**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KỲ MÔN**

**NHẬP MÔN BẢO MẬT THÔNG TIN**

**CÁC PHƯƠNG PHÁP**

**MÃ HOÁ CỔ ĐIỂN**

*Người hướng dẫn :* **GV. HỒ THỊ THANH TUYẾN**

*Người thực hiện :* **Nguyễn Võ Hoàng Vũ – 51900286**

**Nguyễn Trường Anh – 51900699**

**Lạc Minh Long – 51900758**

**Trần Quang Vinh – 51900781**

**Lê Sỹ Tiến – 52000810**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KỲ MÔN**

**NHẬP MÔN BẢO MẬT THÔNG TIN**

**CÁC PHƯƠNG PHÁP**

**MÃ HOÁ CỔ ĐIỂN**

*Người hướng dẫn :* **GV. HỒ THỊ THANH TUYẾN**

*Người thực hiện :* **Nguyễn Võ Hoàng Vũ – 51900286**

**Nguyễn Trường Anh – 51900699**

**Lạc Minh Long – 51900758**

**Trần Quang Vinh – 51900781**

**Lê Sỹ Tiến – 52000810**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

**LỜI CẢM ƠN**

Trước tiên, chúng tôi xin gửi lời cảm ơn chân thành và lòng biết ơn sâu sắc đến GV. Hồ Thị Thanh Tuyến. Người đã luôn hỗ trợ và hướng dẫn tận tình cho chúng tôi trong suốt quá trình nghiên cứu và hoàn thành bài đồ án cuối kỳ này.

Bên cạnh đó, chúng tôi cũng xin gửi lời cảm ơn đến khoa Công nghệ thông tin trường Đại học Tôn Đức Thắng, vì đã tạo điều kiện cho chúng tôi được học tập và nghiên cứu môn học này. Khoa đã luôn sẵn sàng chia sẻ và mang tới những kiến thức bổ ích, cũng như những kinh nghiệm cho chúng tôi trong quá trình tra cứu, tham khảo tài liệu. Điều đó đã giúp ích không chỉ cho việc thực hiện và hoàn thành đề tài nghiên cứu, mà còn giúp ích cho việc học tập, rèn luyện trong quá trình thực hành tại trường Đại học Tôn Đức Thắng nói chung.

Cuối cùng, sau khoảng thời gian học tập trên lớp, chúng tôi đã hoàn tất đề tài nghiên cứu nhờ vào sự hướng dẫn, giúp đỡ và những kiến thức mà quý thầy cô đã mang đến. Do giới hạn về mặt kiến thức và khả năng lý luận nên chúng tôi vẫn còn nhiều thiếu sót và hạn chế. Kính mong sự chỉ dẫn và đóng góp từ quý thầy cô để bài nghiên cứu này của chúng tôi được hoàn thiện hơn. Hơn nữa, nhờ những lời góp ý từ thầy cô và bạn bè, chúng tôi sẽ hoàn thành tốt hơn ở những bài nghiên cứu tiếp theo trong tương lai. Chúng tôi mong quý thầy cô và bạn bè – những người luôn quan tâm và hỗ trợ chúng tôi – luôn tràn đầy sức khoẻ và sự bình an.

**XIN CHÂN THÀNH CẢM ƠN !**

**BÁO CÁO ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Chúng tôi xin cam đoan đây là sản phẩm nghiên cứu của riêng chúng tôi và được sự hướng dẫn của GV. Hồ Thị Thanh Tuyến. Các nội dung nghiên cứu, kết quả trong đồ án này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác, đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung báo cáo của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm 2022*

*Tác giả*

*( ký tên và ghi rõ họ tên )*

Nguyễn Võ Hoàng Vũ

Nguyễn Trường Anh

Lạc Minh Long

Trần Quang Vinh

Lê Sỹ Tiến

**PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

**Phần xác nhận của giảng viên hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Tp. Hồ Chí Minh, ngày tháng năm 2022*

*( kí và ghi họ tên )*

**Phần đánh giá của giảng viên chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Tp. Hồ Chí Minh, ngày tháng năm 2022*

*( kí và ghi họ tên )*

**TÓM TẮT**

Với sự phát triển của Internet, việc trao đổi thông tin, chia sẻ dữ liệu ngày càng trở nên dễ dàng; có thể nói thông tin có thể được trao đổi ở bất kỳ đâu, bất kỳ lúc nào mà không gặp trở ngại về khoảng cách chỉ cần có kết nối Internet. Tuy nhiên, việc này cũng tiềm tàng nhiều nguy cơ và rủi ro từ việc bị đánh cắp thông tin khi truyền đi từ các bên thứ ba như hacker. Thời gian vừa qua, vấn đề an toàn dữ liệu cá nhân trở thành đề tài nóng, khi tại Việt Nam nói riêng và thế giới nói chung, liên tiếp xuất hiện các vụ việc dữ liệu cá nhân bị rò rỉ, rao bán trên các trang Web đen và bị tin tặc đánh cắp.

Vì vậy, việc bảo mật dữ liệu được xem như là một quy trình, một chính sách và là công nghệ để đảm bảo dữ liệu được bảo mật khỏi những truy cập bất ổn từ trong lẫn ngoài hoặc hư hỏng, thất thoát dữ liệu. Các công nghệ bảo mật tích hợp với các phương pháp mã hoá ra đời đã giúp ích rất nhiều cho việc bảo mật thông tin. Việc mã hoá dữ liệu là cần thiết để bảo vệ tốt thông tin người dùng, các ứng dụng hầu như đã hỗ trợ đầy đủ các phương thức mã hoá trong môi trường Internet hiện nay.

Bên cạnh đó, một số ít người dùng vẫn sử dụng các giao thức, ứng dụng thông thường – không được mã hoá – điều này tạo cơ hội cho nhiều đối tượng muốn chiếm đoạt thông tin, tài sản cá nhân thông qua việc rò rỉ thông tin.

Nắm bắt được những vấn đề trên và nhận thức được tầm quan trọng cũng như sức ảnh hưởng lớn trong việc bảo mật thông tin. Nhằm tăng tính đảm bảo các yêu cầu quy định trong việc bảo mật thông tin cá nhân, chống phần mềm độc hại thế hệ mới và cũng như phòng chống mất mát dữ liệu. Nhóm nghiên cứu chúng tôi đã lựa chọn và đưa ra đồ án nghiên cứu với đề tài : **“Các phương pháp mã hoá”**.

Với kinh nghiệm và năng lực của nhóm, chúng tôi mong có thể phát triển và sáng tạo một công cụ nhỏ trong việc mã hoá thông tin, mã hoá dữ liệu với những tính năng bảo mật cần thiết và đảm bảo tính an toàn cho người sử dụng. Từ những nghiên cứu về môi trường làm việc, yêu cầu cụ thể của các doanh nghiệp nói chung và người dùng nói riêng được đưa vào hệ thống. Phần mềm được xây dựng dựa trên nền tảng lập trình ổn định, mạnh mẽ, hỗ trợ bảo mật dữ liệu với các khoá riêng tư, tránh các cuộc tấn công độc hại và các mối đe doạ nội bộ.

**MỤC LỤC**

[LỜI CẢM ƠN 2](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590577)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN 4](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590578)

[TÓM TẮT 5](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590581)

[MỤC LỤC 6](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590580)

[DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT 7](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590581)

[DANH MỤC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 7](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590581)

[CHƯƠNG I – MỞ ĐẦU 8](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590577)

1.1 Giới thiệu [7](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590578)

1.2  [8](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590579)

[1.3 8](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590580)

1.4  [8](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590578)

1.5  [9](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590579)

[1.6 9](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590580)

1.7  [10](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590578)

1.8  [12](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590578)

1.9  [13](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590579)

[CHƯƠNG II – TEXT 15](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590581)

[2.10 15](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590581)

[TÀI LIỆU THAM KHẢO 18](file:///D:\University\Sophomore\Requirements%20Analysis%20And%20Design\BÁO%20CÁO%20GIỮA%20KỲ\BÁO%20CÁO%20GIỮA%20KỲ.docx#_Toc56590581)

**DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT**

Write…

**DANH MỤC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ**

Write…

**CHƯƠNG I – TỔNG QUAN VỀ PHƯƠNG PHÁP MÃ HOÁ**

**1.1 Sự cần thiết của mã hoá**

Hiện nay với sự phát triển không ngừng của công nghệ thông tin thì hầu hết tất cả các dịch vụ cũng như thông tin đang dần chuyển sang lưu trữ không chỉ trong nội bộ của một công ty nữa mà còn lưu trữ trên thuật toán đám mây, tạo nên sự thuận lợi rất lớn trong việc truyền nhận dữ liệu hơn rất nhiều. Đồng thời chúng ta cũng đang trong thời đại công nghiệp 4.0 với vạn vật đều có thể kết nối với nhau (Internet of Things – IoT), nên các thiết bị đều chia sẽ thông tin với nhau, tạo sự thuận lợi cho người dùng khi sử dụng nhiều thiết bị. Ví dụ : khi bạn dùng iphone và chuyển sang dùng thêm một ipad thì bạn chỉ cần đăng nhập tài khoản apple là dữ liệu sẽ tự động được đồng bộ ngay.

Tuy nhiên, việc lưu trữ các thông tin ở phạm vị ngoài máy tính cũng như nhiều thiết bị kết nối và chia sẻ thông tin như vậy sẽ khiến nguy cơ bị đánh cắp thông tin rất cao vì thông tin cần phải truyền qua nhiều trạm mới đến được nơi cần đến; do đó việc mã hóa thông tin, dữ liệu truyền đi hoặc khi lưu trữ là rất quan trọng.

Do đó, mã hóa sẽ có các vai trò sau :

+ Đảm bảo tính riêng tư : Mã hóa sẽ giúp thông tin, dữ liệu của mỗi cá nhân sẽ không thể bị đọc bởi bất kỳ ai mà không có sự ủy quyền hoặc cho phép. Giúp ngăn chặn các hacker mạng.

+ Bảo vệ : Giúp dữ liệu không bị rò rĩ khi trong quá trình truyền đi hoặc nghỉ (lưu trữ). Như khi mất ổ cứng nhưng ổ cứng đã được mã hóa thì có thể đảm bảo dữ liệu vẫn sẽ được bảo mật và thông tin trong ổ cứng đó sẽ không bị rò rĩ. Vì vậy, với truyền thông thì mã hóa sẽ giúp bảo về người dùng khi chia sẽ các thông tin nhạy cảm thông qua các mạng truyền thông; giúp ngăn chặn các cuộc tấn công như: main-in-the-middle.

+ Xác thực :Mã hóa khóa công khai thiết lập rằng máy chủ gốc của trang web sở hữu khóa riêng do đó sẽ được cấp chứng chỉ SSL hợp pháp. Từ đó đảm bảo các thông tin truyền qua lại giữa phía Client và Server sẽ luôn được mã hóa và giảm khả năng bị đánh cắp thông tin.

+ Quy định : Nhiều quy định của ngành và chính phủ quy định các công ty phải xử lý mã hóa dữ liệu người dùng, chuyển về dạng mã hóa đảm bảo tính an toàn thông tin. Ví dụ như quy định : HIPAA, PCI-DSS và GDPR.

**1.2 Lịch sử hình thành của mã hoá**

Mật mã đã xuất hiện từ rất lâu, từ thời cổ đại và tiếp tục phát triển cho đến ngày nay, với mục đích vẫn như ban đầu đó là giữ bí mật về thông tin.

***1.2.1 Mật mã Spartan thời cổ đại***

Từ năm 600 trước công nguyên, người Spartan dùng các thiết bị gọi là *Scytale* để gửi đi các thông điệp bí mật.



*Hình 2 – Thiết bị Scytale*

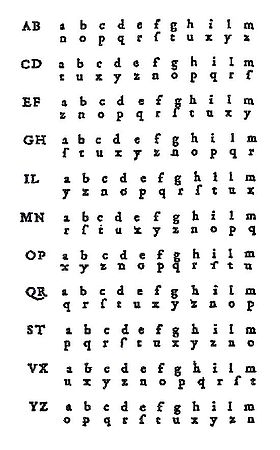
***1.2.2 Mã hoá Roman và Cyphers (bộ mã hoá)***

Từ năm 60 trước công nguyên, vị tướng Julius Caesar đã tạo ra một dạng mật mã bằng cách tăng vị trí của các ký tử lên ba đơn vị theo bảng chữ cái (như A sẽ thành D, B sẽ thành E, …).



*Hình 3 – Mật mã của Julius Caesar*

Năm 1553 – Giovan Battista Bellaso đã tạo ra một bộ mã hóa (cypher) đầu tiên bằng việc sử dụng một khóa mã hóa (key encryption) và khóa đó sẽ được người nhận sử dụng để đọc nội dung được gửi.

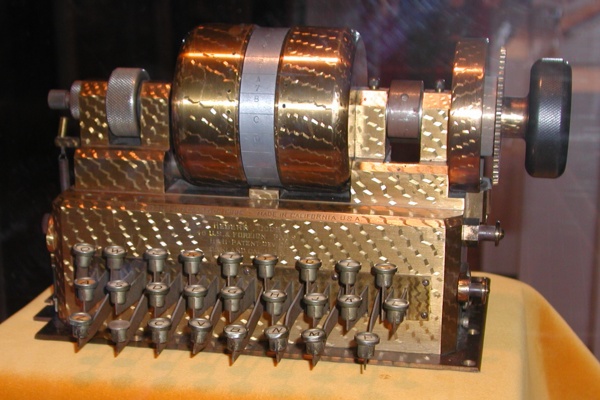


*Hình 4 – Đối chiếu Cypher của Giovan Battista Bellaso*

Năm 1854 – Charles Wheatstone phát minh ra Playfair Cipher, là một kiểu mã hóa cao hơn bằng việc mã hóa lần lượt một cặp ký tự thay vì từng cái một, do đó khó để hiểu hơn.

***1.2.3 Máy Rotor Hebern***

Năm 1917 – Edward Hebern (người Mỹ) đã tạo ra một máy cơ điện và khóa sẽ được nhúng trong một đĩa quay; là một nguyên mẫu đầu tiên của máy rotor. Nó mã hóa một bảng thay thế khi một ký tự mới được nhập.



*Hình 5 – Máy cơ điện của Edward Hebern*

Năm 1918 – Một kỹ sư người Đức là Arthur Scherbius đã tạo ra máy Enigma, thay vì sử dụng các rotor như Edward Hebern thì chiếc máy này sử dụng các cảnh quạt để tạo mã hóa.



*Hình 6 – Máy Enigma*

***1.2.4 Mật mã trong Thế chiến thứ 2 (WW2)***

Năm 1932–Nhà mật mã người Ba Lan có tên Marian Rejewski đã tìm ra cách hoạt động của máy Enigma. Năm 1939, Ba Lan chia sẻ thông tin cho phe Đồng minh và điều này giúp cho Alan Turing bẻ khóa được mật mã này, góp phần chiến thắng về sau cho phe Hiệp ước.

Năm 1945 –Claude E.Shannon thuộc Bell Labs đã xuất bản một bài viết với tên là “A mathematical theory of cryptography” – Dịch: Lý thuyết toán học về mật mã. Đánh dấu điểm khởi đầu cho mã hóa hiện nay.

