử trong nhóm con chia hết order của nhóm con đó).

Giả sử order của Q là n lẻ. Xét điểm $P = \frac{n+1}{2}Q$, do n lẻ nên dễ thấy $P \in H$. Đặt $P = (x, y)$.

Do $2P = (n + 1)Q = Q$ (do Q có order là n) nên theo công thức cộng hai điểm trên elliptic ta có

$$\begin{aligned}x' &= \left(\frac{3x^2 + a}{2y} \right) - 2x \\&= \frac{9x^4 + 6x^2a + a^2 - 8x(x^3 + ax)}{(2y)^2} \\&= \frac{x^4 - 2x^2a + a^2}{(2y)^2} \\&= \left(\frac{x^2 - a}{2y} \right).\end{aligned}$$

Biểu thức cuối cho thấy rằng x' là số chính phương modulo p .

2021-bit key

Đề bài

Một generator dùng pseudo-random để sinh ra một dãy bit (0 hoặc 1) từng bước một. Để bắt đầu generator, một người phải trả 1 *nsucoin* và generator sẽ sinh ngẫu nhiên một bit (dãy bit độ dài bằng 1). Khi đó, với một dãy S được sinh có độ dài l , $l \geq 1$, một trong các động tác sau được thực hiện:

1. Một dãy ngẫu nhiên độ dài 4 được thêm vào S , khi đó dãy S' có độ dài $l + 4$. Việc này tốn 2 *nsucoin*.
 2. Một dãy ngẫu nhiên độ dài $2l$ được thêm vào S , khi đó dãy S' có độ dài $3l$. Việc này tốn 5 *nsucoin*.
- Bob cần sinh độ dài chính xác 2021 bit. Số lượng *nsucoin* nhỏ nhất để thực hiện việc này là bao nhiêu?

Giải

Để thấy rằng khi $l > 2$ thì sử dụng động tác thứ hai khiến độ dài dãy tăng lên rất nhanh như lại tốn thêm coin.

Như vậy khi $l > 6$ mình sẽ chứng minh rằng động tác thứ hai hiệu quả hơn.

Để ý rằng nếu mình sử dụng động tác thứ nhất ba lần liên tiếp, khi đó từ dãy có độ dài l mình thu được dãy mới độ dài $l + 4 + 4 + 4 = l + 12$ và việc này tốn 6 *nsucoin*.

Trong khi đó, nếu mình dùng động tác thứ hai một lần thì từ dãy độ dài l mình thu được dãy độ dài $3l$ và tốn 5 *nsucoin*.

Mình muốn $3l > l + 12$ vì mình đã có $3l$ tương ứng 5 *nsucoin* và $l + 12$ tương ứng 6 *nsucoin*. Như vậy $l > 6$.

Chiến thuật lúc này là mình sẽ dùng động tác thứ hai để triple độ dài bất cứ lúc nào có thể. Nói cách khác là khi đi ngược từ 2021 về 0 thì khi nào số chia hết cho 3, mình sẽ dùng động tác thứ hai, không thì dùng động tác thứ nhất.

Quá trình sẽ diễn ra theo bảng sau.

Phép tính	Số nsucoin thu được
$2021 - 4 = 2017$	2 nsucoin
$2017 - 4 = 2013$	2 nsucoin
$2013/3 = 671$	5 nsucoin
$671 - 4 = 667$	2 nsucoin
$667 - 4 = 663$	2 nsucoin
$663/3 = 221$	5 nsucoin
$221 - 4 = 217$	2 nsucoin
$217 - 4 = 213$	2 nsucoin
$213/3 = 71$	5 nsucoin
$71 - 4 = 67$	2 nsucoin
$67 - 4 = 63$	2 nsucoin
$63/3 = 21$	5 nsucoin

Từ 21 trở đi không thể áp dụng quy tắc trên nữa vì theo chứng minh trên chiến thuật chỉ hiệu quả với $l > 6$.
Mình khai triển $21 = 1 + 4 + 4 + 4 + 4 + 4$, thực hiện 4 phép trừ (động tác đầu) tốn $5 \cdot 2 = 10$ nsucoin và trừ 1 tốn 1 nsucoin.

Như vậy tổng số nsucoin nhỏ nhất cần là 47.

4.4 Các cuộc thi năm 2022

4.4.1 DownUnder CTF 2022

Baby ARX

Bài này thuộc dạng stream cipher với việc hai thằng kè nhau sẽ đưa ra một ciphertext.

Ở đây ta thấy rằng, với p_i và p_{i+1} là hai ký tự của plaintext sẽ cho ra ký tự c_i của ciphertext.

Như vậy mình đã biết một đoạn plaintext ban đầu là DUCTF rồi nên phần còn lại là bruteforce từng ký tự và so sánh với ciphertext để ra được ký tự ban đầu.

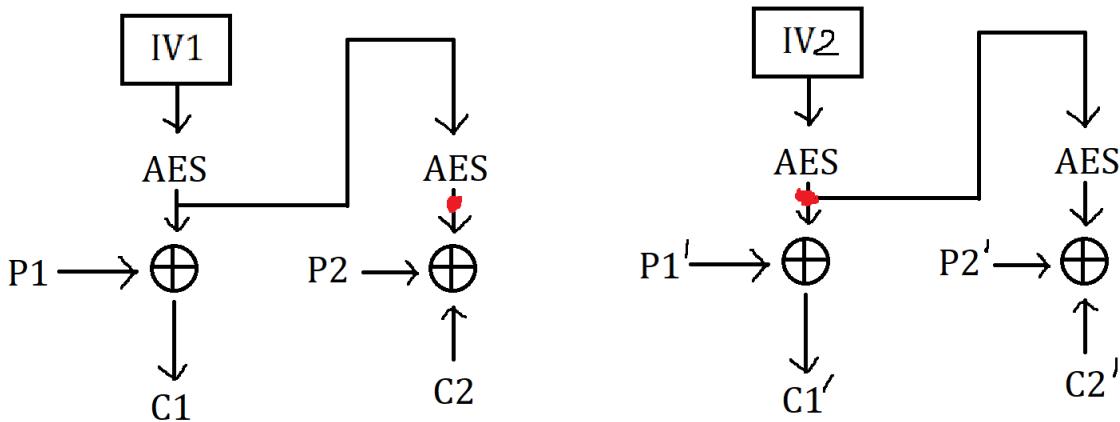
OFB

Một bài mã khối gây rối loạn tiền định.

Ở đây mỗi lần netcat lên server sẽ random 300 bytes và nhét vào giữa đoạn Decrypt this... (có dấu cách ở cuối nữa nha!!!!) và FLAG. Thêm nữa mỗi lần netcat lên sẽ random một khóa AES nên chúng ta sẽ chỉ tấn công trong một session netcat.

Mình được phép kiểm soát hai IV cho hai lần encrypt. Sau mỗi lần encrypt mình sẽ được nhận ciphertext. Do đó mình đoán rằng IV thứ hai sẽ có liên quan gì đó đến IV đầu và ciphertext đầu.

Trời không phụ lòng người, có công đoán mò có ngày làm nên. =)))) Mình đã đoán được $IV2$ để có thể decrypt.



Hình 4.1: OFB

Cách giải bài này nằm ở hai dấu chấm đỏ lõm trên hình vẽ.

Trong mode OFB, $C_i = P_i \oplus O_i$ với O_i là encrypt AES của O_{i-1} . Như vậy nếu mình chọn $IV2$ sao cho sau khi encrypt thì nó bằng đúng O_2 ở lần encrypt đầu, thì tất cả các O'_i sau đó luôn giống với O_i sau một đơn vị. Tức là $O'_i = O_{i+1}$, $i = 1, 2, \dots$

Trong OFB thì $C_i = P_i \oplus O_i$ và $C'_i = P'_i \oplus O'_i$. Do $O'_i = O_{i+1}$ theo cách chọn $IV2$ của mình nên mình sẽ có

$$C_{i+1} \oplus P_{i+1} = C'_i \oplus P'_i \iff P_{i+1} = C_{i+1} \oplus C'_i \oplus P_i,$$

mà P_1 mình có rồi nên mình sẽ suy ra được tất cả P_i còn lại với $i = 2, 3, \dots$

Flag mình tìm được:

CRT

Một bài discrete logarithm trên modulo đa thức.

Ở đây đề cho chúng ta hai đa thức được random là A và B (cả hai đều nằm trong modulo $f(x)$ trên $\mathbb{Z}_p[x]$). Với hai số random n và m bị giấu, đề cho mình đa thức $C = A^n \cdot B^m$.

Nhiệm vụ của mình phải tìm ba vector trong \mathbb{F}_p là φ_A , φ_B và φ_C sao cho tích có hướng của φ_A^n và φ_B^m bằng đúng φ_C . Ở đây φ_A^n nghĩa là mỗi phần tử của φ_A được mũ n (mod p), $\varphi_A = (x_A, y_A, z_A)$ thì

$$\varphi_A^n = (x_A^n \text{ mod } p, y_A^n \text{ mod } p, z_A^n \text{ mod } p).$$

Như vậy mình cần giải bài toán discrete logarithm cho polynomial ring. Phần khó của bài này là làm sao tìm được order của mỗi đa thức. Mình thấy rằng $f(x)$ có thể factor thành tích của ba đa thức bậc 1. Như vậy việc giải bài toán trên modulo $f(x)$ trở thành giải bài toán trên từng đa thức bậc 1 rồi CRT chúng lại với nhau.

Nhưng mình không biết cách tìm order của mỗi đa thức

Chán thật, mình chọn $\varphi_A = \varphi_B = \varphi_C = (0, 0, 0)$ và nó luôn đúng chả quan tâm n, m chi cả. Unintended solution. =)))

Cám ơn mọi người đã đọc writeup của mình. Hẹn gặp lại.

4.4.2 NSUCRYPTO 2022

Lời nói đầu

Thời kì đen tối ...

Đề thi năm 2022 khá lạ và phức tạp. Bảng điểm round 2 dành cho University student cũng ảo ma không kém.

Super dependent S-box

Đề bài

Harry muốn tìm một super dependent S-box cho mã hóa mới. Anh ấy dùng một hoán vị liên kết chặt chẽ với mọi biến của nó và muốn ước lượng số các hoán vị như vậy.

Một hàm boolean vectorial $F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_n(\mathbf{x}))$, với $\mathbf{x} \in \mathbb{F}_2^n$, là một **hoán vị** trên \mathbb{F}_2^n nếu nó là một ánh xạ one-to-one từ \mathbb{F}_2^n tới \mathbb{F}_2^n .

Các hàm tọa độ $f_k(\mathbf{x})$ (là các hàm boolean từ \mathbb{F}_2^n tới \mathbb{F}_2) được gọi là *essential depend* trên các biến x_j nếu tồn tại các giá trị $b_1, b_2, \dots, b_{j-1}, b_{j+1}, \dots, b_n \in \mathbb{F}_2$ sao cho

$$f_k(b_1, b_2, \dots, b_{j-1}, 0, b_{j+1}, \dots, b_n) \neq f_k(b_1, b_2, \dots, b_{j-1}, 1, b_{j+1}, \dots, b_n).$$

Nói cách khác, essential depend trên biến x_j nghĩa là trong dạng biểu diễn ANF (đa thức Zhegalkin) của hàm f có sự có mặt của biến x_j .

Example 22

Xét $n = 3$. Khi đó hàm boolean $f(x_1, x_2, x_3) = x_1x_2 \oplus x_3$ essential depend trên cả ba biến, nhưng hàm $g(x_1, x_2, x_3) = x_1x_2 \oplus x_2 \oplus 1$ chỉ essential depend trên x_1 và x_2 .

Câu hỏi. Tìm số lượng hoán vị trên \mathbb{F}_2^n mà các hàm tọa độ của nó đều essential depend trên cả n biến.

Q1. Tìm đáp án cho $n = 2, 3$.

Q2. Tìm đáp án cho n (special prize).

Giải

Q1 có thể được giải với SageMath. Tuy nhiên năm đó mình không dùng SageMath mà dùng Python thuận tiện đáp án sai mất. Đáp án cho Q1 với $n = 2$ là 0 (liệt kê tất cả hàm ra) và đáp án cho $n = 3$ là 24576.

Ở Q2 các team giải ra chứng minh được rằng, đáp án sẽ là một số chia hết cho $2^n \cdot n!$.

A long awaited event

Đề bài

Bob nhận được một thông điệp

L78V8LC7GBEYEE

về một sự kiện quan trọng từ Alice.

Alice sử dụng một bảng chữ cái 37 ký tự gồm chữ cái từ A tới Z, số từ 0 tới 9 và dấu space. Các ký tự được encode như sau:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

U	V	W	X	Y	Z	0	1	2	3	4	5	6	7	8	9	SPACE	
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	

Để mã hóa Alice sử dụng hàm f sao cho $f(x) = ax^2 + bx + c \pmod{37}$ với số a, b, c nào đó và hàm f thỏa tính chất

$$f(x - y) - 2f(x)f(y) + f(1 + xy) = 1 \pmod{37},$$

với mọi số nguyên x, y .

Q. Hãy giải mã thông điệp Bob nhận được.

Giải

Sử dụng hàm f , ta có:

1. Nếu cho $y = 0$ thì

$$f(x) - 2f(x) \cdot f(0) + f(1) = 1 \pmod{37}$$

với mọi $x \in \mathbb{Z}_{37}$.

Điều này tương đương với

$$(1 - 2f(0)) \cdot f(x) = 1 - f(1) \pmod{37}$$

với mọi $x \in \mathbb{Z}_{37}$.

Đẳng thức trên đúng với mọi $x \in \mathbb{Z}_{37}$ khi và chỉ khi

$$1 - 2f(0) = 1 - f(1) = 0.$$

Suy ra $f(0) = 19$ và $f(1) = 1$.

Cho $x = y = 1$ thì

$$f(2) = 1 - f(0) - 2 \cdot (f(1))^2 = 21 \pmod{37}.$$

Với các cặp giá trị $(x, f(x))$ là $(0, 19)$, $(1, 1)$ và $(2, 21)$ thì ta tìm lại được (bằng nội suy hoặc phép thê) đa thức $f(x) = 19x^2 + 19 \pmod{37}$.

Khi đó, với mỗi giá trị $f(x)$ ta có hai giá trị x thỏa mãn đẳng thức. Truy ngược ra đáp án khả thi cho ta thông điệp ban đầu.

Ciphertext	L	7	8	V	8	L	C	7	G	B	E	Y	E	E
Plaintext 1	N	S	R	C	R	N	P	S	O	B	J	L	J	J
	13	18	17	2	17	13	15	18	14	1	9	11	9	9
Plaintext 2	Y	T	U	9	U	Y	W	T	X	SPACE	2	0	2	2
	24	19	20	35	20	24	22	19	23	36	28	26	28	28

Theo bảng tên thì thông điệp là NSUCRYPTO 2022.

4.4.3 Sekai CTF 2022

Mình làm 1 bài rồi đúng hình. :)))

FaLLProof

Đề bài có thể tải ở file `chall.py`.

Ở bài này hệ mật mã tạo một public key từ một secret và một hàm hash.

Tạo public key

Để tạo public key cần truyền vào một chuỗi byte secret và hàm hash. Chúng ta đã biết hàm hash luôn cho output có độ dài cố định không liên quan đến độ dài input.

Mình gọi hàm hash là $H(x)$ cho input x và độ dài (cố định) của hàm hash là L . Mình kí hiệu việc thực hiện hash k lần là $H^{(k)}(x)$. Tức là $H^{(k)}(x) = H(H(\dots(H(x))\dots))$ (k lần). Như vậy public key tương ứng với secret s đầu vào là $[H^{(1)}(s), H^{(2)}(s), \dots, H^{(4*L)}(s)]$.

Prototype của hàm tạo public key dùng SHA512 nhưng ở chương trình chính dùng SHA256. Do đó $L = 32$ bytes và độ dài public key là 64.

Encode message

Bước vào hàm encrypt, message ban đầu sẽ được pad thành bội của 32 (theo **độ dài public key / 4 - độ dài hàm hash**) và được chia thành từng chunk 32 bytes.

Happiness

Hàm **happiness** có một tính chất thú vị.

Giả sử mình có đầu vào là $x = x_n 2^n + x_{n-1} 2^{n-1} + \dots + x_1 2 + x_0$.

Khi đó:

$$\begin{aligned} x \gg 1 &= x_n 2^{n-1} + x_{n-1} 2^{n-2} + \dots + x_2 2 + x_1 \\ x \gg 2 &= x_n 2^{n-2} + x_{n-1} 2^{n-3} + \dots + x_3 2 + x_2 \\ &\dots \\ x \gg n &= x_n, \end{aligned}$$

suy ra

$$\begin{aligned} &x - (x \gg 1 + x \gg 2 + \dots + x \gg n) \\ &= x_n 2^n + x_{n-1} 2^{n-1} + \dots + x_1 2 + x_0 \\ &- [x_n(2^{n-1} + 2^{n-2} + \dots + 1) + x_{n-1}(2^{n-2} + \dots + 1) + \dots + x_1] \\ &= x_n \left(2^n - \frac{2^n - 1}{2 - 1}\right) + x_{n-1} \left(2^{n-1} - \frac{2^{n-1} - 1}{2 - 1}\right) + \dots + x_1(2 - 2 + 1) + x_0 \\ &= x_n + x_{n-1} + \dots + x_1 + x_0. \end{aligned}$$

Tóm lại hàm **happiness** tính tổng các bit của x . :)))

Encrypt

Giả sử mình gọi chunk đầu là m . Do m có 32 bytes, tương ứng 256 bit, nên mình viết dưới dạng nhị phân là

$$m = m_{255} 2^{255} + m_{254} 2^{254} + \dots + m_1 2 + m_0,$$

$m_i \in \{0, 1\}$.

Mình tiếp tục kí hiệu public key thứ j là p_j , $j = \overline{0, 127}$. Và mình cũng viết dưới dạng nhị phân

$$p_j = p_{j,255}2^{255} + p_{j,254}2^{254} + \cdots + p_{j,1}2 + p_{j,0}.$$

Do đó phép AND cho kết quả

$$(m_{255} \cdot p_{j,255}) \cdot 2^{255} + (m_{254} \cdot p_{j,254}) \cdot 2^{254} + \cdots + (m_1 \cdot p_{j,1}) \cdot 2 + (m_0 \cdot p_{j,0}).$$

Qua hàm *happiness* chính là tổng

$$m_{255} \cdot p_{j,255} + m_{254} \cdot p_{j,254} + \cdots + m_1 \cdot p_{j,1} + m_0 \cdot p_{j,0}.$$

Do đó với 128 public key mình có thể viết dưới dạng ma trận như sau

$$\begin{pmatrix} p_{0,255} & p_{0,254} & \cdots & p_{0,1} & p_{0,0} \\ p_{1,255} & p_{1,254} & \cdots & p_{1,1} & p_{1,0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{126,255} & p_{126,254} & \cdots & p_{126,1} & p_{126,0} \\ p_{127,255} & p_{127,254} & \cdots & p_{127,1} & p_{127,0} \end{pmatrix} \cdot \begin{pmatrix} m_{255} \\ m_{254} \\ \cdots \\ m_1 \\ m_0 \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ \cdots \\ c_{126} \\ c_{127} \end{pmatrix}.$$

Vấn đề ở đây là ma trận P không phải ma trận vuông. Do đó mình không thể tính nghịch đảo được :((

Để ý rằng m luôn không đổi mỗi lần netcat nên mình chỉ cần request thêm một lần nữa và có thêm 64 public key nữa. Khi đó ghép các public key đó nối tiếp xuống dưới ma trận P cũng như ghép thêm các ciphertext mới xuống dưới cột c thì mình đã đủ ma trận vuông để giải.

Code giải các bạn xem ở `solve.py`.

Cám ơn các bạn đã đọc writeup siêu dài cho một bài ... không dài lăm.

4.4.4 ISITDTU Quals 2022

Glitch in the matrix

Đề bài các bạn có thể xem ở `chall.py`.

Bài này mình re-writeup từ hint của của một idol. :)))))

Thử thách ở đây là cần đoán một token hex độ dài 64 bit với secret key là `SECRET_BASIS`.

Gọi token hex có 64 bit là $(m_1, m_2, \dots, m_{64})$. Với `SECRET_BASIS` là ma trận trên \mathbb{F}_2 kích thước 128×128 , ta thực hiện encrypt như sau:

- random một chuỗi 64 bit C ;
- nếu $m_i = 1$ thì thực hiện $f(\text{SECRET_BASIS}[:64], C)$, nếu là 0 thì $f(\text{SECRET_BASIS}[64:], C)$;
- nghĩa là secret key được chia thành hai nửa trái phải theo chiều dọc.

Hàm f thực hiện như sau:

- Với C là một chuỗi 64 bit, đặt $C = (c_1, c_2, \dots, c_{64})$;
- Nếu $c_i = 1$ thì dòng i được xor vào kết quả;
- Hay viết dưới dạng vector sẽ là

$$c_1 \cdot (m_{1,1}, m_{1,2}, \dots, m_{1,128}) + \cdots + c_{64} \cdot (m_{64,1}, m_{64,2}, \dots, m_{64,128}).$$

Ý tưởng để làm bài này là mình lấy thật nhiều ciphertext tương ứng với một plaintext (password). Do từng bit của plaintext được mã hóa riêng nhau nên mình sẽ tìm mối liên hệ giữa hai block mã hóa để xem chúng có cùng thuộc nửa trái (hoặc nửa phải) của secret key hay không.

Vì 64 bit plaintext được mã hóa thành 64×128 bit của ciphertext, mình chia mỗi ciphertext thành 64 block. Giả sử mình lấy n ciphertext, mỗi ciphertext cũng được chia thành 64 block thì mình sẽ có 64 ma trận $n \times 128$.

Nếu hai ma trận có chung base (cùng thuộc nửa trái hoặc nửa phải) thì hiệu của chúng sẽ có cùng rank với ma trận đầu. Do đó mình giả sử bit đầu là 0, vậy thì những block thứ 2 tới 64 (tương ứng bit thứ 2 tới 64) nếu có cùng rank thì sẽ mang bit 0, không thì mang bit 1. Do đó mình cần lấy n ciphertext đủ lớn để block đầu có rank là 64 (max).

Lưu ý rằng do mình giả sử bit đầu là 0, nên nếu bit đầu của plaintext là 1 thì sẽ không giải ra. Do đó xác suất cách làm này là $1/2$.

Code giải của mình ở `solve.py`.

4.5 Các cuộc thi năm 2023

4.5.1 NSUCRYPTO 2023

Affine cipher

Đây là bài 1 của round 2 và được giải bởi bạn Chương.

Đề bài

Ta xét bảng chữ cái $\{A, \dots, Z, \alpha, \beta, \gamma\}$ có 29 chữ cái. Ta đánh số A, \dots, Z từ 0 tới 25, và α, β, γ là 26, 27, 28.

Ta sử dụng cryptosystem mã hóa từng khối hai ký tự, gọi là bigram. Với x và y là hai ký tự của bigram, thì plaintext sẽ là $P = 29x + y$.

Mã hóa sử dụng biến đổi affine (giống hệ mã affine) là $C = aP + b \pmod{841}$.

Khi phân tích một đoạn văn bản dài, người ta phát hiện ra rằng các bigram sau xuất hiện nhiều nhất " $\beta\gamma$ ", "UM" và "LC". Đồng thời, trong tiếng Anh thì các bigram "TH", "HE" và "IN" cũng xuất hiện nhiều nhất.

Q. Có thể giải mã "KEUDCR" mà không cần khóa hay không? Còn key thì sao?

Giải

Theo thống kê các bigram xuất hiện nhiều nhất trong ciphertext và trong plaintext sẽ khớp nhau. Do đó có thể thấy "TH" mã hóa thành " $\beta\gamma$ " và "HE" mã hóa thành "LC". Như vậy ta có hệ phương trình

$$812 = a \cdot 558 + b \pmod{841}$$

$$321 = a \cdot 207 + b \pmod{841}$$

Giải hệ ta có $a = 15, b = 10$. Đây là key.

Từ đây chúng ta có thể giải mã thành **CRYPTO** là plaintext ban đầu. Bài này ăn trọn 4/4 điểm.

Simple ideas for primes

Đề bài

Chúng ta xem xét một số dãy số bao gồm các số nguyên tố.

- *Số Fermat*, $F_k = 2^{2^k} + 1$, với k bắt đầu từ 0. Ta có các số F_0, F_1, F_2, F_3, F_4 là các số nguyên tố, còn F_5 thì không phải.
- *Số Mersenne*, $M_k = 2^k - 1$. Ta có M_2, M_3, M_5, M_7 là các số nguyên tố, trong khi M_{11} là hợp số. Các số nguyên tố Mersenne là các số dạng $2^k - 1$ với k là số nguyên tố.

- Dãy số 31, 331, 3331, 33331, 333331, 3333331 là các số nguyên tố được xây dựng theo quy tắc trên, nhưng số 33333331 là hợp số chia hết cho 17.

Ta nói dãy Fermat trên có *sequence primality parameter* là 5, dãy Mersenne bằng 4, dãy cuối cùng bằng 7.

Q. Xây dựng một dãy bao gồm các số nguyên tố như vậy. Điều kiện quan trọng ở đây là các số hạng được xác định bởi chỉ số của dãy, không phụ thuộc vào các số trước nó.

Giải

Bắt đầu với dãy Euler

$$f(n) = n^2 + n + 41.$$

Đây là dãy các số nguyên tố với $n = 0, 1, \dots, 39$ và $f(40)$ là hợp số. Như vậy đây là dãy nguyên tố độ dài 40.

Và tất nhiên, dãy "ai cũng biết" thì chỉ được 2 điểm thôi. =(((

Sau khi tham khảo những thí sinh khác thì có một số cách xây dựng nhằm cải tiến điều này, tham khảo từ¹.

Nếu ta chuyển dãy trên thành

$$g(n) = f(n - 40) = (n - 40)^2 + (n - 40) + 41 = n^2 - 79n + 1601$$

thì thu được dãy số nguyên tố với độ dài 80. Các nhà toán học thế kỷ 20 đã chứng minh được rằng, nếu $p(x)$ là một đa thức sinh ra dãy số nguyên tố với $0 \leq x \leq n$ thì đa thức $p(n - x)$ cũng vậy.

Trong bảng, dãy nguyên tố có độ dài lớn nhất là 56 được biểu diễn bởi đa thức

$$\frac{1}{4} (n^5 - 133n^4 + 6729n^3 - 158379n^2 + 1720294n - 6823316).$$

Dựa theo lời giải của team Himanshu Sheoran, Yo Iida và Pranshu Kumar chúng ta có thể sinh một dãy có độ dài bất kì.

Lời giải dựa trên bài blog của A. W. Walker². Bài blog này phân tích về một bài báo năm 1977 bởi Chang và Lih với tiêu đề *Polynomial Representation of Primes* nhưng hiện tại không có bản online.

Bài báo này đưa ra một phương pháp xây dựng đa thức $F(n)$ bậc n mà với mọi $x \in [0, n]$ thì $F(n)$ đều là số nguyên tố.

Bài báo có thể được tóm gọn như sau. Xét đa thức

$$F(x) = 1 + \left| \sum_{n=0}^M \frac{a_n}{x - n} \prod_{j=0}^M (x - j) \right|$$

sẽ sinh ra các số nguyên tố với mọi $x \in [0, M]$ nếu và chỉ nếu a_n phân biệt và $(a_n \cdot M! + 1)$ là các số nguyên tố. Như vậy ta có một thuật toán đơn giản để bruteforce các đa thức trên.

Algorithm 4 (Thuật toán sinh dãy nguyên tố độ dài M)

Input: Độ dài dãy nguyên tố M

Output: Đa thức $F(x)$ cho kết quả là số nguyên tố với mọi $x \in [0, M]$

- coeffs = [] chứa các số hạng a_n

¹ <https://mathworld.wolfram.com/Prime-GeneratingPolynomial.html>

² <https://awwalker.com/2017/02/27/prime-generating-polynomials/>

2. $an \leftarrow 1$
3. While chưa đủ $M + 1$ số hạng trong coeffs
 1. If $(an \cdot M! + 1)$ là số nguyên tố
 1. kết nạp an vào dãy coeffs
 2. EndIf
 3. $an \leftarrow an + 1$
4. EndWhile

Mixed hashes

Đề bài

Alice và Bob trao đổi các thông điệp mã hóa. Họ dùng thuật toán mã hóa khối PRESENT với key 80-bit và ECB mode. Ở đây, thông tin được lưu dạng ảnh .ppm.

Header của file .ppm gồm 3 dòng theo dạng P6\nX\nY\n255. Trong đó X và Y là kích thước của ảnh theo chiều ngang và dọc.

Để đảm bảo an toàn, header sẽ được loại bỏ trước khi encrypt. Để có thể khôi phục header, hash của header sẽ được gửi đi thay vì header. Khi đó 3 phần của header sẽ được ngăn cách bởi dấu cách (space) thay vì newline như trên, nghĩa là P6 X Y 255.

Bob chuẩn bị 8 ảnh (trong file đính kèm) mà không có header. Bob encrypt 8 ảnh đó với cùng một key theo thuật toán PRESENT và ECB mode. Bob cũng gửi hash của 8 headers đi kèm. Tuy nhiên các hash đã bị trộn lẫn với nhau. Liệu chúng ta có thể khôi phục thông điệp mà Bob muốn gửi Alice?

Giải

Bài này là bài 3 ở round 1 và round 2. Trong thời gian 2 round mình đều giải ra (round 2 chi tiết hơn và trình bày đẹp hơn :v).

Đề cho một file mẫu là mikky.ppm. Khi phân tích file này mình thấy rằng, nếu gọi w và h là độ rộng và độ cao của ảnh (lấy từ header) thì độ dài file không có header là $3 \cdot w \cdot h$.

Sau khi encrypt bằng thuật toán mã hóa khối với ECB mode, độ dài sẽ là $3 \cdot w \cdot h + pd$, trong đó pd là padding. Theo thuật toán PRESENT thì $0 \leq pd \leq 8$.

Với dự đoán rằng $w \approx h$, mình lấy căn bậc hai của độ dài các file để cho, và đưa ra dự đoán $w, h \in [400, 600]$. Nếu sai thì mình tăng độ rộng khoảng này thôi.

Tiếp theo, bruteforce w và h trong khoảng này, cho tới khi hash "P6 x y 255" xuất hiện trong số các hash trên, và

$$0 \leq \text{len}(\text{ciphertext}) - 3 * w * h \leq 8$$

thì mình lấy w và h này. Thế là mình có header.

Do cả 8 file được encrypt bởi cùng một key PRESENT, và key có 80 bit tương ứng 10 bytes, hay 10 ký tự, nhìn đê mình nhận thấy có chuỗi P6 X Y 255 là hợp lý. Như vậy key cho PRESENT là chuỗi P6 X Y 255.

Cuối cùng, mình giải mã lần lượt từng file với key trên, ghép header tương ứng vào, như vậy là mình giải mã được tất cả file rồi.

Bài này được 5/6 điểm vì không nộp code tính toán header, mất điểm vì chủ quan.

Column functions

Đề bài

Alice muốn xây dựng mã đối xứng mạnh bằng việc giải một số bài toán khó.

Xét 2^n hàm vectorial one-to-one đôi một khác nhau, $G_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, với $i = 1, \dots, 2^n$. Sử dụng các hàm này, chúng ta xây dựng một ma trận binary đặc biệt và xác định một số tính chất của nó.

Với $n = 2^m$, $m \geq 5$, ta định nghĩa ma trận M kích thước $2^n \times n2^n$ theo quy tắc sau. Hàm thứ i , $i = 1, \dots, 2^n$, là ghép của các giá trị $G_i(0, 0, \dots, 0, 0)$, $G_i(0, 0, \dots, 0, 1)$, ..., $G_i(1, 1, \dots, 1, 1)$. Các cột của M có thể được xem như các vector của $n2^n$ hàm boolean, mỗi hàm n biến. Ta gọi chúng là *column functions*.

Chứng minh hoặc phản bác giả thuyết sau cho ít nhất một giá trị $m \geq 5$: với mọi cách xây dựng ma trận như trên, tồn tại $2^{n/2}$ columns functions $f_1, \dots, f_{2^{n/2}}$ sao cho tồn tại một hàm boolean nonzero $f : \mathbb{F}_2^{2^{n/2}} \rightarrow \mathbb{F}_2$ thỏa mãn các điều kiện sau:

- với mọi $\mathbf{x} \in \mathbb{F}_2^n$

$$f(f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_{2^{n/2}}(\mathbf{x})) = 0;$$

- với mọi $\mathbf{y} \in \mathbb{F}_2^{2^{n/2}}$, giá trị $f(\mathbf{y})$ có thể tính với không quá $2^{n/2}$ phép cộng và phép nhân modulo 2.

Example 23

Với $m = 1$ thì $n = 2$ và ta xây dựng ma trận 4×8 . Xét các hàm boolean vectorial one-to-one G_1, G_2, G_3, G_4 từ \mathbb{F}_2^n tới \mathbb{F}_2^n xác định bởi các giá trị $(0, 1, 2, 3)$, $(0, 2, 1, 3)$, $(0, 3, 1, 2)$ và $(3, 2, 1, 0)$. Khi đó ma trận là

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Ta cần tìm $2^{n/2} = 2$ column functions. Gọi f_1 và f_2 là hàm bool ứng với cột đầu và cột thứ hai của ma trận, và $f(x_1, x_2) = x_1 \oplus x_2$. Khi đó $f(f_1(\mathbf{x}), f_2(\mathbf{x})) \equiv 0$ vì $f_1(\mathbf{x}) = f_2(\mathbf{x}) = 0$ với mọi $\mathbf{x} \in \mathbb{F}_2^n$.

Ta cũng thấy rằng có thể chọn f_1 và f_2 là cột 5 và 6 của ma trận. Khi đó, đặt $f(x_1, x_2) = x_1 x_2$ thì $f(f_1(\mathbf{x}), f_2(\mathbf{x})) \equiv 0$ vì $f_1(\mathbf{x}) \neq f_2(\mathbf{x})$ với mọi $\mathbf{x} \in \mathbb{F}_2^n$.

Trong cả hai trường hợp ta chỉ cần đúng một phép tính. Lưu ý rằng hàm f thỏa f_1 và f_2 được gọi là *algebraically dependent*.

Giải

Cách giải của mình không đúng hoàn toàn nên chỉ được 1/8. Dưới đây trình bày cách giải của đội Robin Jadoul, Jack Pope và Esrever Yu được 8/8 điểm.

Giả thuyết trong đề bài đúng với m lớn. Ý tưởng chính là chúng ta cố định các cột sẽ chọn, có một số lượng lũy thừa các hàm f (rất rất lớn) nhưng chỉ có số lượng đa thức các hàng (ít lớn hơn), do đó chúng ta có thể chọn hàm f triệt tiêu tất cả hàng.

Theorem 16

Cho ma trận binary M kích thước $2^n \times n2^n$. Với $n + 1$ column functions bất kì, tồn tại một hàm nonzero $f : \mathbb{F}_2^{n+1} \rightarrow \mathbb{F}_2$ triệt tiêu trên các column functions và sử dụng nhiều nhất $2n + 1$ toán tử cộng và nhân.

❶ Chứng minh

Gọi f_1, f_2, \dots, f_{n+1} là các column functions. Đặt

$$S = \{(f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_{n+1}(\mathbf{x})) \in \mathbb{F}_2^{n+1} : \mathbf{x} \in \mathbb{F}_2^n\}$$

là tập hợp các bộ giá trị từ các column functions. Vì $|S| < |\mathbb{F}_2^{n+1}|$, ta có thể chọn vector $\mathbf{z} = (z_1, \dots, z_{n+1}) \in \mathbb{F}_2^{n+1} \setminus S$. Từ đây ta định nghĩa hàm f là

$$f(x_1, \dots, x_{n+1}) = (x_1 \oplus (z_1 \oplus 1)) \cdot (x_2 \oplus (z_2 \oplus 1)) \cdots (x_{n+1} \oplus (z_{n+1} \oplus 1))$$

Lúc này, f cho output là 1 chỉ khi \mathbf{z} là input (?), do đó với mọi $\mathbf{y} \in S$ thì $f(\mathbf{y}) = 0$.

Đối với số lượng phép tính, với mọi i , $z_i \oplus 1$ là hằng số nên ta không xét đến khi tính số lượng phép tính cho hàm f . Dựa trên vector \mathbf{z} , có tối đa $n+1$ phép cộng (tương ứng $x_i \oplus (z_i \oplus 1)$) và n phép nhân $n+1$ hạng tử với nhau. Vì vậy số phép tính tối đa là $2n+1$.

❶ Corollary 4

Giả thuyết trên đúng với mọi $m \geq 4$.

❶ Chứng minh

Theo định lý trên ta có f (bài toán yêu cầu hàm f lấy $2^{n/2}$ input. Tuy nhiên chúng ta có thể thêm các dummy input miễn là số lượng cột không lớn hơn $n+1$ - mục tiêu ban đầu). Bây giờ ta chỉ cần chặn lại số lượng cột mà ta cần và số lượng phép tính.

Với số lượng cột, ta cần $n+1 \leq 2^{n/2}$. Với số lượng phép tính, ta cần $2n+1 \leq 2^{n/2}$. Kết hợp cả hai ta có $n \geq 9$, tương đương với $m \geq 4$.

Bình luận

Về tổng thể cách giải của team kia hợp lý trừ phần mình đánh dấu, chưa hiểu lắm khi \mathbf{z} là input nghĩa là sao.

Thêm nữa, chưa có cơ sở để nghĩ ra bài giải này. Có vẻ như Esrever đã rất cẩn thận để giải ra. :)))

UPDATE. Ý nghĩa của phần đánh dấu (?) là do phép tính

$$f(x_1, \dots, x_{n+1}) = (x_1 \oplus (z_1 \oplus 1)) \cdot (x_2 \oplus (z_2 \oplus 1)) \cdots (x_{n+1} \oplus (z_{n+1} \oplus 1))$$

Kết quả phép tính ra 1 khi và chỉ khi tất cả hạng tử bằng 1. Nói cách khác $x_i \oplus (z_i \oplus 1) = 1$ với mọi $1 \leq i \leq n+1$.

Điều này tương đương với $x_i = z_i$ với mọi $1 \leq i \leq n+1$. Nhưng điều này vô lý vì khi đó $\mathbf{x} \equiv \mathbf{z}$ mà hai vector này nằm ở hai tập rời nhau.

Do đó kết quả hàm f luôn là 0. Ta có điều phải chứng minh.

Primes

Đây là bài 5 của round 2 và được giải bởi bạn Uyên.

Đề bài

Marcus chọn hai số nguyên tố lớn p và q rồi tính $n = p \cdot q$ và $m = p + q$. Sau đó số $n \cdot m$ được sử dụng trong cryptosystem.

Khi test Marcus thấy các số p và q cho tích $n \cdot m$ kết thúc bởi 2023. Điều đó khả thi không?

Giải

Do p và q là các số nguyên tố lớn nên chúng lẻ. Suy ra $m = p + q$ là số chẵn, nên tích $n \cdot m$ cũng là số chẵn. Số chẵn kết thúc bởi 0, 2, 4, 6, 8 nên không thể kết thúc bởi 3. Do đó điều này không thể xảy ra.

An aggregated signature

Bài này không biết làm. :(((

Đề bài

Trong một tổ chức quốc tế lớn, gọi là **NSUCRYPTO association**, mọi người quyết định tổ chức một tờ báo thông tin (news journal) trong lĩnh vực mật mã. Tổ chức muốn rằng, tin tức chỉ được công bố khi đã được kiểm duyệt bởi một nhóm lớn các nhà mật mã. Vì vậy, 10 000 chuyên gia mật mã đã được mời tới làm biên tập cho tờ báo.

Quy định công bố như sau. Tin tức được công bố khi nó được ký bởi tất cả các thành viên biên tập. Tuy nhiên các nhà mật mã không rảnh để thu thập 10 000 chữ ký cá nhân. Do đó mọi người muốn một chữ ký postquantum dùng chung mà không thể chia nhỏ ra các chữ ký cá nhân.

Yêu cầu là cần xây dựng mô hình chữ ký thỏa mãn các yêu cầu sau:

1. kích thước chữ ký không quá lớn, có thể hơn vài kilobytes;
2. kích thước public key (để kiểm tra chữ ký) là nhỏ. Kích thước của key nên là cố định (hoặc gần cố định) kể cả khi số lượng chuyên gia tăng lên, ví dụ 20 000;
3. việc kiểm tra chữ ký tốn không quá 2 phút;
4. chữ ký có thể chống lại các tấn công trên máy tính lượng tử.

A unique coding

Bài này khi nhìn đê thì "có vẻ" câu hỏi Q2 là trường hợp nhỏ hơn của Q1. Minh giải Q2 (không chắc đúng hoàn toàn) nên lời giải sau đây áp dụng cho cả Q1 và Q2.

Đề bài

Xét binary error-correcting code \mathcal{C} với độ dài n . Code này chỉ đơn giản là tập con của \mathbb{F}_2^n thôi và ta truyền một phần tử của code này qua các kênh truyền.

Khi đi qua các kênh truyền các bit có thể bị sai, dẫn tới bị đảo bit. Khi nhận được vector $\mathbf{y} \in \mathbb{F}_2^n$, ta sẽ decode thành một phần tử thuộc \mathcal{C} mà có khoảng cách gần \mathbf{y} nhất, nói cách khác là Hamming weight ngắn nhất.

Xét cơ chế decode maximal-likelihood. Giả sử ta nhận được $\mathbf{y} \in \mathbb{F}_2^n$, ta muốn xét các trường hợp lỗi xảy ra ít nhất, gọi là $d_{\mathbf{y}}$, nghĩa là

$$d_{\mathbf{y}} = \min_{\mathbf{x} \in \mathcal{C}} \text{wt}(\mathbf{x}, \mathbf{y}).$$

Tiếp theo, đặt

$$\mathcal{D}(\mathbf{y}) = \{\mathbf{x} \in \mathcal{C} : wt(\mathbf{x}, \mathbf{y}) = d_{\mathbf{y}}\}.$$

Cuối cùng ta decode \mathbf{y} thành phần tử \mathbf{x} nào đó trong $\mathcal{D}(\mathbf{y})$.

Chúng ta quan tâm tới các trường hợp code \mathcal{C} khiến $|\mathcal{D}(\mathbf{y})| = 1$ với mọi $\mathbf{y} \in \mathbb{F}_2^n$. Nói cách khác khi nhận được \mathbf{y} bất kì của \mathbb{F}_2^n ta có thể decode thành một dạng duy nhất.

Q1. Code \mathcal{C} như nào thì thỏa mãn tính chất này?

Q2. Code \mathcal{C} như nào thỏa mãn tính chất này và là không gian tuyến tính con của \mathbb{F}_2^n ?

Giải

Đầu tiên mình có nhận xét (khá rõ ràng) sau đây:

❶ Remark 11

Với mọi n , code $\mathcal{C} \equiv \mathbb{F}_2^n$ thỏa mãn tính chất trên.

Chúng ta có thể thấy rằng với mọi $\mathbf{y} \in \mathbb{F}_2^n$ nhận được thì sẽ decode thành chính nó trong \mathcal{C} .

Tiếp theo, ta xét nửa trên của \mathbb{F}_2^n . Trong hệ thập phân thì đó là các số từ 0 tới $2^{n-1} - 1$ và có dạng

$$\mathbf{x} = (0, x_1, x_2, \dots, x_{n-1}).$$

Nói cách khác, code \mathcal{C} là tập hợp

$$\mathcal{C} = \{\mathbf{x} = (0, x_1, x_2, \dots, x_{n-1}) : x_i \in \mathbb{F}_2\}.$$

Code \mathcal{C} này thỏa mãn tính chất trên và mình sẽ chứng minh ngay sau đây.

❷ Chứng minh

Giả sử ta nhận được $\mathbf{y} \in \mathbb{F}_2^n$. Ta có hai trường hợp:

- nếu $\mathbf{y} = (0, y_1, y_2, \dots, y_{n-1})$, hay nói cách khác biểu diễn thập phân của \mathbf{y} là từ 0 tới $2^{n-1} - 1$, thì \mathbf{y} được decode thành chính nó trong \mathcal{C} . Khi này $d_{\mathbf{y}} = 0$ nhỏ nhất và không có vector nào khác cho Hamming weight bằng 0 trừ chính nó.
- nếu $\mathbf{y} = (1, y_1, y_2, \dots, y_{n-1})$, hay nói cách khác biểu diễn thập phân của \mathbf{y} là từ 2^{n-1} tới $2^n - 1$, thì \mathbf{y} được decode thành $\mathbf{x} = (0, y_1, y_2, \dots, y_{n-1})$ trong \mathcal{C} . Khi này $d_{\mathbf{y}} = 1$ nhỏ nhất vì khac mỗi bit đầu tiên và cũng không có vector nào khác cho Hamming weight bằng 1.

Tiếp theo, mình viết các vector trong \mathcal{C} thành các hàng của 1 ma trận $2^{n-1} \times n$. Gọi A là ma trận hoán vị các cột của ma trận $2^{n-1} \times n$ đó. Khi đó A là ma trận có tính chất: trên mỗi hàng và trên mỗi cột có đúng một phần tử (bằng 1) và ma trận A khả nghịch. Ví dụ, với $n = 4$, ma trận để hoán vị cột 2 với cột 4 là

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Khi đó, nếu mình nhân ma trận $2^{n-1} \times n$ của code \mathcal{C} với bất kì ma trận A nào như vậy thì code \mathcal{C}' nhận được cũng thỏa mãn tính chất trên.

Example 24

Với $n = 4$ thì code \mathcal{C} gồm các vector

$$\mathcal{C} = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111\}.$$

Với ma trận A hoán vị cột 2 và 4 như trên ta có

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} = \mathcal{C}'.$$

Bây giờ mình sẽ chứng minh rằng với mọi ma trận A hoán vị các cột như vậy thì code \mathcal{C}' cũng thỏa mãn tính chất.

i Chứng minh

Đặt

$$\mathcal{C} = \{(0, x_1, x_2, \dots, x_{n-1}), x_i \in \mathbb{F}_2\}.$$

Gọi A là ma trận hoán vị cột kích thước $n \times n$. Khi đó ánh xạ

$$A : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n, \quad \mathbf{y} \rightarrow \mathbf{y} \cdot A$$

là song ánh do A là ma trận khả nghịch. Khi đó xét code

$$\mathcal{C}' = \{\mathbf{x} \cdot A : \mathbf{x} \in \mathcal{C}\}.$$

Mình vẫn có hai trường hợp.

Trường hợp 1. Với $\mathbf{y} = (0, y_1, y_2, \dots, y_{n-1}) \in \mathbb{F}_2^n$ từ 0 tới $2^{n-1} - 1$ như trên. Xét $\mathbf{y}' = \mathbf{y} \cdot A$.

Khi đó, với $\mathbf{x} = (0, y_1, y_2, \dots, y_{n-1}) \in \mathcal{C}$, ta có $\mathbf{x}' = \mathbf{x} \cdot A \in \mathcal{C}'$. Từ đây suy ra

$$wt(\mathbf{x}' \oplus \mathbf{y}') = wt((\mathbf{x} \cdot A) \oplus (\mathbf{y} \cdot A)) = wt((\mathbf{x} \oplus \mathbf{y}) \cdot A) = wt(\mathbf{0} \cdot A) = 0$$

Ở đây $\mathbf{0} = (0, 0, \dots, 0) \in \mathbb{F}_2^n$.

Nói cách khác $d_{\mathbf{y}'} = 0$ và có duy nhất một vector \mathbf{x}' được định nghĩa như trên thỏa mãn. Do đó $|\mathcal{D}(\mathbf{y}')| = 1$.

Trường hợp 2. Với $\mathbf{y} = (1, y_1, y_2, \dots, y_{n-1}) \in \mathbb{F}_2^n$ từ 2^{n-1} tới $2^n - 1$ như trên. Ta cũng xét $\mathbf{y}' = \mathbf{y} \cdot A$.

Khi đó, với $\mathbf{x} = (0, y_1, y_2, \dots, y_{n-1}) \in \mathcal{C}$, ta cũng có $\mathbf{x}' = \mathbf{x} \cdot A \in \mathcal{C}'$. Từ đây ta có

$$wt(\mathbf{x}' \oplus \mathbf{y}') = wt((\mathbf{x} \cdot A) \oplus (\mathbf{y} \cdot A)) = wt((\mathbf{x} \oplus \mathbf{y}) \cdot A) = wt((1, 0, 0, \dots, 0) \cdot A) = 1$$

Ở phép nhân vector $(1, 0, \dots, 0)$ với ma trận A , vì ma trận A chỉ có duy nhất một cột có dạng $(1, 0, \dots, 0)^T$ nên kết quả phép nhân là một vector có đúng một phần tử 1, còn lại là 0.

Nói cách khác $d_{\mathbf{y}'} = 1$ và có duy nhất một vector \mathbf{x}' được định nghĩa như trên thỏa mãn. Do đó $|\mathcal{D}(\mathbf{y}')| = 1$.

Như vậy ta đã chứng minh xong.

Hoàn toàn tương tự, khi code \mathcal{C} là các vector bắt đầu với hai số 0 thì ta lần lượt xét \mathbf{y} trong các khoảng $[0, 2^{n-2} - 1]$, $[2^{n-2}, 2^{n-1} - 1]$, $[2^{n-1}, 2^{n-1} + 2^{n-2} - 1]$, $[2^{n-1} + 2^{n-2} - 1, 2^n - 1]$. Nghĩa là

$$\mathcal{C} = \{\mathbf{x} = (0, 0, x_1, x_2, \dots, x_{n-2} : x_i \in \mathbb{F}_2^n)\}.$$

Khi đó ta xét các vector \mathbf{y} có dạng:

- $\mathbf{y} = (0, 0, y_1, y_2, \dots, y_{n-2})$;
- $\mathbf{y} = (0, 1, y_1, y_2, \dots, y_{n-2})$;
- $\mathbf{y} = (1, 0, y_1, y_2, \dots, y_{n-2})$;
- $\mathbf{y} = (1, 1, y_1, y_2, \dots, y_{n-2})$.

Theo quy nạp thì code C bắt đầu với i số 0 đều đúng, $0 \leq i \leq n$. Nghĩa là

$$\mathcal{C} = \{\mathbf{x} = (0, \dots, 0, x_1, x_2, \dots, x_{n-i} : x_i \in \mathbb{F}_2)\}.$$

Sau đó chúng ta lại áp dụng phép nhân với ma trận hoán vị cột A như bên trên thì các code \mathcal{C}' cũng thỏa mãn.

Vấn đề ở đây là, những code \mathcal{C} như vậy là không gian vector sinh bởi i vector ($0 \leq i \leq n$) trong các vector sau:

$$\begin{aligned}\mathbf{v}_1 &= (1, 0, 0, \dots, 0, 0) \\ \mathbf{v}_2 &= (0, 1, 0, \dots, 0, 0) \\ \mathbf{v}_3 &= (0, 0, 1, \dots, 0, 0) \\ &\dots = \dots \\ \mathbf{v}_n &= (0, 0, 0, \dots, 0, 1).\end{aligned}$$

Số cách chọn i vector từ n vector là

$$C_n^0 + C_n^1 + \dots + C_n^n = 2^n \text{ cách.}$$

Nói cách khác có 2^n code \mathcal{C} thỏa tính chất đề bài và là không gian tuyến tính của \mathbb{F}_2^n .

Example 25

Với $n = 3$ thì các code sau thỏa mãn tính chất

$$\begin{aligned}\mathcal{C}_1 &= \{000\}, \\ \mathcal{C}_2 &= \{000, 001\}, \\ \mathcal{C}_3 &= \{000, 010\}, \\ \mathcal{C}_4 &= \{000, 100\}, \\ \mathcal{C}_5 &= \{000, 001, 010, 011\}, \\ \mathcal{C}_6 &= \{000, 001, 100, 101\}, \\ \mathcal{C}_7 &= \{000, 010, 100, 110\}, \\ \mathcal{C}_8 &= \{000, 001, 010, 011, 100, 101, 110, 111\}.\end{aligned}$$

Bình luận

Đối với Q1 có thể thấy rằng bất cứ code nào chỉ chứa đúng một vector sẽ thỏa mãn điều kiện. Lý do là vì bất cứ \mathbf{y} nào được gửi tới cũng sẽ decode ra vector đó.

Bài này mình được 6/12 điểm vì đưa ra cách xây dựng tốt, trình bày đẹp.

Algebraic cryptanalysis

Bài này là bài 7 ở round 1 và là bài 8 ở round 2. Bài này mình giải khá qua loa ở round 1 và được giải đầy đủ, rõ ràng hơn bởi người đồng đội vip pro Chương ở round 2.

Đề bài

Bob muốn xây dựng stream cipher **BOB-0.1**.

Bob sử dụng một binary key độ dài 8 là $K = (k_1, \dots, k_8)$. Sau đó anh ấy sinh ra dãy nhị phân β theo quy tắc:

- $\beta_n = k_n$ khi $n = 1, 2, \dots, 8$;
- $\beta_n = \beta_{n-1} \oplus \beta_{n-8}$ khi $n \geq 9$.

Sau đó Bob sinh dãy nhị phân γ dùng trong phép XOR với plaintext. Dãy γ được tạo bởi quy tắc $\gamma_n = \beta_n \cdot \beta_{n+2} \oplus \beta_{n+7}$ với $n \geq 1$.

Alice chặn được 8 bit của γ sau khi để lỡ 1200 bit. Các bit đó là *00100001*. Liệu Alice có thể tìm lại được key K ban đầu không?

Giải

Độ dài K là 8 bit, nếu chúng ta bruteforce $K = (k_1, \dots, k_8)$ rồi sinh ra 1208 bit γ theo quy tắc trên và so sánh xem $\gamma_{1201}, \dots, \gamma_{1208}$ nào khớp với 8 bit trên thì ta có thể biết được K ban đầu là gì.

Và, bất ngờ chưa, có tới hai trường hợp K thỏa mãn :v :v

Bây giờ thì chúng ta cần xem xem tại sao lại có hai trường hợp thỏa mãn.

Cùng nhau khai triển $\beta_{n+1}, \dots, \beta_{n+8}$ theo $(\beta_{n-7}, \dots, \beta_n)$ nào.

$$\begin{aligned}\beta_{n+1} &= \beta_n \oplus \beta_{n-7} \\ \beta_{n+2} &= \beta_{n+1} \oplus \beta_{n-6} = \beta_n \oplus \beta_{n-7} \oplus \beta_{n-6} \\ \beta_{n+3} &= \beta_{n+2} \oplus \beta_{n-5} = \beta_n \oplus \beta_{n-7} \oplus \beta_{n-6} \oplus \beta_{n-5} \\ \beta_{n+4} &= \beta_{n+3} \oplus \beta_{n-4} = \beta_n \oplus \beta_{n-7} \oplus \beta_{n-6} \oplus \beta_{n-5} \oplus \beta_{n-4} \\ \beta_{n+5} &= \beta_{n+4} \oplus \beta_{n-3} = \beta_n \oplus \beta_{n-7} \oplus \beta_{n-6} \oplus \beta_{n-5} \oplus \beta_{n-4} \oplus \beta_{n-3} \\ \beta_{n+6} &= \beta_{n+5} \oplus \beta_{n-2} = \beta_n \oplus \beta_{n-7} \oplus \beta_{n-6} \oplus \beta_{n-5} \oplus \beta_{n-4} \oplus \beta_{n-3} \oplus \beta_{n-2} \\ \beta_{n+7} &= \beta_{n+6} \oplus \beta_{n-1} = \beta_n \oplus \beta_{n-7} \oplus \beta_{n-6} \oplus \beta_{n-5} \oplus \beta_{n-4} \oplus \beta_{n-3} \oplus \beta_{n-2} \oplus \beta_{n-1} \\ \beta_{n+8} &= \beta_{n+7} \oplus \beta_n = \beta_{n-7} \oplus \beta_{n-6} \oplus \beta_{n-5} \oplus \beta_{n-4} \oplus \beta_{n-3} \oplus \beta_{n-2} \oplus \beta_{n-1}\end{aligned}$$

Nếu viết ở dạng phép nhân ma trận modulo 2 ta có

$$\begin{pmatrix} \beta_{n+1} \\ \beta_{n+2} \\ \beta_{n+3} \\ \beta_{n+4} \\ \beta_{n+5} \\ \beta_{n+6} \\ \beta_{n+7} \\ \beta_{n+8} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \beta_{n-7} \\ \beta_{n-6} \\ \beta_{n-5} \\ \beta_{n-4} \\ \beta_{n-3} \\ \beta_{n-2} \\ \beta_{n-1} \\ \beta_n \end{pmatrix}$$

Ma trận to to kia là ma trận khả nghịch. Do đó, nếu chúng ta có các $\beta_{n+1}, \dots, \beta_{n+8}$ thì chúng ta có thể tìm ngược lại $\beta_{n-7}, \dots, \beta_n$. Tiếp tục quá trình này cuối cùng ta có thể tìm lại $(\beta_1, \dots, \beta_8) = K$.

Tiếp theo, cũng tương tự, chúng ta biểu diễn dãy γ theo β .

$$\begin{aligned}\gamma_{n+1} &= \beta_{n+1} \cdot \beta_{n+3} \oplus \beta_{n+8} \\ \gamma_{n+2} &= \beta_{n+2} \cdot \beta_{n+4} \oplus \beta_{n+1} \oplus \beta_{n+8} \\ \gamma_{n+3} &= \beta_{n+3} \cdot \beta_{n+5} \oplus \beta_{n+1} \oplus \beta_{n+2} \oplus \beta_{n+8} \\ \gamma_{n+4} &= \beta_{n+4} \cdot \beta_{n+6} \oplus \beta_{n+1} \oplus \beta_{n+2} \oplus \beta_{n+3} \oplus \beta_{n+8} \\ \gamma_{n+5} &= \beta_{n+5} \cdot \beta_{n+7} \oplus \beta_{n+1} \oplus \beta_{n+2} \oplus \beta_{n+3} \oplus \beta_{n+4} \oplus \beta_{n+8} \\ \gamma_{n+6} &= \beta_{n+6} \cdot \beta_{n+8} \oplus \beta_{n+1} \oplus \beta_{n+2} \oplus \beta_{n+3} \oplus \beta_{n+4} \oplus \beta_{n+5} \oplus \beta_{n+8} \\ \gamma_{n+7} &= \beta_{n+7} \cdot \beta_{n+1} \oplus \beta_{n+7} \cdot \beta_{n+8} \oplus \beta_{n+1} \oplus \beta_{n+2} \oplus \beta_{n+3} \oplus \beta_{n+4} \oplus \beta_{n+5} \oplus \beta_{n+6} \oplus \beta_{n+8} \\ \gamma_{n+8} &= \beta_{n+8} \cdot \beta_{n+1} \oplus \beta_{n+8} \cdot \beta_{n+2} \oplus \beta_{n+1} \oplus \beta_{n+2} \oplus \beta_{n+3} \oplus \beta_{n+4} \oplus \beta_{n+5} \oplus \beta_{n+6} \oplus \beta_{n+7}\end{aligned}$$

Trường hợp 1. $\beta_{n+1} = 0$. Khi đó từ γ_{1201} tới γ_{1208} tương đương với hệ phương trình

$$\begin{aligned}0 &= \beta_{n+8} \\ 0 &= \beta_{n+2} \cdot \beta_{n+4} \\ 1 &= \beta_{n+3} \cdot \beta_{n+5} \oplus \beta_{n+2} \\ 0 &= \beta_{n+4} \cdot \beta_{n+6} \oplus \beta_{n+2} \oplus \beta_{n+3} \\ 0 &= \beta_{n+5} \cdot \beta_{n+7} \oplus \beta_{n+2} \oplus \beta_{n+3} \cdot \beta_{n+4} \\ 0 &= \beta_{n+2} \oplus \beta_{n+3} \oplus \beta_{n+4} \oplus \beta_{n+5} \\ 0 &= \beta_{n+2} \oplus \beta_{n+3} \oplus \beta_{n+4} \oplus \beta_{n+5} \oplus \beta_{n+6} \\ 1 &= \beta_{n+2} \oplus \beta_{n+3} \oplus \beta_{n+4} \oplus \beta_{n+5} \oplus \beta_{n+6} \oplus \beta_{n+7}\end{aligned}$$

Hệ phương trình trên có nghiệm duy nhất $(\beta_i) = (0, 1, 1, 0, 0, 0, 1, 0)$.

Trường hợp 2. $\beta_{n+1} = 1$. Tương tự hệ phương trình cũng có nghiệm duy nhất $(\beta_i) = (1, 1, 1, 0, 0, 0, 0, 1)$.

Như vậy ta có hai nghiệm thỏa mãn chuỗi 8 bit $\gamma_{1201}, \dots, \gamma_{1208}$. Do không có điều kiện nào thêm, ta không thể xác định đâu là khóa trong hai trường hợp trên.

Bài này bạn Chứng được 4/4 điểm. Good job nigga. ^)

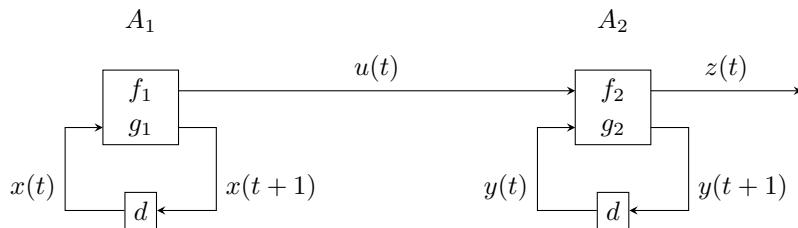
Finite-state machines

Bài này không biết làm!!!

Đề bài

Alice muốn tạo một generator để sinh một dãy số với độ dài chu kỳ lớn nhất có thể. Vì cô ấy biết về finite-state machine, generator G sẽ được xây dựng bởi hai machine A_1 và A_2 sao cho:

- $A_1 = (\mathbb{F}_2^n, \mathbb{F}_2, g_1, f_1)$ với hàm state-transition (hàm chuyển trạng thái) $g_1 : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ và hàm output $f_1 : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$, $n \geq 1$;
- $A_2 = (\mathbb{F}_2, \mathbb{F}_2^m, \mathbb{F}_2, g_2, f_2)$ với hàm state-transition $g_2 : \mathbb{F}_2 \times \mathbb{F}_2^m \rightarrow \mathbb{F}_2^m$ và hàm output $f_2 : \mathbb{F}_2 \times \mathbb{F}_2^m \rightarrow \mathbb{F}_2$, $m \geq 1$.



Với mỗi $t = 1, 2, \dots$, đặt

1. $x(t)$ và $y(t)$ là trạng thái của A_1 và A_2 , $x(1)$ và $y(1)$ là các giá trị khởi tạo;
2. $x(t+1) = g_1(t)$ là trạng thái tiếp theo của A_1 và $u(t) = f_1(x(t))$ là output bit của A_1 ;
3. $y(t+1) = g_2(u(t), y(t))$ là trạng thái tiếp theo của A_2 và $z(t) = f_2(u(t), y(t))$ là output bit của A_2 .

Dãy $z(1), z(2), z(3), \dots$ là output của generator G . Để thấy rằng dãy sinh bởi G có chu kỳ nhỏ nhất không vượt quá 2^{n+m} .

Theo thí nghiệm Alice thấy rằng, dãy output của G sẽ có chu kỳ nhỏ nhất nhỏ hơn 2^{n+m} nếu Hamming weight của f_1 là chẵn. Hãy chứng minh hoặc phủ định nhận xét của Alice.

Giải

Note theo gợi ý của thầy Kolomeec, chưa hiểu hết.

Để thấy rằng nếu $z(t)$ có chu kỳ tối đa thì các máy trạng thái A_1 và A_2 phải có chu kỳ tối đa lần lượt là 2^n và 2^m .

Các giá trị của g_2 có thể chia ra hai phần là $g_2(0, \mathbf{x})$ và $g_2(1, \mathbf{x})$ với $\mathbf{x} \in \mathbb{F}_2^m$. Trong đó 0 và 1 xác định từ f_1 .

Nếu g_2 có chu kỳ cực đại thì trọng số phải là 2^m , nói cách khác là hàm cân bằng, suy ra $g_2(0, \mathbf{x})$ và $g_2(1, \mathbf{x})$ đều có số chẵn phần tử.

Điều này có nghĩa là g_2 sẽ sinh ra dãy có chu kỳ tạo thành hoán vị chẵn.

Tuy nhiên nếu g_2 có chu kỳ cực đại thì chu kỳ đó phải tạo thành hoán vị lẻ vì khi đó hoán vị

$$1 \rightarrow 2 \rightarrow \dots \rightarrow t \rightarrow 1 = (1, 2)(1, 3) \cdots (1, t)$$

có $t - 1$ tranposition. Nói cách khác đây là hoán vị lẻ vì $t = 2^m$ chẵn.

Như vậy để chu trình đạt chu kỳ tối đa thì nó phải là hoán vị lẻ, mâu thuẫn với phân tích ở trên g_2 sẽ sinh ra dãy là hoán vị chẵn.

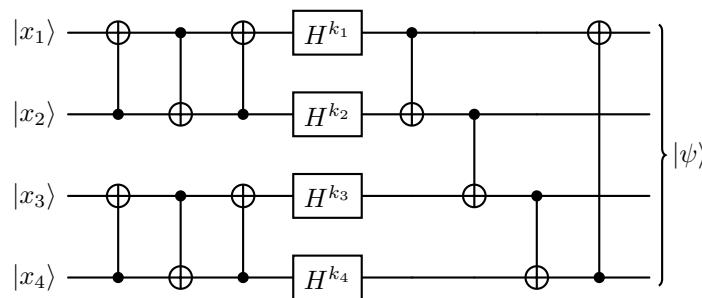
Trong cách giải này có hai chỗ mìn không biết thầy lấy đâu ra: tại sao g_2 cân bằng thì cả $g_2(0, \mathbf{x})$ và $g_2(1, \mathbf{x})$ đều phải có số chẵn phần tử mà không phải là đều có số lẻ? Liên hệ giữa trọng số hàm bool và hoán vị chẵn/lẻ là gì?

Quantum encryption

Đây là bài 8 của round 1 và bài 10 của round 2. Bài này sai gần bước cuối mới cay.

Đề bài

Bob tạo một thuật toán mã hóa encrypt 4 bit (x_1, x_2, x_3, x_4) bằng key cũng 4 bit (k_1, k_2, k_3, k_4) với mạch sau:



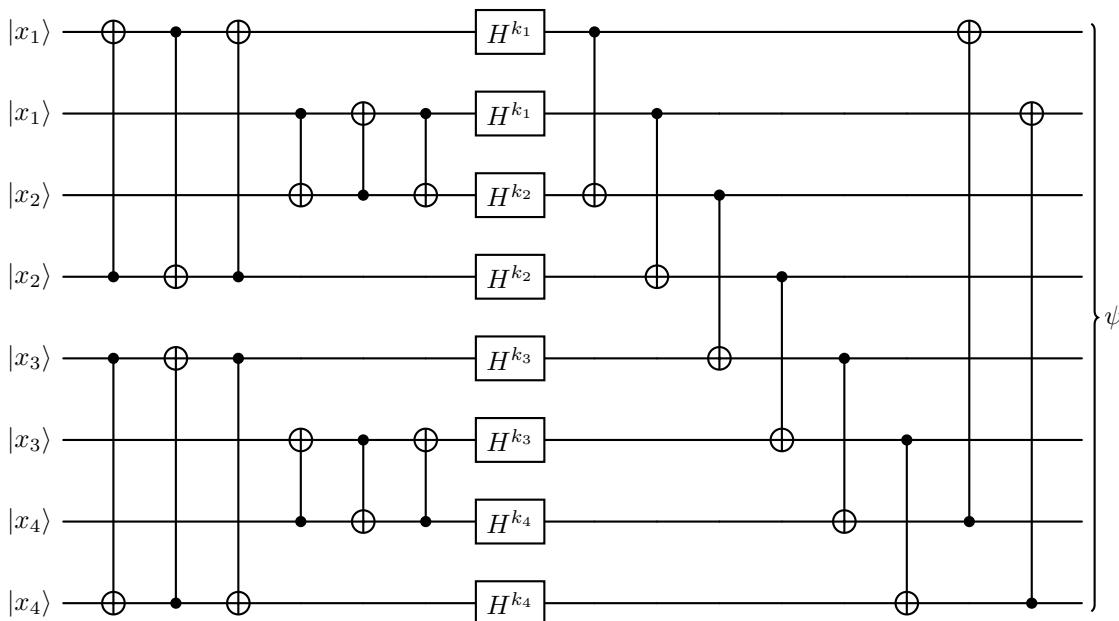
Plaintext 4 bit (x_1, x_2, x_3, x_4) được biểu diễn ở dạng 4-qubit "plainstate" $|x_1, x_2, x_3, x_4\rangle$. Quantum state này là input cho mạch ở dạng qubit đơn đi qua các cổng.

Ở đây hai loại cổng được sử dụng là CNOT và Hadamard.

Kí hiệu H^b với $b \in \{0, 1\}$ có nghĩa là, nếu $b = 0$ thì cổng đồng nhất I được sử dụng (không thay đổi), còn nếu $b = 1$ thì cổng Hadamard sẽ được sử dụng.

Kết quả sau khi qua mạch là "cipherstate" $|\psi\rangle$.

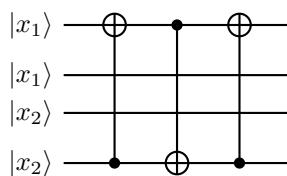
Bob có nhiệm vụ tăng số qubit đầu vào lên nhằm giảm các sai số khi tính toán và truyền dữ liệu trên kênh quantum. Do đó Bob biến đổi thành mạch sau:



Alice nói rằng cô ấy có thể tìm lại được key nếu biết N amplitude của kết quả $|\psi\rangle$. Do có 8 qubits ở kết quả nên số lượng amplitude tối đa là $2^8 = 256$, nói cách khác $N \leq 256$. Vậy Alice cần ít nhất bao nhiêu amplitude là đủ để tìm lại key?

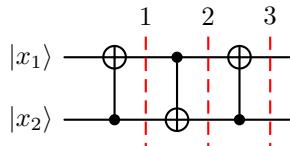
Giải

Đầu tiên xét 4 dây trên, 4 dây dưới tương tự.

