**TRƯỜNG ĐẠI HỌC MỎ - ĐỊA CHẤT**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙞 🕮 🙜**

A picture containing food, room

Description automatically generated

**BÀI TẬP LỚN MÔN AN TOÀN VÀ BẢO MẬT HỆ THỐNG THÔNG TIN**

Giáo viên hướng dẫn: Dương Chí Thiện

Nhóm thực hiện: 4

Tên sinh viên: Lê Văn Dũng : 1821050920

Nguyễn Sỹ Thái: 1821050921

***Hà Nội, 10/2021.***

**Tóm Tắt**

Mật mã học đóng một vai trò rất lớn trong cuộc sống hàng ngày công nghệ cao của chúng ta và chúng ta phụ thuộc sâu sắc vào khoa học che giấu thông tin trong tầm nhìn đơn giản. Có rất nhiều cách để đạt được điều này, trong đó lý thuyết số đóng một vai trò rất lớn trong mật mã để đảm bảo rằng thông tin không thể dễ dàng phục hồi mà không có kiến thức đặc biệt.

Một trong những thuật toán mã hóa đáng tin cậy và an toàn nhất hiện nay là thuật toán RSA, cung cấp mã hóa và hiệu suất tuyệt vời bằng mật mã bất đối xứng, còn được gọi là mật mã khóa công khai.

Báo cáo này tập trung vào toán học đằng sau thuật toán, cùng với chức năng cốt lõi và thực hiện của nó. Ngoài ra, việc triển khai mã và quy trình mã hóa và giải mã được cung cấp chi tiết.

**Mục Lục**

[**1 Giới Thiệu** **4**](#_Toc24155)

[**2 Thuật Toán RSA** **5**](#_Toc24156)

[2.1 Sinh Khóa 5](#_Toc24157)

[2.2 Mã hóa và Giải mã 6](#_Toc24158)

[2.3 Ví Dụ áp dụng 7](#_Toc24159)

[**3 Giớ thiệu về lý thuyết** **10**](#_Toc24160)

[3.1 GCD and Số Nguyên Tố 10](#_Toc24161)

[3.2 Định lý nhỏ của Fermat’s 10](#_Toc24162)

[3.3 Số học mô-đun 10](#_Toc24163)

[3.4 Định lý Euler’s và Hàm Totient 11](#_Toc24164)

[3.5 Thuật toán Euclide mở rộng 11](#_Toc24165)

[**4 Giới thiệu về Mật mã học** **12**](#_Toc24166)

[4.1 Mật mã đối xứng và bất đối xứng 12](#_Toc24167)

[4.2 Một ví dụ thực tế 14](#_Toc24168)

[**5 Triển khai RSA** **17**](#_Toc24169)

[5.1 Mã nguồn 18](#_Toc24170)

[**6 Phần kết luận** **22**](#_Toc24171)

[**7 Tham khảo** **23**](#_Toc24172)

**Danh sách số liệu**

|  |  |
| --- | --- |
| 1 ASCII bảng cho bảng chữ cái. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 7 |
| 2 Pierre de Fermat . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 10 |
| 3 Quá trình mã hóa đối xứng . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 12 |
| 4 Quá trình mã hóa bất đối xứng. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 13 |
| 5 Diễn viên trong kịch bản này. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 14 |
| 6 Alice gửi tin nhắn cho Bob . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 15 |
| 7 Eve nắm bắt tin nhắn được mã hóa . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 16 |
| 8 Xem trước chương trình . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 17 |

# Giới thiệu

Đạt được mục tiêu mã hóa tin nhắn để ẩn thông tin trong tầm nhìn rõ ràng có thể được thực hiện theo nhiều cách. Mật mã học đã tồn tại hàng ngàn năm và sự tiến hóa và khoa học của mật mã đã được cải thiện vượt quá trong vài thập kỷ qua. Có nhiều thuật toán có sẵn có thể được sử dụng để mã hóa và giải mã các tin nhắn, trong đó một trong những thuật toán được sử dụng nhiều nhất được gọi là RSA.

Thuật toán được giới thiệu bởi ba nhà nghiên cứu vào năm 1976, được đặt tên là Ronald Rivest, Adi Shamir và Leonard Adleman, và dựa trên các thông điệp mã hóa bằng cách sử dụng lũy ​​thừa mô-đun và chia sẻ các khóa công khai và khóa riêng. Trong thuật toán mã hóa đối xứng, có một khóa bí mật được sử dụng để cả mã hóa và giải mã dữ liệu. Mã hóa đối xứng tương đối nhanh, nhưng phương pháp mã hóa này có thể gây ra sự cố về việc phân phối khóa bí mật, vì nó cần được sử dụng cho cả mã hóa và giải mã dữ liệu.

Không giống như các thuật toán đối xứng, chẳng hạn như ví dụ AES, các thuật toán khóa công khai yêu cầu tính toán của cả khóa công khai và khóa riêng. Điều khiến RSA trở nên đặc biệt so với các thuật toán mã hóa khác là các khóa này phải được tính toán bằng toán học và không phải là số ngẫu nhiên được tạo.

Cùng với mật mã đối xứng, có một phương pháp khác gọi là mật mã bất đối xứng, nơi có khóa riêng và khóa công khai được sử dụng trong quá trình mã hóa. Khóa công khai rõ ràng là công khai và không cần phải giữ bí mật, trong khi khóa riêng phải được giữ giữa các bên liên quan của việc truyền tin nhắn. Vì vậy, bất cứ ai cũng có thể mã hóa một thông báo bằng khóa chung, nhưng chỉ những người biết khóa riêng có thể giải mã tin nhắn. Nếu không nó sẽ vẫn được mã hóa. Phương pháp này giải quyết một số thách thức chia sẻ khóa và giúp chia sẻ các phím dễ dàng hơn.

Phần thế hệ quan trọng của thuật toán RSA là khá trung tâm và quan trọng, và đây là thứ còn thiếu trong hầu hết các thuật toán khóa đối xứng, nơi phần tạo khóa không thực sự phức tạp theo tính toán toán học.

RSA ngày nay được sử dụng trong một loạt các trình duyệt web, trò chuyện và dịch vụ email, VPN và các kênh truyền thông khác. Nó thường được sử dụng đơn giản vì mọi người tin tưởng vào thuật toán để cung cấp mã hóa đủ tốt cho mục đích của họ, và nó đã được chứng minh là an toàn.