***1.2.5 Mật mã hiện đại***

Với những thế kỷ trước thì việc mã hóa được các chính quyền sử dụng để bảo mật thông tin như trong: chiến tranh, ngoại giao hoặc gián điệp. Còn ngày nay, việc mã hóa thông tin đã lan rộng hơn và mang nhiều hơn về yếu tố cá nhân.

Đầu những năm 1970 – IBM đã thành lập một nhóm “crypto group” – Dịch : Nhóm tiền điện tử; có trách nhiệm thiết kế một block cypher để bảo mật dữ liệu của khách hàng. Và vào nằm 1973, Mỹ đã chấp nhận nó như một tiêu chuẩn quốc gia, gọi là Tiêu chuẩn mã hóa dữ liêu hay DES (Data Encryption Standard). Và được sử dụng cho tới năm 1997.

Năm 2000 – Một tiêu chuẩn mã hóa nâng cao đã xuất hiện, thay thế cho DES, được biết đến là AES (Asymmetric Key – Khóa không đối xứng : tức người gửi và người nhận có cùng một khóa bí mật). Ngày nay AES được sử dụng rất rộng rãi.

Năm 2005 –Ra mắt Elliptic-curve cryptography (ECC) – Mật mã đường công Elliptic, là một sơ đồ mật mã khóa công khai tiên tiến cho phép các khóa mã hóa ngắn hơn.

**1.3 Khái niệm mã hoá**

Theo Webster’s Revised Unabridged Dictionary: Cryptography is “the act or art of writing secret characters” – Dịch: Mã hóa là một hành động hoặc một nghệ thuật viết các ký tự bí mật. Hoặc theo như Free Online Dictionary of Computing: Cryptography is “encoding data so that it can only be decoded by specific individuals” – Dịch: Mã hóa là việc mã hóa dữ liệu để nó chỉ có thể được giải mã bởi một số người đặc biệt.

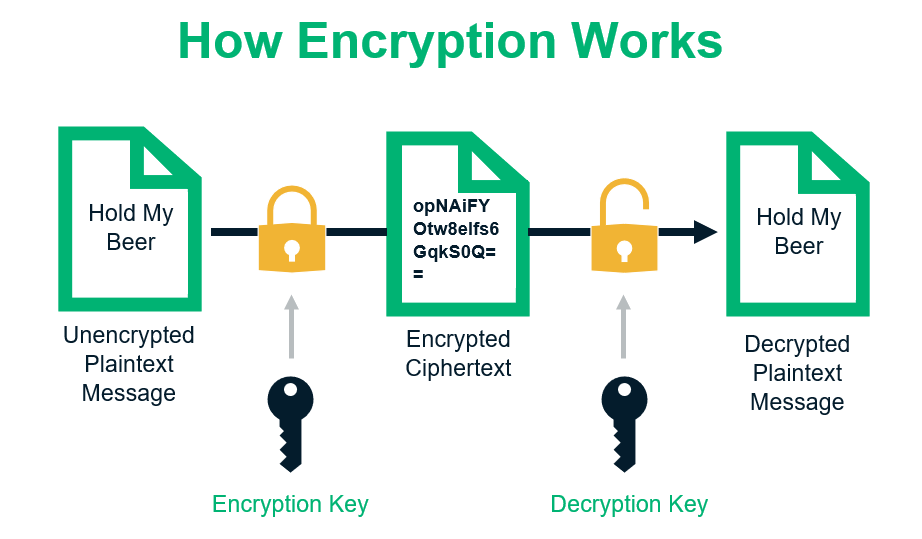
*Mã hóa (encryption)* là một cách xáo trộn dữ liệu nhằm đảm bảo chỉ những bên được ủy quyền mới có thể hiểu được thông tin của dữ liệu đó. Về mặt kỹ thuật, đó là một quá trình chuyển đổi dữ liệu từ bản gốc mà con người có thể hiểu sang bản mã hóa mà con người không thể hiểu. Hiểu một cách đơn giản thì mã hóa sẽ biến đổi dữ liệu ban đầu thành một dạng dữ liệu mới, và để đọc được thì phải tiến hành giải mã về dạng ban đầu thông qua *crytographic key (chìa khóa mật mã)*.

*Khoá mật mã (crytographic key)* là một chuỗi các ký tự được sử dụng trong thuật toán mã hoá để thay đổi dữ liệu sao cho nó xuất hiện ngẫu nhiên. Giống như một khoá vậy lý, nó khoá (mã hoá) dữ liệu để chỉ người nào có khoá chính mới có thể mở khoá (giải mã) nó.

Một hệ mã hoá sẽ gồm hai khâu :

+ *Mã hóa (encrytion)* : khi dữ liệu đang lưu trữ hoặc đang chuyển đi.

+ *Giải mã (decryption)* : khi dữ liệu được chuyển đến nơi được ủy quyền và tiến hành đọc.



*Hình 1 – Mô tả quá trình của một hệ mã hoá*

**1.4 Các thuật ngữ phổ biến trong mã hoá**

Trong mã hóa, để thuận tiện cho việc mô tả thông tin, người ta đã quy định một số thuật ngữ chuyên ngành dùng để mô tả các thành phần cũng như các quy trình trong mã hóa, sau đâu là một số thuật ngữ cơ bản được dùng nhiều nhất :

+ *PlainText :* dùng để chỉ phần nội dung gốc (có thể đọc), khi chưa mã hóa.

+ *CipherText :* dùng để chỉ nội dung đã được mã hóa thông qua một giải thuật mã hóa nào đó.

+ *Encryption :* dùng để chỉ quy trình áp dụng một giải thuật mã hóa vào một nội dung có thể đọc (*PlainText*) và sau đó biến đổi nội dung đó thành dạng không thể đọc (*CipherText*).

+ *Decryption :* trái ngược lại với *Encryption*, thì thuật ngữ *Decryption* dùng để chỉ quy trình chuyển đổi nội dung không thể đọc (*CypherText)* về dạng nôi dụng có thể đọc (*PlainText)*.

+ *Keys :* dùng để chỉ chìa khóa dùng trong việc mã hóa cũng như giải mã.

+ *Hash :* dùng để chỉ dữ liệu của bạn sẽ được mã hóa thành một dạng không thể đọc, tuy nhiên khác với *Encryption* khi dữ liệu có thể được *Decryption* thì *Hash* sẽ không thể chuyển về dạng có thể đọc được nữa. Do đó, *Hash* thường được dùng cho mục đích xác thực (như mật khẩu,…).

+ *Salt :* dùng để chỉ một bước trung gian trước khi mật khẩu của bạn được *Hash*, lúc này chương trình sẽ thêm một bước bảo mật bổ sung bằng cách thêm một đoạn dữ liệu vào mật khẩu nhằm tăng tính bảo mật

(Ví dụ : Bạn nhập mật khẩu là 123 nhưng sau đó mật khẩu sẽ thật sự sẽ là 123addSalt).

+ *Symmetric and Asymmetric Algoriths :* dùng để chỉ hai thuật toán mã hóa chính đang được dùng phổ biến hiện nay, trong đó :

* Symmetric Algorith : là thuật toán đối xứng khi dùng cùng một *keys* cho cả giai đoạn *Encryption* và *Decryption.*
* Asymmetric Algorith : là thuật toán bất đối xứng khi dùng *public keys* ở giai đoạn *Encryption* và *private keys* *Decrytion.*

**+** *Public and Private Keys :* dùng để chỉ các khóa được dùng trong thuật toán mã hóa bất đối xứng, được một người dùng tạo ra, trong đó *Public Key* là khóa có thể dùng cho chia sẽ với nhiều người (dùng trong việc *Encryption* các thông tin gửi cho người tạo*)*, trong khi đó *Private Key* là khóa mà chỉ có người tạo sở hửu (dùng để *Decryption* các thông tin nhận được).

**+** *HTTPS (Hypertext Tranfer Protocal) :* đây là giao thức mở rộng của HTTP, dùng để bảo mật dữ liệu truyền tải qua mang được an toàn hơn khi các dữ liệu đó đều sẽ được mã hóa.

**+** *End-to-End Encryption (E2EE) :* dùng để nói chỉ một phương thức mã hóa mà ở đây chỉ có người gửi tin và người nhận tin mới có thể hiểu được thông điệp mã hóa, và sẽ không ai khác có thế biết được kể cả nhà cung cấp mạng.

**+** *Backdoor :* dùng để chỉ các cách thức để qua mặt một *Encryption,* thường xuất hiện do một lỗ hổng trong thuật toán hoặc do ai đó cố tình để lổ hỏng này, và thông qua lỗ hổng đó thì người khác có thể dễ dàng truy cập được thông tin được mã hóa mà không cần biết khóa.

**1.5 Cấu trúc chung của một bộ mã hoá**

Một bộ mã hóa sẽ bao gồm các thành phần chủ yếu sau :

+ PlainText : thông tin trước khi được mã hóa.

+ CipherText : thông tin sau khi được mã hóa.

+ Key : chìa khóa, dùng trong mã hóa và giải mã.

+ Encryption/Decryption : phương pháp mã hóa và giải mã.

Trong đó gồm hai quá trình chính:

+ Mã hóa: Quá trình này được thực hiện khi áp dụng một hàm toán học Encryption lên PlainText để biến thành dạng CipherText.

+ Giải mã: Quá trình này được thực hiện khi áp dụng một hàm toán học Decryption lên CipherText để chuyển về dạng PlainText.

**1.6 Thuật toán của mã hoá**

Thuật toán mã hoá là một thuật toán nhằm mã hoá thông tin của chúng ta, biến đổi thông tin từ dạng rõ sang dạng mờ, để ngăn cản việc đọc trộm nội dung của thông tin – dù *Hacker* có được thông tin đó cũng không hiểu nội dung chứa trong nó là gì.

Thông thường các thuật toán sử dụng một hoặc nhiều *Key* (một chuỗi chìa khoá để mã hoá và giải mã thông tin) để mã hoá và giải mã (ngoài trừ những thuật toán cổ điển). Có thể coi *Key* này như một cái *password* để có thể đọc được nội dung mã hoá. Người gửi sẽ dùng *Key* mã hoá để mã hoá thông tin sang dạng mờ và người nhận sẽ sử dụng *Key* giải mã để giải mã thông tin sang dạng rõ. Chỉ những người nào có *Key* giải mã mới có thể đọc được nội dung.

Nhưng đôi khi *Hacker* không có *key* giải mã vẫn có thể đọc được thông tin, bằng cách phá vỡ thuật toán. Có một nguyên tắc là bất kỳ thuật toán mã hoá nào cũng đều có thể bị phá vỡ, do đó không có bất kỳ thuật toán mã hoá được coi là an toàn mãi mãi. Độ an toàn của thuật toán được dựa vào nguyên tắc :

+ Nếu chi phí để giải mã một khối lượng thông tin lớn hơn giá trị của khối lượng thông tin đó thì thuật toán đó được tạm coi là an toàn (không ai lại bỏ ra 50 năm để giải mã một thông tin mà chỉ mang lại 1000 đô lợi nhuận).

+ Nếu thời gian để phá vỡ một thuận toán là quá lớn (giả sử lớn hơn 100 năm) thì thuật toán được tạm coi là an toàn.

Năm thuật toán mã hoá thông dụng nhất được sử dụng hiện nay :

+ *DES – Data Encryption Standard*

*+ TripleDES – 3DES hoặc TDES*

*+ RSA – Ron Rivest, Adi Shamir và Leonard Adleman*

*+ AES – Advanced Encryption Standard*

*+ Twofish*

**1.7 Phân biệt giữa Encode và Encryption**

*Encode* là biến đổi dữ liệu từ định dạng này sang định dạng khác dựa trên một quy luật nào đó để có thể sử dụng dữ liệu trên các hệ thống. Có thể thấy mã ASCII là ví dụ điển hình cho việc *encode,* ví dụ : A 🡪 01000001.

Còn *Encryption* là phương pháp biến đổi dữ liệu kèm theo việc bảo mật dữ liệu. *Encryption* sử dụng thuật toán mã hoá để có thể mã hoá dữ liệu và kèm theo đó là khoá *Key* để có thể giải mã. Việc làm này tránh cho bên thứ ba có thể đánh cắp thông tin.

Có thể thấy rằng, cả hai cách trên đều biến đổi dữ liệu từ dạng này sang dạng khác nhưng mục đích của hai cách là khác nhau. *Encode* được sử dụng để tăng thêm tính khả dụng của dữ liệu đối với hệ thống khác và dữ liệu có thể dễ dàng giải mã với thuật toán công khai. Nhưng với *Encryption* thực hiện mã hoá dữ liệu nhằm bảo vệ dữ liệu cần có khoá để có thể giải mã dữ liệu giúp giữ được tính bảo mật đối với các bên thứ ba.

**CHƯƠNG II – CÁC PHƯƠNG PHÁP MÃ HOÁ**

**2.1 Mã hoá cổ điển**

***2.1.1 Tổng quan về mã hoá***

Trong mật mã học, mã hoá cổ điển là một dạng của mật mã học đã được sử dụng trong lịch sử phát triển của loài người nhưng ngày nay đã trở nên lạc hậu do các phương thức mã hóa này quá đơn giản và những kẻ tấn công có thể dễ dàng bẻ khóa thông qua nhiều phương thức như tấn công vét cạn (ví dụ như dùng máy tính thử hết mọi trường hợp) hay dựa trên tấn công thống kê (dựa trên tần suất xuất hiện của các chữ cái).

Nói chung, mật mã học cổ điển hoạt động trên cơ sở bảng chữ cái (chẳng hạn các ký tự từ "A" tới "Z" trong tiếng Anh), và chúng được thực hiện bằng tay hay một số máy móc cơ khí đơn giản. Ngược lại, các mô hình mã hóa hiện đại sử dụng các máy tính hay các công nghệ số hóa khác, và hoạt động mã hóa dựa trên việc thay thế các bit hay byte. Các phương thức mã hóa cổ điển thông thường dễ bị tổn thương (phá mã) bởi các tấn công văn bản mã hóa, đôi khi thậm chí kẻ tấn công không cần biết các chi tiết cụ thể của hệ thống mã hóa, bằng cách sử dụng các công cụ như phân tích tần suất. Đôi khi người ta cũng cho rằng các phương thức mã hóa như cách thức mã hóa của cỗ máy Enigma thuộc về các phương thức mã hóa cổ điển mặc dù cách thức mã hóa này đã sử dụng các thiết bị và công nghệ hiện đại nhất vào thời điểm đó (trong thời kỳ của Thế chiến II).

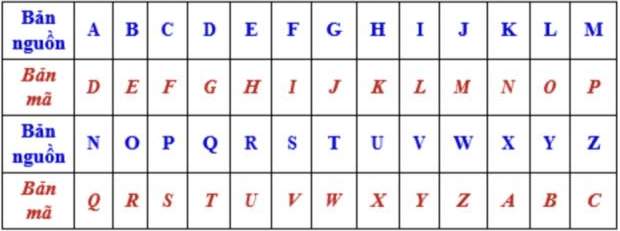
Các phương thức mã hóa cổ điển chủ yếu dựa trên mật mã hóa hoán vị và mật mã hóa thay thế. Trong mật mã hóa thay thế, các ký tự (hoặc nhóm ký tự) được thay thế một cách có quy luật trong toàn bộ thông điệp bằng các ký tự khác (hoặc nhóm ký tự), chẳng hạn câu I am Mr. Enigma from được thay bằng câu This is morning star, sau đó các ký tự còn lại trong bảng chữ cái được thay thế theo một quy luật nào đó xác định trước. Trong phương thức mật mã hóa hoán vị thì các ký tự được giữ không đổi, nhưng trật tự của chúng trong bản tin lại thay đổi theo một quy luật nào đó. Có các thuật toán phức tạp để thực hiện việc mật mã hóa bằng cách tổ hợp hai phương thức trên để tạo ra sản phẩm mã hóa; các phương thức mã hóa khối hiện đại như DES hay AES thực hiện việc lặp đi lặp lại một số bước thay thế và hoán vị.