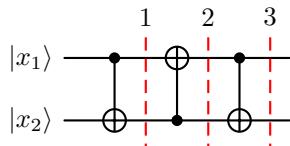


Chúng ta xét dây 1 và 4 của mạch (tương tự cho dây 2 và 3). Áp dụng cổng CNOT liên tiếp 3 lần ta có

$$|x_1\rangle \otimes |x_2\rangle \rightarrow |x_1 \oplus x_2\rangle \otimes |x_2\rangle \rightarrow |x_1 \oplus x_2\rangle \otimes |x_1\rangle \rightarrow |x_2\rangle \otimes |x_1\rangle$$



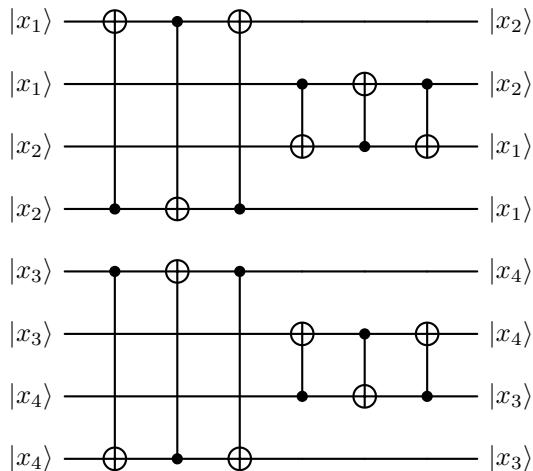
Hình 4.2: Dây 1 và dây 4



Hình 4.3: Dây 2 và dây 3

Nói cách khác là đảo bit. :v

Tương tự cho các cặp dây (5, 8) và (6, 7). Do đó khi tới trước các cổng Hadamard thì thứ tự các qubit từ trên xuống dưới là hình sau.

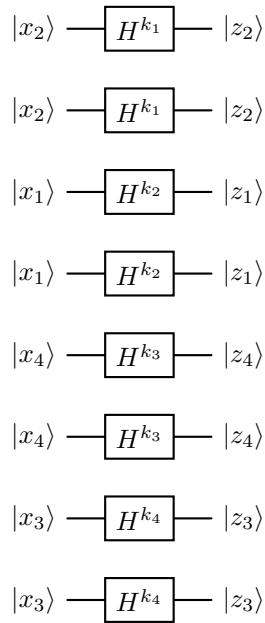


Hình 4.4: Qubits trước Hadamard

Mạch ở dây 1 và 2 đều có dạng $|x_2\rangle$ đi qua H^{k_1} nên sau khi qua cổng mìn đặt $|z_2\rangle = H^{k_1}|x_2\rangle$.

Tương tự, $|z_1\rangle = H^{k_2}|x_1\rangle$ cho dây 3 và 4, $|z_4\rangle = H^{k_3}|x_4\rangle$ cho dây 5 và 6, $|z_3\rangle = H^{k_4}|x_3\rangle$ cho dây 7 và 8.

Mạch sau khi đi qua Hadamard có dạng



Hình 4.5: Qubits sau Hadamard

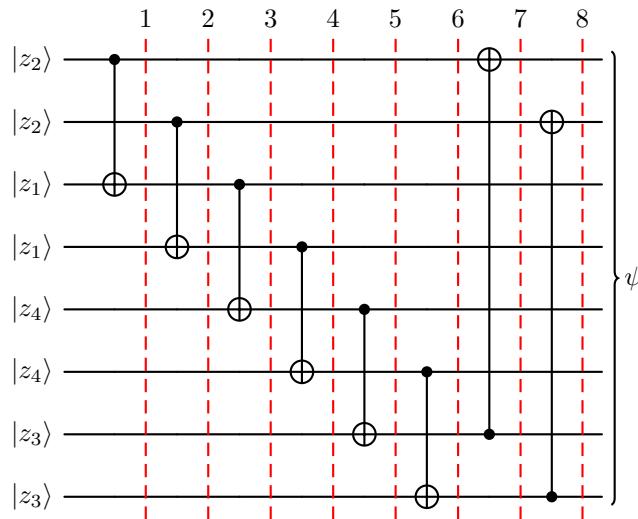
Ở đây chúng ta có một lưu ý nhỏ có thể giúp ích trong việc giới hạn số lượng amplitude theo đề bài. Nếu $k_1 = 0$ thì $|z_2\rangle = |x_2\rangle$. Nếu $k_1 = 1$ thì $|z_2\rangle = \frac{|0\rangle + (-1)^{x_2}|1\rangle}{\sqrt{2}}$. Như vậy, hệ số trước $|0\rangle$ của $|z_2\rangle$ có thể là $0, 1, \frac{1}{\sqrt{2}}$ đều không âm.

Bây giờ chúng ta quay lại toán tử CNOT. Ma trận tương ứng của toán tử CNOT là $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$. Kết quả sau khi thực hiện toán tử CNOT là hệ số trước $|00\rangle$ và $|01\rangle$ giữ nguyên, còn hệ số trước $|10\rangle$ và $|11\rangle$ đổi chỗ cho nhau.

Đối với 3 qubit, mình **dự đoán** tương tự.

Ở cổng CNOT đầu tiên, dây 1 control dây 3. Nếu mình chỉ xét 3 dây đầu thì tích các qubit gồm $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$, $|111\rangle$.

Áp dụng "chiến thuật" tương tự, mình chỉ quan tâm vị trí 1 và 3. Nghĩa là hệ số của $|0x0\rangle$ và $|0x1\rangle$ giữ nguyên, còn hệ số trước $|1x0\rangle$ và $|1x1\rangle$ đổi chỗ cho nhau, với $x \in \{0, 1\}$. Nói cách khác, 8 hệ số trước amplitude chỉ thay đổi vị trí chứ không nhiều hơn hay ít đi, hay tập hợp hệ số giữ nguyên.



Như vậy, giả sử $|z_2\rangle = a|0\rangle + b|1\rangle$, $|z_1\rangle = c|0\rangle + |1\rangle$, $|z_4\rangle = e|0\rangle + f|1\rangle$, $|z_3\rangle = g|0\rangle + h|1\rangle$. Khi đó kết quả cipherstate là

$$|\psi\rangle = |z_2\rangle \otimes |z_2\rangle \otimes |z_1\rangle \otimes |z_1\rangle \otimes |z_4\rangle \otimes |z_4\rangle \otimes |z_3\rangle \otimes |z_3\rangle$$

Xét

$$|z_2\rangle \otimes |z_2\rangle = a^2|00\rangle + ab|01\rangle + ab|10\rangle + b^2|11\rangle.$$

Ở đây có 3 hệ số khác nhau là (a^2, ab, b^2) . Với lưu ý bên trên $a \geq 0$ nên từ a^2 tính được a . Từ a , ta cần thêm ab để xác định b .

Như vậy mình cần $2^4 = 16$ hệ số để tìm lại các key ban đầu.

Bình luận

Thế éo nào mà mình lại nhầm khúc cuối mà lấy cả a^2 , ab và b^2 nên kết quả ra $3^4 = 81$. Tất nhiên là **SAI BÉT** nên chỉ được $2/8$. :(((

AntCipher

Bài này là bài số 2 ở round 1 và là bài số 11 ở round 2. Lúc thi round 1 mình không biết giải, còn ở round 2 thì mình đã giải theo cách như sau.

Đề bài

Đặt

$$\begin{aligned} f = & (x_1 \vee x_2 \vee x_9) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_9) \wedge (\neg x_1 \vee x_2 \neg x_9) \wedge (x_1 \vee \neg x_2 \vee x_9) \wedge \\ & (x_1 \vee x_2 \vee x_3) \wedge (\neg x_9 \vee \neg x_{10} \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_9 \vee x_{10} \vee \neg x_4) \wedge \\ & (\neg x_1 \vee x_2 \vee x_5) \wedge (x_9 \vee \neg x_{10} \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_6) \wedge (x_9 \vee x_{10} \vee \neg x_6) \wedge \\ & (x_1 \vee x_2 \vee x_3 \vee x_4 \vee \neg x_7) \wedge (x_2 \vee x_3 \vee x_4 \vee \neg x_7 \vee \neg x_8). \end{aligned}$$

Hàm f gồm 10 biến được viết dưới dạng CNF (conjunctive normal form). Thuật toán mã hóa dựa trên hàm f biến đổi hai bit plaintext (x_1, x_2) thành hai bit ciphertext (x_9, x_{10}) khi giá trị hàm $f = \text{True}$. Hàm f này có 10 biến x_1, x_2, \dots, x_{10} và 46 literals, là các hạng tử trong biểu diễn CNF của hàm. Ví dụ với dấu ngoặc thứ hai có 3 literals là $\neg x_1$, $\neg x_2$ và $\neg x_9$.

Q. Vì các giới hạn tính toán nên chúng ta chỉ có thể sử dụng tối đa 16 biến với 20 literals. Nhắc lại rằng hàm f ở trên có 10 biến và 46 literals. Hãy tìm cách biểu diễn tương đương của thuật toán mã hóa trên với giới hạn đã cho.

Giải

Khi mình code hàm để tính giá trị hàm f và xem xét những vector

$$\mathbf{x} = (x_1, \dots, x_{10})$$

mà $f = True$, mình nhận thấy rằng:

- nếu $(x_1, x_2) = (0, 0)$ thì $(x_9, x_{10}) = (1, 0)$
- nếu $(x_1, x_2) = (0, 1)$ thì $(x_9, x_{10}) = (1, 1)$
- nếu $(x_1, x_2) = (1, 0)$ thì $(x_9, x_{10}) = (0, 0)$
- nếu $(x_1, x_2) = (1, 1)$ thì $(x_9, x_{10}) = (0, 1)$

Mình nhận ra rằng các biến $x_3, x_4, \dots, x_7, x_8$ hoàn toàn không tác động lên việc mã hóa từ (x_1, x_2) thành (x_9, x_{10}) (ít nhất là ở những chỗ $f = True$:v).

Như vậy bài toán được rút gọn thành hàm boolean 4 biến x_1, x_2, x_9 và x_{10} . Ở đó

$$f(0010) = f(0111) = f(1000) = f(1101) = 1.$$

Các vector còn lại thì $f = 0$. Ở dưới là bảng chân trị.

x_1	x_2	x_9	x_{10}	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Từ bảng chân trị trên, sử dụng phương pháp bìa Karnaugh mình rút gọn được thành

$$\begin{aligned} f(x_1, x_2, x_9, x_{10}) = & (\neg x_1 \vee \neg x_9) \wedge (x_1 \vee x_9) \wedge \\ & (\neg x_1 \vee \neg x_2 \vee x_{10}) \wedge (x_1 \vee x_2 \vee \neg x_{10}) \wedge \\ & (\neg x_1 \vee x_2 \vee \neg x_{10}) \wedge (x_1 \vee \neg x_2 \vee x_{10}). \end{aligned}$$

CNF này có 4 biến và 16 literals, thỏa mãn yêu cầu đề bài và ăn trọn 6/6 điểm.

Kết luận

Năm 2023 khá chill :))) học được nhiều điều :))) không hối hận vì đã liều :)))

Xin chân thành cảm ơn bạn Chương và bạn Uyên đã đồng hành cùng mình trong kì NSUCRYPTO 2023.

4.5.2 Siberian Mathematical Contest 2023

Đề và lời giải gốc (tiếng Nga) các bạn có thể tải ở [solutions_SMC_2023.pdf](#).

Đề dành cho năm nhất**Bài 1****Đề bài**

Tồn tại hay không một đa thức $p(x)$ sao cho mọi hệ số khác không của nó không phải số nguyên, nhưng với hai số nguyên bất kì khác nhau a và b ta đều có $\frac{p(a) - p(b)}{b - a}$ là số nguyên?

Lời giải

Tồn tại, ví dụ như $p(x) = \frac{x^4 + x^2}{2}$:

$$\frac{p(a) - p(b)}{a - b} = \dots = \frac{(a+b)(a^2 + b^2 + 1)}{2}.$$

Vì $a + b$ và $a^2 + b^2 + 1$ khác tính chẵn lẻ nên tích trên luôn chia hết cho 2 và do đó là số nguyên.

Bài 2**Đề bài**

Cho dãy $\{a_n\}$, $n \geq 0$ thỏa điều kiện $a_0 = 1$, $a_1 = \frac{1}{2}$,

$$a_n = \frac{n}{2}a_{n-1} + \frac{n(n-1)}{2}a_{n-2} + \frac{(-1)^n(1-n)}{2(n+1)}.$$

Tìm $\lim_{n \rightarrow \infty} \frac{a_n}{n!}$.

Lời giải

Từ quy nạp toán học có thể tìm được

$$a_n = na_{n-1} + \frac{(-1)^n}{n+1}.$$

Từ đó,

$$\begin{aligned} a_n &= n \left((n-1)a_{n-2} + \frac{(-1)^{n-1}}{n} \right) + \frac{(-1)^n}{n+1} \\ &= n(n-1)a_{n-2} + (-1)^{n-1} + \frac{(-1)^n}{n+1} \\ &= \dots \\ &= n!a_0 - \frac{n!}{2!} + \frac{(-1)^2 n!}{3!} + \frac{(-1)^3 n!}{4!} + \dots + \frac{(-1)^{n-1} n!}{n!} + \frac{(-1)^n n!}{n!} \end{aligned}$$

Nói cách khác,

$$\begin{aligned}\frac{a_n}{n!} &= a_0 + \frac{(-1)^1}{2!} + \frac{(-1)^2}{3!} + \frac{(-1)^3}{4!} + \cdots + \frac{(-1)^{n-1}}{n!} + \frac{(-1)^n}{n!} \\ &= \sum_{k=0}^n \frac{(-1)^k}{(k+1)!} = 1 - \sum_{k=0}^{n+1} \frac{(-1)^k}{k!} \rightarrow 1 - \frac{1}{e}\end{aligned}$$

Bài 3

Đề bài

Cho hàm đơn điệu liên tục tăng nghiêm ngặt $f(x)$ xác định trên nửa khoảng $[0, +\infty)$ và thỏa $f(0) = 0$. Đặt $g(x)$ là hàm ngược của $f(x)$. Chứng minh rằng, với mọi số nguyên m, n thì bất đẳng thức sau thỏa mãn:

$$mn \leq \sum_{k=0}^m [f(k)] + \sum_{s=0}^n [g(s)].$$

Lời giải

Giả sử, $[f(m)] \geq n$. Đặt $[f(s)] = l_s$. Theo tính đơn điệu thì $l_s \leq f(s) < l_{s+1}$. Do $f(x)$ liên tục và đơn điệu tăng nghiêm ngặt nên $g(x)$ cũng liên tục và đơn điệu.

$$\begin{aligned}\sum_{k=0}^m [f(k)] &= l_1 + l_2 + \cdots + l_m; \\ \sum_{s=0}^n [g(s)] &= 1(l_2 - l_1) + 2(l_3 - l_2) + 3(l_4 - l_3) + \cdots + p(n - l_p).\end{aligned}$$

với p nào đó mà $l_p < n \leq l_{p+1}$. Từ đây ta có

$$l_1 + l_2 + \cdots + l_p + 1(l_2 - l_1) + 2(l_3 - l_2) + \cdots + (p-1)(l_p - l_{p-1}) = pl_p.$$

Vì $l_{p+1}, l_{p+2}, \dots, l_m > n$ nên

$$\begin{aligned}\sum_{k=0}^m [f(k)] + \sum_{s=0}^n [g(s)] &= pl_p + p(n - l_p) + l_{p+1} + l_{p+2} + \cdots + l_m \\ &= pn + l_{p+1} + l_{p+2} + \cdots + l_m > pn + (m-p)n = mn.\end{aligned}$$

Chứng minh tương tự cho $[f(m)] < n$ (thay $f(x)$ thành $g(x)$).

Bài 4

Đề bài

Cho đa thức với hệ số nguyên

$$f(x) = x^{2m} + x^{m+2} - 4x^m + x^{m-n} + 1.$$

với $m > n$ là hai số tự nhiên nguyên tố cùng nhau. Tìm ước chung lớn nhất giữa $f(x)$ và $f'(x)$.

Lời giải

Ta thấy rằng $x = 1$ là nghiệm bậc hai của $f(x)$. Ta sẽ chứng minh rằng $f(x)$ và $f'(x)$ không có nghiệm tổng quát (nghiệm phức) nào khác. Giả sử có nghiệm $z \in \mathbb{C}$. Khi đó

$$z^m + z^{-m} + z^n + z^{-n} = 4, m(z^m - z^{-m}) + n(z^n - z^{-n}) = 0.$$

Đặt $u = z^m$ và $v = z^n$. Khi đó

$$u + u^{-1} + v + v^{-1} = 4, m(u - u^{-1}) + n(v - v^{-1}) = 0.$$

Nếu u và v bằng 1 thì $z = 1$ vì $(m, n) = 1$. Nếu $u \neq 1$ và $v \neq 1$ thì ta nghiệm của hệ phương trình là $(u, v) \in \{(u_0, v_0), (u_0^{-1}, v_0^{-1})\}$ với

$$u_0 = -\frac{m^2 + 3n^2 + 2n\sqrt{2(m^2 + n^2)}}{m^2 - n^2}, v_0 = \frac{3m^2 + n^2 + 2m\sqrt{2(m^2 + n^2)}}{m^2 - n^2}.$$

Ở đây u_0 và v_0 là các số thực và thỏa mãn đẳng thức $u_0^n = v_0^m$. Nhưng vì $1 < |u_0| < v_0$ nên khi $m > n$ các đẳng thức trên không đúng.

4.5.3 Amateurs CTF 2023

Minimalaestic

Đề bài ở `aes.py`.

Phân tích hiện trường

Đây là một bài AES nhưng trên ma trận 2×2 . Ma trận đầu vào có dạng

$$\begin{pmatrix} s_{00} & s_{01} \\ s_{10} & s_{11} \end{pmatrix}.$$

Các động tác cơ bản giống với AES gốc, bao gồm: add round key, shift rows, mix columns và sub bytes. Đối với vòng cuối sử dụng một biến thể của sub byte, shift rows và add round key. Ở bài này có 100 vòng biến đổi bình thường và 1 vòng cuối sử dụng các biến thể trên.

Sub Bytes và Final Sub Bytes

SBox được sử dụng trong bài là SBox của AES gốc. Do đó mình cũng không nghĩ rằng sẽ khai thác được gì ở đây. Ở đây có một điều mình cần nhớ là Sub Bytes biến đổi trên cột đầu, và Final Sub Bytes biến đổi trên cột sau.

Đối với Sub Bytes thì

$$\begin{pmatrix} s_{00} & s_{01} \\ s_{10} & s_{11} \end{pmatrix} \rightarrow \begin{pmatrix} s_{00} & S(s_{01}) \\ s_{10} & S(s_{11}) \end{pmatrix}.$$

Đối với Final Sub Bytes thì

$$\begin{pmatrix} s_{00} & s_{01} \\ s_{10} & s_{11} \end{pmatrix} \rightarrow \begin{pmatrix} S(s_{00}) & s_{01} \\ S(s_{10}) & s_{11} \end{pmatrix}.$$

Shift Rows và Final Shift Rows

Đối với Shift Rows thay đổi vị trí 2 bytes ở cột (?) đầu.

$$\begin{pmatrix} s_{00} & s_{01} \\ s_{10} & s_{11} \end{pmatrix} \rightarrow \begin{pmatrix} s_{10} & s_{01} \\ s_{00} & s_{11} \end{pmatrix}.$$

Đối với Final Shift Rows thay đổi vị trí 2 bytes ở cột (?) sau.

$$\begin{pmatrix} s_{00} & s_{01} \\ s_{10} & s_{11} \end{pmatrix} \rightarrow \begin{pmatrix} s_{00} & s_{11} \\ s_{10} & s_{01} \end{pmatrix}.$$

Mình thấy rằng phép biến đổi ngược đối với shift rows và final shift rows cũng là chính nó.

Mix Columns

Đối với Mix Columns, cột đầu mới sẽ là cột đầu cũ XOR với cột sau.

$$\begin{pmatrix} s_{00} & s_{01} \\ s_{10} & s_{11} \end{pmatrix} \rightarrow \begin{pmatrix} s_{00} \oplus s_{01} & s_{01} \\ s_{10} \oplus s_{11} & s_{11} \end{pmatrix}.$$

Tương tự shift rows, phép biến đổi ngược của mix columns cũng là chính nó.

Add Round Keys

Phép Add Round Keys ở bài này khá thú vị (mặc dù mình cũng không khai thác từ đó).

```
def add_round_key(s, k):
    for i in range(2):
        for j in range(2):
            s[i][j] ^= k[k[k[k[i*2+j]]%4]%4]
```

Đối với final add round keys thì chỉ thực hiện phép XOR trên cột sau.

Okay, tới đây thì việc phân tích mã hóa của bài này đã tạm xong và mình cũng không thấy điểm nào có thể dùng để khai thác (hoặc chưa thấy). Điều làm mình quan tâm là hàm `schedule_key`.

Key Schedule

```
def schedule_key(k):
    for i in range(4):
        for j in range(2*ROUNDS):
            k[i] = pow(pow(sbox[sbox[sbox[((k[i] << 4) ^ k[i]) << 4) ^ k[i]] % 256]], 
            pow(k[i], k[i]), 256),
            pow(sbox[k[i]], sbox[k[i]]), 256)

def final_schedule(k):
    for i in range(4):
        k[i] = sbox[k[i]]
```

Hàm sinh khóa con khá lạ. Do đó mình thử in ra khóa con ở các vòng với một chút điều chỉnh ở hàm `encrypt` với các khóa được random.

```
def encrypt(p, k):
    ciphertext = b""
    for i in split_blocks(p):
        key = k.copy()
        i = bytes2matrix(i)
        add_round_key(i, key)
        for j in range(ROUNDS):
            schedule_key(key)
```

(continues on next page)

(continued from previous page)

```

        print(f"[j], {key}")
        sub_bytes(i)
        shift_rows(i)
        mix_columns(i)
        add_round_key(i, key)
    final_schedule(key)
    print(f"100, {key}")
    final_sub(i)
    final_shift(i)
    final_add(i, key)
    ciphertext += matrix2bytes(i)
return ciphertext

pt = b"haha"
for _ in range(1000):
    key = [random.choice(range(256)) for _ in range(4)]
    encrypt(pt, key)

```

Các khóa con trong các vòng từ 0 tới 99 đều có vẻ như nằm trong một tập hợp nhất định là $\{0, 1, 175\}$. Từ đó khóa con ở vòng 100 (vòng final) cũng nằm trong một tập hợp nhất định do đã đi qua hàm sbox (`final_schedule`) và kết quả là tập $\{99, 121, 124\}$.

Truy tìm đầu mối

Nơi cryptanalysis bắt đầu

Chiến thuật của mình là tìm khóa trước khi đi vào vòng lặp với known-plaintext là format của flag. Chú ý rằng ở đây không cần phải tìm khóa ban đầu mà chỉ cần tìm khóa trước khi vào vòng lặp, tức là khóa tham gia vào phép XOR `add_round_key` đầu tiên.

```

def encrypt(p, k):
    ciphertext = b""
    for i in split_blocks(p):
        key = k.copy()
        i = bytes2matrix(i)
        add_round_key(i, key)
        # Find intermediate key used in add_round_key #
        for j in range(ROUNDS):
            schedule_key(key)
            sub_bytes(i)
            shift_rows(i)
            mix_columns(i)
            add_round_key(i, key)
        final_schedule(key)
        final_sub(i)
        final_shift(i)
        final_add(i, key)
        ciphertext += matrix2bytes(i)
    return ciphertext

```

Để tìm khóa ở điểm được đánh dấu, mình sẽ đi ngược từ ciphertext lên. Mình bruteforce các khóa con dùng trong vòng lặp ($3^4 = 81$ trường hợp). Ứng với mỗi khóa con cho vòng lặp mình có một khóa con cho vòng cuối cùng (final). Kết hợp hai khóa đó và ciphertext mình sẽ tìm được state trước khi vào vòng lặp. Cuối

cùng mình XOR kết quả đó cho known-plaintext thì sẽ được khóa ở điểm được đánh dấu.

Từ nhận xét bên trên, 100 vòng (0 tới 99) sử dụng cùng một khóa con, mình đặt là k_1 . Ở vòng final sử dụng một khóa con, mình đặt là k_2 . Quan hệ giữa chúng là $k_2 = \text{final_schedule}(k_1)$.

Man-In-The-Middle

Với mỗi block 4 bytes ciphertext và known-plaintext tương ứng, mình đi ngược từ dưới lên để tìm key trung gian. Lưu ý rằng mình chỉ cần xây dựng bảng `inv_sbox` vì các phép biến đổi khác có phép biến đổi ngược là chính nó.

```
final_add(ciphertext, k2_)
final_shift(ciphertext)
inv_final_sub(ciphertext)
for j in range(ROUNDS):
    add_round_key(ciphertext, k1_)
    mix_columns(ciphertext)
    shift_rows(ciphertext)
    inv_sub_bytes(ciphertext)

pt = matrix2bytes(plaintext)
ct = matrix2bytes(ciphertext)
key = xor(pt, ct)
```

Với format flag là `amateursCTF{` có 12 bytes tương ứng 3 block, mình thực hiện biến đổi trên với 12 bytes ciphertext tương ứng. Với mỗi block mình sẽ tìm ra được tập hợp các key tương ứng với các k_1 (và k_2 tương ứng với k_1). Sau đó mình giao các tập hợp key lại sẽ được key ban đầu.

Nói cách khác

```
candidates1 = set()
candidates2 = set()
candidates3 = set()

for k1 in product(K1, repeat=4):
    k2 = final_schedule_(list(k1))

    k1_ = list(k1).copy()
    k2_ = list(k2).copy()

    # Phase 1
    plaintext = bytes2matrix(flag[:4])
    ciphertext = bytes2matrix(ctx[:4])
    final_add(ciphertext, k2_)
    final_shift(ciphertext)
    inv_final_sub(ciphertext)
    for j in range(ROUNDS):
        add_round_key(ciphertext, k1_)
        mix_columns(ciphertext)
        shift_rows(ciphertext)
        inv_sub_bytes(ciphertext)

    pt = matrix2bytes(plaintext)
    ct = matrix2bytes(ciphertext)
```

(continues on next page)

(continued from previous page)

```

key = xor(pt, ct)
candidates1.add(bytes(key))

# Phase 2
plaintext = bytes2matrix(flag[4:8])
ciphertext = bytes2matrix(ctx[4:8])
final_add(ciphertext, k2)
final_shift(ciphertext)
inv_final_sub(ciphertext)
for j in range(ROUNDS):
    add_round_key(ciphertext, k1_)
    mix_columns(ciphertext)
    shift_rows(ciphertext)
    inv_sub_bytes(ciphertext)

pt = matrix2bytes(plaintext)
ct = matrix2bytes(ciphertext)
key = xor(pt, ct)
candidates2.add(bytes(key))

# Phase 3
plaintext = bytes2matrix(flag[8:12])
ciphertext = bytes2matrix(ctx[8:12])
final_add(ciphertext, k2_)
final_shift(ciphertext)
inv_final_sub(ciphertext)
for j in range(ROUNDS):
    add_round_key(ciphertext, k1_)
    inv_mix_columns(ciphertext)
    inv_shift_rows(ciphertext)
    inv_sub_bytes(ciphertext)

pt = matrix2bytes(plaintext)
ct = matrix2bytes(ciphertext)
key = xor(pt, ct)
candidates3.add(bytes(key))

key = list(candidates1.intersection(candidates2).intersection(candidates3))
print(key)

```

Kết quả khi giao ba tập hợp chỉ có đúng một key b'\x00**\x00'. Quá tốt!!!

Ờ mà khoan, từ từ đã :))) Có gì đó không đúng lắm. Cụ thể là khi mình encrypt hai block plaintext đầu thì nó không ra đúng ciphertext. Cụ thể hơn nữa, encrypt ra đúng ở vị trí 0 và 2, sai ở vị trí 1 và 3 cho mỗi block (?!?!).

Sửa chữa lỗi lầm

Mình mất cả ngày để tìm lỗi sai nhưng thất bại. Và mình đã chuyển sang *tán công cưỡng ép*. Vì encrypt đúng ở vị trí 0 và 2 còn sai ở vị trí 1 và 3 nên mình kết luận được là k_2 có vấn đề, tức là key ở bước `final_add`.

Mình thử in ra k_1 và k_2 tương ứng với khóa tìm được bên trên b'\x00**\x00'.

```

for k1 in product(K1, repeat=4):
    k2 = final_schedule_(list(k1))

    k1_ = list(k1).copy()
    k2_ = list(k2).copy()

    # Phase 1
    plaintext = bytes2matrix(flag[:4])
    ciphertext = bytes2matrix(ctx[:4])
    final_add(ciphertext, k2_)
    final_shift(ciphertext)
    inv_final_sub(ciphertext)
    for j in range(ROUNDS):
        add_round_key(ciphertext, k1_)
        mix_columns(ciphertext)
        shift_rows(ciphertext)
        inv_sub_bytes(ciphertext)

    pt = matrix2bytes(plaintext)
    ct = matrix2bytes(ciphertext)
    key = xor(pt, ct)
    candidates1.add(bytes(key))
    if bytes(key) == b'\x00**\x00':
        print(k1, k2)

```

Hai trường hợp có thể xảy ra cho cặp (k_1, k_2) là $([1, 0, 0, 1], [124, 99, 99, 124])$ và $([1, 0, 175, 1], [124, 99, 121, 124])$. Mình chỉ cần xét trường hợp 1 là được.

Khi nhìn vào hàm `final_add` thì mình biết chắc rằng $k[k[k[k[i*2+1]]]]$ chắc chắn nằm trong $[124, 99]$ nên mình bóc đại $k_2 = [124, 124, 124, 124]$, và nó đã thành công (????).

Hàm `decrypt` mình fix cứng k_1 và k_2 luôn, và decrypt ra toàn bộ flag ban đầu.

```

def decrypt(c, k):
    plaintext = b""

    for i in split_blocks(c):
        key = k.copy()
        k1, k2 = [1, 0, 0, 1], [124, 124, 124, 124]

        i = bytes2matrix(i)

        final_add(i, k2)
        final_shift(i)
        inv_final_sub(i)
        for j in range(ROUNDS):
            add_round_key(i, k1)
            mix_columns(i)
            shift_rows(i)
            inv_sub_bytes(i)

        plaintext += bytes(xor(matrix2bytes(i), key))
    return plaintext

```

(continues on next page)

(continued from previous page)

```
# b' amateursCTF{th1s_1s_wh4t_b4d_k3y_sch3dul1ng_d03s_t0_a_p3rson_109bcd1f}\x02\x02.
```

Nếu bạn nhìn thấy được điểm sai nào đó trong bài làm của mình dẫn tới *tấn công cưỡng ép* thì có thể nói cho mình biết. Thực sự thì mình không biết mình đang lag chỗ nào :confused:

Owo Time Pad

Đề bài ở file `main.py`.

Bài này sẽ random một key có độ dài là số nguyên tố cùng nhau với độ dài của plaintext. Đặt l là độ dài plaintext và n là độ dài key. Chương trình tạo key mới bằng cách lặp lại key cũ l lần, tương tự plaintext mới sẽ là plaintext cũ lặp lại n lần. Chú ý rằng $\gcd(l, n) = 1$, điều này rất quan trọng giải bài này.

Khi factor độ dài của ciphertext thì mình thấy có hai khả năng xảy ra của độ dài key là 32 hoặc 79. Mình giải với $n = 79$ (sai thì quay lại làm 32 :v).

Như một thói quen, để xử lý các bài toán với số lớn mình thường xem xét những trường hợp số nhỏ để xem mối liên hệ giữa chúng.

Giả sử $l = 5$ và $n = 3$. Đặt key là $\mathbf{k} = (k_0, k_1, k_2)$ và $\mathbf{P} = (p_0, p_1, p_2, p_3, p_4)$.

Khi đó ciphertext sẽ được ghép cặp XOR như sau

$$\begin{bmatrix} k_0 & k_1 & k_2 & k_0 & k_1 & k_2 \\ p_0 & p_1 & p_2 & p_3 & p_4 & p_0 & p_1 & p_2 & p_3 & p_4 & p_0 & p_1 & p_2 & p_3 & p_4 \\ c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} \end{bmatrix}.$$

Nhìn vào k_0 , mình thấy rằng k_0 tác động lần lượt lên p_0, p_3, p_1, p_4 và p_2 , tương ứng với các ciphertext c_0, c_3, c_6, c_9 và c_{12} . Như vậy mình chỉ cần *sắp xếp* lại p_0, p_3, \dots về đúng vị trí của nó là được.

Vì $\gcd(l, n) = 1$ nên mình nhớ tới một tính chất mà chúng ta hay dùng để chứng minh định lý Wilson hoặc Euler là nếu $\{g_1, g_2, \dots, g_{\phi(n)}\}$ là hệ thăng dư thu gọn modulo n và a là số sao cho $\gcd(a, n) = 1$ thì tập

$$\{ag_1 \bmod n, ag_2 \bmod n, \dots, ag_{\phi(n)} \bmod n\}$$

cũng là hệ thăng dư thu gọn modulo n . Nói cách khác hai tập hợp

$$\{g_1, g_2, \dots, g_{\phi(n)}\} \text{ và } \{ag_1 \bmod n, ag_2 \bmod n, \dots, ag_{\phi(n)} \bmod n\}$$

là hoán vị của nhau.

Mình thử như sau:

- với $i = 0$ thì $0 \cdot 3 = 0 \bmod 5$;
- với $i = 1$ thì $1 \cdot 3 = 3 \bmod 5$;
- với $i = 2$ thì $2 \cdot 3 = 1 \bmod 5$;
- với $i = 3$ thì $3 \cdot 3 = 4 \bmod 5$;
- với $i = 4$ thì $4 \cdot 3 = 2 \bmod 5$.

Nhìn chuỗi $(0, 3, 1, 4, 2)$ quen quen. Đó chính là p_0, p_3, p_1, p_4, p_2 ở trên.

Như vậy mình có công thức tổng quát là $p_{i \cdot 3 \bmod 5} = c_{i \cdot 3}$, hay tổng quát hơn $p_{i \cdot n \bmod l} = c_{i \cdot n}$ với $i = 0, 1, \dots, l - 1$.

Sau đó mình chỉ cần bruteforce 256 trường hợp k_0 nữa. Ở đây mình thấy với $k_0 = 165$ thì có chuỗi PNG (?).

Như vậy XOR với 165 sẽ có flag. Code giải mình để ở `main.py`.

Okay vậy là xong bài.

4.5.4 Backdoor CTF 2023

PRSA

Trong bài này chúng ta có đa thức modulo $p(x) \cdot q(x)$. Flag được biến đổi thành dãy nhị phân và là hệ số cho đa thức $n(x) \bmod p(x) \cdot q(x)$. Ciphertext khi đó là $n(x)^e \bmod p(x) \cdot q(x)$.

Chúng ta làm tương tự RSA, factor modulus. Tiếp theo, gọi $d_p = \deg p(x)$ thì số phần tử khác 0 trong $\text{GF}(2^{d_p})$ là $2^{d_p} - 1$. Tương tự cho $q(x)$ số phần tử khác 0 là $2^{d_q} - 1$. Do đó mình tính nghịch đảo của $e = 2^{256}$ trong modulus $(2^{d_p} - 1) \cdot (2^{d_q} - 1)$.

Knapsack

Đề bài cho mình một mảng arr và tùy vào bit của $secret$ mà cộng arr vào s hoặc không. Điều này gợi nhớ về sử dụng lattice để giải bài toán knapsack. Mình tham khảo trong quyển *An introduction to mathematical cryptography* và xây dựng lattice như sau:

$$A_{M,S} = \begin{pmatrix} 2 & 0 & 0 & \cdots & 0 & a_0 \\ 0 & 2 & 0 & \cdots & 0 & a_1 \\ 0 & 0 & 2 & \cdots & 0 & a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2 & a_{39} \\ 1 & 1 & 1 & \cdots & 1 & s \end{pmatrix}$$

Sử dụng thuật toán BKZ trên ma trận $A_{M,S}$ trên được dòng đầu là \bar{n} , nhân \bar{n} với $A_{M,S}^{-1}$ ta được flag.

Note: thuật toán LLL không hiệu quả ở đây còn thuật toán BKZ được sử dụng là do BKZ tốt hơn trong việc giải SVP nhưng chạy lâu hơn (theo giải thích của 1 bạn trong discord giải).

Curvy Curves

Ở bài này cần factor n là có thể giải ra. Tuy nhiên việc factor khá khó và sau giải mình mới biết là sử dụng phương pháp William $p + 1$. NHƯNG, một mạnh thường quân nào đó đã factor ra và mình chỉ lấy từ factordb về (vào ngày thứ 2 của CTF) và giải ra.

Trong bài này thực hiện việc cộng các điểm theo công thức sau. Gọi $P = (x_P, y_P)$ và $Q = (x_Q, y_Q)$ là tọa độ điểm P và Q . Khi đó phép cộng $R = P + Q$ là:

$$x_R = x_P \cdot x_Q + D \cdot y_P \cdot y_Q \bmod n$$

$$y_R = x_1 \cdot y_2 + x_2 \cdot y_1 \bmod n$$

Mình viết dưới dạng ma trận:

$$\begin{pmatrix} x_P & D \cdot y_P \\ y_P & x_P \end{pmatrix} \cdot \begin{pmatrix} x_Q & D \cdot y_Q \\ y_Q & x_Q \end{pmatrix} = \begin{pmatrix} x_R & D \cdot y_R \\ y_R & x_R \end{pmatrix}$$

Khi đó mình ghi lại ma trận $M = \begin{pmatrix} x & D \cdot y \\ y & x \end{pmatrix}$ và ciphertext là $C = M^e = \begin{pmatrix} Cx & D \cdot Cy \\ Cy & Cx \end{pmatrix}$.

Phần này giống với RSA, mình cần tìm order của nhóm các điểm thỏa mãn $x^2 - D \cdot y^2 \equiv 1 \pmod{n}$ (điều kiện này ở hàm `get_point`). Order của nhóm này là $(p+1) \cdot (q+1)$ nên ta chỉ cần tính $d = e^{-1} \bmod (p+1) \cdot (q+1)$ và $M = C^d$. Flag là tọa độ x của M .

Safe Curvy Curve

Bài này tương tự bài trên và cũng có một mạnh thường quân nào đó đã donate factor của n .

Secure Matrix Transmissions

Ở bài này thực hiện trao đổi khóa cho AES. Việc sinh khóa thực hiện như sau:

- Sinh một ma trận *secret* trong $\text{GL}(6, \mathbb{F}_p)$, đặt là S
- Sinh hai ma trận public A và B

Khi đó, Alice tạo khóa bí mật là ma trận chéo a , và tính

$$u = S^{-1} \cdot A \cdot S \cdot a \cdot S^{-1} \cdot A^{-1} \cdot S$$

Tương tự, Bob tạo khóa bí mật là ma trận chéo b , và tính

$$v = S^{-1} \cdot B \cdot S \cdot b \cdot S^{-1} \cdot B^{-1} \cdot S$$

Khóa chung được tính là

$$\begin{aligned} & S \cdot A^{-1} \cdot S \cdot u \cdot S^{-1} \cdot A \cdot S + b \\ &= S \cdot A^{-1} \cdot S \cdot (S^{-1} \cdot A \cdot S \cdot a \cdot S^{-1} \cdot A^{-1} \cdot S) \cdot S^{-1} \cdot A \cdot S + b \\ &= a + b \end{aligned}$$

Mình nhận thấy rằng

$$u = (S^{-1} \cdot A \cdot S) \cdot a \cdot (S^{-1} \cdot A \cdot S)^{-1}$$

với a là ma trận chéo. Như vậy a là một chéo hóa của u nên mình chỉ cần chéo hóa u là tìm được a . Tương tự, chéo hóa v sẽ tìm được b .

Vấn đề ở đây là việc sắp xếp các trị riêng trên các cột của a . Do đó mình brute tất cả các cách xếp 6 trị riêng của u lên a . Tương tự cho $6!$ hoán vị các trị riêng của v lên b .

Rebellious

Bài này thuộc dạng discrete logarithm. Với hai điểm $P = (x_P, y_P)$ và $Q = (x_Q, y_Q)$, kết quả phép cộng $R = P + Q$ là

$$x_R = a^{-1} \cdot (x_P \cdot x_Q - y_P \cdot y_Q \cdot a^2 \cdot b^{-2}) \bmod p$$

$$y_R = a^{-1} \cdot (x_P \cdot y_Q + y_P \cdot x_Q) \bmod p$$

Nếu mình viết dưới dạng ma trận thì phép nhân cho kết quả sau:

$$\begin{pmatrix} x_P & -y_P \cdot a^2 \cdot b^{-2} \\ y_P & x_P \end{pmatrix} \cdot \begin{pmatrix} x_Q & -y_Q \cdot a^2 \cdot b^{-2} \\ y_Q & x_Q \end{pmatrix} = a \begin{pmatrix} x_R & -y_R \cdot a^2 \cdot b^{-2} \\ y_R & x_R \end{pmatrix}$$

Đề bài tương đương việc giải discrete logarithm

$$a^{k-1} \cdot \begin{pmatrix} x_1 & -y_1 \cdot a^2 \cdot b^{-2} \\ y_1 & x_1 \end{pmatrix}^k = \begin{pmatrix} x_2 & -y_2 \cdot a^2 \cdot b^{-2} \\ y_2 & x_2 \end{pmatrix}$$

$$\text{Đặt } M = a \cdot \begin{pmatrix} x_1 & -y_1 \cdot a^2 \cdot b^{-2} \\ y_1 & x_1 \end{pmatrix},$$

$$\text{và } N = a \cdot \begin{pmatrix} x_2 & -y_2 \cdot a^2 \cdot b^{-2} \\ y_2 & x_2 \end{pmatrix} \text{ thì việc giải bài toán tương đương với } M^k = N.$$

Mình chéo hóa ma trận $M = P \cdot D \cdot P^{-1}$. Khi đó $D^k = P^{-1} \cdot N \cdot P$. Do D là ma trận chéo nên việc giải bài toán trên tương đương discrete logarithm trên $\text{GF}(p)$: $D_{0,0}^k = (P^{-1} \cdot N \cdot P)_{0,0} \pmod p$.

Secure Matrix Transmissions v2

Bài này mình không giải ra trong CTF. Nhìn lướt qua thì khá giống bài Secure_Matrix_Transmissions ở trên nhưng đời không như mơ :)))

Bài này đề tạo ba ma trận trong $GL(6, \mathbb{F}_p)$ là a, b và w .

Alice chọn hai số bí mật n, m và tính $a^n \cdot w \cdot b^m$.

Bob chọn hai số bí mật r, s và tính $a^r \cdot w \cdot b^s$.

Khóa chung là $a^{n+r} \cdot w \cdot b^{m+s}$ và được dùng cho AES.

Đối với các bài dạng lũy thừa ma trận thì cách hay được dùng là chéo hóa. Tuy nhiên các ma trận a và b không có đủ 6 trị riêng trong $GF(p)$. Do đó chúng ta cần mở rộng ra một trường lớn hơn chứa tất cả các trị riêng của a và b .

Điều này có nghĩa là, ví dụ mình xét đa thức có hệ số thực là $f(x) = x^2 + 1 = 0$. Các hệ số của $f(x)$ nằm trong trường \mathbb{R} , tuy nhiên các nghiệm của nó lại nằm trong trường \mathbb{C} chứa \mathbb{R} .

Tương tự ở bài này, phương trình đặc trưng là một đa thức bậc 6 với hệ số trong $GF(p)$. Như vậy các trị riêng của nó phải nằm trong một trường nào đó chứa $GF(p)$. Mình chọn $GF(p^{10})$ với đa thức tối giản bắt kì (nào cũng ra).

Khi đó, mình chéo hóa được $a = P^{-1} \cdot c \cdot P$ và $b = Q^{-1} \cdot d \cdot Q$. Thay vào u mình có

$$u = a^n \cdot w \cdot b^m = P^{-1} \cdot c^n \cdot P \cdot w \cdot Q^{-1} \cdot d^m \cdot Q,$$

tương đương với

$$P \cdot u \cdot Q^{-1} = c^n \cdot (P \cdot w \cdot Q^{-1}) \cdot d^m$$

Đặt $U = P \cdot u \cdot Q^{-1}$ và $W = P \cdot w \cdot Q^{-1}$. Nếu mình gọi c_0, c_1, \dots, c_5 là các trị riêng của a , tương tự d_0, d_1, \dots, d_5 là các trị riêng của b thì phương trình trên tương đương:

$$\begin{bmatrix} c_0^n & & \\ & \ddots & \\ & & c_5^n \end{bmatrix} \cdot \begin{bmatrix} W_{00} & & \\ & \ddots & \\ & & W_{55} \end{bmatrix} \cdot \begin{bmatrix} d_0^m & & \\ & \ddots & \\ & & d_5^m \end{bmatrix} = \begin{bmatrix} U_{00} & & \\ & \ddots & \\ & & U_{55} \end{bmatrix}$$

Mình thấy tương đương rằng $c_0^n \cdot W_{00} \cdot d_0^m = U_{00}$ và $c_1^n \cdot W_{10} \cdot d_0^m = U_{10}$. Suy ra

$$\frac{c_1^n}{c_0^n} \cdot \frac{W_{10}}{W_{00}} = \frac{U_{10}}{U_{00}} \pmod{p}$$

Tổng quát hơn thì

$$\left(\frac{c_i}{c_0}\right)^n = \frac{U_{i0}}{U_{00}} \cdot \frac{W_{00}}{W_{i0}} \pmod{p}, \quad 1 \leq i \leq 5$$

Tương tự cho d_i ta có

$$\left(\frac{d_i}{d_0}\right)^m = \frac{U_{0i}}{U_{00}} \cdot \frac{W_{00}}{W_{0i}} \pmod{p}, \quad 1 \leq i \leq 5$$

Như vậy mình có thể tính được $\left(\frac{c_i}{c_0}\right)^n$ và $\left(\frac{d_i}{d_0}\right)^m$. Mình viết lại phép nhân ma trận trên thành:

$$c_0^n \cdot \begin{bmatrix} 1 & & \\ & \ddots & \\ & & (c_5/c_0)^n \end{bmatrix} \cdot \begin{bmatrix} W_{00} & & \\ & \ddots & \\ & & W_{55} \end{bmatrix} \cdot d_0^m \cdot \begin{bmatrix} 1 & & \\ & \ddots & \\ & & (d_5/d_0)^m \end{bmatrix} = \begin{bmatrix} U_{00} & & \\ & \ddots & \\ & & U_{55} \end{bmatrix}$$

Đặt

$$W_1 = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & (c_5/c_0)^n & \\ & & & \end{bmatrix} \cdot \begin{bmatrix} W_{00} & & & \\ & \ddots & & \\ & & W_{55} & \\ & & & \end{bmatrix} \cdot \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & (d_5/d_0)^m & \\ & & & \end{bmatrix}$$

thì $c_0^n \cdot d_0^m \cdot W_1 = U$. Vẽ trái là số nhân với ma trận, do đó ta có thể lấy bất kì vị trí tương ứng của U và W_1 để tính $c_0^n \cdot d_0^m$.

Thực hiện tương tự cho ma trận V mình cũng tính được $c_0^r \cdot d_0^s$. Suy ra

$$\begin{aligned} K &= a^{n+r} \cdot w \cdot b^{m+s} = P^{-1} \cdot c^{n+r} \cdot P \cdot w \cdot Q^{-1} \cdot d^{m+s} \cdot Q \\ &= P^{-1} \cdot c^n \cdot c^r \cdot W \cdot d^m \cdot d^s \cdot Q \\ &= P^{-1} \cdot \begin{bmatrix} c_0^n & & & \\ & \ddots & & \\ & & c_5^n & \\ & & & \end{bmatrix} \cdot \begin{bmatrix} c_0^r & & & \\ & \ddots & & \\ & & c_5^r & \\ & & & \end{bmatrix} \cdot W \cdot \begin{bmatrix} d_0^m & & & \\ & \ddots & & \\ & & d_5^m & \\ & & & \end{bmatrix} \cdot \begin{bmatrix} d_0^s & & & \\ & \ddots & & \\ & & d_5^s & \\ & & & \end{bmatrix} \cdot Q \\ &= (c_0^n \cdot d_0^m) \cdot (c_0^r \cdot d_0^s) \cdot P^{-1} \cdot \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & (c_5/c_0)^n & \\ & & & \end{bmatrix} \cdot \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & (c_5/c_0)^r & \\ & & & \end{bmatrix} \cdot W \\ &\quad \cdot \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & (d_5/d_0)^m & \\ & & & \end{bmatrix} \cdot \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & (d_5/d_0)^s & \\ & & & \end{bmatrix} \cdot Q \end{aligned}$$

Ở đây mình có thể tính tất cả thành phần $c_0^n \cdot d_0^m$, $c_0^r \cdot d_0^s$, và các ma trận. Từ đó mình có thể tìm được ma trận K và dùng nó để lấy flag.

4.5.5 Google CTF 2023

LEAST COMMON GENOMINATOR?

Bài này sinh ra 8 số nguyên tố để mã hóa. Từ một trạng thái seed ban đầu s , số tiếp theo được sinh ra bởi số trước bởi công thức $s_{i+1} = (ms_i + c) \bmod n$.

Chúng ta chỉ biết seed ban đầu, không biết m , c hay n . Để cho chúng ta dãy 6 số đầu tạo bởi chuỗi trên, giả sử là s_0, s_1, s_2, \dots

Mình thấy rằng $s_1 = (ms_0 + c) \bmod n$, $s_2 = (ms_1 + c) \bmod n$ và $s_3 = (ms_2 + c) \bmod n$.

Trừ vế theo vế mình có $s_2 - s_1 = m(s_1 - s_0) \bmod n$ và $s_3 - s_2 = m(s_2 - s_1) \bmod n$.

Suy ra $n \mid (s_2 - s_1) - m(s_1 - s_0)$ và $n \mid (s_3 - s_2) - m(s_2 - s_1)$. Như vậy nhân chéo lên để khử m mình thu được

$$\begin{cases} n \mid (s_2 - s_1)^2 - m(s_1 - s_0)(s_2 - s_1) \\ n \mid (s_3 - s_2)(s_1 - s_0) - m(s_2 - s_1)(s_1 - s_0) \end{cases},$$

hay tương đương với

$$n \mid (s_2 - s_1)^2 - (s_3 - s_2)(s_1 - s_0)$$

Tương tự như vậy với các cặp khác, dùng gcd mình sẽ tìm ra được n .

Khi đã có n , nhớ lại rằng $s_2 - s_1 = m(s_1 - s_0) \bmod n$ sẽ suy ra được m . Cuối cùng do $s_1 = (ms_0 + c) \bmod n$ nên sẽ tìm được c .

Có đủ n , m và c mình chạy hàm như đề bài là sẽ tìm được các số nguyên tố từ đó decrypt ra được kết quả.

Cursved

Bài này định nghĩa một đường cong (có lẽ vậy) là tập hợp các điểm thỏa phương trình $x^2 = Dy^2 + 1 \pmod{p}$ với $D = 3$ và p là số nguyên tố cho trước.

Phép cộng hai điểm (x_1, y_1) và (x_2, y_2) được định nghĩa là điểm $(x_1x_2 + Dy_1y_2, x_1y_2 + x_2y_1)$ (tất cả trong modulo p).

Để bài cho chúng ta biết điểm generator là $G = (2, 1)$, public key là một điểm không đổi trong tất cả lần chạy (điểm P). Mình gọi k là private key, là một số trong modulo $n = p - 1$ sao cho $P = kG$.

Hướng tấn công của bài này là xây dựng một homomorphism từ nhóm các điểm trên lên $\text{GF}(p)$ và giải discrete logarithm trên $\text{GF}(p)$. Mình xét ánh xạ

$$\varphi : \mathbb{F}_p \times \mathbb{F}_p \rightarrow \mathbb{F}_p, \quad (x, y) \mapsto x + yW$$

Ở đây W là một số nào đó thuộc $\text{GF}(p)$. Mình muốn ánh xạ này là homomorphism thì tương đương với điều kiện

$$\varphi(x_1, y_1) \cdot \varphi(x_2, y_2) = \varphi(x_1x_2 + Dy_1y_2, x_1y_2 + x_2y_1)$$

Tương đương với

$$(x_1 + y_1W) \cdot (x_2 + y_2W) = (x_1x_2 + Dy_1y_2) + (x_1y_2 + x_2y_1)W$$

Khai triển và thu gọn mình có được $W^2 = D$. Như vậy $W = \sqrt{3} \pmod{p}$.

Và đúng là D là số chính phương modulo p thật, quá tốt :))))

Như vậy mình sẽ chuyển việc tính toán discrete logarithm $P = kG$ trên tập hợp các điểm trên thành việc tính toán discrete logarithm ứng với phương trình

$$(x_G + y_GW)^k = (x_P + y_FW) \pmod{p}$$

Tới đây thì khá bế tắc vì $p - 1$ có các thừa số rất lớn. Mình đi "hỏi thăm" 1 vòng thì biết có tool open source rất mạnh để tính việc này là [cado-nfs](#).

Đặt $g = x_G + y_GW \pmod{p}$ và $h = x_P + y_FW \pmod{p}$. Lúc giải mình thấy một điều lạ lùng là mình không dùng tool trên tính dlog modulo 2 được, nên mình chỉ tính dlog của g và h trên modulo $(p - 1)/2$. Cado-nfs sẽ trả về base, mình gọi là b , và dlog của g và h lần lượt là t_1 và t_2 .

Do $g = b^{t_1}$, $h = b^{t_2}$, $h = g^k$ nên $b^{t_2} = b^{t_1k} \pmod{p}$. Suy ra $t_2 = t_1k \pmod{\frac{p-1}{2}}$, từ đó mình tìm được k và t_1 này thỏa mãn $P = kG$.

Như vậy mình đã khôi phục lại private key và sẵn sàng ký bất cứ *nonce* nào gửi tới.

MHK2

1. Tạo private key

Để tạo private key, server random hai dãy số $s_1 = (s_{11}, s_{12}, \dots, s_{1n})$ và $s_2 = (s_{21}, s_{22}, \dots, s_{2n})$ với $n = 256$ và s_{ij} 64 bit.

Hệ thống sinh dãy số cho tới khi $p_1 = \sum_{i=1}^n s_{1i} + 2$ là số nguyên tố, tương tự cho $p_2 = \sum_{i=1}^n s_{2i} + 2$ là số nguyên tố. Sau đó hệ thống chọn ngẫu nhiên số $e_1 \in \left[\frac{p_1}{2}, p_1\right]$ và $e_2 \in \left[\frac{p_2}{2}, p_2\right]$.

Cuối cùng tính

$$s = (s_{11} + s_{21}, s_{12} + s_{22}, \dots, s_{1n} + s_{2n}) = (s_1, s_2, \dots, s_n).$$

2. Tạo public key

Từ hai dãy s_1 và s_2 ở trên hệ thống tính hai dãy $a_1 = (e_1 * s_{11}, e_1 * s_{12}, \dots, e_1 * s_{1n})$ (tất cả trong modulo p_1) và $a_2 = (e_2 * s_{21}, e_2 * s_{22}, \dots, e_2 * s_{2n})$ (tất cả trong modulo p_2).

Đặt

$$b_1 = (s_{11} \bmod 2, s_{12} \bmod 2, \dots, s_{1n} \bmod 2).$$

Đặt

$$b_2 = (s_{21} \bmod 2, s_{22} \bmod 2, \dots, s_{2n} \bmod 2).$$

Đặt

$$b = (s_1 \bmod 2, s_2 \bmod 2, \dots, s_n \bmod 2).$$

Cuối cùng, c được chọn ngẫu nhiên, hoặc là b_1 , hoặc là b_2 .

3. Mã hóa

Để mã hóa một bit, bit $\in \{0, 1\}$, đầu tiên tạo hai dãy ngẫu nhiên $r_1 = (r_{11}, r_{12}, \dots, r_{1n})$ và $r_2 = (r_{21}, r_{22}, \dots, r_{2n})$ với $r_{ij} \in \{0, 1\}$.

Khi đó ciphertext là $c_1 = \sum_{i=1}^n a_{1i} \cdot r_{1i}$ và $c_2 = \sum_{i=1}^n a_{2i} \cdot r_{2i}$ (điều kiện đổi với bit ở m_1, m_2 và eq nhưng hiện tại chúng ta không cần dùng tới).

4. Khôi phục private key

Chi tiết cách giải bài này mình tham khảo writeup của Google và Mystiz. Về mặt lý thuyết mình không hiểu sao họ có thể nghĩ ra lattice hay như vậy :)))) Do đó ở đây mình chỉ trình bày cách xây dựng lattice để giải bài này.

Ý tưởng là từ public key a_1 (a_2 tương tự) mình sẽ khôi phục lại p_1 và e_1 .

Đặt $A_1 = \sum_{i=1}^n a_{1i}$.

Do $a_{1i} = e_1 \cdot s_{1i} \bmod p_1$ với $i = \overline{1, n}$ nên $A_1 = \sum_{i=1}^n e_1 \cdot s_{1i} = e_1 \sum_{i=1}^n s_{1i}$. Mà mình đã biết $p_1 = \sum_{i=1}^n s_{1i} + 2$ nên phương trình trên tương đương với $A_1 = e_1 * (p_1 - 2)$. Lấy modulo p_1 ta có $A_1 \equiv -2e_1 \bmod p_1$.

Từ việc $a_{1i} \equiv e_1 \cdot s_{1i} \bmod p_1$, ta có $2a_{1i} \equiv 2e_1 \cdot s_{1i} \equiv -A_1 \cdot s_{1i} \bmod p_1$.

Nghĩa là tồn tại số $x_{1i} \in \mathbb{Z}$ sao cho $2a_{1i} + A_1 \cdot s_{1i} = x_{1i} \cdot p_1$ với $i = \overline{1, n}$.

Mình có thể đánh giá x_{1i} đơn giản như sau:

$$x_{1i} = \frac{2a_{1i} + A_1 s_{1i}}{p_1} \leqslant \frac{2 \max(a_{1i}) + A_1 \cdot 2^{128}}{\max(a_{1i})}$$

xấp xỉ 136 bit.

Lấy $i = 1$ ta có $2a_{11} + A_1 \cdot s_{11} = x_{11} \cdot p_1$ (1).

Lấy $2 \leq i \leq n$ ta có $2a_{1i} + A_1 \cdot s_{1i} = x_{1i} \cdot p_1$ (2).

Để khử p_1 mình nhân hai vế phương trình (1) cho x_{1i} và nhân hai vế phương trình (2) cho x_{11} , rồi trừ vế theo vế thu được

$$2a_{11} \cdot x_{1i} - 2a_{1i} \cdot x_{11} + A_1(s_{11} \cdot x_{1i} - s_{1i} \cdot x_{11}) = 0.$$

Ở đây chúng ta đã biết a_{11} , a_{1i} và A_1 . Việc xây dựng lattice sẽ dựa trên các hệ số x_{1i} , x_{11} và $s_{11} \cdot x_{1i} - s_{1i} * s_{11}$. Do đó mình viết

$$\begin{array}{ll} x_{11} & \cdot (\dots, -2a_{1i}, \dots) \\ x_{1i} & \cdot (\dots, 2a_{11}, \dots) \\ \dots & \cdot (\dots, \dots, \dots) \\ (s_{11} \cdot x_{1i} - s_{1i} * s_{11}) & * (\dots, A_1, \dots) \\ \dots & \cdot (\dots, \dots, \dots) \end{array}$$

Với $i = \overline{2, n}$ thì mình sẽ cần 255 cột. Tuy nhiên chúng ta thường sử dụng 1 để khi tính ra lattice thì sẽ thu được số cần tìm, cụ thể ở đây là x_{11} và $s_{11} \cdot x_{1i} - s_{1i} * x_{11}$. Do đó mình sẽ modify nhẹ lại lattice trên thành

$$\begin{array}{ll} x_{11} & \cdot (1, \dots, -2a_{1i}, \dots) \\ x_{1i} & \cdot (\dots, \dots, 2a_{11}, \dots) \\ \dots & \cdot (\dots, \dots, \dots, \dots) \\ (s_{11} \cdot x_{1i} - s_{1i} * s_{11}) & * (0, 1, \dots, A_1, \dots) \\ \dots & \cdot (\dots, \dots, \dots, \dots) \end{array}$$

Ở dòng $s_{11} \cdot x_{1i} - s_{1i} * s_{11}$ chúng ta không muốn nó đụng độ với số x_{11} bên trên nên sẽ dịch qua phải một ô. Ta cần 255 cột như vậy vì sẽ có $i = \overline{2, 256}$. Và để sắp xếp $-2a_{1i}$ cũng cần 255 cột như đã nói ở trên. Vậy lattice này có $1 + 255 + 255 = 511$ cột và cũng là số hàng. Lattice như vậy sẽ có dạng ma trận như sau

$$\begin{pmatrix} 1 & 0 & 0 & \dots & -2a_{12} & -2a_{13} & \dots \\ 0 & 0 & 0 & \dots & 2a_{11} & 0 & \dots \\ 0 & 0 & 0 & \dots & 0 & 2a_{11} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 1 & \dots & \dots & A_1 & 0 & \dots \\ 0 & 0 & 1 & \dots & 0 & A_1 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix}$$

Tới lúc này chúng ta đã hoàn thành ... 10% chặng đường. Lý do là vì với lattice này chúng ta không thể giải ra short vector mong muốn được. Dựa trên lattice mình hy vọng sẽ chạy ra vector

$$v = (x_{11}, s_{11} \cdot x_{12} - s_{12} \cdot x_{11}, \dots, s_{11} \cdot x_{1n} - s_{1n} \cdot x_{11}, 0, 0, \dots, 0)$$

Tuy nhiên cần nhớ rằng s_{1i} có 64 bit và x_{1i} xấp xỉ 136 bit. Bằng tính toán có thể thấy $s_{11} \cdot x_{1i} - s_{1i} \cdot x_{11}$ cũng có 64 bit. Như vậy chúng ta cần scale lattice trên để các giá trị không sai khác nhau quá lớn.

Cụ thể, do x_{11} có 136 bit nên ta sẽ nhân với $\frac{1}{2^{136}}$, do $s_{11} \cdot x_{1i} - s_{1i} \cdot x_{11}$ có 64 bit nên ta sẽ nhân với $\frac{1}{2^{128}}$. Đối với $-2a_{1i}$, $2a_{11}$ và A_1 thì mình cần chúng lớn để lattice tìm ra short vector, nên nhân với 2^{1024} .

Sử dụng tính chất

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \times \begin{pmatrix} b_1 & 0 & 0 & 0 \\ 0 & b_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & b_n \end{pmatrix} = \begin{pmatrix} a_{11}b_1 & a_{12}b_2 & \dots & a_{1n}b_n \\ a_{21}b_1 & a_{22}b_2 & \dots & a_{2n}b_n \\ \dots & \dots & \dots & \dots \\ a_{n1}b_1 & a_{n2}b_2 & \dots & a_{nn}b_n \end{pmatrix}$$

Mình sẽ nhân lattice trên với ma trận sau là sẽ scale được hệ số theo nhu cầu

$$\begin{pmatrix} \frac{1}{2^{136}} & & & & & \\ & \frac{1}{2^{128}} & & & & \\ & & \ddots & & & \\ & & & \frac{1}{2^{128}} & & \\ & & & & 2^{1024} & \\ & & & & & \ddots \\ & & & & & & 2^{1024} \end{pmatrix}$$

Phần tử đầu tiên của short vector là x_{11} (có lẽ vậy :v). Thật ra thì phần tử đầu tiên của short vector là $x_{11} \bmod a_{11}$ hoặc $-x_{11} \bmod a_{11}$. Phía trên mình đã tính được chẵn trên của x_{11} là $\frac{2 \max(a_{1i}) + A_1 \cdot 2^{128}}{\max(a_{1i})}$. Từ đây mình duyệt vòng for qua các giá trị $x_{11} + ka_{11}$ và $-x_{11} + ka_{11}$ để tìm các số s_{1i} theo cách sau

a. Khôi phục modulo

Nhắc lại $2a_{11} + A_1 \cdot s_{11} = x_{11} \cdot p_1$. Modulo hai về cho A_1 thì được $2a_{11} = x_{11} * p_1 \bmod A_1$. Ở đây có thể tìm được $p_1 = 2a_{11}x_{11}^{-1} \bmod A_1$. Tuy nhiên chúng ta cần lưu ý rằng điều kiện để tồn tại nghịch đảo là $\gcd(x_{11}, A_1) = 1$. Do đó để đảm bảo chúng ta tính toán thêm một bước.

Đặt $g = \gcd(2a_{11}, A_1)$. Khi đó phương trình đồng dư có nghiệm khi và chỉ khi $\gcd(x_{11} \cdot p_1, A_1)$ chia hết cho g . Sau khi thỏa các điều kiện này, đặt $\frac{A_1}{g} = A'_1$, $2a'_{11} = \frac{2a_{11}}{g} \bmod \frac{A_1}{g}$, và $x'_{11} = \frac{x_{11}}{g} \bmod \frac{A_1}{g}$. Khi đó $p_1 = 2a'_{11} \cdot x'^{-1}_{11} \bmod A'_1$.

b. Khôi phục e

Bên trên mình đã có $A_1 = -2e_1 \bmod p_1$ nên mình sẽ tìm được e_1 . Sau đó từ $a_1 = (a_{11}, a_{12}, \dots, a_{1n})$ và e_1 mình tính lại được tất cả s_{1i} và kiểm tra xem $\sum_{i=1}^n s_i + 2 \equiv p_1$, đồng thời không có số s_{1i} nào vượt quá 64 bit.

c. Giải

Đoạn code mình lấy của Google. Lưu ý rằng sau khi LLL xong mình cần scale ngược lại độ lớn ban đầu, ở đây là chia cho ma trận Q . Do một chút lười nên ở đây thay a_2 thành a_1 để áp dụng cho p_1 .

Cuối cùng chạy decrypt và lấy flag thôi.

```
# CTF{faNYPAcKs_ARe_4maZiNg_AnD_und3Rr@t3d}
```

MYTLS

Bài này dựa trên nguyên lý handshake của TLS (mãi sau mình đàm đạo với các bạn khác mới biết :v). Trong bài này đề cho các file sau:

- admin-ecdhcert.pem
- ca-crt.pem
- Dockerfile
- guest-ecdhcert.pem
- guest-ecdhkey.pem
- server-ecdhcert.pem
- server.py
- start.sh

Ở bài này xảy ra hai công đoạn trao đổi khóa, mình sẽ gọi là *share_key* và *share_ephemeral_key*.

Đối với *share_ephemeral_key*, mình sẽ cần gửi lên một chuỗi hex độ dài 32 ký tự, và một public key ECDH theo dạng PEM. Server cũng sẽ tạo một chuỗi hex độ dài 32 ký tự, private key và public key ECDH, sau đó gửi public key ECDH này về cho mình cũng ở dạng PEM.

Khi đó *share_ephemeral_key* sẽ được tính là

```
server_ephemeral_secret = client_ephemeral_private_key.exchange(
    ec.ECDH(), server_ephemeral_public_key)
```

Điều này là tương đương với đoạn code sau của đề vì việc trao đổi khóa sẽ giống nhau ở hai bên trao đổi khóa.

```
server_ephemeral_secret = server_ephemeral_key.exchange(
    ec.ECDH(), client_ephemeral_public_key)
```

Do đó mình chỉ cần tạo một ECDH key mỗi lần trao đổi khóa, hoặc tạo một lần cũng được.

Tiếp theo là `share_key`.

Ở đầu bài, mình sẽ cần cung cấp một public key (certificate) cho server. Server sẽ sử dụng `ca-crt.pem` để verify xem cert của mình có hợp lệ không (có được ký bởi CA không). Server cũng sẽ đọc private key tương ứng (ở file `server-ecdhkey.pem`) và tiến hành trao đổi khóa ECDH lần hai.

```
server_secret = server_key.exchange(ec.ECDH(), client_cert.public_key())
```

Đoạn code trên thực hiện việc thực hiện trao đổi khóa với certificate (public key) nhận từ client, và server private key.

Mình thấy rằng đề đã cung cấp cho mình hai file cert và key của guest đã được ký bởi CA. Do đó mình dùng `guest-ecdhcert.pem` làm `client_cert`, và đề cũng đã cho mình `server-ecdhcert.pem` nên mình có thể tính toán khóa trao đổi ở phía mình.

```
server_secret = client_key.exchange(ec.ECDH(), server_cert.public_key())
```

Đoạn code sau đây sẽ thực hiện việc bắt tay trên TLS.

Tuy nhiên tới đây chúng ta vấp phải một vấn đề, đó là subject được ký trên certificate phải là `admin.tls`, trong khi nếu sử dụng `guest-ecdhcert.pem` thì subject là `guest.tls`. Hmm tình hình có vẻ khá phức tạp. Tới đây thì mình chịu chết, sau giải mới làm ra :D

Khi nhìn vào file Docker, mình thấy rằng file `server-ecdhkey.pem` cũng được chép vào container. Từ đó, ở mỗi lần chạy vòng lặp, mình sẽ chỉ định path của file thành `..../app/server-ecdhkey.pem` để thoát khỏi path `/tmp/storage`, vì khi đó mình sẽ "đọc" được một ít thông tin nào đó từ `server-ecdhkey.pem`.

Do `server-ecdhcert.pem` có 241 bytes, mình đoán rằng `server-ecdhkey.pem` cũng có 241 bytes. Do đó chiến thuật của mình là ghi đè lên 240 bytes đầu của `server-ecdhkey.pem`, server sẽ trả về hash $H(240_bytes_ghi_đè // byte_cuối)$. Mình có thể bruteforce byte cuối với 240 bytes ghi đè ban đầu fix sẵn.

Sau đó mình connect lại. Với byte cuối đã biết, mình sẽ bruteforce byte kế cuối với 239 bytes ghi đè. Lúc này server trả về hash $H(239_bytes_ghi_đè // byte_kế_cuối // byte_cuối)$.

Như vậy mình bruteforce từ dưới lên với công thức $((240-số bytes đã biết) // byte_còn_brute // bytes_đã_biết)$ (có 241 bytes).

Full code để bruteforce `server-ecdhkey.pem`:

Sau khi đã có `server-ecdhkey.pem`, mình quay lại bypass phần kiểm tra subject của certificate. Trong các file cert được cho thì có `admin-ecdhcert.pem` là có subject chúng ta cần (CN=admin.tls). Mình đã có `server-ecdhcert.pem` và `server-ecdhkey.pem` (vừa leak) để tiến hành trao đổi khóa. Tuy nhiên `server-ecdhcert.pem` thì lại không có CN=admin.tls.