# Thuật toán RSA

RSA (Rivest-Shamir-Adleman) là một thuật toán mã hóa bất đối xứng được sử dụng để mã hóa và giải mã thông điệp của các máy tính hiện đại. Các quốc gia bất đối xứng có hai khóa khác nhau được sử dụng trong quá trình mã hóa và giải mã, cũng được gọi là mật mã khóa công khai. Điều này chỉ đơn giản là do một trong hai khóa có thể được cung cấp cho bất cứ ai mà không cần khai thác bảo mật của thuật toán.

Thuật toán RSA liên quan đến cả khóa riêng và khóa công khai. Khóa công khai có thể được biết và xuất bản cho bất kỳ ai, vì nó được sử dụng để mã hóa các tin nhắn từ bản rõ về bản mã. Tuy nhiên, các tin nhắn được mã hóa với khóa chung cụ thể này chỉ có thể được giải mã với khóa riêng tương ứng. Quá trình tạo khóa của thuật toán RSA là thứ khiến nó trở nên an toàn và đáng tin cậy ngày nay, vì nó chứa một mức độ phức tạp cao so với các thuật toán mã hóa khác.

## Tạo khóa

Không giống như các thuật toán đối xứng, chẳng hạn như ví dụ AES, các thuật toán khóa công khai yêu cầu tính toán của cặp (kpublic, kprivate). Điều khiến RSA trở nên đặc biệt so với các thuật toán mã hóa khác là các khóa này phải được tính toán bằng toán học và không phải là số ngẫu nhiên được tạo.

Phần thế hệ quan trọng của thuật toán RSA là khá trung tâm và quan trọng, và đây là thứ còn thiếu trong hầu hết các thuật toán khóa đối xứng, nơi phần tạo khóa không thực sự phức tạp theo tính toán toán học.

Quá trình thế hệ quan trọng của thuật toán RSA bao gồm năm bước:

1. Chọn hai số nguyên tố lớn (P và Q)
2. Tính toán *n* = *p* · *q*
3. Tính toán *φ*(*n*) = (*p* − 1) · (*q* − 1)
4. Chọn mốt số nguyên *e* sao cho 1 *< e < φ*(*n*), và:
   1. Đảm bảo rằng *gcd*(*e,φ*(*n*)) = 1
   2. Đảm bảo rằng *e* và *φ*(*n*) là nguyên tố
5. Tính toán số nguyên *d*, sao cho *d* = *e*−1 mod *φ*(*n*)

Sau khi hoàn thành năm bước này, chúng Chúng em đã tạo ra hai khóa bất đối xứng có thể được sử dụng để mã hóa và giải mã thêm trong quá trình này: khóa công khai bao gồm n và e, trong khi khóa riêng bao gồm d.

*Kpublic* = (*n,e*) *Kprivate* = (*d*)

Thông thường sử dụng số lượng lớn trong quá trình tạo cho p và q, thường là ít nhất 512 bit, tạo ra n 1024 bit (n = p · q). Điều này làm tăng mức độ phức tạp và làm cho quá trình phân tích mật mã trở nên khó khăn hơn đáng kể về các cuộc tấn công của lực lượng bruceforce.

## Mã hóa và giải mã

Sau khi tính toán tất cả các biến cần thiết cho việc tạo khóa, đã đến lúc mã hóa và giải mã tin nhắn bằng thuật toán. Điều này tất nhiên được đưa ra thực tế là Kpublic đã được tạo ra, và bao gồm n và e. Công thức này rất đơn giản cho cả quá trình mã hóa và giải mã, trong đó nói rằng:

**Mã hóa:** *c* ≡ *me* (mod *n*), trong đó m là thông điệp trong văn bản thuần túy.

**Giải mã:** *m* ≡ *cd* (mod *n*), trong đó c là mật mã.

Bất cứ ai muốn mã hóa thư đều cần sử dụng khóa công khai của thư đọc thuộc lòng, để đảm bảo rằng thư chỉ có thể giải mã được bởi đúng cá nhân để nó chỉ giải mã bằng một khóa riêng tư cụ thể. Những người đọc thuộc lòng chia sẻ chìa khóa công khai, trong khi vẫn giữ bí mật khóa riêng tư.

Người gửi sau đó muốn gửi một tin nhắn M, được chuyển đổi thành một số nhỏ hơn n và điều này được thực hiện bởi một giao thức có thể đảo ngược được gọi là sơ đồ đệm. Thư được tính toán thành một mật mã được mã hóa, cuối cùng được gửi đến receipent.

Đề án đệm được sử dụng trong quá trình mã hóa khá quan trọng và không có kế hoạch này sẽ có một số vấn đề. Sơ đồ phần đệm đảm bảo rằng không có giá trị nào của tin nhắn không an toàn, chẳng hạn như ví dụ các giá trị M = 0 hoặc M = 1 sẽ tương ứng tính toán các bản mã bằng 0 hoặc 1, do các thuộc tính của Lối thừa. Khi số mũ được sử dụng trong quy trình tạo khóa là nhỏ, điều này có thể gây ra kết quả không phải là mô-đun của Chúng em ít hơn mô đun n. Điều này có nghĩa là các bản mã có thể bị hỏng và được giải mã một cách dễ dàng bằng cách tính toán gốc E-th của bản mã mà không nhất thiết phải liên quan đến mô đun.

Ví dụ: khi mã hóa một văn bản với giá trị số là 0, nó sẽ mã hóa dưới dạng m = 0, sau đó một lần nữa tính toán bản mã c = 0, không lo ngại về các giá trị của n và e. Điều tương tự cũng xảy ra với giá trị số là 1, tạo ra giá trị 1 trong bản mã. Điều này tạo ra một mô hình không an toàn, có thể được phân tích bởi những kẻ tấn công và dễ dàng được giải mã sau khi đạt được một số kiến ​​thức về quá trình mã hóa.

Để tránh những vấn đề như vậy trong thuật toán, người ta thường thực hiện một lớp đệm ngẫu nhiên vào tin nhắn trước khi mã hóa xảy ra. Điều này là để đảm bảo rằng thư không chứa một số giá trị không an toàn và nội dung mật mã được mã hóa chứa một số giá trị đệm tạo ra một nguồn mã lớn hơn. Điều này làm tăng mức độ phức tạp của mã hóa và rất có thể sẽ làm cho một cuộc tấn công từ điển khó thành công hơn.

Khi tin nhắn đến phía người nhận của kênh giao tiếp, mật mã được giải mã bằng khóa riêng trong quy trình sau: m ≡ cd (mod n), trong đó c là mật mã.