***2.1.2 Hệ mã CAESAR***

Hơn 2.000 năm trước đây, hoàng đế La Mã Julius Caesar (100 năm trước CN) đã đề xuất một thuật toán mang tên ông với ý tưởng như sau: Giả sử ta có bảng ký tự 26 chữ cái A, B, C, …Z được xếp thành một vòng tròn. Mỗi chữ cái trong bản nguồn sẽ được thay thế bởi chữ cái thứ ba sau nó để có bản mã.

Ví dụ : Thay mỗi chữ cái bởi chữ thứ ba sau nó trong bảng chữ cái: MOI NGAY TOI CHON MOT NIEM VUI (plaintext) -> PRLQ JDB WRL FKRQ PRW QLHP YXL (cipher text)

Việc thay thế được thực hiện dựa trên bảng sau :

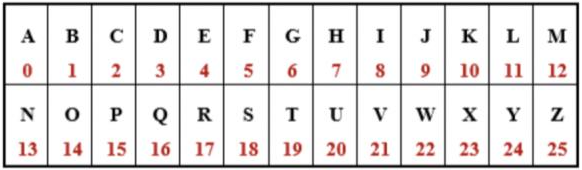


*Hình – Ví dụ về mã hoá thay thế trong mã hoá cổ điển*

Quá trình mã hoá sẽ đối chiếu từ các ký tự của bản nguồn đến các ký tự của bản mã. Quá trình giải mã sẽ được thực hiện ngược lại.

Nếu việc thay thế được thực hiện không chỉ bởi ký tự thứ 3 sau nó mà là ở một khoảng cách k nào đó khác thì ta sẽ có thuật toán Caesar mở rộng.

Trong trường hợp này, giá trị k sẽ là khoá và phải được giữ bí mật. Vì bảng chữ cái chỉ có 26 chữ cái nên giá trị k chỉ có thể từ 0 đến 25 (0 là trường hợp tầm thường). Để việc mã hoá có thể lập trình được, ta sẽ gán các ký tự A, B, C,… bởi các con số 0, 1, 2,…như bảng dưới đây :



*Hình – Bảng mã hoá trong mã hoá cổ điển*

Khi đó, việc mã hoá và giải mã được biểu diễn dưới dạng công thức toán học như sau :

+ Ta có : P = C = K = Z26;

+ Mã hóa : ek(x) = (x + k) mod 26

+ Giải mã : dk(y) = (y – k) mod 26

=> Với k = 3 thì đó chính là thuật toán Caesar.

Trong nhiều trường hợp, người ta không sử dụng số k như là khoá mà người ta sử dụng ký tự mã ứng với ký tự nguồn A là khoá. Vậy trong thuật toán Caesar, khoá sẽ là chữ D.

***2.1.3 Mã hoá thay thế đơn***

Trong thuật toán Caesar mở rộng, tất cả các ký tự đều dịch chuyển một khoảng cách như nhau là k. Mở rộng thuật toán này, ta sẽ cho các ký tự có thể di chuyển với những khoảng cách khác nhau. Đó chính là kỹ thuật thay thế đơn từ.

Trong trường hợp này, khoá mã hoá sẽ không còn là một số nữa mà sẽ là một bộ 26 số từ 0 đến 25, hay, nói khác đi, là một hoán vị của dãy các ký tự A đến Z.

*Ví dụ* : một khoá mã hoá của thuật toán thay thế đơn từ DKVQFIBJWPESCXHTMYAUOLRGZN. Dựa vào khoá này ta có thể biết AàD, BàK,…

*Ví dụ* : chuỗi ký tự THUAT TOAN THAY THE DON TU sẽ được mã hoá với khoá trên thành: UJODU UHDX UJDZ UJF QHX UO.

Như vậy, mỗi khoá của thuật toán thay thế đơn từ là một hoán vị của chuỗi ký tự A đến Z. Khoá dài như vậy nên không dễ dàng nhớ được nó. Để có thể nhớ được một khoá dài thì người ta thoả thuận một phương pháp tạo khoá từ một “từ khoá” (key word) ngắn.

Ví dụ, một khoá thay thế đơn từ có thể được tạo từ một từ khoá Julius Caesar như sau :

+ Viết các ký tự của từ khoá liền nhau : JULIUSCAESAR

+ Tính từ trái qua phải, loại đi các ký tự trùng lắp : JULISCAER

+ Viết tiếp các ký tự của bảng chữ cái theo thứ tự từ ký tự cuối, ký tự nào đã có rồi thì bỏ qua : JULISCAERTVWXYZBDFGHKMNOPQ

+ Như vậy là ta đã có một hoán vị 26 ký tự có thể dùng làm khoá

Trên thực tế, người ta có thể sử dụng các phương pháp khác nhau để biến đổi một từ khoá dễ nhớ thành một khoá mã hoá khó nhớ.

Về cơ bản, các biến đổi này đều có những bước như loại bỏ các ký tự trùng trong từ khoá và thêm các ký tự khác cho đủ 26 ký tự. Tuy nhiên, cách thức thực hiện có thể khác nhau.

Quy trình biến đổi từ khoá thành khoá cần phải được thống nhất giữa người gửi và người nhận. Trong các thuật toán mã hoá hiện đại, quy trình biến đổi từ khoá ban đầu thành các khoá thực sự dùng để mã hoá thường được công bố công khai cùng với thuật toán mã hoá.

***2.1.4 Mã hoá hoán vị***

Về bản chất thì kỹ thuật hoán vị chỗ chính là trường hợp đặc biệt của kỹ thuật thay thế. Trong kỹ thuật này, tập hợp các ký tự của bản nguồn sẽ không thay đổi so với bản mã mà chỉ thay đổi vị trí của các ký tự. Có một số kỹ thuật đổi chỗ đơn giản như sau :

+ *Đảo ngược từ (Mirror cipher)* : Các ký tự trong bản mã được viết theo ngược lại so với bản nguồn : TOI AN COM à MOC NA IOT

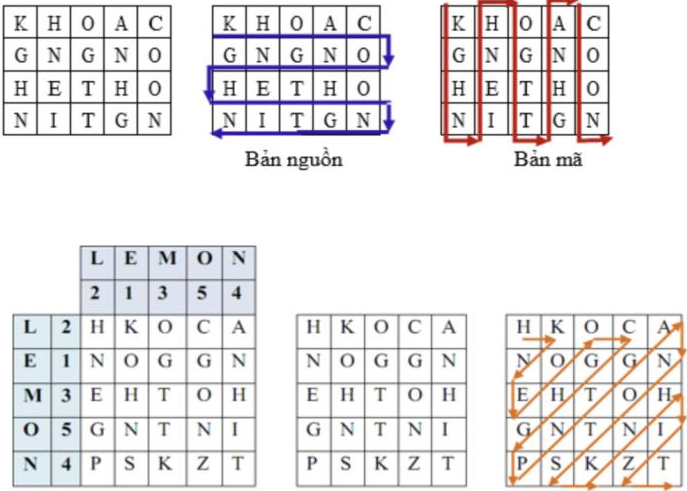
+ *Hình học (Geometric Figure)* : Viết bản nguồn theo một mẫu và đọc theo mẫu khác.

+ *Đổi chỗ theo hàng (Row Transposition ciphers)* : Viết bản nguồn theo hàng, hoán vị các cột theo khóa và sau đó đọc lại theo hàng để có được bản mã.

+ *Đổi chỗ lộn xộn (Nihilist cipher)*: Đổi chỗ dòng và cột. Viết thông điệp theo hàng, theo khóa. Để có bảng mã, ta đọc từ trái sang phải theo từng hàng, thứ tự hàng được xác định bằng khóa viết theo cột.

+ *Đổi chỗ đường chéo (Diagonal cipher)* : Viết thông điệp giống như trên và thông điệp theo đường zig-zag để có bản mã.

Khoá của thuật toán đổi chỗ theo hàng chính là số phần tử của khoá và hoán vị của các phần tử đó.



*Hình – Quy trình mã hoá hoán vị trong mã hoá cổ điển*

Thám mã bắt đầu từ việc dự đoán số phần tử đó (chính là số cột của bảng). Để làm điều đó, ta tìm kiếm tất cả các khả năng đổi chỗ trong chu kỳ dự kiến để tìm ra mẫu chung (sử dụng danh sách các cặp, bộ ba,…có nghĩa).

Nếu các ký tự được sắp xếp lại trong một nhóm thì ta thử xem xét việc sắp xếp tương tự trong các nhóm khác. Khi đã tìm được các cụm từ có nghĩa, chúng ta sẽ đồng thời tìm được thứ tự đảo của khoá và sẽ suy ra khoá.

***2.1.5 Mã hoá kết hợp***

Trong thực tế người sử dụng có thể kết hợp cả hai kỹ thuật thay thế và đổi chỗ vào trong cùng một thuật toán.

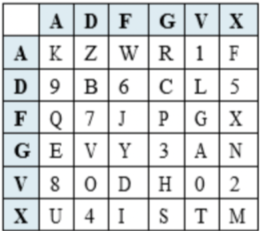
Ví dụ dưới đây cho thấy việc kết hợp đó đã tạo ra một thuật toán mã hoá hiệu quả được quân đội Đức sử dụng trong đại chiến thế giới lần thứ nhất. Đó là thuật toán ADFGVX (năm 1918).

Thuật toán được mô tả như sau :

+ Chỉ có sáu chữ cái này được sử dụng để mã hoá dữ liệu và sau đó dữ liệu được truyền đi bằng tín hiệu Morse. Mã MORSE của các chữ này phân biệt rất rõ.

+ 36 ký tự (gồm 26 chữ cái và 10 chữ số) được xếp vào một bảng 6\*6 với tên hàng và cột là các ký tự ADFGVX. Mỗi chữ cái trong bản nguồn được thay bởi một cặp hai chữ cái là chỉ số hàng – cột.

+ Sử dụng kỹ thuật đổi chỗ cùng với khóa để tách và đổi chỗ các chữ cái. Ví dụ ta cần mã hoá bản nguồn: GIONOSUNGLA530

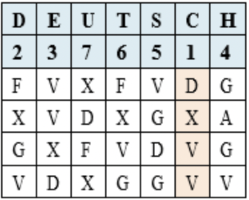


*Hình – Ví dụ mã hoá kết hợp trong mã hoá cổ điển*

Theo bảng trên, mỗi ký tự được mã hoá bởi 2 ký tự khác, thành : FV XF VD GX VD XG XA GX FV DV GV DX GG VV.

Chuỗi ký tự trung gian này sẽ được mã hoá bằng kỹ thuật đổi chỗ với từ khoá: DEUTSCH (2376514).

Chuỗi này sẽ được viết vào bảng theo thứ tự từ trái qua phải, từ trên xuống dưới :



*Hình - Ví dụ mã hoá kết hợp trong mã hoá cổ điển*

Đọc bảng theo cột và khoá ta sẽ nhận được chuỗi mã cuối cùng là: DXVV FXGV VVXD GAGV VGDG FXVG XDFX.

Thuật toán này thực tế không an toàn, nó đã bị thám mã bởi quân đội Pháp và đã bị bẻ khoá ngay trong năm 1918.

***2.1.6 Kỹ thuật thay thế đa từ***

Trong kỹ thuật thay thế đơn từ, mỗi ký tự nguồn đều biến thành một ký tự xác định dựa vào khoá, không phụ thuộc vào vị trí của ký tự đó trong bản nguồn.

Ví dụ, nếu ký tự A đã xác định là sẽ biến đổi thành D thì tất cả các ký tự A trong bản nguồn đều biến thành D trong bản mã.

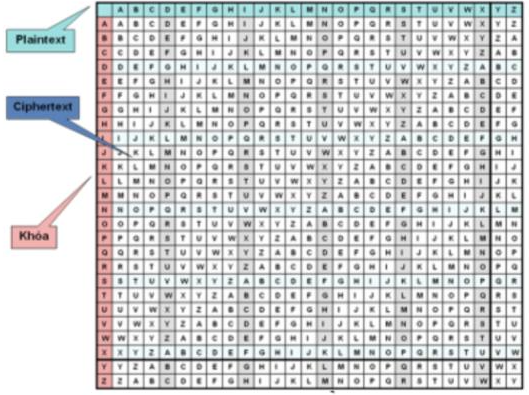
Kỹ thuật thay thế đa từ cho phép, ví dụ, các ký tự A trong bản nguồn có thể biến thành các ký tự khác nhau trong bản mã. Ta sẽ xem xét một đại diện của kỹ thuật thay thế đa từ với thuật toán do nhà toán học người Pháp Blaise de Vigenere (1523–1596) đề xuất. Ví dụ, thuật toán có thể được trình bày như sau :

+ Từ khoá CIPHER được xếp kề nhau : CIPHERCIPHERCIPHER

+ Sử dụng chuỗi đó như là tập hợp các khoá để mã hoá các ký tự bản nguồn. Ví dụ bản nguồn là THUAT TOAN THAY THE DA TU thì ký tự T sẽ mã hoá bằng khoá C (dùng mã Caesar mở rộng) thành V, ký tự H với khoá I thành P, …

+ Cuối cùng ta nhận được bản mã: VPJHX KQIC ALRA BWL HR VC Để có thể xác định nhanh kết quả của mật mã Caesar mở rộng.

Vigenere sử dụng bảng sau để tra cứu :



*Hình – Bảng tra cứu kỹ thuật thay thế đa từ trong mã hoá cổ điển*

Tuy nhiên, hiện nay ta không cần tra cứu thủ công như ngày xưa mà có thể viết một đoạn mã ngắn để thực hiện thao tác này.

Khoá của thuật toán mã hoá đa từ có độ dài bằng độ dài của bản nguồn. Tuy nhiên, việc xây dựng khoá (key) từ từ khoá (keyword) như trên có điểm yếu do việc lặp lại một cách đơn giản từ khoá. Trên thực tế, chúng ta có thể dùng một số phương pháp khác nhau để xây dựng khoá.

Vigenere đề xuất sử dụng phương pháp tạo khoá tự động (Autokey) bằng cách sử dụng keyword ghép vào phần đầu của bản nguồn và sử dụng như là khóa để mã hoá cho bản nguồn.

Ví dụ có thể dùng khoá CIPHERTHUATTOANTHAYTHEDATU để mã hoá bản nguồn THUAT TOAN THAY THE DA TU.

Như vậy mã hoá T dùng khoá C sẽ có kết quả là V, H dùng khoá I sẽ có kết quả là P,…Ngoài ra, có thể tạo khóa bằng cách lấy các ký tự bắt đầu từ một vị trí (trang, dòng) nào đó trong một cuốn sách xác định trước. Người ta gọi đó là mã hoá bằng sách (Book cipher).