Ở đây mình gửi lên `admin-ecdhcert.pem`, nhưng khi tính khóa trao đổi thì sử dụng `server-ecdhkey.pem` (không gửi `server-ecdhcert.pem`) và nó đã work!!! Đọc writeup của mọi người thì họ gọi là KCI attack do tính chất trao đổi khóa Diffie-Hellman.

Mình sửa đổi một tí file trên để lấy flag (gửi lên cert là *admin-ecdhcert.pem* và dùng *server-ecdhkey* để tính *share_key*).

PRIMES

Đề bài cho mình một dãy các số nguyên tố p_i cố định và một số nguyên tố q .

Giả sử plaintext được biểu diễn ở dạng chuỗi bit $b = (b_1, b_2, \dots, b_n)$ thì ciphertext sẽ là $x = \prod_{i=1}^n p_i^{b_i} \pmod{q}$.

Tuy nhiên message m mà đề cho không encrypt ra x tương ứng, mình sẽ gọi là y . Nghĩa là nếu biểu diễn $m = (m_1, m_2, \dots, m_n)$ thì mình có $y = \prod_{i=1}^n p_i^{m_i} \pmod{q}$. Ở đây m và flag có cùng độ dài và sự sai lệch bit không đáng kể.

Hmm, sai lệch bit không đáng kể? :v

Đặt

$$e = xy^{-1} = \prod_{i=1}^n p_i^{b_i - m_i} = \prod_{i=1}^n p_i^{e_i} \pmod{q}.$$

Khi đó $e_i \in \{-1, 0, 1\}$.

Suy ra $e = nd^{-1} \pmod{q}$ với n và d phân tích được thành lũy thừa dương của các p_i , từ đó tồn tại $s \in \mathbb{Z}$ để $ed = n + sq$.

Do $|ed - sq| = n$ nên tương đương với $\left| \frac{ed}{qd} - \frac{sq}{qd} \right| = \frac{n}{qd}$, hay $\left| \frac{e}{q} - \frac{s}{d} \right| = \frac{n}{qd}$.

Bài này dựa trên ý tưởng [xấp xỉ Diophantine](#). Ở đây điều kiện để tồn tại chuỗi liên phân số hội tụ là $nd < \frac{q}{2}$.

Khi đó $\frac{s}{d}$ là liên phân số hội tụ dần tới $\frac{1}{d^2}$ do $\frac{n}{qd} < \frac{1}{d^2}$.

Sử dụng SageMath mình có thể tìm được dãy các phân số $\frac{s_j}{d_j}$ hội tụ tới $\frac{1}{d^2}$. Do đó chiến thuật để giải bài này là tìm d_j mà có thể phân tích thành lũy thừa không âm của các p_i (tương ứng y). Sau đó với $d_j e$ mà cũng có thể phân tích thành lũy thừa không âm của các p_i (tương ứng với x) thì ta sẽ tìm được phân tích tốt nhất để sửa các bit error e_i .

4.5.6 Dice CTF 2023

Provably Secure 1/2

Chúng ta có 3 lựa chọn (cho đường đồi) như sau:

- **Solve:** chỉ ra m_{bit} là 0 hay 1, nếu đúng thì vượt 1 ải, vượt thành công 128 ải thì qua môn!!!;
- **Query Encryption:** ở mỗi round server tạo hai key RSA để mình dùng. Mình cần nhập hai message có độ dài 16 byte và gửi lên server. Dựa vào bit random m_{bit} mà server sẽ encrypt m_0 hay m_1 . Server trả về ciphertext tương ứng (hai ciphertext với tổng độ dài 512 byte);
- **Query Decryption:** để decrypt mình cần gửi lên server ciphertext 512 byte và chỉ được decrypt mỗi ciphertext một lần.

NOTE: mình chỉ được encrypt và decrypt 8 lần mỗi loại.

Hàm encrypt làm việc như sau:

- lấy tham số là hai public key pk_0 và pk_1 , và plaintext msg có 16 byte;

- random một chuỗi r có 16 byte;
- tính $r_prime = r \oplus msg$;
- tính $ct_0 = \text{ENC}(r, pk_0)$ và $ct_1 = \text{ENC}(r_prime, pk_1)$;
- cả hai ciphertext đều dùng padding là OAEP và hash là SHA256 nên độ dài là 256 byte. Hàm trả về ghép hai chuỗi ciphertext lại (512 byte).

Hàm decrypt làm ngược lại và ở kết quả cuối thì xor hai plaintext lại ($msg = r_prime \oplus r$).

Đề chỉ cho hai public key, mình thì thích private key hơn. :))

Mình nhận ra một điều, giả sử ciphertext của mình là $\text{ENC}(r, pk_0)$ và $\text{ENC}(r_prime = r \oplus msg, pk_1)$ thì nếu mình gửi hai lần decrypt với các ciphertext:

- $\text{ENC}(r, pk_0)$ và $\text{ENC}(00^{16}, pk_1)$ thì kết quả trả về là $r \oplus 00^{16} = r$;
- $\text{ENC}(00^{16}, pk_0)$ và $\text{ENC}(r_prime = r \oplus msg, pk_1)$ thì kết quả trả về là $00^{16} \oplus r \oplus msg = r \oplus msg$.

Khi đó mình xor hai plaintext này lại là được msg và so sánh xem nó trùng với m_0 hay m_1 mình gửi lên server ban đầu.

NOTE: thật ra ở bài 1 không có công đoạn kiểm tra ciphertext không nằm trong ciphertext có sẵn nên mình chỉ cần gửi lên ciphertext vừa được encrypt là ra. Code trên được dùng để giải cả hai bài Provably Secure 1/2.

Flag **Provably Secure**: `dice{yeah_I_lost_like_10_points_on_that_proof_lmao}`.

Flag **Provably Secure 2**: `dice{my_professor_would_not_be_proud_of_me}`.

BBBB

Bài này khoai thật sự. :))))

Bài này giống một phần bài **BBB** của giải SECCON nhưng có chút khác bợt.

Đề cho mình một số nguyên tố p 512 bit và số b nhỏ hơn p . Mình cần nhập số a và từ đó các số mũ e dùng trong RSA sẽ được tạo bởi hàm `rng`.

Dựa trên bài của giải SECCON, chiến thuật làm bài này là cố gắng khiến hàm `rng` tạo nhiều $e = 11$ nhất có thể (ở bài này sẽ là 3 vì rất khó lấy đủ 5).

Điểm khó của bài này là hàm `rng` tuyển tính, nghĩa là với mỗi output chỉ tìm được đúng một input. Tuy nhiên chúng ta có thể tìm `rng` sao cho các input tạo thành vòng, nói cách khác là:

$$\begin{aligned} aX_i + b &\equiv X_{i+1} & (\text{mod } p) \\ aX_{i+1} + b &\equiv X_{i+2} & (\text{mod } p) \\ aX_{i+2} + b &\equiv X_{i+3} & (\text{mod } p) \\ aX_{i+3} + b &\equiv X_{i+4} & (\text{mod } p) \\ aX_{i+4} + b &\equiv X_i & (\text{mod } p) \end{aligned}$$

Ở đây $X_i = X_{i+5}$, tức là sau 5 lần thì các giá trị X_i lặp lại và mình sẽ tìm a thỏa mãn đồng này.

Trừ phương trình dưới cho phương trình trên về theo v , mình có:

$$\begin{aligned} a(X_{i+1} - X_i) &\equiv X_{i+2} - X_{i+1} & (\text{mod } p) \\ a(X_{i+2} - X_{i+1}) &\equiv X_{i+3} - X_{i+2} & (\text{mod } p) \\ a(X_{i+3} - X_{i+2}) &\equiv X_{i+4} - X_{i+3} & (\text{mod } p) \\ a(X_{i+4} - X_{i+3}) &\equiv X_i - X_{i+4} & (\text{mod } p) \\ a(X_i - X_{i+4}) &\equiv X_{i+1} - X_i & (\text{mod } p) \end{aligned}$$

Như vậy $a^5 \equiv 1 \pmod{p}$. Nếu phương trình này có nghiệm khác 1 thì ta chọn làm tham số a .

Tiếp theo, mình cần $e = 11$ nằm trong vòng lặp này nên mình cứ chọn $X_0 = e = 11$ rồi theo trình tự $X_{i+1} = aX_i + b \pmod{p}$ thôi.

Câu hỏi là tại sao lại cần tới 3 bộ có $e = 11$ mà không phải 2?

Vì với mỗi public key 2048 bit, phương pháp Coppersmith sẽ hoạt động hiệu quả khi $(2048*T) \times (1/11 - \varepsilon) \approx 8 \times (L + 4)$ với L là độ dài flag (tối đa là 49) và 4 byte random. Như vậy khi $T = 3$ thì $\varepsilon > 0$ là điều mình cần nhắm tới. Minh tham khảo ở [hastads](#).

Cuối cùng, sử dụng CRT và hastad attack (ví dụ như ở [github](#)) để giải.

Mình cần lưu ý rằng mình đã biết 5 byte đầu của flag là `dice{` nên mình có thể giảm độ dài flag cần tìm xuống, từ đó Coppersmith sẽ hiệu quả hơn.

Mỗi phương trình của mình có dạng $(\text{FLAG} \cdot 256^{16} + r_i)^e = c_i \pmod{n_i}$ mà FLAG có 5 byte đầu là `dice{` nên mình có thể gọi `C = bytes_to_long(b"dice{") << (8 * (L + 4 + 16))`. Phương trình trở thành

$$(\text{FLAG}' \cdot 256^{16} + C + r_i)^e = c_i \pmod{n_i}.$$

NOTE: việc chọn beta và epsilon khá khó khăn, công thức trong code là từ writeup người giải ra, và không phải lúc nào cũng có a thỏa cũng như đủ 3 bộ có $e = 11$.

4.5.7 UTCTF 2023

Đề và bài giải ở [đây](#).

Affinity

Đề cho mình hai source code: `aes.py` và `encrypt_pub.py`.

Trong bài này có điểm khác thường so với thuật toán AES gốc là **SBOX tuyến tính**. Điều đó có nghĩa là với 2 cặp plaintext-ciphertext bất kì (P_0, C_0) và (P_i, C_i) thì có một ma trận A thỏa $P_0 \oplus P_i = A \cdot (C_0 \oplus C_i)$.

Lưu ý là mỗi block AES có 16 byte tương đương 128 bit, nên A là ma trận 128×128 trên GF(2). Như vậy mình chọn $P_0 = 0^{128}$, và $P_i = (0, \dots, 1, \dots, 0)$ với 1 nằm ở vị trí i ($i = 1, \dots, 128$). Với đủ 128 cặp như vậy mình khôi phục được ma trận A từ đó suy ra flag với $P_t = P_0 \oplus A \cdot (C_0 \oplus C_t)$.

Provably Insecure

Ở bài này Alice cho chúng ta public key RSA (n, e) , và cặp (s, m) với $s = m^d \pmod{n}$.

Nhiệm vụ là tìm các số (n', e', d') sao cho với mỗi số random x thì các điều kiện sau thỏa mãn:

- $n' \neq n$ và $e' \neq e$;
- $n' > s$;
- $x^{e'd'} = 1 \pmod{n'}$;
- $e' > 1$;
- $s^{e'} = m \pmod{n'}$.

Quan trọng là ở điều kiện cuối, ở hai modulo khác nhau nhưng cho cùng kết quả khi decrypt.

Cách làm của mình là, nếu $s^{e'} = m \pmod{n'}$ thì tương đương với $(m^{d'})^{e'} = m \pmod{n'}$. Như vậy mình chọn modulo n' và tính discrete log $m^{d'} = s \pmod{n'}$ và có d' . Sau đó mình tìm nghịch đảo e' của d' trong $\varphi(n')$.

Cách làm này cần 2 điều kiện:

- tồn tại discrete log $m^{d'} = s$ trong modulo n' ;

- $\gcd(d', \varphi(n')) = 1$ để tìm e' .

Mình cứ request tới khi có (s, m) thỏa mãn thôi. Với cách làm này thì không cần quan tâm (n, e) .

Looks Wrong tom E

Bài này có 10 round, mỗi round sẽ sinh một số lượng ma trận trong $\text{GF}(10^9 + 7)$ (tối đa 10 ma trận mỗi round).

Với mỗi ma trận, server tạo random vector s và error nhỏ e , rồi tính $s \cdot A + e$ và gửi ma trận A lẫn vector $s \cdot A + e$ cho mình.

Nhiệm vụ của mình là chọn ma trận trong số 10 ma trận đó (hoặc ít hơn), và gửi lên vector s' sao cho tích $s \cdot A$ là vector nhỏ. Cụ thể là $b = s \cdot A = (b_1, b_2, \dots)$ thì $b_i < 4w$ hoặc $-b_i < 4w$.

Sau nhiều nỗ lực nghĩ LLL thì mình phát hiện rằng mình cứ gửi $s = (0, 0, \dots)$ thì kết quả luôn là vector 0 :))

Bài cuối mình không thấy sự liên quan hay gì từ đề nên ngâm ngùi chịu chết.

Cám ơn các bạn đã đọc writeup của mình.

4.5.8 Wolv CTF 2023

keyexchange

Description: Diffie-Hellman is secure right

Với ba số nguyên tố n, s, a , server gửi mình số $s^a \pmod{n}$. Mình cần nhập số b và server trả về $(s^a)^b \pmod{n}$. Mình chỉ cần chọn $b = 1$ là **secret_key** sẽ là số $s^a \pmod{n}$ được nhận lúc này. Từ đó decrypt ra flag.

Z2kDH

Description: Sick of how slow it is to take the modulus of a large prime number? Tired of performing exponentiation over some weird group like ed25519? Just use the integers mod 2^k group with the unit g = 5! Efficient, reliable, doesn't require any hard math!

Với output là

```
99edb8ed8892c664350acbd5d35346b9b77dedfae758190cd0544f2ea7312e81
40716941a673bbda0cc8f67fdf89cd1cf1f22a92fe509411d5fd37d4cb926af
```

Gọi a, b lần lượt là private key của Alice và Bob. Khi đó output trên tương đương với $(g^a \pmod{p})//4$ và $(g^b \pmod{p})//4$ với $p = 2^{258}$.

Như vậy mình cần bruteforce 2 bit cuối của mỗi public key và từ đó tìm ra private key của từng người. Cuối cùng thế vào hàm Z2kDH_exchange để tìm shared key là flag.

Galois-t is this?

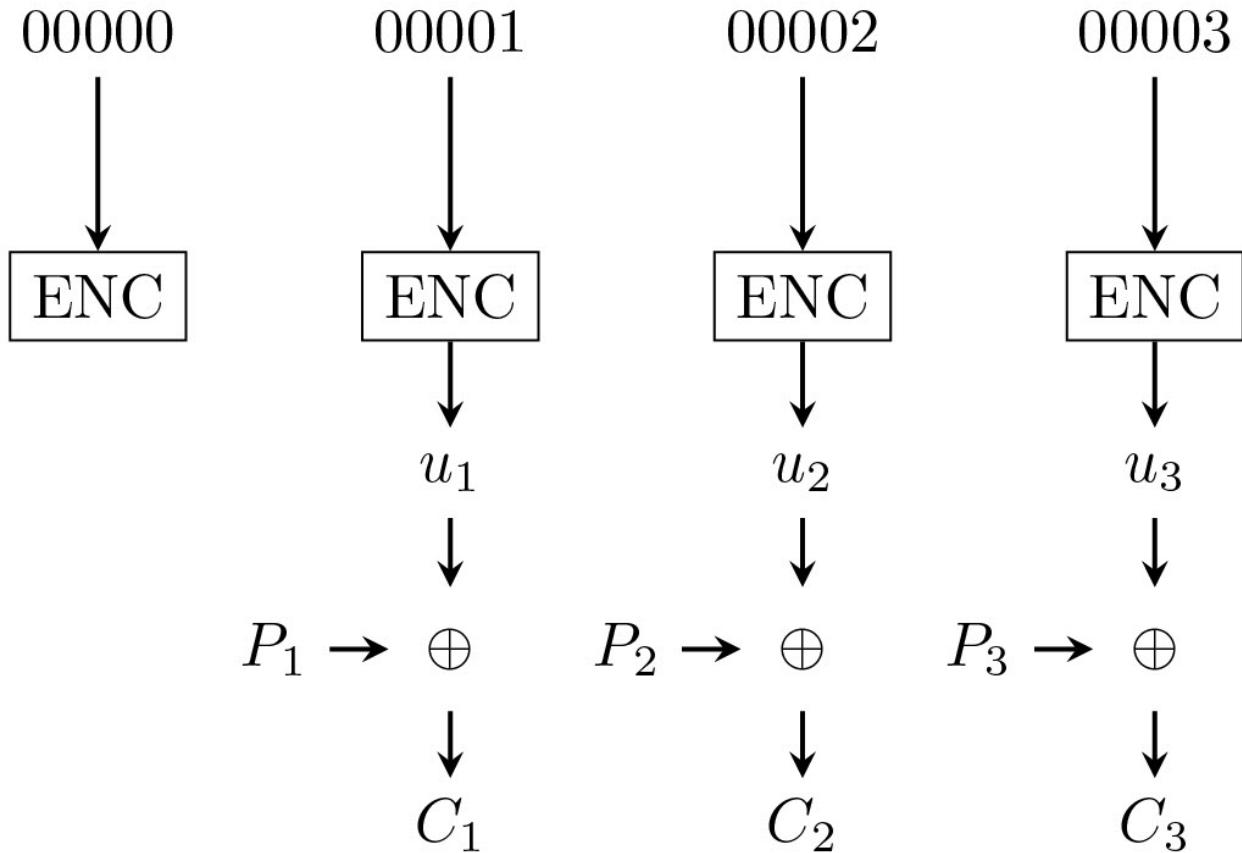
Description: I'm expecting a super secret message... If you can give it to me, I'll give you a flag!

Bài này thực chất là tính tag trong mode GCM của AES. Các bạn có thể xem thêm về mode CTR và GCM trong bài tìm hiểu lại Google CTF 2020 của mình (bài Pythia).

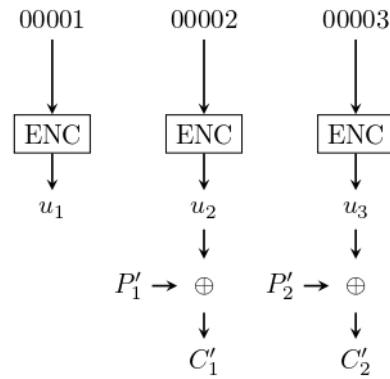
Trong bài này chúng ta có ba lần request với ba nonce khác nhau để encrypt hoặc decrypt.

Xét hàm **incr** mình thấy counter thực hiện cộng lên 1 ở mỗi bước.

Nếu mình bắt đầu từ nonce = 0, với ba block plaintext P_1, P_2, P_3 thì ciphertext tương ứng sẽ như sau.



Tại sao mình cần ba block plaintext trong khi `message` của mình là hai block (32 byte)? Vì ý tưởng của mình khi decrypt là gửi nonce = 1 lên, khi đó các giá trị trung gian (AES của counter) giữ nguyên với lúc encrypt.



Lưu ý rằng ở hình 1, mình encrypt với $P = 0$ và nonce = 0 nên mình sẽ có $C_i = u_i$. Ở hình 2 mình có $P'_i \oplus C'_i = P_{i+1} \oplus C_{i+1} = C_{i+1}$, nên mình sẽ tìm được C'_i ứng với P'_i là message ($C'_i = P'_i \oplus C_{i+1}$).

Vậy còn bước authentication?

Nhắc lại về cách tính tag. Trong $\text{GF}(2^{128})$, giả sử với các ciphertext C_1, C_2, \dots, C_n , với $H = \text{AES}_K(0^{128})$, $L = \text{len}(A) \parallel \text{len}(C)$ và $I = \text{AES}_K(J_0)$ với J_0 là nơi bắt đầu counter.

Ở đây $n = 2$, nghĩa là mình cần tìm $T = C_1 H^4 + C_2 H^3 + AH^2 + LH + I$ với A là associated data (ở trong bài là header).

Ở đây chúng ta đã có C'_1, C'_2, A, L, I (I chính là C_1 , các bạn thấy không? :v), vậy còn H ?

Tới đây mình encrypt với nonce = 2 và ciphertext là chuỗi rỗng. Khi đó tag của mình là $T'' = AH^2 + L''H + I$, nhưng A đã bị pad bởi mấy số 0 ở cuối nên phép nhân H_mult luôn cho ra 0, do đó $T'' = L''H + I$ (L' là vì lúc này $\text{len}(C) = 0$). Như vậy mình có thể tìm H rồi.

Như vậy quy trình giải bài này là:

- encrypt với nonce = 0 và $P_i = 0^{128}$ với $i = 1, 2, 3$;
- encrypt với nonce = 2 và $P_i = \emptyset$;
- tính H từ đó tính tag khi nonce = 1;
- decrypt với $C = \text{message}$ và nonce = 1.

Cám ơn các bạn đã đọc writeup của mình.

4.6 Các cuộc thi năm 2024

4.6.1 NSUCRYPTO 2024

Trong hai bài viết mình sẽ giới thiệu cách giải các bài trong NSUCRYPTO 2024 từ bài giải của team mình lẫn từ team các bạn mà mình tham khảo sau khi kết thúc giải. Hiện tại chỉ có team của Robin Jadoul, Jack Pope, Esrever Yievs chia sẻ bài giải gần hoàn chỉnh trên discord của một cộng đồng toxic nào đó mà ai cũng biết nên mình sẽ gọi là team JPY khi đề cập tới cách giải của các bạn (lấy chữ cái đầu họ của ba người :v).

Lời nói đầu

Đầu tiên mình xin cảm ơn rất nhiều người đã ủng hộ, giúp đỡ mình trong mấy năm qua, để mình có thể đi tới ngày hôm nay.

Đề thi năm nay khiến mình có chút hoang mang khi bắt đầu làm. Round 2 có 11 câu với hai câu special prize. Thường những năm trước có từ 4 tới 5 câu special prize và đây là nơi đột biến xảy ra. Nếu chỉ có hai câu thì mình không hy vọng được giải cao mà mục tiêu là giải trọn vẹn và không mắc lỗi ở tất cả câu khác.

Và điều kì diệu đã xảy ra :))) team mình được hai bài BEST SOLUTION do hai bạn teammate giải, còn mình thì ... ké :D

Chân thành cảm ơn bạn Chương và bạn Uyên đã đồng hành cùng mình hai năm nay.

RSA signature

Đây là bài 7 ở round 1 và bài 1 ở round 2.

Đề bài

Chúng ta cần ký thông điệp M bằng thuật toán RSA. Như bình thường, đặt $N = p \cdot q$ là RSA-modulus với p và q là hai số nguyên tố lớn.

Đặt e là public exponent và d là secret exponent với

$$e \cdot d = 1 \pmod{(p-1)(q-1)}.$$

Signature của chúng ta sẽ là

$$S = M^d \pmod{N}.$$

Giả sử kẻ tấn công biết giá trị

$$M_p = M^{d_p} \pmod{p},$$

nhưng không biết giá trị

$$M_q = M^{d_q} \pmod{q},$$

trong đó

$$d_p = d \pmod{p-1}, \quad d_q = d \pmod{q-1}.$$

Nếu kẻ tấn công biết modulo N (nhưng không biết p và q), biết public exponent e (nhưng không biết d) và biết thông điệp gốc M . Các thông số bí mật nào kẻ tấn công có thể tìm ra?

Lời giải

Vì $S \equiv M^d \pmod{N}$ nên suy ra $S \equiv M^d \pmod{p}$ vì

$$(S - M^d) \vdots N \vdots p,$$

tương tự $S \equiv M^d \pmod{q}$.

Ngoài ra ta có

$$S^e \equiv M^{ed} \equiv M \pmod{N}. \quad (4.1)$$

Tương tự phía trên ta cũng có

$$S^e \equiv M^{ed} \equiv M \pmod{p},$$

nhưng $d_p \equiv d \pmod{p-1}$, tương đương với

$$d = d_p + k_p \cdot (p-1)$$

với $k_p \in \mathbb{Z}$.

Do đó

$$\begin{aligned} S^e &\equiv M^{e \cdot (d_p + k_p \cdot (p-1))} \pmod{p} \\ &\equiv M^{e \cdot d_p} \pmod{p} \quad (\text{định lý Euler}) \\ &\equiv M_p^e \pmod{p}. \end{aligned} \quad (4.2)$$

Từ hai phương trình (4.1) và (4.2) suy ra

$$M_p^e \equiv M \pmod{p},$$

hay ta có thể nói là

$$(M_p^e \pmod{N}) - M \vdots p.$$

Từ đây, sử dụng thuật toán Euclid ta tìm được

$$p = \gcd((M_p^e \pmod{N}) - M, N).$$

Vậy là ta đã tìm được p nên có thể tìm được tất cả tham số secret khác của RSA là q, d, d_p và d_q .

solve.py

```
from Crypto.Util.number import getPrime
import random

def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

p, q = [getPrime(1024) for _ in range(2)]

print(f"p = {p}")
print(f"q = {q}")

e = 65537
N = p * q
d = pow(e, -1, (p - 1) * (q - 1))

M = random.randint(1, N - 1)

dp = d % (p - 1)
Mp = pow(M, dp, p)

print(f"N = {N}")
print(f"e = {e}")
print(f"M = {M}")
print(f"Mp = {Mp}")

p_guess = gcd(pow(Mp, e, N) - M, N)
assert p == p_guess
print("Successfully recovered p")
```

AntCipher 2.0

Đây là bài 2 ở round 2.

Đề bài

Một phút một lần, GPS được track bởi vệ tinh. Khi đó, vĩ độ ở dạng IEEE 754 single-precision floating-point sẽ được chuyển thành dãy 32 bit. Tương tự cho kinh độ.

Khi đó hai dãy nhị phân sẽ được nối lại (lattile || longitude) thành plaintext có 64 bit. Sau đó plaintext được XOR với keystream để tạo thành ciphertext 64 bit.

Cách thực hiện sinh khóa như sau. Tại điểm khởi tạo, secret key 64 bit được viết vào thanh ghi R dung lượng 64 bit.

Với mỗi số i với $i \geq 1$, dãy K_i 64 bit sẽ được sinh từ input là R và keystream cũng dùng để cập nhật thanh ghi R , nghĩa là sau khi tính toán thì giá trị K_i sẽ được ghi vào R .

Xét dạng CNF của hàm C , với CNF là conjunction of disjunction (tích của các tổng), có dạng sau:

$$\begin{aligned} C = & (x_1 \vee v_2 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_5) \wedge (x_1 \vee x_3 \vee \neg x_5) \wedge (\neg x_1 \vee \neg x_3 \vee x_5) \\ & \wedge (x_2 \vee x_3 \vee \neg x_5) \wedge (\neg x_2 \vee \neg x_3 \vee x_5) \wedge (x_1 \vee x_2 \vee \neg x_6) \wedge (\neg x_1 \vee \neg x_2 \vee x_6) \\ & \wedge (x_1 \vee x_4 \vee \neg x_6) \wedge (\neg x_1 \vee \neg x_4 \vee x_6) \wedge (x_2 \vee x_4 \vee \neg x_6) \wedge (\neg x_2 \vee \neg x_4 \vee x_6) \\ & \wedge (x_1 \vee x_3 \vee \neg x_7) \wedge (\neg x_1 \vee \neg x_3 \vee x_7) \wedge (x_1 \vee x_4 \vee \neg x_7) \wedge (\neg x_1 \vee \neg x_4 \vee x_7) \\ & \wedge (x_3 \vee x_4 \vee \neg x_7) \wedge (\neg x_3 \vee \neg x_4 \vee x_7) \wedge (x_2 \vee x_3 \vee \neg x_8) \wedge (\neg x_2 \vee \neg x_3 \vee x_8) \\ & \wedge (x_2 \vee x_4 \vee \neg x_8) \wedge (\neg x_2 \vee \neg x_4 \vee x_8) \wedge (x_3 \vee x_4 \vee \neg x_8) \wedge (\neg x_3 \vee \neg x_4 \vee x_8) \end{aligned}$$

Phương trình $C = 1$ biểu diễn một hàm không tuyến tính F_G lấy đầu vào là 4 bits (x_1, x_2, x_3, x_4) và sinh ra 4 bits (x_5, x_6, x_7, x_8) làm output.

Ở vòng lặp thứ i của cipher, 64 bits của R được chia thành 16 đoạn, mỗi đoạn 4 bit. Mỗi đoạn thành input cho F_G và kết quả là 16 đoạn 4 bits mới được nối với nhau thành K_i có 64 bits.

1704 ciphertexts được truyền đi không gắp vấn đề. Tuy nhiên ở hai keystream tiếp theo gắp sự cố về đĩa cứng nên chỉ nhận được ciphertext thứ 1704 là dãy

```
1001 1000 0011 1101 0110 0011 1101 0101 1011 0011 1011 0111 0000 0000 1000 0011
```

và hai keystrem 1702 và 1703 nhưng bị mất một phần:

$$\begin{aligned} K_{1702} = & 0101\ 1001\ 1111\ 0011\ 00X1\ X111\ 1X00\ 00X0 \\ & 111X\ X000\ XXXX\ XXXX\ XXXX\ XXXX\ XXXX\ XXXX, \\ K_{1703} = & XXXX\ XXXX\ XXXX\ XXXX\ XXXX\ XXXX\ XXXX\ XXXX \\ & X111\ 000X\ X010\ 01X1\ 0X10\ 0101\ 0000\ 1111. \end{aligned}$$

Hãy tìm lại plaintext thứ 1704.

Lời giải

Đầu tiên mình cần tìm những bộ số (x_1, \dots, x_8) mà $C = 1$.

```
from itertools import product

def f(x1, x2, x3, x4, x5, x6, x7, x8):
    result = 1
    result = result & (x1 | x2 | ~x5)
    result = result & (~x1 | ~x2 | x5)
    result = result & (x1 | x3 | ~x5)
    result = result & (~x1 | ~x3 | x5)
    result = result & (x2 | x3 | ~x5)
    result = result & (~x2 | ~x3 | x5)
    result = result & (x1 | x2 | ~x6)
    result = result & (~x1 | ~x2 | x6)
    result = result & (x1 | x4 | ~x6)
    result = result & (~x1 | ~x4 | x6)
    result = result & (x2 | x4 | ~x6)
    result = result & (~x2 | ~x4 | x6)
    result = result & (x1 | x3 | ~x7)
    result = result & (~x1 | ~x3 | x7)
    result = result & (x1 | x4 | ~x7)
    result = result & (~x1 | ~x4 | x7)
```

(continues on next page)

(continued from previous page)

```

result = result & (x3 | x4 | ~x7)
result = result & (~x3 | ~x4 | x7)
result = result & (x2 | x3 | ~x8)
result = result & (~x2 | ~x3 | x8)
result = result & (x2 | x4 | ~x8)
result = result & (~x2 | ~x4 | x8)
result = result & (x3 | x4 | ~x8)
result = result & (~x3 | ~x4 | x8)

return result

key = { }

for x1, x2, x3, x4, x5, x6, x7, x8 in product(range(2), repeat=8):
    t = f(x1, x2, x3, x4, x5, x6, x7, x8)
    if t == 1:
        print(f"[x1, x2, x3, x4] -> [x5, x6, x7, x8]")
        a = x1 + (x2 << 1) + (x3 << 2) + (x4 << 3)
        b = x5 + (x6 << 1) + (x7 << 2) + (x8 << 3)
        key[a] = b

```

Bảng biến đổi sẽ như sau:

(x_1, x_2, x_3, x_4)	\rightarrow	(x_5, x_6, x_7, x_8)
(0, 0, 0, 0)	\rightarrow	(0, 0, 0, 0)
(0, 0, 0, 1)	\rightarrow	(0, 0, 0, 0)
(0, 0, 1, 0)	\rightarrow	(0, 0, 0, 0)
(0, 0, 1, 1)	\rightarrow	(0, 0, 1, 1)
(0, 1, 0, 0)	\rightarrow	(0, 0, 0, 0)
(0, 1, 0, 1)	\rightarrow	(0, 1, 0, 1)
(0, 1, 1, 0)	\rightarrow	(1, 0, 0, 1)
(0, 1, 1, 1)	\rightarrow	(1, 1, 1, 1)
(1, 0, 0, 0)	\rightarrow	(0, 0, 0, 0)
(1, 0, 0, 1)	\rightarrow	(0, 1, 1, 0)
(1, 0, 1, 0)	\rightarrow	(1, 0, 1, 0)
(1, 0, 1, 1)	\rightarrow	(1, 1, 1, 1)
(1, 1, 0, 0)	\rightarrow	(1, 1, 0, 0)
(1, 1, 0, 1)	\rightarrow	(1, 1, 1, 1)
(1, 1, 1, 0)	\rightarrow	(1, 1, 1, 1)
(1, 1, 1, 1)	\rightarrow	(1, 1, 1, 1)

Có thể thấy đầu ra không chứa đủ 16 trường hợp mà chỉ gồm các trường hợp là

$(0, 0, 0, 0), (0, 0, 1, 1), (0, 1, 0, 1), (1, 0, 0, 1), (0, 1, 1, 0), (1, 0, 1, 0), (1, 1, 0, 0), (1, 1, 1, 1).$

Ở đây mỗi vector đều có số lượng chẵn phần tử 1.

Xét biến đổi từ K_{1702} thành K_{1703} như sau:

K_{1702}	\rightarrow	K_{1703}
0101	\rightarrow	XXXX
1001	\rightarrow	XXXX
1111	\rightarrow	XXXX
0011	\rightarrow	XXXX
00X1	\rightarrow	XXXX
X111	\rightarrow	XXXX
1X00	\rightarrow	XXXX
00X0	\rightarrow	XXXX
111X	\rightarrow	X111
X000	\rightarrow	000X
XXXX	\rightarrow	X010
XXXX	\rightarrow	01X1
XXXX	\rightarrow	0X10
XXXX	\rightarrow	0101
XXXX	\rightarrow	0000
XXXX	\rightarrow	1111

Dựa vào bảng biến đổi ở trên và lưu ý đầu ra (có 8 trường hợp tất cả) thì ta có kết luận cho từng bộ 4 như sau:

1. 0101 \rightarrow 010.
2. 1001 \rightarrow 0110.
3. 1111 \rightarrow 1111.
4. 0011 \rightarrow 0011.
5. 00X1 \rightarrow 0000. Vì K_{1702} nhận được từ K_{1701} nên X chỉ có thể là 1 và kết quả ở K_{1703} sẽ là 0000.
6. X111 \rightarrow 1111.
7. 1X00 \rightarrow 0000. Tương tự, vì K_{1702} nhận được từ K_{1701} nên X chỉ có thể là 1 và kết quả ở K_{1703} là 0000.
8. 0X00 \rightarrow 0000.
9. 111X \rightarrow 1111.
10. X000 \rightarrow 0000.
11. XXXX \rightarrow 1010. Trong mỗi bộ số trong 8 trường hợp ở trên thì số lượng chữ số 1 là chẵn.
12. XXXX \rightarrow 0101. Tương tự.
13. XXXX \rightarrow 0110. Tương tự
14. XXXX \rightarrow 0101.
15. XXXX \rightarrow 0000.
16. XXXX \rightarrow 1111.

Như vậy mình đã khôi phục được K_{1703} và có thể decrypt.

```
# https://gist.github.com/AlexEshoo/d3edc53129ed010b0a5b693b88c7e0b5
def ieee_754_conversion(n, sgn_len=1, exp_len=8, mant_len=23):
    """
    Converts an arbitrary precision Floating Point number.
    """

```

(continues on next page)

(continued from previous page)

Note: Since the calculations made by python inherently use floats, the accuracy is poor at high precision.

```

:param n: An unsigned integer of length `sgn_len` + `exp_len` + `mant_len` to be decoded as a float
:param sgn_len: number of sign bits
:param exp_len: number of exponent bits
:param mant_len: number of mantissa bits
:return: IEEE 754 Floating Point representation of the number `n`
"""

if n >= 2 ** (sgn_len + exp_len + mant_len):
    raise ValueError("Number n is longer than prescribed parameters allows")

sign = (n & (2 ** sgn_len - 1) * (2 ** (exp_len + mant_len))) >> (exp_len + mant_len)
exponent_raw = (n & ((2 ** exp_len - 1) * (2 ** mant_len))) >> mant_len
mantissa = n & (2 ** mant_len - 1)

sign_mult = 1
if sign == 1:
    sign_mult = -1

if exponent_raw == 2 ** exp_len - 1: # Could be Inf or NaN
    if mantissa == 2 ** mant_len - 1:
        return float('nan') # NaN

    return sign_mult * float('inf') # Inf

exponent = exponent_raw - (2 ** (exp_len - 1) - 1)

if exponent_raw == 0:
    mant_mult = 0 # Gradual Underflow
else:
    mant_mult = 1

for b in range(mant_len - 1, -1, -1):
    if mantissa & (2 ** b):
        mant_mult += 1 / (2 ** (mant_len - b))

return sign_mult * (2 ** exponent) * mant_mult

ciphertext = "1001 1000 0011 1101 0110 0011 1101 0101 1011 0011 1011 0111 0000 0000 1000
→0011"
ciphertext = list(map(int, "".join(ciphertext.split(" ")).split(" ")))

keystream = "0101 0110 1111 0011 0011 1111 1100 0000 1111 0000 1010 0101 0110 0101 0000
→1111"
keystream = [int(i, 2) for i in keystream.split(" ")]
keystream = [key[k] for k in keystream] # Find K_{1704} from K_{1703} by table 1
keystream = "".join(f"[k:04b]" for k in keystream)
keystream = list(map(int, keystream))

plaintext = [c^k for c, k in zip(ciphertext, keystream)] # Recover plaintext
plaintext = "".join(map(str, plaintext))

```

(continues on next page)

(continued from previous page)

```

assert len(plaintext) == 64
m, n = int(plaintext[:32], 2), int(plaintext[32:], 2)      # Extract first 32 bits and
                                                               ↵ last 32 bits

latitude = ieee_754_conversion(m)    # Convert first 32 bits to latitude
longitude = ieee_754_conversion(n)   # Convert last 32 bits to longitude

print(f"({latitude}, {longitude})")

```

Mình nhận được tọa độ là

$$(-25.79496192932129, 146.58416748046875).$$

Tọa độ này chỉ tới Australia (Augathella), cụ thể là một công viên tên là "Meat Ant Park". Chữ "Ant" có vẻ khớp với đề bài.

Steganography and codes

Đây là bài 3 ở round 2.

Đề bài

Sam và Betty sử dụng kênh mở cho giao tiếp bí mật. Họ không muốn ai biết về hội thoại của mình.

Sam có thể gửi cho Betty một trong 16 thông điệp.

Khi đó, Sam lấy một ảnh với format RGB bất kì, đổi pixel đầu tiên theo cách nào đó và publish lên website.

Betty sẽ tải bức ảnh đó xuống và phân tích xem Sam đã gửi thông điệp nào.

Sam sẽ làm gì với bức ảnh? Chúng ta nên chú ý rằng phải thay đổi pixel sao cho không thể nhận biết sự thay đổi bởi mắt thường.

Cụ thể, mỗi pixel RGB được biểu diễn ở dạng 24 bit:

- 8 bit đầu là độ sáng của màu đỏ (r_1, \dots, r_8);
- 8 bit tiếp theo là độ sáng màu xanh lá (g_1, \dots, g_8);
- 8 bit cuối là độ sáng màu xanh dương (b_1, \dots, b_8).

Sam không được thay đổi các bit r, g, b từ 1 tới 5 vì điều đó sẽ dễ nhận thấy khi nhìn bằng mắt.

Nếu Sam đổi một trong các bit r_6, r_7, g_6, g_7 và b_6 sẽ tốn 2 coins mỗi lần đổi.

Nếu Sam đổi một trong các bit r_8, g_8, b_7 và b_8 thì tốn 1 coin mỗi lần đổi.

Hãy đưa ra một phương pháp coding cho 16 thông điệp vào một pixel mà tốn không quá 2 coins (mà không thể nhận thấy bằng mắt thường).

Thêm nữa hãy đưa ra một phương pháp để Betty có thể biết được thông điệp mà Sam giấu vào ảnh là gì. Lưu ý rằng Betty không biết ảnh gốc như thế nào.

Lời giải

Tạm thời trống.

Weak key schedule for DES

Đây là bài 6 ở round 1 và bài 4 ở round 2.

Đề bài

Alice sử dụng thuật toán DES để mã hóa file *Book.txt* thành *Book_Cipher.txt*. Tuy nhiên khi code thuật toán DES thì Alice đã code lỗi và khiến cả thuật toán chỉ sử dụng subkey đầu tiên cho tất cả các vòng (thay vì mỗi vòng mỗi khóa con như DES chuẩn).

Hãy giúp Carol tìm lại khóa ban đầu và decrypt bản mã sau:

```
86991641D28259604412D6BA88A5C0A6471CA722
2C52482BF2D0E841D4343DFB877DC8E0147F3D5F
20FC18FF28CB5C4DA8A0F4694861AB5E98F37ADB
C2D69B35779D9001BB4B648518FE6EBC00B2AB10
```

Lời giải

Bài này được giải bởi bạn Chương và được thêm BEST SOLUTION :)))

Ý tưởng của bài này là slide attack. Các bạn có thể xem lời giải ở bài viết về [slide attack](#) của mình.

Reverse engineering

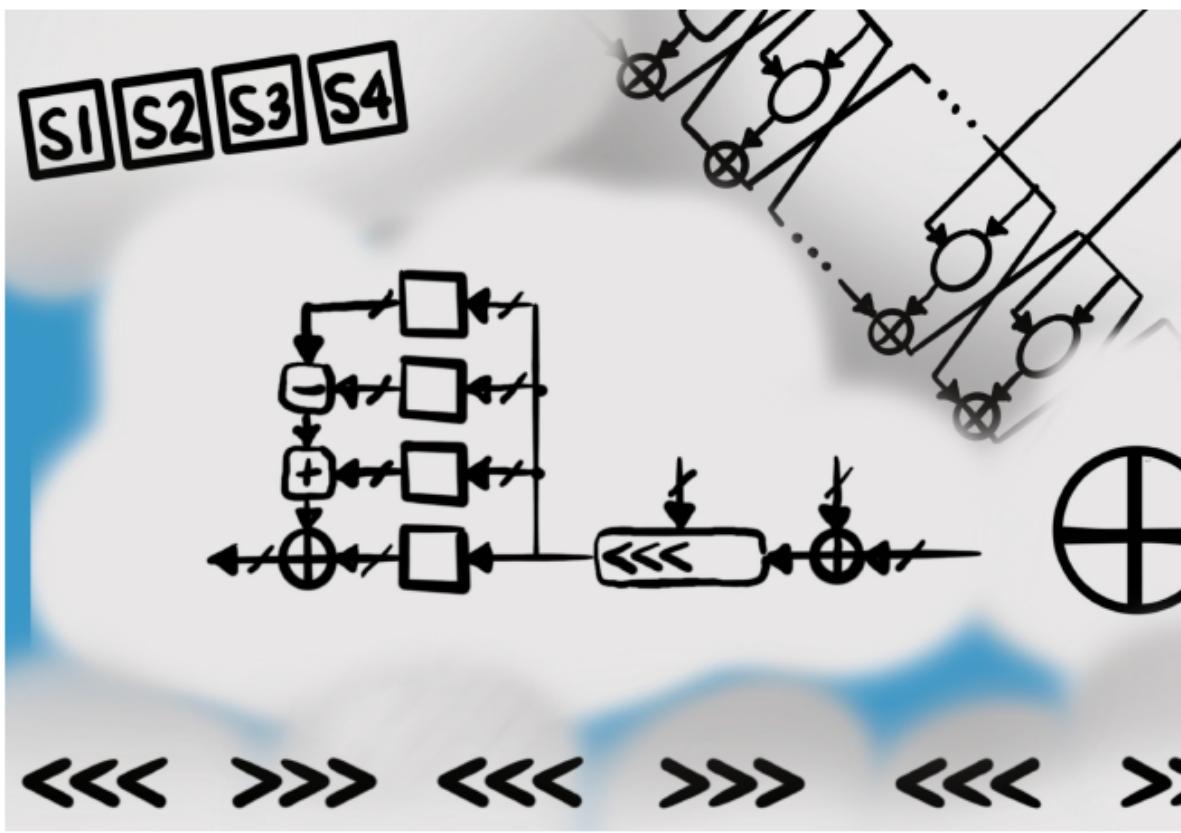
Đây là bài 5 ở round 2.

Đề bài

Sau khi reverse engineering cài đặt của một thuật toán mật mã nào đó chưa biết thì Bob nhận được hàm boolean sau:

$$f_{2n}(x_1, \dots, x_{2n}) = \bigoplus_{i=1}^n x_i x_{i+n} \prod_{j=i+1}^n (x_j \oplus x_{j+n}).$$

Bob cố gắng hiểu ý nghĩa mật mã của hàm boolean trên là gì, và nhanh chóng tìm thấy một "liên hệ". Đây là hàm gì?



Mở đầu

Bài này mình chứng minh được tính chất của hàm đề bài cho chữ không tìm ra ý nghĩa của hàm nên chỉ được 4/6 điểm.

Tham khảo lời giải của team JPY thì đây là hàm kiểm tra tràn bit số nguyên (integer overflow).

Mình sẽ trình bày cả cách giải của mình lẫn cách giải của team bạn.

Hình đề cho là CAST cipher - tiêu chuẩn mã hóa của Canada.

S-boxes trong CAST cipher được lựa chọn từ các hàm boolean có nonlinearity cao nhất nên trong lúc làm bài mình cắm đầu vào tìm một tính chất gì đó của hàm boolean trên liên quan tới nonlinearity.

Mình đã chứng minh được hai điều:

- trọng số Hamming của hàm f_{2n} bằng $2^{n-1} \cdot (2^n - 1)$;
- nonlinearity của hàm f_{2n} bằng 2^{2n-2} .

Nhắc lại hàm boolean f_{2n} đề cho là hàm

$$f_{2n}(x_1, \dots, x_{2n}) = \bigoplus_{i=1}^n x_i x_{i+n} \prod_{j=i+1}^n (x_j \oplus x_{j+n}).$$

Trọng số Hamming của hàm boolean

Để tìm công thức hoặc tính chất thì thường mình sẽ thử với các số nhỏ và sau đó đoán công thức.

Với $n = 1$ thì $f_2(x_1, x_2) = x_1 x_2$. Để thấy rằng $f_2 = 1$ khi $x_1 = x_2 = 1$.

Thay vì viết bảng chân trị gồm 4 dòng thì mình viết thành bảng ô vuông 2×2 :

	$x_2 = 0$	$x_2 = 1$
$x_1 = 0$	$f_2 = 0$	$f_2 = 0$
$x_1 = 1$	$f_2 = 0$	$f_2 = 1$

Với $n = 2$ thì hàm boolean là

$$f_4(x_1, x_2, x_3, x_4) = [x_1 x_3 (x_2 \oplus x_4)] \oplus x_2 x_4 = x_1 x_2 x_3 \oplus x_1 x_3 x_4 \oplus x_2 x_4.$$

Cũng tương tự, mình viết bảng chân trị thành bảng ô vuông 4×4 :

	$x_3 x_4 = 00$	$x_3 x_4 = 01$	$x_3 x_4 = 10$	$x_3 x_4 = 11$
$x_1 x_2 = 00$	$f_4 = 0$	$f_4 = 0$	$f_4 = 0$	$f_4 = 0$
$x_1 x_2 = 01$	$f_4 = 0$	$f_4 = 0$	$f_4 = 0$	$f_4 = 1$
$x_1 x_2 = 10$	$f_4 = 0$	$f_4 = 0$	$f_4 = 1$	$f_4 = 1$
$x_1 x_2 = 11$	$f_4 = 0$	$f_4 = 1$	$f_4 = 1$	$f_4 = 1$

Với $n = 3$ thì mình viết bảng chân trị thành bảng 8×8 :

	000	001	010	011	100	101	110	111
000	0	0	0	0	0	0	0	0
001	0	0	0	0	0	0	0	1
010	0	0	0	0	0	0	1	1
011	0	0	0	0	0	1	1	1
100	0	0	0	0	1	1	1	1
101	0	0	0	1	1	1	1	1
110	0	0	1	1	1	1	1	1
111	0	1	1	1	1	1	1	1

Đến đây thì mình nhận thấy hàm f_{2n} sẽ nhận giá trị 1 tại:

- 1 ô hàng thứ hai;
- 2 ô hàng thứ ba;
- ...
- $2^n - 1$ ô hàng cuối.

Như vậy trọng số Hamming được dự đoán là

$$1 + 2 + \dots + (2^n - 1) = \frac{(2^n - 1) \cdot 2^n}{2} = 2^{n-1} \cdot (2^n - 1).$$

Sau đây mình sẽ chứng minh công thức trọng số Hamming này bằng quy nạp.

Với $n = 1$, hàm $f_2(x_1, x_2) = x_1 x_2$ có trọng số Hamming là $1 = 2^{1-1} \cdot (2^1 - 1)$. Như vậy công thức đúng với $n = 1$.

Giả sử công thức đúng với $n = k \geq 1$, nghĩa là hàm

$$f_{2k}(x_1, \dots, x_{2k}) = \bigoplus_{i=1}^k x_i x_{i+k} \prod_{j=i+1}^k (x_j \oplus x_{j+k})$$

có trọng số Hamming là $2^{k-1} \cdot (2^k - 1)$.

Với $n = k + 1$, xét hàm

$$f_{2k+2}(x_1, \dots, x_k, y, x_{k+1}, \dots, x_{2k}, z) = \bigoplus_{i=1}^k x_i x_{i+k} \left[\prod_{j=i+1}^k (x_j \oplus x_{j+k}) \cdot (y \oplus z) \right] \oplus yz.$$

Ta có các trường hợp sau:

- khi $y = z = 0$ thì $y \oplus z = 0$ và $yz = 0$. Khi đó mọi phép nhân $\prod_{j=i+1}^k (x_j \oplus x_{j+k}) \cdot (y \oplus z) = 0$ nên khiến cả hàm $f_{2k+2} = 0$ tại tất cả vector (x_1, \dots, x_{2k}) ;
- khi $y = 0$ và $z = 1$ thì $y \oplus z = 1$ và $yz = 0$. Thay vào hàm f_{2k+2} ta có

$$f_{2k+2}(x_1, \dots, x_k, 0, x_{k+1}, \dots, x_{2k}, 1) = \bigoplus_{i=1}^k x_i x_{i+k} \left[\prod_{j=i+1}^k (x_j \oplus x_{j+k}) \cdot 1 \right] \oplus 0 \equiv f_{2k}(x_1, \dots, x_{2k}).$$

Như vậy khi $y = 0$ và $z = 1$ thì f_{2k+2} chính xác bằng f_{2k} . Nghĩa là trọng số Hamming của f_{2k+2} lúc này bằng trọng số Hamming của f_{2k} và bằng $2^{k-1} \cdot (2^k - 1)$ theo giả thiết quy nạp.

- khi $y = 1$ và $z = 0$ thì tương tự, trọng số Hamming của f_{2k+2} trong trường hợp này cũng bằng $2^{k-1} \cdot (2^k - 1)$.
- khi $y = z = 1$ thì $y \oplus z = 0$ và $yz = 1$. Khi đó tất cả biểu thức $\prod_{j=i+1}^k (x_j \oplus x_{j+k}) \cdot (y \oplus z) = 0$ nên hàm $f_{2k+2} = 1$ với mọi đầu vào $(x_1, \dots, x_{2k}) \in \mathbb{F}_2^{2k}$. Như vậy trọng số Hamming của f_{2k+2} ở trường hợp này là 2^{2k} .

Kết hợp bốn trường hợp lại thì trọng số Hamming của f_{2k+2} là

$$\text{wt}(f_{2k}) = 2^{k-1} \cdot (2^k - 1) \cdot 2 + 2^{2k} = 2^k(2^{k+1} - 1).$$

Như vậy công thức trọng số Hamming đúng với $k + 1$ nên theo quy nạp thì công thức đúng với mọi $n \geq 1$.

Biến đổi Walsh-Hadamard và nonlinearity

Gọi $W_f(\mathbf{a})$ là hệ số Walsh của hàm boolean f (trên $2n$ biến) ứng với vector $\mathbf{a} \in \mathbb{F}_2^{2n}$.

Đặt $\mathbf{0}$ là vector độ dài $2n$ chỉ gồm các số 0, nghĩa là $\mathbf{0} = (0, \dots, 0)$.

Một tính chất đơn giản của phổ Walsh là

$$W_f(\mathbf{0}) = 2^{2n} - 2 \cdot \text{wt}(f),$$

với f là hàm boolean $2n$ biến.

Từ kết quả về trọng số ở trên, thay vào công thức ta có

$$W_f(\mathbf{0}) = 2^{2n} - 2 \cdot 2^{n-1} \cdot (2^n - 1) = 2^n.$$

Bây giờ xét hệ số Walsh cho vector dạng

$$\mathbf{a} = (0, 0, \dots, a_n, 0, 0, \dots, a_{2n}).$$

Công thức tính hệ số Walsh là

$$W_f(\mathbf{a}) = \sum_{\mathbf{x} \in \mathbb{F}_2^{2n}} (-1)^{f(\mathbf{x}) \oplus x_n a_n \oplus x_{2n} a_{2n}}.$$

Nếu $a_n = a_{2n} = 0$ thì ta có trường hợp $W_f(\mathbf{0})$ như vừa rồi.

Nếu $a_n = 1$ và $a_{2n} = 0$ ta sẽ chứng minh hệ số Walsh lúc này có giá trị tuyệt đối lớn nhất. Tương tự đối với $a_n = 0$ và $a_{2n} = 1$ cũng sẽ cho hệ số Walsh có giá trị tuyệt đối lớn nhất.

Khi $a_n = 1$ và $a_{2n} = 0$ thì hệ số Walsh sẽ có dạng

$$W_f(\mathbf{a}) = \sum_{\mathbf{x} \in \mathbb{F}_2^{2n}} (-1)^{f(\mathbf{x}) \oplus x_n},$$

ở đây $\mathbf{x} = (x_1, \dots, x_{2n})$.

Ta có hai trường hợp:

Trường hợp 1. Nếu $x_n = 0$ thì trong số 2^{2n-1} vector $(x_1, \dots, x_{n-1}, 0, x_{n+1}, \dots, x_{2n})$, ta cần tìm số lượng vector khiến $f_{2n} = 1$.

Theo chứng minh về trọng số Hamming ở trên thì nếu $x_{2n} = 0$ thì hàm f_{2n} (hiện là hàm $2n - 2$ biến do x_n và x_{2n} đã cố định) sẽ luôn có giá trị 0. Tuy nhiên chỉ với dữ liệu này thì không đủ để biết có bao nhiêu vector khiến $f_{2n} = 1$.

Thay đổi cách tiếp cận nhưng xét $x_{2n} = 1$, khi đó hàm f_{2n} cũng là hàm $2n - 2$ biến do x_n và x_{2n} đã cố định, nhưng hàm f_{2n} sẽ bằng 1 tại đúng $2^{n-2} \cdot (2^{n-1} - 1)$ vectors trong số 2^{2n-1} vectors đang xét. Vậy theo nguyên lý bù trừ thì $f_{2n} = 0$ tại

$$2^{2n-1} - 2^{n-2} \cdot (2^{n-1} - 1) = 3 \cdot 2^{2n-3} + 2^{n-2} \text{ vectors.}$$

Lúc này ta có thể tính

$$\sum_{\substack{\mathbf{x} \in \mathbb{F}_2^{2n} \\ x_n=0}} (-1)^{f(\mathbf{x}) \oplus 0} = (-1)^0 \cdot [2^{n-2} \cdot (2^{n-1} - 1)] + (-1)^1 \cdot (3 \cdot 2^{2n-3} + 2^{n-2}) = 2^{2n-2} + 2^{n-1}. \quad (4.3)$$

Trường hợp 2. Nếu $x_n = 1$, tương tự, trong số 2^{2n-1} vector $(x_1, \dots, x_{n-1}, 1, x_{n+1}, \dots, x_{2n})$ ta cần tìm xem có bao nhiêu vector khiến $f_{2n} = 1$.

Sử dụng chứng minh cho trọng số Hamming ở trên thì

1. Nếu $x_{2n} = 0$ thì

$$f_{2n}(x_1, \dots, x_{n-1}, 1, x_{n+1}, \dots, 0)$$

bằng 1 tại $2^{n-2} \cdot (2^{n-1} - 1)$ vectors.

2. Nếu $x_{2n} = 1$ thì

$$f_{2n}(x_1, \dots, x_{n-1}, 1, x_{n+1}, \dots, 1)$$

bằng 1 tại tất cả 2^{2n-2} vectors.

Tổng kết lại, hàm $f_{2n} = 1$ tại

$$A = 2^{n-2} \cdot (2^{n-1} - 1) + 2^{2n-2}$$

vectors. Điều này dẫn tới $f_{2n} = 0$ tại $2^{2n-1} - A$ vectors.

Ở đây do $x_n = 1$ nên khi tính $\sum(-1)^{f(\mathbf{x}) \oplus 1}$ ta cần chuyển dấu

$$\begin{aligned} \sum_{\substack{\mathbf{x} \in \mathbb{F}_2^{2n} \\ x_n=1}} (-1)^{f(\mathbf{x}) \oplus 1} &= (-1) \cdot (2^{2n-1} - A) + 1 \cdot A = 2A - 2^{2n-1} \\ &= 2^{2n-1} + 2^{n-1} \cdot (2^{n-1} - 1) - 2^{2n-1} = 2^{2n-2} - 2^{n-1}. \end{aligned} \quad (4.4)$$

Kết hợp (4.3) và (4.4) ta có hệ số Walsh ứng với vector \mathbf{a} có $a_n = 1$ là

$$W_f(\mathbf{a}) = 2^{2n-2} + 2^{n-1} + 2^{2n-2} - 2^{n-1} = 2^{2n-1}.$$

Tương tự, tại vector \mathbf{a}' có $a_{2n} = 1$ thì ta cũng có $W_f(\mathbf{a}') = 2^{2n-1}$.

Theo định lí Parseval thì

$$\sum_{\mathbf{a} \in \mathbb{F}_2^n} (W_f(\mathbf{a}))^2 = 2^{2 \cdot 2n} = 2^{4n}$$

vì f ở đây là hàm boolean $2n$ biến.

Như vậy ta chỉ cần chứng minh được rằng không tồn tại vector nào khác \mathbf{a} và \mathbf{a}' có hệ số Walsh lớn hơn hoặc bằng 2^{2n-1} .

Thật vậy, do tính giao hoán của phép XOR (phép cộng) và phép nhân nên công thức của f_{2n} cho thấy hai bộ (x_1, \dots, x_n) và (x_{n+1}, \dots, x_{2n}) có thể thay thế nhau, nghĩa là

$$f_{2n}(x_1, \dots, x_n, x_{n+1}, \dots, x_{2n}) \equiv f_{2n}(x_{n+1}, \dots, x_{2n}, x_1, \dots, x_n).$$

Do đó hệ số Walsh cũng có tính đối xứng như vậy. Nếu tồn tại vector \mathbf{b} có hệ số Walsh $W_f(\mathbf{b}) \geq 2^{2n-1}$ thì cũng tồn tại vector \mathbf{b}' khác \mathbf{b} có hệ số Walsh $W_f(\mathbf{b}') \geq 2^{2n-1}$.

Khi đó ta có

$$\underbrace{(W_f(\mathbf{a}))^2}_{=(2^{2n-1})^2} + \underbrace{(W_f(\mathbf{a}'))^2}_{=(2^{2n-1})^2} + \underbrace{(W_f(\mathbf{0}))^2}_{=(2^n)^2} + \underbrace{(W_f(\mathbf{b}))^2}_{>(2^{2n-1})^2} + \underbrace{(W_f(\mathbf{b}'))^2}_{>(2^{2n-1})^2} > 2^{4n} + 2^{2n}.$$

Điều này vô lý theo đẳng thức Parseval nên không tồn tại vector \mathbf{b} có hệ số Walsh lớn hơn hoặc bằng 2^{2n-1} .

Như vậy giá trị tuyệt đối lớn nhất trong các hệ số Walsh là 2^{2n-1} , từ đó nonlinearity là

$$N_f = 2^{2n-1} - \frac{1}{2} \max |W_f(\mathbf{a})| = 2^{2n-1} - \frac{1}{2} \cdot 2^{2n-1} = 2^{2n-1} - 2^{2n-2} = 2^{2n-2}.$$

Như vậy mình đã chứng minh xong công thức cho nonlinearity.

Lời giải của team JPY và gợi ý của thầy Kolomeec

Sau khi hết giải thì mình hỏi ý kiến thầy Kolomeec về cách làm của mình thì có vẻ là không khớp đáp án. Đáp án là nếu ta xem hai vector (x_1, \dots, x_n) và (x_{n+1}, \dots, x_{2n}) như các số nguyên, tức là

$$\begin{aligned} u &= x_1 + 2x_2 + \dots + 2^{n-1}x_n, \\ v &= x_{n+1} + 2x_{n+2} + \dots + 2^{n-1}x_{2n}, \end{aligned}$$

thì hàm boolean f_{2n} có giá trị 1 nếu $u + v \geq 2^n$ và ngược lại, đạt giá trị 0 nếu $u + v < 2^n$.

Gợi ý của thầy hoàn toàn khớp với bài giải của team JPY, trong đó nói rằng hàm f_{2n} là hàm kiểm tra tràn số khi cộng hai số nguyên (integer overflow). Ví dụ khi cộng hai số kiểu *unsigned int* thì có thể xảy ra hiện tượng tràn bit vì *unsigned int* chỉ có 32 bit. Điều này hợp lý vì CAST cipher có một phép cộng modulo 2^{32} , nghĩa là có liên quan đến hình vẽ.

Mình cũng sẽ dùng quy nạp để chứng minh khi $u + v \geq 2^n$ thì f_{2n} nhận giá trị 1.

Với $n = 1$, hàm $f_2(x_1, x_2) = x_1 x_2$ nhận giá trị 1 khi $x_1 = x_2 = 1$. Lúc này $u = x_1 = 1$ và $v = x_2 = 1$ nên $u + v = 2 \geq 2^1$. Như vậy mệnh đề đúng với $n = 1$.

Giả thiết quy nạp: giả sử mệnh đề đúng với $n = k \geq 1$, nghĩa là hàm boolean

$$f_{2k}(x_1, \dots, x_k, x_{k+1}, \dots, x_{2k}) = 1$$

khi $u + v \geq 2^k$ với

$$\begin{aligned} u &= x_1 + 2x_2 + \dots + 2^{k-1}x_k, \\ v &= x_{k+1} + 2x_{k+2} + \dots + 2^{k-1}x_{2k}. \end{aligned}$$

Với $n = k + 1$, xét hàm

$$f_{2k+2}(x_1, \dots, x_k, y, x_{k+1}, \dots, x_{2k}, z) = \bigoplus_{i=1}^k x_i x_{i+k} \left[\prod_{j=i+1}^k (x_j \oplus x_{j+k}) \cdot (y \oplus z) \right] \oplus yz.$$

Mình cũng sẽ có bốn trường hợp cho (y, z) :

1. Khi $y = z = 0$ thì $f_{2k+2} = 0$ với mọi (x_1, \dots, x_{2k}) nên chúng ta bỏ qua.
2. Khi $y = 1$ và $z = 0$ thì $f_{2k+2} \equiv f_{2k}$ như chứng minh ở phần trên. Khi đó f_{2k+2} bằng 1 tại các vector (x_1, \dots, x_{2k}) khiến f_{2k} bằng 1, nói cách khác là

$$\begin{aligned} u' &= x_1 + 2x_2 + \dots + 2^{k-1}x_k + 2^k y = u + 2^k y, \\ v' &= x_{k+1} + 2x_{k+2} + \dots + 2^{k-1}x_{2k} + 2^k z = v + 2^k z. \end{aligned}$$

Với $y = 1$ và $z = 0$ thì

$$u' + v' = u + 2^k y + v = (u + v) + 2^k y \geq 2^k + 2^k \cdot 1 = 2^{k+1}.$$

Như vậy nếu $y = 1$ và $z = 0$ thì mệnh đề cần chứng minh đúng cho $n = k + 1$.

1. Khi $y = 0$ và $z = 1$, chứng minh tương tự trường hợp $y = 1$ và $z = 0$.
2. Khi $y = 1$ và $z = 1$ thì f_{2k+2} luôn bằng 1, tương ứng với

$$u' + v' = u + 2^k y + v + 2^k z = (u + v) + 2^k \cdot (1 + 1) \geq 2^k + 2 \cdot 2^k > 2^{k+1}.$$

Như vậy mệnh đề đúng cho $n = k + 1$ với mọi trường hợp (y, z) và ta có điều cần chứng minh.

Open competition: NSUCRYPTO lightweight cipher

Đây là bài 6 ở round 2.

Đề bài

Ghi chú

Problem for special prize

NSUCRYPTO team tổ chức cuộc thi phát triển mã khối light-weight mới.

Một số yêu cầu:

- Block size là 64 bits.

- Key size là 80, 96 hoặc 64 bits.
- Số vòng là 32.
- Cấu trúc xây dựng tùy ý. Nghĩa là SPN, ARX, Feistel đều có thể sử dụng, thậm chí là một cách xây dựng mới.

Thí sinh cần xem xét PRESENT light-weight cipher (2007) và cố gắng tìm ra điều có thể làm tốt hơn mã khối này.

Hãy so sánh đáp án của bạn với PRESENT: khi thực hiện, khi chống phá mã (linear, differential, algebraic, ...) và đưa ra các giải thích cần thiết.

Lời giải

Đây là bài có tính mở rất cao nhưng để tìm ra điều gì đó mới (và hiệu quả) trong 7 ngày là rất khó. Do đó lúc thi mình đã xây dựng một block cipher mới tốt hơn PRESENT về mặt lưu trữ, còn các tính chất khác như kháng phá mã thì mình không đề cập tới. Bài này mình được 6/10 điểm.

Về PRESENT thì mình đã có tóm tắt trong một bài viết của blog này nên mình không viết lại. PRESENT cipher là thuật toán light-weight, tức là thuật toán được thiết kế nhỏ gọn để chạy trên các thiết bị có bộ xử lý và lưu trữ nhỏ (như các hệ thống nhúng).

Cải tiến của mình dựa trên việc PRESENT sử dụng mạng SP nên sẽ cần lưu trữ phép biến đổi để mã hóa và phép biến đổi ngược để giải mã:

- cần lưu trữ đồng thời S-box là $S[x]$ và inverse S-box là $S^{-1}[x]$;
- cần lưu trữ biến đổi tuyến tính pLayer và biến đổi ngược của nó;
- addRoundKey có phép biến đổi ngược là chính nó nên mình bỏ qua.

Như vậy, ta tốn hai phần bộ nhớ: một cho phép biến đổi mã hóa và một cho phép biến đổi ngược để giải mã.

Làm sao để cải tiến? Mình dùng mô hình Feistel, hoặc mô hình Feistel tổng quát (generalized Feistel network) để xây dựng mã khối mới.

Khối đầu vào (plaintext) có 64 bits sẽ được chia thành 4 phần bằng nhau, mỗi phần có 16 bits:

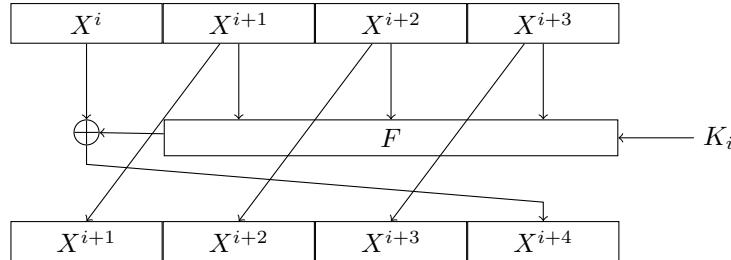
$$P = (X^0, X^1, X^2, X^3).$$

Khóa K có 64 bits sẽ sinh ra các khóa con K_0, K_1, \dots, K_{31} cho 32 vòng.

Tại vòng thứ i với $0 \leq i \leq 31$, X^{i+4} được tính như sau

$$X^{i+4} = X^i \oplus F(X^{i+1} \oplus X^{i+2} \oplus X^{i+3}, K_i),$$

trong đó $F : \mathbb{F}_2^{16} \times \mathbb{F}_2^{16} \rightarrow \mathbb{F}_2^{16}$ là round function.



Ciphertext sau 32 vòng sẽ là trạng thái cuối viết theo thứ tự ngược lại, nghĩa là

$$C = (X^{35}, X^{34}, X^{33}, X^{32}).$$

Round function. Round function $F : \mathbb{F}_2^{16} \times \mathbb{F}_2^{16} \rightarrow \mathbb{F}_2^{16}$ gồm ba phép biến đổi là cộng modulo 2^{16} , S-box và linear layer.

1. Phép cộng modulo 2^{16} , đặt $Y = X^i \oplus X^{i+1} \oplus X^{i+2}$, khi đó ta tính $Y + K_i \pmod{2^{16}}$ và kí hiệu là $Y \boxplus K_i$.
2. S-box là ánh xạ $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$, trong đó 16 bits từ phép cộng trên được chia thành 4 phần, mỗi phần có 4 bits đi qua S-box. Sau đó ta nối các kết quả lại thành 16 bits mới.
3. Linear layer L . Ta xor X với $X \lll 11$, trong đó \lll là phép dịch 11 bits sang trái.

Như vậy

$$S(X) = S(a_0, a_1, a_2, a_3) = (\text{Sbox}(a_0), \text{Sbox}(a_1), \text{Sbox}(a_2), \text{Sbox}(a_3)),$$

với $X = (a_0, a_1, a_2, a_3) \in \mathbb{F}_2^{16}$ và $a_i \in \mathbb{F}_2^4$.

S-box được cho bởi bảng sau

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	0	F	A	2	B	9	5	8	3	D	7	1	E	6	4

Linear layer như sau:

$$L(X) = X \oplus (X \lll 11).$$

Round key. Khóa đầu vào K có 64 bits sẽ được chia thành 8 phần K_0, K_1, \dots, K_7 . Các vòng dùng khóa con như sau:

1. Các vòng 0, 1, ..., 7 sử dụng lần lượt các khóa K_0, K_1, \dots, K_7 .
2. Các vòng 8, 9, ..., 15 sử dụng lần lượt các khóa K_0, K_1, \dots, K_7 .
3. Các vòng 16, 17, ..., 23 sử dụng lần lượt các khóa K_0, K_1, \dots, K_7 .
4. Các vòng 24, 25, ..., 31 sử dụng lần lượt các khóa K_7, K_6, \dots, K_0 .