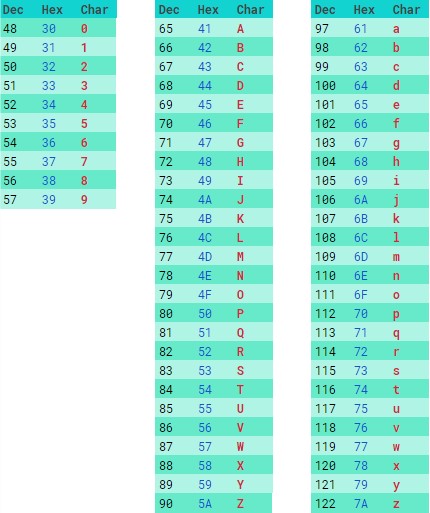
## Một ví dụ thực tế

Có hai diễn viên trong kịch bản này: Alice và Bob. Alice muốn gửi tin nhắn cho Bob và muốn mã hóa tin nhắn bằng mã hóa RSA. Bob chọn hai số nguyên tố, p = 101 và q = 113. Biến sau được tính bằng hai số này, n = p · q = 11413.

Bước tiếp theo là tính giá trị Phi (φ), sao cho φ(n) = (p−1)· (q−1). Điều này mang lại cho φ(n) = 100·112 = 11200. Với thực tế là 11200 = 26527, chúng ta có thể tính toán một số nguyên e sẽ được sử dụng thêm như một số mũ trong quá trình mã hóa, miễn là số mũ không chia hết cho các số 2, 5 hoặc 7. Điểm này được xác minh là một phần của quy trình tạo khóa, trong đó (e,φ(n)) = 1.

Bob chọn số mũ, nói ví dụ E = 3. Thông thường sẽ là khôn ngoan để chọn một số mũ lớn hơn, về mặt bảo mật, nhưng chúng Chúng em sẽ sử dụng một số nhỏ trong kịch bản này để đơn giản. Sử dụng số phận này, Bob có thể tạo khóa riêng, sao cho D = E-1 MOD 11200 = 6597 = KPRIVATE (đó là phím bí mật). BOB xuất bản các biến khóa công khai, N và E, có thể được sử dụng thêm để mã hóa các thông báo được văn bản. Quá trình tạo quan trọng đã hoàn tất và quá trình truyền tin nhắn tiếp tục.

Bước tiếp theo là phần mã hóa thực tế, trong đó mật mã được thiết lập bằng toán học. Trong trường hợp này, thông báo là "USN", trò chuyện với các số "85 83 78" khi chuyển đổi từ văn bản sang thập phân bằng bảng mã ASCII, được hiển thị trong Hình 1.



Hình 1: bảng ASCII cho bảng chữ cái

Vì 8583783 là một số lượng rất lớn, thuật toán mã hóa từng ký tự riêng biệt để "U" được mã hóa đầu tiên, tiếp theo là "S" và sau đó là "N" cuối cùng. Alice muốn gửi mật mã cho Bob và tính toán c = 853 mod 11413 = 9236. Ký tự đầu tiên đã được mã hóa và quá trình tương tự xảy ra một lần nữa cho các ký tự còn lại của văn bản thuần túy.

**U:** 853 mod 11413 = **9236**

**S:** 833 mod 11413 = **1137**

**N:** 783 mod 11413 = **6619**

Toàn bộ văn bản thuần túy đã được mã hóa và bản mã cuối cùng là 9236 1137 6619. Mật mã được gửi cho Bob và anh ta giải mã tin nhắn bằng cùng một thuật toán, tiếp theo là cùng một khóa công khai và khóa riêng mà Alice sử dụng để mã hóa tin nhắn. Bob chỉ có thể giải mã tin nhắn vì anh ta nhận thức được khóa riêng tư, nếu không nó sẽ không thể chuyển đổi mật mã trở lại văn bản thuần túy về mặt phân tích mật mã và tấn công .

Bob rõ ràng biết KPRIVate khóa riêng = 6597 và sử dụng nó để giải mã tin nhắn mà anh ta nhận được từ Alice. Kênh liên lạc giữa Alice và Bob nhận thức được thực tế rằng mỗi khối chứa nhiều chữ số, trong đó mỗi ký tự của bản rõ được mã hóa thành một số có bốn chữ số trong bản mã. Sử dụng công thức giải mã, BOB tính toán p = 92366597 mod 11413 = 85. Quá trình tương tự xảy ra một lần nữa cho các khối còn lại của bản mã, sao cho:

**9236:** 92366597 mod 11413 = 85 = **U**

**1137:** 11376597 mod 11413 = 83 = **S**

**6619:** 66196597 mod 11413 = 78 = **N**

Bản mã đã được giải mã thành công và Bob cuối cùng cũng có thể đọc văn bản. Để tóm tắt các bước của quá trình mã hóa và giải mã thông điệp:

1. Bob chọn hai số nguyên tố P và Q, và tính toán *n* = *p* · *q*
2. Bob chọn một số mũ *e*
3. Bob chia sẻ n và e và làm cho các biến này công khai, trong khi p, q và d được giữ bí mật.
4. Alice mã hóa tin nhắn và gửi bản mã đến Bob.
5. Bob giải mã bản mã bằng cách sử dụng khóa riêng và đọc bản rõ.

Đó là khá nhiều cách thuật toán RSA hoạt động. Bảo mật trong thuật toán này phụ thuộc vào thực tế rằng đây là chức năng một chiều và cách duy nhất để giải mã bản mã là với kiến ​​thức thích hợp về hệ số n = p · q. Bản mã hóa vẫn được mã hóa mà không có kiến ​​thức này và thực sự không có cách nào để giải mã nó, miễn là các số nguyên tố đủ lớn (như trong ít nhất 512 bit). Đó là lý do tại sao thuật toán RSA được coi là một trong những thuật toán an toàn và đáng tin cậy nhất hôm nay và hy vọng sẽ vẫn theo cách này trong một thời gian dài.

# Giới thiệu về lý thuyết số

Chúng em sẽ giới thiệu một số khái niệm lý thuyết và mật mã số được sử dụng trong thuật toán RSA, như một phần giới thiệu toán học ngắn gọn về thuật toán và chức năng cốt lõi của nó.