**2.2 Mã hoá một chiều (HASH)**

***2.2.1 Tổng quan về mã hoá***

Hashing là quá trình biến một dữ liệu đầu vào có độ dài bất kỳ thành một chuỗi đầu ra đặc trưng có độ dài cố định. Hashing được thực hiện thông qua hàm băm (hash function).

Một cách tổng quát hàm băm là bất kỳ hàm nào có thể được sử dụng để ánh xạ dữ liệu có kích thước tùy ý thành các giá trị kích thước cố định. Các giá trị được trả về bởi hàm băm được gọi là giá trị băm, mã băm, thông điệp băm, hoặc đơn giản là “hash”.



*Hình - Ảnh minh hoạ về mã hoá một chiều*

Trong blockchain, các giao dịch có độ dài khác nhau sẽ được băm thông qua một thuật toán băm nhất định và tất cả đều cho đầu ra có độ dài cố định bất kể độ dài của giao dịch đầu vào là bao nhiêu. “Chẳng hạn, Bitcoin sử dụng thuật toán SHA-256 để băm các giao dịch cho kết quả đầu ra có độ dài cố định là 256 bit (32 byte) cho dù giao dịch chỉ là một từ hoặc giao dịch phức tạp với lượng dữ liệu khổng lồ. Điều làm cho việc theo dõi các giao dịch trở nên dễ dàng hơn khi truy xuất và theo dõi lại các giá trị băm. Kích thước của hàm băm sẽ phụ thuộc vào hàm băm được sử dụng.”

Kỹ thuật hashing thường được sử dụng và có ứng dụng rộng rãi trong việc đảm bảo tính toàn vẹn cho dữ liệu, trong blockchain là các hàm băm mật mã (cryptographic hash function) do có một số tính chất quan trọng phù hợp cho việc đảm bảo an toàn dữ liệu.

***2.2.2 Hàm băm***

Hàm băm mật mã là các hàm băm phù hợp để sử dụng trong mật mã. Cũng giống như các hàm băm thông thường, nó là một thuật toán toán học ánh xạ dữ liệu có kích thước tùy ý thành một chuỗi bit có kích thước cố định. Ngoài ra nó còn đảm bảo tính chất là một hàm một chiều, nghĩa là, một hàm mà trên thực tế không thể có ngược. Nếu có một giá trị băm đầu ra, ta sẽ không thể suy ngược lại được giá trị đầu vào là gì để có thể băm ra một thông điệp băm như vậy, hoặc ít nhất là rất khó suy luận được ra, trừ khi bạn vét cạn hết toàn bộ các khả năng có thể của thông điệp đầu vào. Đây là tính chất vô cùng quan trọng của hàm băm mật mã biến nó thành một công cụ cơ bản của mật mã hiện đại.



*Hình – Ảnh minh hoạ về hàm băm*

Các hàm băm mật mã có nhiều ứng dụng trong an toàn thông tin. Nó được sử dụng nhiều trong chữ ký số, mã xác thực thông điệp (MAC) và các hình thức xác thực khác. Ngoài ra, chúng cũng có thể được sử dụng như các hàm băm thông thường, để lập chỉ mục dữ liệu trong bảng băm, lấy đặc trưng của dữ liệu, phát hiện dữ liệu trùng lặp hoặc làm tổng kiểm tra để phát hiện lỗi các dữ liệu ngẫu nhiên.

Tính chất của hàm băm mật mã :

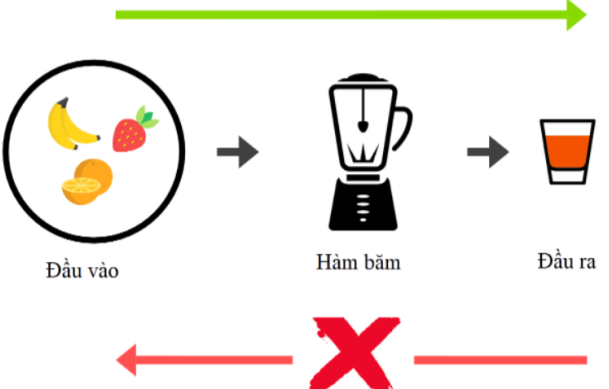
+ *Tính tất toán :* 1 thông điệp đầu vào luôn tạo ra cùng 1 hàm băm

+ *Tính hiệu quả :* Có khả năng tính toán nhanh chóng giá trị băm của bất kỳ thông điệp nào

+ *Tính nhạy cảm :* Đảm bảo rằng bất kỳ một thay đổi nào, dù là nhỏ nhất trên dữ liệu đều sẽ gây ra sự thay đổi cực lớn trên giá trị băm và tạo ra giá trị băm hoàn toàn khác, và không hề có liên hệ gì với giá trị băm cũ (hiệu ứng tuyết lở - Avalanche )

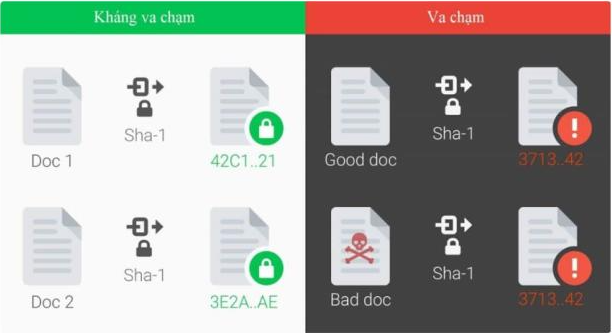
Ngoài ra, với mục đích đảm bảo an toàn cho dữ liệu, các hàm băm mật mã phải có khả năng chịu được tất cả các loại tấn công mã hóa đã biết. Trong lý thuyết mật mã, mức độ an toàn của hàm băm mật mã đã được xác định bằng các thuộc tính sau :

+ *Tính kháng tiền :* khái niệm này có liên quan đến tính chất một chiều của hàm băm.



*Hình – Ảnh minh hoạ về tính kháng tiền trong hàm băm*

+ *Tính kháng va chạm :* Rất khó để tìm thấy hai thông điệp khác nhau m1 và m2 sao cho hash (m1) = hash (m2). Một giá trị như vậy được gọi là va chạm của hàm băm mật mã.



*Hình – Ảnh minh hoạ về tính kháng va chạm trong hàm băm*

Tính kháng va chạm bao hàm cả tính kháng tiền ảnh thứ hai, nhưng không bao gồm tính chất kháng tiền ảnh thứ nhất. Trên thực tế, hàm băm chỉ có khả năng kháng tiền ảnh thứ hai được coi là không an toàn và do đó không được khuyến nghị cho các ứng dụng thực tế.

***2.2.3 Những dạng HASH cơ bản***

***+*** *Hàm băm MD5 :* MD5 được Ronald Rivest thiết kế vào năm 1991 để thay thế hàm băm MD4 trước đó . MD5 tạo ra một bản tóm tắt có kích thước 128 bit (16 byte). Với sức mạnh tính toàn và sự phát triển của công nghệ thám mã thời gian gần đây, chúng ta có thể tính toán các va chạm trong MD5 với độ phức tạp phép toán chỉ trong vòng vài giây khiến thuật toán không phù hợp với hầu hết các trường hợp sử dụng trong thực tế

***+*** *Hàm băm SHA–1 :* được phát triển như một phần của dự án Capstone của Chính phủ Hoa Kỳ. Phiên bản đầu tiên, thường được gọi là SHA-0 được xuất bản năm 1993. SHA-1 tạo ra bản tóm tắt có kích thước 160 bit (20 byte). Các va chạm chống lại thuật toán SHA-1 đầy đủ có thể được tạo ra bằng cách sử dụng tấn công phá vỡ. Do đó, hàm băm này cho đến nay được coi là không đủ an toàn.

***+*** *Hàm băm SHA–2 :* Do có nhiều phiên bản thuật toán khác nhau do đó kích thước đầu ra của họ SHA-2 cũng khác nhau tùy theo thuật toán. Phần mở rộng của tên phía sau tiền tố “SHA” chính là độ dài của thông điệp băm đầu ra. Ví dụ với SHA-224 thì kích thước đầu ra là 224 bit (28 byte), SHA-256 tạo ra 32 byte, SHA-384 tạo ra 48 byte và cuối cùng là SHA- 512 tạo ra 64 byte. Và chúng ta có thể đã biết rằng Bitcoin sử dụng hàm băm SHA-256 là một phiên bản trong họ SHA-2 này.

***+*** *Hàm băm SHA–3 :* SHA*–*3 được NIST phát hành vào ngày 5 tháng 8 năm 2015. Đây có lẽ là tiêu chuẩn hàm băm mới nhất cho đến hiện nay. SHA-3 là một tập con của họ nguyên thủy mật mã rộng hơn là Keccak. Thuật toán Keccak được đưa ra bởi Guido Bertoni, Joan Daemen, Michael Peeters và Gilles Van Assche. Keccak dựa trên cấu trúc bọt biển (sponge). Cấu trúc này cũng có thể được sử dụng để xây dựng các nguyên thủy mã hóa khác như các hệ mật mã dòng. SHA-3 cũng có các kích cỡ đầu ra tương tự như SHA-2 bao gồm: 224, 256, 384 và 512 bit.

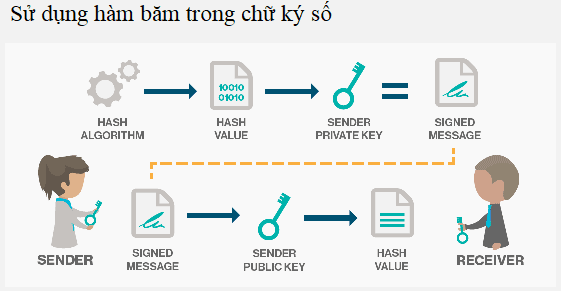
*+ Hàm băm RIPEMD–160 :* là họ hàm băm được phát triển tại Leuven,Bỉ lần đầu tiên được công bố vào năm 1996 dựa trên các nguyên tắc thiết kế được sử dụng trong MD4. RIPEMD-160 tạo ra một bản tóm tắt gồm 160 bit (20 byte). RIPEMD có hiệu năng tương tự như SHA-1 nhưng ít được phổ biến hơn. Và cho đến nay RIPEMD-160 chưa bị phá vỡ.

+ *Hàm băm Bcrypt :* bcrypt là một hàm băm mật khẩu được thiết kế bởi Niels Provos và David Mazières, dựa trên mật mã Blowfish, và được trình bày tại USENIX vào năm 1999. Bên cạnh việc kết hợp một giá trị ngẫu nghiên salt để bảo vệ chống lại các tấn công rainbow attack, bcrypt còn là một hàm có khả năng thích ứng: theo thời gian, số lần lặp có thể được tăng lên để làm cho nó chậm hơn, do đó nó vẫn có khả năng chống lại các cuộc tấn công vét cạn ngay cả khi tăng sức mạnh tính toán có lớn đến mức nào đi chăng nữa.

+ *Hàm băm Whirlpool :* là một hàm băm mật mã được thiết kế bởi Vincent và Barreto. Nó được mô tả đầu tiên vào năm 2000. Whirlpool dựa trên phiên bản sửa đổi đáng kể của Tiêu chuẩn mã hóa nâng cao (AES). Whirlpool tạo ra một bản tóm tắt có độ dài 512 bit (64 byte) của dữ liệu.

+ *Hàm băm BLAKE2 :* Một phiên bản cải tiến của BLAKE có tên BLAKE2 đã được công bố vào ngày 21 tháng 12 năm 2012. Khi chạy trên các kiến trúc 64x64 và ARM, BLAKE2b cho tốc độ nhanh hơn SHA-3, SHA-2, SHA-1 và MD5. Mặc dù BLAKE và BLAKE2 chưa được tiêu chuẩn hóa như SHA-3, nhưng nó đã được sử dụng trong nhiều giao thức bao gồm hàm băm mật khẩu Argon2 do hiệu quả cao mà nó mang lại cho các dòng CPU hiện đại. Do BLAKE cũng là ứng cử viên cho tiêu chuẩn SHA-3, vì vậy, BLAKE và BLAKE2 đều có các kích thước đầu ra giống như SHA-3 và có thể tùy chọn khi sử dụng trong thực tế.

***2.2.4 Ứng dụng của HASH***

******

*Hình – Ảnh minh hoạ về ứng dụng của HASH*

+ Hashing trong định danh tệp hoặc dữ liệu

Giá trị băm cũng có thể được sử dụng như một phương tiện để định danh tập tin một cách tin cậy. Một số hệ thống quản lý mã nguồn, như Git, Mercurial hay Monotone, sử dụng giá trị sha1sum của nội dung tệp, cây thư mục, thông tin thư mục gốc để định danh chúng.

+ Hashing trong xác minh tính toàn vẹn của thông điệp hoặc tập tin

Một ứng dụng quan trọng nhất của hashing là xác minh tính toàn vẹn của thông điệp.Chúng ta khá quen thuộc với các ứng dụng này. Khi download một phần mềm hoặc tệp tin nào đó trên một số trang web, ta được cung cấp kèm theo các mã băm MD5 hoặc SHA1. Khi đó sau khi tải về tập tin, chúng ta có thể tính và so sánh giá trị băm của tệp tải về với giá trị băm được cung cấp trên web, nếu có sự sai khác tức là tệp tin chúng ta tải về đã bị sửa đổi.

+ Hashing trong tạo và xác nhận chữ ký

Hầu như tất cả các lược đồ chữ ký số đều yêu cầu tính toán bản tóm lược của thông điệp bằng các hàm băm mật mã. Điều này cho phép việc tính toán và tạo chữ ký được thực hiện trên một khối dữ liệu có kích thước tương đối nhỏ và cố định thay vì trên toàn bộ văn bản dài. Tính chất toàn vẹn thông điệp của hàm băm mật mã được sử dụng để tạo các lược đồ chữ ký số an toàn và hiệu quả.

**2.3 Mã hoá đối xứng (Symmetric Key Encryption)**

***2.3.1 Tổng quan về mã hoá***

Mã hoá đối xứng là phương pháp mã hóa có sử dụng khóa mã hóa (Key Encryption), khóa giúp thuật toán có thể nhìn vào để từ đó tiến hành mã hóa cũng như giải mã. Cả người gửi và người nhận sẽ dùng chung một khóa bí mật cho việc mã hóa cũng như giải mã thông điệp gửi và nhận.

Dó đó, phương pháp mã hóa đối xứng còn được gọi là phương pháp mã khóa khóa bí mật hoặc phương pháp mã khóa khóa riêng tư. Vì độ an toàn của phương pháp này phụ thuộc vào khóa bí mật dùng cho mã hóa và giải mã, khóa đó phải được thống nhất và giữ an toàn giữa bên gửi và bên nhận, nếu bị bên thứ ba lấy được thì sẽ có thể bị đọc các thông điệp gửi đi. Các thuật toán thường thấy hiện này là DES và AES.



*Hình – Mô hình mã hoá đối xứng*

Quy trình mã hoá được diễn ra như sau :

+ Người gửi sẽ mã hóa thông tin muốn truyền đi bằng một thuật toán kèm theo khóa.

+ Người nhận sẽ nhận được thông tin đã mã hóa kèm theo khóa và dùng khóa để giải mã thông tin.

Ưu điểm : Phương pháp mã hóa đối xứng có tốc độ mã hóa và giải mã nhanh hơn so với phương pháp mã hóa khóa công khai.