Decryption. Để giải mã ta thực hiện theo thứ tự ngược lại. Giả sử ciphertext là

$$C = (X^{35}, X^{34}, X^{33}, X^{32}),$$

ta viết theo thứ tự ngược lại là

$$C_{32} = (X^{32}, X^{33}, X^{34}, X^{35}).$$

Tại vòng thứ i với $i = 31, \dots, 0$, X^i sẽ được tính bởi

$$X^i = X^{i+4} \oplus F(X^{i+1} \oplus X^{i+2} \oplus X^{i+3}, K_i),$$

trong đó F là round function được định nghĩa ở trên. Cuối cùng plaintext là

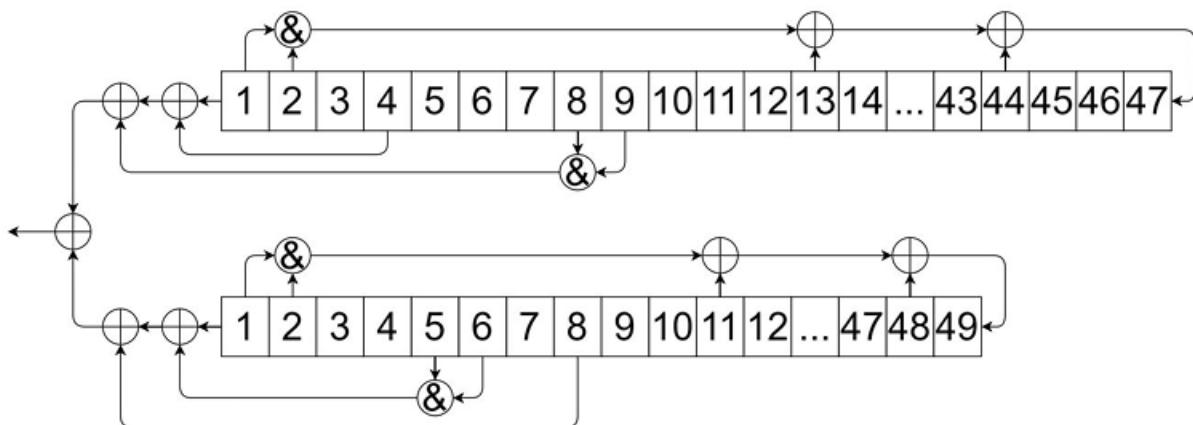
$$P = (X^0, X^1, X^2, X^3).$$

A nonlinear generator

Đây là bài 4 ở round 1 và bài 7 ở round 2.

Đề bài

Alice tạo ra một bộ sinh khóa như hình sau:



Bộ sinh khóa gồm hai thanh ghi độ dài 47 và 49 với các hàm không tuyến tính (non-linear feedback).

Tại mỗi thời điểm $t = 1, 2, \dots$, mỗi thanh ghi sẽ sinh keystream tại trạng thái ở thời điểm t và sau đó chuyển sang trạng thái ở thời điểm $t + 1$.

Cụ thể, trạng thái của thanh ghi ở thời điểm t là

$$A(t) = (a_1(t), \dots, a_{47}(t)) \text{ và } B(t) = (b_1(t), \dots, b_{49}(t)).$$

Hai thanh ghi được dịch chuyển đồng thời. Ví dụ

$$A(t+1) = (a_1(t), \dots, a_{47}(t), (a_1(t) \& a_2(t)) \oplus a_{13}(t) \oplus a_{44}(t)).$$

Keystream γ với độ dài 8192 tạo bởi bộ sinh khóa trên được ghi lại trong file *keystream.txt*. Ngoài ra chúng ta cũng có trạng thái $A(8192)$ và $B(8192)$ là:

$$\begin{aligned} A(8192) &= (00101001110001001110111001100001010100000101110), \\ B(8192) &= (0000010000101001000011000001010111001110000100101). \end{aligned}$$

Bạn có thể tìm lại trạng thái khởi tạo $A(1)$ và $B(1)$ không?

Lời giải

Bài này chúng ta bruteforce từ cặp bit $A(i)$ và $B(i)$ bắt đầu từ vị trí 8191 về 1.

Mình sẽ viết lại dây của đề bài. Dây trên và dây dưới lần lượt là a_0, a_1, \dots và b_0, b_1, \dots và các phần tử sau được sinh ra từ các phần tử trước theo công thức

$$a_{i+47} = (a_i \& a_{i+1}) \oplus a_{i+12} \oplus a_{i+43}$$

và

$$b_{i+49} = (b_i \& b_{i+1}) \oplus b_{i+10} \oplus b_{i+47}.$$

Keystream $\{g_n\}$ sẽ được sinh theo công thức

$$\begin{aligned} g_i &= a_i \oplus a_{i+3} \oplus (a_{i+7} \& a_{i+8}) \\ &\oplus b_i \oplus (b_{i+4} \& b_{i+5}) \oplus b_{i+7}. \end{aligned}$$

Như vậy mình sẽ bruteforce từ vị trí i từ 8190 về 0. Ở mỗi vị trí mình có hai trường hợp a_i và hai trường hợp b_i . Nếu đặt được tới $i = 0$ thì đây là dãy cần tìm.

Keystream là một chuỗi 8192 bits nên mình sẽ lưu ở file `keystream.txt`.

```
from itertools import product
import sys

sys.setrecursionlimit(100000)

g = list(map(int, open("keystream.txt").read()))

def check_a(val, x, i):
    return x[i+47] == (val & x[i+1]) ^ x[i+12] ^ x[i+43]

def check_b(val, y, i):
    return y[i+49] == (val & y[i+1]) ^ y[i+10] ^ y[i+47]

def check_g(valx, valy, x, y, z, i):
    return z[i] == valx ^ x[i+3] ^ (x[i+7] & x[i+8]) ^ valy ^ (y[i+4] & y[i+5]) ^ y[i+7]

a = [-1] * 8191 + list(map(int, '00101001110001001110111001100001010100000101110'))
b = [-1] * 8191 + list(map(int, '0000010000101001000011000001010111001110000100101'))

def bruteforce(i):
    if i == -1:
        assert "".join(map(str, a[:47])) ==
→"011111000011100001100100010110001100011001110"
        assert "".join(map(str, b[:49])) ==
→"1100110011010001010101000101110100011011010010110"
        return
    else:
        for ai, bi in product(range(2), repeat=2):
            if check_a(ai, a, i) and check_b(bi, b, i) and check_g(ai, bi, a, b, g, i):
                a[i], b[i] = ai, bi
                bruteforce(i-1)
                a[i], b[i] = -1, -1

bruteforce(8190)
```

Bài này được giải bởi bạn Uyên và là BEST SOLUTION. Điều thú vị là rất nhiều đội được BEST SOLUTION cho bài này :))

Unsecure SP-network

Đây là bài 5 ở round 1 và bài 8 ở round 2.

Đề bài

Bob có nghe về mạng SP và quyết định xây dựng mật mã của riêng mình.

Block size (kích thước khối) là 32 bits. Bob tạo S-boxes kích thước 2×2 là P -layer sử dụng một phần của secret key.

Ở đây P là một biến đổi tuyến tính $P : \mathbb{F}_2^{32} \rightarrow \mathbb{F}_2^{32}$, nghĩa là

$$P(x \oplus y) = P(x) \oplus P(y)$$

với mọi $x, y \in \mathbb{F}_2^{32}$.

Ngoài ra, secret key là $K \in \mathbb{F}_2^{128}$. Với nó Bob đặt

$$K^i = (K_{32(i \bmod 4)+1}, \dots, K_{32(i \bmod 4)+32})$$

có độ dài 32 với mọi $i \in \{1, \dots, 100\}$.

Như vậy vòng thứ i của mã khôi sẽ như sau:

$$r_i(x) = P(S(x_1 \oplus K_1^i, x_2 \oplus K_2^i), S(x_3 \oplus K_3^i, x_4 \oplus K_4^i), \dots, S(x_{31} \oplus K_{31}^i, x_{32} \oplus K_{32}^i))$$

với $x \in \mathbb{F}_2^{32}$.

Ciphertext sẽ là

$$c = K^{100} \oplus r_{99}(r_{98}(\dots r_1(m)))$$

với m là plaintext ban đầu, $m = (m_1, \dots, m_{32})$ và $m_i \in \mathbb{F}_2$.

Tuy nhiên Bob nhanh chóng nhận ra Carol có thể đọc được plaintext mà không gặp vấn đề nếu Carol biết 100 cặp plaintext và ciphertext bất kì.

Tại sao cô ấy làm được?

Lời giải

Giả sử việc mã hóa có dạng $C = \text{Enc}(P)$. Mình sẽ chứng minh rằng với hai plaintext P và P' , nếu ciphertext tương ứng là C và C' thì

$$C \oplus C' = \text{Enc}(P \oplus P').$$

Một điều khá buồn cười là bài này mình mất kha khá thời gian vì những suy nghĩ ... vĩ mô. Theo đề bài có thể nhận thấy S-box S là ánh xạ từ \mathbb{F}_2^2 tới \mathbb{F}_2^2 . Theo quy tắc nhân có tất cả $2^{2^2} = 16$ ánh xạ từ \mathbb{F}_2^2 tới \mathbb{F}_2 (số lượng hàm boolean hai biến). Do đó có $16^2 = 256$ trường hợp S-box S .

Đối với các S-box tuyến tính hoặc affine thì câu chuyện không phải vấn đề. Tuy nhiên các trường hợp khác rất khó kiểm soát. Lúc này bạn Uyên nói với mình có $4! = 24$ trường hợp của S-box thôi, là số lượng hoán vị trên tập $\{0, 1, 2, 3\}$. Lý do là vì thuật toán mã hóa sử dụng mạng SP, do đó mỗi phép biến đổi phải có phép biến đổi ngược. Chân lý đây rồi, ánh sáng đây rồi :v :v Vậy mà mình nghĩ mãi, haizz.

Ý tưởng của mình là sử dụng phá mã vi sai (differential cryptanalysis).

Với ánh xạ $M : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ mình gọi:

1. **Input differential** là $x_1 \oplus x_2$.
2. **Output differential** là $M(x_1) \oplus M(x_2)$.

Mình dùng SageMath để kiểm tra nhanh phân bố vi sai trên 24 hoán vị (ứng với 24 S-box).

```
import itertools
from sage.crypto.sbox import SBox

for perm in itertools.permutations(range(4)):
    print(f"S-box: {perm}")
    print(SBox(perm).difference_distribution_table())
    print()
```

Một dấu hiệu chung cho mọi S-box là nếu input differential cố định thì output differential cũng cố định.

Ví dụ, với S-box tương ứng với hoán vị

$$S = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 0 & 3 \end{pmatrix}$$

thì bảng phân bố vi sai là

	00	01	10	11
00	4	0	0	0
01	0	0	0	4
10	0	4	0	0
11	0	0	4	0

Theo bảng trên, nếu input differential là 10 thì output differential chắc chắn là 01.

Như vậy mình có nhận xét sau.

Remark 12

Với S-box bất kì là hoán vị trên tập $\{0, 1, 2, 3\}$, nếu input differential cố định thì output differential cũng cố định.

Đầu tiên mình cần viết lại các ký hiệu trong bài này.

Plaintext có 32 bit:

$$m = (m_1, \dots, m_{32}), \quad m_i \in \mathbb{F}_2.$$

S-box là hoán vị trên tập $\{0, 1, 2, 3\}$. Nói cách khác S-box là song ánh S từ \mathbb{F}_2^{32} tới \mathbb{F}_2^{32} .

Đặt \mathcal{S} là ánh xạ biến đổi 32 bit thành 32 bit mới với S-box, nghĩa là

$$\begin{aligned} S : \mathbb{F}_2^{32} \times \mathbb{F}_2^{32} &\rightarrow \mathbb{F}_2^{32} \\ (x, K) &\mapsto X, \end{aligned}$$

trong đó:

- $x = (x_1, \dots, x_{32})$ là khối 32 bit
- $K = (K_1, \dots, K_{32})$ là khóa con ở vòng tương ứng
- $X = (X_1, \dots, X_{32})$ nhận được từ

$$\begin{aligned} (X_1, X_2) &= S(x_1 \oplus K_1, x_2 \oplus K_2), \\ (X_3, X_4) &= S(x_3 \oplus K_3, x_4 \oplus K_4), \\ &\dots \\ (X_{31}, X_{32}) &= S(x_{31} \oplus K_{31}, x_{32} \oplus K_{32}). \end{aligned}$$

Biến đổi tuyến tính là ánh xạ P từ \mathbb{F}_2^{32} tới \mathbb{F}_2^{32} .

Secret key là $K \in \mathbb{F}_2^{128}$. Subkey ở vòng thứ i nhận được từ K theo công thức

$$K^i = (K_{32(i \bmod 4)+1}, \dots, K_{32(i \bmod 4)+32}),$$

với $i \in \{1, \dots, 100\}$. Mỗi subkey K^i có độ dài 32 bit.

Biến đổi ở vòng thứ i là

$$r_i(x) = P(\mathcal{S}(x, K^i)).$$

Ciphertext sau 100 vòng sẽ là

$$c = K^{100} r_{99}(r_{98}(\dots(r_1(m)))).$$

Tiếp theo, sử dụng differential attack mình sẽ tìm liên hệ known-plaintext. Mình xét vòng đầu tiên trước.

Giả sử mình có hai plaintext là

$$m = (m_1, \dots, m_{32}) \text{ và } m' = (m'_1, \dots, m'_{32}) \in \mathbb{F}_2^{32}.$$

Ta có

$$\begin{aligned} (m_1, m_2) &\rightarrow S(m_1 \oplus K_1^1, m_2 \oplus K_2^1) = (X_1, X_2), \\ (m'_1, m'_2) &\rightarrow S(m'_1 \oplus K_1^1, m'_2 \oplus K_2^1) = (X'_1, X'_2). \end{aligned}$$

Input differential cho S-box là

$$(m_1 \oplus K_1^1 \oplus m'_1 \oplus K_1^1, m_2 \oplus K_2^1 \oplus m'_2 \oplus K_2^1) = (m_1 \oplus m'_1, m_2 \oplus m'_2).$$

Ở đây subkey không ảnh hưởng input differential và đây là yếu tố quan trọng cần chú ý.

Nếu mình cố định input differential $(m_1 \oplus m'_1, m_2 \oplus m'_2)$ thì output differential

$$S(m_1 \oplus K_1^1, m_2 \oplus K_2^1) \oplus S(m'_1 \oplus K_1^1, m'_2 \oplus K_2^1) = (X_1, X_2) \oplus (X'_1, X_2) = (X_1 \oplus X'_1, X_2 \oplus X'_2)$$

cũng cố định. Tương tự cho các cặp còn lại $(m_3, m_4), \dots, (m_{31}, m_{32})$.

Như vậy ở vòng đầu ta có

$$\begin{aligned} r_1(m) &= P(\mathcal{S}(m, K^1)) = P(X_1, X_2, \dots, X_{32}), \\ r_1(m') &= P(\mathcal{S}(m', K^1)) = P(X'_1, X'_2, \dots, X'_{32}). \end{aligned}$$

Suy ra

$$r_1(m) \oplus r_1(m') = P(X_1, \dots, X_{32}) \oplus P(X'_1, \dots, X'_{32}) = P(X_1 \oplus X'_1, \dots, X_{32} \oplus X'_{32})$$

vì P là biến đổi tuyến tính. Rõ ràng nếu $X_i \oplus X'_i$ cố định với mọi $i = 1, \dots, 32$ thì kết quả $r_1(m) \oplus r_1(m')$ cũng cố định theo.

Áp dụng kết quả này cho 99 vòng. Ở vòng cuối (vòng 100) ta XOR với K^{100} nhưng thực ra là

$$\begin{aligned} c \oplus c' &= (K^{100} \oplus r_{99}(r_{98}(\dots r_1(m)))) \oplus (K^{100} \oplus r_{99}(r_{98}(\dots r_1(m')))) \\ &= r_{99}(r_{98}(\dots r_1(m))) \oplus r_{99}(r_{98}(\dots r_1(m'))), \end{aligned}$$

tức là K^{100} cũng không ảnh hưởng differential như K_1, \dots, K_{99} .

Tóm lại mình nhận được kết quả sau.

Remark 13

Nếu ta biết plaintext P và ciphertext tương ứng là C , thì với mọi ciphertext C' mới nào đó ta luôn có thể tìm lại được plaintext tương ứng. Nói cách khác là chú ý ở đầu bài

$$C \oplus C' = \mathbf{Enc}(P \oplus P').$$

❶ Chứng minh

Đặt DDT_S là ánh xạ từ input differential thành output differential của S-box. Do input differential cố định thì output differential, DDT_S sẽ nhận được từ bảng phân bố vi sai bằng cách, nếu phần tử ở hàng i và cột j bằng 4 thì $DDT_S(i) = j$.

Ví dụ, với S-box

$$S = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 0 & 3 \end{pmatrix}$$

như trên thì

$$DDT_S = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 3 & 1 & 2 \end{pmatrix}.$$

Rõ ràng DDT_S là hoán vị, hay song ánh. Do đó tồn tại biến đổi ngược

$$DDT_S^{-1} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 2 & 3 & 1 \end{pmatrix}.$$

Đặt P^{-1} là nghịch đảo của P .

Giả sử $c = (c_1, \dots, c_{32})$ và $c' = (c'_1, \dots, c'_{32})$ là các ciphertexts.

Đặt $P^{-1}(c \oplus c') = (C_1, \dots, C_{32})$. Áp dụng DDT_S^{-1} ta có

$$DDT_S^{-1}(C_1, C_2), DDT_S^{-1}(C_3, C_4), \dots, DDT_S^{-1}(C_{31}, C_{32}),$$

và tương tự cho 98 vòng còn lại. Cuối cùng ta sẽ nhận được differential của các plaintexts P và P' .

Do đã biết P và differential $P \oplus P'$ vừa tìm được, ta có thể tìm P' .

```
# binmatrix.py

# https://github.com/xiangzejun/binary_matrix
# For feedback or questions, please contact at xiangzejun@ie.ac.cn

# Implemented by Xiang Zejun, State Key Laboratory of Information Security,
# Institute Of Information Engineering, CAS

from functools import reduce

class DataError(Exception):
    """
    Define my data exception.
    Check whether the elements of the matrix are binaries.
    """
    def __init__(self, x, y):
        """
        store the coordinate of the entry which is not binary.
        """
        self.x = x
        self.y = y
```

(continues on next page)

(continued from previous page)

```

def printError(self):
    print("The element at [{0}][{1}] is NOT binary!".format(self.x, self.y))

class FormatError(Exception):
    """
    Define my format exception.
    Check whether input is a matrix or a square matrix.
    """
    def __init__(self, s):
        self.error = "The input is " + s
    def printError(self):
        print(self.error)

class RankError(Exception):
    """
    Define my rank exception.
    Check whether the square matrix is full rank when calculating its inverse.
    """
    def __init__(self, r):
        self.r = r
    def printError(self):
        print("The matrix is NOT full rank. (rank = {0})".format(self.r))

class BinMatrix:
    def __init__(self, m = [[1]]):
        """
        Initialize a matrix.
        """
        self.m = m
        self.r_len = len(self.m) # row number
        self.c_len = len(self.m[0]) # column number
        # self.length = len(self.m)

    def __convertMatrixToInt(self):
        """
        Convert each row of the binary matrix to an integer.
        """
        return [int(reduce(lambda x , y: x + y, map(str, self.m[i])), 2)
                for i in range(self.r_len)]

    def __appendUnitMatrix(self):
        """
        Append a unit matrix to m_int.
        """
        m_int = self.__convertMatrixToInt()
        return [(1 << (self.r_len + self.c_len - 1 - i)) ^ m_int[i]
                for i in range(self.r_len)]

    def __chooseElement(self, r, c, m_int):
        """
        Choose a none-zero row started from position [r][c].
        """

```

(continues on next page)

(continued from previous page)

```

        err = "The row index can not exceed the column index in row-reduced echelon_
matrix."
        assert r <= c, err

        if c == self.c_len:
            return None
        else:
            mask = (1 << (self.c_len - 1 - c))
            temp = [(m_int[i] & mask) for i in range(r, self.r_len)]
            if mask not in temp:
                return self.__chooseElement(r, c + 1, m_int)
            else:
                return (temp.index(mask) + r, c)

    @staticmethod
    def __switchRows(r1, r2, m_int):
        """
        Switch r1-th and r2-th rows of m_int.
        """
        temp = m_int[r1]
        m_int[r1] = m_int[r2]
        m_int[r2] = temp

    def __addRows(self, r, c, m_int):
        """
        Add the r-th row to all the other rows if the c-th element of
        the corresponding rows are nonzero.
        """
        mask = (1 << (self.c_len - 1 - c))
        it = list(range(self.r_len))
        it.remove(r)
        for i in it:
            if m_int[i] & mask != 0:
                m_int[i] ^= m_int[r]

    def __isMatrix(self):
        """
        Check whether the input is a matrix.
        """
        if [len(l) for l in self.m].count(self.c_len) != self.r_len:
            raise FormatError("NOT a matrix!")
        else:
            pass

    def __isSquareMatrix(self):
        """
        Check whether the input is a square matrix.
        """
        if [len(l) for l in self.m].count(self.r_len) != self.r_len:
            raise FormatError("NOT a Square matrix!")

```

(continues on next page)

(continued from previous page)

```

    else:
        pass

def __isBinary(self):
    """
    Check whether the entries in the input are binaries.
    """
    for i in range(len(self.m)):
        for j in range(len(self.m[i])):
            if self.m[i][j] not in [0,1]:
                raise DataError(i, j)
            else:
                pass

def rank(self):
    """
    Calculate the Rank of the matrix.
    """
    self.__isMatrix()
    self.__isBinary()
    m_int = self.__convertMatrixToInt()
    r = 0
    c = 0
    for i in range(self.r_len):
        arg = self.__chooseElement(r, c, m_int)
        if arg != None:
            r_temp = arg[0]
            c = arg[1]
            self.__switchRows(r, r_temp, m_int)
            self.__addRows(r, c, m_int)
            r += 1
            c += 1
        else:
            return r
    return self.r_len

def det(self):
    """
    Calculate the Det of the matrix.
    """
    self.__isSquareMatrix()
    self.__isBinary()
    if self.rank() == self.r_len:
        return 1
    else:
        return 0

def inv(self):
    """
    Calculate the inverse of the matrix.
    """
    self.__isSquareMatrix()

```

(continues on next page)

(continued from previous page)

```

self.__isBinary()
m_adj = self.__appendUnitMatrix()
r = 0
c = 0
for i in range(self.r_len):
    arg = self.__chooseElement(r, c, m_adj)
    if arg != None:
        r_temp = arg[0]
        c = arg[1]
        self.__switchRows(r, r_temp, m_adj)
        self.__addRows(r, c, m_adj)
        r += 1
        c += 1
    else:
        raise RankError(r)
return [
    list(map(int, list(format((m_adj[i] >> self.c_len), "0" + str(self.r_len) +
    ↵"b"))))
    for i in range(self.r_len)]

```

```

# cipher.py

from binmatrix import BinMatrix
import numpy as np
import random

sbox = list(range(4))
random.shuffle(sbox)      # generate random S-box which is permuation

P = None
while True: # generate random invertible matrix on F_2
    P = np.random.randint(2, size=(32, 32))
    if np.linalg.det(P) % 2 == 1:
        break

P = [row.tolist() for row in P] # convert from numpy.ndarray to list
P_inv = BinMatrix(P).inv()      # calculate inverse of P

difference_distribution_table = [[0 for _ in range(4)] for __ in range(4)]
# i-th row -> input differential
# o-th row -> output differential
for a in range(4):
    for b in range(4):
        i = a ^ b
        o = sbox[a] ^ sbox[b]
        difference_distribution_table[i][o] += 1

sbox_diff = [0] * 4
# calculate DDT_S
for i in range(4):
    for j in range(4):
        if difference_distribution_table[i][j] == 4:

```

(continues on next page)

(continued from previous page)

```

    sbox_diff[i] = j

sbox_diff_inv = [sbox_diff.index(i) for i in range(4)] # DDT_S^{-1}

def matmul(M: list[list[int]], v: list[int]) -> list[int]:
    # multiplication of matrix M and vector v
    result = [0] * len(v)
    for i in range(len(v)):
        result[i] = sum(M[i][j] * v[j] for j in range(len(v))) % 2
    return result

def encrypt(m: list[int], K: list[int]) -> list[int]:
    # encryption with plaintext m, key K
    Ki = [K[i:i+32] for i in range(0, len(K), 32)]
    pt = m.copy()
    for i in range(99):
        pt = [p ^ k for p, k in zip(pt, Ki[i % 4])]
        for j in range(0, len(pt), 2):
            z = (pt[j] << 1) + pt[j+1] # get two bits for input of S-box
            z = sbox[z]
            pt[j] = z >> 1 # write two new bits
            pt[j+1] = z & 1 # after S-box
        pt = matmul(P, pt)
    pt = [p ^ k for p, k in zip(pt, Ki[100 % 4])]
    return pt

```

```

# solve.py

import random
from cipher import *

pt1 = random.choices(range(2), k = 32) # Known
pt2 = random.choices(range(2), k = 32) # Unknown

K = random.choices(range(2), k = 128)

ct1 = encrypt(pt1, K) # Known
ct2 = encrypt(pt2, K) # Known

ct_diff = [x ^ y for x, y in zip(ct1, ct2)] # ct1 ^ ct2

for _ in range(99):
    ct_diff = matmul(P_inv, ct_diff)
    for i in range(0, len(ct_diff), 2):
        z = (ct_diff[i] << 1) + ct_diff[i+1]
        z = sbox_diff_inv[z]
        ct_diff[i] = z >> 1
        ct_diff[i+1] = z & 1

pt2_guess = [x^y for x, y in zip(ct_diff, pt1)] # Recover pt2
assert pt2 == pt2_guess

```

Một điều thú vị là có thể dùng nhiều S-box và quá trình tấn công vẫn tương tự. Ví dụ

$$(x_1, \dots, x_{32}) \xrightarrow{K^i} (S_1(x_1 \oplus K_1^i, x_2 \oplus K_2^i), S_2(x_3 \oplus K_3^i, x_4 \oplus K_4^i), \dots, S_{16}(x_{31} \oplus K_{31}^i, x_{32} \oplus K_{32}^i)),$$

với S_i là hoán vị bất kì trên $\{0, 1, 2, 3\}$, $i = 1, \dots, 16$.

Ở đây ta sẽ áp dụng cùng phương pháp như trên nhưng ta sẽ tính DDT_{S_i} cho mỗi S-box S_i .

Lời giải hoàn chỉnh

Thật ra mình đã đọc thiếu đề và chưa xét trường hợp S-box và P là các phép biến đổi bí mật (giống secret key). Đây là lý do mình không được full điểm câu này.

S-box, theo phân tích của mình ở trên, thực chất là một ánh xạ affine. Nhắc lại ánh xạ affine là ánh xạ có dạng $S(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \oplus \mathbf{b}$, trong đó \mathbf{A} là ma trận, \mathbf{x} và \mathbf{b} là các vector cột.

Khi đó, với hai đầu vào \mathbf{x} và \mathbf{x}' sao cho $\mathbf{x} \oplus \mathbf{x}'$ cố định (input differential) thì ta có

$$S(\mathbf{x}) \oplus S(\mathbf{x}') = \mathbf{A} \cdot (\mathbf{x} \oplus \mathbf{x}')$$

cũng cố định do \mathbf{A} đã xác định.

Theo đó toàn bộ quá trình encrypt là một ánh xạ affine lớn, do đó chúng ta hoàn toàn có thể tìm được ánh xạ affine đó nếu biết đủ 32 vector độc lập tuyến tính.

Sử dụng *cipher.py* ở trên và đoạn code sau để giải:

```
import random
from sage.all import random_vector, vector, GF, matrix, ones_matrix
from cipher import encrypt

random.seed(1333)

# def encrypt(m, p, s, k):
#     return

K = [random.randrange(0, 2) for _ in range(128)]
ms = [random_vector(GF(2), 32) for _ in range(100)]

cs = [encrypt(list(map(int, m)), K) for m in ms]

ms = matrix(GF(2), ms)
cs = matrix(GF(2), cs)
A = matrix(GF(2), 32, 32)
b = vector(GF(2), 32)

for i in range(32):
    for bb in range(2):
        try:
            res = ms.solve_right(cs[:, i] + bb * ones_matrix(GF(2), 100, 1))
            A[:, i] = res
            b[i] = bb
        except:
            pass

u = random_vector(GF(2), 32)
assert encrypt(list(map(int, u)), K) == list(map(int, u * A + b))
```

Ở đây mình viết plaintext ở dạng hàng. Giả sử plaintext và ciphertext là

$$M = (m_1, \dots, m_n), \quad C = (c_1, \dots, c_n).$$

Ánh xạ affine của chúng ta tương ứng với

$$(c_1, \dots, c_n) = (m_1, \dots, m_n) \cdot \mathbf{A} + (b_1, \dots, b_n),$$

trong đó \mathbf{A} là ma trận $n \times n$ (trong trường hợp bài này $n = 32$).

Như vậy mình cần tìm ma trận \mathbf{A} và vector $\mathbf{b} = (b_1, \dots, b_n)$. Nếu có T phương trình như vậy

$$\begin{aligned} (c_{11}, \dots, c_{1n}) &= (m_{11}, \dots, m_{1n}) \cdot \mathbf{A} + (b_1, \dots, b_n), \\ (c_{21}, \dots, c_{2n}) &= (m_{21}, \dots, m_{2n}) \cdot \mathbf{A} + (b_1, \dots, b_n), \\ &\dots = \dots \\ (c_{T1}, \dots, c_{Tn}) &= (m_{T1}, \dots, m_{Tn}) \cdot \mathbf{A} + (b_1, \dots, b_n), \end{aligned}$$

thì mình giải từng cột thứ i của ma trận A và từng số hạng b_i . Nghĩa là

$$\begin{pmatrix} c_{11} \\ c_{12} \\ \vdots \\ c_{T1} \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ m_{T1} & m_{T2} & \cdots & m_{Tn} \end{pmatrix} \cdot \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{pmatrix} + \begin{pmatrix} b_1 \\ b_1 \\ \vdots \\ b_1 \end{pmatrix}.$$

Như vậy chúng ta bruteforce b_1 và kiểm tra xem phương trình có nghiệm $\begin{pmatrix} a_{11} \\ \vdots \\ a_{n1} \end{pmatrix}$ không. Nếu có thì đây là kết quả cần tìm.

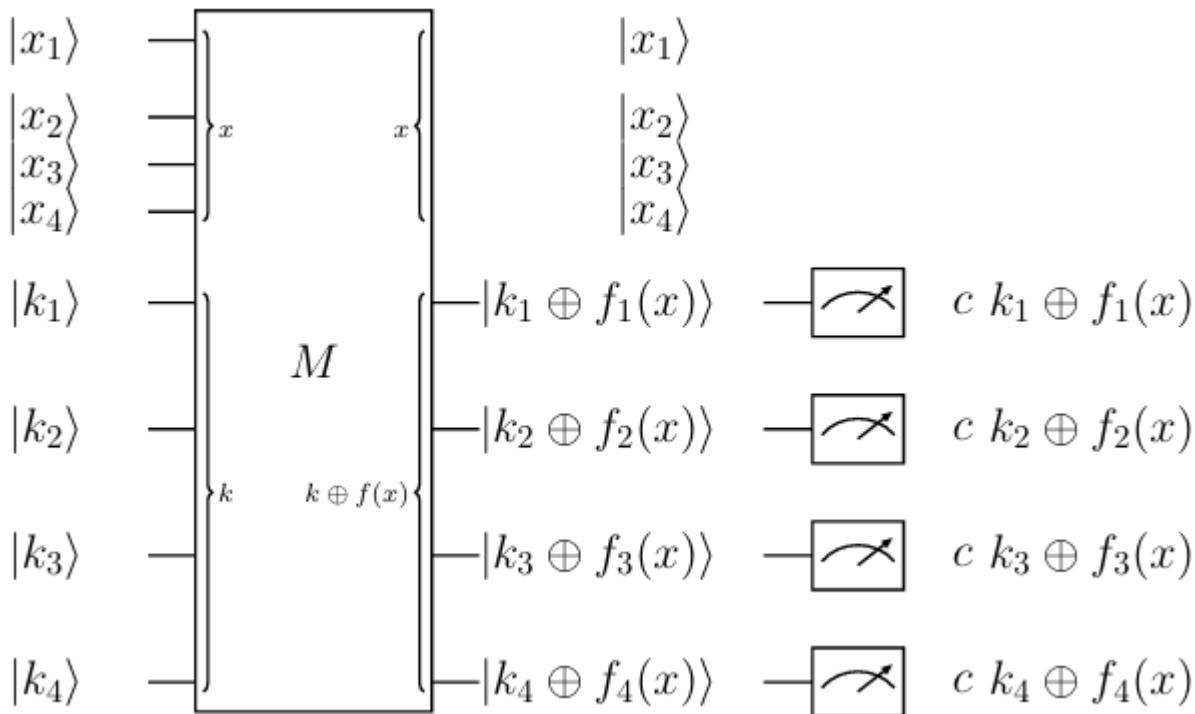
Tiến hành cho từng cột của \mathbf{A} và từng phần tử b_i sẽ tìm được ánh xạ affine.

Unknown function

Đây là bài 8 ở round 1 và bài 10 ở round 2.

Đề bài

Bob tạo một máy lượng tử M mã hóa 4-bit word với 4-bit secret key $k = (k_1, k_2, k_3, k_4)$ theo mạch sau:



Máy này thực hiện trên 4-bit plaintext $x = (x_1, x_2, x_3, x_4)$ với 4-qubit tương ứng, gọi là "plainstate", là $|x_1, x_2, x_3, x_4\rangle$, tương tự với "keystate" là $|k_1, k_2, k_3, k_4\rangle$.

Mô hình tính toán là

$$|x\rangle|k\rangle \xrightarrow{M} |x\rangle|k \oplus f(x)\rangle,$$

trong đó

$$f(x) = (f_1(x), f_2(x), f_3(x), f_4(x))$$

là hàm Boolean vectorial khả nghịch trên 4 biến. Khi đó "cipherstate" là

$$|k_1 \oplus f_1(x), k_2 \oplus f_2(x), k_3 \oplus f_3(x), k_4 \oplus f_4(x)\rangle$$

và cuối cùng ta thu được ciphertext.

Alice có thể khôi phục secret key k không nếu cô ấy không biết hàm f ?

Giả sử Alice có oracle access tới máy lượng tử với khóa cố định k và thu được thông tin

$$f(0, 0, 0, 0) \oplus f(1, 1, 1, 1) \oplus f(1, 0, 0, 0) = c \in \mathbb{F}_2^4$$

với c đã biết.

Lời giải

Đặt $\mathbf{x} = (x_1, x_2, x_3, x_4) \in \mathbb{F}_2^4$.

Kí hiệu bảng chân trị của các hàm bool $f_1(\mathbf{x})$, $f_2(\mathbf{x})$, $f_3(\mathbf{x})$ và $f_4(\mathbf{x})$ là

x_1	x_2	x_3	x_4	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	$f_3(\mathbf{x})$	$f_4(\mathbf{x})$
0	0	0	0	m_0	n_0	p_0	q_0
0	0	0	1	m_1	n_1	p_1	q_1
0	0	1	0	m_2	n_2	p_2	q_2
0	0	1	1	m_3	n_3	p_3	q_3
0	1	0	0	m_4	n_4	p_4	q_4
0	1	0	1	m_5	n_5	p_5	q_5
0	1	1	0	m_6	n_6	p_6	q_6
0	1	1	1	m_7	n_7	p_7	q_7
1	0	0	0	m_8	n_8	p_8	q_8
1	0	0	1	m_9	n_9	p_9	q_9
1	0	1	0	m_{10}	n_{10}	p_{10}	q_{10}
1	0	1	1	m_{11}	n_{11}	p_{11}	q_{11}
1	1	0	0	m_{12}	n_{12}	p_{12}	q_{12}
1	1	0	1	m_{13}	n_{13}	p_{13}	q_{13}
1	1	1	0	m_{14}	n_{14}	p_{14}	q_{14}
1	1	1	1	m_{15}	n_{15}	p_{15}	q_{15}

Ở đây m_i là giá trị của hàm $f_1(\mathbf{x})$ tại dòng thứ i với $0 \leq i \leq 15$. Tương tự cho n_i , p_i và q_i .

Ta đã biết

$$f(0, 0, 0, 0) \oplus f(1, 1, 1, 1) \oplus f(1, 0, 0, 0) = \mathbf{c} \in \mathbb{F}_2^4$$

với \mathbf{c} đã biết.

Như vậy ta có

$$\begin{aligned} f(0, 0, 0, 0) &= (m_0, n_0, p_0, q_0), \\ f(1, 1, 1, 1) &= (m_{15}, n_{15}, p_{15}, q_{15}), \\ f(1, 0, 0, 0) &= (m_8, n_8, p_8, q_8), \end{aligned}$$

suy ra

$$\mathbf{c} = (m_0 \oplus m_{15} \oplus m_8, n_0 \oplus n_{15} \oplus n_8, p_0 \oplus p_{15} \oplus p_8, q_0 \oplus q_{15} \oplus q_8).$$

"Cipherstate"

$$|k_1 \oplus f_1(\mathbf{x}), k_2 \oplus f_2(\mathbf{x}), k_3 \oplus f_3(\mathbf{x}), k_4 \oplus f_4(\mathbf{x})\rangle$$

chính là 4 bit ciphertext

$$(k_1 \oplus f_1(\mathbf{x}), k_2 \oplus f_2(\mathbf{x}), k_3 \oplus f_3(\mathbf{x}), k_4 \oplus f_4(\mathbf{x})).$$

Vì Alice có truy cập oracle tới máy với khóa cố định k nên

- Nếu Alice gửi plaintext $(0, 0, 0, 0)$ thì sẽ nhận lại ciphertext

$$\mathbf{C}_0 = (k_1 \oplus m_0, k_2 \oplus n_0, k_3 \oplus p_0, k_4 \oplus q_0).$$

- Nếu Alice gửi plaintext $(1, 1, 1, 1)$ thì sẽ nhận lại ciphertext

$$\mathbf{C}_{15} = (k_1 \oplus m_{15}, k_2 \oplus n_{15}, k_3 \oplus p_{15}, k_4 \oplus q_{15}).$$

- Nếu Alice gửi plaintext $(1, 0, 0, 0)$ thì sẽ nhận lại ciphertext

$$\mathbf{C}_8 = (k_1 \oplus m_8, k_2 \oplus n_8, k_3 \oplus p_8, k_4 \oplus q_8).$$

Nói cách khác là

$$\begin{aligned} \mathbf{C}_0 \oplus \mathbf{C}_{15} \oplus \mathbf{C}_8 &= (k_1 \oplus (m_0 \oplus m_{15} \oplus m_8), k_2 \oplus (n_0 \oplus n_{15} \oplus n_8), k_3 \oplus (p_0 \oplus p_{15} \oplus p_8), k_4 \oplus (q_0 \oplus q_{15} \oplus q_8)) \\ &= (k_1, k_2, k_3, k_4) \oplus \mathbf{c}. \end{aligned}$$

Ta đã biết vector \mathbf{c} , và có thể tính $\mathbf{C}_0 \oplus \mathbf{C}_{15} \oplus \mathbf{C}_8$ nên hoàn toàn có thể tìm lại được khóa \mathbf{k} mà không cần hàm f .

A simple hash function

Đây là bài 11 ở round 2.

Đề bài

Carol tạo một thuật toán hash mới. Khóa $k = (k_1, \dots, k_6)$ là vector nhị phân độ dài 6.

Đầu vào hàm hash là một dãy các chữ số. Dãy này được chia thành các khối độ dài 6. Nếu độ dài của dãy không chia hết cho sau thì ta thêm lần lượt các chữ số 1, 2, 3, ... vào sau dãy cho đến khi độ dài của dãy chia hết cho 6.

Mỗi khối, giả sử là (p_1, \dots, p_6) , được biến đổi thành số nguyên theo quy tắc

$$(-1)^{k_1} \cdot p_1 + \dots + (-1)^{k_6} \cdot p_6.$$

Ở đây $(-1)^0 = 1$ và $(-1)^1 = -1$.

Kết quả sau khi tính toán các khối n_1, n_2, \dots sau đó được tổng hợp thành giá trị hash là

$$H = n_1 - n_2 + n_3 - n_4 + \dots$$

Carol áp dụng thuật toán hash này vào hệ thống log của trang web ngân hàng. Mỗi khi người dùng nhập mật khẩu P thì hệ thống sẽ tính toán hash $H(P, K)$ và so sánh với giá trị hash lưu trên cơ sở dữ liệu để xem người dùng có nhập đúng mật khẩu không.

Nhưng mà Carol nhận ra hệ thống này không an toàn. Việc tìm va chạm (collision) là có thể và từ đó kẻ tấn công có thể truy cập hệ thống. Họ làm như thế nào?

Hãy đề xuất thuật toán đơn giản nhất để tìm va chạm cho mọi dãy đầu vào P biết trước nếu khóa K không biết.

Hơn nữa hãy tìm va chạm ngắn nhất cho dãy trong ví dụ (dãy $P = 134875512293$). Lưu ý rằng do không biết K nên chúng ta cũng không biết giá trị hash $H(P, K)$.

Lời giải

Bài này mình có chút nhầm lẫn nên không được full điểm.

Giả sử input có t khối là n_1, n_2, \dots, n_t . Mỗi khối có 6 chữ số.

Mình sẽ chứng minh rằng khi t chẵn thì ta luôn tìm được collision mà không cần khóa K .

Đầu tiên, với $t = 2$, giả sử hai khối là

$$n_1 = (p_{1,1}, \dots, p_{1,6}), \quad n_2 = (p_{2,1}, \dots, p_{2,6}).$$

Giả sử khóa là $K = (k_1, \dots, k_6)$. Khi đó hash là

$$\begin{aligned} H &= (-1)^{k_1} \cdot p_{1,1} + \dots + (-1)^{k_6} \cdot p_{1,6} \\ &\quad - [(-1)^{k_1} \cdot p_{2,1} + \dots + (-1)^{k_6} \cdot p_{2,6}] \end{aligned}$$

và nếu mình tăng hoặc giảm $p_{1,i}$ và $p_{2,i}$, với $1 \leq i \leq 6$, cùng một lượng, ví dụ như

$$p'_{1,1} = p_{1,1} + a, \quad p'_{2,1} = p_{2,1} + a,$$

và thay $p'_{1,1}$ và $p'_{2,1}$ vào hash thì được

$$\begin{aligned} H' &= (-1)^{k_1} \cdot p'_{1,1} + \dots + (-1)^{k_6} \cdot p'_{1,6} - [(-1)^{k_1} \cdot p'_{2,1} + \dots + (-1)^{k_6} \cdot p'_{2,6}] \\ &= (-1)^{k_1} \cdot (p_{1,1} + a) + \dots + (-1)^{k_6} \cdot p_{1,6} - [(-1)^{k_1-1} \cdot (p_{2,1} + a) + \dots + (-1)^{k_6} \cdot p_{2,6}] \\ &= H + (-1)^{k_1} \cdot a - [(-1)^{k_1} \cdot a] = H. \end{aligned}$$

Điều đó nghĩa là nếu tăng hoặc giảm $p_{1,i}$ và $p_{2,i}$ cùng lượng, tức là

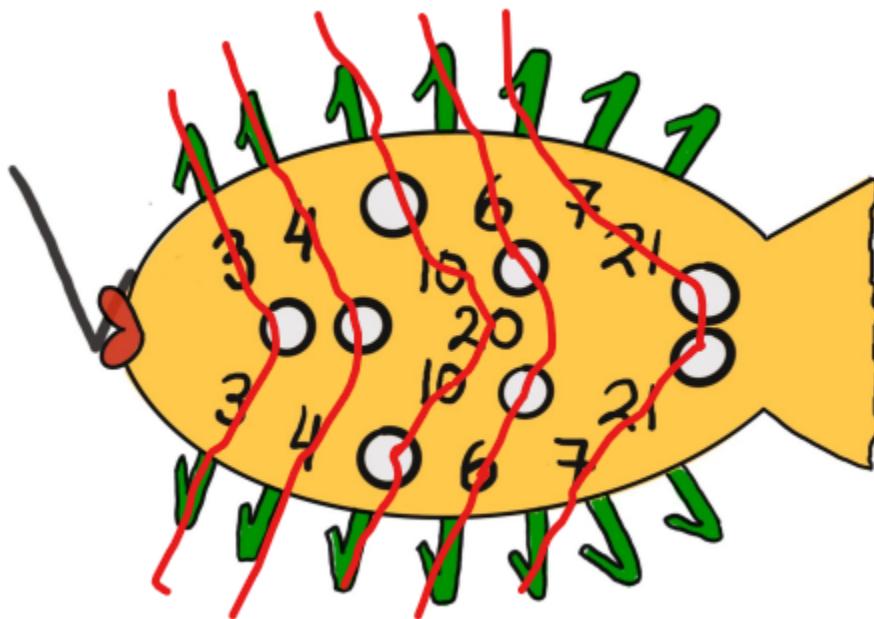
$$p'_{1,i} = p_{1,i} + a_i, \quad p'_{2,i} = p_{2,i} + a_i, \text{ where } a_i \in \{-1, 0, 1\},$$

và phải thỏa điều kiện $1 \leq p'_{1,i} \leq 9$ và $1 \leq p'_{2,i} \leq 9$ thì mình sẽ tìm được input mới có cùng hash.

Khi $t = 4, 6, \dots$ thì sử dụng phương pháp tương tự cho các cặp khối n_{2m+1} và n_{2m+2} (mỗi cặp sử dụng các giá trị a_i tùy ý miễn thỏa các điều kiện trên) thì ta luôn tìm được collision bất kể khóa K là gì.

Cryptographic Fish

Đây là bài 1 ở round 1.



Đây là tam giác Pascal.

Mình đã có một hàng là

$$1 \rightarrow 3 \rightarrow 3 \rightarrow 1.$$

Hàng kế tiếp là

$$1 \rightarrow 4 \rightarrow 6 \rightarrow 4 \rightarrow 1.$$

Hàng kế nữa là

$$1 \rightarrow 5 \rightarrow 10 \rightarrow 20 \rightarrow 10 \rightarrow 5 \rightarrow 1.$$

Tiếp theo sẽ là

$$1 \rightarrow 6 \rightarrow 15 \rightarrow 15 \rightarrow 6 \rightarrow 1$$

vì theo thứ tự chúng ta sẽ cần C_6^2 (đi từ trên xuống) và C_6^4 (đi từ dưới lên).

Hàng cuối sẽ là

$$1 \rightarrow 7 \rightarrow 21 \rightarrow 35 \rightarrow 35 \rightarrow 21 \rightarrow 7 \rightarrow 1.$$

Còn về ô đầu tiên thì cần điền số 2 do dòng trước 3 là 2 theo tam giác Pascal.

Như vậy kết quả là

$$2 + 6 + 5 + 5 + 15 + 15 + 35 + 35 = 118.$$

4.6.2 Internet Olympiad 2024

Vòng 2

Bài 2

Đề bài

Cho hai số tự nhiên x, y sao cho

$$A = \frac{2y}{x(y-x)}, B = \frac{(y-x)(y+1)}{2y^2} \in \mathbb{Z}.$$

Tìm x, y .

Lời giải

Do $A, B \in \mathbb{Z}$ nên

$$A \cdot B = \frac{2y}{x(y-x)} \cdot \frac{(y-x)(y+1)}{2y^2} = \frac{y+1}{xy} \in \mathbb{Z}$$

Suy ra tồn tại $k \in \mathbb{Z}$ sao cho $y+1 = kxy$, tương đương với $y(kx-1) = 1$. Điều này chỉ xảy ra khi $y = 1, x = 2$ (có một nghiệm khác là $y = 1, x = 1$ nhưng sẽ không thỏa mãn số của A).

Bài 8

Đề bài

Cho x, y là các số thực thỏa $x^2 + y^2 + xy = x + y$. Tìm giá trị lớn nhất của $x^2 + y^2$.

Lời giải

Thầy mình bảo đây là dạng bậc hai nên có thể biến đổi để thành phương trình ellipse. Ở đây mình giải theo cách của mình.

Ta có

$$\begin{aligned}x^2 + y^2 + xy &= x + y \\2x^2 + 2y^2 + 2xy &= 2x + 2y \\x^2 + y^2 &= -(x + y)^2 + 2(x + y) = -t^2 + 2t = f(t)\end{aligned}$$

Ta có $f'(t) = -2t + 2$, $f'(t) = 0 \Leftrightarrow t = 1$.

Do đó $f(t)_{\max} = f(1) = 1$.

Bài 9

Đề bài

Xét đa thức $P(x) = x^4 - 4x^2 - x + 1$. Tính $\sum_{i=1}^4 \frac{2x_i + 1}{(x_i^2 - 1)^2}$ với x_i là các nghiệm của $P(x)$.

Lời giải

Ta biến đổi

$$P(x) = x^4 - 4x^2 - x + 1 = (x^2 - 1)^2 - x(2x + 1).$$

Do x_i là nghiệm nên $P(x_i) = 0$, tương đương với $\frac{1}{x_i} = \frac{2x_i + 1}{(x_i^2 - 1)^2}$.

Tổng trên tương đương với $\sum \frac{1}{x_i}$. Dùng Viete có thể tính ra.

Bài 10

Đề bài

Cho hàm số

$$f(x) = \frac{4x - 3}{(x + 2)(x + 2^2) \dots (x + 2^{2023})}$$

Gọi M là đạo hàm của $f(x)$ tại $x_0 = 0$. Tính giá trị $2^{1013 \cdot 2023} \cdot M - 7 \cdot 2^{2023}$.

Lời giải

Đặt $f(x) = \frac{4x - 3}{g(x)}$ thì

$$f'(x) = \frac{4g(x) - (4x - 3)g'(x)}{g^2(x)}.$$

Do

$$g(x) = \prod_{i=1}^{2023} (x + 2^i),$$

ta lấy log hai vế thì được

$$\ln g = \sum_{i=1}^{2023} \ln(x + 2^i)$$

Suy ra

$$\frac{g'}{g} = \sum_{i=1}^{2023} \frac{1}{x+2^i}$$

nên suy ra

$$g'(0) = g(0) \cdot \sum_{i=1}^{2023} \frac{1}{2^i}$$

$$\text{Như vậy } f'(0) = \frac{4 - 3 \sum_{i=1}^{2023} \frac{1}{2^i}}{\prod_{i=1}^{2023} 2^i} = M.$$

Vòng siêu chung kết (Uzbekistan)

Bài 1

Đề bài

Một người đi bộ từ vị trí A đến vị trí B tại thời điểm T với $0 < T < 12$. Cùng lúc đó có một người đi bộ khác từ B đến A . Hai người đi với cùng vận tốc và gặp nhau lúc 12 giờ nhưng không dừng lại mà tiếp tục đi. Người đầu tiên tới B lúc 16 giờ và người thứ hai tới A lúc 21 giờ. Tìm thời điểm T .

Lời giải

Kết quả là 6.

Bài 2

Đề bài

Tâm của 6 đường tròn có cùng bán kính R nằm trên cùng đường thẳng nằm ngang. Khoảng cách giữa các tâm bằng nhau và diện tích vùng giao nhau cũng bằng nhau và bằng 8.

Giả sử A là điểm thấp nhất của đường tròn ngoài cùng bên trái và B là điểm cao nhất của đường tròn ngoài cùng bên phải. Diện tích phần giới hạn bởi đường thẳng AB và các đường tròn bằng 40. Tìm bán kính R .

Lời giải

$$\text{Đáp án là } R = \frac{2\sqrt{5}}{\sqrt{\pi}}.$$

Bài 3

Chả hiểu đề nói gì @@@

Bài 4

Đề bài

Đặt

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0$$

là đa thức có bậc lẻ với hệ số nguyên. Xét tập hợp các điểm của nó trên đồ thị với tọa độ là các số nguyên:

$$M = \{P_i = (b_i, f(b_i)) : b_i \in \mathbb{Z}\}.$$

Chứng minh rằng tập hợp tất cả các cặp $(P_i, P_j) \in M$ với $i \neq j$ thì khoảng cách giữa hai điểm đó là số nguyên và hữu hạn.

Giải

Đặt $d(P, Q) \in \mathbb{Z}$ là khoảng cách giữa hai điểm $P(a, f(a))$ và $Q(b, f(b))$. Khi đó

$$d^2 = (b - a)^2 + (f(b) - f(a))^2$$

là số chính phương. Tuy nhiên

$$f(b) - f(a) = a_n(b^n - a^n) + a_{n-1}(b^{n-1} - a^{n-1}) + \dots + a_1(b - a) = (b - a) \cdot U.$$

Do đó

$$d^2 = (b - a)^2 + ((b - a) \cdot U)^2 = (b - a)^2(1 + U^2)$$

và $1 + U^2$ cũng là số chính phương. Điều này xảy ra khi và chỉ khi $U^2 = 0$, hay $f(b) = f(a)$. Đối với đa thức bậc lẻ thì luôn tồn tại số N sao cho với $x \in (-\infty, N) \cup (N, +\infty)$ thì đa thức $f(x)$ tăng nghiêm ngặt (khi $a_n > 0$) và giảm nghiêm ngặt (khi $a_n < 0$). Do đó trên khoảng $(-N, N)$ số lượng cặp (P, Q) mà $f(b) = f(a)$ là hữu hạn.

Bài 5

Đề bài

Cho ma trận vuông A bậc 2025 sao cho tổng mọi phần tử của $(E + A)^{-1}$, với E là ma trận đơn vị, bằng 10. Tính tổng mọi phần tử của ma trận $(E + A^{-1})^{-1}$.

Lời giải

Đáp án là 2015.

Bài 6

Đề bài

Hàm $f(x)$ liên tục trên $[0, +\infty)$. Biết rằng mọi giá trị của hàm đó nằm trong đoạn $[0, 1]$ và với mọi x, y không âm đều thỏa $f(x + y) \leq f(x)f(y)$. Chứng minh rằng với mọi x không âm ta đều có $\int_0^x f(u) du \geq x\sqrt{f(2x)}$.

Giải

Vì $f(x) \in [0, 1]$, từ điều kiện $f(x + y) \leq f(x)f(y)$ suy ra $f(x + y) \leq f(x)$. Như vậy $f(x)$ không tăng.

Do đó

$$\left(\int_0^x f(u) du \right)^2 = \int_0^x \int_0^x f(u)f(t) du dt \geq \int_0^x dt \int_0^x f(u+t) du = \int_0^x dt \int_t^{x+t} f(s) ds.$$

Vì với mọi $t \in [0, x]$ và $s \in [t, x+t]$ ta có $s \leq 2x$ nên $f(s) \leq f(2x)$ và

$$\int_0^x dt \int_t^{x+t} f(s) ds \geq f(2x) \int_0^x dt \int_t^{x+t} ds = f(2x) \cdot x^2.$$

Kết quả là $\int_0^x f(u) du \geq x\sqrt{f(2x)}$.

Bài 7

Đề bài

Đặt A là tập hợp gồm các cặp (x, y) sao cho $x \in [0, 1]$ và $y \in [0, 1]$, nghĩa là

$$A = \{(x, y) : x \in [0, 1], y \in [0, 1]\}$$

Trên tập A xét hàm số

$$Q(x, y) = \sum_{\frac{1}{2} \leq \frac{m}{n} \leq 2, n, m \in \mathbb{N}} x^m y^n$$

với n, m là các số nguyên dương. Tìm giá trị của giới hạn

$$\lim_{\substack{(x,y) \rightarrow (1,1) \\ (x,y) \in A}} |(1 - xy^2) \cdot (1 - x^2y) \cdot Q(x, y)|.$$

Lời giải

Đáp án là 3.

Bài 8

Đề bài

Chứng minh rằng tổng

$$\frac{1}{4} + \frac{1}{7} + \dots + \frac{1}{3n+1}$$

không phải số nguyên với mọi số tự nhiên n .

Bài 9

Đề bài

Đặt x_1, x_2, \dots, x_n với $n > 1$ là các số thuộc $[0, 1]$ và khác nhau đôi một.

Đặt A_k là trung bình của tất cả tích gồm k phần tử khác nhau.

Chứng minh rằng dãy A_k không tăng.

Bài 10

Đề bài

Trên mặt phẳng cho các điểm A_1, A_2, \dots, A_n không nằm cùng một đường thẳng. Với mọi $1 \leq i, j, k \leq n$ và $i \neq j$, ta định nghĩa δ_{ijk} bằng 1 nếu điểm A_k nằm trên đường thẳng $A_i A_j$, bằng 0 nếu ngược lại.

Chứng minh rằng hệ vector

$$\vec{v}_{ij} = (\delta_{ij1}, \delta_{ij2}, \dots, \delta_{ijn}), 1 \leq i < j \leq n$$

sinh ra không gian \mathbb{R}^n .

4.6.3 0xL4ugh CTF 2024

RSA_GCD

```
i chall.py
import math
from Crypto.Util.number import *
from secret import flag,p,q
from gmpy2 import next_prime
m = bytes_to_long(flag.encode())
n=p*q

power1=getPrime(128)
power2=getPrime(128)
out1=pow((p+5*q),power1,n)
out2=pow((2*p-3*q),power2,n)
eq1 = next_prime(out1)

c = pow(m,eq1,n)

with open('chall2.txt', 'w') as f:
    f.write(f"power1={power1}\npower2={power2}\neq1={eq1}\nout2={out2}\nc={c}\nn={n}")
```

Đề bài cho mình thuật toán RSA với các tham số gồm power1, power2, eq1, out2, c và n.

Trong đó c là ciphertext RSA với modulo là n, số mũ là eq1.

Mình đã có out2 nhưng chưa có out1. Vì eq1 là số nguyên tố tiếp theo kề từ out1, nếu mình gọi eq0 là số nguyên tố trước eq1 thì out1 sẽ nằm trong khoảng [eq0, eq1] và mình sẽ bruteforce out1 trong khoảng này.

Để tìm p và q , do $o_1 = (p + 5q)^{e_1} \pmod{n}$ và $o_2 = (2p - 3q)^{e_2} \pmod{n}$ (ở đây o_1 và o_2 tương ứng out1 và out2, e_1 và e_2 tương ứng power1 và power2).

Vì $o_1^{e_2} = (p + 5q)^{e_1 e_2} \pmod{n}$, mà $q \mid n$ nên

$$o_1^{e_2} = (p + 5q)^{e_1 e_2} \pmod{q} = p^{e_1 e_2} \pmod{q}.$$

Thêm một bước nữa là

$$o_1^{e_2} \cdot 2^{e_1 e_2} = (2p)^{e_1 e_2} \pmod{q}.$$

Tương tự, $o_2^{e_1} = (2p)^{e_1 e_2} \pmod{q}$.

Như vậy $o_1^{e_2} \cdot 2^{e_1 e_2} - o_2^{e_1}$ chia hết cho q nên có ước chung lớn nhất với n bằng q . Từ đó mình tìm được q và p và decrypt plaintext. Các bạn có thể xem code giải ở đây.

Poisson

Đề bài mình để ở đây.

Đề bài cho mình đường cong elliptic với điểm generator G . Flag được biểu diễn bằng dãy nhị phân my_priv .

Ở mỗi vòng lặp ở vị trí $count$, chương trình sẽ

- sinh random số k ;
- tính điểm $Q = my_priv * G$;
- random điểm M trong elliptic;
- tính điểm $C_1 = k * G$;
- tính điểm $C_2 = M + k * Q$;
- tính $new_priv = poisson(my_priv, ind)$ là đảo bit ở vị trí ind . Ví dụ, với dãy nhị phân 111001 và ind là 2 thì kết quả là 110001 (vị trí thứ hai là bit 1 đổi thành bit 0);
- tính điểm $dec = C_2 - new_priv * C_1$.

Mình để ý rằng, khi đảo một bit thì lúc trừ số ban đầu và số sau khi đảo sẽ ra lũy thừa của 2 (có thể là số âm). Với ví dụ ở trên, $111001 - 110001 = 1000$ chính là 2^3 .

Như vậy, biến đổi một chút mình có

$$\begin{aligned} dec &= C_2 - new_priv * C_1 = M + k * Q - new_priv * k * G \\ &= M + k * my_priv * G - k * new_priv * G \\ \Leftrightarrow dec - M &= (my_priv - new_priv) * k * G \\ &= (my_priv - new_priv) * C_1. \end{aligned}$$

Do đó, ở vị trí i , nếu $dec - M = 2^i * C_1$ thì bit ở vị trí i là 1 (do 1 đổi thành 0 nên số lúc trước lớn hơn số lúc sau). Tương tự, nếu $dec - M = -2^i * C_1$ thì bit ở vị trí i là 0 .

Code giải mình để ở đây.

L4ugh

Đề gồm hai file là `challenge.py` và `utils.py`.

Ở bài này, đầu tiên chương trình sinh số nguyên tố d có 666 bit sao cho 333 bit đầu cũng là số nguyên tố.

Đặt $d = d_e \cdot 2^{333} + d_g$, tương ứng trong chương trình d_e là `d_evil` còn d_g là `d_good`.

Chương trình cho phép chọn một trong ba option. Mình sẽ dùng option 1 để tìm d_e , dùng option 2 để tìm d_g . Để sử dụng option 3 mình cần cung cấp d hoàn chỉnh.

1. Tìm d "good"

Phần này dễ hơn nên mình làm trước. Từ `d_good` và 10 số nguyên tố sinh ngẫu nhiên, mình cần nhập số `user_input` nào đó không nhiều hơn 333 bit và chương trình sẽ trả lại mảng $d_g \cdot input + p_i$, $0 \leq i \leq 9$ và p_i là số nguyên tố.

Mình muốn d_g và p_i độc lập với nhau, nghĩa là càng ít liên quan nhau càng tốt. Khi đó mình sẽ có lợi thế trích xuất thông tin hơn. Để ý rằng do p_i là số nguyên tố 333 bit nên nếu chọn $input = 2^{333}$ thì p_i là 333 bit thấp của kết quả và d_g là 333 bit cao của kết quả. Tuy nhiên 2^{333} có 334 bit nên không được. Ở đây mình dùng trick lò là chọn $input = 2^{333} - 1$.

Khi đó

$$r_i = d_g \cdot (2^{333} - 1) + p_i = d_g \cdot 2^{333} + (p_i - d_g).$$

Lúc này d_g thực sự là 333 bit cao của r_i nhưng chỉ khi $p_i - d_g > 0$. Từ 10 phần tử của mảng mìn có thể chọn ra vị trí mà $r_i - d_g \cdot (2^{333} - 1)$ là số nguyên tố. Khi đó mìn lấy d_g thôi.

```
from pwn import process, remote
import json
from Crypto.Util.number import isPrime
from sage.all import *

pr = process(["python", "challenge.py"])
#pr = remote("20.55.48.101", 1337)
pr.recvuntil(b'all input data is in json')

pr.recvuntil(b'option:\t')
pr.sendline(json.dumps({"option": "2"}).encode())

pr.recvuntil(b"Enter your payload:\t")
pr.sendline(str(2**333-1).encode())

res = eval(pr.recvline().strip().decode()[7:])

kres = [r // (2**333) for r in res]

d_goods = [r - k * (2**333 - 1) for r, k in zip(res, kres)]
d_good = None

for r, k in zip(res, kres):
    d_ = r - k * (2**333 - 1)
    if isPrime(d_):
        d_good = k

assert d_good != None
print(f"d_good = {d_good}")

d_good =_
↪11676101755124723561993227185337871743077799520817686580225233864252891286503981386427236336665397269
```

2. Tìm d "evil"

Với d_e là số nguyên tố, chương trình sinh các số $N_i = p_i \cdot q_i$ với p_i, q_i là các số nguyên tố và tính nghịch đảo e_i của d_e trong modulo $\phi(N_i) = (p_i - 1)(q_i - 1)$.

Ta có $e_i d_e \equiv 1 \pmod{\phi(N_i)}$, tương đương với

$$e_i d_e - 1 = k \phi(N_i) = k_i (N_i - p_i - q_i + 1),$$

hay là

$$k_i(p_i + q_i - 1) - 1 = k_i N_i - e_i d_e.$$

Do N_i, e_i là các số 1024 bit, và d_e là số 333 bit nên suy ra k_i cũng khoảng 333 bit. Như vậy về phải có $1024 + 333$ bit. Trong khi đó, về trái sẽ có $333 + 512$ bit, nhỏ hơn nhiều so với về phải. Điều này dẫn mìn tới lattice.

Mình xây dựng lattice là ma trận sau

$$\begin{bmatrix} e_0 & e_1 & e_2 & 1 & 0 & 0 & 0 \\ N_0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & N_1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & N_2 & 0 & 0 & 0 & 1 \end{bmatrix}$$

thì khi nhân hàng đầu với $-d_e$, nhân hàng hai với k_0 , hàng thứ ba với k_1 và hàng thứ tư với k_2 thì mình có vector ngắn trong lattice là

$$\mathbf{v} = (k_0(p_0 + q_0 - 1) - 1, \dots, \dots, -d_e, k_0, k_1, k_2).$$

Chạy LLL và lấy vị trí thứ ba trong vector ngắn (thêm giá trị tuyệt đối) là mình có d_e rồi.

Nhưng mà khoan, không ra?

Nguyên nhân là vì các số hạng trong vector ngắn \mathbf{v} không có cùng số bit. Ở trên mình đã nói $k_i(p_i + q_i - 1) - 1$ có 333 + 512 bit, trong khi $-d_e$ có 333 bit và k_i cũng có 333 bit. Do đó mình sẽ phải "scale" để chúng có số bit bằng nhau ở vector ngắn.

Quay ngược lên trên lattice, mình nhân thêm hệ số 2^{512} để đảm bảo khi nhân mỗi hàng với $-d_e$ và k_i thì kết quả là vector ngắn \mathbf{v} có số bit như nhau.

Lattice của mình lúc này sẽ là

$$\begin{bmatrix} e_0 & e_1 & e_2 & 2^{512} & 0 & 0 & 0 \\ N_0 & 0 & 0 & 0 & 2^{512} & 0 & 0 \\ 0 & N_1 & 0 & 0 & 0 & 2^{512} & 0 \\ 0 & 0 & N_2 & 0 & 0 & 0 & 2^{512} \end{bmatrix}.$$

Vector ngắn lúc này là

$$\mathbf{v} = (k_0(p_0 + q_0 - 1) - 1, \dots, \dots, -d_e \cdot 2^{512}, k_0 \cdot 2^{512}, k_1 \cdot 2^{512}, k_2 \cdot 2^{512}).$$

Như vậy mỗi phần tử trong \mathbf{v} đều có 512 + 333 bit rồi.

```
pr.recvuntil(b'option:\t')
pr.sendline(json.dumps({"option": "1"}).encode())

Ns = eval(pr.recvline().strip().decode()[3:])
es = eval(pr.recvline().strip().decode()[3:])

print([int(e).bit_length() for e in es])

Mat = Matrix(ZZ, 4, 7)
for i in range(3):
    Mat[0, i] = es[i]
Mat[0, 3] = 2**512

for i in range(3):
    Mat[i+1, i] = Ns[i]
    Mat[i+1, i+4] = 2**512

N = Mat.LLL()

#print(N[0])
```

(continues on next page)

(continued from previous page)

```
d_evil = abs(N[0][3]) // 2**512

assert isPrime(d_evil)

print(f"d_evil = {d_evil}")
```

Output của đoạn code trên là:

```
[1022, 1021, 1022]
d_evil =
→ 9164192793237501841603085694166740261176767425415039425767329941821517155591215118383740675363639829
```

3. CBC Oracle

Tới đây thì dễ thở hơn rồi! Đầu tiên mình encrypt một lần và nhận về ciphertext. Chúng ta sẽ flip phần ciphertext để khi decrypt ra sẽ có đoạn `isadmin: true`.

Giả sử ciphertext gồm các block C_0, C_1, \dots Plaintext gồm các block tương ứng là P_0, P_1, \dots

Giả sử tiếp, đoạn `isadmin: false` nằm trong block P_i . Quá trình giải mã để tìm P_i theo công thức $P_i = C_{i-1} \oplus \text{DEC}(C_i)$.

Do mình muốn P'_i chứa `isadmin: true` (thêm dấu cách ở cuối để số ký tự trước và sau khi đổi bằng nhau). Khi đó mình sẽ đổi

$$C'_{i-1} = C_{i-1} \oplus \text{???isadmin: true ???}.$$

Các block ciphertext còn lại giữ nguyên.

Mình mang ciphertext mới đi decrypt thì sẽ nhận được lỗi do block đầu tiên khi decrypt ra không đúng (khác ciphertext ban đầu nên khi XOR với IV không ra).

```
d = d_evil * 2**333 + d_good

pr.recvuntil(b'option:\t')
pr.sendline(json.dumps({"option": "3", "d": int(d)}).encode())

pr.recvuntil(b'2.sign in')
pr.sendline(json.dumps({"option": "1", "user": "user"}).encode())

data = pr.recvline().strip().decode()
data = data.replace("\\", "\\")\n\n

print(data, len(data))

idx = data.index("False")

ctx = pr.recvline().strip().decode()

iv, ct = ctx[:32], ctx[32:]
iv, ct = bytes.fromhex(iv), list(bytes.fromhex(ct))

tmp = [int(i).__xor__(int(j)) for i, j in zip(b'False', b'True ')]
for i, j in zip(range(idx, idx + len(tmp)), range(len(tmp))):
```

(continues on next page)

(continued from previous page)

```

    ct[i - 16] = int(ct[i - 16]).__xor__(int(tmp[j]))

pr.recvuntil(b'option:\t')
pr.sendline(json.dumps({"option": "3", "d": int(d)}).encode())

pr.recvuntil(b'2.sign in')
pr.sendline(json.dumps({"option": "2", "token": iv.hex() + bytes(ct).hex()}).encode())

test = pr.recvline().strip().decode()

{"id": 190504, "isadmin": False, "username": "user"} 52

```

Do đó mình cũng phải sửa \$IV\$ để khi decrypt C_0 và xor với IV' thì sẽ ra được P_0 ban đầu.