## GCD và số nguyên tố

Số nguyên tố có vai trò rất lớn trong thuật toán RSA, và các số như 2, 3, 5, 5, 6, 11, 13, 17, ..., là những con số tự nhiên lớn hơn 1 không thể được biểu thị dưới dạng sản phẩm của tự nhiên nhỏ hơn số. Mỗi định nghĩa, một nguyên tố là một số nguyên lớn hơn 1 là chia hết bằng cách không có số nguyên dương nào khác ngoài 1 và chính nó. Số nguyên tố quay trở lại thời cổ đại, nơi nó được biết đến hàng ngàn năm trước rằng có vô số nguyên tố hiện có.

Các số nguyên tố ngày nay rất cần thiết trong các hệ thống mật mã hiện đại, và bao gồm nhiều tính chất quan trọng trong mật mã. Điều quan trọng là phải nhớ rằng số 1 không phải là số nguyên tố. Số nguyên tố được sử dụng cụ thể trong quy trình tạo khóa của thuật toán RSA và thực sự là những gì toàn bộ thuật toán dựa trên.

Số ước phổ thông phổ biến nhất (GCD) của hai hoặc nhiều số nguyên là số nguyên dương lớn nhất phân chia từng số nguyên. Ví dụ, GCD 53 và 59 là 1. Nếu GCD (A, B) = 1, chúng Chúng em nói rằng A là Coprime to b. Ví dụ: GCD (3,59) = 1, chỉ ra rằng 3 là Coprime đến 59, cũng có nghĩa là 59 là Coprime đến 3.

Đối với số lượng lớn, ước số chung lớn nhất có thể nhanh chóng được tính bằng cách sử dụng thuật toán EUCLIDEAN. Thuật toán cho thấy D = GCD (A, B) luôn có thể được viết dưới dạng D = SA + TB, trong đó S và T là các số nguyên dương. Tính toán số chung chung lớn nhất của hai số nguyên trực tiếp bằng cách sử dụng trực tiếp hệ số nguyên tố của các số nguyên này được chứng minh là khá kém hiệu quả. Cách tính toán số nguyên tố này là tốn thời gian và do đó thuật toán Euclide thường được sử dụng cho các số lượng lớn, vì nó cung cấp một cách điện toán thanh lịch hơn so với ước số chung lớn nhất.

Nếu chúng ta có thể chỉ ra rằng các quy định chung của A và B giống như các phân chia chung của B và R, nó đã được chứng minh rằng GCD (A, B) = GCD (B, R), do thực tế là Cả hai cặp này phải bao gồm các ước phối chung lớn nhất. Để hiển thị một ví dụ về cách phân chia chung lớn nhất của hai số nguyên có thể được tính bằng thuật toán EUCLIDEAN:

662 = 414 · 1 + 248

414 = 248 · 1 + 166

248 = 166 · 1 + 82

166 = 82 · 2 + 2

82 = 2 · 41

Các tính toán chứng minh rằng ước số chung lớn nhất của (414, 662) = 2, vì 2 là phần còn lại cuối cùng.

## Định lý nhỏ của Fermat

Một trong những định lý cơ bản của lý thuyết số được sử dụng trong thuật toán RSA là định lý nhỏ của Fermat. Pierre de Fermat là một trong những người đóng góp lớn nhất của toán học và được biết đến với các lý thuyết số của mình. Ông đã đóng góp với một định lý rất nổi tiếng trong lý thuyết số, được gọi là Định lý cuối cùng của Fermat.



Hình 2: Pierre de Fermat

Định lý này nói rằng, đối với bất kỳ số nguyên N ≥ 3, phương trình XN + YN = Zn không có giải pháp với X, Y và Z tất cả các số nguyên dương. Định lý nhỏ của Fermat rất trung tâm trong thuật toán RSA, và nói rằng AP A (MOD P) cho mỗi số nguyên A và cho mỗi Prime P.

Nếu P là một số nguyên tố, thì 2p-1 ≡ 1 (mod p). Ví dụ: để chứng minh rằng số 35 không phải là nguyên tố, nó có thể được hiển thị bằng bình phương liên tiếp:

24 ≡ 16

28 ≡ 256 ≡ 11

216 ≡ 121 ≡ 16

232 ≡ 256 ≡ 11

Do đó, có thể kết luận rằng 234 ≡ 232,22 ≡ 11.4 9 6≡ 1 (MOD 35). Nói cách khác, nó không phải là một số nguyên tố. Đây là một phương pháp nhanh chóng và hiệu quả để chứng minh rằng 35 là hỗn hợp mà không nhất thiết phải tìm các yếu tố của nó. Định lý của Fermat sẽ được đề cập nhiều lần trong suốt quá trình giải thích về thuật toán RSA, vì nó đóng góp với nhiều tính chất quan trọng trong mật mã hiện đại.

## Số học mô-đun.

Thông thường trong lý thuyết số chúng Chúng em chỉ quan tâm đến phần còn lại của một số nguyên khi số nguyên được chia cho một số nguyên dương khác. Vì chúng ta thường chỉ quan tâm đến những người còn lại, có một ký hiệu đặc biệt cho họ là tốt. Ký hiệu A Mod M được sử dụng để đại diện cho phần còn lại khi một số nguyên A được chia cho một số nguyên dương m.

Một ký hiệu liên quan khác thường được sử dụng, chỉ ra rằng hai số nguyên có cùng một phần còn lại khi các số nguyên được chia cho một số nguyên dương m khác. Điều này chỉ ra rằng nếu a và b là cả số nguyên và m là một số nguyên dương, sau đó là ≡ b (mod m) nếu và chỉ khi mod m = b mod m.

Các phương trình số học mô-đun này sẽ được sử dụng nhiều lần trong thuật toán RSA, vì các phương pháp mã hóa và giải mã phụ thuộc vào toán tử modulo.

## Định lý Euler và Hàm Totient

Earlier the Fermat’s little theorem was introduced where the theorem says that if *p* is a prime number and *a* is not a multiple of *p*, then *ap*−1 = 1 (mod *p*). However Euler’s generalization of Fermat’s theorem indicates that if *a* is relatively prime to *m*, then *aφ*(*m*) = 1 (mod *m*), where *φ*(*m*) is called a totient function by Euler.

Cái gọi là hàm totient này sẽ đếm số nguyên dương tương đối nguyên tố và nhỏ hơn m. Định lý Euler được sử dụng trong quá trình mã hóa RSA, trong đó hai số nguyên tố enourmous p, q được sử dụng, trong khi p và q được giữ riêng tư và n = p · q được giữ công khai. Tuy nhiên, khóa riêng d giải quyết cho khóa công khai e, sao cho de = 1 (mod φ(n)). Vì các giá trị p và q được biết đến, có thể tính toán φ(n) = (p − 1) · (q − 1), cùng với việc tính toán khóa công khai *e*.