Nhược điểm :

+ Phương pháp mã hóa đối xứng chưa thật sự an toàn vì: Quá trình mã hóa và giải mã đều dùng chung một khóa bí mật, vì vậy người gửi và người nhận cần phải giữ an toàn và bí mật tuyệt đối cho khóa này; nếu bị lọt khóa bí mật ra ngoài thì người bên thứ ba hoàn toàn có thể suy ra được khóa dùng cho quá trình giải mã và ngược lại.

+ Việc thỏa thuận và quản lý khóa chung mang đến sự khó khăn và phức tạp vì người gửi và người nhận phải luôn thống nhất về khóa với nhau; đồng thời khóa cần được gửi cho nhau thông qua các kênh an toàn để tránh bị lộ ra bên ngoài khi có sự thay đổi về khóa.

***2.3.2 Mã khối (Block chain)***

Mã khối được cấu trúc trên nguyên tắc là các thông điệp, bản tin sẽ được chia ra làm các khối có kích thước bằng nhau (đơn vị tính là bit), việc mã hóa sau đó sẽ được tiến hành trên từng khối độc lập.

Độ an toàn và bảo mật của mã khối sẽ phụ thuộc vào độ dài của khối cũng như độ phức tạp của thuật toán mã hóa được dùng. Nếu phân chia độ dài các khối quá ngắn thì việc dò tìm đặc tính cấu trúc thống kê của bản tin sẽ không mấy khó khăn; và việc tăng kích thước của khối thì mức độ cấu trúc thống kê sẽ tăng theo số mũ; nhưng nếu kích thước khối tăng dần kích thước bản tin thì tác dụng mã khối sẽ giảm dần đi. Dó đó cần phải phân chia kích thược của khối cho phù hợp.

Với mã khối sẽ có 4 phương pháp ứng dụng thường gặp :

+ Phương pháp dùng từ điển điện tử – ECB (Electronic CodeBook).

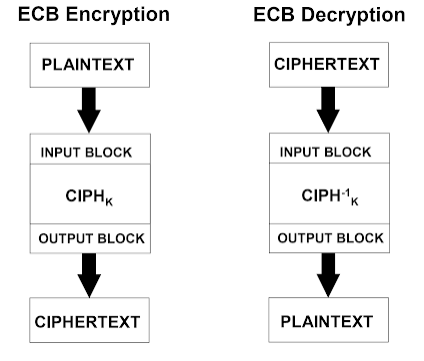
+ Phương pháp móc xích các khối đã mã hóa – CBC (Cipher Block Chaining).

+ Phương pháp phản hồi bản tin đã mã hóa – CFB (Cipher Feedback).

+ Phương pháp phản hồi đầu ra – OFB (Output Feedback).

***2.3.2.1 Phương pháp dùng từ điển điện tử (Electronic Codebook)***

Phương pháp ECB trong Mã khối là phương pháp mã hóa từng khối độc lập. Với việc sử dụng cùng một khóa mã K cho cả quá trình mã hóa và giải mã, lúc này mỗi giá trị plaintext được ánh xạ một – một thành một giá trị ciphertext thông qua khóa mã khóa.



*Hình – Quy trình mã hoá và giải mã của ECB*

*Quá trình mã hoá :*

+ Biểu thức định nghĩa : Cj = CIPHk(Pj) với j=1, 2, 3, …, n

=> PlainText là đầu vào sẽ thông qua thuật toán mã hóa với khóa mã khóa K để tạo ra CipherText

*Quá trình giải mã :*

+ Biểu thức định nghĩa : Pj = CIPHINVk(Cj)

với j=1, 2, 3 ,…, n

=> CipherText là đầu vào sẽ thông qua thuật toán giải mã với khóa mã K để chuyển về dạng CipherText.

*Ưu điểm :*

+ Lỗi bit không bị lan truyền. Nếu một khối xảy ra lỗi bit khi thực hiện mã hóa thì chỉ gây lỗi cho quá trình giải mã của khối đó, không ảnh hưởng đến việc giải mã của các khối còn lại.

+ Có thể thực hiện song song nhiều khối cùng một lúc vì các khối không có sự liên quan lẫn nhau; từ đó giúp tăng tốc độ mã hóa.

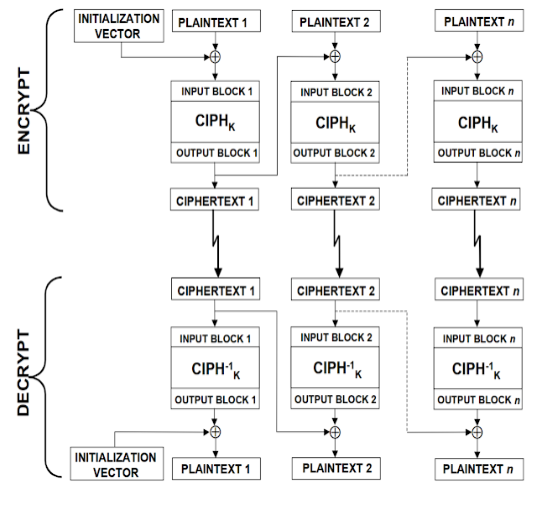
*Nhược điểm :*

+ Độ bảo mật kém vì các giá trị Plaintext và CipherText đều được ánh xạ một – một với nhau, do đó người bên thứ ba có thể sử dụng các dạng dữ liệu tương tự để phát hiện ra quy luật. Ngoài ra, việc xóa bỏ, chèn thêm hoặc đảo vị trí các bit còn có thể làm sai lệch thông tin được mã hóa.

+ Các khối tin không có sự liên hệ với nhau, khiến việc bảo mật trở nên yếu hơn.

***2.3.2.2 Phương pháp móc xích khối mã hoá (Cipher Block Chaining)***

Phương pháp CBC trong Mã khối là phương pháp sử dụng kết quả mã hóa (CipherText) của khối trước đó để tổ hợp với đầu vào (PlainText) của khối tiếp theo, trước khi tiến hành quá trình mã hóa. Như vậy mỗi một khối khi được mã hóa sẽ không chỉ phụ thuộc vào PlainText của khối đó mà còn phụ thuộc vào CipherText của khối mã trước đó, tạo nên sự liên hệ giữa các khối mã với nhau.



*Hình – Quy trình mã hoá và giải mã của CBC*

*Quá trình mã hoá :*

+ Biểu thức định nghĩa :

C1 = CIPHk(P1+IV)

Cj = CIPHk(Pj+Cj-1) với j=2, 3, …, n

+ Với lần mã hoá đầu tiên :

PlainText ở khối đầu sẽ XOR với vector khởi tạo IV.

Sau đó sẽ tiếp tục thực thi thuật toán mã khóa với khóa mã K.

+ Với các lần mã hoá về sau :

PlainText của khối sẽ được XOR với CipherText của lần mã hóa khối trước đó.

Sau đó sẽ tiếp tục thực thi thuật toán mã khóa với khóa mã K.

*Quá trình giải mã :*

+ Biểu thức định nghĩa :

P1 = CIPHINVk(C1)+IV

Pj = CIPHINVk(Cj)+Cj-1 với j=2, 3, …, n

+ Với lần giải mã đầu tiên :

CipherText thực hiện thuật toán giải mã với khóa K.

Sau đó kết quả thu được sẽ XOR với vector khởi tạo (IV) để thu được PlainText.

+ Với các lần giải mã về sau :

CipherText thực hiện thuật toán giải mã với khóa K.

Sau đó kết quả thu được sẽ XOR với CipherText của lần giải mã trước đó để thu được PlainText.

*Ưu điểm :*

+ Khả năng bảo mật được cải thiện hơn so với ECB.

+ Kết quả CipherText của các khối sau khi mã hóa sẽ có sự khác nhau do có sự kết hợp với Vector khởi tạo hoặc với CipherText của các lần mã hóa trước. Giúp thông tin trở nên khó bị đánh cắp hơn.

+ Quá trình giải mã vẫn có thể thực hiện song song nhiều khối cùng lúc.

*Nhược điểm :*

+ Logic ngoài việc quan tâm đến thuật toán mã hóa thì còn phải quan tâm đến việc quản lý độ dài chuỗi dữ liệu mã hóa và bộ tạo giá trị ngẫu nhiên cho Vector khỏi tạo (VI).

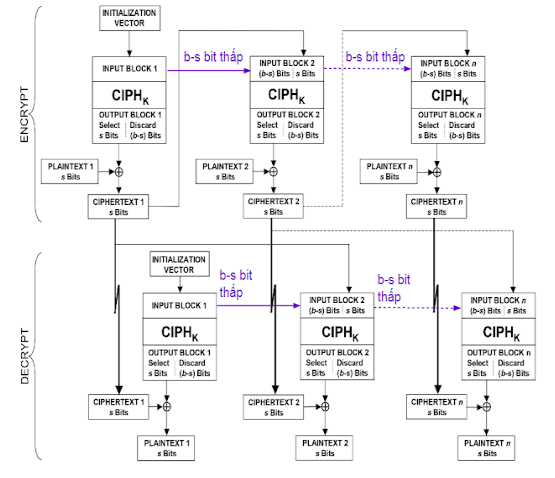
+ Lỗi bit bị lan truyền, vì khối sau có quan hệ với khối trước đó nên khi CipherText của một khối bị lỗi thì sẽ không chỉ ảnh hưởng việc giải mã thông tin khối đó mà còn sẽ kéo theo việc giải mã thông tin của các khối tiếp theo bị sai lệch.

+ Không thể thực hiện mã hóa song song vì khối sau cần sử dụng kết quả mã hóa (CipherText) của khối trước để tiến hành mã hóa (trừ khối đầu tiên).

***2.3.2.3 Phương pháp phản hồi bản tin đã mã hoá (Cipher Feedback)***

Phương pháp CFB trong Mã khối là phương pháp sử dụng CipherText của lần mã hóa hiện tại để phản hồi (Feedback) đến đầu vào của lần mã hóa tiếp theo; tức là kết quả CipherText hiện tại sẽ là đầu vào của thuật toán mã hóa để tạo ra CipherText cho lần mã hóa tiếp theo.

Phương pháp này thường được sử dụng cho các tập tin mã hóa gửi đi dưới dạng byte hoặc bit, với yêu cầu không làm chậm trễ tốc độ đường truyền khi gửi và nhận.



*Hình – Quy trình mã hoá và giải mã của CFB*

*Quá trình mã hoá :*

+ Biểu thức định nghĩa :

I1 = IV

Ij = LSBb-s(Ij-1) | C#j-1  với j=2, 3, ..., n

Oj = CIPHk(Ij)                 với j=1, 2, ..., n

C#j = P#j + MSBs(Oj)      với j=1, 2, ..., n

+ Với lần mã hoá đầu tiên :

Khối dữ liệu vào lấy Vector khởi tạo (IV) ứng với I1.

Vector khởi tạo (IV) sẽ được mã hóa để tạo ra một khối giá trị chứa b bit, ứng với Oj mà ở đây sẽ là O1

Số bit nhận được ở kết quả trên sẽ tiếp tục được MSB để lấy ra s bit và sau đó sẽ XOR với s bit của PlainText để tạo ra s bit CipherText, ứng với C#j mà ở đây sẽ là C#1.

+ Với các lần mã hoá về sau :

Khối dữ liệu ngõ vào sẽ lấy b-s bit LSB của khối ngõ vào kết hợp với s bit CipherText của lần mã hóa trước đó, ứng vói biểu thức Ij

Sau đó giá trị thu được sẽ được mã hóa để tạo ra một khối giá trị chứa b bit, ứng với Oj.

Số bit nhận được ở kết quả trên sẽ tiếp tục được MSB để lấy ra s bit và sau đó sẽ XOR với s bit của PlainText để tạo ra s bit CipherText, ứng với C#j.

*Quá trình giải mã :*

+ Biểu thức định nghĩa :

I1 = IV

Ij = LSBb-s(Ij-1) | C#j-1  với j=2, 3, ..., n

Oj = CIPHINVk(Ij)        với j=1, 2, ..., n

P#j = C#j + MSBs(Oj)      với j=1, 2, ..., n

+ Với lần giải mã đầu tiên:

Khối dữ liệu vào lấy Vector khởi tạo (IV) ứng với I1.

Vector khởi tạo (IV) sẽ được mã hóa để tạo ra một khối giá trị chứa b bit, ứng với Oj mà ở đây sẽ là O1

Số bit nhận được ở kết quả trên sẽ tiếp tục được MSB để lấy ra s bit và sau đó sẽ XOR với s bit của CipherText ứng với khối đầu tiên để tạo ra s bit PlainText, ứng với P#j mà ở đây sẽ là P#1.

+ Với các lần giải mã về sau:

Khối dữ liệu ngõ vào sẽ lấy b-s bit LSB của khối ngõ được mã hóa trước đó và kết hợp với s bit CipherText của khối trước đó, ứng với biểu thức Ij

Sau đó giá trị thu được sẽ được mã hóa để tạo ra một khối giá trị chứa b bit, ứng với Oj.

Số bit nhận được ở kết quả trên sẽ tiếp tục được MSB để lấy ra s bit và sau đó sẽ XOR với s bit của CipherText ứng với khối hiện tại để tạo ra s bit PlainText, ứng với P#j.

*So sánh CBC và CFB :*

*+* Việc mã hóa không áp dụng trực tiếp lên PlainText mà việc mã hóa sẽ được dùng lên Vector khởi tạo hoặc CipherText trước đó.

+ Số lượng bit dữ liệu được mã hóa hoặc giải mã có thể lớn hơn hoặc bằng số lượng bit thuật toán hỗ trợ.

*Ưu điểm :*

*+* Khả năng bảo mật cao hơn ECB.

*+* CipherText của khối dữ liệu Plaintext khác nhau do có sự kết hợp với Vector khởi tạo hoặc CipherText của khối dữ liệu trước đó.

*+* Quá trình giải mã có thể thực hiện song song giữa nhiều khối với nhau.

*+* Độ dài khối dữ liệu mã hóa có thể được tùy biến thông qua tham số s.

*Nhược điểm :*

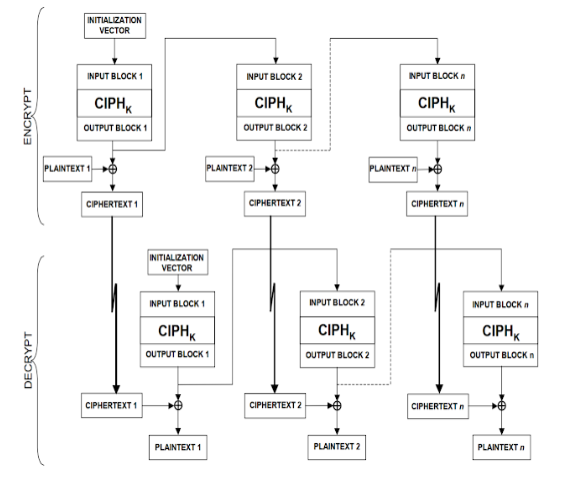
+ Thiết kế logic phức tạp.

+ Lỗi bit bị lan truyền, nếu một bit bị lỗi trên CipherText sẽ ảnh hưởng đến kết quả của khối dữ liệu đó khi được giải mã, đồng thời còn kéo theo các khối dữ liệu sau cũng bị sai.