Để ý rằng $P'_0 = IV \oplus \text{DEC}(C'_0)$ nên $P_0 = P'_0 \oplus IV \oplus P_0 \oplus \text{DEC}(C'_0)$, suy ra $IV' = P'_0 \oplus IV \oplus P_0$.

```

assert test[:17] == 'Unpadding Error: '

test = test[19:-1]

def convert(st: str):
    result = []
    i = 0
    while i < len(st):
        if st[i:i+2] != "\\\x":
            result.append(ord(st[i]))
            i += 1
        else:
            result.append(int(st[i+2:i+4], 16))
            i += 4
    return bytes(result)

pti = convert(test) # P_0'
pti = bytes(p_ ^ d_ for p_, d_ in zip(pti, data[:16].encode())) # XOR with P_0
pti = bytes(p_ ^ i_ for p_, i_ in zip(pti, iv)) # XOR with IV

pr.recvuntil(b'option:\t')
pr.sendline(json.dumps({"option": "3", "d": int(d)}).encode())

pr.recvuntil(b'2.sign in')
pr.sendline(json.dumps({"option": "2", "token": pti.hex() + bytes(ct).hex()}).encode())

pr.recvline()
pr.recvline()
pr.recvline()

pr.recvuntil(b"2.Exit\n")
pr.sendline(json.dumps({"option": "1"}).encode())

print(pr.recvline())

pr.close()

```

(continues on next page)

(continued from previous page)

```
# b '0xL4ugh{cryptocats_B3b0_4nd_M1ndfl4y3r}\n'
# b '0xL4ugh{Fak3_Fl@g}\n'
```

Một vấn đề là đoạn `isadmin: false` phải nằm cùng block nếu không sẽ không decrypt ra, việc này may rủi tùy vào x được random như nào. Một vấn đề khác là **bytes trong string** nên mình phải viết hàm extract chuỗi byte đó ra nên không chắc sẽ tránh được lỗi.

Nhưng chạy tới một lúc nào đó sẽ có flag thôi :)))

Writeup của mình đến đây là hết. Cám ơn các bạn đã đọc.

4.6.4 ASIS CTF 2024

Bài này mình làm sau khi giải kết thúc.

Clement

Đề bài ở file `source.py`.

Ở bài này chúng ta cần nhập vào số n . Chương trình sẽ tính $k = n^2 + 4n + 3$.

Hàm `factoreal` sẽ kiểm tra $4 \cdot n! + n + 5 \equiv 0 \pmod{k}$ hay không.

Để giải bài này mình cần định lý Wilson:

❶ Định lý Wilson

Nếu p là số nguyên tố thì

$$(p - 1)! \equiv -1 \pmod{p}$$

Do $k = n^2 + 4n + 3 = (n + 1) \cdot (n + 3)$, mình mong muốn tìm số n thỏa mãn hai phương trình

$$\begin{aligned} 4 \cdot n! + n + 5 &\equiv 0 \pmod{n + 1} \\ 4 \cdot n! + n + 5 &\equiv 0 \pmod{n + 3} \end{aligned}$$

trong đó $n + 1$ và $n + 3$ là các số nguyên tố. Khi đó nghiệm trong modulo k theo CRT sẽ là 0.

Nếu $n + 1$ là số nguyên tố, theo định lý Wilson thì $n! \equiv -1 \pmod{n + 1}$. Như vậy

$$4 \cdot n! + n + 5 \equiv 4 \cdot (-1) + (n + 1) + 4 \equiv -4 + 4 \equiv 0 \pmod{n + 1}$$

Nếu $n + 3$ là số nguyên tố thì $(n + 2)! \equiv -1 \pmod{n + 3}$. Khi đó

$$\begin{aligned} 4 \cdot n! + n + 5 &\equiv 4 \cdot \frac{(n + 3)!}{(n + 2) \cdot (n + 1)} + (n + 3) + 2 \pmod{n + 3} \\ &\equiv 4 \cdot \frac{(-1)}{(n + 2) \cdot (n + 1)} + 2 \equiv 0 \pmod{n + 3}. \end{aligned}$$

Như vậy nếu điều kiện cuối là $4 \cdot \frac{-1}{(n + 2) \cdot (n + 1)} + 2 \equiv 0 \pmod{n + 3}$ thỏa mãn thì mình đã tìm được số n cần thiết.

Đây là bài toán về coding với timeout nên chúng ta có thể lưu các số nguyên tố t bit vào file trước khi remote tới server (precompute).

4.6.5 TJCTF 2024

Một ngày đẹp trời 20/05/20xx ...

Đề bài của CTF này có nhiều câu rất dài nên mình không để ở đây. Các bạn có thể tải đề từ github của cuộc thi tại [đây](#).

weird-crypto

Đây là một bài RSA cơ bản và không may là lúc thi mình không vào được factordb, nhưng oops đủ nhỏ để bruteforce nên cũng không hề gì.

accountleak

Bài này cần tìm `subs` để recover lại `p` và `q`. Tương tự bài trên, `subs` cũng đủ nhỏ để mình bruteforce đến khi nào tìm được cặp `p` và `q` là hai số nguyên tố và decrypt `my_password`.

```
from sage.all import *
from pwn import process, context, remote
import re

context.log_level = 'Debug'

pr = process(["python3", "server.py"])
# pr = remote("tjc.tf", 31601)

pr.recvline()
c, n = list(map(int, re.findall(r'\d+', pr.recvline().strip().decode())))

pr.recvline()
pr.recvline()
pr.sendlineafter(b'<You>', b'yea')
pr.recvline()
dont_leak_this = int(re.findall(r'\d+', pr.recvline().strip().decode())[0])

print(f"c = {c}")
print(f"n = {n}")
print(f"dont_leak_this = {dont_leak_this}")

for sub in range(1, 2**20):
    if (n + sub**2 - dont_leak_this) % sub != 0: continue
    s = (n + sub**2 - dont_leak_this) // sub
    PR = PolynomialRing(ZZ, 'x')
    x = PR.gen()
    f = x**2 - s*x + n
    roots = f.roots()
    if len(roots) == 0: continue
    p, q = roots[0][0], roots[1][0]
    assert p * q == n
    d = pow(65537, -1, (p-1) * (q-1))
    my_password = pow(c, d, n)
    pr.recvline()
    pr.recvline()
    pr.sendlineafter(b"<You>", str(my_password).encode())
    pr.recvline()
```

(continues on next page)

(continued from previous page)

```

pr.recvline()
print(pr.recvline())

pr.close()

```

assume

Ở bài này mình chỉ cần kiểm tra xem if hay else được thực thi bằng việc kiểm tra message thứ 5 của mỗi đoạn có bằng $(g^a)^b$ không.

```

with open("log.txt") as f:
    lines = [line.strip() for line in f.readlines()]

p = int(lines[0])
msg = lines[1:]
g = 6

flag = ""
pos = 0

for ln in range(0, len(msg), 6 * 20):
    for i in range(ln, ln + 6 * 20, 6):
        ga = int(msg[i].split(" ")[-1])
        b = int(msg[i+1].split(" ")[-1])
        gb = int(msg[i+2].split(" ")[-1])
        c = msg[i+3].split(" ")[-1]
        guess = int(msg[i+4].split(" ")[-1])
        if pow(ga, b, p) == guess:
            flag += c
            pos += 1
            break
    print(flag)

print(flag)

```

iodomethane

Ở bài này, nếu gọi K là khóa thì K là ma trận 3×3 . Giả sử flag gồm các bytes $f_0 f_1 \dots f_n$ thì kết quả output là phép nhân ma trận

$$\begin{pmatrix} f_0 & f_1 & f_2 \\ f_3 & f_4 & f_5 \\ \ddots & & \end{pmatrix} \cdot K = \begin{pmatrix} e_0 & e_1 & e_2 \\ e_3 & e_4 & e_5 \\ \ddots & & \end{pmatrix}.$$

Với flag format là `tjctf{`, mình đã biết 6 ký tự đầu của flag, mình chỉ cần bruteforce thêm 3 ký tự nữa là có thể kết hợp với output để tìm K . Ở đây K cần thỏa mãn là ma trận khả nghịch và khi decrypt mỗi dòng của flag (trừ dòng cuối) phải cho kết quả nằm trong alphabet.

Code giải mình để ở `solve.py`.

alkane

Bài này mình giải ra sau cùng do một số nhầm lẫn.

Giả sử gọi dòng i của $schedule$ là $S^{(i)} = (s_0^{(i)}, s_1^{(i)}, \dots, s_{63}^{(i)})$.

Gọi key là $K = k_0k_1\dots k_{128}$. Khi đó mảng arr sẽ được tạo bằng cách

$$arr_i = k_{s_0^{(i)}} \oplus k_{s_1^{(i)}} \oplus \dots \oplus k_{s_{63}^{(i)}}.$$

Ý tưởng của mình là xem phép tính trên như một tích vô hướng của hai vector $\bar{a}_i = (a_0, \dots, a_{127})$ và K , với $a_j = 1$ nếu j nằm trong $S^{(i)}$, ngược lại thì $a_j = 0$.

Khi mình xây dựng ma trận M gồm các dòng là các vector \bar{a}_i như trên thì mình không decrypt ra được. Mãi về sau mình mới phát hiện rằng trong $S^{(i)}$ có một số phần tử trùng nhau chứ không hoàn toàn là 64 phần tử phân biệt.

Khi đó mình cần xem số lần mỗi phần tử xuất hiện trong $S^{(i)}$ là chẵn hay lẻ.

Một vấn đề khác là ma trận M có rank bằng 127, không phải full rank. Do đó để tìm tất cả nghiệm của hệ phương trình mình cần cộng với tất cả tổ hợp tuyến tính của các vector trong kernel.

```
from sage.all import *

m = b'yellow submarine'
c = b'\xb7\x8e\xb3\xd9\xfd\xf2\x1f\xa2\xeaz\xe3\x0f\x00\xj\x08'

assert len(m) == len(c)

def xor(a, b):
    return bytes(x^y for x, y in zip(a, b))

def get_bit(b, n):
    byte = b[n // 8]
    return (byte >> (7 - (n % 8))) & 1

def set_bit(b, n, v):
    byte = b[n // 8]
    byte &= ~(v << (7 - (n % 8)))
    byte |= v << (7 - (n % 8))
    b[n // 8] = byte
    return b

def rrot(x1, n):
    arr = bytearray(x1)
    for i in range(len(x1)):
        arr[(i + n) % len(x1)] = x1[i]
    return arr

def lrot(x1, n):
    arr = bytearray(x1)
    for i in range(len(x1)):
        arr[i] = x1[(i + n) % len(x1)]
    return arr

out = xor(m, c)
```

(continues on next page)

(continued from previous page)

```

M = matrix(GF(2), 128, 128)

from challenge import schedule

for i in range(128):
    for bit_loc in schedule[i]:
        M[i, bit_loc] += 1

arr = vector(GF(2), [get_bit(out, i) for i in range(128)])

try:
    sol = M.solve_right(arr)
except: exit(0)

ker = M.right_kernel()

assert len(ker.basis()) == 1

sol_ = sol + ker.basis()[0]
assert M * sol == arr and M * sol_ == arr

key = bytearray(b"\x00" * 16)
for i in range(128):
    set_bit(key, i, int(sol[i]))

print(b"tjctf{" + key + b"}")

key_ = bytearray(b"\x00" * 16)
for i in range(128):
    set_bit(key_, i, int(sol_[i]))

print(b"tjctf{" + key_ + b"}")

```

lightweight-crypto-guard-system

Đây là một bài LCG cũng không quá phức tạp.

Với seed ban đầu x_0 , chuỗi sẽ được sinh theo công thức $x_{i+1} = (ax_i + b) \text{ mod } m$. Ở đây cả a , b và m đều không biết.

Có nhiều hướng dẫn mình tham khảo để tìm lại modulo m và a , b (mình lười :v). Ví dụ như ở [link](#).

Nhưng vẫn đề ở bài này là chúng ta được cho một đoạn liên tục x_0, x_1, x_2, \dots mà thay vào đó là x_1, x_{1+n}, x_{2n+1} (tổng có 6 số).

Thực ra cách giải vẫn giống nhau. Đầu tiên mình recover modulo m . Sau đó để ý rằng

$$\begin{aligned}
 x_1 &= (ax_0 + b) \text{ mod } m \\
 x_2 &= (ax_1 + b) \text{ mod } m = (a \cdot (ax_0 + b) + b) = (a^2x_0 + ab + b) \text{ mod } m \\
 x_3 &= (ax_2 + b) \text{ mod } m = \dots = a^3x_0 + a^2b + ab + b = a^3x_0 + \frac{a^3 - 1}{a - 1} \cdot b \text{ mod } m.
 \end{aligned}$$

Cứ tương tự vậy mình có $x_{i+n} = (Ax_n + B) \text{ mod } m$,

với $A = a^n \bmod m$ và $B = \frac{a^n - 1}{a - 1} \cdot b \bmod m$.

Như vậy mình có một LCG mới và giải ra được A, B .

Sau đó mình recover lại a và b . Như vậy là mình sinh được toàn bộ dãy.

hulksmash

Với hint là file `keysmashes.txt` thì các dòng trong file sẽ giúp mình tìm key.

Để ý rằng mỗi dòng chỉ gồm các ký tự `fjdlska;` (8 ký tự, mỗi ký tự xuất hiện hai lần).

Để ý thêm xíu nữa, mình phát hiện ra mỗi ký tự xuất hiện hai lần vào vị trí hoặc đều chẵn, hoặc đều lẻ. Cụ thể là các ký tự `fsda` sẽ ở vị trí chẵn còn `jlk`; ở vị trí lẻ.

Như vậy mình bruteforce tất cả tổ hợp như vậy và thử decrypt ...

```
from Crypto.Cipher import AES
from itertools import combinations

with open("keysmashes.txt") as rf:
    lines = [line.strip() for line in rf.readlines()]

alphabet = list(b"fjdlska;")
assert len(alphabet) == 8

ct = bytes.fromhex(
    "ed05f1440f3ae5309a3125a91dfb0edef306e1a64d1c5f7d8cea88cdb7a0d7d66bb36860082a291138b48c5a6344c1ab"
    "1")

odd = set([1, 3, 5, 7, 9, 11, 13, 15])
even = set([0, 2, 4, 6, 8, 10, 12, 14])

for x1, x2 in combinations(odd, 2):
    ox = odd.difference([x1, x2])
    for y1, y2 in combinations(ox, 2):
        oy = ox.difference([y1, y2])
        for z1, z2 in combinations(oy, 2):
            t1, t2 = oy.difference([z1, z2])
            assert set([x1, x2, y1, y2, z1, z2, t1, t2]) == odd
            for a1, a2 in combinations(even, 2):
                ea = even.difference([a1, a2])
                for b1, b2 in combinations(ea, 2):
                    eb = ea.difference([b1, b2])
                    for c1, c2 in combinations(eb, 2):
                        d1, d2 = eb.difference([c1, c2])
                        key = [0] * 16
                        key[x1] = key[x2] = ord('j')
                        key[y1] = key[y2] = ord('l')
                        key[z1] = key[z2] = ord('k')
                        key[t1] = key[t2] = ord(';')

                        key[a1] = key[a2] = ord('f')
                        key[b1] = key[b2] = ord('d')
                        key[c1] = key[c2] = ord('s')
                        key[d1] = key[d2] = ord('a')
```

(continues on next page)

(continued from previous page)

```

aes = AES.new(bytes(key), AES.MODE_ECB)
try:
    print(aes.decrypt(ct).decode())
except:
    pass

```

Tiếp theo mình sẽ trình bày các bài mình làm sau khi giải kết thúc (có tham khảo writeup).

c-8

Đề bài cho mình các file `enc.py`, `dec.py`, `re_plaus`, `plausibly.deniable`. Với hint là affine cipher và modulo $n = 18446744073709551629$ mình cần khôi phục lại bốn file trên.

Do n có 65 bit nên việc mã hóa theo mã affine sẽ mã hóa 8 bytes một lần ra 9 bytes của ciphertext. Các bạn cũng có thể thấy rằng byte đầu của mỗi đoạn 9 bytes các file trên là 0x00 hoặc 0x01.

Mã affine có dạng $y = ax + b \bmod n$ với a và b là hai số chưa biết cần đi tìm. Mình đã có y và cần dự đoán x nào sẽ mã hóa ra y tương ứng. Đoạn đầu của các file mã hóa với Python thường sẽ dùng các thư viện như `pycryptodome`. Mình thử một loạt các kiểu import và tìm ra phần đầu sẽ là

```
from Crypto.Cipher import AES
```

Mình chỉ cần 16 bytes ở trên thôi là được. Dựa vào đó mình khôi phục lại a và b rồi decrypt bốn file ban đầu.

u-235

Lát viết

titanium-isopropoxide

Bài này mã hóa bằng file C++ với thuật toán khá phức tạp.

Bug ở bài này là key được tạo ra từ S-box không đủ mạnh. Sử dụng known-plaintext là flag format `tjctf{` và XOR với 6 bytes đầu của ciphertext của hai file mình:

```

flag2 = open("flag2.txt.enc", "rb").read()
flag3 = open("flag3.txt.enc", "rb").read()

def xor(a, b):
    return bytes(x ^ y for x, y in zip(a, b))

known_plaintext = b"tjctf{"

key = xor(flag2, known_plaintext)

f2 = []
f3 = []
for i in range(len(flag2)):
    f2.append(key[i % len(key)] ^ flag2[i])
    f3.append(key[i % len(key)] ^ flag3[i])

print(bytes(f2))
print(bytes(f3))

```

Ở kết quả thu được từ đoạn code, mình thấy rằng *flag2* khá giống với đoạn văn có nghĩa. Sau dấu `{` là từ `this` (vì có sẵn chữ `is`), sau dấu `_` là `cube`. Như vậy, mình XOR đoạn `tjctf{this_cube` thì sẽ thấy được key:

```
b'<U\xed\x18\xed\x18\xed\x18\xed\x18\xed\x18\xed\x18\xf7\x1e\xed'
```

Mình thấy rằng key bắt đầu bởi `<U` và lặp lại cặp byte `\xed\x18`. Do đó mình có thể decrypt ra flag.

```
flag2 = open("flag2.txt.enc", "rb").read()

def xor(a, b):
    return bytes(x^y for x, y in zip(a, b))

known_plaintext = b"tjctf{"

key = xor(flag2, known_plaintext)

while len(key) != 70:
    key += b"\xed\x18"

f2 = []
for i in range(len(flag2)):
    f2.append(key[i % len(key)] ^ flag2[i])

print(bytes(f2))
```

lithium-stearate

Sau khi giải kết thúc một pro đã hint cho mình: Slide attack.

Bài này mình được cho 20 cặp plaintext-ciphertext cùng với ciphertext của flag. Thông tin về cipher:

- độ dài block là 16 bits (2 bytes);
- độ dài khóa là 64 bits (8 bytes);
- số vòng thực hiện là 100000 vòng.

Hàm mã hóa:

```
Word encrypt(Word in, Key k)
{
    Word out = in;
    for (int i = 0; i < rounds; i++)
    {
        out = r(out, ksch(k, i));
    }
    return out;
}
```

Trong đó `rounds = 100000` và `ksch` là key schedule sinh ra subkey cho mỗi vòng.

round function

Round function là hàm r với cấu trúc như sau:

```
Word r(Word in, RKey kin)
{
    RKey k = kin;
    in ^= (k & 0xffff) ^ ((k >> 16) & 0xffff);
    return P_f(S_f(in));
}
```

Trong đó, P_f và S_f là hai hàm:

```
Word S_f(Word in)
{
    return (Sbox[in >> 8] << 8) | Sbox[in & 0xff];
}

Word P_f(Word in)
{
    int r = 3;

    Word out = 0;
    out |= (in >> 8) & 0xff;
    out |= (in & 0xff) << 8;
    out ^= out >> 8;

    Word out2 = 0;
    out2 |= out >> r;
    out2 |= out << (16 - r);

    return out2;
}
```

Như vậy, S_f là hàm sử dụng S-box cho trước, và P_f là hàm hoán vị. Mình kí hiệu hai hàm này là S_f và P_f .

Như vậy round function có dạng $out = P_f(S_f(out \oplus k))$, trong đó r là số 16 bits và k cũng là số 16 bits. Ở đây lưu ý rằng tham số thứ hai của r là số 32 bits nhưng thực chất là lấy nửa đầu XOR nửa cuối, như vậy key được dùng có 16 bits.

key schedule

Thuật toán sinh khóa ksch lấy đầu vào là khóa ban đầu 8 bytes và vòng hiện tại i để sinh ra khóa k_i . Khi thử in ra tất cả 1000000 khóa khi mã hóa một plaintext bất kì thì chúng ta có thể thấy rằng **chỉ có đúng bốn subkey xoay vòng**.

Kết hợp với lúc nãy mình chỉ ra, mỗi round function lấy vào key 32 bits nhưng thực chất chỉ là 16 bits. Như vậy nếu bruteforce cả bốn vòng với bốn subkeys thì sẽ cần $16 \times 4 = 64$ bits. À thì nó cũng chả khác việc key ban đầu có 64 bits :))) nên chúng ta sẽ không bruteforce kiểu này.

Chúng ta sẽ xem kỹ hơn hàm sinh khóa con ksch.

```
RKey ksch(Key k, int i)
{
```

(continues on next page)

(continued from previous page)

```
// .....
k |= 0xff << 24;
k |= 0xffULL << (24 + 32);
// .....
}
```

Đoạn trên cho thấy rằng, sau một hồi biến đổi, hàm `ksch` sẽ set bit thứ 3 và thứ 7 của biến `k` thành `0xff`, nghĩa là `k` lúc này có dạng `0xff-----ff-----` (64 bits).

```
RKey ksch(Key k, int i)
{
    // .....
    if (i == 0)
    {
        r = k & 0xffffffff;
        r ^= 1162466901;
        r ^= r >> 16;
        r *= 3726821653;
    }
    if (i == 1)
    {
        r = ((k >> 32) & 0xffffffff) ^ (k & 0xffffffff);
        r ^= 3811777446;
        r = (r * 1240568533);
    }
    if (i == 2)
    {
        r = ((k >> 32) & 0xffffffff) ^ (k & 0xffffffff);
        r ^= 3915669785;
        r = (r * 1247778533);
    }
    if (i == 3)
    {
        r = ((k >> 32) & 0xffffffff) ^ (k & 0xffffffff);
        r ^= 3140176925;
        r = (r * 1934965865);
    }

    return r;
}
```

Ở phần kế của hàm `ksch` có 3/4 trường hợp `r` là XOR của 32 bits cao và 32 bits thấp. Ở trên mình đã phân tích rằng bytes thứ 3 và thứ 7 của `k` sẽ là `0xff` nên khi XOR như vậy sẽ triệt tiêu byte đầu tiên của `r`, nên `r` chỉ còn 24 bits và chúng ta sẽ bruteforce ở đây.

Trong trường hợp `i = 0` thì chương trình cũng chỉ lấy 32 bits thấp nên chúng ta có thể bỏ qua. Lý do sẽ được trình bày ở phần sau.

slide attack

Giới thiệu qua loa: slide attack là phương pháp tấn công block cipher thông qua việc chọn các cặp plaintext-ciphertext (P, C) và (P', C') mà $P' = F(P)$ và $C' = F(C)$ với $F(x)$ là hàm nào đó. Khi thỏa mãn các điều kiện trên thì ta có thể thực hiện slide attack.

Để tìm hàm $F(x)$ như vậy thì phụ thuộc vào cipher nào. Thông thường hàm $F(x)$ sẽ chứa một hoặc nhiều round, trong đó chứa tất cả khóa con của cipher.

Ví dụ. Xét mô hình cipher đơn giản sau:

$$P \xrightarrow{K_0} O_1 \xrightarrow{K_1} O_2 \xrightarrow{K_0} O_3 \xrightarrow{K_1} O_3 = C.$$

Như vậy hàm $F(x)$ ở đây sẽ chứa hai subkey là K_0 và K_1 . Do đó sơ đồ của F sẽ là:

$$P \xrightarrow{F} O_2 \xrightarrow{F} C.$$

Giả sử với một cặp plaintext-ciphertext khác là:

$$P' \xrightarrow{K_0} O_1 \xrightarrow{K_1} O_2 \xrightarrow{K_0} O_3 \xrightarrow{K_1} O_3 = C'.$$

hay tương đương là

$$P' \xrightarrow{F} O'_2 \xrightarrow{F} C'.$$

Slide attack lúc này sẽ hoạt động nếu $P' = O_2 = F(P)$ và $C' = F(C)$. Khi vẽ ra sẽ có dạng:

$$\begin{array}{ccc} P \xrightarrow{F} & O_2 \xrightarrow{F} & C \\ P' \xrightarrow{F} & O'_2 \xrightarrow{F} & C' \end{array}$$

Quay lại bài lithium-stearate này, do chỉ có bốn subkeys nên có thể xem hàm F ở trên có dạng:

$$F(P, K) = G(G(G(G(P, K_0), K_1), K_2), K_3) = P'.$$

Trong đó K là khóa ban đầu 64 bits và mỗi K_i là khóa con 32 bits. Hàm $G(x, k) = P_f(S_f(x \oplus k))$ là round function.

Như vậy chiến thuật để giải bài này là:

- Tìm trong 20 cặp plaintext-ciphertext để cho các cặp (P, C) và (P', C') sao cho $F(P, K) = P'$ và $F(C, K) = C'$.
- Từ hai cặp plaintext-ciphertext trên khôi phục khóa K .

Tuy nhiên việc này khó khăn vì không có khóa K thì không kiểm tra được điều kiện $F(P, K) = P'$. Do đó chúng ta sẽ làm ngược lại.

Như ở trên đã phân tích, K_1, K_2 và K_3 được sinh ra từ một k có 24 bits, do đó chúng ta bruteforce các k như vậy và tính ra K_1, K_2 và K_3 tương ứng.

Với mỗi hai plaintext P_i và P_j ta tính $K_0 = S_f^{-1}(P^{-1}(T)) \oplus P$, với

$$T = G^{-1}(G^{-1}(G^{-1}(P', K_3), K_2), K_1)..$$

Khi đã có đủ K_0, K_1, K_2 và K_3 , mình sẽ kiểm tra điều kiện $C' = F(C, K)$. Nếu thỏa mãn thì mình có được một cặp **slide pair** và sẽ thử mã hóa hai cặp plaintext-ciphertext bất kì để xem việc mã hóa với khóa có đúng không.

Khi tìm được khóa rồi thì mình giải mã lại flag.

Sau khi chạy `findkey.cpp` thì mình tìm được key là 11892, 8704, 3384, 38922. Sau đó mình decrypt ra flag. Ở đây mình compile với flag `-O3` và `openmp` vì mình không có nhiều kinh nghiệm dùng GCC lắm. :))) Code cũng sẽ chạy nhanh hơn nếu sử dụng các flag khác, cũng như C++ khác như 11, ...

Lệnh compile:

```
g++ -O3 findkey.cpp -o findkey -fopenmp
```

Cuối cùng chúng ta giải mã với file `decrypt.cpp`.

Mình dành hẳn một phần cho một bài vì bài này khá dài và phức tạp.

Đây cũng là bài viết đầu tiên của mình về một phương pháp tấn công cực kì phổ biến là differential attack (phá mã vi sai).

tetraethyllead

Đây là một cipher sử dụng mô hình Feistel để mã hóa. Mô hình Feistel thường xuyên bị tấn công theo phương pháp differential, có thể kể đến như: DES, GOST, hay các phiên bản nhỏ hơn của chúng.

Do đó yêu cầu chống lại differential attack (cũng như người anh em khác của nó là linear attack) trở thành tiêu chuẩn để xây dựng block cipher hiện nay.

Mô hình Feistel

Trong mô hình Feistel, mỗi block của plaintext sẽ được chia đôi thành hai nửa trái phải - $P = L_0 \| R_0$.

Ở bài này, plaintext có 8 bytes và key cũng có 8 bytes.

Trong mô hình Feistel chuẩn, ở mỗi vòng $i + 1$, sẽ thực hiện biến đổi sau:

$$L_{i+1} = R_i, \quad R_{i+1} = L_i \oplus F(R_i, K_{i+1}).$$

Trong đó:

- K_{i+1} là khóa ở vòng $i + 1$ với $i = 0, 1, \dots$
- F được gọi là round function. Hàm F phụ thuộc vào cipher là loại nào. Ví dụ thuật toán DES thì F là các phép biến đổi Expand, P-box và S-box. Hoặc đối với thuật toán GOST thì F gồm cộng modulo 2^{32} , S-box và dịch 11 bit sang trái.

Ở mô hình Feistel bên trên (cũng là mô hình chuẩn) thì hàm F cố định cho mỗi vòng. Tuy nhiên ở bài này thì round function ở mỗi vòng khác nhau. Cụ thể thì ở vòng một dùng S-box r_2 , ở vòng thứ hai là S-box r_1 , các vòng sau thì dùng hàm r_{345} .

Mình có thể viết quá trình biến đổi như sau:

Vòng 1	$L_1 = R_0$	$R_1 = L_0 \oplus r_2(R_0, S_2)$
Vòng 2	$L_2 = R_1$	$R_2 = L_1 \oplus r_1(R_1, S_1)$
Vòng 3	$L_3 = R_2$	$R_3 = L_2 \oplus r_{345}(R_2, k_2, 3)$
Vòng 4	$L_4 = R_3$	$R_4 = L_3 \oplus r_{345}(R_3, k_2, 4)$
Vòng 5	$L_5 = R_4$	$R_5 = L_4 \oplus r_{345}(R_4, k_1 \oplus k_2, 5)$
Vòng 6	$L_6 = R_5$	$R_6 = L_5 \oplus r_{345}(R_5, k_1, 6)$
Vòng 7	$L_7 = L_6 \oplus r_{345}(R_6, k_2, 7)$	$R_7 = R_6 \oplus L_7 = R_6 \oplus r_{345}(R_6, k_2, 7)$

Lưu ý rằng vòng cuối hơi khác một tí.

Differential attack

Differential là gì???

Định nghĩa. Xét hàm S từ \mathbb{F}_2^n tới \mathbb{F}_2^m . Với mỗi cặp vector $a, b \in \mathbb{F}_2^n$ thì ta nói $a \oplus b$ là **input differential** và $S(a) \oplus S(b)$ là **output differential** ứng với hàm S .

Hàm S thường là các S-box trong block cipher. Các S-box thường không tuyến tính, nghĩa là ta không có $S(\mathbf{a} \oplus \mathbf{b}) = S(\mathbf{a}) \oplus S(\mathbf{b})$.

Differential dựa trên quan sát rằng khi $\mathbf{a} \oplus \mathbf{b}$ cố định thì output differential $S(\mathbf{a}) \oplus S(\mathbf{b})$ phân bố không đều. Giả sử mình có S-box như sau:

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S(x)$	3	14	1	10	4	9	5	6	8	11	15	2	13	12	0	7

Nếu mình duyệt qua tất cả cặp $\mathbf{a}, \mathbf{b} \in \mathbb{F}_2^4$ thì mình sẽ có quan sát sau:

- Nếu input vi sai là 0 thì output vi sai là 0 với xác suất 1.
- Nếu input vi sai là 1 thì output vi sai là 13 với xác suất 6/16.
- Nếu input vi sai là 4 thì output vi sai là 7 với xác suất 6/16.
- Nếu input vi sai là 8 thì output vi sai là 5 với xác suất 6/16.
- Nếu input vi sai là 15 thì output vi sai là 14 với xác suất 6/16.

Đây là những output differential với **xác suất cao nhất** ứng với mỗi input differential cố định $\mathbf{a} \oplus \mathbf{b}$.

Điều đó có nghĩa là cứ trung bình 16 cặp $\mathbf{a} \oplus \mathbf{b}$ cho kết quả là 1 thì có 6 cặp cho output differential là 13.

Tận dụng điều này, chúng ta sẽ giải bài tetraethyllead.

Giả sử (P, C) và (P', C') là hai cặp plaintext-ciphertext sao cho khi input differential $P \oplus P'$ cố định thì xác suất xảy ra của output differential $C \oplus C'$ cao nhất. Khi tìm được input và output differential như vậy, ta mã hóa nhiều plaintext khác nhau và khả năng (xác suất) xuất hiện ciphertext tương ứng thỏa mãn differential đó sẽ cao.

Giả sử $P = (L_0, R_0)$ và $P' = (L'_0, R'_0)$ là hai nửa trái phải tương ứng với P và P' .

Đặt $\Delta L_{in} = L_0 \oplus L'_0$ và $\Delta R_{in} = R_0 \oplus R'_0$ là input differential trước khi mã hóa.

Đặt $\Delta L_{out} = L_7 \oplus L'_7$ và $\Delta_{in} = R_7 \oplus R'_7$ là output differential sau khi mã hóa.

Chúng ta sẽ đi qua từng hàm để xem với input-output differential nào thì khả năng xảy ra là cao nhất.

Differential vòng 1

Sau vòng 1 ta có:

- $L_1 = R_0$ và $R_1 = L_0 \oplus r_2(R_0, S_2)$.
- $L'_1 = R'_0$ và $R'_1 = L'_0 \oplus r_2(R'_0, S_2)$.

Khi đó differential ở vòng 1 là:

$$\begin{aligned}\Delta L_1 &= L_1 \oplus L'_1 = R_0 \oplus R'_0 \\ \Delta R_1 &= R_1 \oplus R'_1 = L_0 \oplus L'_0 \oplus r_2(R_0, S_2) \oplus r_2(R'_0, S_2).\end{aligned}$$

Như vậy ΔR_1 phụ thuộc vào vi sai của hàm r_2 .

```
def r2(i, box):
    out = []
    i = long_to_bytes(i)

    for byte in i:
        out.append(box[byte])
```

(continues on next page)

(continued from previous page)

```

for idx in range(len(out) - 2, -1, -1):
    out[idx] ^= out[idx + 1]

return bytes_to_long(b"".join([long_to_bytes(l) for l in out]))

```

Giả sử bốn bytes đầu vào của r_2 là $a\|b\|c\|d$. Tiếp theo ta thay thế bốn bytes này bởi giá trị S-box của nó là $S_2(a)\|S_2(b)\|S_2(c)\|S_2(d)$. Vòng lặp thứ hai XOR chồng các bytes lên nhau để có kết quả:

$$S_2(a) \oplus S_2(b) \oplus S_2(c) \oplus S_2(d) \| S_2(b) \oplus S_2(c) \oplus S_2(d) \| S_2(c) \oplus S_2(d) \| S_2(d).$$

Như vậy nếu $R_0 = a\|b\|c\|d$ và $R'_0 = a'\|b'\|c'\|d'$ thì:

$$\begin{aligned} r_2(R_0, S_2) \oplus r_2(R'_0, S_2) &= S_2(a) \oplus S_2(b) \oplus S_2(c) \oplus S_2(d) \oplus S_2(a') \oplus S_2(b') \oplus S_2(c') \oplus S_2(d') \\ &\quad \| S_2(b) \oplus S_2(c) \oplus S_2(d') \oplus S_2(b') \oplus S_2(c') \oplus S_2(d') \\ &\quad \| S_2(c) \oplus S_2(d) \oplus S_2(c') \oplus S_2(d') \\ &\quad \| S_2(d) \oplus S_2(d') \end{aligned}$$

Dễ thấy rằng nếu chúng ta chọn $d = d'$ thì byte cuối triệt tiêu.

Tiếp theo chọn $c = c'$ thì byte kế cuối cũng triệt tiêu.

Tiếp theo chọn $b = b'$ thì byte thứ ba (từ phải sang trái) cũng triệt tiêu.

Khi đó vi sai sẽ là

$$r_2(R_0, S_2) \oplus r_2(R'_0, S_2) = S_2(a) \oplus S_2(a') \| 00 \| 00 \| 00.$$

Bằng một cách *ảo ma* nào đó thì output differential $S_2(a) \oplus S_2(a') = 0x80$ có xác suất xảy ra cao nhất khi input differential là $0x80$. Cái này writeup nói nhưng chúng ta cũng có thể kiểm tra phân bố vi sai của r_2 .

Đi ngược lên trên thì $a\|b\|c\|d \oplus a'\|b'\|c'\|d' = 80\|00\|00\|00$. Đây chính là vi sai cho $R_0 \oplus R'_0$.

Như vậy $\Delta L_1 = 0x80000000$ (với xác suất là 1) và $\Delta R_1 = L_0 \oplus L'_0 \oplus 0x80000000$ (với xác suất cao).

Differential vòng 2

Tương tự, đối với r_1 chúng ta cũng dùng phương pháp tương tự.

Sau vòng 2 ta có:

- $L_2 = R_1$ và $R_2 = L_1 \oplus r_1(R_1, S_1)$;
- $L'_2 = R'_1$ và $R'_2 = L'_1 \oplus r_1(R'_1, S_1)$.

Khi đó differential ở vòng 2 là:

$$\Delta L_2 = L_2 \oplus L'_2 = R_1 \oplus R'_1 = L_0 \oplus L'_0 \oplus 0x80000000.$$

với xác suất cao.

Tương tự

$$\Delta R_2 = R_2 \oplus R'_2 = L_1 \oplus L'_1 \oplus r_1(R_1, S_1) \oplus r_1(R'_1, S_1) = 0x80000000 \oplus r_1(R_1, S_1) \oplus r_1(R'_1, S_1).$$

Khi này, $R_1 \oplus R'_1$ rất khó kiểm soát nên chúng ta sẽ khiến $R_1 = R'_1$. Khi đó $r_1(R_1, S_1) \oplus r_1(R'_1, S_1) = 00$ với xác suất bằng 1.

Nếu $R_1 = R'_1$ thì quay ngược lên trên, $\Delta R_1 = 00 = L_0 \oplus L'_0 \oplus 0x80000000$ nên $L_0 \oplus L'_0 = 0x80000000$.

Như vậy mình đã tìm được input differential cho cả hàm mã hóa là $L_0 \oplus L'_0 = 0x80000000$ và $R_0 \oplus R'_0 = 0x80000000$.

Lúc này, differential ở vòng 2 có xác suất xảy ra cao nhất là $L_2 \oplus L'_2 = 0x00000000$ và $R_2 \oplus R'_2 = 0x80000000$.

Tiếp theo, mình cần biết output differential nào của toàn bộ hàm `encrypt` sẽ có khả năng xảy ra nhất đối với input differential này.

Hàm z345

```
def r345(word, k, rnum):
    word ^= rrot(word, -463 + 439 * rnum + -144 * rnum**2 + 20 * rnum**3 - rnum**4) ^
    ↪lrot(word, 63 + -43 * rnum + 12 * rnum**2 + -rnum**3)

    word = (4124669716 + word * bytes_to_long(k))**3

    word ^= word << 5
    word ^= word << 5

    word ^= rrot(word, -463 + 439 * rnum + -144 * rnum**2 + 20 * rnum**3 - rnum**4) ^
    ↪lrot(word, 63 + -43 * rnum + 12 * rnum**2 + -rnum**3)

    return rrot(word, -504 + 418 * rnum -499 * rnum**2 + -511 * rnum**3 + 98 * rnum**4) &
    ↪ 0xffffffff
```

Khi thay các giá trị `rnum` vào thì mình thấy có 3 vòng `rrot` và `lrot` ngược nhau (`rrot(word, 17)` và `lrot(word, 15)` trên 32 bit) nên ở những vòng này `rrot` và `lrot` sẽ triệt tiêu nhau do đà thúc.

Tiếp theo, `word ^= word << 5` sẽ không làm thay đổi differential `0x80`. Lấy ví dụ với 8 bit $\bar{a} = a_0a_1a_2a_3a_4a_5a_6a_7$ và $\bar{b} = b_0b_1b_2b_3b_4b_5b_6b_7$. Giả sử $\bar{a} \oplus \bar{b} = 0x80$ thì:

$$\begin{aligned} \bar{a} \oplus (\bar{a} << 5) \oplus \bar{b} \oplus (\bar{b} << 5) &= a_0a_1a_2a_3a_4a_5a_6a_7 \oplus a_5a_6a_700000 \\ &\quad \oplus b_0b_1b_2b_3b_4b_5b_6b_7 \oplus b_5b_6b_700000 \\ &= \underbrace{(a_0a_1a_2a_3a_4a_5a_6a_7 \oplus b_0b_1b_2b_3b_4b_5b_6b_7)}_{0x80} \\ &\quad \oplus \underbrace{(a_5a_6a_700000 \oplus b_5b_6b_700000)}_{0x00} = 0x80. \end{aligned}$$

Cuối cùng, phép nhân modulo 2^{32} có 50% duy trì differential `0x80000000` và do đó differential này vẫn có xác suất cao cho cả `z345`.

Giải

Chiến thuật để giải bài này là:

- Gửi lên các cặp plaintext sao cho $\Delta L_{in} = 0x80000000$ và $\Delta R_{in} = 0x80000000$.
- Nhận các cặp ciphertext tương ứng thỏa mãn $\Delta L_{out} \oplus R_{out} = 0x80000000$.

Các cặp plaintext, ciphertext như trên sẽ giúp ta khôi phục lại khóa.

Note. Để thuận tiện thì mình sẽ cố định khóa trong `server.py` và chỉ cần tìm ra khóa là xong. Ở trong giải thì sau khi request đủ 1024 lần encrypt server cũng không đặt timeout nên chúng ta có thể để script chạy bao lâu tùy ý.

File `solve.py` sau đây tương tác với `server.py` để lấy về các cặp plaintext, ciphertext thỏa mãn điều kiện trên.

Sau đó sử dụng thư viện SHA256 từ project `SHA256` (gồm file `SHA256.h` và `SHA256.cpp`).

Chúng ta cần viết `solve.h` chứa định nghĩa các hàm sẽ dùng để tìm khóa.

Đầu tiên chúng ta sẽ khôi phục `k2` đi ngược từ ciphertext lên. Chúng ta nhận các khóa thỏa mãn $\Delta L_6 \oplus \Delta R_6 = 0x80000000$ (vòng 6). Để tìm `k2` thì hàm `main` sẽ là:

```
int main()
{
    std::vector<uint32_t> key2 = bruteforce_k2();
    for (int i = 0; i < key2.size(); i++)
        std::cout << std::hex << key2[i] << std::endl;
    return 0;
}
```

Sau đó với mỗi `k2` chúng ta bruteforce các `k1` và kiểm tra `k = k1 || k2` nào sẽ encrypt đúng. Ở đây do mình đã cố định khóa ở `server.py` nên mình biết `k2` nào là đúng để tiết kiệm thời gian viết lại writeup. Sau đó thì chỉ cần bruteforce `k1` thôi.

```
int main()
{
    uint8_t k2[] = { 0xef, 0x13, 0x37, 0xff };
    get_sbox(k2, sbox2);

    uint64_t p = 2194090266659289430ULL;
    uint64_t c = 5313144679078469543ULL;

    #pragma omp for
    for (uint64_t k1 = 0; k1 < 0xffffffff; k1++)
    {
        uint64_t key = (k1 << 32) | 0xef1337ff;
        if (encrypt(p, key) == c)
        {
            std::cout << "Found key: " << std::hex << key << std::endl;
        }
    }
    return 0;
}
```

Cám ơn các bạn đã đọc writeup dài lê thê của mình.

4.6.6 CryptoFox 2024

Olympiad mật mã và bảo mật thông tin CryptoFox 2024

Задача 1. дешифрование файла

Câu hỏi

Lời giải

Bài này cho file encrypt bằng thuật toán Kuznyechik 256 bit. Với password bị ẩn đi, khóa cho Kuznyechik được sinh bằng việc hash password bằng Streebog, sau đó các bit sau khi hash đi qua một hoán vị.

Do Streebog là hash 512 bit nên sau khi qua bảng hoán vị vẫn có 512 bit. Do đó khóa là 256 bit thấp sau khi hash.

Cả Sreetebog và Kuznyechik là các tiêu chuẩn mật mã nên không thể phá. Lỗi nằm ở bảng hoán vị.

Sau khi thử tạo key từ một số password (tiếng Nga), ta thấy rằng trong 256 bit của khóa chỉ có một số ít bit 1. Đặc biệt là các vị trí xuất hiện 1 là không nhiều. Do đó chúng ta có thể giảm không gian 256 bit xuống một tập nhỏ hơn.

Giả sử ta tạo các key Kuznyechik K_1, K_2, \dots từ việc hash các password w_1, w_2, \dots , nghĩa là $K_i = H(w_i)$, đặt I_i là tập hợp các vị trí bit 1 xuất hiện.

Đặt $I = I_1 \cup I_2 \cup \dots$ là tập các vị trí có thể là bit 1 khi hash một password bất kỳ. Tập này chứa không quá 10 phần tử.

Khi đó khóa Kuznyechik để mã hóa sẽ là khóa có bit 1 nằm ở các vị trí theo tập $J \subset I$. Thủ tất cả tập con của I đến khi decrypt và decode thành công.

Khóa decrypt viết ở dạng hex là

```
0000e00000000080000000000000000000000000000000400000000000000004000000004
```

❶ Kết quả (thơ tiếng Nga)

Мой дядя самых честных правил,

Когда не в шутку занемог,

Он уважать себя заставил

И лучше выдумать не мог.

Его пример другим наука;

Но, боже мой, какая скука

С больным сидеть и день и ночь,

Не отходя ни шагу прочь!

Какое низкое коварство

Полуживого забавлять,

Ему подушки поправлять,

Печально подносить лекарство,

Вздыхать и думать про себя:

Когда же черт возьмет тебя!

Задача 2. Анализ GOST-CryptoFox

Phân tích GOST-CryptoFox

Đặt $V_n(2^m)$ là không gian vector n chiều trên trường \mathbb{F}_{2^m} , \boxplus là phép cộng trên vành $\mathbb{Z}_{2^{32}}$ (phép cộng modulo 2^{32}), \oplus là phép cộng trên $V_n(2)$ (phép XOR).

Đặt $\psi : V_{32}(2) \rightarrow \mathbb{Z}_{2^{32}}$ là hàm biến đổi vector 32 bit thành số nguyên 32 bit. Công thức của hàm ψ là

$$\psi(\beta) = \bar{\beta} = \sum_{i=1}^{32} \beta_i 2^{32-i}$$

với $\beta = (\beta_1, \beta_2, \dots, \beta_{32}) \in V_{32}(2)$ và $\bar{\beta} \in \mathbb{Z}_{2^{32}}$.

Trường \mathbb{F}_{2^8} dùng đa thức tối giản $1 + x^2 + x^3 + x^4 + x^8$ với nghiệm θ . Phần tử trên \mathbb{F}_{2^8} được viết dưới dạng vector $V_8(2)$.

GOST-CryptoFox dùng 32 vòng, độ dài khối là 64 bit và độ dài khóa là 256 bit.

Ngoài ra, trong thuật toán sử dụng:

- các hoán vị $\pi_1, \pi_2, \pi_3, \pi_4$ trên tập $\{1, 2, \dots, 8\}$ để chỉ định khóa con cho từng vòng;
- các hoán vị 8 bit s_1, s_2, s_3, s_4 là các S-box của thuật toán.

Thuật toán sinh khóa con

Khóa đầu vào K có 256 bit.

- Khóa K đầu tiên được chia thành 8 đoạn 32 bit, $K = K_1 \| K_2 \| \dots \| K_8$. Như vậy $K \in V_{256}(2)$, $K_i \in V_{32}(2)$ với $1 \leq i \leq 8$
- Ở các vòng 1-8, sử dụng hoán vị π_1 để chỉ định khóa con, cụ thể là ở vòng thứ i sẽ dùng khóa $k_i = K_{\pi_1(i)}$, $1 \leq i \leq 8$
- Ở các vòng 9-16, sử dụng hoán vị π_2 để chỉ định khóa con, cụ thể là ở vòng thứ $i+8$ sẽ dùng khóa con $k_{i+8} = K_{\pi_2(i)}$, $1 \leq i \leq 8$
- Ở các vòng 17-24, sử dụng hoán vị π_3 để chỉ định khóa con, cụ thể là ở vòng thứ $i+16$ sẽ dùng khóa con $k_{i+16} = K_{\pi_3(i)}$, $1 \leq i \leq 8$
- Ở các vòng 25-32, sử dụng hoán vị π_4 để chỉ định khóa con, cụ thể là ở vòng thứ $i+24$ sẽ dùng khóa con $k_{i+24} = K_{\pi_4(i)}$, $1 \leq i \leq 8$

Khi đó các khóa con cho 32 vòng là k_1, k_2, \dots, k_{32} .

Thuật toán mã hóa

GOST-CryptoFox 2024 dựa trên mô hình Feistel. Round function $f_k : V_{64}(2) \rightarrow V_{64}(2)$ với khóa $k \in V_{32}(2)$:

$$f_k : (\alpha_1, \alpha_2) \rightarrow (\alpha_2, \alpha_1 \oplus g_k(\alpha_2, k)), \quad \alpha_1, \alpha_2 \in V_{32}(2)$$

với $g_k : V_{32}(2) \rightarrow V_{32}(2)$ xác định bởi

$$g_k : \alpha \rightarrow h(S(\psi^{-1}(\psi(\alpha) \boxplus \psi(k)))),$$

trong đó:

- h là ánh xạ tuyến tính trên $V_{32}(2)$ xác định bởi ma trận 4×4 là \mathbf{H} trên \mathbb{F}_{2^8} ,

$$h(\lambda^{(1)}, \dots, \lambda^{(4)}) = (\lambda^{(1)}, \dots, \lambda^{(4)})\mathbf{H}, \quad \lambda^{(1)}, \dots, \lambda^{(4)} \in \mathbb{F}_{2^8},$$

$$\mathbf{H} = \begin{pmatrix} \theta & \theta \oplus 1 & 1 & 1 \\ 1 & \theta & \theta \oplus 1 & 1 \\ 1 & 1 & \theta & \theta \oplus 1 \\ \theta \oplus 1 & 1 & 1 & \theta \end{pmatrix},$$

- S-box $S = (s_1, s_2, s_3, s_4)$ gồm 4 S-box con theo quy tắc

$$S(\beta) = (s_1(\beta^{(1)}), \dots, s_4(\beta^{(4)})), \quad \beta = (\beta^{(1)}, \dots, \beta^{(4)}) \in V_4(2^8),$$

nghĩa là khi đó:

- hai vector 32 bit α và k sẽ được chuyển thành số nguyên 32 bit và cộng modulo 2^{32} , kết quả sau đó được chuyển lại thành vector 32 bit

$$\lambda = \psi^{-1}(\psi(\alpha) \boxplus \psi(k));$$

- vector 32 bit λ được chia thành 4 đoạn $\lambda^{(1)}, \dots, \lambda^{(4)}$ là các phần tử \mathbb{F}_{2^8} và biểu diễn $\lambda = (\lambda^{(1)}, \dots, \lambda^{(4)})$ là không gian 4 chiều $V_4(2^8)$. Sau đó tính $\beta = \lambda H$;
- vector 32 bit β được chia thành 4 vector 8 bit là $\beta^{(1)}, \dots, \beta^{(4)}$. Sau đó mỗi vector $\beta^{(i)}$ đi qua S-box con s_i , $i = 1, 2, 3, 4$. Kết quả được ghép lại nhau thành $S(\beta)$.

Mã hóa là hàm $b_K : V_{64}(2) \rightarrow V_{64}(2)$ định nghĩa bởi phương trình:

$$b_K(\alpha) = g_{k_32}g_{k_31} \dots g_{k_2}g_{k_1}(\alpha), \quad \alpha \in V_{64}(2)$$

Câu hỏi

- Trong hệ mật mã GOST-CryptoFox trên đầy đủ 32 vòng tìm các vi sai liên quan đến các khóa với xác suất là 1 và chỉ sử dụng các khóa liên quan và cặp plaintext thích hợp. Điều đó nghĩa là, với mỗi khóa chưa biết $K \in V_{256}$ tìm phần tử $\varepsilon \in V_{256} \setminus \{\bar{0}_{256}\}$ và $\delta_1, \delta_2 \in V_{64}$ sao cho với cách chọn ngẫu nhiên plaintext $\alpha \in V_{64}(2)$ đẳng thức vi sai với xác suất là 1 là phương trình:

$$b_K(\alpha) \oplus b_{K \oplus \varepsilon}(\alpha \oplus \delta_1) = \delta_2 \quad (4.5)$$

- (Câu hỏi research). Thay phép \boxplus bằng phép \oplus trong mô hình GOST-CryptoFox 2024 ở trên và kí hiệu là mô hình GOST-CryptoFox- \oplus . Thực hiện tấn công trên mô hình GOST-CryptoFox- \oplus 2024 với nhiều vòng nhất có thể và đánh giá độ phức tạp. Có khả năng tấn công GOST-CryptoFox- \oplus đầy đủ 32 vòng hay không?

Lời giải bị sai của mình

Для каждого неизвестного ключа шифрования $K \in V_{256}$ нужно найти такие $\varepsilon \in V_{256} \setminus \{\bar{0}_{256}\}, \delta_1, \delta_2 \in V_{64}$ что при случайному и равновероятном выборе открытого текста $\alpha \in V_{64}(2)$ равенство с вероятностью 1 справедливо равенство

$$b_K(\alpha) \oplus b_{K \oplus \varepsilon}(\alpha \oplus \delta_1) = \delta_2$$

Допустим ключ шифрования K разделяется на 8 частей K_1, K_2, \dots, K_8 , где $K_1, \dots, K_8 \in V_{32}(2)$. Значит $K = K_1 \| K_2 \| \dots \| K_8$.

Процесс шифрования задается равенством

$$b_K(\alpha) = g_{k_32} \dots g_{k_2}g_{k_1}(\alpha)$$

где $\alpha \in V_{64}(2)$ -- двоичный вектор длины 64.

Допустим $\alpha = \alpha_1 \| \alpha_2$, где $\alpha_1, \alpha_2 \in V_{32}$. В первом раунде, мы рассмотрим часть α_2 .

У нас есть

$$g_{k_1}(\alpha_2) = h(S(\psi^{-1}(\psi(\alpha_2) \boxplus \psi(k_1))))$$

Аналогично, обозначим

- $\varepsilon = \varepsilon_1 \| \varepsilon_2 \| \dots \| \varepsilon_8$, где $\varepsilon \in V_{256}(2)$ и $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_8 \in V_{32}(2)$
- $\delta_1 = \delta_{11} \| \delta_{12}$, где $\delta_1 \in V_{64}(2)$, $\delta_{11}, \delta_{12} \in V_{32}$

Тогда

$$g_{k_1 \oplus \varepsilon_2}(\alpha_2 \oplus \delta_{12}) = h(S(\psi^{-1}(\psi(\alpha_2 \oplus \delta_{12}) \boxplus \psi(k_1 \oplus \varepsilon_2))))$$

Заметим, что если $\varepsilon_2 \equiv \delta_{12}$, тогда $\psi(\alpha_2) \boxplus \psi(k_1) \equiv \psi(\alpha_2 \oplus \delta_{12}) \boxplus \psi(k_1 \oplus \varepsilon_2)$. Например, если у меня есть (на кольце $2^4 = 16$) $a = 13$ и $k = 6$. Тогда $a \boxplus b = 3 \pmod{2^4}$. Выбираем $b = 10$ и $l = 10$, тогда $a \oplus b = 13 \oplus 10 = 1101 \oplus 1010 = 0111 = 7$, и $k \oplus l = 6 \oplus 10 = 0110 \oplus 1010 = 1100 = 12$. Получаем $7 \boxplus 12 = 3 \pmod{2^4}$.

Через первый раунд,

$$f_{k_1} : (a_1, a_2) \rightarrow (a'_1 = a_2, a'_2 = a_1 \oplus g_{k_1}(a_2))$$

еще одно свойство

$$f_{k_1 \oplus \varepsilon_2} : (a_1 \oplus \delta_{11}, a_2 \oplus \delta_{12}) \rightarrow (A'_1 = a_2 \oplus \delta_{12}, A'_2 = a_1 \oplus \delta_{11} \oplus g_{k_1 \oplus \varepsilon_2}(a_2 \oplus \delta_{12}))$$

Как показан выше, $a'_2 \oplus A'_2 = \delta_{11}$ и $a'_1 \oplus A'_1 = \delta_{12}$.

Через второй раунд,

$$f_{k_2} : (a'_1, a'_2) \rightarrow (a''_1 = a'_2, a''_2 = a'_1 \oplus g_{k_2}(a'_2))$$

Это следует $a''_1 = a'_2 = a_1 \oplus g_{k_1}(a_2)$ и $a''_2 = a_2 \oplus g_{k_2}(a_1 \oplus g_{k_1}(a_2))$.

Аналогично

$$f_{k_2 \oplus \varepsilon_1} : (A'_1, A'_2) \rightarrow (A''_1 = A'_2, A''_2 = A'_1 \oplus g_{k_2 \oplus \varepsilon_1}(A'_2))$$

Это следует $A''_1 = A'_2 = a_1 \oplus \delta_{11} \oplus g_{k_1 \oplus \varepsilon_2}(a_2 \oplus \delta_{12})$ и $A''_2 = A'_1 \oplus g_{k_2 \oplus \varepsilon_1}(A'_2) = a_2 \oplus \delta_{12} \oplus g_{k_2 \oplus \varepsilon_1}(a_1 \oplus \delta_{11} \oplus g_{k_1 \oplus \varepsilon_2}(a_2 \oplus \delta_{12}))$.

Поэтому

$$a''_1 \oplus A''_1 = \delta_{11}$$

так как $g_{k_1 \oplus \varepsilon_2}(a_2 \oplus \delta_{12}) \equiv g_{k_1}(a_2)$ как доказано выше.

И пусть $G = a_1 \oplus g_{k_1 \oplus \varepsilon_2}(a_2 \oplus \delta_{12}) = a_1 \oplus g_{k_1}(a_2)$, тогда у нас есть

$$a''_2 \oplus A''_2 = \delta_{12} \oplus g_{k_2}(G) \oplus g_{k_2 \oplus \varepsilon_1}(\delta_{11} \oplus G)$$

Аналогично, если выберем $\varepsilon_1 \equiv \delta_{11}$, мы получим $g_{k_2}(G) \equiv g_{k_2 \oplus \varepsilon_1}(\delta_{11} \oplus G)$ и отсюда $a''_2 \oplus A''_2 = \delta_{12}$.

Можем отметить, что после двух раундов, для любого открытого текста $a \in V_{64}$ и для любого подключей шифрования $k_1, k_2 \in V_{32}$, дифференциальный будет фиксирован

$$a''_1 \| a''_2 \oplus A''_1 \| a''_2 = \delta_{11} \| \delta_{12}$$

Продолжаем этот процесс и получим, что

$$\begin{aligned}\varepsilon_3 &= \delta_{12}, \\ \varepsilon_4 &= \delta_{11}, \\ \varepsilon_5 &= \delta_{12}, \\ \varepsilon_6 &= \delta_{11}, \\ \varepsilon_7 &= \delta_{12}, \\ \varepsilon_8 &= \delta_{11}\end{aligned}$$

и дифференциал между $a^{(i)}$ и $A^{(i)}$ после i -го раунда, где $1 \leq i \leq 32$, равен

$$a^{(i)} \oplus A^{(i)} = \begin{cases} \delta_{12} \| \delta_{11}, & \text{если } i \text{ нечетный,} \\ \delta_{11} \| \delta_{12}, & \text{если } i \text{ четный} \end{cases}$$

Ответ: такие векторы, удовлетворяющие равенство

$$b_K(a) \oplus b_{K \oplus \varepsilon}(a \oplus \delta_1) = \delta_2$$

с вероятностью 1 построены со следующим образом:

1. Выбрать любой ненулевой вектор $\delta_1 = \delta_{11} \parallel \delta_{12}$, где $\delta_1 \in V_{64}(2)$ и $\delta_{11}, \delta_{12} \in V_{32}(2)$
2. Построить вектор $\varepsilon = \delta_{12} \parallel \delta_{11} \parallel \delta_{12} \parallel \delta_{11} \parallel \delta_{12} \parallel \delta_{11} \parallel \delta_{12} \parallel \delta_{11}$, где $\varepsilon \in V_{256}$
3. Тогда получим $\delta_2 \equiv \delta_1 = \delta_{11} \parallel \delta_{12} \in V_{64}$

Задача 3. Инвариантные структуры

Câu hỏi

Alice và Bob sử dụng thuật toán PRINTcipher-48 để mã hóa dữ liệu. Eva rất muốn biết Alice và Bob đã trao đổi thông tin gì bằng thuật toán này. Eva biết được perm key là 7e92c53f và 10 cặp plaintext-ciphertext theo bảng sau:

№	Открытый текст	Шифртекст
1	5ef30b0e8810	63b05da1572d
2	08419219c0a9	19919629a291
3	2ba2a41913b3	39f39539b0af
4	27da9fea907c	1feeee47a39b
5	2b52a13970b2	1b32b8080293
6	c0cb1af1b482	9c9e8cb63ff9
7	6cd9e41199a3	A6238ffd3462
8	0b40872901a5	2bf09e0a70ab
9	1a8d558c018c	2d16eba7a
10	38c3840ae3a9	2953ab0bf3b1

Hãy giúp Eva khôi phục 80 bit của khóa bí mật. Tài liệu mô tả thuật toán cũng như cách tấn công dựa trên invariant structure (cấu trúc bất biến) [38].

Задача 4. Корректирующий код и шифр Вернама

Câu hỏi

Alice gửi cho Bob thông điệp được mã hóa bằng mã Vernam.

Ở đây, để mã hóa thông điệp, Alice chọn một mã sửa lỗi nhị phân độ dài 256 với khả năng sửa một bit. Bob giải mã và decode thông điệp. One-time-key được sinh trước đó và được truyền bí mật giữa Alice và Bob, nghĩa là Alice gửi qua kênh truyền ciphertext y bằng:

$$y = \text{code}(x) \oplus \text{key},$$

trong đó:

- $x \in \{0, 1\}^m$ - thông điệp
- $y \in \{0, 1\}^{256}$ - ciphertext
- $\text{key} \in \{0, 1\}^{256}$ - khóa bí mật one-time
- m - độ dài thông điệp và là kích thước của code

- code : $\{0, 1\}^m \rightarrow \{0, 1\}^{256}$ - hàm encode của mã sửa lỗi
- \oplus - phép XOR.

Bob nhận được ciphertext y và giải mã bằng cách

$$x = \text{decode}(y \oplus key),$$

với decode : $\{0, 1\}^{256} \rightarrow \{0, 1\}^m$ là hàm decode của mã sửa lỗi.

Anton và Bella muốn sử dụng kênh truyền của Alice và Bob để trao đổi thông tin. Anton có thể lặng lẽ đọc và thay đổi ciphertext y trước khi Alice chuyển qua kênh truyền. Bella thì có thể đọc ciphertext từ kênh truyền. Cả Anton và Bella đều không biết khóa bí mật one-time, mà chỉ biết về mã sửa lỗi (độ dài m và thuật toán). Biết rằng xác suất xảy ra lỗi trên kênh truyền là 0.

Anton có thể gửi thông điệp cho Bella, sử dụng kênh truyền của Alice và Bob, mà không ảnh hưởng hay tiết lộ sự tồn tại của bản thân không? Nếu có, lượng thông tin lớn nhất mà Anton có thể gửi đi trong một thông-điệp-Alice là bao nhiêu?

Задача 5. Эффективная реализация

Câu hỏi

Alice nhận được một thiết bị phần cứng, cho phép thực hiện các phép tính trên đường cong elliptic, bao gồm: cộng, trừ, nhân đôi, nhân ba và nhân 7 (nhân vô hướng một điểm với 7).

Để kiểm tra độ chính xác của thiết bị, Alice kiểm tra phép nhân vô hướng một điểm cho 2024.

Có ấy tìm được một dãy gồm 9 hành động, bao gồm 1 phép cộng, 5 phép nhân đôi, 2 phép nhân ba và 1 phép nhân 7.

Có thể tính phép nhân $2024G$ với ít hành động hơn không? Nếu có, hãy chỉ ra cách tính. Nếu không thể, hãy chứng minh.

➊ Lời giải bị sai của mình

Пусть G - начальная точка. Нам нужно вычислять точку $2024G$.

Так как $2024 = 2^3 \cdot 11 \cdot 23$, значит это число не имеет вид $2^i \cdot 3^j \cdot 7^k$, мы получим, что 2024 должно разделиться на два слагаемых:

$$2024 = 2^a \cdot 3^b \cdot 7^c + 2^m \cdot 3^n \cdot 7^p$$

После пытаться получим, что $2024 = 2^3 + 2^5 \cdot 3^2 \cdot 7$. Это значит, нужно одно действие сложения, 5 действий удвоения ($5 = \max\{3, 5\}$), 2 действия утроения и 1 усемерения. Общее число действия равно 9.

Теперь докажу, что не можно вычислять $2024 \cdot G$ с меньше количеством действий, чем 9.

Мы уже разделили 2024 на сумму двух слагаемых. Сейчас попробуем разделить на сумму трех слагаемых, т.е.

$$2024 = 2^a \cdot 3^b \cdot 7^c + 2^m \cdot 3^n \cdot 7^p + 2^u \cdot 3^v \cdot 7^w$$

Так как здесь уже есть 2 действия сложения, мы хотим уменьшить степени 2^5 , или 3^2 , или 7. Заметим, что

- Удвоение эквивалент сложение, $2 = 1 + 1$, или $2G = G + G$. Это значит, что в каждом случае требуется одно действие (либо удвоение, либо сложение)

- Утроение - $3 = 2 + 1$. Здесь, для утраения требуется одно действие, а $2 + 1$ требуются два действия (удвоение и сложение). Поэтому разделение 3 на $2+1$ не является актуальным
- Усемерение - $7 = 2 \cdot 3 + 1 = 2^2 + 3$. Здесь, для усемерения требуется одно действие, для $2 \cdot 3 + 1$ требуются 3 действия (одно удвоение, одно утроение, одно сложение), и для $2^2 + 3$ требуются 4 действия (два удвоения, одно утроения, одно сложение)
- $3^2 = 7+2 - 3^2$ требуются два действия утраения, а $7+2$ нужно 3 действия (удвоение, усемерение и сложение)
- $7^2 = 2^4 \cdot 3 + 1 = \dots = 2^i \cdot 3^j \cdot 7^k + 2^r \cdot 3^s \cdot 7^t$ (все способов разделения 7^2 на сумму 2 слагаемых с факторами 2, 3 и 7) - для каждого способа требуются более 2 действия

Это значит, когда разделить 2024 на сумму трёх слагаемых, количество действия либо сохраняется, либо увеличивается.

Ответ: НЕВОЗМОЖНО.

Задача 7. Дифференциальная равномерность

Câu hỏi

Ánh xạ

$$f : GF(2^n) \rightarrow GF(2^n)$$

được gọi là differential δ -uniformity, nếu với mọi $a \in GF(2^n)^*$ và với mọi $b \in GF(2^n)$ phương trình

$$f(x + a) + f(x) = b$$

có không quá δ nghiệm trên $GF(2^n)$.

Ví dụ, hàm nghịch đảo trên $GF(2^8)$ xác định bởi

$$I(x) = \begin{cases} x^{-1}, & \text{nếu } x \neq 0 \\ 0, & \text{ngược lại} \end{cases}$$

là hàm differential 4-uniformity.

Xét toán tử nhóm \circ trên $GF(2^n)$ với phần tử đơn vị là e , sao cho với mọi $x \in GF(2^n)$ ta có $x \circ x = e$.

Gọi f là ánh xạ

$$f : GF(2^n) \rightarrow GF(2^n)$$

là ánh xạ differential δ -uniformity theo toán tử nhóm \circ , nếu với mọi $a \in GF(2^n) \setminus \{e\}$ và với mọi $b \in GF(2^n)$ thì phương trình

$$f(x \circ a) \circ f(x) = b$$

có không quá δ nghiệm trên $GF(2^n)$.

Hãy tìm toán tử \circ sao cho hàm $I(x)$ có differential formidity thấp nhất.

Задача 8. Разделение секретов

Câu hỏi

Ông chủ trước khi đi nghỉ mát đã thay đổi code 6 chữ số của két sắt và, trong mọi trường hợp, chia bí mật đó theo sơ đồ "2 trong N " bằng việc vẽ một "đường thẳng" bí mật dưới vành modulo 1 triệu, trong đó code là tọa độ ban đầu của đường thẳng đó.

Hoàn cảnh khiến Andrey và Vladimir phải khẩn trương mở két sắt trong lúc không liên lạc được với ông chủ. Hãy giúp Andrey và Vladimir mở két với thông tin của họ: login của Andrey $x_A = 137059$, chia sẻ bí mật $y_A = 844061$, login của Vladimir là $x_B = 752024$, chia sẻ bí mật $y_B = 639516$.

Lời giải bị sai của mình

Даны:

- Логин Андрея $x_A = 137059$, доля секрета $y_A = 844061$
- Логин Владимира $x_B = 752024$, доля секрета $y_B = 639516$

Секреты созданы по схеме "2 из N ". Это значит существует числа i_A, i_B , удовлетворяющие равенство

$$y_A = C_{i_A}^2 \pmod{10^6}, \quad y_B = C_{i_B}^2 \pmod{10^6}$$

Здесь C_b^a - сочетание a элементов из множества b элементов.

Нужно отметить, что пароль является шестизначный код, это следует, что $10^5 \leq i_A, i_B < 10^6$. С использованием компьютерной программы можем найти такие числа. Эти числа будут:

$$i_A = 138967, \quad i_B = 674937$$

Секретная прямая проходит через две точки (x_A, i_A) и (x_B, i_B) . На вещественном множестве \mathbb{R} , уравнение прямой, которая проходит через две точки (x_1, y_1) и (x_2, y_2) , равно

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} \Leftrightarrow (y_2 - y_1)(x - x_1) - (x_2 - x_1)(y - y_1) = 0$$

Секретная прямая вычисляется над кольцом вычетов по модулю 10^6 , поэтому уравнение секретной прямой равно

$$\begin{aligned} (i_B - i_A)(x - x_A) - (x_B - x_A)(y - i_A) &= 0 \pmod{10^6} \\ \Leftrightarrow 535970x + 385035y + 328925 &= 0 \pmod{10^6} \end{aligned}$$

Теперь нам нужно найти начальную ординату прямой, т.е. самый минимальный y , $10^5 \leq y < 10^6$, который удовлетворяет это уравнение для $x = 0$

$$535970 \cdot 0 + 385035y + 328925 = 0 \pmod{10^6}$$

Это уравнение имеет некоторые решения, это 52745, 252745, 452745, 652745 и 852745. Мы берем самый минимальный 252745.