Định lý Euler lại có ích khi ai đó muốn gửi tin nhắn m, sử dụng khóa công khai e, sao cho Chúng em (mod n). Theo đó, việc tính toán thư được mã hóa cũng xảy ra theo định lý Euler, sao cho (me)d = mφ(n) = m (mod n).

Có nhiều trường hợp sử dụng cho định lý Euler và hàm totient trong lý thuyết số, vì nó cũng thường được sử dụng trong thử nghiệm nguyên thủy, nơi nó kiểm tra và chứng minh nếu một số không phải là số nguyên tố. Hàm totient, hay hàm Phi, thường xảy ra trong các ứng dụng thực tế và được sử dụng rất nhiều trong mật mã hiện đại.

## Thuật toán Euclide mở rộng

Thuật toán Euclide đã được đề cập trước đó, nơi nó được sử dụng để tính toán các chia chung lớn nhất, và bây giờ có một thuật toán Euclide mở rộng, về cơ bản là thuật toán Euclide chạy ngược lại. Như một sự mở rộng đối với thuật toán Euclide, nó tính toán các hệ số nhận dạng của Bezout, là các số nguyên x và y sao cho ax + by = gcd(a,b). Thuật toán này đặc biệt hữu ích khi các số nguyên a và b là đồngprome và thuật toán Euclide mở rộng được sử dụng rộng rãi trong mật mã học hiện đại, cụ thể là thuật toán RSA nơi nó tính toán nghịch đảo nhân mô-đun trong các phương pháp mã hóa và giải mã khóa công khai.

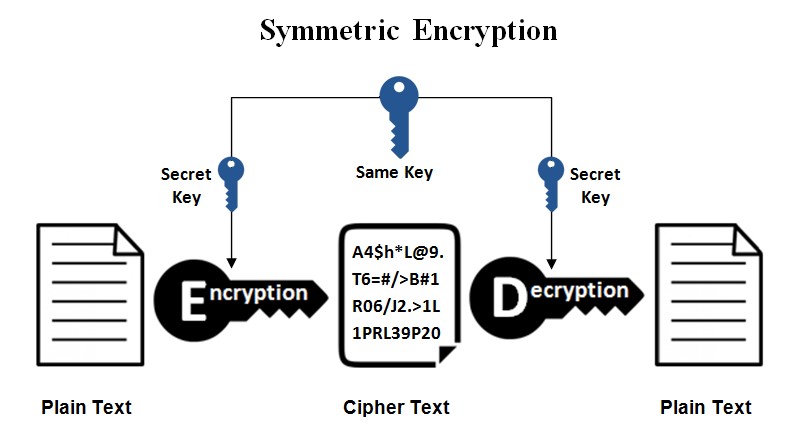
Có thể tìm thấy các số nguyên x và y bằng cách đảo ngược các bước trong thuật toán Euclide. Ý tưởng ban đầu là bắt đầu với chia phổ biến lớn nhất và đệ quy tự làm việc ngược lại, nơi cuối cùng nó tìm thấy kết quả cho x và y, là hai số nguyên.

# Giới thiệu về Mật mã học

## Mật mã đối xứng và bất đối xứng

Trong thuật toán mã hóa đối xứng có một khóa bí mật được sử dụng để mã hóa và giải mã dữ liệu. Mã hóa đối xứng tương đối nhanh, nhưng phương pháp mã hóa này có thể gây ra sự cố về mặt phân phối khóa bí mật, vì nó cần được sử dụng cho cả mã hóa và giải mã dữ liệu.

Nếu Alice gửi một tin nhắn được mã hóa đối xứng cho Bob, cô cũng cần thông báo cho anh ta về khóa bí mật, để anh ta có thể sử dụng nó để giải mã tin nhắn. Điều này có thể dẫn đến một số thách thức không cần thiết. Có nhiều thuật toán mã hóa đối xứng hiện có, chẳng hạn như mã hóa Caesar, AES và DES. Một số thuật toán này vẫn được sử dụng ngày nay và có thể được dựa vào, vì mã hóa đối xứng là an toàn và đủ nhanh cho hầu hết các mục đích.

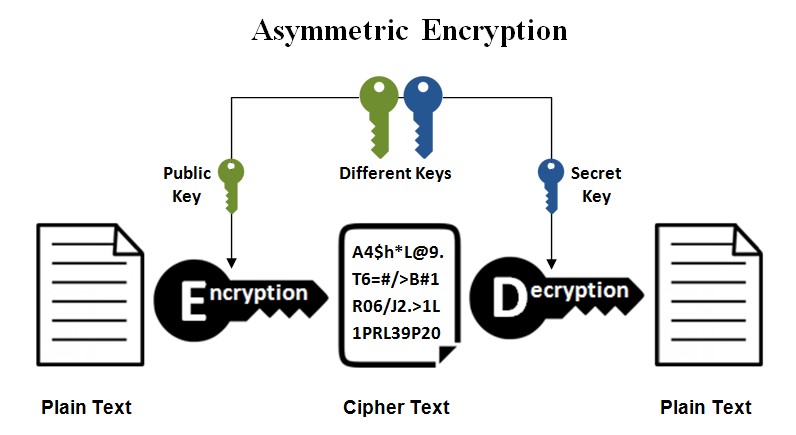


Hình 3: Quy trình mã hóa đối xứng

Cùng với mật mã đối xứng, có một phương pháp khác được gọi là mật mã bất đối xứng, trong đó có khóa riêng và khóa công khai được sử dụng trong quá trình mã hóa. Khóa công khai rõ ràng là công khai và không cần phải giữ bí mật, trong khi khóa riêng phải được giữ giữa các bên liên quan của việc truyền tải thông điệp. Vì vậy, bất cứ ai cũng có thể mã hóa thư bằng khóa công khai, nhưng chỉ những người biết khóa riêng mới có thể giải mã thư. Nếu không, nó sẽ vẫn được mã hóa. Phương pháp này giải quyết một số thách thức chia sẻ khóa và giúp chia sẻ khóa dễ dàng hơn. Ví dụ về thuật toán này là RSA, mã hóa ElGamal và thuật toán Diffie-Hellman.