+ Không thể thực hiện việc mã hóa song song vì khối dữ liệu sau cần kết quả CipherText của khối dữ liệu trước (trừ khối đầu tiên).

***2.3.2.4 Phương pháp phản hồi đầu ra (Output Feedback)***

Phương pháp OFB trong Mã khối là phương pháp sử dụng giá trị ngõ ra của lần mã hóa hiện tại để phản hồi (Feedback) đến đầu vào của lần mã hóa tiếp theo; tức là giá trị ngõ ra của lần mã hóa khối trước đó sẽ được dùng làm giá trị đầu vào cho việc tạo CipherText của khối tiếp theo. Phương pháp này có thể chập nhận một sai số nào đó.



*Hình – Quy trình mã hoá và giải mã của OFB*

*Quá trình mã hoá :*

+ Biểu thức định nghĩa :

I1 = IV

Ij = Oj-1                            với j = 2, 3, ..., n

Oj = CIPHk(Ij)                 với j = 1, 2, ..., n

Cj = Pj + Oj                      với j = 1, 2, ..., n-1

C\*n = P\*n + MSBu(On)

+ Với lần mã hóa đầu tiên :

Giá trị Vector khởi tạo sẽ được sử dụng là giá trị đầu vào mã hóa, ứng với I1.

Sau đó giá trị này sẽ được đưa vào thuật toán mã hóa với khóa mã K, ứng với Qj.

Với kết quả nhận được sẽ tiến hành XOR với PlainText để thu được CipherText, ứng với Cj.

+ Với các lần mã hóa về sau :

Giá trị của khối ngõ ra trước đó sẽ được sử dụng là giá trị đầu vào mã hóa, ứng với Ij.

Sau đó giá trị này sẽ được đưa vào thuật toán mã hóa với khóa mã K, ứng với Qj.

Với kết quả nhận được sẽ tiến hành XOR với PlainText để thu được CipherText, ứng với Cj.

+ Với các lần mã hóa cuối cùng :

Giá trị của khối ngõ ra trước đó sẽ được sử dụng là giá trị đầu vào mã hóa, ứng với Ij.

Sau đó giá trị này sẽ được đưa vào thuật toán mã hóa với khóa mã K, ứng với Qj.

Với kết quả nhận được sẽ tiến hành XOR với PlainText theo độ dài còn lại của PlainText để thu được CipherText, ứng với C\*n.

*Quá trình giải mã :*

+ Biểu thức định nghĩa:

I1 = IV

Ij = Oj-1                với j = 2, 3, ..., n

Oj = CIPHk(Ij)              với j = 1, 2, ..., n

Pj = Cj + Oj                      với j = 1, 2, ..., n-1

P\*n = C\*n + MSBu(On)

+ Với lần giải mã đầu tiên :

Giá trị Vector khởi tạo sẽ được sử dụng là giá trị đầu vào mã hóa, ứng với I1.

Sau đó giá trị này sẽ được đưa vào thuật toán mã khóa với khóa mã K, ứng với Qj.

Với kết quả thu được sẽ XOR với CipherText để thu được PlainText, ứng với Pj.

+ Với các lần giải mã về sau :

Giá trị của khối ngõ ra trước đó sẽ được sử dụng là giá trị đầu vào mã hóa, ứng với Ij.

Sau đó giá trị này sẽ được đưa vào thuật toán mã hóa với khóa mã K, ứng với Qj.

Với kết quả nhận được sẽ tiến hành XOR với CipherText để thu được PlainText, ứng với Cj.

+ Với các lần mã hóa cuối cùng :

Giá trị của khối ngõ ra trước đó sẽ được sử dụng là giá trị đầu vào mã hóa, ứng với Ij.

Sau đó giá trị này sẽ được đưa vào thuật toán mã hóa với khóa mã K, ứng với Qj.

Với kết quả nhận được sẽ tiến hành XOR với CipherText theo độ dài còn lại của CipherText để thu được PlainText, ứng với P\*n.

*So sánh CBC và OFB :*

Việc mã hóa không áp dụng trực tiếp lên PlainText mà việc mã hóa sẽ được dùng lên Vector khỏi tạo và các ngõ ra của những lần mã hóa trước đó.

*Ưu điểm :*

+ Khả năng bảo mật cao hơn ECB.

+ CipherText của khối dữ liệu Plaintext khác nhau do có sự kết hợp với Vector khởi tạo hoặc CipherText của khối dữ liệu trước đó.

+ Lỗi bit không bị lan truyền, khi CipherText của một khối bị lỗi sẽ chỉ ảnh hưởng kết quả của khối đó chứ không ảnh hưởng đến kết quả của các khối còn lại.

*Nhược điểm :*

+ Không thể thực hiện việc mã hóa/giải mã song song vì khối dữ liệu sau cần kết quả của khối ngõ ra lần mã hóa/giải mã trước đó.

***2.3.3 Mã dòng (Stream Cipher)***

Mã dòng là cơ chế mã hóa sử dụng khóa dòng để mã hóa bản rõ theo từng bit hoặc từng khối.

Một hệ mã dòng là một bộ (**P, C, K, L, F, E, D**), thỏa mãn :

**P** là một tập hữu hạn các PlainText.

**C** là một tập hữu hạn các CipherText.

**K** là một tập hữu hạn các khóa chính.

**L** là một tập hữu hạn các khóa dòng.

**F** là một tập các bộ sinh khóa dòng, fi = K x Li-1 x Pi-1 -> L

**E** là các hàm mã hóa ứng với từng khóa dòng.

**D** là các hàm giải mã ứng với từng khóa dòng.

Với mỗi khóa dòng z ∈ **L** sẽ có một hàm mã hóa ez ∈ **E**: **P -> C** và một hàm giải mã dz ∈ **D**: **C -> P** sao cho dz(ez(x)) = x với mọi x ∈ **P.**

Nếu bộ sinh khóa dòng không phụ thuộc vào PlainText thì gọi là đồng bộ, khi đó **K** được xem như “hạt giống” để tạo ra các dòng khóa z1, z2,…

Trên thực tế, khóa dòng thường được dùng cho các văn bản nhị phân, P = C = Z2 = {0, 1}, và có phép mã hóa và giải mã là :

ez(x) = x + z mod 2

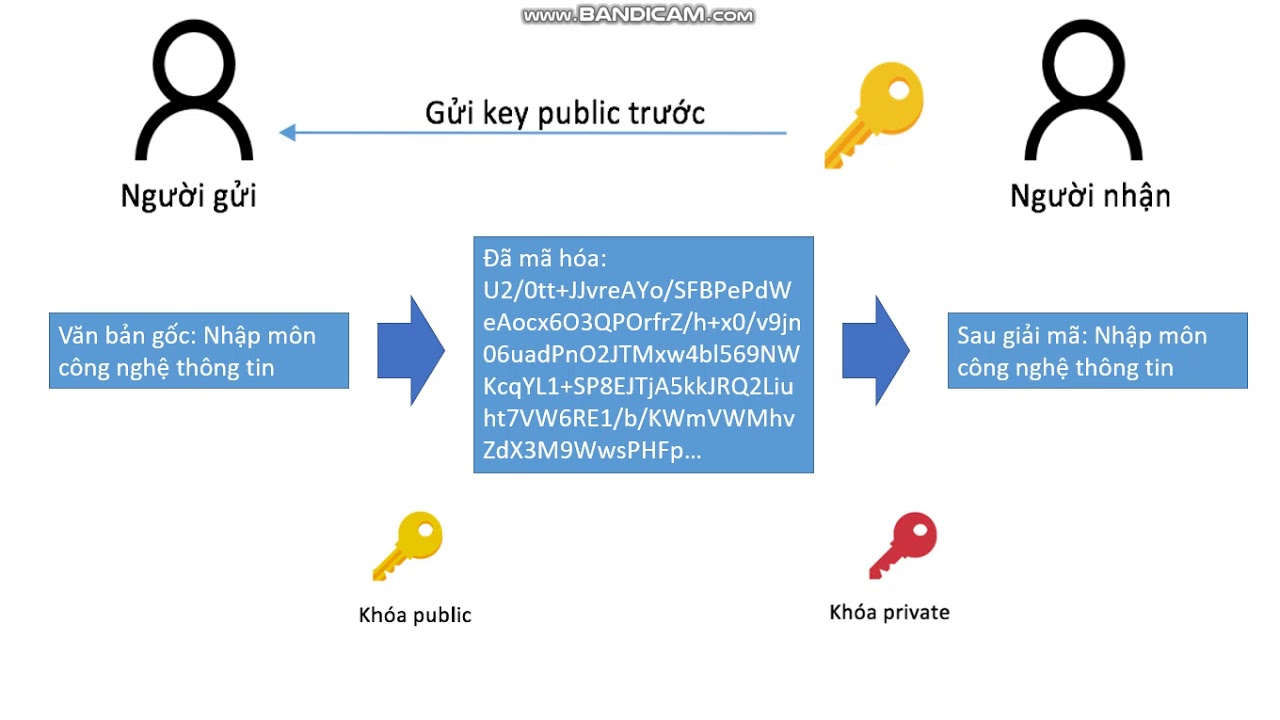
dz(y) = y + z mod 2

**2.4 Mã hoá bất đối xứng (Public Key Encryption)**

***2.4.1 Tổng quan về mã hoá***

Mã hóa bất đối xứng dùng để gửi dữ liệu một cách an toàn qua các mạng không an toàn như Internet. Đây cũng là phương pháp mã hóa sử dụng khóa, có thể nói nó là phiên bản nâng cấp mã hóa đối xứng. Thay đổi của nó là thay vì cả người gửi và người nhận cùng dùng chung một khóa mã khóa thì lúc này hai khóa của hai bên sẽ hoàn toàn khác nhau; trong đó khóa mã hóa gọi là public key, khóa giải mã gọi là private key.

Một số hệ mật mã bất đối xứng phổ biến như RSA (1977), ElGamal (1978), Merkle-Helman.



*Hình – Quy trình mã hoá bất đối xứng*

***2.4.2 Đánh giá phương pháp mã hoá***

*Ưu điểm :*

+ Bên thứ ba khó có thể giải mã dữ liệu dù có lấy được dữ liệu.

+ Các phương pháp mã hóa công cộng giúp cho việc trao đổi mã khóa trở nên dễ dàng hơn.

+ Thuật toán được viết công khai và có thể dùng nhiều lần, chỉ cần giữ bí mật cho private key của riêng mình.

*Nhược điểm :*

+ Tốc độ giải mã chậm hơn so với mã hóa đối xứng. Khoảng thời gian lâu hơn bao nhiêu tùy vào thuật toán, cách thức mã hóa và key.

Mã hóa bất đối xứng thường được sử dụng chủ yếu trên các mạng công khai trên Internet. Giao thức HTTPS là một ví dụ, nó là dạng mã hóa thông tin đang di chuyển, người ta còn có thể dùng mã hóa để đảm bảo an toàn cho rất nhiều thứ khác. Tất nhiên, không gì là hoàn hảo chính vì thế nó vẫn có lỗ hổng trong cơ chế.

**2.5 Khác nhau giữa mã hoá đối xứng và mã hoá bất đối xứng**

+ Mã hóa đối xứng là một kỹ thuật đã được giới thiệu từ lâu, trong khi mã hóa bất đối xứng là kỹ thuật mới hơn.

+ Mã hóa bất đối xứng mất nhiều thời gian hơn để thực hiện do logic phức tạp liên quan. Vì lý do này, mã hóa đối xứng vẫn được ưu tiên sử dụng khi truyền dữ liệu hàng loạt.

+ Mã hóa bất đối xứng an toàn hơn vì nó sử dụng các key khác nhau cho quá trình mã hóa và giải mã.

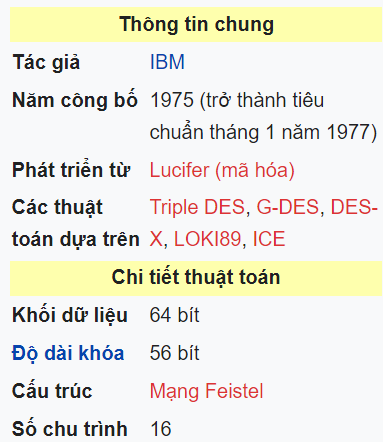
Cả hai phương thức mã hóa đều sở hữu những ưu, nhược điểm riêng. Tuy nhiên nếu nhìn từ góc độ bảo mật, mã hóa bất đối xứng chắc chắn là lựa chọn tối ưu hơn so với mã hoá đối xứng.

**CHƯƠNG III – CÁC THUẬT TOÁN MÃ HOÁ PHỔ BIẾN**

**3.1 Tiêu chuẩn mã hoá dữ liệu (DES)**

***3.1.1 Tổng quan về thuật toán***

*DES – Data Encryption Standard –* là một phương pháp mật mã hoá được FIPS (Tiêu chuẩn Xử Lý Thông Tin Liên bang Hoa Kỳ) chọn làm chuẩn chính thức vào năm 1976. Sau đó, chuẩn này được sử dụng rộng rãi trên phạm vi thế giới. Ngay từ đầu, thuật toán của nó đã gây ra rất nhiều trang cãi, do nó bao gồm các thành phần thiết kế mật, độ dài khoá tương đối ngắn và các nghi ngờ về “cửa sau” để Cơ quan An Ninh quốc gia Hoa Kỳ (NSA) có thể bẻ khoá. Do đó, DES đã được giới nghiên cứu xem xét rất kỹ lưỡng, việc này đã thúc đẩy hiểu biết hiện đại về mật mã khối *(block cipher)* và các phương pháp thám mã tương ứng.



*Hình – thông tin thuật toán DES*

***3.1.2 Mô tả thuật toán***

*DES* là thuật toán mã hoá khối, nó xử lý từng khối thông tin của [bản rõ](https://vi.wikipedia.org/wiki/Plain_text) có độ dài xác định và biến đổi theo những quá trình phức tạp để trở thành khối thông tin của [bản mã](https://vi.wikipedia.org/w/index.php?title=B%E1%BA%A3n_m%C3%A3&action=edit&redlink=1) có độ dài không thay đổi. Trong trường hợp của *DES*, độ dài mỗi khối là 64 bit. *DES* cũng sử dụng [khóa](https://vi.wikipedia.org/wiki/Kh%C3%B3a_(m%E1%BA%ADt_m%C3%A3)) để cá biệt hóa quá trình chuyển đổi. Nhờ vậy, chỉ khi biết khóa mới có thể giải mã được văn bản mã. Khóa dùng trong *DES* có độ dài toàn bộ là 64 bit.

Tuy nhiên chỉ có 56 bit thực sự được sử dụng; 8 bit còn lại chỉ dùng cho việc kiểm tra. Vì thế, độ dài thực tế của khóa chỉ là 56 bit. Giống như các thuật toán mã hóa khối khác, khi áp dụng cho các văn bản dài hơn 64 bit, *DES* phải được dùng theo một phương pháp nào đó.

*DES* có tính chất bù :



Trong đó là phần bù của {\displaystyle x}theo từng bít (1 thay bằng 0 và ngược lại). *EK* là bản mã hoá của *E* với khoá *K, P và C* là văn bản rõ (trước khi mã hoá) và văn bản mã (sau khi mã hoá). Do tính chất bù, ta có thể giảm độ phức tạp của tấn công bạo lực xuống 2 lần (tương ứng với 1 bít) với điều kiện là ta có thể lựa chọn bản rõ.