Секретный код будет $C_{252745}^2 \pmod{10^6} = 891140$.

Ответ: 891140.

Задача 9. Схемная сложность S-блока

Câu hỏi

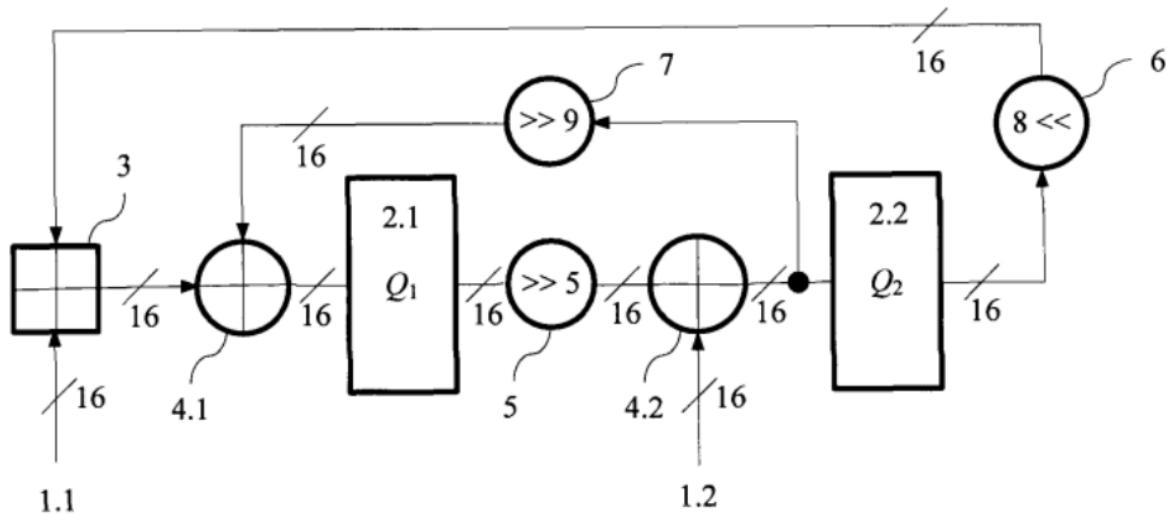
Kí hiệu độ phức tạp mạch của hàm boolean vector (S-box) F trong cơ sở B là $L_B(F)$. Ta nói S-box khả nghịch $F : \{0, 1\}^n \rightarrow \{0, 1\}$ là bất đối xứng (asymmetric computing), nếu với cơ sở B nào đó ta có $L_B(F) \neq L_B(F^{-1})$.

- Chỉ ra một S-box bất đối xứng như vậy.
- Xây dựng một họ vô hạn các S-box bất đối xứng.

3. Xây dựng một họ vô hạn các S-box bất đối xứng $\{F_n\}$ với $F_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$ và $\lim_{n \rightarrow \infty} \frac{L_B(F_n)}{L_B(F_n^{-1})} \neq 1$.
(Đây là bài toán chưa có lời giải).

Задача 10. Анализ алгоритма блочного шифрования «TryHard» (исследовательская)

Câu hỏi



Hình trên thể hiện một vòng của thuật toán mã khối. Tất cả toán tử chạy trong bộ nhớ 16 bit. Các kí hiệu trên hình được giải nghĩa như sau:

Kí hiệu	Ý nghĩa
1.1, 1.2	Input round key
2.1, 2.2	Input word
3	Phép cộng modulo 2^{16}
4.1, 4.2	Toán tử XOR
5	Dịch trái 5 bit
6	Dịch phải 6 bit
7	Dịch trái 9 bit

Thuật toán sinh khóa con cho các vòng giống với GOST 28147-89 (Magma). Trong đó nửa 16 bit đầu là round key cho 1.1 và 16 bit sau là round key cho 1.2.

Đầu ra của thuật toán được biểu diễn ở bảng sau.

Tên gọi	Ý nghĩa
Kích thước khối đầu ra	32 bit
Kích thước khóa	64 bit
Số lượng vòng mã hóa	20 vòng

Câu hỏi. Hãy đánh giá số lượng vòng của thuật toán mã hóa, trong đó số lượng vòng cho phép đánh giá sự độc lập thống kê giữa các plaintext và ciphertext, mà qua đó khôi phục được một phần hoặc toàn bộ khóa. Hãy đề xuất thuật toán khôi phục khóa mã hóa (phải hiệu quả hơn bruteforce).

4.6.7 ISITDTU Quals 2024

Share Mixer 1

Cho p là số nguyên tố lớn 256 bit. Chúng ta sẽ nhập vào dãy x_0, x_1, \dots, x_{n-1} với $n \leq 256$.

Sau đó đê (hàm *share_mixer*) tạo random một dãy c_0, c_1, \dots, c_{30} và đặt $c_{31} = \text{flag}$. Sau đó hai dãy xs và cs sẽ được xáo trộn thành dãy $\{x'_i\}$ và $\{c'_i\}$, và tính *shares* là dãy

$$s_j = \sum_{i=0}^{31} c'_i \cdot (x'_j)^i \pmod{p}$$

với $j = 0, 1, \dots, n-1$.

Ý tưởng là mình sẽ gửi lên dãy gồm 1 số 1, 2 số 2, ..., 15 số 15. Thêm nữa mình gửi 1 số 16, ..., 15 số 30. Cuối cùng là hai số 31 và 32.

Share Mixer 2

Chưa làm ra. :)))

Sign

Chưa làm ra. :)))

Thats so random

Chưa làm ra. :)))

4.7 Các cuộc thi năm 2025

4.7.1 CryptoFox 2025

Olympiad mật mã và bảo mật thông tin CryptoFox 2025.

Số thứ tự	Tên bài	Tình trạng
0	Хакерская атака	Chưa giải
1	Кривизна функции	Giải một phần ý 1
2	Странная вселенная	Chưa giải
3	Магическая шкатулка	Chưa giải
4	Разности для FOX128	Hoàn thành ý 1
5	Шифрование в RSA	Hoàn thành
6	Подозрительная фотография	Chưa giải
7	Замены и перестановки	Hoàn thành
8	Загадочная ошибка	Chưa giải
9	Генератор гаммы на основе XSL-схемы	Chưa giải
10	Регистровое преобразование	Chưa giải
11	Трудный PBKDF	Chưa giải

Giới thiệu

Đầu tiên mình xin phép có một số ý kiến về việc trình bày đê thi. Đê thi có một điểm khá khó chịu là không được biên tập thống nhất. Minh hiểu rằng có nhiều người ra đê nhưng ít nhất cũng phải có người chịu trách nhiệm biên tập lại đê mỗi câu chữ. Kí hiệu ở mỗi câu mỗi khác nhau làm mất tính thống nhất của một cuộc thi. Ví dụ, ở câu 1 dùng $1 \leq \sigma(f)$, còn câu 2 dùng $\alpha, \beta \geq 0$, khác cách viết dấu bằng ở bất đẳng thức. Một

ví dụ khác là kí hiệu trường hữu hạn, ở câu 1 dùng \mathbb{Z}_p nhưng câu 4 lại dùng \mathbb{F}_{2^8} . Trong trường hợp này mình nghĩ nên thống nhất dùng $GF(p)$ và $GF(2^8)$, hoặc \mathbb{F}_p và \mathbb{F}_{2^8} , sẽ hợp lí hơn. Sự không thống nhất cuối cùng còn ở chỗ mỗi đề có font chữ, cỡ chữ khác nhau, thậm chí có đề đánh số trang nhưng có đề không. Vì lý do trên nên khi viết lại đề, mình sẽ thay đổi kí hiệu cho thống nhất.

Về phần nội dung trong đề thì có 4 câu về reverse engineering (câu 0, câu 3, câu 6 và câu 8), còn lại 7 câu về mật mã. Các câu reverse engineering cũng là những thuật toán mật mã được giấu bên trong chương trình. Với kinh nghiệm hạn hẹp về reverse engineering thì mình không làm được những bài đó. =)))

Задача 0. Хакерская атака

Đây là một bài reverse engineering, và mình không biết làm. :)

Задача 1. Кривизна функции

Câu hỏi

Đặt p là số nguyên tố, $p \geq 3$, $\mathbb{F}_p = \{0, 1, \dots, p\}$ là trường hữu hạn modulo p , $\gamma = e^{\frac{2\pi i}{p}}$ là nghiệm phức primitive bậc p của 1.

Với ánh xạ $f : \mathbb{F}_p \rightarrow \{0, 1\}$ và với phần tử $a \in \mathbb{F}_p$ ta định nghĩa số

$$\nu_f(a) = \frac{1}{p} \sum_{x \in \mathbb{F}_p} (-1)^{f(x)} \gamma^{-ax}.$$

Ta định nghĩa curvature của hàm f là đại lượng

$$\sigma(f) = \sum_{a \in \mathbb{F}_p} |\nu_f(a)|.$$

1. Chứng minh rằng

$$1 \leq \sigma(f) < \sqrt{p},$$

và chặn dưới đạt dấu bằng khi và chỉ khi f là hàm hằng.

2. Chứng minh rằng, nếu $p = 2^{k+1} - 1$ và ánh xạ f_t , với $t = 0, 1, \dots, k$, biến đổi từng phần tử $x \in \mathbb{F}_p$ theo nghĩa $f_t(x) = x_t \in \{0, 1\}$, trong đó x_t là biểu diễn nhị phân của x , hay

$$x = x_0 + 2x_1 + \dots + 2^k x_k,$$

thì ta có bất đẳng thức

$$\sigma(f_t) < \frac{4}{\pi} \ln p + \frac{9}{5}.$$

Gợi ý cho câu 2: sử dụng tổng Vinogradova I.M. là

$$\sum_{h=0}^{p-1} \left| \sum_{j=0}^{N-1} \gamma^{hj} \right| < \frac{2}{\pi} p \ln p + \frac{2}{5} p + N,$$

với mọi số tự nhiên p và N .

[TODO] Giải

Bài này mình giải được một phần câu 1.

Sử dụng biến đổi Fourier rồi rạc, đặt

$$u_x = (-1)^{f(x)}, \quad x \in \mathbb{F}_p.$$

Đặt $U_a = p \cdot \nu_a(f)$ thì ta có

$$U_a = \sum_{x=0}^{p-1} u_x e^{-2\pi i \frac{a}{p} x}$$

với $a = 0, 1, \dots, p-1$.

Khi đó, theo biến đổi Fourier rời rạc ngược có thể thấy liên hệ giữa dãy $\{u_x\}$ theo $\{U_a\}$ là

$$u_x = \frac{1}{p} \sum_{a=0}^{p-1} U_a \cdot e^{2\pi i \frac{x}{p} a}$$

với $x = 0, 1, \dots, p-1$.

Theo định lí Parceval thì

$$\sum_{x=0}^{p-1} |u_x|^2 = \frac{1}{p} \sum_{a=0}^{p-1} |U_a|^2,$$

hay

$$\sum_{x=0}^{p-1} \left| (-1)^{f(x)} \right|^2 = \frac{1}{p} \sum_{a=0}^{p-1} |p \cdot \nu_f(a)|^2 = \frac{1}{p} \sum_{a=0}^{p-1} p^2 |\nu_f(a)|^2 = p \sum_{a=0}^{p-1} |\nu_f(a)|^2.$$

Sử dụng bất đẳng thức Cauchy, với hai vector bất kì

$$\mathbf{s} = (s_0, s_1, \dots, s_{n-1}), \mathbf{t} = (t_0, t_1, \dots, t_{n-1})$$

ta có

$$(s_0 t_0 + s_1 t_1 + \dots + s_{n-1} t_{n-1})^2 \leq (s_0^2 + s_1^2 + \dots + s_{n-1}^2)(t_0^2 + t_1^2 + \dots + t_{n-1}^2).$$

Cho $s_i = 1$ và $t_i = |\nu_f(i)|$ với $i = 0, 1, \dots, p-1$ ta có

$$p \sum_{a=0}^{p-1} |\nu_f(a)|^2 = (1+1+\dots+1) \left(\sum_{a=0}^{p-1} |\nu_f(a)|^2 \right) \geq \left(\sum_{a=0}^{p-1} |\nu_f(a)| \right)^2.$$

Do $f : \mathbb{F}_p \rightarrow \{0, 1\}$ nên $(-1)^{f(x)} \in \{-1, 1\}$, suy ra $\left| (-1)^{f(x)} \right|^2 = 1$ với mọi $x = 0, 1, \dots, p-1$.

Như vậy bất đẳng thức trên có thể thu được

$$p = \sum_{x=0}^{p-1} \left| (-1)^{f(x)} \right|^2 = p \sum_{a=0}^{p-1} |\nu_f(a)|^2 \geq \left(\sum_{a=0}^{p-1} |\nu_f(a)| \right)^2 = (\sigma(f))^2,$$

tương đương với

$$\sigma(f) \leq \sqrt{p}.$$

Dấu bằng xảy ra khi và chỉ khi $|\nu_f(a)|$ là hằng số nhưng dễ thấy điều này không thể xảy ra. Do đó có thể kết luận

$$\boxed{\sigma(f) < \sqrt{p}.}$$

Задача 2. Странная вселенная**Câu hỏi**

Trong một vũ trụ lạ thường nọ, gọi là Strange Universe, chúng ta thực hiện các thí nghiệm quantum.

Kết thúc một thí nghiệm, chúng ta nhận được toán tử quantum A và các trạng thái quantum $|x\rangle$ và $|y\rangle$, ở đây

$$A(\alpha|x\rangle + \beta|y\rangle) \neq \alpha A(|x\rangle) + \beta A(|y\rangle)$$

với $\alpha, \beta \geq 0$ và $\alpha + \beta = 1$. Chúng ta muốn tiếp tục thí nghiệm nhưng các thiết bị đã hỏng.

Chúng ta sử dụng một kênh truyền quantum để trao đổi thông tin, và chúng ta cần trả lời các câu hỏi sau:

1. Trong Strange Universe, giao thức mật mã BB84 có không an toàn không? Nếu có thì điều kiện nào khiến việc này xảy ra?
2. Trong Strange Universe, giao thức mật mã E91 có không an toàn không? Nếu có thì điều kiện nào khiến việc này xảy ra?
3. Trong Strange Universe có tính chất thú vị nào khác của kênh truyền quantum không?

[TODO] Giải**Задача 3. Магическая шкатулка**

Lại một bài reverse engineering khác và mình cũng không làm ra.

Задача 4. Разности для FOX128**Câu hỏi**

Grisha phát triển một thuật toán mã khối là FOX128.

Đặt $V_{16}(2^8)$ là không gian vector có số chiều bằng 16 trên trường \mathbb{F}_{2^8} .

Đặt $S(X)$ là nhóm symmetric, tác độ lên tập hữu hạn X .

Phép biến đổi không tuyến tính của FOX128 là hoán vị π' . Khi đó với

$$\alpha = (\alpha_1, \alpha_2, \dots, \alpha_{16}) \in V_{16}(2^8)$$

ta sẽ có

$$\pi'(\alpha) = (\pi(\alpha_1), \pi(\alpha_2), \dots, \pi(\alpha_{16}))$$

với $\pi \in S(\mathbb{F}_{2^8})$.

Hoán vị π có thể được viết dưới dạng

$$\pi'' = (\pi(0), \pi(1), \dots, \pi(255))$$

như sau

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	252	238	221	17	207	110	49	22	251	196	250	218	35	197	4	77
1	233	119	240	219	147	46	153	186	23	54	241	187	20	205	95	193
2	249	24	101	90	226	92	239	33	129	28	60	66	139	1	142	79
3	5	132	2	174	227	106	143	160	6	11	237	152	127	212	211	31
4	235	52	44	81	234	200	72	171	242	42	104	162	253	58	206	204
5	181	112	14	86	8	12	118	18	191	114	19	71	156	183	93	135
6	21	161	150	41	16	123	154	199	243	145	120	111	157	158	178	177
7	50	117	25	61	255	53	138	126	109	84	198	128	195	189	13	87
8	223	245	36	169	62	168	67	201	215	121	214	246	124	34	185	3
9	224	15	236	222	122	148	176	188	220	232	40	80	78	51	10	74
10	167	151	96	115	30	0	:math:	68	26	184	56	130	100	159	38	65
							98`									
11	173	69	70	146	39	94	85	47	140	163	165	125	105	213	149	59
12	7	88	179	64	134	172	29	247	48	55	107	228	136	217	231	137
13	225	27	131	73	76	63	248	254	141	83	170	144	202	216	133	:math:`97`
14	32	113	103	164	45	43	9	91	203	155	37	208	190	229	108	82
15	89	166	116	210	230	244	180	192	209	102	175	194	57	75	99	182

Ở bảng trên, phần tử hàng i và cột j ($0 \leq i, j \leq 15$) là giá trị $\pi(16i + j)$.

Phép biến đổi tuyến tính của phần tử $\alpha = (\alpha_1, \dots, \alpha_{16})$ với $\alpha \in V_{16}(2^8)$ được kí hiệu là h . Khi đó với phần tử $\alpha = (\alpha_1, \dots, \alpha_{16})$ ta chuyển thành dạng ma trận

$$A = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 \\ \alpha_9 & \alpha_{10} & \alpha_{11} & \alpha_{12} \\ \alpha_{13} & \alpha_{14} & \alpha_{15} & \alpha_{16} \end{pmatrix}.$$

Khi đó $h(\alpha)$ là ma trận B , được tính theo công thức:

$$B = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \cdot A,$$

với phép cộng và nhân được thực hiện giống phép cộng và nhân trên các phần tử của trường.

Phép cộng với khóa k là ánh xạ có dạng

$$\nu_k(\alpha) = \alpha \oplus k,$$

với \oplus là toán tử cộng từng bit theo modulo 2 (phép XOR), và $\alpha, k \in V_{16}(2^8)$.

Khi đó, hàm mã hóa đối với bản rõ $\alpha \in V_{16}(2^8)$ có dạng

$$g_{k_1 k_2}(\alpha) = \nu_{k_2} h \pi' v_{k_1}(\alpha).$$

Grisha sử dụng mode ECB để mã hóa từ *ПРИВЕДЕНИЕ*. Ở đây anh ấy dùng bảng sau để encode kí tự tiếng Nga sang phần tử thuộc \mathbb{F}_{2^8} và để gọn mình sẽ viết ở dạng thập phân.

A	У	О	И	Э	Ы	Я	Ю	Е	Ё	Б	В	Г	Д	Ж	З	Й
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
К	Л	М	Н	П	Р	С	Т	Ф	Х	Ц	Ч	Ш	Щ	Ђ	Ђ	...
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	...

Khi sử dụng FOX128 cipher thì chúng ta sẽ thêm vào phía trước các số 0 cho đủ vector gồm 16 phần tử. Ví dụ sau khi encode thông điệp chúng ta có vector $(25, 2, 17, 23)$ thì chúng ta pad thành

$$\alpha = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 25, 2, 17, 23).$$

Chúng ta biết rằng khi mã hóa thông điệp *ПРИВЕДЕНИЕ*, ta thu được bản mã

$$\beta_1 = (216, 121, 18, 144, 93, 121, 17, 11, 114, 81, 251, 135, 200, 53, 54, 79)$$

Sau đó Grisha mã hóa một thông điệp khác với cùng khóa con k_1, k_2 như trên với những thông điệp có nghĩa khác nhau nhằm mục đích nghiên cứu sự phụ thuộc giữa các bản rõ với nhau và với bản mã. Khi đó, với một bản rõ nào đó Grisha đã mã hóa thành

$$\beta_2 = (216, 121, 230, 68, 93, 121, 229, 223, 114, 81, 251, 83, 200, 53, 194, 79).$$

Grisha đã làm mất thí nghiệm của mình và không còn thông tin về khóa, cũng như bản rõ tương ứng với β_2 . Hãy giúp anh ấy.

1. Khôi phục bản rõ tương ứng bản mã β_2 .
2. Đưa ra thuật toán tối ưu khôi phục lại k_2 hoặc một phần của nó, nếu biết rằng khi tạo khóa con k_2 chỉ sử dụng các nguyên âm.

[TODO] Giải

Mình giải được câu 1 và chưa kịp làm câu 2.

Giả sử chúng ta mã hóa hai bản rõ $\mathbf{a}, \mathbf{a}' \in V_{16}(2^8)$ với cùng khóa \mathbf{k}_1 và \mathbf{k}_2 .

Đặt

$$\mathbf{a} = (a_1, a_2 \dots, a_{16}), \quad \mathbf{a}' = (a'_1, a'_2, \dots, a'_{16}),$$

với a_i và $a'_i \in \mathbb{F}_{2^8}$, $i = 1, 2, \dots, 16$.

Đặt

$$\mathbf{k}_1 = (k_{1,1}, k_{1,2}, \dots, k_{1,16}), \quad \mathbf{k}_2 = (k_{2,1}, k_{2,2}, \dots, k_{2,16}),$$

với $k_{1,i}$ và $k_{2,i} \in \mathbb{F}_{2^8}$, $i = 1, \dots, 16$.

Sau phép biến đổi $\nu_{\mathbf{k}_1}$ ta được

$$\begin{aligned} \nu_{\mathbf{k}_1}(\mathbf{a}) &= (a_1 \oplus k_{1,1}, \dots, a_{16} \oplus k_{1,16}) = (b_1, \dots, b_{16}), \\ \nu_{\mathbf{k}_1}(\mathbf{a}') &= (a'_1 \oplus k_{1,1}, \dots, a'_{16} \oplus k_{1,16}) = (b'_1, \dots, b'_{16}), \end{aligned}$$

ở đây đặt $b_i = a_i \oplus k_{1,i}$, tương tự $b'_i = a'_i \oplus k_{1,i}$.

Sau phép biến đổi thứ hai π' ta có

$$\begin{aligned} \pi'(b_1, \dots, b_{16}) &= (c_1, \dots, c_{16}), \\ \pi'(b'_1, \dots, b'_{16}) &= (c'_1, \dots, c'_{16}), \end{aligned}$$

với $c_i = \pi(b_i)$ và $c'_i = \pi(b'_i)$.

Sau phép biến đổi tuyến tính h ta có

$$h(c_1, \dots, c_{16}) = (d_1, \dots, d_{16}), \quad h(c'_1, \dots, c'_{16}) = (d'_1, \dots, d'_{16}).$$

Cuối cùng, phép biến đổi ν_{k_2} :

$$\begin{aligned}\nu_{k_2}(d_1, \dots, d_{16}) &= (d_1 \oplus k_{2,1}, \dots, d_{16} \oplus k_{2,16}) = (e_1, \dots, e_{16}), \\ \nu_{k_2}(d'_1, \dots, d'_{16}) &= (d'_1 \oplus k_{2,1}, \dots, d'_{16} \oplus k_{2,16}) = (e'_1, \dots, e'_{16}).\end{aligned}$$

Lúc này bản mã chính là các vector (e_1, \dots, e_{16}) và (e'_1, \dots, e'_{16}) .

Nếu XOR hai bản mã với nhau thì

$$(e_1 \oplus e'_1, \dots, e_{16} \oplus e'_{16}) = (e_1, \dots, e_{16}) \oplus (e'_1, \dots, e'_{16}) = (d_1 \oplus d'_1, \dots, d_{16} \oplus d'_{16}),$$

nhưng h là phép biến đổi tuyến tính nên

$$(d_1 \oplus d'_1, \dots, d_{16} \oplus d'_{16}) = h(c_1, \dots, c_{16}) \oplus h(c'_1, \dots, c'_{16}) = h(c_1 \oplus c'_1, \dots, c_{16} \oplus c'_{16}).$$

Như vậy có thể nói rằng

$$(e_1 \oplus e'_1, \dots, e_{16} \oplus e'_{16}) = h(c_1 \oplus c'_1, \dots, c_{16} \oplus c'_{16}).$$

Ma trận H và nghịch đảo của nó là như nhau, do đó có thể viết lại biểu thức trên thành

$$h(e_1 \oplus e'_1, \dots, e_{16} \oplus e'_{16}) = (c_1 \oplus c'_1, \dots, c_{16} \oplus c'_{16}).$$

Vì $c_i = \pi(a_i \oplus k_{1,i})$ và $c'_i = \pi(a'_i \oplus k_{1,i})$, và ta tính được $h(e_1 \oplus e'_1, \dots, e_{16} \oplus e'_{16})$ nên sẽ tìm được các phần tử $c_1 \oplus c'_1, \dots, c_{16} \oplus c'_{16}$.

Lúc này ta thử các $k_{1,i}$ với $i = 1, 2, \dots, 16$ để xem $k_{1,i}$ nào thỏa mãn

$$c_i \oplus c'_i = \pi(a_i \oplus k_{1,i}) \oplus \pi(a'_i \oplus k_{1,i})$$

với điều kiện là $0 \leq a'_i \leq 32$ (theo bảng code bên trên).

Chương trình giải mình để ở đây.

Theo kết quả chạy chương trình thì các vị trí i sau cho $a_i = a'_i$:

$$i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14\}.$$

Như vậy từ tiếng Nga chúng ta cần tìm có dạng *ПРИВ_ДЕНИ_*. Lưu ý rằng chúng ta đã bỏ các số 0 đứng trước trong quá trình decode.

Ở đây, dễ thấy sau kí tự B phải là nguyên âm, tức là vị trí $i = 10$. Có hai ứng cử viên cho vị trí này là kí tự O (encode thành 2) và I (encode thành 3). Trong tiếng Nga có từ "привидение" nên ta sẽ chọn kí tự I cho $i = 10$.

Vì "привидение" là từ giống trung nên kết thúc của nó (trong 6 cách) là "е", "я", "и" và "й". Từ chương trình giải bên trên chỉ có kí tự $Я$ là được chấp nhận.

Như vậy, kết quả là *ПРИВИДЕНИЯ*.

Задача 5. Шифрование в RSA

Câu hỏi

Ta sử dụng thuật toán RSA để mã hóa với tham số sau

$$\begin{aligned}n &= 307113764813174451979648861837374183706400686964058131652523053759767112686 \\ &\quad 921105760986215994148574439656212523327564727423276973650662361700716117034 \\ &\quad 585310519473492500005549242633169443931102631059829073180528952729440631440 \\ &\quad 857431430574567858285873536218804878054530675205445624011186425555088867207 \\ &\quad 5353043\end{aligned}$$

$$e = 5.$$

Ta chặn được hai bản mã:

$$c1 = 693089265758848025485688850665080349728312487495309688707596506807354538873 \\ 067996032536142769206361054315964464537700685311216289298705857554525830204 \\ 6252205886190035304282857903311014974554686997348541958002897243$$

$$c2 = 693089265758848025485688850665080349728313078514173314915077310887781132564 \\ 24750714124272412023498368164062692763390490140270935400218398173406875322 \\ 8613850558221595769683504261263861404892836544032097068963073024.$$

Hãy giải mã các thông điệp, biết rằng bản rõ m_1 và m_2 có liên hệ $m_2 = m_1 + 1$.

Giải

Sử dụng Coppersmith attack ở file `solve_rsa.py`.

Задача 6. Подозрительная фотография

Thêm một bài reverse engineering và lần này là Android. Mình chịu chêt.

Задача 7. Замены и перестановки

Câu hỏi

Chúng ta chặn được một thông điệp mã hóa được truyền qua radio và lưu vào file `ct.txt`. Chúng ta cần biết thông điệp ban đầu là gì.

Biết rằng thông điệp được mã hóa bằng mã dòng (stream cipher). Khóa là một dãy $\gamma_1, \gamma_2, \dots$ được sinh ra từ một khóa ban đầu và đi qua thuật toán sinh khóa.

Thuật toán sinh khóa nhận đầu vào là khóa 256 bits.

256 bits đầu tiên nhận được từ việc mã hóa dãy các bytes θ bằng thuật toán Skipjack với khóa K_{root} (80 bits) và IV (64 bits) là

$$IV = \{0x43, 0x72, 0x79, 0x70, 0x74, 0x46, 0x6f, 0x78\}.$$

Việc mã hóa được thực hiện bằng mode CFB, nghĩa là

$$K_0 = \text{Encrypt}_{cfb}(K_{root}, IV, [0x00] \cdot 32),$$

trong đó $K_{root} \in \{0, 1\}^{80}$, $IV \in \{0, 1\}^{64}$, và bản mã là $K_0 \in \{0, 1\}^{256}$.

256 bits tiếp theo được sinh bởi kết quả mã hóa bên trên, tức là

$$K_1 = \text{Encrypt}_{cfb}(K_{root}, IV, K_0).$$

Thực hiện tương tự như vậy ta có công thức tổng quát là

$$K_i = \text{Encrypt}_{cfb}(K_{root}, IV, K_{i-1}),$$

với $i \in \mathbb{N}$, $K_i \in \{0, 1\}^{256}$.

Dãy $\{\gamma_n\}$ nhận được từ các giá trị K_i và K_0 . Ở đây, 4 bits đầu tiên của dãy $\{\gamma_n\}$ nhận được từ K_0 theo quy tắc

$$\gamma_j = \begin{cases} 0 & \text{nếu } K_0[j] = 0, \\ 1 & \text{trong trường hợp khác,} \end{cases}$$

trong đó $K_0[j]$ là khối 64 bits thứ i của K_0 , $\gamma_j \in \{0, 1\}$ với $j = 0, 1, 2, 3$.

Các bit tiếp theo được sinh tương tự theo quy tắc

$$\gamma_j = \begin{cases} 0 & \text{nếu } K_{j/4}[j \% 4] = 0, \\ 1 & \text{trong trường hợp khác,} \end{cases}$$

với $a \% b$ là số dư khi chia a cho b , $j \in \mathbb{N}$.

Ví dụ ta có khóa

$$K_j = \{0x0c, 0xea, 0x66, 0xe0, 0x96, 0xbd, 0xf1, 0xc0, \\ 0x9c, 0xc2, 0xf1, 0x62, 0xa5, 0x44, 0x1f, 0xb7, \\ 0xdd, 0x4b, 0x2b, 0xbf, 0xd6, 0xd9, 0x64, 0x73, \\ 0xb4, 0x97, 0x7b, 0x39, 0x06, 0x02, 0x4c, 0xb8\}$$

thì ta sẽ chia được thành 4 đoạn, mỗi đoạn 64 bits (hay 8 bytes) như sau

$$K_j[0] = \{0x0c, 0xea, 0x66, 0xe0, 0x96, 0xbd, 0xf1, 0xc0\}, \\ K_j[1] = \{0x9c, 0xc2, 0xf1, 0x62, 0xa5, 0x44, 0x1f, 0xb7\}, \\ K_j[2] = \{0xdd, 0x4b, 0x2b, 0xbf, 0xd6, 0xd9, 0x64, 0x73\}, \\ K_j[3] = \{0xb4, 0x97, 0x7b, 0x39, 0x06, 0x02, 0x4c, 0xb8\}.$$

Bản mã nhận được theo công thức

$$c_i = p_i \oplus \gamma_i$$

với $c_i, p_i \in \{0, 1\}$ và $i = 0, 1, 2, \dots$

Hãy phân tích và khôi phục thông điệp ban đầu, và xác định họ của tác giả nếu biết file được encode bằng cp1251.

Giải

Thuật toán sinh khóa nhận đầu vào là khóa 256 bits.

256 bits đầu tiên nhận được từ việc mã hóa dãy các bytes 0 bằng thuật toán Skipjack với khóa K_{root} (80 bits) và IV (64 bits) là

$$IV = \{0x43, 0x72, 0x79, 0x70, 0x74, 0x46, 0x6f, 0x78\}.$$

Việc mã hóa được thực hiện bằng mode CFB, nghĩa là

$$K_0 = \text{Encrypt}_{cfb}(K_{root}, IV, [0x00] \cdot 32),$$

trong đó $K_{root} \in \{0, 1\}^{80}$, $IV \in \{0, 1\}^{64}$, và bản mã là $K_0 \in \{0, 1\}^{256}$.

Đặt $\text{Enc}(P)$ là hàm mã hóa bản rõ P 64 bits bằng thuật toán Skipjack với khóa K_{root} 80 bits.

Giả sử $K_0 = (K_0[0], K_0[1], K_0[2], K_0[3]) \in \{0, 1\}^{256}$, với $K_0[j] \in \{0, 1\}^{64}$, $j = 0, 1, 2, 3$. Theo mode CFB thì

$$\begin{aligned} K_0[0] &= \text{Enc}(IV) \oplus ([0x00] \cdot 8) = \text{Enc}(IV), \\ K_0[1] &= \text{Enc}(K_0[0]) \oplus ([0x00] \cdot 8) = \text{Enc}(K_0[0]), \\ K_0[2] &= \text{Enc}(K_0[1]) \oplus ([0x00] \cdot 8) = \text{Enc}(K_0[1]), \\ K_0[3] &= \text{Enc}(K_0[2]) \oplus ([0x00] \cdot 8) = \text{Enc}(K_0[2]). \end{aligned}$$

256 bits tiếp theo được tạo với cơ chế tương tự:

$$K_1 = \text{Encrypt}_{cfb}(K_{root}, IV, K_0).$$

Giả sử $K_1 = (K_1[0], K_1[1], K_1[2], K_1[3]) \in \{0, 1\}^{256}$ với $K_1[j] \in \{0, 1\}^{64}$, $j = 0, 1, 2, 3$. Tương tự, theo mode CFB thì

$$\begin{aligned} K_1[0] &= \text{Enc}(IV) \oplus K_0[0], \\ K_1[1] &= \text{Enc}(K_0[0]) \oplus K_0[1], \\ K_1[2] &= \text{Enc}(K_0[1]) \oplus K_0[2], \\ K_1[3] &= \text{Enc}(K_0[2]) \oplus K_0[3]. \end{aligned}$$

Tổng quát:

$$K_i = \text{Encrypt}_{cfb}(K_{root}, IV, K_{i-1}),$$

với $i \in \mathbb{N}$ và $K_i \in \{0, 1\}^{256}$.

Giả sử

$$K_i = (K_i[0], K_i[1], K_i[2], K_i[3]) \in \{0, 1\}^{256}$$

với $K_i[j] \in \{0, 1\}^{64}$, $j = 0, 1, 2, 3$.

Theo mode CFB thì

$$\begin{aligned} K_i[0] &= \text{Enc}(IV) \oplus K_{i-1}[0], \\ K_i[1] &= \text{Enc}(K_{i-1}[0]) \oplus K_{i-1}[1], \\ K_i[2] &= \text{Enc}(K_{i-1}[1]) \oplus K_{i-1}[2], \\ K_i[3] &= \text{Enc}(K_{i-1}[2]) \oplus K_{i-1}[3]. \end{aligned}$$

Dãy $\{\gamma_n\}$ nhận được từ các giá trị K_i và K_0 . Ở đây, 4 bits đầu tiên của dãy $\{\gamma_n\}$ nhận được từ K_0 theo quy tắc

$$\gamma_j = \begin{cases} 0 & \text{nếu } K_0[j] = 0, \\ 1 & \text{trong trường hợp khác,} \end{cases}$$

trong đó $K_0[j]$ là khối 64 bits thứ j của K_0 , $\gamma_j \in \{0, 1\}$ với $j = 0, 1, 2, 3$.

Các bit tiếp theo được sinh tương tự theo quy tắc

$$\gamma_j = \begin{cases} 0 & \text{nếu } K_{j/4}[j \% 4] = 0, \\ 1 & \text{trong trường hợp khác,} \end{cases}$$

với $a \% b$ là số dư khi chia a cho b , $j \in \mathbb{N}$.

Chúng ta sẽ tìm quy luật đối với dãy $\{\gamma_n\}$:

Ta cần file `problem-7/skipjack.py` để thực hiện thuật toán Skipjack.

Tiếp theo, chúng ta thử mã hóa với K_{root} bất kì để quan sát xem có quy luật nào cho K_i không với file `problem-7/find_rules.py`.

Để thấy bản mã (dạng hex) có dạng sau:

```

0d16dceddfc805c1 47122ce9d3b7983a 02fd48e270666df7 28a8d35b20fbd167
00000000000000000000 c4a1be9a50040a49 a266cc2ec5ada370 59c24fdb72f24d49
0d16dceddfc805c1 83b3927383b39273 1374429b74bf2dc5 b2e5c38addfd9bbc
00000000000000000000 00000000000000000000 90c7d0e8f70cbfb6 b4975af551b93659
0d16dceddfc805c1 47122ce9d3b7983a 923a980a876ad241 f786cb6cd09fa275
00000000000000000000 c4a1be9a50040a49 32a11cc632a11cc6 a16a8af18673e3e8
0d16dceddfc805c1 83b3927383b39273 83b3927383b39273 1078044437616d5d
00000000000000000000 00000000000000000000 00000000000000000000 93cb9637b4d2ff2e
0d16dceddfc805c1 47122ce9d3b7983a 02fd48e270666df7 bb63456c94292e49
00000000000000000000 c4a1be9a50040a49 a266cc2ec5ada370 ca09d9ecc620b267
0d16dceddfc805c1 83b3927383b39273 1374429b74bf2dc5 212e55bd692f6492
00000000000000000000 00000000000000000000 90c7d0e8f70cbfb6 275cccc2e56bc977
0d16dceddfc805c1 47122ce9d3b7983a 923a980a876ad241 644d5d5b644d5d5b
00000000000000000000 c4a1be9a50040a49 32a11cc632a11cc6 32a11cc632a11cc6
0d16dceddfc805c1 83b3927383b39273 83b3927383b39273 83b3927383b39273
00000000000000000000 00000000000000000000 00000000000000000000 00000000000000000000

```

Ta chỉ quan tâm các khối 64 bits toàn các byte 0 nên bản mã dạng hex trên tương đương với

```

00000000000000000000 -----
00000000000000000000 00000000000000000000 -----
00000000000000000000 -----
00000000000000000000 00000000000000000000 00000000000000000000 -----
00000000000000000000 -----
00000000000000000000 00000000000000000000 -----
00000000000000000000 -----
00000000000000000000 00000000000000000000 00000000000000000000 00000000000000000000

```

Ở đây, ----- là khối khác 0. Thử mã hóa nhiều lần với K_{root} khác nhau thì ta thấy bản mã đều có dạng chung như trên.

Từ đó có thể nói dãy $\{\gamma_n\}$ không phụ thuộc vào khóa. Lúc này ta có thể sử dụng K_{root} tùy ý và sinh ra dãy $\{\gamma_n\}$.

File giải mã là `problem-7/solve.py`.

Kết quả giải mã là

EVALUATION OF THE SITUATION UNTIL NOW IT HAD TO BE EXPECTED THAT THE ENEMY WOULD TRY TO CROSS THE DANUBE WITH THE TROOPS ASSEMBLING AT VIDIN LOMPALANKA AND NEAR DRUTSCHUK WITH THE GOAL TO CUT OFF THE RAILWAYS BETWEEN ORSOVA AND CRAIOVA AND TO PUSH FORWARD TO BUCHAREST SINCE NOVEMBER 1 1918 IT APPEARS THAT THE SERBIAN ARMIES TOGETHER WITH THREE FRENCH DIVISIONS AREENGAGED IN AN ADVANCE TOWARD BELGRADE SEMENDRIA AND THE INTENDED ATTACK AT VIDI AND LOM PALANKA SEEMS TO HAVE BEEN ABANDONED IT HAS TO BE EXPECTED THE DEPLOYMENT OF STRONGER FORCES AT THE DANUBE SOUTH OF SVISTOV RUSTSCHUK SPECIALLY AFTER THE PEACE AGREEMENT OF TURKEY

HENCE IT IS MOST LIKELY THAT THE SERBIAN ARMIES REINFORCED BY THE FRENCH INTEND TO CROSS THE DANUBE NEAR BELGRADE SEMENDIA AND THE INTO SOUTHERN REPEAT SOUTHERN HUNGARY WHILE THE TASK OF THE FRENCH ARMY MARCHING SOUTH OF SVISTOV RUTSCHUK REMAINS THE OFFENSIVE TOWARD BUCHAREST IN CONJUNCTION WITH THIS OPERATION IT CANNOT BE RULED OUT THAT THE ROMANIAN FORCES COMING FROM MOLDAVIA OVER THE PASSES OF TOELGYESGIMES AND OITOS WILL INVADE TRANSYLVANIA THUS THE LINES OF COMMUNICATIONS IN THE REAR OF THE OCCUPATION ARMY WHICH HAVE UP TO NOW AS A RESULT OF IS THREATENED WITH ATTACK AND THE FURTHER OCCUPATION OF THE WALL ACHIAAS LAID DOWN IN ORDER FROM OKHEAD QUARTERS 2RMNR11161 WITHOUT DOUBT AND WITH REGARD TO THE AMMUNITION FOOD AND THE COAL STOCK SIT IS UNFEASIBLE IF THE GENERAL ARM ISTICE DOES NOT BECOME EFFECTIVE IN THE FORESEEABLE FUTURE IT IS SUGGESTED THAT THE OCCUPATION ARMY BE WITHDRAWN ATONCE FROM ROMANIA AND TOGETHER WITH THE GERMAN UNITS OF THE FIRST ARMY TO START THE MARCH TOUPPERSILES IA THROUGH HUNGARY APPROVAL IS REQUESTED SIGNED KMIAGROP

Ở cuối đoạn có ghi *SIGNED KMIAGROP* nên họ của tác giả là *KMIAGROP*.

Задача 8. Загадочная ошибка

Thêm một bài reverse engineering C++. Bài này có dùng thêm OpenSSL nhưng mình vẫn không biết làm.

Задача 9. Генератор гаммы на основе XSL-схемы

Câu hỏi

Chúng ta tạo thuật toán sinh khóa cho mã dòng

1. Hàm sinh của khóa là ánh xạ $F : \mathbb{F}_2^{64} \rightarrow \mathbb{F}_2^{64}$, được xây dựng dựa trên mô hình XSL, rất phổ biến trong việc thiết kế mã khóa.
2. Phép biến đổi không tuyến tính là S-box, dựa trên ánh xạ $S : \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8$. S-box có thể biểu diễn qua hoán vị sau

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	208	5	11	222	234	63	49	228	31	202	196	17	37	240	254	43
1	117	160	174	123	79	154	148	65	186	111	97	180	128	85	91	142
2	239	58	52	225	213	0	14	219	32	245	251	46	26	207	193	20
3	74	159	145	68	112	165	171	126	133	80	94	139	191	106	100	177
4	242	39	41	252	200	29	19	198	61	232	230	51	7	210	220	9
5	87	130	140	89	109	184	182	99	152	77	67	150	162	119	121	172
6	205	24	22	195	247	34	44	249	2	215	217	12	56	237	227	54
7	104	189	179	102	82	135	137	92	167	114	124	169	157	72	70	147
8	188	105	103	178	134	83	93	136	115	166	168	125	73	156	146	71
9	25	204	194	23	35	246	248	45	214	3	13	216	236	57	55	226
10	131	86	88	141	185	108	98	183	76	153	151	66	118	163	173	120
11	38	243	253	40	28	201	199	18	233	60	50	231	211	6	8	221
12	158	75	69	144	164	113	127	170	81	132	138	95	107	190	176	101
13	59	238	224	53	1	212	218	15	244	33	47	250	206	27	21	192
14	161	116	122	175	155	78	64	149	110	187	181	96	84	129	143	90
15	4	209	223	10	62	235	229	48	203	30	16	197	241	36	42	255

Ở bảng trên, phần tử ở hàng i và cột j ($0 \leq i, j \leq 15$) là giá trị S-box của $16i + j$.

1. Biến đổi tuyến tính L dựa trên LFSR, nghĩa là $L : \mathbb{F}_2^{64} \rightarrow \mathbb{F}_2^{64}$ với hệ số của đa thức đặc trưng được viết dưới dạng $0x12a9d9b8c0edf6e79$ (ở đây bit cao là hệ số của hạng tử bậc cao, most significant bit). Ở mỗi vòng, thanh ghi sẽ được khai báo bởi một trong các hằng số của vòng và đi qua 64 vòng.

Như vậy, quá trình sinh khóa có dạng

$$k^{(i)} = S(k^{(i-1)}) \oplus L^{64}(C^{(i)}),$$

với $k^{(i-1)}$ là khóa được sinh trước đó, $k^{(0)}$ là khóa đầu vào để sinh toàn bộ dãy khóa, và $C^{(i)}$ là hằng số của vòng, gồm

- 0x6ea276726c487ab8,
- 0x5d27bd10dd849401,
- 0xdc87ece4d890f4b3,
- 0xba4eb92079cbeb02,
- 0xb2259a96b4d88e0b,
- 0xe7690430a44f7f03,
- 0x7bcd1b0b73e32ba5,
- 0xb79cb140f2551504,
- 0x156f6d791fab511d,
- 0xeabb0c502fd18105,
- 0xa74af7efab73df16,
- 0x0dd208608b9efe06,
- 0xc9e8819dc73ba5ae,
- 0x50f5b570561a6a07,
- 0xf6593616e6055689,
- 0xadfb18027aa2a08.

Phương trình mã hóa có dạng

$$y^{(i)} = x^{(i)} \oplus k^{(i)}$$

với $y^{(i)}, x^{(i)} \in \mathbb{F}_2^{64}$.

Người truyền tin muốn kiểm tra độ an toàn của thuật toán sinh khóa. Anh ta mã hóa một phần của đoạn văn đầu trong tiểu thuyết "Alice in Wonderland" (tiếng Anh) và thu được bản mã là

3291999942924df2eb53323558623a949f5d90c4ba2cf0d9883c21ee0fc8e5348de9d8fece8b55693a5682396c33b57cdcaa6

Nhiệm vụ của bạn là phân tích thuật toán mã hóa và nếu có lỗ hổng, hãy khai thác nó và tìm khóa mã hóa.

[TODO] Giải

Задача 10. Регистровое преобразование

Câu hỏi

Eva chặn được một đoạn gamma được sinh dựa trên shift register (xem chi tiết ở file `problem-10/PQR.py`):

$$\gamma = 01000000000000010100100001101000111101100010001111.$$

Hãy tìm giá trị register ban đầu.

[TODO] Giải**Задача 11. Трудный PBKDF****Câu hỏi**

Việc trao đổi file dien ra trên kênh truyền. Để mã hóa file chúng ta dùng thuật toán sau dựa trên mật khẩu.

Đầu vào là mật khẩu *password* gồm 4 chữ số thập phân, và thông điệp *pt*.

Ta tính khóa *key* = SHA256(*password*) và initial vector là

$$IV = \text{PBKDF} - \text{HMAC} - \text{SHA256}(salt, 2^{30}, 128).$$

Sử dụng thuật toán mã khối AES256 với mode CBC và khóa *key*, initial vector *IV*, và padding bằng PKCS#7.

Chúng ta có một số file đã bị mã hóa. Biết rằng, qua kênh truyền có thể có nhiều nên file có thể chứa một số bytes đã bị hỏng. Một trong số đó chứa flag, hãy tìm nó.

[TODO] Giải**Kết luận**

Để viết sau.

4.7.2 TJCTF 2025

Đề và code giải mình để ở [đây](#).

alchemist-recipe**Đề bài****Giải**

Đây là một bài obfuscate code khiến chúng ta không hiểu được ý nghĩa của từng hàm, từng biến.

Đầu tiên hãy bắt đầu từ hàm `main`.

```
def main():
    try:
        with open("flag.txt", "rb") as f:
            flag_content = f.read().strip()
    except FileNotFoundError:
        print("Error: flag.txt not found. Create it with the flag content.")
        return

    if not flag_content:
        print("Error: flag.txt is empty.")
        return

    print(f"Original Recipe (for generation only): {flag_content.decode(errors='ignore')}")
    ←

    jellybean = glorbulate_sprockets_for_bamboozle(SNEEZE_FORK)
    encrypted_recipe = snizzle_bytegum(flag_content, jellybean)
```

(continues on next page)

(continued from previous page)

```

with open("encrypted.txt", "w") as f_out:
    f_out.write(encrypted_recipe.hex())

print(f"\nEncrypted recipe written to encrypted.txt:")
print(encrypted_recipe.hex())

if __name__ == "__main__":
    main()

```

Ở dòng tính `encrypted_recipe`, đầu vào là `flag_content` nên chúng ta có thể đoán được `snizzle_bytegum` là hàm encrypt và `jellybean` khóa vì hàm encrypt luôn có đầu vào là bản rõ và khóa.

Khi đó mình dùng tính năng Rename của IDE để thay `jellybean` thành `key`, và thay `snizzle_bytegum` thành `encrypt`.

Trước đó, `jellybean` được tính bởi hàm `glorbulate_sprockets_for_bamboozle` nên có thể đoán rằng đây là hàm sinh khóa và mình cũng đổi tên thành `keygen`.

Như vậy hàm `main` của mình có dạng

```

def main():
    try:
        with open("flag.txt", "rb") as f:
            flag_content = f.read().strip()
    except FileNotFoundError:
        print("Error: flag.txt not found. Create it with the flag content.")
        return

    if not flag_content:
        print("Error: flag.txt is empty.")
        return

    print(f"Original Recipe (for generation only): {flag_content.decode(errors='ignore')}")
    ↵

    key = keygen(SNEEZE_FORK)
    encrypted_recipe = encrypt(flag_content, key)

    with open("encrypted.txt", "w") as f_out:
        f_out.write(encrypted_recipe.hex())

    print(f"\nEncrypted recipe written to encrypted.txt:")
    print(encrypted_recipe.hex())

```

Tiếp theo chúng ta xét hàm `keygen` hoặc `encrypt`.

Hàm `keygen` có dạng

```

def keygen(blurbo):
    zing = []
    yarp = hashlib.sha256(blurbo.encode()).digest()
    zing['flibber'] = list(yarp[:WUMBLE_BAG])
    zing['twizzle'] = list(yarp[WUMBLE_BAG:WUMBLE_BAG+16])

```

(continues on next page)

(continued from previous page)

```

glimbo = list(yarp[WUMBLE_BAG+16:])
snorb = list(range(256))
sploop = 0
for _ in range(256):
    for z in glimbo:
        wob = (sploop + z) % 256
        snorb[sploop], snorb[wob] = snorb[wob], snorb[sploop]
        sploop = (sploop + 1) % 256
zing['drizzle'] = snorb
return zing

```

Thông thường đầu vào của hàm sinh khóa là một seed nào đó nên mình sẽ thay `blorbo` thành `seed`.

Hàm keygen trả về biến `zing` nên có thể thay `zing` thành `key`, chính là khóa mã hóa.

Biến `WUMBLE_BAG` có độ dài là 8 và cũng được sử dụng trong hàm `encrypt` nên đó chính là `BLOCK_SIZE`. Ngoài ra chúng ta không cần xem xét quá nhiều bên trong hàm `keygen`, chỉ cần biết rằng khóa có ba phần, phần đầu `key['flibber']` có 8 bytes, phần hai `key['twizzle']` có 8 bytes, và phần cuối `key['drizzle']` có 256 bytes. Do `seed` cố định nên `key` sinh ra cũng cố định. Để dễ nhìn hơn thì mình cũng đổi tên các biến khác nhưng các bạn không làm cũng không sao. Khi đó hàm `keygen` có dạng

```

def keygen(seed):
    key = {}
    key_hash = hashlib.sha256(seed.encode()).digest()
    key['flibber'] = list(key_hash[:BLOCK_SIZE])
    key['twizzle'] = list(key_hash[BLOCK_SIZE:BLOCK_SIZE+16])
    glimbo = list(key_hash[BLOCK_SIZE+16:])
    S = list(range(256))
    i = 0
    for _ in range(256):
        for z in glimbo:
            j = (i + z) % 256
            S[i], S[j] = S[j], S[i]
            i = (i + 1) % 256
    key['drizzle'] = S
    return key

```

Tiếp theo, xét hàm `encrypt`.

```

def encrypt(bubbles, jellybean):
    fuzz = BLOCK_SIZE - (len(bubbles) % BLOCK_SIZE)
    if fuzz == 0:
        fuzz = BLOCK_SIZE
    bubbles += bytes([fuzz] * fuzz)
    glomp = b""
    for b in range(0, len(bubbles), BLOCK_SIZE):
        splinter = bubbles[b:b+BLOCK_SIZE]
        zap = scrungle_crank(splinter, jellybean)
        glomp += zap
    return glomp

```

Như đã phân tích ở hàm `main`, tham số đầu của `encrypt` là bản rõ và tham số thứ hai là khóa. Như vậy mình đổi tên `bubbles` thành `plaintext` và `jellybean` thành `key`.

Chúng ta có thể đoán rằng `fuzz` là padding (PKCS7) vì `plaintext` được thêm vào các bytes cho đủ độ dài

BLOCK_SIZE (trước là WUMBLE_BAG).

Sau đó splinter là các khối bản rõ (độ dài 8) nên có thể đoán rằng scrungle_crank là hàm mã hóa từng khối, mình đổi tên thành encrypt_block.

Như vậy hàm encrypt của mình có dạng

```
def encrypt(plaintext, key):
    pad = BLOCK_SIZE - (len(plaintext) % BLOCK_SIZE)
    if pad == 0:
        pad = BLOCK_SIZE
    plaintext += bytes([pad] * pad)
    ciphertext = b""
    for b in range(0, len(plaintext), BLOCK_SIZE):
        block = plaintext[b:b+BLOCK_SIZE]
        zap = encrypt_block(block, key)
        ciphertext += zap
    return ciphertext
```

Cuối cùng, xét hàm encrypt_block.

```
def encrypt_block(dingus, sprockets):
    if len(dingus) != BLOCK_SIZE:
        raise ValueError(f"Must be {BLOCK_SIZE} wumps for crankshaft.")
    zonked = bytes([sprockets['drizzle'][x] for x in dingus])
    quix = sprockets['twizzle']
    splatted = bytes([zonked[i] ^ quix[i % len(quix)] for i in range(BLOCK_SIZE)])
    wiggle = sprockets['flibber']
    waggly = sorted([(wiggle[i], i) for i in range(BLOCK_SIZE)])
    zort = [oof for _, oof in waggly]
    plunk = [0] * BLOCK_SIZE
    for y in range(BLOCK_SIZE):
        x = zort[y]
        plunk[y] = splatted[x]
    return bytes(plunk)
```

Tham số thứ nhất của encrypt_block chính là bản rõ nên mình đổi tên dingus thành plaintext, tương tự tham số thứ hai là khóa nên mình đổi sprockets thành key.

Như mình đã nói ở trên, key gồm ba phần.

1. Phần cuối key['drizzle'] của khóa có độ dài 256 bytes nên có thể thấy zonked là S-box, mình đổi tên thành Sbox_key_third.
2. Phần thứ hai key['twizzle'] của khóa có 8 bytes nên mình đổi tên quix thành key_second.
3. Tiếp theo, splatted là XOR của phần cuối và phần thứ hai nên mình đổi tên thành third_X_second.
4. Phần đầu key['flibber'] có 8 bytes nên mình đổi tên wiggle thành key_first.
5. Tiếp theo, waggly là sắp xếp lại giá trị của key_first nên mình đổi tên thành key_first_srt.
6. Mình bỏ qua zort
7. Nhìn xuống một chút mình thấy hàm encrypt_block return bytes(plunk) nên có thể đoán plunk là bản mã, mình đổi tên thành ct.

Như vậy hàm encrypt_block có dạng

```

def encrypt_block(block, key):
    if len(block) != BLOCK_SIZE:
        raise ValueError(f"Must be {BLOCK_SIZE} wumps for crankshaft.")
    Sbox_key_third = bytes([key['drizzle'][x] for x in block])
    key_second = key['twizzle']
    third_X_second = bytes([Sbox_key_third[i] ^ key_second[i % len(key_second)] for i in range(BLOCK_SIZE)])
    key_first = key['flibber']
    key_first_srt = sorted([(key_first[i], i) for i in range(BLOCK_SIZE)])
    zort = [0] * BLOCK_SIZE
    ct = [0] * BLOCK_SIZE
    for y in range(BLOCK_SIZE):
        x = zort[y]
        ct[y] = third_X_second[x]
    return bytes(ct)

```

Khi đã viết lại tên biến thì chúng ta thực hiện ngược lại là giải mã được.

bacon-bits

Đề bài

Giải

Đề bài chuyển các kí tự của flag thành dãy 5 bit. Xét dãy bit song song với chuỗi `text`, nếu là bit 1 thì kí tự ở vị trí tương ứng của `text` là chữ hoa, ngược lại là chữ thường.

Một điều cần lưu ý là kí tự `i` và `j` đều được chuyển thành chuỗi bit 01000, tương tự, kí tự `u` và `v` đều được chuyển thành cùng chuỗi bit 10011. Do đó khi tìm ngược lại plaintext cần xét cả hai trường hợp.

close-secrets

Đề bài

Đề bài sử dụng Diffie-Hellman với các số nguyên tố p và g . Chọn $a \in [p - 10, p]$ và $b \in [g - 10, g]$.

Đề bài tính $u = g^a \pmod{p}$ và $v = g^b \pmod{p}$ để trao đổi khóa. Khóa chung là $u^b \equiv v^a \pmod{p}$.

Khi đó `xor_key_str` là SHA256 của khóa chung và được dùng trong `dynamic_xor_encrypt`, cụ thể là xor từng bytes của flag với `xor_key_bytes`.

Tiếp theo, hàm `encrypt_outer` lấy ASCII của từng kí tự, cộng với `key_offset` và nhân với `key`. Do `key` cố định (khóa chung) nên `key_offset` cũng cố định.

Giải

Chúng ta chỉ cần bruteforce a và b rồi làm ngược lại hai hàm trên.

dotdotdotv2

Đề bài

Đề bài chuyển một chuỗi dài ơi là dài thành dãy bit (mỗi kí tự tương ứng 8 bits) và lập ma trận có 64 cột, còn số hàng tùy thuộc độ dài cuối cùng. Ta gọi ma trận này là flag.

Đề bài sinh khóa là ma trận 64×64 với các phần tử thuộc đoạn $[0, 0xdeadbeef]$. Ta gọi ma trận này là key.

Như vậy kết quả là phép nhân ma trận $\text{res} = \text{flag} \cdot \text{key}$.

Giải

Độ dài của **filler** là 500 bytes, tương ứng 4000 bits. Ma trận 64×64 sẽ cần 4096 bits. Chúng ta biết format flag là `tjctf{` nên thêm vào 4 bytes nữa là đủ để có ma trận 63×64 .

Chúng ta sẽ chuyển tất cả tính toán sang \mathbb{F}_2 thay vì để nguyên các số lớn kia.

Khi đó ta gọi ma trận 63×64 là `ff`, và dùng 63 dòng đầu của ma trận kết quả `res` ta sẽ tìm được ma trận `key`.

Ở đây `ff` có hạng là 55 nên ma trận `key` tìm được không phải duy nhất, tệ hơn nữa là `key` không khả nghịch. Do đó chúng ta sẽ dùng giả nghịch đảo (pseudo inverse) của `key` để tính `flag`.

Kết quả khá bất ngờ, mình nhận được chuỗi `b'In cybersecuritY`. Như vậy mình chỉ cần sửa kí tự `R` thành `r` nữa là xong. Để ý rằng nếu xét 8 kí tự, tương ứng 64 bit, thì chỉ cần cộng 1 vào bit thứ 58.

double-trouble**Đề bài**

Mã hóa AES hai lần, mỗi lần mã hóa với nửa khóa cố định và nửa khóa còn lại là một trong $4^8 = 65536$ trường hợp.

Giải

Mình sử dụng OpenMP và tối ưu `-O2` để bruteforce khóa tương ứng.

pseudo-secure**Đề bài**

Sinh số ngẫu nhiên.

Giải

Sử dụng module `extend_mt19937_predictor` ở [ExtendMT19937Predictor](#).

seeds**Đề bài**

Sử dụng seed để sinh ra khóa cho thuật toán AES.

Giải

Seed "có vẻ" cố định và luôn cho ra cùng khóa, bất kể thời gian.

theartofwar**Đề bài**

Đề bài sinh e modulus n_i và mã hóa cùng bản rõ m với $c_i = m^e \bmod n_i$.

Giải

Sử dụng định lí số dư Trung Hoa (broadcast attack) và tính căn bậc e để tìm lại m .

Writeup tới đây là hết. Cám ơn các bạn đã đọc.

Bên lề sách của Fermat

5.1 Ngoài lề

5.1.1 Lý thuyết trò chơi

Nhập môn lý thuyết trò chơi

Phần này mình tham khảo từ quyển Tài liệu chuyên tin học quyển 3 [39].

Một số khái niệm

1. *Trạng thái* (hay *state*) (trong sách Tài liệu chuyên tin gọi là *vị trí* nhưng để phù hợp với nhiều bài viết khác như máy Turing, máy trạng thái thì gọi là *trạng thái* sẽ hợp lý hơn) là tập hợp thông tin về trò chơi tại từng thời điểm. Ví dụ như vị trí các quân cờ trên bàn cờ vua tại nước đi thứ 1, thứ 2, ...

Ta chia trạng thái làm hai loại:

- Những trạng thái chứa khả năng dẫn tới chiến thắng được gọi là trạng thái có lợi.
 - Ngược lại thì gọi là trạng thái không có lợi.
2. *Luật chơi* (hay *rule*) là những quy định cho phép người chơi thực hiện các biến đổi chuyển trò chơi từ trạng thái này sang trạng thái khác. Ví dụ như cách di chuyển của mỗi quân cờ trong cờ vua (đi như thế nào, ăn quân như nào, ...).

Một bước đi hợp lệ là phép biến đổi theo đúng luật chơi.

Trạng thái kết thúc là trạng thái từ đó không thể di chuyển tiếp.

3. Trò chơi đối kháng là trò chơi hai người, khi một người có lợi thế thì người kia gặp bất lợi.

Có hai loại trò chơi đối kháng

- Trò chơi có thông tin đầy đủ - người chơi biết mọi thông tin về trạng thái hiện tại của đối phương. Ví dụ trong cờ vua, mỗi người chơi biết vị trí của tất cả quân cờ của đối thủ.
 - Trò chơi có thông tin không đầy đủ - người chơi biết một phần hoặc không biết thông tin về trạng thái hiện tại của đối phương. Ví dụ chơi bài tây thì mỗi người không biết người khác có những quân bài gì. Ví dụ khác là trò chơi bầu cua, người đặt cược không biết được thông tin của ba quân súc sắc khi đặt cược.
4. Trong mọi trò chơi, quá trình chơi có thể được biểu diễn dưới dạng cấu trúc cây có hướng (đôi khi là đồ thị có hướng) gọi là *cây trò chơi*. Trong đó:

- *Nút* biểu diễn trạng thái.
- *Nút gốc* là trạng thái khởi đầu.
- *Nút lá* là trạng thái kết thúc.
- *Cung* (i, j) thể hiện bước đi hợp lệ từ trạng thái i tới trạng thái j .

Trò chơi tổ hợp cân bằng

Mô tả

Trò chơi tổ hợp cân bằng là trò chơi đối kháng thỏa mãn các điều kiện:

- có hai người chơi (từ đây về sau, người đi đầu kí hiệu là A và người đi sau là B);
- có một tập hữu hạn các trạng thái có thể xảy ra của trò chơi, kí hiệu là X ;
- có luật chơi Q . Quy luật chơi áp dụng cho hai người chơi là cân bằng, nghĩa là mỗi người chơi tới lượt mình đều có quyền chọn một phép di chuyển hợp lệ tùy ý;
- hai người chơi lần lượt, mỗi lần thực hiện một phép di chuyển hợp lệ;
- trò chơi kết thúc khi đạt trạng thái kết thúc;
- nếu trò chơi không bao giờ kết thúc thì có thể sử dụng rút thăm, hoặc giới hạn số lượng bước đi tối đa, hoặc cầu hòa.

Tập P . Tập N

1. Tất cả các trạng thái kết thúc đều thuộc P .
2. Từ mỗi trạng thái thuộc N luôn có ít nhất một di chuyển tới trạng thái thuộc P .
3. Từ mỗi trạng thái thuộc P , mọi di chuyển đều tới trạng thái thuộc N .

Như vậy ta cần xây dựng **thuật toán giành thắng** cho đấu thủ A khi **ban đầu A nhận trạng thái thuộc N** là: đấu thủ A luôn di chuyển tới các trạng thái thuộc P để ép đối thủ B chỉ có thể di tới trạng thái thuộc P .

Tổng Nim. Trò chơi Nim

Để dễ xác định tập P và N ta dùng khái niệm tổng Nim.

Tổng Nim là phép XOR hai số nguyên không âm.

Trò chơi Nim chuẩn là trò chơi có ba cọc lần lượt chứa x_1 , x_2 và x_3 quân. Hai người chơi lần lượt tạo các bước di chuyển quân với quy tắc: chọn một cọc tùy ý còn quân và lấy bớt một số quân từ cọc này. Người thắng là người lấy được quân cuối cùng.

Chúng ta có thể sử dụng tổng Nim để xác định tập P với định lí Bouton.

❶ Theorem 17 (Định lí Bouton)

Mỗi trạng thái (x_1, x_2, x_3) trong trò chơi Nim ba cọc là trạng thái P khi và chỉ khi tổng Nim $x_1 \oplus x_2 \oplus x_3 = 0$.

Từ định lí Bouton ta xây dựng chiến thuật giành thắng như sau.

Giả sử trạng thái hiện tại (cho trò chơi Nim với n cọc) là (x_1, \dots, x_n) , ứng với tổng Nim

$$g = x_1 \oplus x_2 \oplus \dots \oplus x_n > 0.$$

Ta có thể chứng minh tồn tại phần tử x_i sao cho $x'_i = (g \oplus x_i) \leq x_i$. Đây là cách đi giành thắng với mục tiêu là giảm cọc thứ i từ x_i quân thành x'_i quân.

❶ Example 26

Ví dụ đơn giản nhất với hai cọc (x_1, x_2) . Như vậy trạng thái thuộc P tương đương với $x_1 \oplus x_2 = 0$, hay $x_1 = x_2$. Từ đây suy ra trạng thái thuộc N khi $x_1 \neq x_2$.

Giả sử A là người chơi trước. Theo thuật toán giành thắng ở trên thì nếu A ở trạng thái thuộc N thì A luôn thắng. Nghĩa là khi trạng thái là (x_1, x_2) với giả sử $x_1 < x_2$ (trường hợp $x_1 > x_2$ tương tự) thì A chỉ cần lấy đi ở cọc thứ hai một lượng $x_2 - x_1$ để hai cọc có số quân bằng nhau. Cuối cùng thì trạng thái kết thúc là $(0, 0)$ và A sẽ chiến thắng.

5.1.2 Một số đồ thị hàm số sưu tầm

Để vẽ trái tim như ở [đây](#) ta dùng phương trình

$$y = x^{2/3} + 0,9(3,3 - x^2)^{1/2} \cdot \sin(m \cdot \pi \cdot x)$$

với $m = 6,50$.

5.1.3 Về một vị giáo sư và định lý Godel

❶ Cảnh báo

Mình sẽ không sử dụng từ ngữ khó nghe nhưng nội dung sẽ gây khó chịu. Độc giả cân nhắc trước khi xem!!!

Dạo gần đây mình xem một loạt video trên Youtube của kênh *Nhận thức mới* về định lí bất toàn của Godel do giáo sư Phạm Việt Hưng (PVHg) trình bày.

Đây là một chủ đề khá thú vị và mới mẻ với mình. Sau khi xem xong series video và đọc các bài viết trên blog cá nhân của giáo sư thì mình vừa thấy buồn cười vừa thấy khó chịu. Do đó mình quyết định viết bài này.

Hệ quả định lí Godel

Theo giáo sư, việc một sự vật tác động lên chính nó là không thể xảy ra. Điều này có thể xem là hệ quả của định lí Godel.

Giáo sư đưa ra ví dụ về hạt Higgs ở tập 2 của series. Mình không biết hạt Higgs là hạt gì vì kiến thức phổ thông của mình với vật lí chỉ dừng lại ở proton, electron và neutron. Giáo sư giải thích rằng, hạt Higgs là hạt truyền khối lượng cho các hạt khác lớn hơn, nhưng tiếp tục đặt vấn đề là hạt nào truyền khối lượng cho hạt Higgs? Theo định lí Godel, hạt Higgs không thể truyền khối lượng cho chính nó, vì khi đó hạt Higgs là một hệ tự quy chiếu. Như vậy đưa tới hệ quả là Lý thuyết về mọi thứ (TOE, Theory Of Everything) vẫn đang bế tắc, suy ra việc lí giải nguồn gốc khối lượng các vật cũng không thể đạt được.

Tiếp theo, ở tập 3, giáo sư nói về việc trí tuệ nhân tạo (AI, Artificial Intelligence) không thể trở nên giống con người. Con người chúng ta sử dụng ý thức, là bộ não, để tạo ra một "bộ não" máy móc. Như vậy, cũng theo định lí Godel, vì con người không thể hiểu chính bộ não của bản thân, thì làm sao có thể tạo ra một "bộ não" giống bản thân, có trực giác, có cảm xúc, có ý thức?

Hai vấn đề trên cho thấy một hệ quả quan trọng của định lí Godel do giáo sư trình bày

Một hệ tự quy chiếu hoặc mâu thuẫn, hoặc không thể kiểm chứng được tính đúng sai.

Đến đoạn này mình vẫn nghĩ mọi thứ bình thường, nhưng hai tập tiếp theo về việc tìm nguồn gốc sự sống của giáo sư làm mình thấy rất khó hiểu.

Trước đó, giáo sư có nhiều bài viết trên trang cá nhân có nội dung giống hai tập tiếp theo trên Youtube cũng về nguồn gốc sự sống. Nói đơn giản thì giáo sư phê phán thuyết tiến hóa của Darwin và đưa ra vấn đề về nguồn gốc sự sống, thậm chí gọi đó là "sự dối trá". Tới đây mình có một thắc mắc. Nếu hạt Higgs không thể truyền khối lượng cho chính nó, con người không thể sử dụng bộ não của bản thân để tạo ra "bộ não" nhân tạo (AI), thì sự sống cũng không thể giải thích nguồn gốc sự sống?

Logic ở đây là, theo định lí Godel, một lý thuyết giải thích nguồn gốc mọi thứ, là không tồn tại đối với hệ tự quy chiếu. Vì vậy việc truy tìm nguồn gốc của một hiện tượng nào đó luôn đi tới bế tắc.

Nguồn gốc khối lượng là từ hạt nào? Nếu hạt Higgs truyền khối lượng cho các hạt khác thì hạt nào truyền khối lượng cho nó? Vấn đề ở đây là bản thân hạt Higgs không thể truyền khối lượng cho chính nó.