Nếu chúng ta so sánh mã hóa đối xứng và bất đối xứng, chúng ta có thể thấy rằng bất đối xứng chậm hơn một chút do thực tế là nó xử lý hai khóa và phải mất thêm thời gian để xử lý điều này.

Điều quan trọng cần lưu ý là cả mã hóa đối xứng và bất đối xứng đều an toàn và không thể cạnh tranh hoặc được so sánh trực tiếp, bởi vì cả hai đều phục vụ một mục đích lớn cho các trường hợp sử dụng khác nhau. Một số cá nhân có thể thích đối xứng vì nó đơn giản và cung cấp đủ bảo mật cho mục đích của họ, trong khi những người khác thích bất đối xứng do phương pháp phân phối chính của nó. Cả hai loại mã hóa đều có ưu và nhược điểm, ví dụ, mã hóa đối xứng nhanh hơn bất đối xứng, trong khi nó yếu về mặt phân phối khóa cho các bên liên quan.



Hình 4: Quy trình mã hóa bất đối xứng

Tuy nhiên, mã hóa bất đối xứng giải quyết thách thức phân phối chính, nhưng nó có nhược điểm là chậm so với mã hóa đối xứng. Rốt cuộc, cả hai phương pháp mã hóa đều được sử dụng trong công nghệ hiện đại và cả hai đều phục vụ một mục đích lớn về tính bảo mật và tính toàn vẹn.

Một ví dụ thực tế về mật mã bất đối xứng:

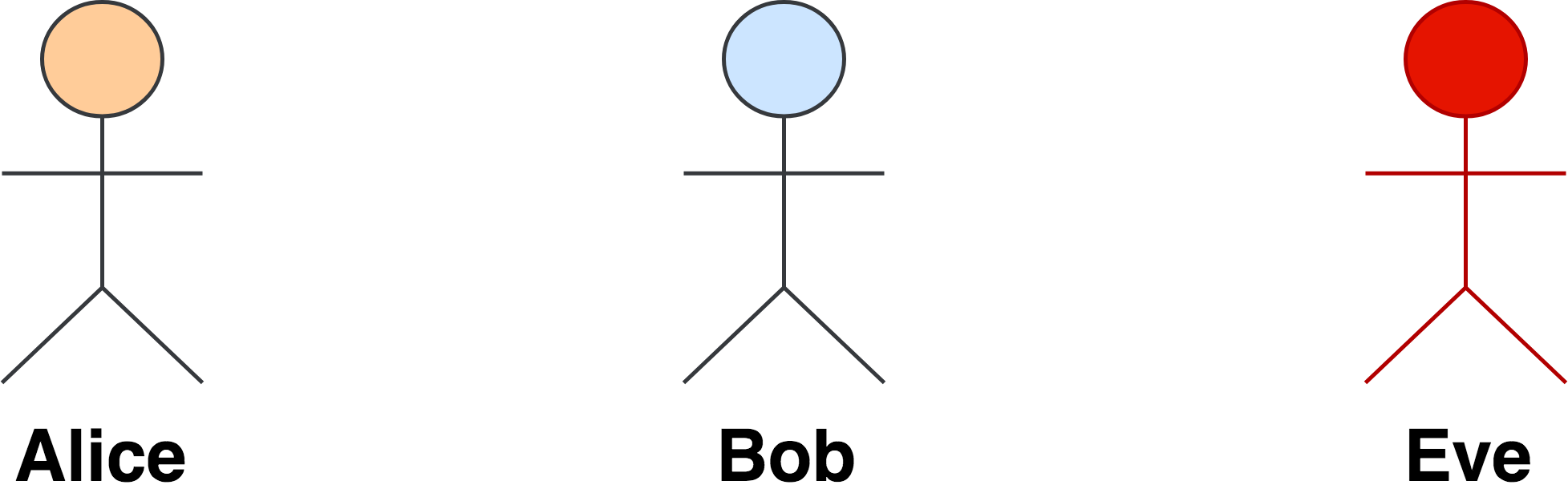
1. Một máy khách như trình duyệt web gửi khóa công khai của nó đến máy chủ và đưa ra yêu cầu về một số dữ liệu.
2. Máy chủ phản hồi trình duyệt web bằng cách mã hóa dữ liệu bằng cùng một khóa công khai.
3. Máy chủ gửi dữ liệu được mã hóa này trở lại máy khách.
4. Máy khách nhận dữ liệu được mã hóa này và giải mã nó bằng khóa riêng.

Vì quá trình này không đối xứng, không ai khác ngoại trừ máy khách (trình duyệt web) có thể giải mã dữ liệu, ngay cả khi cá nhân bên thứ ba có quyền truy cập vào khóa công khai. Cùng một khái niệm này sẽ được giới thiệu trong thuật toán RSA là tốt.

## Một ví dụ thực tế

CIA Triad là một mô hình bảo mật là viết tắt của bảo mật, tính toàn vẹn và tính sẵn có, và là trọng tâm chính trong bảo mật thông tin để cân bằng sự bảo vệ thông tin trực tuyến. CIA Triad rất cơ bản trong an toàn mạng và ba khái niệm nên được đảm bảo trong bất kỳ hệ thống an toàn nào để tránh các vi phạm dữ liệu nhạy cảm và mất thông tin cá nhân trong công nghệ hiện đại ngày nay.

Chúng em sẽ chứng minh các khái niệm của CIA thông qua một ví dụ thực tế bằng cách sử dụng hai diễn viên: Alice and Bob. Thực sự cũng có một diễn viên thứ ba, người nghe lén có tên Eve. Chúng em sẽ sử dụng ba cái tên này trong quá trình minh họa ví dụ thực tế này. Alice muốn gửi thông điệp "USN Kongsberg là tốt nhất!" đến Bob. Cả hai đều đang sử dụng một ứng dụng nhắn tin an toàn trên điện thoại của họ và từ thời điểm Alice gửi tin nhắn đến thời điểm Bob nhận được nó, có một số bước xử lý trong nền về tính bảo mật, tính toàn vẹn và tính khả dụng để đảm bảo rằng tin nhắn được chuyển đến Bob mà không có bất kỳ vi phạm hoặc lỗi bảo mật nào.