Ngoài ra, *DES* còn có 4 *khoá yếu* *(weak keys)*. Khi sử dụng khoá yếu thì mã hoá *E* và giải mã *D* sẽ cho ra cùng kết quả :



Bên cạnh đó, còn có 6 cặp *khóa nửa yếu* *(semi-weak keys).* Mã hóa với một khóa trong cặp, *{\displaystyle K\_{1}}K1*, tương đương với giải mã với khóa còn lại *K2* :



Tuy nhiên có thể dễ dàng tránh được những khóa này khi thực hiện thuật toán, có thể bằng cách thử hoặc chọn khóa một cách ngẫu nhiên. Khi đó khả năng chọn phải khóa yếu là rất nhỏ.

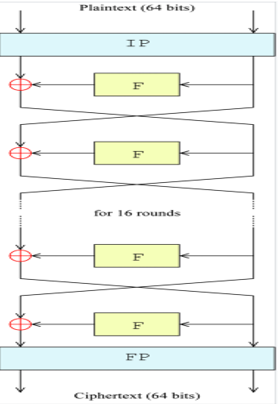
*DES* đã được chứng minh là không tạo thành [nhóm](https://vi.wikipedia.org/wiki/Nh%C3%B3m_(%C4%91%E1%BA%A1i_s%E1%BB%91)). Nói một cách khác, tập hợp {\displaystyle \{E\_{K}\}}(cho tất cả các khóa có thể) theo phép hợp thành không tạo thành một nhóm hay gần với một nhóm *(Campbell and Wiener, 1992).* Vấn đề này đã từng là một câu hỏi mở trong khá lâu và nếu như tạo thành nhóm thì *DES* có thể bị phá vỡ dễ dàng hơn bởi vì việc áp dụng *DES* nhiều lần (ví dụ như trong *TripleDES*) sẽ không làm tăng thêm độ an toàn của *DES*.

***3.1.3 An toàn và sự giải mã***

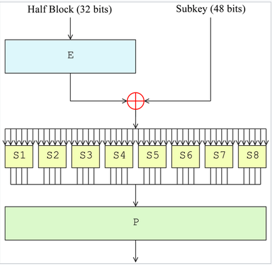
Mặc dù đã có nhiều nghiên cứu về phá mã *DES* hơn bất kỳ phương pháp mã hóa khối nào khác nhưng phương pháp phá mã thực tế nhất hiện nay vẫn là tấn công *Brute-force*. Nhiều đặc tính mật mã hóa của *DES* đã được xác định và từ đó ba phương pháp phá mã khác được xác định với mức độ phức tạp nhỏ hơn tấn công bạo lực. Tuy nhiên các phương pháp này đòi hỏi một số lượng [bản rõ](https://vi.wikipedia.org/wiki/Plain_text) quá lớn (để [tấn công lựa chọn bản rõ](https://vi.wikipedia.org/w/index.php?title=T%E1%BA%A5n_c%C3%B4ng_l%E1%BB%B1a_ch%E1%BB%8Dn_b%E1%BA%A3n_r%C3%B5&action=edit&redlink=1)) nên hầu như không thể thực hiện được trong thực tế.



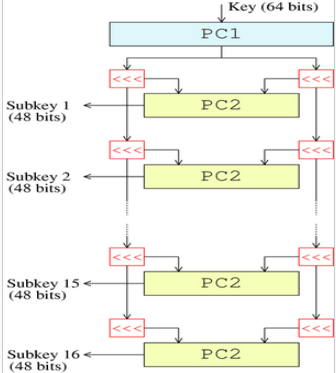
*Hình – Hệ thống phá mã DES*



*Hình – Cấu trúc thuật toán DES*

**

*Hình – Hàm F dùng trong DES*

**

*Hình – Quá trình tạo khoá con trong DES*

**3.2 Tiêu chuẩn mã hoá tiên tiến (AES)**

***3.2.1 Tổng quan về thuật toán***

*AES – Advanced Encryption Standard –* là một thuật toán [mã hóa khối](https://vi.wikipedia.org/wiki/M%C3%A3_h%C3%B3a_kh%E1%BB%91i) được [chính phủ](https://vi.wikipedia.org/wiki/Ch%C3%ADnh_ph%E1%BB%A7) [Hoa kỳ](https://vi.wikipedia.org/wiki/Hoa_k%E1%BB%B3) áp dụng làm tiêu chuẩn mã hóa. Giống như tiêu chuẩn tiền nhiệm [*DES*](https://vi.wikipedia.org/wiki/DES_(%C4%91%E1%BB%8Bnh_h%C6%B0%E1%BB%9Bng)), *AES* được kỳ vọng áp dụng trên phạm vi thế giới và đã được nghiên cứu rất kỹ lưỡng. *AES* được chấp thuận làm tiêu chuẩn liên bang bởi [Viện tiêu chuẩn và công nghệ quốc gia Hoa kỳ](https://vi.wikipedia.org/w/index.php?title=Vi%E1%BB%87n_ti%C3%AAu_chu%E1%BA%A9n_v%C3%A0_c%C3%B4ng_ngh%E1%BB%87_qu%E1%BB%91c_gia_Hoa_k%E1%BB%B3&action=edit&redlink=1) (NIST) sau một quá trình tiêu chuẩn hóa kéo dài 5 năm.

Thuật toán được dựa trên bản thiết kế [*Square*](https://vi.wikipedia.org/w/index.php?title=Square_(m%C3%A3_h%C3%B3a)&action=edit&redlink=1) có trước đó của *Daemen* và *Rijmen*, còn *Square* lại được thiết kế dựa trên [*Shark*](https://vi.wikipedia.org/w/index.php?title=Shark_(m%C3%A3_h%C3%B3a)&action=edit&redlink=1). Khác với [*DES*](https://vi.wikipedia.org/wiki/DES_(%C4%91%E1%BB%8Bnh_h%C6%B0%E1%BB%9Bng)) sử dụng [mạng *Feistel*](https://vi.wikipedia.org/w/index.php?title=M%E1%BA%A1ng_Feistel&action=edit&redlink=1), *Rijndael* sử dụng [mạng thay thế-hoán vị](https://vi.wikipedia.org/w/index.php?title=M%E1%BA%A1ng_thay_th%E1%BA%BF-ho%C3%A1n_v%E1%BB%8B&action=edit&redlink=1). *AES* có thể dễ dàng thực hiện với tốc độ cao bằng [phần mềm](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m) hoặc [phần cứng](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_c%E1%BB%A9ng) và không đòi hỏi nhiều [bộ nhớ](https://vi.wikipedia.org/w/index.php?title=B%E1%BB%99_nh%E1%BB%9B_(m%C3%A1y_t%C3%ADnh)&action=edit&redlink=1). Do *AES* là một tiêu chuẩn mã hóa mới, nó đang được triển khai sử dụng đại trà.



*Hình – Thông tin thuật toán AES*

***3.2.2 Mô tả thuật toán***

Mặc dù 2 tên *AES* và *Rijndael* vẫn thường được gọi thay thế cho nhau nhưng trên thực tế thì 2 [thuật toán](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n) không hoàn toàn giống nhau. *AES* chỉ làm việc với các khối dữ liệu (đầu vào và đầu ra) 128 bít và [khóa](https://vi.wikipedia.org/wiki/Kh%C3%B3a_(m%E1%BA%ADt_m%C3%A3)) có độ dài 128, 192 hoặc 256 bít trong khi *Rijndael* có thể làm việc với dữ liệu và khóa có độ dài bất kỳ là bội số của 32 bít nằm trong khoảng từ 128 tới 256 bít. Các khóa con sử dụng trong các chu trình được tạo ra bởi [quá trình tạo khóa con *Rijndael*](https://vi.wikipedia.org/w/index.php?title=Qu%C3%A1_tr%C3%ACnh_t%E1%BA%A1o_kh%C3%B3a_con_Rijndael&action=edit&redlink=1). Mỗi khóa con cũng là một cột gồm 4 byte. Hầu hết các phép toán trong thuật toán *AES* đều thực hiện trong một [trường hữu hạn](https://vi.wikipedia.org/wiki/Tr%C6%B0%E1%BB%9Dng_h%E1%BB%AFu_h%E1%BA%A1n) của các byte. Mỗi khối dữ liệu 128 bit đầu vào được chia thành 16 byte (mỗi byte 8 bit),có thể xếp thành 4 cột, mỗi cột 4 phần tử hay là một [ma trận](https://vi.wikipedia.org/wiki/Ma_tr%E1%BA%ADn) 4x4 của các byte,nó được gọi là ma trận trạng thái, hay vắn tắt là trạng thái (tiếng Anh: *state*, trạng thái trong Rijndael có thể có thêm cột). Trong quá trình thực hiện thuật toán các toán tử tác động để biến đổi ma trận trạng thái này.

*Mở rộng khoá (KeyExpansion) :* Là quá trình tạo các vòng khóa từ khóa chính, mỗi khóa con chứa 4 byte.

Quy trình mã hoá :

+ Khởi động vòng lặp : *AddRoundKey* – mỗi cột của trạng thái đầu lần lượt được kết hợp với một khóa con theo thứ tự từ đầu dãy khóa.

+ Vòng lặp chính :

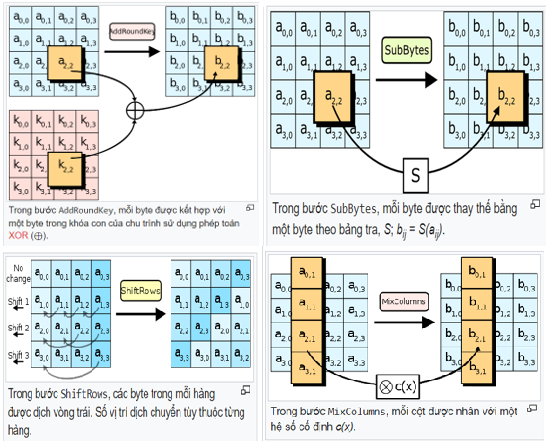
*SubBytes –* đây là phép thế (phi tuyến) trong đó mỗi byte trong trạng thái sẽ được thế bằng một byte khác theo bảng tra *(Rijndael S-box)*

*ShiftRows –* dịch chuyển, các hàng trong trạng thái được dịch vòng theo số bước khác nhau.

*MixColumns –* quá trình trộn làm việc theo các cột trong khối theo một phép biến đổi tuyến tính.

*AddRoundKey*

+ Vòng lặp cuối : *SubBytes – ShiftRows – AddRoundKey*

**

*Hình – Quy trình thuật toán trong AES*

Tối ưu hoá : Đối với các hệ thống 32 bít hoặc lớn hơn, ta có thể tăng tốc độ thực hiện thuật toán bằng cách sáp nhập các bước *SubBytes, ShiftRows, MixColumns* và chuyển chúng thành dạng bảng. Có cả thảy bốn bảng với 256 mục, mỗi mục là 1 từ 32 bít, bốn bảng này chiếm 4096 byte trong bộ nhớ. Khi đó, mỗi chu trình sẽ được bao gồm 16 lần tra bảng và 12 lần thực hiện phép *XOR* 32 bít cùng với 4 phép *XOR* trong bước *AddRoundKey*.

Trong trường hợp kích thước các bảng vẫn lớn so với thiết bị thực hiện thì chỉ dùng một bảng và tra bảng kết hợp với hoán vị vòng quanh.

***3.2.3 An toàn và giải mã***

Vào thời điểm năm [2006](https://vi.wikipedia.org/wiki/2006), dạng tấn công lên *AES* duy nhất thành công là [tấn công kênh bên](https://vi.wikipedia.org/wiki/T%E1%BA%A5n_c%C3%B4ng_k%C3%AAnh_b%C3%AAn) *(side channel attack).* Vào [tháng 6](https://vi.wikipedia.org/wiki/Th%C3%A1ng_s%C3%A1u) năm [2003](https://vi.wikipedia.org/wiki/2003), chính phủ [Hoa kỳ](https://vi.wikipedia.org/wiki/Hoa_k%E1%BB%B3) tuyên bố *AES* có thể được sử dụng cho thông tin mật. Điều này đánh dấu lần đầu tiên [công chúng](https://vi.wikipedia.org/wiki/C%C3%B4ng_ch%C3%BAng) có quyền tiếp xúc với thuật toán [mật mã](https://vi.wikipedia.org/wiki/M%E1%BA%ADt_m%C3%A3_h%E1%BB%8Dc) mà NSA phê chuẩn cho thông tin TUYỆT MẬT. Nhiều [phần mềm](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m) [thương mại](https://vi.wikipedia.org/wiki/Th%C6%B0%C6%A1ng_m%E1%BA%A1i) hiện nay sử dụng mặc định khóa có độ dài 128 bít.

Phương pháp thường dùng nhất để tấn công các dạng [mã hóa khối](https://vi.wikipedia.org/wiki/M%C3%A3_h%C3%B3a_kh%E1%BB%91i) là thử các kiểu tấn công lên phiên bản có số chu trình thu gọn. Đối với khóa 128 – 192 – 256 bít, *AES* có tương ứng 10, 12 và 14 chu trình. Tại thời điểm năm [2006](https://vi.wikipedia.org/wiki/2006), những tấn công thành công được biết đến là 7 chu trình đối với khóa 128 bít, 8 chu trình với khóa 192 bít và 9 chu trình với khóa 256 bit.

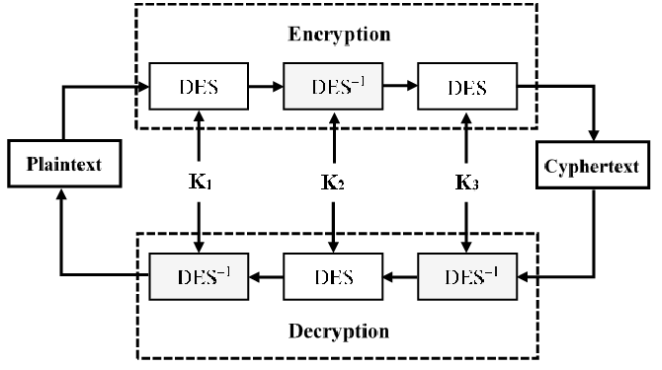
Một số nhà khoa học trong lĩnh vực [mật mã](https://vi.wikipedia.org/wiki/M%E1%BA%ADt_m%C3%A3_h%E1%BB%8Dc) lo ngại về an ninh của *AES*. Họ cho rằng ranh giới giữa số chu trình của thuật toán và số chu trình bị phá vỡ quá nhỏ. Nếu các kỹ thuật tấn công được cải thiện thì *AES* có thể bị phá vỡ.