Nguồn gốc tư duy của AI là từ tư duy con người. Nhưng con người không thể giải thích bộ não của bản thân, hay nói cách khác là nguồn gốc ý thức và tư duy của bản thân, thì làm sao tạo ra "bộ não" nhân tạo cũng có ý thức và tư duy giống bản thân?

Nguồn gốc sự sống là từ một sự sống trước đó mà thành (tiến hóa, thoái hóa, ...). Như vậy, sự sống có thể giải thích nguồn gốc của sự sống không? Thuyết tiến hóa của Darwin nói rằng sự sống đều bắt nguồn từ một "gốc" (gốc của cây sự sống). Nhưng qua lời giáo sư thì "gốc" đó không tồn tại. Giáo sư đề cập tới một giải thích khác về thuyết tiến hóa của Darwin là sự sống bắt đầu từ một "thảm" (thảm cỏ của sự sống). Trong đó sự sống bắt nguồn từ nhiều vị trí trong thảm cỏ đó, không phải từ một gốc đơn thuần như thuyết của Darwin. Thậm chí như vậy vẫn không làm hài lòng các nhà khoa học (thực ra là giáo sư). Như vậy, theo logic của mình ở trước, mọi thuyết tiến hóa giải thích nguồn gốc sự sống đều sai, theo những ví dụ giáo sư đưa ra, và thông qua định lí Godel?

Theo trải nghiệm cá nhân của mình, việc phê phán hay châm biếm thuyết tiến hóa của Darwin không phải điều gì mới mẻ. Từ nhiều năm trước mình đã từng nghe một thầy cà khịa kiểu "thế quái nào khỉ và người lại có cùng tổ tiên". Mình không biết nhiều về các thuyết tiến hóa, nhưng video của giáo sư thật sự cung cấp cho mình nhiều kiến thức và bài học. Bài học quan trọng nhất mình rút ra là:

Chúng ta có thể ứng dụng định lí Godel để bác bỏ những rất nhiều cỗ găng của các nhà khoa học.

Mình không biết giáo sư có tôn sùng quá mức định lí Godel hay không, nhưng việc bác bỏ nỗ lực của người khác mà không đưa ra một lý thuyết tiến bộ hơn hay có chứng minh vững chắc mà chỉ dựa vào một định lí TOÁN HỌC để giải thích tính TRIẾT HỌC thì mình thấy khá buồn cười.

Về Cantor

Phần này không nói về tính đúng sai cả về triết học lẫn toán học. Mình chỉ có ý kiến về cách trình bày của giáo sư.

Về bài viết Về Cái Bất khả Quyết định.

Trong bài viết, giáo sư nhắc rằng "người đời gán cho Cantor là điên rồ", nhưng sau đó giáo sư lại mở ngoặc "thực tế cuối cùng ông đã mắc bệnh thần kinh". Ở đây mình rất khó chịu vì cách viết không đầu đuôi (không nói lý do Cantor bị bệnh thần kinh) dễ khiến người khác nghĩ ông ấy đưa ra quá nhiều ý kiến kì lạ tới nỗi bị thần kinh.

Câu chuyện thực sự là khi cỗ găng chứng minh giả thuyết Continuum mà không có tiến triển (mà giáo sư cũng đề cập trong bài viết) thì Cantor cũng đang bị chèn ép rất mạnh bởi phe chống đối Lý thuyết tập hợp của ông (do Kronecker và Poincare đứng đầu). Hậu quả là những bài báo của ông đã không thể công bố trên các tạp chí uy tín do quyền lực ngầm từ phe Kronecker và Poincare, nên ông chỉ có thể đăng trên các

tạp chí thường, uy tín thấp. Điều này khiến ông không thể có vị trí công việc tốt tại các đại học lớn ở nước Đức thời bấy giờ khiến cuộc sống khó khăn.

Ngoài ra một bi kịch đã xảy ra với Cantor là đứa con trai út mà ông rất yêu thương đã qua đời. Quá nhiều biến cố ập đến khiến tinh thần Cantor rơi vào hỗn loạn và cuối cùng là bị bệnh.

Mình không biết giáo sư có thật sự biết về cuộc đời Cantor hay không trước khi viết những dòng đó. Mình rất hy vọng là có vì văn phong như vậy không thể biết giáo sư nói đùa hay nói thật.

Một lần nữa, việc tôn sùng định lí Godel quá mức là niềm tin của giáo sư và mình không bận tâm. Nhưng Cantor là idol của mình, và những người động tới idol của bản thân thì ... các bạn biết rồi đó. (^_~)

Viết tối đây mình bỗng dung nhớ lại một phân cảnh trong phim Pirates of the Caribbean (Cướp biển vùng Caribbean) phần 5. Trong đó, khi nữ chính gặp một nhà thiên văn và nói lên ý tưởng của mình, thì nhà thiên văn đã cười và nói lại nữ chính là đồ điên. Nữ chính cũng không vui, nói là thiên văn là đồ ngu.

Trong bài viết trên, giáo sư PVHG nói Cantor là đồ điên. Đáng tiếc là Cantor đã qua đời từ lâu nên Cantor không thể gọi giáo sư là đồ ng... được (cũng có thể Cantor không muốn nói?). Minh thì không ở vị thế có khả năng nói giáo sư ng... (nhưng mình rất muốn) (^^).

Thương thay Cantor, đời sau, người thì xem thường và chửi ông, người muốn bảo vệ ông thì lực hèn sức mọn!!!

Về sự vô hạn

Vẫn là bài viết Về Cái Bất khả Quyết định.

Giáo sư trích dẫn hai câu nói: "Con người ta đã nghĩ ra khái niệm vô hạn, nhưng chẳng có gì là vô hạn trên thế gian này, ngoài sự ngu xuẩn của con người", "Chỉ có hai thứ vô hạn: vũ trụ và CÁI NGU của con người; tôi không chắc về cái thứ nhất". Câu sau thì chắc các bạn cũng khá quen thuộc rồi vì là câu nói (hay được mang ra làm meme) của Einstein.

Mình không biết quan niệm của giáo sư về sự vô hạn (hay Lý thuyết tập hợp) của Cantor là gì, nhưng hai câu trích dẫn trên cho thấy một sự châm biếm về điều này. Mình xin phép phản biện lại như sau.

Ví dụ đầu tiên về sự vô hạn bắt nguồn từ phép **đếm**.

Con người từ xa xưa đã biết đếm. Hết 1, rồi 2, rồi lại 3, cứ thế cho đến mãi mãi. Từ đó, tập hợp vô hạn đầu tiên xuất hiện, tập \mathbb{N} . Vì các con số cứ xuất hiện một cách tự nhiên, số sau là số trước cộng thêm 1, nên ta gọi là *tập hợp số tự nhiên*.

Như vậy có thể thấy, bản thân tập hợp vô hạn TỰ NHIÊN XUẤT HIỆN chứ không phải là ảo tưởng của các nhà toán học.

Tiếp theo, con người dần chấp nhận sự xuất hiện của số âm. Từ gốc 0, làm ngược lại quá trình trên, ta trừ 1 thay vì cộng. Khi đó ta cũng có $-1, -2$, và cứ như vậy tới mãi mãi. Ta đã xây dựng xong tập vô hạn thứ hai, tập số nguyên \mathbb{Z} .

Trong bộ sách kinh điển *Elements* của Euclid, ông nói rằng: "Một phần luôn nhỏ hơn toàn bộ". Nhìn lại hai anh bạn \mathbb{N} và \mathbb{Z} thì chúng ta thấy rõ ràng rằng \mathbb{N} là một phần của \mathbb{Z} , hay nói kiểu toán học là **tập hợp con**. Như vậy phải chăng \mathbb{N} có ít "phần tử" hơn \mathbb{Z} ? Nếu chúng ta sống ở thời Euclid, thì việc này được cho là "hiển nhiên", nhưng tới thời Cantor thì cái "hiển nhiên" đó mới được kiểm chứng.

Ví dụ thứ hai là nghịch lí của Zeno.

Nghịch lí này khá nổi tiếng và mình đã có giới thiệu trong bài viết về các tập vô hạn rồi. Đại ý là nếu anh hùng Achilles trong thần thoại Hy Lạp chạy đua với con rùa nhưng xuất phát sau thì không bao giờ bắt kịp được con rùa.

Nghịch lí này cho thấy người thời xưa có những quan niệm chưa đúng về sự vô hạn, chưa có các khái niệm như vô cùng nhỏ, vô cùng lớn.

Ví dụ thứ ba về sự vô hạn bắt nguồn từ số vô tỉ.

Pythagoras đã quá quen thuộc với chúng ta về định lí mang tên ông cho tam giác vuông. Thời đó, khi cho hai cạnh góc vuông có độ dài bằng 1 thì độ dài cạnh huyền là một số vô tỉ $\sqrt{2}$. Tuy nhiên các số vô tỉ, ngoài phần thập phân không tuần hoàn ra, trong nhiều trường hợp còn có một điểm khó chịu khác là **ta không thể biểu diễn nó**, thậm chí tệ hơn là **ta không biết tới sự tồn tại của nó**.

Tỉ lệ giữa chu vi hình tròn với hai lần bán kính của nó, cho ta số π . Giới hạn đặc biệt $\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n$ cho ta số e . Số π và e là hai ví dụ về **số siêu việt**, còn những số có thể biểu diễn qua các phép tính cộng, trừ, nhân, chia và khai căn được gọi là **số đại số**. Vấn đề là số lượng số siêu việt nhiều hơn số đại số rất nhiều và chính chúng mới giúp "làm đầy" trục số thực \mathbb{R} .

Ý nghĩa của việc "làm đầy" ở đây là **tính liên tục** của tập hợp số thực. Nhờ có tính liên tục, chúng ta mới có thể đạo hàm và tích phân, hai công cụ mạnh mẽ cho phép khảo sát các hàm số. Minh cần lưu ý rằng nếu hàm số có đạo hàm tại một điểm thì nó chắc chắn liên tục tại điểm đó, nhưng ngược lại chưa chắc, liên tục chưa chắc có đạo hàm. Nói cách khác tính liên tục là nền tảng cho những phép tính khó hơn. Nếu hàm số không liên tục tại một điểm, thì việc đạo hàm là không thể, nên quay xe tìm cách khác thì hơn!!!

Vấn đề là nếu các giáo sư chối bỏ sự vô hạn của tập hợp số thực, vậy các giáo sư có thể không dùng những công cụ mạnh mẽ của toán học như đạo hàm, tích phân trong nghiên cứu vật lí được không?

Trùng hợp thay, các bài toán về mật mã học đang bế tắc vì làm việc với các tập rời rạc, nơi không có tính liên tục, thậm chí còn hữu hạn. Các bạn, thậm chí giáo sư, có thể nói mình rằng: "Tập hữu hạn còn không xử lý được mà muốn dây vào tập vô hạn". Trên thực tế, tập vô hạn như \mathbb{R} và \mathbb{C} cho chúng ta những công cụ mạnh mẽ: giới hạn, sau đó là tính liên tục, sau đó là vi tích phân. Khi mất đi những công cụ đó, mà các tập rời rạc là một ví dụ, thì bài toán trở nên khó khăn kinh khủng. Anh Vũ Hữu Tiệp (blog Machine Learning cơ bản) cũng có nói rằng "còn đạo hàm được là còn hy vọng". Như vậy mới thấy, tập vô hạn cho chúng ta nhiều điều kiện thuận lợi, những công cụ mạnh mẽ để khảo sát chúng, nhưng sẽ gây ra nghịch lý nếu không biết cách "chế ngự" chúng như trường hợp của nghịch lý Zeno.

Tổng kết lại, từ việc đếm rất bình thường, cho tới những tập hợp có phần tử khó nhăn, thì sự vô hạn sẽ đưa chúng ta tới nhiều trường hợp dở khóc dở cười. Như mình đã nói ở trên, người ng... hay nói những ý tưởng kì quặc là điên, nhưng tiếc thay người "điên" đó đã mất nêu không thể bắt lại người ng... Thật đáng tiếc!!!

Về giáo dục

Nếu đã tới bậc giáo sư thì chắc hẳn công bố khoa học phải rất nhiều. Ngoài ra, nghiệp vụ sư phạm là một phần bắt buộc để thành giáo sư nên ý kiến của giáo sư PVHg về giáo dục rất đáng để tâm. Tuy nhiên, theo mình, có một số điểm chưa thỏa đáng.

Giáo sư có bài viết về việc Ấn Độ đã loại bỏ thuyết tiến hóa của Darwin ra khỏi chương trình phổ thông. Đối với giáo sư, đây là tin mừng. Mừng vì các thế hệ học sinh, sinh viên tiếp theo sẽ không bị thuyết tiến hóa dối trá ấy (theo lời giáo sư) làm lu mờ nhận thức.

Tuy nhiên, đối với học sinh thuộc thế hệ sau như mình thì mình nhận thấy một số vấn đề trong cách nghĩ của giáo sư.

Một là, nếu loại bỏ thuyết tiến hóa của Darwin thì chắc chắn phải có một nội dung khác (ví dụ, một thuyết tiến hóa khác) bù vào phần đó. Nói cách khác, khôi lượng chương trình học KHÔNG THAY ĐỔI. Trong quyển sách *Lời than vãn của một nhà toán học* của Paul Lockhart, trên trường các học sinh thấy tiết toán ... chán òm. Hầu hết chúng mình thấy chán vì những kiến thức khô khan, học thuộc lòng, công thức xào đi nấu lại của những môn khoa học tự nhiên. Như vậy, vấn đề là nếu thay thuyết tiến hóa của Darwin thành một nội dung khác, mà theo giáo sư PVHg là giúp các thế hệ sau "không bị lừa dối" nữa, thì kết quả có thực sự màu hồng vậy không?

Bao nhiêu học sinh sau này sẽ tiếp tục học tập hay làm việc trên các lĩnh vực liên quan sinh học? Tiếp theo, khi nghiên cứu sâu về sinh học, điều gì đảm bảo các bạn sẽ vẫn tiếp tục "bị lừa" như giáo sư nói? Rõ ràng khi trình độ các bạn tăng lên, khi các bạn tìm tài liệu, thì các bạn sẽ nhận ra có nhiều thuyết tiến hóa đang tồn tại, và thuyết của Darwin mà ngày ấy được học ở trường chỉ là một ví dụ.

Việc giáo sư bác bỏ Darwin là một ví dụ cho **tư duy địch-ta**, trong đó địch sai, còn ta đúng. Darwin sai, nên vứt Darwin đi mà học cái khác. Đây là một lỗi tư duy nguy hiểm vì nó khiến sự đa dạng, nhiều chiều trong tư duy bị mất đi. Thay vào đó, sao chúng ta không bắt đầu với những câu hỏi đơn giản như: điểm nào trong thuyết tiến hóa của Darwin không hợp lý (ở đây giáo sư đã giải thích, chính là cây sự sống, tồn tại một cái "gốc" của sự sống). Tiếp theo, câu hỏi mở rộng hơn, chẳng hạn như: có thể "chữa" chỗ không hợp lý này không (giáo sư cũng có đề cập tới thảm cỏ sự sống). Giáo sư đã nghĩ được tới vậy, tại sao lại cố xúy các bạn trẻ bỏ đi Darwin? Thay vào đó, chúng ta có thể xem xét tới các thuyết tiến hóa khác, điểm mạnh, điểm yếu, những chỗ chưa hợp lý, những vùng xám trong mỗi thuyết tiến hóa? Khoa học phát triển trên nguyên lý **Đứng trên vai người đi trước** (khẩu hiệu của Google Scholar cũng là **Stand on the shoulders of giants**). Mình nghĩ rằng việc xem xét các ý tưởng khác nhau sẽ tốt hơn việc cứ chê bai mãi một lý thuyết.

Giáo sư cho rằng giáo dục hiện tại trên nhiều nước đang giết chết tư duy, không đi vào bản chất. Đối với đứa mới qua những ngày tháng cùi nhخnh như mình, ký ức của 12 năm phổ thông vẫn còn cháy bỏng lầm. Nếu môn nào mà mình cũng hiểu rõ bản chất, chắc nǎo mình sẽ vỡ làm đôi mắt. (^_^\^)

Einstein nói rằng: "Ai cũng là thiên tài, nhưng nếu bạn bắt con cá leo cây thì nó suốt đời sẽ nghĩ mình đần độn". Thiên tính của mỗi người mỗi khác. Có người học tốt khoa học tự nhiên, có bạn lại học tốt khoa học xã hội. Kể cả mình thiên về khoa học tự nhiên, thì mình cũng chỉ hiểu bản chất toán, còn vật lí, hóa học, sinh học thì mình bó tay. Vấn đề là, hiểu bản chất toán để làm gì nếu không gấp lại toán trên đường đời? Vi tích phân xuất phát từ các bài toán chuyển động cơ học trong vật lí. Nhưng việc hiểu rõ ý nghĩa vật lí của đạo hàm và tích phân có tác dụng gì nếu các bạn thi đại học khối C?

Thứ nhất, giáo dục cung cấp những kiến thức cơ sở cho mọi người ở tất cả môn để làm nền tảng cho những hướng đi trong tương lai. Mình lấy ví dụ là môn toán. Cái quan trọng nhất chúng ta cần học là **TÍNH TOÁN**. Đi chợ, biết tính tiền. Đi học các ngành kĩ thuật, biết tính toán số liệu. Đi học bác sĩ, biết tính toán lâm sàng (số ca mắc bệnh, tỉ lệ chữa khỏi, ...). Mình không thấy nhiều trường hợp thực sự cần hiểu bản chất vi tích phân, mà thực tế là cần **CÁCH TÍNH ĐẠO HÀM** và **TÍCH PHÂN**.

Thứ hai, giáo sư cũng nói đúng về một vấn đề (may quá ít ra vẫn có phần mình đồng tình với giáo sư). Đó là câu chuyện về tuổi của vị thuyền trưởng. Khi toán học xa rời thế giới thực, mà điều dễ thấy nhất chính là đơn vị tính, thì sẽ xảy ra câu chuyện dở khóc dở cười.

Câu chuyện về tuổi của vị thuyền trưởng là một bài toán đơn giản của bậc tiểu học. ví dụ như cho số lượng quả táo trên thuyền là 15, cho số lượng thủy thủ trên thuyền là 10, hỏi tuổi của vị thuyền trưởng là bao nhiêu? Các đại lượng không hề cùng đơn vị đo, nhưng những bạn học sinh ở Pháp những năm đầu thế kỷ 20 đã không ngần ngại tính tuổi của vị thuyền trưởng là $15 + 10 = 25$. Khi gán các đơn vị vào chúng ta sẽ thấy

$$15 \text{ (quả táo)} + 10 \text{ (thủy thủ)} = 25 \text{ (tuổi)}.$$

Rất may là chương trình hiện nay ở Việt Nam đã giáo dục kĩ khi còn ở bậc tiểu học. Những phép tính chu vi, diện tích, thể tích luôn được kiểm tra cẩn thận về đơn vị đo. Điều này vẫn được duy trì trong chương trình ở cấp 2 và 3.

Tuy nhiên, vấn đề là nếu bó buộc vào trong các bài toán vật lí thì cần gì phải học môn toán nữa? Như vậy chẳng phải đang bó buộc tư duy lại sao? Rằng mọi thứ phải liên quan đến bài toán cụ thể, vẫn đề cụ thể, mà không quan tâm đến tính trừu tượng, sự tưởng tượng, tính sáng tạo? Như vậy khi những hiện tượng mà chúng ta không thấy bằng mắt thường như vũ trụ, lượng tử, làm sao chúng ta biết tới sự tồn tại của nó? Thần Athena sẽ đưa lời sấm cho chúng ta chẳng?

Tổng kết lại, giáo dục là một vấn đề có sự tham gia của rất nhiều thành phần và tác động đến nhiều người nên rất khó để phù hợp cho tất cả các bên. Giáo sư nghĩ theo cách của người học giỏi, còn mình thì dốt nên chỉ thấy phương án của giáo sư tạo thêm áp lực. (^_^\^)

Chủ nghĩa duy lý và tư duy chủ quan

Trong tập 3 (?) giáo sư PVHg có châm biếm một câu nói của Hilbert. Hilbert nói rằng "Tôi nghĩ là ...". Chúng ta thấy rằng đây là tư duy chủ quan của Hilbert, và vì ông là tượng đài, người đi đầu của chủ nghĩa hình thức trong toán học, nên giáo sư PVHg đã nhấn mạnh cụm từ "Tôi nghĩ" của Hilbert và có phần châm biếm ông.

Vẫn là bài viết [Về Cái Bất khả Quyết định](#), giáo sư đề cập rằng "..., thầy Bửu cho rằng Định lý Cuối cùng của Fermat không thuộc phạm trù bất khả quyết định, vì thầy tin Fermat nói thật ...". Vâng, ở đây giáo sư PVHg không châm biếm quan điểm và niềm tin của giáo sư Tạ Quang Bửu.

Vấn đề là, tại sao ông Hilbert NGHĨ thì giáo sư châm biếm, còn giáo sư Bửu CHO RẰNG, giáo sư Bửu TIN, thì giáo sư PVHg không ý kiến gì, thậm chí có phần còn đồng tình, mặc dù cả Hilbert lẫn giáo sư Bửu đều đưa ra suy nghĩ chủ quan, không có bằng chứng về logic nào cả?

Phải chăng, giáo sư PVHg cũng giống nhiều trường hợp "chỉ nghe những gì mình muốn nghe"? Nhận thức của giáo sư Tạ Quang Bửu ăn khớp với định lí Godel, mà giáo sư PVHg lại rất tôn sùng định lí đó. Đây có phải nguyên nhân khiến giáo sư Hưng đồng tình với giáo sư Bửu?

Lời tổng kết

Một lần nữa, mình cần nhắc lại rằng khoa học phát triển theo nguyên lí **Đứng trên vai người đi trước**. Các nghiên cứu khoa học, hoặc phải dựa trên những nghiên cứu có trước, nếu không thì những ý tưởng đột phá cần phải chờ thời gian kiểm chứng (như các giả thuyết mới, phương pháp mới).

Không quan trọng định lí Godel có tác động mạnh mẽ tới logic, triết học, nhận thức luận, ... tới mức nào. Điều quan trọng là không thể chỉ sử dụng định lí Godel để bác bỏ những lý thuyết trước đó. Điều này chỉ làm trì trệ khoa học vì không những ném hết những công sức của những người đi trước mà còn không có điểm gì tiến bộ.

Giáo sư PVHg đã tổng quát hóa một định lí toán học lên tầm vóc triết học, thể hiện sự suy diễn liên ngành. Sau đó giáo sư lại dùng lí luận triết học đó để "quay chụp" cho những ngành khoa học khác. Giáo sư cũng rất dày công, ché Darwin, Cantor, Hilbert, ... từ năm này sang năm khác, tháng này sang tháng khác, nền tảng này sang nền tảng khác. Các bạn có thể thấy giáo sư có nhiều bài viết từ nhiều năm trước, xuất bản cả sách, và năm 2024 thì lên cả Youtube, chỉ để truyền bá tính "độc hại" trong các tư tưởng kia.

Cuối cùng, giáo sư PVHg thực chất chỉ là suy diễn ý nghĩa triết học từ định lí Godel trong toán học, mà ngày nay chúng ta gọi là nhét chữ. Việc các nhà khoa học thời trước đưa ra lý thuyết mới, họ cũng đâu bắt ai phải tin lý thuyết của mình? Có người tin, có người không tin. Có người tìm cách chứng minh lý thuyết của họ đúng, cũng có người tìm cách bác bỏ lý thuyết của họ, tất nhiên là bằng lập luận chặt chẽ hay bằng chứng xác thực. Dù cách này hay cách khác, cộng đồng khoa học luôn tiến bộ và đi lên. Ở một số trường hợp, lý thuyết của người này giải thích được hiện tượng, nhưng ở một số trường hợp khác, họ thường chối cho lý thuyết khác hiệu quả hơn.

Cuối cùng, việc giáo sư PVHg đang làm chỉ đơn giản là chê bai những lý thuyết đi trước, cỏ xúy chúng ta đừng tin chúng, nhưng giáo sư không hề nói chúng ta cần tin gì. Nhận thức của con người dù có hạn, và chúng ta cần chấp nhận, thì cũng không việc gì phải ngừng học tập, ngừng tư duy, rồi đi chê bai người đi trước cả!!!

Cám ơn các bạn đã xem.

Saint Petersburg, ngày 12 tháng 08 năm 2025

Lê Quốc Dũng

5.1.4 Chủ nghĩa khắc kỷ và phân phối nhị thức

Phân phối nhị thức

Sự tiêu cực trong chủ nghĩa khắc kỷ và phân phối nhị thức có liên hệ mật thiết nhau tới mức không ngờ.

Mình xin phép nhắc lại về phân phối nhị thức (binomial distribution).

Nếu một sự kiện có khả năng xảy ra là p với $0 \leq p \leq 1$ thì trong một dãy n sự kiện như vậy độc lập nhau, xác suất để có k sự kiện xảy ra là

$$P(\xi = k) = C_n^k \cdot p^k \cdot (1-p)^{n-k}.$$

Một ví dụ đơn giản của phân phối nhị thức là bài kiểm tra trắc nghiệm. Giả sử một đề thi có 10 câu, mỗi câu có bốn đáp án A, B, C, D, và chỉ có một đáp án đúng cho mỗi câu.

Khi đó, $n = 10$ và xác suất để chọn ngẫu nhiên đáp án đúng cho mỗi câu là $p = 1/4$. Gọi k là số câu trả lời đúng khi chọn ngẫu nhiên đáp án từng câu.

Mình sẽ thử lập bảng phân bố xác suất với $k = 0, 1, \dots, 10$.

k	0	1	2	3	4	5
$P(\xi = k)$	0.056314	0.187712	0.281568	0.250282	0.145998	0.058399
k	6	7	8	9	10	
$P(\xi = k)$	0.016222	0.003090	0.000386	$29 \cdot 10^{-5}$	10^{-6}	

Đối với trường hợp $k = 10$, tức là lúc chúng ta "lại" đúng hết cả 10 câu, các bạn có thấy xác suất nhỏ tới mức chán chả buồn nói không? Như vậy có thể thấy xác suất một điều tốt xảy ra luôn rất thấp.

5.1.5 Sưu tầm 1

Nguồn: https://vk.com/po_matematike

Tiêu đề gốc. Метод сопряжённых градиентов: линейная алгебра с ускорением.

Tiêu đề. Phương pháp gradient liên hợp: đại số tuyến tính với tốc độ cao.

Khi hệ phương trình tuyến tính $\mathbf{Ax} = \mathbf{b}$ có quy mô cực lớn (hàng triệu biến), việc lưu trữ toàn bộ ma trận \mathbf{A} và sử dụng các phương pháp trực tiếp như phân tích LU trở nên bất khả thi. Đặc biệt khi \mathbf{A} là ma trận thưa, đối xứng và xác định dương (định thức dương?). Trong trường hợp này, phương pháp Gradient liên hợp (Conjugate Gradients - CG) là một trong những thuật toán lặp hiệu quả nhất.

Khác với phương pháp Gradient Descent, CG không chỉ "tiến dần" về cực tiểu của hàm bậc hai:

$$f(\mathbf{x}) = \frac{1}{2} \cdot \mathbf{x}^\top \mathbf{Ax} - \mathbf{b}^\top \mathbf{x}$$

mà di chuyển theo các hướng liên hợp với nhau, giúp tránh lãng phí bước đi. Về lý thuyết, sau đúng n bước, CG tìm được nghiệm chính xác, nhưng trên thực tế, quá trình hội tụ thường đạt được sớm hơn nhiều.

Ý tưởng chính. Thay vì chọn hướng gradient $\mathbf{r}_k = \mathbf{b} - \mathbf{Ax}_k$ ở mỗi bước, CG chọn hướng \mathbf{p}_k liên hợp với các hướng trước đó theo \mathbf{A} :

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k, \quad \beta_k = \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}.$$

Cập nhật nghiệm. Nghiệm mới được tính bằng:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \quad \alpha_k = \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top \mathbf{A} \mathbf{p}_k}.$$

Ưu điểm quan trọng. Không cần lưu trữ toàn bộ ma trận \mathbf{A} , chỉ cần khả năng tính nhanh \mathbf{Ax} .

Ứng dụng rộng rãi trong:

- phương pháp tính;
- học máy;
- đồ họa máy tính;
- giải phương trình đạo hàm riêng.

Phương pháp này minh họa cách lựa chọn hình học thông minh (liên hợp thay vì trực giao) giúp tăng tốc đáng kể tính toán.

5.2 Nước Nga du kí

5.2.1 Giới thiệu

Bước đầu đi học

Năm 2019, mình lên đường sang Nga và bắt đầu hành trình học cử nhân. Việc du học Nga có chỗ tốt, cũng có chỗ không tốt. Đây là câu chuyện của mình về những điều mình đã học được, đã thấy và cảm nhận đối với Liên bang Nga.

Chuyện bắt đầu vào một ngày đẹp trời tháng 4 năm 2019. Khi đó mình vẫn đang là sinh viên năm nhất ở trường Đại học Công nghệ Thông tin (UIT) ở Thành phố Hồ Chí Minh. Bạn cùng lớp giới thiệu cho mình học bổng hiệp định của Bộ Giáo dục và Đào tạo đi du học Nga. Phần sau mình sẽ nói rõ về học bổng hiệp định, hiểu đơn giản là phía Nga sẽ trả học phí và phía Việt Nam sẽ cấp bù sinh hoạt phí (SHP) cho mình. Do đó sinh viên Việt Nam ở Nga thường gọi học bổng hiệp định là *học bổng hai phía*.

Mình thấy ý tưởng này khá hay, được đi du học mà gần như không tốn gì. Lúc này tham khảo ý kiến của mọi người xung quanh là khá cần thiết. Đa số tất nhiên là không đồng ý vì du học Nga không phải cái gì quá hấp dẫn ở thời nay. Sau khi tốt nghiệp cấp 3 rất nhiều người bạn của mình đã đi học ở Mỹ, Úc, Pháp, Phần Lan, Trung Quốc, Canada ... Lúc đó mình đoán là dưới miền nam không chuộng đi Nga và sau này khi sang đây thì mình đã chắc chắn về việc đó.

Tuy nhiên, yếu tố quan trọng nhất chính là ba mẹ mình đã cho phép mình đi sau khi đắn đo suy nghĩ. Nếu ngày đó ba mẹ không cho mình đi, mình đã lỡ mất một cơ hội nhìn ngắm thế giới ngoài Việt Nam. Mình rất biết ơn ba mẹ về điều này. Tuy nhiên mình đã làm ba mẹ buồn vì ... không dẫn được bạn nữ nào về giới thiệu ba mẹ suốt mấy năm cử nhân. Hiện tại mình đã gác lại chuyện đó để dồn sức với hy vọng hoàn thành Tiến sĩ Khoa học (Доктор наук) ở 40 tuổi rồi tính chuyện tình duyên sau ^)^.

Chuẩn bị hồ sơ

Tiếp theo thì mình chuẩn bị hồ sơ để xin học bổng hiệp định. Mọi thứ đều bình thường ngoại trừ việc dịch thuật. Mình theo những bài post trên facebook lúc đó để tìm tới một văn phòng nhận dịch thuật tiếng Nga trên đường Trần Cao Vân (gần đường Nguyễn Đình Chiểu). Lúc đó chỉ có một người dịch tiếng Nga trong khi có khá nhiều người nhận dịch tiếng Anh, Trung, Hàn, ... Đây là điểm thứ hai khiến mình đoán rằng ít người miền nam đi học Nga.

Vì số người dịch tiếng Nga ít nên mình tranh thủ chuẩn bị giấy tờ sớm và đem đi dịch. Mọi chuyện diễn ra khá mượt và mình gửi giấy tờ lên Cục hợp tác quốc tế, là cơ quan trực thuộc Bộ Giáo dục và Đào tạo chịu trách nhiệm quản lý lưu học sinh tại các nước cấp học bổng hiệp định. Sau đó mình chờ đến tháng 8 thì có kết quả mình đậu. Ngày 14/10 mình lên đường sang Nga.

Đa số các bạn đi học Nga lần đầu sẽ giống mình là không biết tiếng Nga. Do đó tụi mình sẽ trải qua một khóa học dự bị tiếng Nga 1 năm, sau đó bắt đầu học cử nhân 4 năm, hoặc thạc sĩ 2 năm, hoặc nghiên cứu sinh 4 năm. Nói đơn giản, nếu bạn chưa từng học ở Nga và không có chứng chỉ tiếng Nga đủ cao thì bạn sẽ cần học 1 năm tiếng.

Năm dự bị tiếng

Năm đó có 7 người sang trường mình học dự bị và 1 anh sang học nghiên cứu sinh (trước đó anh này đã học dự bị và thạc sĩ rồi nên bây giờ không cần học dự bị nữa). Spoil: trong 7 người học dự bị trường mình thì chỉ có mỗi mình ở lại để học tiếp cử nhân, còn 6 bạn kia sang trường Bauman học.

5.2.2 Đi học ở Nga tốt hay xấu?

Cá nhân mình thấy việc sinh viên Việt Nam đi du học Nga có nhiều điểm hại hơn là điểm lợi.

Đầu tiên là học bỗng. Vâng, các bạn không nhầm đâu. Một trong những điểm hại lớn nhất khi đi du học Nga lại chính là học bỗng.

Thông thường sinh viên Việt Nam đi học ở Nga sẽ có hai dạng:

- được Chính phủ Nga trả học phí và được Chính phủ Việt Nam cấp bù sinh hoạt phí (SHP);
- được Chính phủ Nga trả học phí, còn SHP tự lo.

Một số ít thì học theo diện tự túc mọi loại phí nhưng số lượng các bạn ở diện này rất ít. Mình học thạc sĩ ở ITMO là diện này.

Quay lại với học bỗng. Khi các bạn không bị áp lực tài chính thì dễ hiểu các bạn sẽ không quý trọng những đồng tiền đó. Việc bỏ bê học tập sau năm dự bị tiếng Nga xảy ra rất thường xuyên. Nhiều bạn bị đuổi học sau năm 1, năm 2 cử nhân, thậm chí một số trường còn đuổi khi hết năm dự bị.

Một điểm bất lợi thứ hai là cộng đồng sinh viên Việt Nam. Vâng, một lần nữa, các bạn không nhầm đâu. Cộng đồng sinh viên Việt Nam tại Nga, theo mình, là hỗn loạn và có nhiều điểm tiêu cực hơn là tích cực.

Các đơn vị sinh viên tại trường thường tổ chức các hoạt động cho sinh viên nhưng các bạn tham gia không phải trên tinh thần tự nguyện, mà là bắt buộc một phần.

Điều buồn cười là nhiều người bảo các trường đại học ở Nga chỉ có học mà không năng động, không nhiều hoạt động. Nguyên nhân là do nhiều bạn chỉ có đi học, lấy 5 điểm, rồi về, chứ có bao giờ để ý trường, khoa đang tổ chức gì đâu mà biết có hoạt động hay không. Đôi khi vì ngại tiếng Nga của bản thân yêu cũng khiến nhiều bạn không tham gia. Mình đã từng giúp khoa chuẩn bị cho ngày hội giới thiệu về khoa (День открытых дверей, Open Day). Mình cũng tham dự các trường hè do các đại học Nga hợp tác tổ chức. Như vậy, hoạt động ngoại khóa lẩn hoạt động học thuật có rất nhiều, chỉ là các bạn không tham gia rồi bảo ở Nga không năng động.

5.2.3 Góc khẩu nghiệp

✿ Nguy hiểm

Phần này có sự toxic cao, độc giả cân nhắc trước khi xem. ^-^

✿ Nguy hiểm

Phần này là ý kiến cá nhân và cảm xúc tiêu cực tích lũy khi tác giả đi học ở Nga, độc giả cân nhắc trước khi xem. ^;^

Phần 2. Sinh viên Việt Nam tại Moscow

Nguyễn Thị Thùy Linh: ngắn gọn là súc sinh, cô (Linh?) hồn.

5.2.4 Các cuộc thi toán ở Nga

Giới thiệu

Các cuộc thi học thuật ở Nga được chia thành bốn cấp chính:

- cấp trường: cuộc thi tổ chức cho sinh viên trong trường tham gia;
- cấp khu vực (региональные): cuộc thi do một hoặc một số trường trong thành phố (và các tỉnh ngoại ô) tổ chức;
- cấp quốc gia (всероссийские): cuộc thi do một hoặc một số trường trên toàn Nga đứng ra tổ chức;
- cấp quốc tế (международные): các cuộc thi được tổ chức bởi một tổ chức khoa học/cơ quan hàn lâm trên thế giới, hoặc hợp tác giữa các cơ quan.

Trong bài viết này, mình xin phép chia sẻ một số cuộc thi về toán và mật mã mà mình có dịp tham gia hoặc thấy qua. Trong mỗi cuộc thi sẽ trình bày các thông tin chính:

- tên cuộc thi;
- địa chỉ website (nếu có);
- cấp độ (trường, khu vực, quốc gia, quốc tế);
- thời gian diễn ra trong năm (ước lượng);
- địa điểm tổ chức.

Đạt giải ở các cuộc thi này giúp cộng điểm để tính xét повышение государственная академическая стипендия (ПГАС). Khi mình được duyệt ПГАС thì ngoài học phí đã được miễn ra (100%) thì còn được cấp thêm 15% cho mỗi tháng ở kì tiếp theo. Do đó mình hy vọng nhiều sinh viên Việt Nam sẽ hưởng ứng.

Thông thường việc tham gia các cuộc thi sẽ cần thông qua khoa toán của trường. Một số cho phép tham gia online vòng loại (tự do) nhưng khi tham gia vòng trong sẽ được đăng ký thông qua trường. Vì vậy mình nghĩ rằng nếu sinh viên muốn tham gia một giải nào thì nên liên hệ với khoa toán để nhờ họ đăng ký, đôi khi có thể được học trong đội tuyển để chuẩn bị tốt hơn.

Olympiad toán khu vực

Olympiad ở Zelenograd, trường МИЭТ

1. Tên đầy đủ: Московской городской олимпиады по математике студентов технических вузов.
2. Địa chỉ website: <https://www.miet.ru/structure/s/243/e/120419/50> (tham khảo).
3. Cấp độ: khu vực.
4. Thời gian: vào khoảng tháng 4 hàng năm.
5. Địa điểm: trường МИЭТ (thành phố Zelenograd, nằm ở ngoại ô Moscow).
6. Đối tượng tham gia: cử nhân hoặc chuyên gia.

Cuộc thi gồm hai bảng: cho năm nhất (первый курс) và cho các năm sau (старшие курсы).

Đề thi gồm 4 câu áp dụng cho cả hai bảng, tập trung vào giải tích và đại số tuyến tính.

Olympiad toán quốc gia

Olympiad "Я - профессионал"

Olympiad "Я - профессионал" có thể nói là cuộc thi "quốc dân" ở Nga. Thực chất, "Я - профессионал" bao gồm nhiều môn khác nhau, trong đó có môn toán.

1. Tên đầy đủ: "Я - профессионал".

2. Địa chỉ website: <https://yandex.ru/profi/>.
3. Cấp độ: quốc gia.
4. Địa điểm: vòng sơ khảo (отборочный этап) theo hình thức online, vòng chung kết (заключительный этап) theo hình thức trực tiếp tại một trường trong khu vực.
5. Thời gian: vòng sơ khảo diễn ra khoảng tháng 11-12, vòng chung kết diễn ra khoảng tháng 3 năm tiếp theo.
6. Đối tượng tham gia: cử nhân, chuyên gia, thạc sĩ.

Ở vòng sơ khảo, hình thức thi là trắc nghiệm điền đáp án. Đề thi gồm 10 câu. Thí sinh đọc đề, giải ra đáp án, và điền đáp án vào ô trống trên trang web. Nếu trả lời đúng, thí sinh có 10 điểm. Nếu trả lời sai, thí sinh có 0 điểm.

Ở vòng chung kết (заключительный этап) đôi khi sẽ có một vòng bán kết (полуфинал) khi số lượng tham gia quá đông. Ở vòng полуфинал, hình thức thi vẫn là trắc nghiệm điền đáp án giống vòng sơ khảo. Sau đó, vòng chung kết tổng (финал) diễn ra theo hình thức tự luận. Đề thi vòng chung kết tổng cũng có 10 câu. Nếu có ý đúng, thí sinh được điểm.

Các môn thi của "Я - профессионал" nói chung đều gồm hai bảng:

- bảng dành cho cử nhân (для бакалавриата);
- bảng dành cho thạc sĩ/chuyên gia (для магистратуры/специалиста).

Nói cách khác, thạc sĩ và chuyên gia sẽ thi chung đề, chung bảng.

Olympiad "Интегрируй" ở МИФИ

Cuộc thi được tổ chức tại trường МИФИ, nằm ở Moscow.

Cuộc thi diễn ra theo nhiều vòng.

Trong cuộc thi, thí sinh sẽ giải các bài về nguyên hàm và tích phân. Vì vậy cuộc thi mới mang tên "Интегрируй".

[TODO] Sưu tầm đề.

Olympiad toán quốc tế

Open International Internet Olympiad - OIIO

1. Tên đầy đủ: Открытая международная интернет-олимпиада по математике.
2. Địa chỉ website: <https://olymp.i-exam.ru/>.
3. Cấp độ: quốc tế.
4. Địa điểm: cuộc thi gồm 3 vòng. Vòng 1 thi tại trường, vòng 2 thi tại một trường nào đó trong khu vực, vòng 3 - chung kết - thi tại một thành phố tại Nga.
5. Thời gian: vòng 1 diễn ra khoảng tháng 3, vòng 2 diễn ra vào tháng 4, vòng 3 diễn ra vào tháng 5.
6. Đối tượng tham dự: cử nhân từ năm 1 tới năm 3.

Ở vòng 1, thí sinh sẽ giải các bài toán trên máy tính qua hệ thống của cuộc thi. Hình thức thi là trắc nghiệm điền đáp án. Thí sinh đọc đề, giải ra kết quả và ghi đáp án và ô trống trên trang web. Nếu trả lời đúng, thí sinh có điểm. Nếu trả lời sai, thí sinh không có điểm. Vòng 1 diễn ra tại trường, thường sẽ được sắp xếp phòng máy.

Ở vòng 2, hình thức thi giống vòng 1. Thí sinh của một khu vực (ví dụ Moscow) sẽ tập trung về một trường nào đó trong khu vực (ví dụ năm 2023 các thí sinh ở Moscow sẽ tập trung về đại học xâyst dựng Moscow).

Ở vòng 3, hình thức thi là tự luận. Đề thi gồm 10 câu, chấm theo thang điểm. Nếu có ý đúng, thí sinh được điểm. Điểm của mỗi câu sẽ tăng theo hệ số dựa trên số lượng người giải ra. Nếu một câu có nhiều thí sinh giải ra, câu đó sẽ giữ nguyên điểm ban đầu. Nếu một câu có ít thí sinh giải ra, điểm sẽ được tăng theo hệ số do ban tổ chức tính toán dựa trên số lượng lời giải đúng. Vòng 3 diễn ra tại trường ФИЭБ ở thành phố Йошкар-Ола, gần Kazan.

Vì số lượng câu hỏi lớn nên nội dung thi bao quát nhiều lĩnh vực toán: giải tích, đại số tuyến tính, hình học, số học, xác suất thống kê, lý thuyết đồ thị, ...

Ý kiến của mình: điểm hay ở vòng 3 là cơ chế tính điểm, giúp những người dù giải được ít câu nhưng điểm vẫn cao vì câu khó thường là câu ít người giải được.

Chưa phân loại

RUDN Olympiad

Lần đầu tiên mình được tham dự thi toán đồng đội theo hình thức MathBoy (trận chiến toán) năm 2023.

Trong cách thi này, mỗi đội có 3 vị trí: người thuyết trình (докладчик), người phản biện (оппонент) và người giám sát (наблюдатель).

Ở mỗi vòng sẽ có 3 đội thi với nhau. Mỗi đội sẽ có 1 vị trí tương ứng với 3 vị trí trên. Sau đây là ví dụ

	Đội 1	Đội 2	Đội 3
Vòng 1	O	D	H
Vòng 2	H	O	D
Vòng 3	D	H	O

Ở mỗi vòng, đội đóng vai trò người thuyết trình lên bảng ghi bài giải trong thời gian cho phép và thuyết trình về bài giải của đội mình. Đội phản biện có nhiệm vụ phản biện bài thuyết trình đó. Đội giám sát, dựa trên bài thuyết trình cũng như phản biện mà ghi chép lại các lỗi, chỗ khó hiểu, ... và trình lên cho giám khảo.

Ngoài ra, đội thuyết trình trước đó phải trình bày viết tay cho giám khảo chấm trước khi lên thuyết trình.

Ở đây có rất nhiều câu chuyện hack não đã xảy ra. Lúc mình thi vòng 1, câu hỏi quá khó nên đội thuyết trình chỉ viết được một ít. Đồng nghĩa việc đội phản biện cũng như đội giám sát ... thất nghiệp, không có gì để nói.

Đối với vòng 2, trận chiến cân bằng hơn, đội mình làm việc giám sát. Dựa trên bài giải của đội thuyết trình, chúng mình thấy những trường hợp chưa được xét tới và có thể bị sai, do đó cả ba đội đều có điểm (đội thuyết trình có nhiều điểm nhất vì các bạn giải hơn 1 nửa rồi).

Đối với vòng 3, đội mình thuyết trình. Đội mình clear bài đó nên giành điểm tuyệt đối cho phần thuyết trình. Tuy nhiên các bạn phản biện cũng không vừa, vẫn cố gắng bắt một số lỗi do trình bày quá cộ đọng. Kết quả là đội mình (thuyết trình) full điểm cho vòng 3, đội phản biện được 3 điểm.

5.3 Sưu tầm những câu chuyện về các nhà toán học

5.4 Men of Mathematics

Mình dịch quyển *Men of Mathematics* của Bell [40].

5.4.1 Lời nói đầu

Men of Mathematics

Cuộc đời và thành tựu của các nhà toán học vĩ đại từ Zeno đến Poincaré.

Mục lục

1. Giới thiệu.

Sự thoả mái cho độc giả. Bình minh của toán học hiện đại. Các nhà toán học có phải là con người? Những bản sao vô vị. Phạm vi vô hạn trong sự tiến hóa của toán học. Những người tiên phong và trinh sát. Manh mối đi qua mê cung. Liên tục và rời rạc. Sự hiềm có đáng chú ý của lẽ thường. Toán học sống động hay thuyết thần bí mơ hồ? Bốn thời kì vĩ đại của toán học. Thời kì của chúng ta là Thời kì Hoàng kim.

2. Những tư duy hiện đại trong cơ thể cổ đại: Zeno (thế kỷ thứ 5 TCN), Eudoxus (408-355 TCN), Archimedes (287?-212 TCN).

Những người cổ đại hiện đại và những người hiện đại cổ đại. Pythagoras, nhà thần bí vĩ đại, nhà toán học vĩ đại hơn. Chứng minh hay trực giác? Gốc rẽ của giải tích hiện đại. Một gã nhà quê làm đảo lộn các triết gia. Những câu đố chưa được giải quyết của Zeno. Người bạn trẻ cần cù của Plato. Sự kiệt quệ vô tận. Những mẩu truyện tranh hữu ích. Archimedes, quý tộc, nhà khoa học vĩ đại nhất thời cổ đại. Những huyền thoại về cuộc đời và tính cách của ông. Những khám phá và tuyên bố về tính hiện đại của ông. Một người La Mã kiên cường. Sự thất bại của Archimedes và chiến thắng của Rome.

3. Quý ông, người lính và nhà toán học: Descartes (1596-1650).

Những ngày xưa tươi đẹp. Một triết gia trẻ nhưng không phải là một kẻ kiêu ngạo. Những lợi thế vô giá của việc nằm trên giường. Những nghi ngờ tiếp thêm sinh lực. Hòa bình trong chiến tranh. Được cải đạo bởi ác mộng. Sự khải thị của hình học giải tích. Thêm nhiều vụ giết mổ. Rạp xiếc, sự ghen tị chuyên nghiệp, sự khoe khoang, những người bạn nữ dễ tính. Sự khó chịu với lửa địa ngục và sự tôn trọng Giáo hội. Được cứu bởi một cặp hồng y. Một Giáo hoàng tự đánh vào đầu mình. Hai mươi năm sống ẩn dật. Phương pháp. Bị phản bội bởi danh tiếng. Làm việc với Elisabeth. Descartes thực sự nghĩ gì về cô ấy. Christine tự phụ. Cô ấy đã làm gì với Descartes. Sự đơn giản khi sáng tạo trong hình học của ông.

4. Hoàng tử của những người nghiệp dư: Fermat (1601-1665).

Nhà toán học vĩ đại nhất của thế kỷ 17. Cuộc sống bận rộn, thực tế của Fermat. Toán học là sở thích của ông. Đóng góp của ông cho giải tích. Nguyên lý vật lý sâu sắc của ông. Hình học giải tích một lần nữa. Arithmetica và logistica (*tạm dịch - số học và logic học*). Sự vượt trội của Fermat trong số học. Một bài toán chưa được giải về số nguyên tố. Tại sao một số định lý lại "quan trọng"? Một bài kiểm tra trí thông minh. "Sự giảm vô hạn." Thách thức chưa được trả lời của Fermat đối với hậu thế.

5. "Sự vĩ đại và khổ đau của con người": Pascal (1623-1662).

Một thần đồng chôn vùi tài năng của mình. Ở tuổi 17, một nhà hình học vĩ đại. Định lý tuyệt vời của Pascal. Sức khỏe tồi tệ và sự say mê tôn giáo. Chiếc máy tính đầu tiên. Sự xuất sắc của Pascal trong vật lý. Chị gái thánh thiện Jacqueline, người cứu rỗi linh hồn. Rượu và phụ nữ? "Hãy đến tu viện!" Được cải đạo trong một cuộc chè chén. Văn học bị lợi dụng cho sự cuồng tín. Helen của Hình học. Một cơn đau răng thiêng thể. Những gì khám nghiệm tử thi tiết lộ. Một tay cờ bạc làm nên lịch sử toán học. Phạm vi của lý thuyết xác suất. Pascal cùng với Fermat tạo ra lý thuyết này. Sự điên rồ của việc đặt cược chống lại Chúa hoặc Quỷ dữ.

6. Trên bờ biển: Newton (1642-1727).

Đánh giá của Newton về bản thân. Một thiên tài trẻ không được chứng nhận. Sự hỗn loạn ở thời đại ông. Đứng trên vai những người khổng lồ. Mỗi quan hệ duy nhất của ông. Những ngày ở Cambridge. Newton trẻ tuổi làm chủ sự vô ích của việc chịu đựng những kẻ ngốc. Đại dịch lớn là một phước lành lớn hơn. Bất tử ở tuổi 24 (hoặc ít hơn). Giải tích. Newton không ai sánh kịp trong toán học thuần túy, tối cao trong triết học tự nhiên. Ruồi nhặng, ong bắp cày và sự bức bối. Nguyên lý. Samuel Pepys và những kẻ rối rít khác.

Sự thắt vong phảng lặng nhất trong lịch sử. Tranh cãi, thần học, niên đại học, giả kim thuật, chức vụ công, cái chết.

7. Bậc thầy của mọi nghề: Leibniz (1646-1716).

Hai đóng góp tuyệt vời. Con của một chính trị gia. Thiên tài ở tuổi 15. Bị lôi cuốn bởi luật pháp. "Đặc tính phổ quát." Lý luận biểu tượng. Bán rẻ vì tham vọng. Một nhà ngoại giao bậc thầy. Ngoại giao là thứ gì đó, những chiến tích ngoại giao của bậc thầy được để lại cho các nhà sử học. Cáo trở thành nhà sử học, chính khách trở thành nhà toán học. Đạo đức ứng dụng. Sự tồn tại của Chúa. Chủ nghĩa lạc quan. Bốn mươi năm vô ích. Bị vứt bỏ như một miếng giẻ bẩn.

8. Bản chất hay nuôi dưỡng? Gia đình Bernoulli (thế kỷ 17 và 18).

Tám nhà toán học trong ba thế hệ. Bằng chứng làm sàng về di truyền. Giải tích biến phân.

9. Hiện thân của phân tích: Euler (1707-1783).

Nhà toán học sung mãn nhất trong lịch sử. Được cứu khỏi thần học. Các nhà cai trị chi trả. Tính thực tiễn của những điều không thực tiễn. Cơ học thiên thể và chiến tranh hải quân. Một nhà toán học do cơ hội và định mệnh. Bị mắc kẹt ở St. Petersburg. Đức tính của sự im lặng. Mù một nửa vào buổi sáng. Chạy trốn đến nước Phổ tự do. Sự hào phóng và thô lỗ của Frederick Đại đế. Trở về nước Nga hiếu khách. Sự hào phóng và thanh lịch của Catherine Đại đế. Mù hoàn toàn vào buổi trưa. Bậc thầy và người truyền cảm hứng cho các bậc thầy trong một thế kỷ.

10. Một kim tự tháp cao vút: Lagrange (1736-1813).

Nhà toán học vĩ đại và khiêm tốn nhất của thế kỷ 18. Sự phá sản tài chính là cơ hội của ông. Ông đã hình thành kiệt tác của mình ở tuổi 19. Sự hào phóng của Euler. Từ Turin, đến Paris, đến Berlin: một đứa con ngoài giá thú biết ơn đã giúp đỡ một thiên tài. Những cuộc chinh phục trong cơ học thiên thể. Frederick Đại đế hạ mình. Cuộc hôn nhân đăng trí. Công việc như một thói quen. Một tác phẩm kinh điển trong số học. Mécanique analytique một kiệt tác sống động. Một cột mốc trong lý thuyết phương trình. Được chào đón ở Paris bởi Marie Antoinette. Kiệt sức thần kinh, uất và sự chán ghét phổ quát ở tuổi trung niên. Được đánh thức bởi Cách mạng Pháp và một cô gái trẻ. Lagrange nghĩ gì về Cách mạng. Hệ thống mét. Những người cách mạng nghĩ gì về Lagrange. Một triết gia chết như thế nào.

11. Từ nông dân đến kẻ kiêu ngạo: Laplace (1749-1827).

Khiêm tốn như Lincoln, kiêu hãnh như Lucifer. Một sự tiếp đón lạnh lùng và một lời chào nồng nhiệt. Laplace tấn công hệ mặt trời một cách hoành tráng. Mécanique céleste. Đánh giá của ông về bản thân. Những người khác nghĩ gì về ông. "Thể năng" cơ bản trong vật lý. Laplace trong Cách mạng Pháp. Mỗi quan hệ thân thiết với Napoleon. Chủ nghĩa hiện thực chính trị của Laplace vượt trội hơn Napoleon.

12. Những người bạn của một hoàng đế: Monge (1746-1818), Fourier (1768-1830).

Con trai của một người mài dao và con trai của một thợ may giúp Napoleon lật đổ chiếc xe ngựa của giới quý tộc. Vở opera hài ở Ai Cập. Hình học mô tả của Monge và Thời đại Máy móc. Phân tích của Fourier và vật lý hiện đại. Sự ngu ngốc của việc tin tưởng vào các hoàng tử hoặc những người vô sản. Chán đến chết và chán đến chết.

13. Ngày vinh quang: Poncelet (1788-1867).

Được hồi sinh từ đống đổ nát của Napoleon. Con đường vinh quang dẫn đến nhà tù. Mùa đông ở Nga năm 1812. Thiên tài làm gì trong tù. Hai năm hình học trong địa ngục. Phần thưởng của thiên tài: sự ngu ngốc của thói quen. Hình học xạ ảnh của Poncelet. Nguyên lý liên tục và đối ngẫu.

14. Hoàng tử của các nhà toán học: Gauss (1777-1855).

Gauss, người ngang hàng với Archimedes và Newton trong toán học. Xuất thân khiêm tốn. Sự tàn bạo của người cha. Sự phát triển trí tuệ sớm không ai sánh kịp. Cơ hội của ông ở tuổi 10. Đến năm 12 tuổi, ông mơ về những khám phá cách mạng, đến năm 18 tuổi, ông đạt được chung. Disquisitiones Arithmeticae. Các tác phẩm mang tính bước ngoặt khác được tóm tắt. Thảm họa Ceres. Napoleon, giàn tiếp cướp đoạt Gauss,

nhận giải nhì. Những tiến bộ cơ bản trong tất cả các nhánh của toán học nhờ Gauss quá nhiều để liệt kê: xem tài khoản đã cho (?). Hiền triết của các hiền triết. Cái chết không mong muốn.

15. Toán học và cối xay gió: Cauchy (1789-1857).

Sự thay đổi trong bản chất của toán học với thế kỷ 19. Thời thơ ấu trong Cách mạng Pháp. Tính toán sai lầm sớm của Cauchy. Lời tiên tri của Lagrange. Kỹ sư Cơ đốc trẻ. Sự sặc sảo tiên tri của Malus. Lý thuyết nhóm. Đứng đầu ở tuổi 27. Một trong những câu đố của Fermat được giải. Hà mã sùng đạo. Bị húc bởi Charles the Goat (?). Các bài báo về thiên văn học và vật lý toán. Sự ngọt ngào và sự cứng đầu bất khả chiến bại. Chính phủ Pháp tự làm mình trở thành kẻ ngốc. Vị trí của Cauchy trong toán học. Nhược điểm của một nhân cách không thể chê trách.

16. Copernicus của hình học: Lobatchevsky (1793-1856).

Danh của góa phụ (?). Kazan. Được bổ nhiệm làm giáo sư và giàn điệp. Khả năng phổ quát. Lobatchevsky với tư cách là một nhà quản lý. Lý trí và hương trầm (?) chống lại dịch tả. Lòng biết ơn của người Nga. Bị sỉ nhục trong thời kỳ đỉnh cao. Mù như Milton, Lobatchevsky đọc kiệt tác của mình. Sự tiến bộ của ông vượt qua Euclid. Hình học phi Euclid. Một Copernicus của trí tuệ.

17. Thiên tài và nghèo khó: Abel (1802-1829).

Na Uy năm 1802. Bị bóp nghẹt bởi sự sinh sản của giáo sĩ. Sự thức tỉnh của Abel. Sự hào phóng của một giáo viên. Một học trò của các bậc thầy. Sai lầm may mắn của ông. Abel và phương trình bậc năm. Chính phủ giải cứu. Chuyển du lịch vĩ đại của Abel qua toán học châu Âu không quá vĩ đại. Sự lịch sự của người Pháp và sự thân thiện của người Đức. Crelle và tạp chí của ông. Tội lỗi không thể tha thứ của Cauchy. "Định lý Abel." Một thứ để giữ cho các nhà toán học bận rộn 500 năm. Đặt vương miện lên một xác chết.

18. Nhà đại số vĩ đại: Jacobi (1804-1851).

Mẹ điện (?) so với toán học. Sinh ra giàu có. Khả năng ngôn ngữ học của Jacobi. Dành cả đời cho toán học. Công trình đầu tiên. Sách sẽ. Một con ngỗng giữa những con cáo. Thời kỳ khó khăn. Hàm elliptic. Vị trí của chúng trong sự phát triển chung. Đảo ngược (?) (*tam dịch - Nghịch đảo*). Công trình trong số học, động lực học, đại số và hàm Abelian. Sự tuyên bố của Fourier. Câu trả lời của Jacobi.

19. Bi kịch Ireland: Hamilton (1805-1865).

Người vĩ đại nhất của Ireland. Giáo dục sai lầm phức tạp (?). Những khám phá ở tuổi 17. Một sự nghiệp đại học độc đáo. Thất vọng trong tình yêu. Hamilton và các nhà thơ. Được bổ nhiệm tại Dunstink. Hệ thống tia. Nguyên lý quang học. Dự đoán về khúc xạ hình nón. Hôn nhân và rượu. Trưởng. Số phức. Luật giao hoán bị bãi bỏ. Quaternion. Những ngọn núi giấy.

20. Thiên tài và sự ngu ngốc: Galois (1811-1832).

Kỷ lục thế giới của mọi thời đại về sự ngu ngốc. Thời thơ ấu của Galois. Các nhà giáo dục vượt trội hơn chính họ. Ở tuổi 16, Galois lặp lại sai lầm của Abel. Chính trị và giáo dục. Các kỳ thi như những trọng tài của thiên tài. Bị săn đuổi đến chết bởi một linh mục.Thêm sự bất tài của học thuật. Cauchy đăng ký một lần nữa. Bị đẩy đến nổi loạn. Một nhà toán học bậc thầy ở tuổi 19. "Một xác chết để kích động dân chúng." Công rãnh bẩn nhất ở Paris. Những người yêu nước lao vào chiến trường danh dự. Đêm cuối cùng của Galois. Câu đố của các phương trình được giải. Bị chôn như một con chó.

21. Cặp song sinh bất biến: Sylvester (1814-1897); Cayley (1821-1895).

Những đóng góp của Cayley. Thời trẻ. Cambridge. Giải trí. Được gọi đến Luật sư đoàn. Mười bốn năm trong luật pháp. Cayley gặp cộng tác viên của mình. Cuộc đời sóng gió hơn của Sylvester. Bị cản trở bởi tôn giáo. Cayley và Sylvester đối lập. Sứ mệnh của Sylvester đến Virginia. Những bước sai lầm tiếp theo. Lý thuyết bất biến. Được gọi đến Đại học Johns Hopkins. Sức sống không thể dập tắt. "Rosalind." Sự thống nhất hình học của Cayley. Không gian n chiều. Ma trận. Oxford ủng hộ Sylvester. Cuối cùng cũng đáng kính.

22. Thầy và trò: Weierstrass (1815-1897); Sonja Kowalewski (1850-1891).

Cha đẻ của giải tích hiện đại. Mỗi quan hệ của Weierstrass với những người đương thời. Những hình phạt của sự xuất sắc. Bị ép vào luật, tự ép mình thoát ra. Bia và kiếng. Một khởi đầu mới. Nợ Gudermann. Mười lăm năm trong bùn. Sự giải thoát kỳ diệu. Vấn đề cuộc đời của Weierstrass. Quá nhiều thành công. Sonja tấn công bậc thầy. Học trò yêu thích của ông. Tình bạn của họ. Lòng biết ơn của một người phụ nữ. Lặp lại, Sonja giành giải thưởng Paris. Weierstrass được vinh danh toàn cầu. Chuỗi lũy thừa. Số học hóa giải tích (?). Nghi ngờ.

23. Độc lập hoàn toàn: Boole (1815-1864).

Toán học Anh. Bị nguyền rủa từ khi sinh ra bởi sự kiêu ngạo. Cuộc đấu tranh của Boole để được giáo dục (?). Chẩn đoán sai. Sự can thiệp của Thiên Chúa. Khám phá các bất biến. Đại số là gì? Một triết gia tấn công một nhà toán học. Cuộc tàn sát khủng khiếp. Cơ hội của Boole. "Các quy luật của tư duy." Logic biểu tượng. Ý nghĩa toán học của nó. Đại số Boolean. Chết trong thời kỳ đỉnh cao.

24. Con người, không phải phương pháp: Hermite (1822-1901).

Những vấn đề cũ và phương pháp mới. Người mẹ tài ba của Hermite. Sự ghét bỏ của ông đối với các kỳ thi. Tự dạy mình. Toán học cao cấp đôi khi dễ hơn toán học cơ bản. Thảm họa giáo dục. Thư gửi Jacobi. Bậc thầy ở tuổi 21. Trả thù những người chấm thi. Hàm Abelian. Bị làm phiền bởi Cauchy. Chủ nghĩa thần bí của Hermite. Giải pháp cho phương trình bậc năm tổng quát. Số siêu việt. Một gợi ý cho những người vẽ vòng tròn. Chủ nghĩa quốc tế của Hermite.

25. Kẻ hoài nghi: Kronecker (1823-1891).

Huyền thoại về một vị thánh người Mỹ. Kronecker may mắn. Chiến thắng ở trường. Tài năng lớn. Số đại số. Những trận chiến với Weierstrass. Sự nghiệp kinh doanh của Kronecker. Trở lại với toán học giàu có. Lý thuyết Galois. Các bài giảng của Kronecker. Sự hoài nghi của ông là đóng góp độc đáo nhất.

26. Linh hồn trong sáng: Riemann (1826-1866).

Nghèo nhưng hạnh phúc. Sự nhút nhát kinh niên của Riemann. Được định sẵn cho nhà thờ. Được cứu. Một giả thuyết nổi tiếng. Sự nghiệp tại Göttingen. "Một toán học mới." Nghiên cứu vật lý. Ứng dụng của tô pô vào giải tích. Bài luận mang tính bước ngoặt về nền tảng của hình học. Gauss nhiệt tình. Phước lành của sự nghèo khó. Góc rẽ của giải tích tensor. Tìm kiếm sức khỏe. Dưới một cây vả (?). Cột mốc của Riemann trong hình học. Độ cong của không gian. Mở đường cho thuyết tương đối.

27. Số học thứ hai: Kummer (1810-1893), Dedekind (1831-1916).

Già trong gỗ (?). Sự uốn cong của Napoleon (?) đối với tính cách vui vẻ của Kummer. Có tài năng như nhau trong trừu tượng và cụ thể. Định lý cuối cùng của Fermat bắt đầu từ đâu. Lý thuyết số hoàn hảo. Phát minh của Kummer có thể so sánh với Lobachevsky. Bề mặt sóng trong bốn chiều. Lớn về thể xác, tâm trí và trái tim. Dedekind, học trò cuối cùng của Gauss. Người trình bày đầu tiên về Galois. Sớm quan tâm đến khoa học. Chuyển sang toán học. Công trình của Dedekind về tính liên tục. Sự ra đời lý thuyết ideal của ông.

28. Nhà bác học toàn diện cuối cùng: Poincaré (1854-1912).

Sự toàn diện và phương pháp của Poincaré. Những trớ ngại thời thơ ấu. Bị toán học cuốn hút. Giữ được sự tinh tú敏锐 trong chiến tranh Pháp-Phổ. Bắt đầu là kỹ sư khai thác mỏ. Tác phẩm vĩ đại đầu tiên. Hàm tự đẳng cấu. "Chìa khóa của vũ trụ đại số." Vấn đề n vật thể. Phần Lan có văn minh không? Phương pháp mới của Poincaré trong cơ học thiên thể. Vũ trụ học. Các khám phá toán học được tạo ra như thế nào. Tường thuật của Poincaré. Những linh cảm và cái chết sớm.

29. Thiên đường đã mất? Cantor (1845-1918).

Những kẻ thù cũ với khuôn mặt mới. Những tín điều thối rữa. Di sản nghệ thuật và sự ám ảnh về cha của Cantor. Trốn thoát, nhưng quá muộn. Công trình cách mạng của ông không đi đến đâu. Sự nhỏ nhẹ của học thuật. Hậu quả thảm khốc của "an toàn trước tiên." Một kết quả mang tính bước ngoặc. Nghịch lý hay sự thật? Sự tồn tại vô hạn của các số siêu việt. Sự hung hăng tiến lên, sự nhút nhát lùi lại. Những tuyên bố ngoạn mục hơn. Hai loại nhà toán học.

5.4.2 Acknowledgments

Nếu không có một lượng lớn các chú thích thì sẽ không thể trích dẫn nguồn cho mọi tuyên bố về sự kiện lịch sử ở các trang sau. Nhưng phần lớn tài liệu tham khảo không có sẵn bên ngoài các thư viện đại học lớn, và hầu hết đều bằng tiếng nước ngoài. Đối với các ngày tháng chính và các sự kiện chính trong cuộc đời của một người cụ thể, tôi đã tham khảo các thông cáo, cáo phó (của những người hiện đại); những thông cáo này được tìm thấy trong các hồ sơ của các hiệp hội học thuật mà người đó là thành viên. Các chi tiết thú vị khác được đưa ra trong thư từ giữa các nhà toán học và trong tập hợp các tác phẩm của họ. Ngoài một số nguồn cụ thể được trích dẫn hiện tại, các thư mục và tài liệu tham khảo sau đây đặc biệt hữu ích.

1. Các ghi chú và nhiều bài báo lịch sử được tóm tắt trong *Jahrbuch über die Fortschritte der Mathematik* (phần về lịch sử toán học).
2. Tương tự trong *Bibliotheca Mathematica*.

Chỉ có ba nguồn đủ "riêng tư" để cần trích dẫn rõ ràng. Cuộc đời của Galois dựa trên tài liệu cổ điển của P. Dupuy trong *Annales scientifiques de l' École normale supérieure* (tập 13, 1896), và các ghi chú được biên tập bởi Jules Tannery. Thư từ giữa Weierstrass và Sonja Kowalewski được Mittag-Leffler xuất bản trong *Acta Mathematica* (cũng như một phần trong *Comptes rendus du 2me Congrès international des Mathématiciens*, Paris, 1902). Nhiều chi tiết về Gauss được lấy từ cuốn sách của W. Sartorius von Waltershausen, *Gauss zum Gedächtniss*, Leipzig, 1856.

Sẽ là liều lĩnh khi tuyên bố rằng mọi ngày tháng hoặc cách viết tên riêng trong cuốn sách này đều chính xác. Các ngày tháng được sử dụng chủ yếu với mục đích định hướng độc giả về tuổi của một người khi họ thực hiện những phát minh độc đáo nhất của mình.

Về cách viết tên, tôi nhận sự bất lực của mình trước những biến thể như Basle, Bale, Basel cho một thị trấn Thụy Sĩ, hoặc Utzendorff, Utzisdorf cho một thị trấn khác, mỗi cách viết được ưa thích bởi một số cơ quan có uy tín. Khi phải chọn giữa James và Johann, hoặc giữa Wolfgang và Farkas, tôi chọn cách dễ dàng hơn và xác định người đó bằng cách khác.

Hầu hết các chân dung được sao chép từ bộ sưu tập David Eugene Smith, Đại học Columbia. Chân dung của Newton là từ một bản khắc gốc được cho mượn bởi Giáo sư E. C. Watson. Các bản vẽ đã được xây dựng chính xác bởi ông Eugene Edwards.

Như trong một dịp trước đây (*The Search for Truth*), tôi rất vui được cảm ơn Tiến sĩ Edwin Hubble và vợ ông, Grace, vì sự hỗ trợ vô giá của họ.