Hình 5: Actors in this scenario

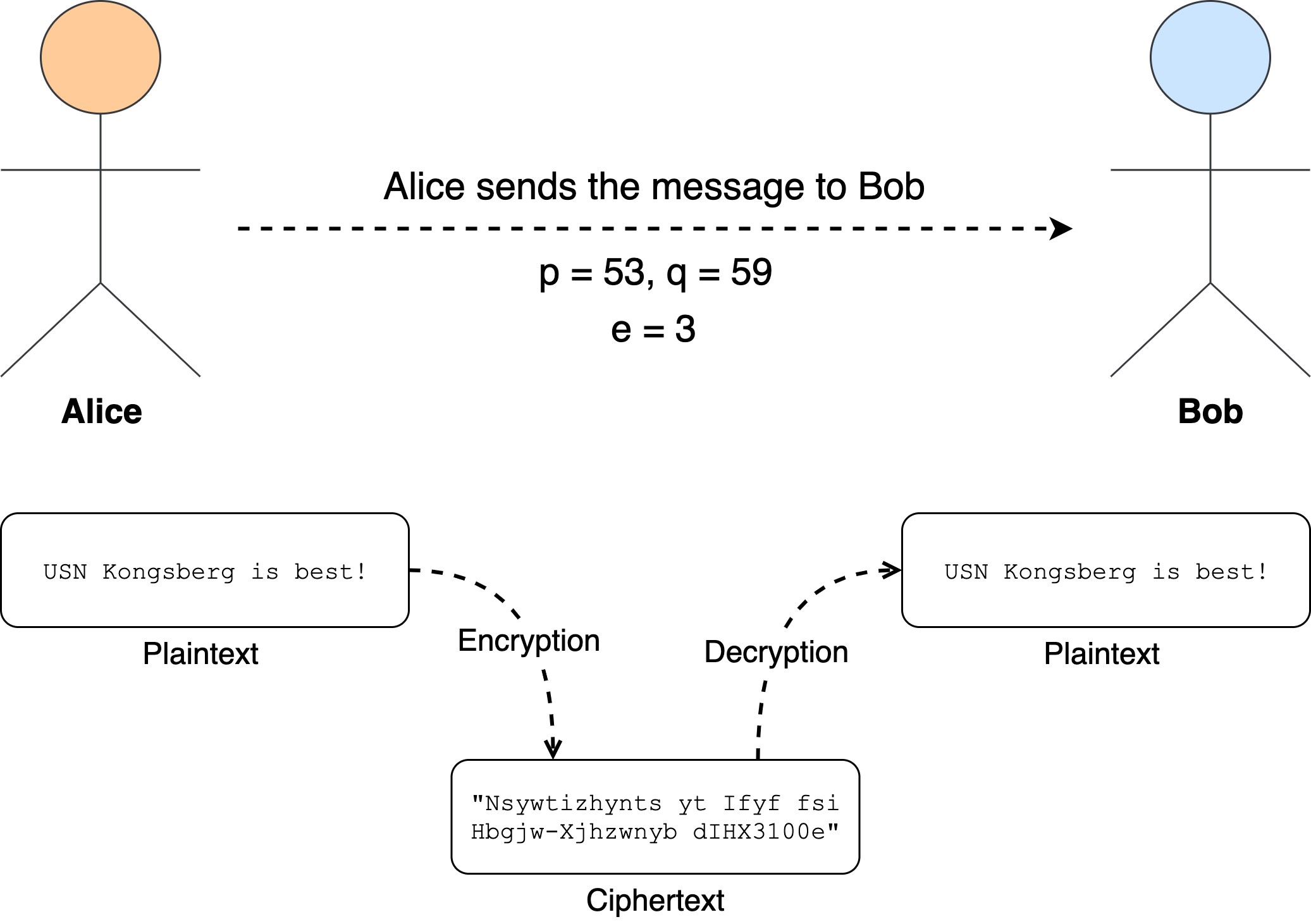
Mục tiêu ở đây là đảm bảo rằng thông điệp mà Alice gửi được gửi đến Bob một cách an toàn, mà không có ai khác can thiệp vào việc truyền tin nhắn. Điều này được thực hiện bởi mật mã học, nơi giao tiếp được mã hóa và bảo vệ, để chỉ Alice và Bob mới có thể hiểu thông điệp đang được gửi. Vì vậy, ngay cả khi ví dụ Eve quản lý để can thiệp vào việc truyền tin nhắn, nó được mã hóa và không thể đọc được dưới dạng văn bản thuần túy.

Có nhiều cách khác nhau trong mật mã để thực hiện mã hóa, cả phương pháp bảo mật và yếu. Chúng ta thường nói về hai thuật toán trong mật mã: thuật toán đối xứng và bất đối xứng. Một thuật toán về cơ bản là một công thức hoặc một quy trình để giải quyết một vấn đề cụ thể, trong trường hợp này là mã hóa dữ liệu.

Trong các thuật toán đối xứng, cả người gửi và người nhận, Alice và Bob, đều phải có cùng một khóa và cùng một thuật toán xử lý. Khóa đối xứng này thường được gọi là khóa bí mật hoặc khóa riêng, được bảo vệ khỏi bất kỳ ai khác ngoại trừ người gửi và người nhận tin nhắn được mã hóa. Nếu khóa riêng được tiết lộ cho Eve, mã hóa an toàn bị xâm phạm và tin nhắn có thể được giải mã bởi bất kỳ ai có thể có quyền truy cập vào khóa.

Tuy nhiên, trong các thuật toán bất đối xứng có hai khóa: khóa công khai và khóa riêng. Alice sử dụng khóa công khai để mã hóa tin nhắn và Bob sử dụng khóa riêng để giải mã nó. Không nguy hiểm nếu khóa công khai thực sự công khai, nhưng khóa riêng tuy nhiên phải được giữ riêng tư và chỉ chủ sở hữu mới có thể biết khóa riêng là gì. Nếu cả khóa công khai và khóa riêng tư được công bố cho bất kỳ ai khác ngoại trừ Alice và Bob, tin nhắn được mã hóa sẽ không còn an toàn và nó có thể được giải mã bất cứ lúc nào.

Trong trường hợp này, Chúng em sẽ sử dụng thuật toán RSA để chứng minh cách tin nhắn đang được mã hóa và giải mã giữa Alice và Bob. Để hoàn thành nhiệm kỳ đầu tiên trong bộ ba CIA, bảo mật, chúng ta cần mã hóa thông điệp Alice gửi cho Bob để đảm bảo rằng tin nhắn được ẩn khỏi bất kỳ ai không được phép đọc nó. Alice viết thông điệp bằng văn bản thuần túy và gửi nó cho Bob. Ứng dụng nhắn tin chuẩn bị tin nhắn bằng cách mã hóa nó bằng RSA.