Phá mã : Tấn công khoá liên quan có thể phá vỡ *AES* 256 bit với 9 chu trình. Tấn công lựa chọn bản rõ có thể phá vỡ *AES 192* và 256 với 8 chu trình, *AES* 128 với 7 chu trình *(Ferguson et al, 2000).*

**3.3 Thuật toán TripleDES**

***3.3.1 Tổng quan về thuật toán***

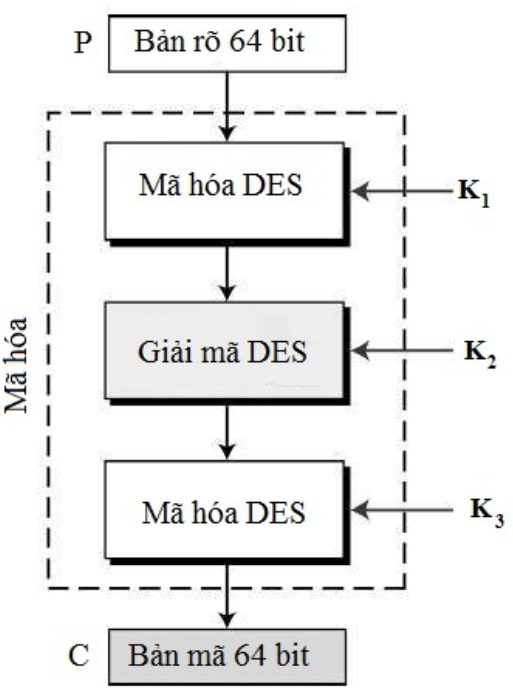
*TripleDES – 3DES –* là phiên bản DES mới hơn, an toàn hơn. Khi *DES* bị crack trong thời gian chưa đầy 23 giờ, người ta nhận ra vấn đề, do vậy, đây là lý do mà *TripleDES* được sinh ra. *TripleDES* tăng tốc quy trình mã hóa bằng cách chạy *DES* ba lần.



*Hình – Cấu trúc thuật toán TripleDES*

***3.3.2 Mô ta thuật toán***

Thuật toán *3DES* sử dụng một nhóm khóa bao gồm 03 khóa *DES* là *K1, K2 và K3*, mỗi khóa có giá trị 64 bít. Thuật toán mã hóa thực hiện như sau :



*Hình – Quy trình mã hoá trong TripleDES*

Trước tiên, thực hiện mã hóa DES với khóa K1, tiếp tục giải mã DES với khóa K2 và cuối cùng mã hóa DES với khóa K3(E – Encryption: quá trình mã hóa; D - Decryption: quá trình giải mã; Bản rõ: Dữ liệu đầu vào của phép mã hóa hoặc dữ liệu đầu ra của phép giải mã; Bản mã: Dữ liệu đầu ra của phép mã hóa hoặc dữ liệu đầu vào của phép giải mã).

Quá trình giải mã với việc  giải mã với khóa K3, sau đó mã hóa với khóa K2,và cuối cùng giải mã với khóa K1. 3DES mã hóa một khối dữ liệu có giá trị 64 bít (bản rõ) thành một khối dữ liệu mới có giá trị 64 bít (bản mã).

Các tiêu chuẩn chỉ ra phương thức lựa chọn nhóm khóa (K1,K2, K3) :

+ Lựa chọn 1: K1,K2, K3 là các khóa độc lập

+ Lựa chọn 2: K1,K2 là hai khóa độc lập và  K3 =K1

+ Lựa chọn 3: K1=K2=K3

*Lựa chọn 1* là phương thức mã hóa mạnh nhất với 168 bít khóa độc lập (168=3x56). Lựa chọn 2 ít bảo mật hơn với 112 bít khóa ( 2x56=112 bít) và lựa chọn 3 chỉ tương đương với việc mã hóa DES 1 lần với 56 bít khóa. Mỗi khóa DES thông thường được lưu trữ và truyền đi trong 8 byte, vì vậy một nhóm khóa yêu cầu 8 hoặc 16, 24 byte cho việc lưu trữ khóa.

***3.3.3 An toàn và giải mã***

Nói chung, *TripleDES* với 3 khoá chính độc lập, có độ dài chính là 168 bit (ba khoá DES 56 bit). Nhưng do các cuộc tấn công đều ở vị trí giao nhau ở giữa nên bảo mật hiệu quả mà nó cung cấp chỉ là 112 bit.

*Lựa chọn khoá 2* – làm giảm kích thước khoá hiệu quả xuống còn 112 bit (vì khoá thứ ba giống như khoá đầu tiên). Tuy nhiên, lựa chọn này dễ bị một số cuộc tấn công như *chosen-plaintext* và *known-plaintext*. Do đó, NIST đã chỉ định cho khoá có 80 bit bảo mật. Điều này có thể được coi là không an toàn và vì thế đã bị loại bỏ vào năm 2017.

Với kích thước khối ngắn 64 bit, làm cho thuật toán dễ bị chặn bởi các cuộc tấn công nếu nó được sử dụng để mã hoá một lượng lớn dữ liệu với cùng một khoá. Một số cuộc tấn công như *Sweet32* trong TLS và OpenVPN.

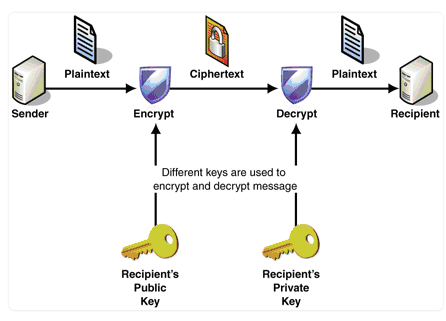
**3.4 Thuật toán RSA**

***3.4.1 Tổng quan về thuật toán***

RSA – được đặt tên theo người sáng tạo của nó là Ron Rivest, Adi Shamir và Leonard Adleman – là một thuật toán mật mã hóa khóa công khai. Đây là thuật toán đầu tiên phù hợp với việc tạo ra chữ ký điện tử đồng thời với việc mã hóa. Nó đánh dấu một sự tiến bộ vượt bậc của lĩnh vực mật mã học trong việc sử dụng khóa công cộng. RSA đang được sử dụng phổ biến trong thương mại điện tử và được cho là đảm bảo an toàn với điều kiện độ dài khóa đủ lớn. Nó là tính năng chính của nhiều giao thức bao gồm [SSH](https://quantrimang.com/bao-ve-ket-noi-internet-thong-qua-ssh-66513), OpenPGP, S/MIME và SSL/TLS. Ngoài ra, trình duyệt sử dụng RSA để thiết lập giao tiếp an toàn qua mạng không an toàn.

RSA vẫn rất phổ biến do độ dài khóa của nó. Một khóa RSA thường dài 1024 hoặc 2048 bit. Tuy nhiên, các chuyên gia bảo mật tin rằng không mất nhiều thời gian để crack RSA 1024 bit, do đó nhiều tổ chức phải chuyển sang khóa 2048 bit mạnh mẽ hơn.

***3.4.2 Mô tả thuật toán***



*Hình – Cấu trúc thuật toán RSA*

Giả sử bạn muốn nói với bạn mình một bí mật. Nếu bạn ở ngay bên cạnh họ, bạn chỉ cần thì thầm điều gì đó. Nếu bạn ở hai phía đối nhau của đất nước, điều đó sẽ không an toàn. Bạn có thể viết nó ra và gửi cho họ qua đường bưu điện hoặc sử dụng điện thoại. Nhưng cách này sẽ không an toàn và bất kỳ ai có động cơ đủ mạnh đều có thể chặn được tin nhắn.

Nếu bí mật đó đủ quan trọng, bạn không nên mạo hiểm viết nó ra. Gián điệp hoặc một nhân viên bưu điện lừa đảo có thể xem nó qua thư của bạn. Tương tự như vậy, ai đó có thể nghe trộm điện thoại của bạn mà bạn không biết và họ sẽ ghi lại mọi cuộc gọi mà bạn thực hiện.

Một giải pháp để ngăn chặn kẻ nghe trộm và truy cập nội dung tin nhắn là mã hóa nó. Về cơ bản, điều này có nghĩa là thêm một mã vào tin nhắn, nó sẽ làm cho tin nhắn thành một mớ hỗn độn. Nếu mã của bạn đủ phức tạp, thì những người duy nhất có thể truy cập vào thư gốc là những người có quyền truy cập vào mã.

Nếu bạn có cơ hội chia sẻ mã với bạn mình trước đó, thì một trong hai người có thể gửi tin nhắn được mã hóa bất cứ lúc nào. Nhưng nếu bạn không có cơ hội chia sẻ mã đó thì sao? Đây là một trong những vấn đề cơ bản của mật mã, đã được giải quyết bằng các sơ đồ mã hóa public-key (còn được gọi là mã hóa bất đối xứng) như RSA.

***3.4.3 An toàn và giải mã***

Mã hóa *RSA* thường được sử dụng kết hợp với các sơ đồ mã hóa khác. Và cho các chữ ký kỹ thuật số có thể chứng minh tính xác thực và tính toàn vẹn của một thông điệp. Nó thường không được sử dụng để mã hóa toàn bộ thư và file vì nó kém hiệu quả và tốn tài nguyên hơn so với mã hóa key đối xứng. Để làm cho mọi thứ hiệu quả hơn, một file thường sẽ được mã hóa bằng thuật toán key đối xứng. Sau đó key đối xứng sẽ được mã hóa bằng mã hóa RSA.

Theo quy trình này, chỉ người có quyền được truy cập vào *RSA private key* mới có thể giải mã key đối xứng. Nếu không thể truy cập key đối xứng, thì không thể giải mã file gốc. Phương pháp này có thể được sử dụng để bảo mật các thư và file mà không mất nhiều thời gian và tài nguyên.

Trong thực tế, mã hóa *RSA* sử dụng các số nguyên tố có độ lớn lớn hơn nhiều và sẽ có một vài trường hợp số phức tạp. Có một số khái niệm khác nhau mà bạn cần tìm hiểu trước khi chúng tôi giải thích về chúng. Chúng bao gồm hàm trapdoor, tạo số nguyên tố, hàm phi Carmichael. Và các quy trình riêng biệt liên quan đến việc tính toán các public key và private key được sử dụng trong quá trình mã hóa và giải mã.

**3.5 Thuật toán Twofish**

***3.5.1 Tổng quan về thuật toán***

*Twofish –* là [mật mã khối](https://wivi.wiki/wiki/Block_cipher)[khóa đối xứng](https://wivi.wiki/wiki/Symmetric-key_algorithm) với [kích thước khối](https://wivi.wiki/wiki/Block_size_(cryptography)) 128 [bit](https://wivi.wiki/wiki/Bit) và [kích thước khóa](https://wivi.wiki/wiki/Key_size) lên đến 256 bit. Nó là một trong năm ứng viên lọt vào vòng chung kết của [cuộc thi Tiêu chuẩn mã hóa nâng cao](https://wivi.wiki/wiki/Advanced_Encryption_Standard_process), nhưng nó không được chọn để tiêu chuẩn hóa. Twofish có liên quan đến mật mã khối trước đó là [Blowfish](https://wivi.wiki/wiki/Blowfish_(cipher)).

***3.5.2 Mô tả thuật toán***

Đặc điểm nổi bật của Twofish là sử dụng các [S-box](https://wivi.wiki/wiki/Substitution_box) phụ thuộc vào khóa được tính toán trước và một [lịch trình khóa](https://wivi.wiki/wiki/Key_schedule) tương đối phức tạp . Một nửa của khóa n-bit được sử dụng làm khóa mã hóa thực tế và nửa còn lại của khóa n-bit được sử dụng để sửa đổi thuật toán mã hóa (hộp S phụ thuộc khóa). Twofish vay mượn một số yếu tố từ các thiết kế khác; ví dụ, [biến đổi Hadamard giả](https://wivi.wiki/wiki/Pseudo-Hadamard_transform) (PHT) từ họ mật mã [SAFER](https://wivi.wiki/wiki/SAFER) . Twofish có [cấu trúc Feistel](https://wivi.wiki/wiki/Feistel_network) giống như [DES](https://wivi.wiki/wiki/Data_Encryption_Standard) . Twofish cũng sử dụng ma trận Có thể [tách biệt khoảng cách tối đa](https://wivi.wiki/wiki/MDS_matrix).

Khi nó đã được giới thiệu vào năm 1998, Twofish là hơi chậm hơn so với [Rijndael](https://wivi.wiki/wiki/Rijndael) (thuật toán chọn cho [Advanced Encryption Standard](https://wivi.wiki/wiki/Advanced_Encryption_Standard) ) cho 128-bit [phím](https://wivi.wiki/wiki/Key_(cryptography)) , nhưng hơi nhanh hơn cho các phím 256-bit. Kể từ năm 2008, hầu như tất cả các bộ xử lý AMD và Intel đều bao gồm khả năng tăng tốc phần cứng của thuật toán Rijndael thông qua tập [lệnh AES](https://wivi.wiki/wiki/AES_instruction_set) ; Các triển khai Rijndael sử dụng tập lệnh giờ đây có tốc độ nhanh hơn so với triển khai Twofish (phần mềm).

***3.5.3 An toàn và giải mã***

Mật mã *Twofish* chưa được [cấp bằng sáng chế](https://wivi.wiki/wiki/Patent) và việc [triển khai tham chiếu](https://wivi.wiki/wiki/Reference_implementation) đã được đặt trong [phạm vi công cộng](https://wivi.wiki/wiki/Public_domain) . Do đó, thuật toán Twofish miễn phí cho mọi người sử dụng mà không có bất kỳ hạn chế nào. Đây là một trong số ít mật mã có trong tiêu chuẩn [OpenPGP](https://wivi.wiki/wiki/OpenPGP) (RFC 4880). Tuy nhiên, Twofish ít được sử dụng rộng rãi hơn [Blowfish](https://wivi.wiki/wiki/Blowfish_(cipher)) , vốn đã tồn tại lâu hơn.

**CHƯƠNG IV - ỨNG DỤNG MÃ HOÁ THỰC TẾ**

**4.1 Tổng quan về ứng dụng**

***4.1.1 Mô tả ứng dụng***

**4.1.2 Công nghệ sử dụng**

Write

**4.1.3 Dữ liệu sử dụng**

Write

**4.1.4 Thuật toán của ứng dụng**

Write

**4.2 Phân tích và đánh giá kết quả**

Write

**TÀI LIỆU THAM KHẢO**

[1] <https://www.techsignin.com/cong-nghe/khai-niem-ve-ma-hoa/>

[2] <https://ssl.vn/ma-hoa-la-gi-cac-loai-ma-hoa-dang-duoc-su-dung-hien-nay.html>

[3] <https://www.cloudflare.com/learning/ssl/what-is-encryption/>

[4] [https://123docz.net//document/2194071-chuong-4-cac-ky-thuat-ma-hoa-thong-tin ts-hoang-xuan-dau.htm](https://123docz.net//document/2194071-chuong-4-cac-ky-thuat-ma-hoa-thong-tin%20ts-hoang-xuan-dau.htm)

[5] [https://www.thalesgroup.com/en/markets/digital-identity-and security/magazine/brief-history encryption#:~:text=In%20the%20early%201970s%3A%20IBM,papers%20on%2 encryption%20were%20classified](https://www.thalesgroup.com/en/markets/digital-identity-and%20security/magazine/brief-history%20encryption#:~:text=In%20the%20early%201970s%3A%20IBM,papers%20on%2 encryption%20were%20classified).

[6] <https://www.tnc.com.vn/5-thuat-toan-ma-hoa-pho-bien-ban-nen-biet.html>

[7] <https://tech.miichisoft.net/ma-hoa-thong-tin/>

[8] <https://thuvientvc.files.wordpress.com/2015/08/chuong1-tongquanatbmtt.pdf>

[9] <https://vnggroup.com.vn/ma-hash-la-gi/>

[10] <https://quantrimang.com/su-khac-biet-giua-ma-hoa-doi-xung-va-bat-doi-xung-168236>