Mặc dù tôi là người duy nhất chịu trách nhiệm cho mọi tuyên bố trong cuốn sách, nhưng việc có được những lời phê bình học thuật (ngay cả khi tôi không luôn luôn hưởng lợi từ chúng) từ hai chuyên gia trong các lĩnh vực mà tôi không thể tự nhận là chuyên gia, đã giúp ích rất nhiều, và tôi tin rằng những lời phê bình mang tính xây dựng của họ đã làm giảm bớt những thiếu sót của tôi. Tiến sĩ Morgan Ward cũng đã phê bình một số chương và đưa ra nhiều gợi ý hữu ích về các vấn đề mà ông là chuyên gia. Toby, như trước đây, đã đóng góp rất nhiều; để ghi nhận những gì cô ấy đã cho tôi, tôi dành tặng cuốn sách này cho cô ấy (nếu cô ấy chấp nhận) - nó là của cô ấy cũng như của tôi.

Cuối cùng, tôi muốn cảm ơn các nhân viên của các thư viện khác nhau đã hào phóng giúp đỡ với việc cho mượn các cuốn sách quý hiếm và tài liệu tham khảo. Đặc biệt, tôi muốn cảm ơn các thủ thư tại Đại học Stanford, Đại học California, Đại học Chicago, Đại học Harvard, Đại học Brown, Đại học Princeton, Đại học Yale, Thư viện John Crerar (Chicago), và Viện Công nghệ California.

--- E. T. BELL

Họ nói, họ nói gì, hãy để họ nói.

---(Khẩu hiệu của Đại học Marischal, Aberdeen)

Khoa học Toán học Thuần túy, trong sự phát triển hiện đại của nó, có thể được coi là sáng tạo độc đáo nhất của tinh thần con người.

---A. N. WHITEHEAD (*Science and the Modern World*, 1925)

Một sự thật toán học không đơn giản cũng không phức tạp trong chính nó, nó là vậy.

---ÉMILE LEMOINE

Một nhà toán học mà không phải là một nhà thơ sẽ không bao giờ là một nhà toán học hoàn chỉnh.

---KARL WEIERSTRASS

Tôi đã nghe chính mình bị buộc tội là một kẻ phản đối, một kẻ thù của toán học, điều mà không ai có thể đánh giá cao hơn tôi, vì nó đạt được chính điều mà tôi đã bị từ chối.

---GOETHE

Các nhà toán học giống như những người yêu nhau. . . . Hãy cho một nhà toán học nguyên lý nhỏ nhất, và anh ta sẽ rút ra một hệ quả mà bạn cũng phải chấp nhận, và từ hệ quả này, ra một hệ quả khác.

---FONTENELLE

Dễ dàng hơn để vẽ một vòng tròn vuông hơn là vượt qua một nhà toán học.

---AUGUSTUS DE MORGAN

Tôi hối tiếc rằng trong bài giảng này, tôi đã phải đưa ra một liều lượng lớn hình học bốn chiều. Tôi không xin lỗi, vì tôi thực sự không chịu trách nhiệm cho việc tự nhiên ở khía cạnh cơ bản nhất của nó là bốn chiều. Mọi thứ là như vậy. . . .

---A. N. WHITEHEAD (The Concept of Nature, 1920)

Số cai trị vũ trụ.

---THE PYTHAGOREANS

Toán học là Nữ hoàng của Khoa học, và Số học là Nữ hoàng của Toán học.

---C. F. GAUSS

Như vậy, số có thể được coi là cai trị toàn bộ thế giới của lượng, và bốn quy tắc của số học có thể được coi là trang bị đầy đủ của nhà toán học.

---JAMES CLERK MAXWELL

Các nhánh khác nhau của Số học -- Tham vọng, Phân tâm, Xấu xí, và Chế giễu.

---THE MOCK TURTLE (Alice in Wonderland)

Chúa tạo ra các số nguyên, phần còn lại là công việc của con người.

---LEOPOLD KRONECKER

[Số học] là một trong những nhánh lâu đời nhất, có lẽ là nhánh lâu đời nhất, của kiến thức nhân loại; và một số bí mật sâu sắc nhất của nó nằm gần những sự thật tầm thường nhất của nó.

---H. J. S. SMITH

Các tác phẩm của Plato không thuyết phục bất kỳ nhà toán học nào rằng tác giả của chúng là người nghiên cứu hình học. . . . Chúng ta biết rằng ông khuyến khích toán học. . . . Nhưng nếu -- điều mà không ai tin -- câu nói upbels ἀγεωμέτρητος εἰσίτω [Hãy để không ai không biết hình học bước vào] của Tzetzes được viết trên cổng của ông, nó sẽ không chỉ ra hình học bên trong hơn là một lời cảnh báo không quên mang theo một gói bánh sandwich sẽ hứa hẹn một bữa tối ngon.

---AUGUSTUS DE MORGAN

Không có con đường hoàng gia dẫn đến hình học.

---MENAECHMUS (nói với ALEXANDER ĐẠI ĐẾ)

Ông đã nghiên cứu và gần như làm chủ sáu cuốn sách của Euclid từ khi ông là thành viên của Quốc hội.

Ông bắt đầu một khóa học kỹ luật tinh thần nghiêm ngặt với ý định cải thiện khả năng của mình, đặc biệt là khả năng logic và ngôn ngữ. Do đó, sự yêu thích của ông đối với Euclid, mà ông mang theo trên đường đi cho đến khi ông có thể dễ dàng chứng minh tất cả các mệnh đề trong sáu cuốn sách; thường học đến tận đêm, với một ngọn nến gần gối, trong khi các luật sư đồng nghiệp của ông, nửa tá trong một phòng, làm đầy không khí với tiếng ngáy không dứt.

---ABRAHAM LINCOLN (Tự truyện ngắn, 1860)

Nghe có vẻ kỳ lạ, sức mạnh của toán học nằm ở việc tránh mọi suy nghĩ không cần thiết và ở sự tiết kiệm tuyệt vời của các hoạt động tinh thần.

---ERNST MACH

Một đường cong duy nhất, được vẽ theo cách của đường cong giá bông, mô tả tất cả những gì tai có thể nghe được như kết quả của một buổi biểu diễn âm nhạc phức tạp nhất. . . . Đối với tôi, đó là một bằng chứng tuyệt vời về sức mạnh của toán học.

---LORD KELVIN

Nhà toán học, được cuốn theo dòng chảy của các ký hiệu, xử lý những sự thật thuần túy hình thức, vẫn có thể đạt được những kết quả vô cùng quan trọng cho việc mô tả vũ trụ vật lý của chúng ta.

---KARL PEARSON

Các ví dụ . . . có thể được nhân lên ad libitum, cho thấy việc giải thích kết quả thí nghiệm thường khó khăn như thế nào nếu không có sự trợ giúp của toán học.

---LORD RAYLEIGH

Nhưng có một lý do khác cho danh tiếng cao của toán học: nó cung cấp cho các khoa học tự nhiên tính chính xác ở một mức độ an toàn nhất định mà, không có toán học, họ không thể đạt được.

---ALBERT EINSTEIN

Toán học là công cụ đặc biệt phù hợp để xử lý các khái niệm trừu tượng thuộc bất kỳ loại nào và không có giới hạn nào đối với sức mạnh của nó trong lĩnh vực này. Vì lý do này, một cuốn sách về vật lý mới, nếu không chỉ thuần túy mô tả công việc thực nghiệm, phải có cơ sở là toán học.

---P. A. M. DIRAC (Quantum Mechanics, 1930)

Khi tôi tiến hành nghiên cứu Faraday, tôi nhận ra rằng phương pháp của ông trong việc hình dung các hiện tượng [của điện từ học] cũng là một phương pháp toán học, mặc dù không được thể hiện dưới dạng ký hiệu toán học thông thường. Tôi cũng nhận thấy rằng các phương pháp này có thể được biểu diễn dưới dạng toán học thông thường, và do đó so sánh với các phương pháp của các nhà toán học chuyên nghiệp.

---JAMES CLERK MAXWELL (A Treatise on Electricity and Magnetism, 1873) *

Truy vấn 64 . . . Liệu các nhà toán học . . . có những bí ẩn của họ, và, hơn nữa, sự chống đối và mâu thuẫn của họ?

---BISHOP BERKELEY

Để tạo ra một triết học lành mạnh, bạn nên từ bỏ siêu hình học nhưng hãy là một nhà toán học giỏi.

---BERTRAND RUSSELL (trong một bài giảng, 1935)

Toán học là siêu hình học tốt duy nhất.

---LORD KELVIN

Làm thế nào mà toán học, vốn là sản phẩm của tư duy con người độc lập với kinh nghiệm, lại có thể thích nghi một cách tuyệt vời với các đối tượng của thực tế?

---ALBERT EINSTEIN (1920).

Mọi khám phá mới đều có dạng toán học, vì không có hướng dẫn nào khác mà chúng ta có thể có.

---C. G. DARWIN (1931).

Khái niệm vô cực! Không có câu hỏi nào khác từng làm rung động tinh thần con người sâu sắc đến vậy.

---DAVID HILBERT (1921).

Khái niệm vô cực là người bạn lớn nhất của chúng ta; nó cũng là kẻ thù lớn nhất của sự bình yên trong tâm trí chúng ta. . . . Weierstrass đã dạy chúng ta tin rằng cuối cùng chúng ta đã thuần hóa và kiểm soát được yếu tố không tuân thủ này. Tuy nhiên, điều đó không phải là trường hợp; nó đã lại thoát ra. Hilbert và Brouwer đã bắt đầu thuần hóa nó một lần nữa. Trong bao lâu? Chúng ta tự hỏi.

---JAMES PIERPONT (Bulletin of the American Mathematical Society, 1928).

Theo ý kiến của tôi, một nhà toán học, trong chừng mực mà ông là một nhà toán học, không cần phải bận tâm với triết học -- một ý kiến, hơn nữa, đã được nhiều triết gia bày tỏ.

---HENRI LEBESGUE (1936).

Chúa luôn hình học hóa.

---PLATO.

Chúa luôn số học hóa. -- C. G. J. JACOBI.

Kiến trúc sư vĩ đại của Vũ trụ giờ đây bắt đầu xuất hiện như một nhà toán học thuần túy.

---J. H. JEANS (The Mysterious Universe, 1930).

Toán học là khoa học chính xác nhất, và các kết luận của nó có khả năng được chứng minh tuyệt đối. Nhưng điều này chỉ đúng vì toán học không cố gắng rút ra các kết luận tuyệt đối. Tất cả các sự thật toán học đều tương đối, có điều kiện.

---Charles Proteus Steinmetz (1923).

Đó là một quy tắc an toàn để áp dụng rằng, khi một tác giả toán học hoặc triết học viết với một sự sâu sắc mơ hồ, ông ta đang nói nhảm.

---A. N. Whitehead (1911).

6

Tài liệu tham khảo

Bibliography

- [1] Đoàn Quỳnh, Trần Nam Dũng, Nguyễn Vũ Lương, and Đặng Hùng Thắng. *Tài liệu chuyên toán. Đại số và Giải tích 11*. Nhà xuất bản giáo dục Việt Nam, 2010.
- [2] Euclid. *Euclid's Elements of Geometry*. Richard Fitzpatrick, revised and corrected edition, 2007. ISBN 978-0-6151-7984-1.
- [3] Lê Quang Ánh. *Thiên tài và số phận. Chuyện kể về các nhà toán học*. Volume 35 of Tủ sách Sputnik. Nhà xuất bản Thế Giới, 2018.
- [4] Nguyễn Tiến Dũng, Trần Thanh Nam, Nguyễn Chí Thức, and Hồ Thị Thảo Trang. *Toán học qua các câu chuyện về tập hợp*. Nhà xuất bản thế giới, 2023. ISBN 978-604-392-585-2.
- [5] Н.Я. Виленкин. *Рассказы о множествах*. МЦНМО, 2019. ISBN 978-5-4439-0986-8.
- [6] Ю.В. Таранников. *Дискретная математика. Задачник*. Юрайт, 2016. ISBN 9785991662833.
- [7] T. W. Judson. *Abstract Algebra. Theory and Applications*. Stephen F. Austin State University, 08 2012. URL: abstract.pugetsound.edu.
- [8] Matthew Macauley. Math 4120 (Modern Algebra). URL: <https://www.youtube.com/@ProfessorMacauley>, 2022. Department of Mathematical Science Clemson University.
- [9] Euclid John Casey. The first six books of the elements of euclid. 2007.
- [10] Vũ Hữu Tiệp. Blog machine learning cơ bản. URL: <https://machinelearningcoban.com>.
- [11] О.Н. Василенко. *Теоретико-числовые алгоритмы в криптографии*. МЦНМО, 2003.
- [12] Hồ Sĩ Đàm, Đỗ Đức Đông, Lê Minh Hoàng, and Nguyễn Thanh Hùng. *Tài liệu giáo khoa chuyên tin. Quyển 1*. Nhà xuất bản giáo dục Việt Nam, 2009.
- [13] Г.П. Агибалов. *Избранные теоремы начального курса криптографии*. Томский государственный университет, 2005.
- [14] D.G. Fon-Der-Flaass. *A bound on correlation immunity*. Elektron. Mat. Izv., 2007.
- [15] T.W. Cusick and P Stănică. *Cryptographic Boolean Functions and Applications*. Acad. Press. Elsevier, 2009.
- [16] Kaisa Nyberg. Differentially uniform mappings for cryptography. In *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of Lecture Notes in Computer Science, 55–64. Springer, 1993. doi:[10.1007/3-540-48285-7_6](https://doi.org/10.1007/3-540-48285-7_6).

- [17] Н.Н. Токарева. *Симметрическая криптография. Краткий курс.* Novosibirsk State University, 2012. ISBN 978-5-4437-0067-0.
- [18] Н.Н. Токарева. *Нелинейные булевые функции.* LAP LAMBERT Academic Publishing, 2011. ISBN 978-3-8433-0904-2.
- [19] И.А. Панкратова. *Булевые функции в криптографии. Учебное пособие.* Издательский Дом Томского государственного университета, 2014.
- [20] М.И. Рожков А.Б. Лось, А.Ю. Нестеренко. *Криптографические методы защиты информации.* Юрайт, 2 edition, 2018. ISBN 978-5-534-01530-0.
- [21] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vinkelsoe. Present: an ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of Lecture Notes in Computer Science, 450–466. Springer, 2007. URL: <https://iacr.org/archive/ches2007/47270450/47270450.pdf>, doi:10.1007/978-3-540-74735-2_31.
- [22] М.А. Черепнев С.Б. Гашков, Э.А. Применко. *Криптографические методы защиты информации.* Академия, 2010. ISBN 978-5-7695-4962-5.
- [23] Whitfield Diffie and George Ledin (translators). SMS4 encryption algorithm for wireless networks. Cryptology ePrint Archive, Paper 2008/329, 2008. URL: <https://eprint.iacr.org/2008/329>.
- [24] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, February 1997. URL: <https://www.rfc-editor.org/info/rfc2104>, doi:10.17487/RFC2104.
- [25] Peter Gutmann. Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). RFC 7366, September 2014. URL: <https://www.rfc-editor.org/info/rfc7366>, doi:10.17487/RFC7366.
- [26] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. 2007-11-28 2007. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=51288.
- [27] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality [including updates through 7/20/2007]. 2004-05-12 2004. doi:<https://doi.org/10.6028/NIST.SP.800-38C>.
- [28] Центр защиты информации и специальной связи ФСБ России. Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров, реализующие аутентифицированное шифрование. 2019-09-06 2019. URL: <https://meganorm.ru/Data2/1/4293727/4293727270.pdf>.
- [29] The Amazing King. The Amazing King. URL: <http://theamazingking.com/>.
- [30] Jérémie Jean. TikZ for Cryptographers. <https://www.iacr.org/authors/tikz/>, 2016.
- [31] Gregory V. Bard. *Algebraic Cryptanalysis.* Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 0387887563.
- [32] Thomas Jakobsen and Lars R. Knudsen. The interpolation attack on block ciphers. In *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, volume 1267 of Lecture Notes in Computer Science, 28–40. Springer, 1997. doi:10.1007/BFb0052332.
- [33] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. *An Introduction to Mathematical Cryptography.* Springer, 2 edition, 2014. ISBN 978-1-4939-1710-5. doi:10.1007/978-1-4939-1711-2.
- [34] Steven D Galbraith. *Mathematics of Public Key Cryptography.* Cambridge University, 2018.
- [35] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978. doi:10.1109/TIT.1978.1055873.

- [36] Robert J. McEliece. A public key cryptosystem based on algebraic coding theory. *DSN Progress Report* 42-44, 1978.
- [37] И.М. Петрушко. *Курс высшей математики. Теория вероятностей. Лекции и практикум : учебное пособие*. Издательство "Лань", 3 edition, 2008. ISBN 978-5-8114-0728-6. URL: <https://e.lanbook.com/book/497>.
- [38] Gregor Leander, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. A cryptanalysis of printcipher: the invariant subspace attack. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, 206–221. Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [39] Hồ Sĩ Đàm, Đỗ Đức Đông, Lê Minh Hoàng, and Nguyễn Thanh Hùng. *Tài liệu giáo khoa chuyên tin. Quyển 3*. Nhà xuất bản giáo dục Việt Nam, 2009.
- [40] E.T. Bell. *Men of Mathematics*. A Fireside book. Touchstone, 1986. ISBN 9780671628185. URL: <https://books.google.ca/books?id=BLFL3coT5i4C>.

Proof Index

DFT-cor-1

DFT-cor-1 (*discrete-mathematics/discrete-fourier-transform*), 56
218

DFT-exp-ring

DFT-exp-ring (*discrete-mathematics/discrete-fourier-transform*), 219
212

DFT-fhat-fcheck

DFT-fhat-fcheck (*discrete-mathematics/discrete-fourier-transform*), 291
215

DFT-lem-z2n

DFT-lem-z2n (*discrete-mathematics/discrete-fourier-transform*), 291
212

DFT-mult-poly

DFT-mult-poly (*discrete-mathematics/discrete-fourier-transform*), 217
??

DFT-no-mult-poly

DFT-no-mult-poly (*discrete-mathematics/discrete-fourier-transform*), 304
218

DFT-rem-Tx

DFT-rem-Tx (*discrete-mathematics/discrete-fourier-transform*), 186
219

DFT-thr-Tx

DFT-thr-Tx (*discrete-mathematics/discrete-fourier-transform*), 173
219

DTF-F-hat-check

DTF-F-hat-check (*discrete-mathematics/discrete-fourier-transform*),
214

GL-n2

GL-n2 (*abstract-algebra/group-action/burnside*), 56

Tx

Tx (*discrete-mathematics/discrete-fourier-transform*),

algo-AES-expand-key

algo-AES-expand-key (*boolean/block/AES*), 286

algo-PRESENT

algo-PRESENT (*boolean/block/present*), 291

algo-berlekamp-massey

algo-berlekamp-massey

(*boolean/stream/linear-feedback-shift-register*),

??

algo-gaussian-lattice-reduction

algo-gaussian-lattice-reduction

(*cryptography/lattice-based/lattice-reduction-algorithms*),

304

algo-gram-schmidt

algo-gram-schmidt (*linalg/euclidean-space*), 186

algo-kmeans-clustering

algo-kmeans-clustering

(*probability/ML/kmeans*), 173

algo-xgcd

algo-xgcd (*number-theory/euclidean*), 192

chinh-tac

chinh-tac (*geometry/affine-geometry/affine-space*),
106

cor-euler-path	def-aut-code
cor-euler-path (<i>discrete-mathematics/graph/index</i>), 249	def-aut-code (<i>cryptography/code-based/equivalence</i>), 381
cor-exist-map	def-autotopy
cor-exist-map (<i>geometry/affine-geometry/affine-space</i>), ??	def-autotopy (<i>abstract-algebra/groups/quasigroup</i>), 43
cor-hoi-tu-lim	def-ax-bao-toan-prob
cor-hoi-tu-lim (<i>calculus/series/index</i>), 153	def-ax-bao-toan-prob (<i>probability/intro</i>), 162
cor-mobius	def-balanced-bool
cor-mobius (<i>boolean/boolean-algebra/boolean-function</i>), 261	def-balanced-bool (<i>boolean/boolean-algebra/boolean-function</i>), 259
cor-thm-concave	def-bao-affine
cor-thm-concave (<i>calculus/calculus/concave</i>), 144	def-bao-affine (<i>geometry/affine-geometry/affine-space</i>), ??
cor-vertice	def-bernoulli-dist
cor-vertice (<i>discrete-mathematics/graph/index</i>), 237	def-bernoulli-dist (<i>probability/intro</i>), 167
def-APN	def-binary-relation
def-APN (<i>boolean/intro</i>), ??	def-binary-relation (<i>discrete-mathematics/binary-relation</i>), ??
def-PN	def-binom-dist
def-PN (<i>boolean/intro</i>), ??	def-binom-dist (<i>probability/intro</i>), 167
def-abelian-group	def-block-code
def-abelian-group (<i>abstract-algebra/groups/group</i>), 35	def-block-code (<i>cryptography/code-based/intro</i>), 372
def-additive-func	def-bool-ANF
def-additive-func (<i>algebra/set/function</i>), ??	def-bool-ANF (<i>boolean/boolean-algebra/boolean-function</i>), 257
def-affine-homomorphism	def-bool-affine
def-affine-homomorphism (<i>geometry/affine-geometry/affine-space</i>), ??	def-bool-affine (<i>boolean/boolean-algebra/boolean-affine</i>), ??
def-affine-map	def-bool-monotone
def-affine-map (<i>geometry/affine-geometry/affine-space</i>), ??	def-bool-monotone (<i>boolean/boolean-algebra/boolean-monotone</i>), 267
def-anh-xa-hop	def-card-code
def-anh-xa-hop (<i>algebra/set/map</i>), ??	def-card-code (<i>cryptography/code-based/intro</i>), 372
def-associate-index	def-cauchy-list
def-associate-index (<i>abstract-algebra/groups/quasigroup</i>), 44	def-cauchy-list (<i>calculus/calculus/monotone</i>), 135

def-cay-khung	def-cyclotomic-poly
def-cay-khung (<i>discrete-mathematics/graph/index</i>), 244	def-cyclotomic-poly (<i>cryptography/lattice-based/intro</i>), 356
def-char-polynomial	def-d-quasigroup
def-char-polynomial (<i>boolean/stream/linear-feedback-shift-register</i>), ??	def-d-quasigroup (<i>abstract-algebra/groups/quasigroup</i>), 42
def-char-polynomial-with-max-period	def-dec-lincode-1
def-char-polynomial-with-max-period (<i>boolean/stream/linear-feedback-shift-register</i>), ??	def-dec-lincode-1 (<i>cryptography/code-based/decode-on-linear-code</i>) 379
def-chuoi-hoi-tu	def-dec-lincode-2
def-chuoi-hoi-tu (<i>calculus/series/index</i>), 152	def-dec-lincode-2 (<i>cryptography/code-based/decode-on-linear-code</i>) 379
def-classical-prob	def-dec-prob-1
def-classical-prob (<i>probability/intro</i>), 157	def-dec-prob-1 (<i>cryptography/code-based/decode-problem</i>), 373
def-coder	def-dec-prob-2
def-coder (<i>cryptography/code-based/linear-code</i>), ??	def-dec-prob-2 (<i>cryptography/code-based/decode-problem</i>), 373
def-codeword	def-dec-prob-3
def-codeword (<i>cryptography/code-based/intro</i>), 372	def-dec-prob-3 (<i>cryptography/code-based/decode-problem</i>), 374
def-commutative-ring	def-decoder
def-commutative-ring (<i>abstract-algebra/rings/ring</i>), 63	def-decoder (<i>cryptography/code-based/decode-problem</i>), 374
def-concave-func	def-deg-ANF
def-concave-func (<i>calculus/calculus/concave</i>), 142	def-deg-ANF (<i>boolean/boolean-algebra/boolean-function</i>), 258
def-connected-component	def-deg-poly
def-connected-component (<i>discrete-mathematics/graph/index</i>), 242	def-deg-poly (<i>algebra/polynomial/polynomial</i>), 27
def-cont-point	def-delta-uniform
def-cont-point (<i>calculus/calculus/continuity</i>), 133	def-delta-uniform (<i>boolean/intro</i>), ??
def-cont-random-var	def-density-cont
def-cont-random-var (<i>probability/intro</i>), 168	def-density-cont (<i>probability/intro</i>), 169
def-coset	def-depend-independ-affine
def-coset (<i>abstract-algebra/groups/group</i>), 38	def-depend-independ-affine (<i>geometry/affine-geometry/affine-space</i>), ??
def-cyclic-group	def-derivative
def-cyclic-group (<i>abstract-algebra/groups/group</i>), 37	def-derivative (<i>calculus/calculus/derivative</i>), ??
def-cyclic-index	
def-cyclic-index (<i>abstract-algebra/group-action/burnside</i>), 59	

def-derivative-on-segment	def-eq-dist
def-derivative-on-segment <i>(calculus/calculus/derivative)</i> , ??	def-eq-dist (<i>probability/intro</i>), 168
def-det	def-equiv-class
def-det (<i>linalg/matrix/matrix-determinant</i>), 176	def-equiv-class (<i>discrete-mathematics/binary-relation</i>), ??
def-differential-attack	def-equiv-codes
def-differential-attack <i>(cryptanalysis/differential-attack/toycipher)</i> , 313	def-equiv-codes (<i>cryptography/code-based/equivalence</i>), 381
def-dim-lincode	def-equiv-mat-perm
def-dim-lincode (<i>cryptography/code-based/linear-code</i>), ??	def-equiv-mat-perm (<i>cryptography/code-based/equivalence</i>), 381
def-diophantus-eq	def-equiv-position
def-diophantus-eq (<i>number-theory/euclidean</i>), 191	def-equiv-position (<i>geometry/affine-geometry/affine-space</i>), ??
def-distance-two-func	def-equiv-relation
def-distance-two-func <i>(boolean/boolean-algebra/boolean-weight)</i> , 274	def-equiv-relation (<i>discrete-mathematics/binary-relation</i>), ??
def-distance-two-vecs	def-euler-cycle
def-distance-two-vecs <i>(boolean/boolean-algebra/boolean-weight)</i> , 274	def-euler-cycle (<i>discrete-mathematics/graph/index</i>), 248
def-distance-vec-to-set	def-euler-graph
def-distance-vec-to-set <i>(boolean/boolean-algebra/boolean-weight)</i> , 274	def-euler-graph (<i>discrete-mathematics/graph/index</i>), 248
def-dual-code	def-euler-path
def-dual-code (<i>cryptography/code-based/linear-code</i>), ??	def-euler-path (<i>discrete-mathematics/graph/index</i>), 248
def-eigens	def-euler-phi
def-eigens (<i>linalg/eigen</i>), ??	def-euler-phi (<i>number-theory/euler</i>), 195
def-elem-abel-group	def-even-func
def-elem-abel-group <i>(abstract-algebra/groups/group)</i> , 37	def-even-func (<i>algebra/set/function</i>), ??
def-ellipse	def-ext-sum
def-ellipse (<i>geometry/analytic-geometry/conics</i>), 91	def-ext-sum (<i>abstract-algebra/groups/group</i>), 39
def-empty-set	def-extension-field
def-empty-set (<i>algebra/set/set</i>), 13	def-extension-field <i>(abstract-algebra/galois-theory/extension-field)</i> , ??
	def-field
	def-field (<i>abstract-algebra/fields/field</i>), 63

def-full-residue	def-group-homomorphism-image
def-full-residue (<i>number-theory/euler</i>), 195	def-group-homomorphism-image (<i>abstract-algebra/homomorphisms/group-homomorphism</i>), 48
def-full-set-event	def-group-homomorphism-kernel
def-full-set-event (<i>probability/intro</i>), 164	def-group-homomorphism-kernel (<i>abstract-algebra/homomorphisms/group-homomorphism</i>), 47
def-func-dist	def-group-isomorphism
def-func-dist (<i>probability/intro</i>), 166	def-group-isomorphism (<i>abstract-algebra/homomorphisms/group-homomorphism</i>), 46
def-galois-group	def-group-monomorphism
def-galois-group (<i>abstract-algebra/galois-theory/extension-field</i>), ??	def-group-monomorphism (<i>abstract-algebra/homomorphisms/group-homomorphism</i>), 46
def-genmat-lincode	def-group-order
def-genmat-lincode (<i>cryptography/code-based/linear-lincode</i>), ??	def-group-order (<i>abstract-algebra/groups/group</i>), 36
def-graph	def-half-euler-graph
def-graph (<i>discrete-mathematics/graph/index</i>), 236	def-half-euler-graph (<i>discrete-mathematics/graph/index</i>), 248
def-graph-connected	def-ham-thuan-nhat-n
def-graph-connected (<i>discrete-mathematics/graph/index</i>), 242	def-ham-thuan-nhat-n (<i>calculus/differential-equation/index</i>), 146
def-graph-isomorphism	def-hamilton-cycle-path
def-graph-isomorphism (<i>discrete-mathematics/graph/index</i>), 239	def-hamilton-cycle-path (<i>discrete-mathematics/graph/index</i>), 249
def-graph-tree	def-hasse-diagram
def-graph-tree (<i>discrete-mathematics/graph/index</i>), 243	def-hasse-diagram (<i>discrete-mathematics/binary-relation</i>), ??
def-group	def-homotopy
def-group (<i>abstract-algebra/groups/group</i>), 35	def-homotopy (<i>abstract-algebra/groups/quasigroup</i>), 46
def-group-automorphism	def-hyperbol
def-group-automorphism (<i>abstract-algebra/homomorphisms/group-homomorphism</i>), 47	def-hyperbol (<i>geometry/analytic-geometry/conics</i>), 93
def-group-epimorphism	def-ideal
def-group-epimorphism (<i>abstract-algebra/homomorphisms/group-homomorphism</i>), 46	def-ideal (<i>abstract-algebra/fields/ideal</i>), 65
def-group-homomorphism	
def-group-homomorphism (<i>abstract-algebra/homomorphisms/group-homomorphism</i>), 45	

def-in-out-degree	def-legendre-symbol
<code>def-in-out-degree</code> (<i>discrete-mathematics/graph/index</i>), 238	<code>def-legendre-symbol</code> (<i>number-theory/quadratic-residue</i>), 198
def-independent-event	def-len-code
<code>def-independent-event</code> (<i>probability/intro</i>), 163	<code>def-len-code</code> (<i>cryptography/code-based/intro</i>), 372
def-inflection	def-len-lincode
<code>def-inflection</code> (<i>calculus/calculus/concave</i>), 145	<code>def-len-lincode</code> (<i>cryptography/code-based/linear-code</i>), ??
def-info-set	def-lim-func-as-list
<code>def-info-set</code> (<i>cryptography/code-based/linear-code</i>), ??	<code>def-lim-func-as-list</code> (<i>calculus/calculus/limits</i>), 132
def-inner-prod-code	def-lim-func-cauchy
<code>def-inner-prod-code</code> (<i>cryptography/code-based/linear-code</i>), ??	<code>def-lim-func-cauchy</code> (<i>calculus/calculus/limits</i>), 132
def-int-sum	def-lim-func-inf
<code>def-int-sum</code> (<i>abstract-algebra/groups/group</i>), 39	<code>def-lim-func-inf</code> (<i>calculus/calculus/limits</i>), 133
def-inversion	def-lim-list
<code>def-inversion</code> (<i>linalg/matrix/matrix-determinant</i>), 176	<code>def-lim-list</code> (<i>calculus/calculus/limits</i>), 131
def-isotopy	def-lim-list-inf
<code>def-isotopy</code> (<i>abstract-algebra/groups/quasigroup</i>), 43	<code>def-lim-list-inf</code> (<i>calculus/calculus/limits</i>), 132
def-jacobi-symbol	def-lim-oneside
<code>def-jacobi-symbol</code> (<i>number-theory/quadratic-residue</i>), 199	<code>def-lim-oneside</code> (<i>calculus/calculus/limits</i>), 133
def-lambda-i	def-lincode
<code>def-lambda-i</code> (<i>cryptography/lattice-based/intro</i>), 356	<code>def-lincode</code> (<i>cryptography/code-based/linear-code</i>), ??
def-latin-table	def-linear-comb
<code>def-latin-table</code> (<i>abstract-algebra/groups/quasigroup</i>), 43	<code>def-linear-comb</code> (<i>linalg/linear-combination</i>), 182
def-lattice	def-linear-depend
<code>def-lattice</code> (<i>cryptography/lattice-based/intro</i>), 353	<code>def-linear-depend</code> (<i>boolean/boolean-algebra/boolean-function</i>), 262
def-lattice-det	def-linear-dependent
<code>def-lattice-det</code> (<i>cryptography/lattice-based/intro</i>), 354	<code>def-linear-dependent</code> (<i>linalg/linear-combination</i>), 182
def-lattice-fundamental-domain	def-linear-image
<code>def-lattice-fundamental-domain</code> (<i>cryptography/lattice-based/intro</i>), 355	<code>def-linear-image</code> (<i>linalg/linear-operator</i>), 179
	def-linear-independent
	<code>def-linear-independent</code> (<i>linalg/linear-combination</i>), 182

def-linear-kernel	def-mult-group
def-linear-kernel (<i>linalg/linear-operator</i>), 179	def-mult-group (<i>abstract-algebra/groups/quasigroup</i>), 44
def-linear-map	def-multiplicative-func
def-linear-map (<i>linalg/linear-operator</i>), 179	def-multiplicative-func (<i>algebra/set/function</i>), ??
def-loop	def-nk-code
def-loop (<i>abstract-algebra/groups/quasigroup</i>), 44	def-nk-code (<i>cryptography/code-based/linear-code</i>), ??
def-map-invertible	def-nkd-code
def-map-invertible (<i>algebra/set/map</i>), ??	def-nkd-code (<i>cryptography/code-based/linear-code</i>), ??
def-mat-incidence	def-nm-code
def-mat-incidence (<i>cryptography/code-based/equivalence</i>), 382	def-nm-code (<i>cryptography/code-based/intro</i>), 372
def-mat-inv	def-nmd-code
def-mat-inv (<i>linalg/matrix/matrix-inversion</i>), 178	def-nmd-code (<i>cryptography/code-based/intro</i>), 372
def-mat-rank	def-normal-subgroup
def-mat-rank (<i>linalg/matrix/matrix-rank</i>), 178	def-normal-subgroup (<i>abstract-algebra/groups/group</i>), 38
def-maximal-ideal	def-odd-func
def-maximal-ideal (<i>abstract-algebra/fields/ideal</i>), 66	def-odd-func (<i>algebra/set/function</i>), ??
def-min-dist	def-op-quan-control
def-min-dist (<i>cryptography/code-based/intro</i>), 372	def-op-quan-control (<i>cryptography/quantum-computing</i>), 403
def-min-dist-lincode	def-op-quan-control-not
def-min-dist-lincode (<i>cryptography/code-based/linear-code</i>), ??	def-op-quan-control-not (<i>cryptography/quantum-computing</i>), 404
def-min-max	def-op-quan-hadamard
def-min-max (<i>discrete-mathematics/binary-relation</i>), ??	def-op-quan-hadamard (<i>cryptography/quantum-computing</i>), 402
def-minimal-residue	def-op-quan-id
def-minimal-residue (<i>number-theory/euler</i>), 195	def-op-quan-id (<i>cryptography/quantum-computing</i>), 402
def-minimum	def-op-quan-not
def-minimum (<i>calculus/calculus/monotone</i>), 134	def-op-quan-not (<i>cryptography/quantum-computing</i>), 402
def-mobius-func	def-orbit
def-mobius-func (<i>number-theory/mobius</i>), ??	def-orbit (<i>abstract-algebra/group-action/group-action</i>), 50
def-monotone-inc	
def-monotone-inc (<i>algebra/set/function</i>), ??	

def-ordered-relation	def-prob-conditional
def-ordered-relation (<i>discrete-mathematics/binary-relation</i>), ??	def-prob-conditional (<i>probability/intro</i>), 163
def-ordered-set	def-prob-equiv-bin-lincode
def-ordered-set (<i>discrete-mathematics/binary-relation</i>), ??	def-prob-equiv-bin-lincode (<i>cryptography/code-based/equivalence</i>), 382
def-parabol	def-prob-graph-iso
def-parabol (<i>geometry/analytic-geometry/conics</i>), 93	def-prob-graph-iso (<i>cryptography/code-based/equivalence</i>), 382
def-parastrophe	def-prob-space
def-parastrophe (<i>abstract-algebra/groups/quasigroup</i>), 44	def-prob-space (<i>probability/intro</i>), 159
def-parity-mat	def-pt-bernoulli
def-parity-mat (<i>cryptography/code-based/linear-code</i>), ??	def-pt-bernoulli (<i>calculus/differential-equation/index</i>), 148
def-parity-vec	def-pt-clairaut
def-parity-vec (<i>cryptography/code-based/linear-code</i>), ??	def-pt-clairaut (<i>calculus/differential-equation/index</i>), 151
def-periodic-func	def-pt-lagrange
def-periodic-func (<i>algebra/set/function</i>), ??	def-pt-lagrange (<i>calculus/differential-equation/index</i>), 151
def-perm	def-pt-tach-bien
def-perm (<i>cryptography/code-based/equivalence</i>), 380	def-pt-tach-bien (<i>calculus/differential-equation/index</i>), 145
def-phang-tong	def-ptvp-thuan-nhat
def-phang-tong (<i>geometry/affine-geometry/affine-space</i>), ??	def-ptvp-thuan-nhat (<i>calculus/differential-equation/index</i>), 146
def-plane	def-ptvp-toan-phan
def-plane (<i>geometry/affine-geometry/affine-space</i>), ??	def-ptvp-toan-phan (<i>calculus/differential-equation/index</i>), 149
def-poisson-dist	def-ptvp-tuyen-tinh
def-poisson-dist (<i>probability/intro</i>), 168	def-ptvp-tuyen-tinh (<i>calculus/differential-equation/index</i>), 146
def-prec	def-quadratic-residue
def-prec (<i>discrete-mathematics/binary-relation</i>), ??	def-quadratic-residue (<i>number-theory/quadratic-residue</i>), 198
def-primitive-poly	def-quasi-linear-depend
def-primitive-poly (<i>boolean/stream/linear-feedback-shift-register</i>), ??	def-quasi-linear-depend (<i>boolean/boolean-algebra/boolean-function</i>), 262
def-principal-ideal	
def-principal-ideal (<i>abstract-algebra/fields/ideal</i>), 66	

def-quasigroup	def-spec-code
def-quasigroup (<i>abstract-algebra/groups/quasigroup</i>), def-spec-code (<i>cryptography/code-based/intro</i>), 373 42	
def-quot-group	def-spec-lincode
def-quot-group (<i>abstract-algebra/groups/group</i>), 39	def-spec-lincode (<i>cryptography/code-based/linear-code</i>), ??
def-quot-set	def-spec-lincode-reg
def-quot-set (<i>discrete-mathematics/binary-relation</i>), def-spec-lincode-reg (<i>cryptography/code-based/weight</i>), 378 ??	
def-random-var	def-speed-lincode
def-random-var (<i>probability/intro</i>), 166	def-speed-lincode (<i>cryptography/code-based/linear-code</i>), ??
def-redundancy-lincode	def-splitting-field
def-redundancy-lincode (<i>cryptography/code-based/linear-code</i>), ??	def-splitting-field (<i>abstract-algebra/galois-theory/extension-field</i>), ??
def-relate-under-action	def-stabilizer
def-relate-under-action (<i>abstract-algebra/group-action/group-action</i>), def-stabilizer (<i>abstract-algebra/group-action/group-action</i>), 50 51	
def-ring	def-subgraph
def-ring (<i>abstract-algebra/rings/ring</i>), 61	def-subgraph (<i>discrete-mathematics/graph/index</i>), 239
def-ring-char	def-subgroup
def-ring-char (<i>abstract-algebra/rings/ring</i>), 62	def-subgroup (<i>abstract-algebra/groups/group</i>), 36
def-ring-hom-kernel	def-subset
def-ring-hom-kernel (<i>abstract-algebra/fields/ring-homomorphism</i>), def-subset (<i>algebra/set/set</i>), 13 67	
def-ring-homomorphism	def-syndrom
def-ring-homomorphism (<i>abstract-algebra/fields/ring-homomorphism</i>), 67	def-syndrom (<i>cryptography/code-based/linear-code</i>), ??
def-set-intersection	def-syndrom-decode
def-set-intersection (<i>algebra/set/set</i>), 14	def-syndrom-decode (<i>cryptography/code-based/decode-on-linear-code</i>), 379
def-set-minus	def-syndrom-recognize
def-set-minus (<i>algebra/set/set</i>), 14	def-syndrom-recognize (<i>cryptography/code-based/decode-on-linear-code</i>), 380
def-set-of-ind-event	def-syndrom-search
def-set-of-ind-event (<i>probability/intro</i>), 164	def-syndrom-search (<i>cryptography/code-based/decode-on-linear-code</i>), 379
def-slid-pair	
def-slid-pair (<i>cryptanalysis/slide-attack/index</i>), ??	

def-systematic-code	definition-0
def-systematic-code (<i>cryptography/code-based/linear-code</i>), ??	definition-0 (<i>algebra/set/map</i>), ??
def-systematic-lincode	definition-10
def-systematic-lincode (<i>cryptography/code-based/linear-code</i>), ??	definition-10 (<i>geometry/affine-geometry/affine-space</i>), ??
def-systematic-mat	definition-3
def-systematic-mat (<i>cryptography/code-based/linear-code</i>), ??	definition-3 (<i>abstract-algebra/rings/ring</i>), 62
def-target-affine	definition-4
def-target-affine (<i>geometry/affine-geometry/affine-space</i>), ??	definition-4 (<i>boolean/boolean-algebra/walsh-hadamard-spectre</i>), 280
def-tich-Descartes	exp-CRT-problem
def-tich-Descartes (<i>algebra/set/map</i>), ??	exp-CRT-problem (<i>number-theory/chinese-remainder-theorem</i>), 203
def-translation	exp-DFT-3
def-translation (<i>geometry/analytic-geometry/transformation</i>), 94	exp-DFT-3 (<i>discrete-mathematics/discrete-fourier-transform</i>), 213
def-trong-tam	exp-DFT-4
def-trong-tam (<i>geometry/affine-geometry/affine-space</i>), ??	exp-DFT-4 (<i>discrete-mathematics/discrete-fourier-transform</i>), 214
def-types-ordered-relation	exp-FFT-4
def-types-ordered-relation (<i>discrete-mathematics/binary-relation</i>), ??	exp-FFT-4 (<i>discrete-mathematics/discrete-fourier-transform</i>), 217
def-types-relation	exp-additive-func
def-types-relation (<i>discrete-mathematics/binary-relation</i>), ??	exp-additive-func (<i>algebra/set/function</i>), ??
def-union-set	exp-adp-1
def-union-set (<i>algebra/set/set</i>), 14	exp-adp-1 (<i>cryptanalysis/differential-attack/difference-and-differential</i> ??
def-univariate-poly	exp-affine-map-fixed
def-univariate-poly (<i>algebra/polynomial/polynomial</i>), 26	exp-affine-map-fixed (<i>geometry/affine-geometry/affine-space</i>), ??
def-vector-comparable	exp-affine-map-id
def-vector-comparable (<i>boolean/boolean-algebra/boolean-monotone</i>), 266	exp-affine-map-id (<i>geometry/affine-geometry/affine-space</i>), ??
def-weight-bool	exp-affine-map-parallel
def-weight-bool (<i>boolean/boolean-algebra/boolean-function</i>), 258	exp-affine-map-parallel (<i>geometry/affine-geometry/affine-space</i>), ??
	exp-affine-space
	exp-affine-space (<i>geometry/affine-geometry/affine-space</i>), 105

exp-anh-xa-tich-Descartes	exp-classical-prob-dice
<code>exp-anh-xa-tich-Descartes</code> (<i>algebra/set/map</i>), ??	<code>exp-classical-prob-dice</code> (<i>probability/intro</i>), 158
exp-automorphism-1	exp-conjunctive-normal-form
<code>exp-automorphism-1</code> (<i>abstract-algebra/galois-theory/extension-field</i>), ??	<code>exp-conjunctive-normal-form</code> (<i>boolean/boolean-algebra/boolean-dnf-cnf</i>), 270
exp-automorphism-2	exp-convert-to-full-CNF
<code>exp-automorphism-2</code> (<i>abstract-algebra/galois-theory/extension-field</i>), ??	<code>exp-convert-to-full-CNF</code> (<i>boolean/boolean-algebra/boolean-dnf-cnf</i>), 270
exp-bent-func	exp-convert-to-full-DNF
<code>exp-bent-func</code> (<i>boolean/boolean-algebra/walsh-hadamard-spectre</i>), 281	<code>exp-convert-to-full-DNF</code> (<i>boolean/boolean-algebra/boolean-dnf-cnf</i>), 269
exp-bijection	exp-cup
<code>exp-bijection</code> (<i>algebra/set/map</i>), ??	<code>exp-cup</code> (<i>discrete-mathematics/combinatoric</i>), 227
exp-binary-relation	exp-cyclic-index
<code>exp-binary-relation</code> (<i>discrete-mathematics/binary-relation</i>), ??	<code>exp-cyclic-index</code> (<i>abstract-algebra/group-action/burnside</i>), 59
exp-binom-dist	exp-cyclotomic-poly
<code>exp-binom-dist</code> (<i>probability/intro</i>), 167	<code>exp-cyclotomic-poly</code> (<i>cryptography/lattice-based/intro</i>), 356
exp-bool-affine-linear	exp-deg-ANF
<code>exp-bool-affine-linear</code> (<i>boolean/boolean-algebra/boolean-affine</i>), ??	<code>exp-deg-ANF</code> (<i>boolean/boolean-algebra/boolean-function</i>), 258
exp-bool-monotone	exp-derivative
<code>exp-bool-monotone</code> (<i>boolean/boolean-algebra/boolean-monotone</i>), 267	<code>exp-derivative</code> (<i>calculus/calculus/derivative</i>), ??
exp-cadinality	exp-derivative-2
<code>exp-cadinality</code> (<i>algebra/set/cadinality</i>), 15	<code>exp-derivative-2</code> (<i>calculus/calculus/derivative</i>), ??
exp-char-poly-with-max-period	exp-det
<code>exp-char-poly-with-max-period</code> (<i>boolean/stream/linear-feedback-shift-register</i>), ??	<code>exp-det</code> (<i>linalg/matrix/matrix-determinant</i>), 177
exp-chuoi-hoi-tu	exp-direct-proof
<code>exp-chuoi-hoi-tu</code> (<i>calculus/series/index</i>), 152	<code>exp-direct-proof</code> (<i>algebra/proof</i>), 29
exp-chuyen-trang-thai	exp-disjunctive-normal-form
<code>exp-chuyen-trang-thai</code> (<i>boolean/stream/linear-feedback-shift-register</i>), ??	<code>exp-disjunctive-normal-form</code> (<i>boolean/boolean-algebra/boolean-dnf-cnf</i>), 268
exp-classical-prob-coin	exp-eq-dist
<code>exp-classical-prob-coin</code> (<i>probability/intro</i>), 158	<code>exp-eq-dist</code> (<i>probability/intro</i>), 161

exp-equiv-pos-3d

`exp-equiv-pos-3d` (*geometry/affine-geometry/affine-space*), ??

exp-expand-sum-prob

`exp-expand-sum-prob` (*probability/intro*), 160

exp-field

`exp-field` (*abstract-algebra/fields/field*), 63

exp-field-poly

`exp-field-poly` (*abstract-algebra/fields/field*), 64

exp-field-q-i

`exp-field-q-i` (*abstract-algebra/galois-theory/extension-field*), ??

exp-field-q-sqrt2-i

`exp-field-q-sqrt2-i` (*abstract-algebra/galois-theory/extension-field*), ??

exp-fourier

`exp-fourier` (*boolean/boolean-algebra/walsh-hadamard-spectre*), 276

exp-full-disjunctive-normal-form

`exp-full-disjunctive-normal-form` (*boolean/boolean-algebra/boolean-dnf-cnf*), 269

exp-full-set-event

`exp-full-set-event` (*probability/intro*), 164

exp-full-set-event-sol

`exp-full-set-event-sol` (*probability/intro*), 164

exp-function

`exp-function` (*algebra/set/function*), ??

exp-function-bijection

`exp-function-bijection` (*algebra/set/function*), ??

exp-function-with-Descartes

`exp-function-with-Descartes` (*algebra/set/function*), ??

exp-gcd

`exp-gcd` (*number-theory/euclidean*), 190

exp-gram-schmidt

`exp-gram-schmidt` (*linalg/euclidean-space*), 186

exp-graph

`exp-graph` (*discrete-mathematics/graph/index*), 236

exp-graph-of-funcs

`exp-graph-of-funcs` (*algebra/set/function*), ??

exp-group-Q

`exp-group-Q` (*abstract-algebra/groups/group*), 36

exp-group-Z

`exp-group-Z` (*abstract-algebra/groups/group*), 36

exp-hasse-diagram

`exp-hasse-diagram` (*discrete-mathematics/binary-relation*), ??

exp-independent-cycles

`exp-independent-cycles` (*abstract-algebra/group-action/burnside*), 58

exp-independent-cycles-2

`exp-independent-cycles-2` (*abstract-algebra/groups/symmetric-group*), ??

exp-induction

`exp-induction` (*algebra/proof*), 30

exp-induction-strong

`exp-induction-strong` (*algebra/proof*), 31

exp-inversion

`exp-inversion` (*linalg/matrix/matrix-determinant*), 176

exp-khai-trien-bool

`exp-khai-trien-bool` (*boolean/boolean-algebra/boolean-function*), 257

exp-lagrange-interpolation

`exp-lagrange-interpolation` (*algebra/polynomial/lagrange-interpolation*), 29

exp-laplas-1

`exp-laplas-1` (*abstract-algebra/homomorphisms/group-homomorphisms*), 49

exp-lim-list	exp-prod-independent-cycles
exp-lim-list (<i>calculus/calculus/limits</i>), 132	exp-prod-independent-cycles (<i>abstract-algebra/groups/symmetric-group</i>), ??
exp-mat-rank	exp-pt-tttn
exp-mat-rank (<i>linalg/matrix/matrix-rank</i>), 179	exp-pt-tttn (<i>calculus/differential-equation/index</i>), 147
exp-max-ideal-field	exp-q-sqrt2
exp-max-ideal-field (<i>abstract-algebra/fields/ideal</i>), 67	exp-q-sqrt2 (<i>abstract-algebra/galois-theory/extension-field</i>), ??
exp-mobius	exp-quasi-linear
exp-mobius (<i>boolean/boolean-algebra/boolean-function</i>), 260	exp-quasi-linear (<i>boolean/boolean-algebra/boolean-function</i>), 264
exp-monotone	exp-quasigroup-1
exp-monotone (<i>algebra/set/function</i>), ??	exp-quasigroup-1 (<i>abstract-algebra/groups/quasigroup</i>), 42
exp-nim-game	exp-quasigroup-2
exp-nim-game (<i>stuff/game/index</i>), 611	exp-quasigroup-2 (<i>abstract-algebra/groups/quasigroup</i>), 42
exp-op-quan-hadamard	exp-qubit
exp-op-quan-hadamard (<i>cryptography/quantum-computing</i>), 402	exp-qubit (<i>cryptography/quantum-computing</i>), 402
exp-orbit	exp-quot-group
exp-orbit (<i>abstract-algebra/group-action/group-action</i>), 50	exp-quot-group (<i>abstract-algebra/groups/group</i>), 39
exp-order-independent-cycles	exp-quot-set
exp-order-independent-cycles (<i>abstract-algebra/groups/symmetric-group</i>), ??	exp-quot-set (<i>discrete-mathematics/binary-relation</i>), ??
exp-periodic-func	exp-solve-diophantus-eq
exp-periodic-func (<i>algebra/set/function</i>), ??	exp-solve-diophantus-eq (<i>number-theory/euclidean</i>), 191
exp-perm	exp-splitting-field-1
exp-perm (<i>cryptography/code-based/equivalence</i>), 381	exp-splitting-field-1 (<i>abstract-algebra/galois-theory/extension-field</i>), ??
exp-perm-operation	exp-splitting-field-2
exp-perm-operation (<i>abstract-algebra/groups/symmetric-group</i>), ??	exp-splitting-field-2 (<i>abstract-algebra/galois-theory/extension-field</i>), ??
exp-phan-chung	exp-stirling-1
exp-phan-chung (<i>algebra/proof</i>), 33	exp-stirling-1 (<i>discrete-mathematics/partition</i>), 211
exp-pm-poly	
exp-pm-poly (<i>algebra/polynomial/polynomial</i>), ??	
exp-poisson-dist	
exp-poisson-dist (<i>probability/intro</i>), 161	

exp-subgroup**exp-subgroup** (*abstract-algebra/groups/group*), 36**exp-symmetric-group****exp-symmetric-group***(abstract-algebra/groups/symmetric-group)*, ??**exp-sys-eqs****exp-sys-eqs** (*linalg/vector-space*), ??**exp-thr-pm-poly****exp-thr-pm-poly** (*algebra/polynomial/polynomial*), ??**exp-tich-Descartes****exp-tich-Descartes** (*algebra/set/map*), ??**exp-uncontinuous****exp-uncontinuous** (*calculus/calculus/limits*), 133**exp-vector-comparable****exp-vector-comparable***(boolean/boolean-algebra/boolean-monotone)*, 267**exp-weight-bool****exp-weight-bool** (*boolean/boolean-algebra/boolean-function*), 258**exp-xdp-1****exp-xdp-1** (*cryptanalysis/differential-attack/difference*), 326**exp-xgcd****exp-xgcd** (*number-theory/euclidean*), 193**exp-xgcd-matrix****exp-xgcd-matrix** (*number-theory/euclidean*), 193**fermat-lemma****fermat-lemma** (*calculus/calculus/avarage-value*), 138**fi****fi** (*discrete-mathematics/discrete-fourier-transform*), 214**hoff-algo-3-40****hoff-algo-3-40** (*solutions/an-introduction-to-mathematical-cryptogr*), 452**hoff-algo-7-45****hoff-algo-7-45** (*solutions/an-introduction-to-mathematical-cryptogr*), 455**lem-burnside****lem-burnside** (*abstract-algebra/group-action/burnside*), 52**lem-coppersmith****lem-coppersmith** (*cryptography/lattice-based/coppersmith*), 369**lem-nm-code****lem-nm-code** (*cryptography/code-based/intro*), 373**lhopital-rule****lhopital-rule** (*calculus/calculus/avarage-value*), 142**maximum****maximum** (*calculus/calculus/monotone*), 134**nsu22-exp-super-sbox****nsu22-exp-super-sbox** (2022/nsucrypto), 475**nsu23-algo-primes****nsu23-algo-primes** (2023/nsucrypto/index), 480**nsu23-col-funcs****nsu23-col-funcs** (2023/nsucrypto/index), 482**nsu23-cor-col-funcs****nsu23-cor-col-funcs** (2023/nsucrypto/index), 483**nsu23-exp-code****nsu23-exp-code** (2023/nsucrypto/index), 485**nsu23-exp-code-2****nsu23-exp-code-2** (2023/nsucrypto/index), 487**nsu23-rmk-code****nsu23-rmk-code** (2023/nsucrypto/index), 485**nsu23-thm-col-funcs****nsu23-thm-col-funcs** (2023/nsucrypto/index), 482**nsu24-rmk-sp****nsu24-rmk-sp** (2024/nsucrypto/index), 539**nsu24-rmk-sp-solve****nsu24-rmk-sp-solve** (2024/nsucrypto/index), 540

parallel	rmk-automorphism
parallel (<i>geometry/affine-geometry/affine-space</i>), ??	rmk-automorphism (<i>abstract-algebra/galois-theory/extension-field</i>), ??
phuong-giao	rmk-bent
phuong-giao (<i>geometry/affine-geometry/affine-space</i>), rmk-bent (<i>boolean/boolean-algebra/walsh-hadamard-spectre</i>), 280	??
prop-eigens	rmk-code-T
prop-eigens (<i>linalg/eigen</i>), ??	rmk-code-T (<i>cryptography/code-based/linear-code</i>), ??
prop-eigens-rank-trace	rmk-conj-aut
prop-eigens-rank-trace (<i>linalg/eigen</i>), ??	rmk-conj-aut (<i>abstract-algebra/galois-theory/extension-field</i>), ??
prop-kernel-image	rmk-coset
prop-kernel-image (<i>linalg/linear-operator</i>), 180	rmk-coset (<i>abstract-algebra/groups/group</i>), 38
prop-mobius	rmk-cyclotomic-poly
prop-mobius (<i>number-theory/mobius</i>), ??	rmk-cyclotomic-poly (<i>cryptography/lattice-based/intro</i>), 356
prop-ring	rmk-cyclotomic-poly-Q
prop-ring (<i>abstract-algebra/rings/ring</i>), 62	rmk-cyclotomic-poly-Q (<i>cryptography/lattice-based/intro</i>), 356
prop-stirling-1	rmk-dec-prob
prop-stirling-1 (<i>discrete-mathematics/partition</i>), 211	rmk-dec-prob (<i>cryptography/code-based/decode-problem</i>), 374
prop-weight	rmk-density-cont
prop-weight (<i>boolean/boolean-algebra/boolean-function</i>), 258	rmk-density-cont (<i>probability/intro</i>), 170
quasi-fg	rmk-density-disc
quasi-fg (<i>boolean/boolean-algebra/boolean-function</i>), 263	rmk-density-disc (<i>probability/intro</i>), 169
recurr-1-order-1	rmk-depend
recurr-1-order-1 (<i>discrete-mathematics/recurrence-relation</i>), 229	rmk-depend (<i>boolean/boolean-algebra/boolean-function</i>), 259
recurr-1-order-2	rmk-depend-balanced
recurr-1-order-2 (<i>discrete-mathematics/recurrence-relation</i>), 229	rmk-depend-balanced (<i>boolean/boolean-algebra/boolean-function</i>), 259
recurr-2-order	rmk-der-extremum
recurr-2-order (<i>discrete-mathematics/recurrence-relation</i>), 230	rmk-der-extremum (<i>calculus/calculus monotone</i>), 134
rmk-amount-associate-index	rmk-diff-and-der
rmk-amount-associate-index (<i>abstract-algebra/groups/quasigroup</i>), 45	rmk-diff-and-der (<i>calculus/calculus/derivative</i>), ??

rmk-differential-attack	rmk-group-homomorphism
rmk-differential-attack (cryptanalysis/differential-attack/toycipher), 315	rmk-group-homomorphism (abstract-algebra/homomorphisms/group-homomorphism), 45
rmk-dim-lattice	rmk-homomorphism
rmk-dim-lattice (cryptography/lattice-based/intro), 354	rmk-homomorphism (abstract-algebra/homomorphisms/group-homomo- 47
rmk-dual-code	rmk-hyperplane-i
rmk-dual-code (cryptography/code-based/linear-code), rmk-hyperplane-i (geometry/affine-geometry/affine-space), ??	??
rmk-ellipse-to-circle	rmk-independent-cycles
rmk-ellipse-to-circle (geometry/analytic-geometry/conics), 93	rmk-independent-cycles (abstract-algebra/group-action/burnside), 59
rmk-euler-fermat	rmk-induction
rmk-euler-fermat (number-theory/euler), 198	rmk-induction (algebra/proof), 31
rmk-euler-phi	rmk-inflection-tangent
rmk-euler-phi (number-theory/euler), 195	rmk-inflection-tangent (calculus/calculus/concave), 145
rmk-extension-field	rmk-isomorphism
rmk-extension-field (abstract-algebra/galois-theory/extension-field), ??	rmk-isomorphism (geometry/affine-geometry/affine-space), ??
rmk-field-equiv	rmk-isotopy
rmk-field-equiv (abstract-algebra/galois-theory/extension-field), ??	rmk-isotopy (abstract-algebra/groups/quasigroup), 43
rmk-fourier	rmk-lattice-fundamental-domain
rmk-fourier (boolean/boolean-algebra/walsh-hadamard-spectre), 274	rmk-lattice-fundamental-domain (cryptography/lattice-based/intro), 355
rmk-fourier-and-walsh	rmk-line-oxy
rmk-fourier-and-walsh (boolean/boolean-algebra/walsh-hadamard-spectre), 279	rmk-line-oxy (geometry/affine-geometry/affine-space), ??
rmk-genmat-lincode	rmk-line-oxyz
rmk-genmat-lincode (cryptography/code-based/linear-code), ??	rmk-line-oxyz (geometry/affine-geometry/affine-space), ??
rmk-geo-to-alg	rmk-mat-rank
rmk-geo-to-alg (geometry/affine-geometry/affine-space), ??	rmk-mat-rank (linalg/matrix/matrix-rank), 178
rmk-group-automorphisms	rmk-maximal-ideal
rmk-group-automorphisms (abstract-algebra/galois-theory/extension-field), ??	rmk-maximal-ideal (abstract-algebra/fields/ideal), 66

rmk-min-max	rmk-prod-independent-cycles
rmk-min-max (<i>discrete-mathematics/binary-relation</i>), ??	rmk-prod-independent-cycles (<i>abstract-algebra/groups/symmetric-group</i>), ??
rmk-mobius	rmk-properties-eq-dist
rmk-mobius (<i>boolean/boolean-algebra/boolean-function</i>), 261	rmk-properties-eq-dist (<i>probability/intro</i>), 168
rmk-mobius-trans	rmk-property-bent
rmk-mobius-trans (<i>boolean/boolean-algebra/boolean-function</i>), 261	rmk-property-bent (<i>boolean/boolean-algebra/walsh-hadamard-spectrum</i>), 280
rmk-op-quan-control	rmk-property-euler-phi
rmk-op-quan-control (<i>cryptography/quantum-computing</i>), 404	rmk-property-euler-phi (<i>number-theory/euler</i>), 195
rmk-orbit	rmk-property-euler-phi-2
rmk-orbit (<i>abstract-algebra/group-action/group-action</i>), 50	rmk-property-euler-phi-2 (<i>number-theory/euler</i>), 196
rmk-ortho	rmk-property-tam-ti-cu
rmk-ortho (<i>linalg/euclidean-space</i>), 187	rmk-property-tam-ti-cu (<i>geometry/affine-geometry/affine-space</i>), ??
rmk-parity-mat	rmk-property-walsh
rmk-parity-mat (<i>cryptography/code-based/linear-code</i>), ??	rmk-property-walsh (<i>boolean/boolean-algebra/walsh-hadamard-spectrum</i>), 280
rmk-parity-matvec	rmk-purpose-quasigroup
rmk-parity-matvec (<i>cryptography/code-based/linear-code</i>), ??	rmk-purpose-quasigroup (<i>abstract-algebra/groups/quasigroup</i>), 45
rmk-perm-2-rows	rmk-quasi-linear-balanced
rmk-perm-2-rows (<i>abstract-algebra/groups/symmetric-group</i>), ??	rmk-quasi-linear-balanced (<i>boolean/boolean-algebra/boolean-function</i>), 263
rmk-plane	rmk-quasigroup
rmk-plane (<i>geometry/affine-geometry/affine-space</i>), ??	rmk-quasigroup (<i>abstract-algebra/groups/quasigroup</i>), 42
rmk-plane-oxyz	rmk-relate-under-action
rmk-plane-oxyz (<i>geometry/affine-geometry/affine-space</i>), ??	rmk-relate-under-action (<i>abstract-algebra/group-action/group-action</i>), 51
rmk-poly-0	rmk-ring
rmk-poly-0 (<i>algebra/polynomial/polynomial</i>), ??	rmk-ring (<i>abstract-algebra/rings/ring</i>), 62
rmk-principal-ideal	rmk-roots-perm
rmk-principal-ideal (<i>abstract-algebra/fields/ideal</i>), 66	rmk-roots-perm (<i>abstract-algebra/galois-theory/extension-field</i>), ??
rmk-prob-conditional	
rmk-prob-conditional (<i>probability/intro</i>), 163	

rmk-set-of-ind-event	thm-ax-bao-toan-prob
rmk-set-of-ind-event (<i>probability/intro</i>), 164	thm-ax-bao-toan-prob (<i>probability/intro</i>), 162
rmk-stirling-1	thm-bayes
rmk-stirling-1 (<i>discrete-mathematics/partition</i>), 211	thm-bayes (<i>probability/intro</i>), 165
rmk-subset	thm-bouton
rmk-subset (<i>algebra/set/set</i>), 13	thm-bouton (<i>stuff/game/index</i>), 610
rmk-sum-code	thm-cauchy
rmk-sum-code (<i>cryptography/code-based/linear-code</i>), ??	thm-cauchy (<i>calculus/calculus/avarage-value</i>), 141
rmk-sys-eqs	thm-cauchy-criteria
rmk-sys-eqs (<i>linalg/vector-space</i>), ??	thm-cauchy-criteria (<i>calculus/calculus monotone</i>), 135
rmk-target-affine	thm-cauchy-criteria-chuoi
rmk-target-affine (<i>geometry/affine-geometry/affine-space</i>), ??	thm-cauchy-criteria-chuoi (<i>calculus/series/index</i>), 152
rmk-tong-0-phang	thm-cayley
rmk-tong-0-phang (<i>geometry/affine-geometry/affine-space</i>), ??	thm-cayley (<i>abstract-algebra/homomorphisms/group-homomorphism</i> 46)
rmk-trong-tam	thm-concave
rmk-trong-tam (<i>geometry/affine-geometry/affine-space</i>), ??	thm-concave (<i>calculus/calculus/concave</i>), 142
rmk-union-orbits	thm-conditions-ptvp-toan-phan
rmk-union-orbits (<i>abstract-algebra/group-action/group-action</i>), 50	thm-conditions-ptvp-toan-phan (<i>calculus/differential-equation/index</i>), 149
rmk-weight	thm-connected-graph-euler-cycle
rmk-weight (<i>boolean/boolean-algebra/boolean-function</i>), 262	thm-connected-graph-euler-cycle (<i>discrete-mathematics/graph/index</i>), 248
theo-dim	thm-criteria-compare
theo-dim (<i>geometry/affine-geometry/affine-space</i>), ??	thm-criteria-compare (<i>calculus/series/index</i>), 153
theorem-1	thm-criteria-integral-cauchy
theorem-1 (<i>cryptography/code-based/weight</i>), 378	thm-criteria-integral-cauchy (<i>calculus/series/index</i>), 154
thm-affine-homomorphism	thm-daisy-chain
thm-affine-homomorphism (<i>geometry/affine-geometry/affine-space</i>), ??	thm-daisy-chain (<i>discrete-mathematics/graph/index</i>), 245
thm-affine-map-comb	thm-define-affine
thm-affine-map-comb (<i>geometry/affine-geometry/affine-space</i>), ??	thm-define-affine (<i>geometry/affine-geometry/affine-space</i>), ??

thm-der-monotone	thm-ineq-cauchy-schwarz
thm-der-monotone (<i>calculus/calculus/monotone</i>), 134	thm-ineq-cauchy-schwarz (<i>linalg/euclidean-space</i>), 185
thm-derivative-comb	thm-inflection
thm-derivative-comb (<i>calculus/calculus/derivative</i>), ??	thm-inflection (<i>calculus/calculus/concave</i>), 145
thm-dim	thm-kuratowski
thm-dim (<i>linalg/vector-space</i>), ??	thm-kuratowski (<i>discrete-mathematics/graph/index</i>), 241
thm-euler	thm-lagrange
thm-euler (<i>number-theory/euler</i>), 197	thm-lagrange (<i>calculus/calculus/avarage-value</i>), 139
thm-euler-formula	thm-lagrange-group
thm-euler-formula (<i>discrete-mathematics/graph/index</i>), 241	thm-lagrange-group (<i>abstract-algebra/groups/group</i>), 37
thm-expand-sum-prob	thm-laplace
thm-expand-sum-prob (<i>probability/intro</i>), 161	thm-laplace (<i>linalg/matrix/matrix-determinant</i>), 178
thm-fermal-mini	thm-linear-map
thm-fermal-mini (<i>number-theory/euler</i>), 198	thm-linear-map (<i>geometry/affine-geometry/affine-space</i>), ??
thm-first-criteria-hoi-tu	thm-max-ideal-field
thm-first-criteria-hoi-tu (<i>calculus/series/index</i>), 153	thm-max-ideal-field (<i>abstract-algebra/fields/ideal</i>), 66
thm-first-isomorphism	thm-n-1-dependence
thm-first-isomorphism (<i>abstract-algebra/homomorphisms/group-homomorphism</i>), 48	thm-n-1-dependence (<i>geometry/affine-geometry/affine-space</i>), ??
thm-full-set-event	thm-number-cay-khung
thm-full-set-event (<i>probability/intro</i>), 164	thm-number-cay-khung (<i>discrete-mathematics/graph/index</i>), 247
thm-galois-group-extension	thm-phang-ss
thm-galois-group-extension (<i>abstract-algebra/galois-theory/extension-field</i>), ??	thm-phang-ss (<i>geometry/affine-geometry/affine-space</i>), ??
thm-giao-2-phang	thm-phang-ss-x
thm-giao-2-phang (<i>geometry/affine-geometry/affine-space</i>), ??	thm-phang-ss-x (<i>geometry/affine-geometry/affine-space</i>), ??
thm-hadamard-ineq	thm-plane-graph
thm-hadamard-ineq (<i>cryptography/lattice-based/intro</i>), 356	thm-plane-graph (<i>discrete-mathematics/graph/index</i>), 241
thm-in-out-degree	thm-prod-of-prob-space
thm-in-out-degree (<i>discrete-mathematics/graph/index</i>), 238	thm-prod-of-prob-space (<i>probability/intro</i>), 162

thm-pythagoras

thm-pythagoras (*geometry/analytic-geometry/intro*),
80

thm-ring-kernel-ideal

thm-ring-kernel-ideal
(*abstract-algebra/fields/ring-homomorphism*),
68

thm-rolle

thm-rolle (*calculus/calculus/avarage-value*), 139

thm-syndrome-dec

thm-syndrome-dec (*cryptography/code-based/decode-on-linear-code*),
380

thm-tam-ti-cu

thm-tam-ti-cu (*geometry/affine-geometry/affine-space*),
??

thm-tam-ti-cu-coeff

thm-tam-ti-cu-coeff
(*geometry/affine-geometry/affine-space*),
??

thm-thales-2d

thm-thales-2d (*geometry/analytic-geometry/intro*),
78

thm-thales-3d

thm-thales-3d (*geometry/analytic-geometry/intro*),
79

thm-vertice-edges

thm-vertice-edges (*discrete-mathematics/graph/index*),
236

thr-about-R

thr-about-R (*algebra/set/infinity*), 25

thr-deg-pm-poly

thr-deg-pm-poly (*algebra/polynomial/polynomial*),
??