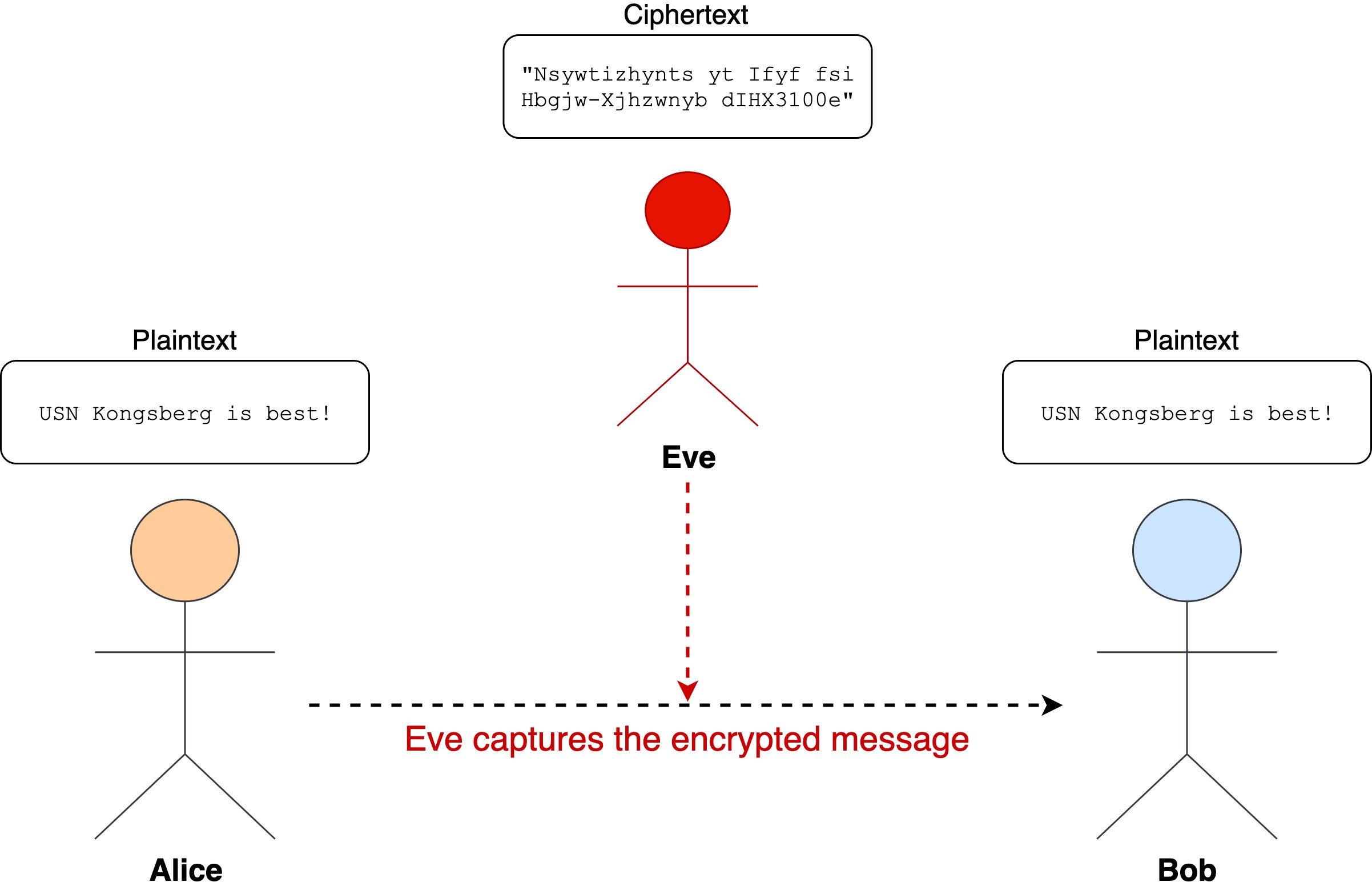


Hình 6: Alice gửi thư đến Bob

Ngay sau khi Bob nhận được tin nhắn, ứng dụng di động giải mã mật mã bằng cùng một thuật toán mà Alice đã sử dụng để mã hóa tin nhắn. Người nhận biết các giá trị khóa công khai, vì vậy nó giải mã mật mã thành văn bản thuần túy và hiển thị kết quả cho Bob ngay khi anh ta đọc tin nhắn trên điện thoại của mình. Và như mong đợi, thông điệp Bob đọc là "USN Kongsberg là tốt nhất!".

Được rồi, vậy nếu có một diễn viên khác tham gia vào kịch bản này thì sao? Có Eve, một người phụ nữ bí ẩn, người đang bí mật theo dõi các hoạt động mạng của Alice. Eve đang đánh hơi các gói trên mạng máy tính và đã chụp thành công các gói mạng chứa tin nhắn riêng tư mà Alice gửi cho Bob. Đây là nơi chúng Chúng em quay lại bộ ba CIA và thuật ngữ bảo mật dữ liệu, nơi thông tin được ẩn về mặt mã hóa để tránh bất kỳ bên thứ ba trái phép nào đọc nó, chẳng hạn như Eve trong kịch bản này.

Mặc dù Eve đã nắm bắt được thông điệp mà Alice gửi cho Bob, nó được viết bằng mật mã và do đó không thể giải mã trừ khi Eve biết khóa riêng được sử dụng trong quá trình truyền tin nhắn.



Hình 7: Eve chụp thư được mã hóa

# Triển khai RSA

Chúng em đã phát triển thuật toán RSA trong Python nơi nó mã hóa và giải mã tin nhắn văn bản bằng các tính toán toán học khác nhau. Chương trình chạy tốt và nó mã hóa thành công và giải mã các thông báo được cung cấp bởi người dùng chương trình.

Người dùng viết văn bản thuần túy vào bảng điều khiển chương trình, mà không cần chuyển đổi thủ công các ký tự thành các giá trị thập phân trước. Chương trình xử lý tất cả các công việc này, vì vậy người dùng chỉ cần gửi một văn bản và không có gì khác. Tuy nhiên, người dùng cần phải nhập hai số nguyên tố, cùng với giá trị số mũ trong quá trình khởi chạy chương trình. Chúng em đã quyết định thực hiện dựa trên người dùng này và Chúng em không muốn khắc cứng các giá trị này vì sẽ dễ dàng hơn để kiểm tra chương trình với các số nguyên tố và giá trị số mũ khác nhau nếu người dùng có thể quyết định các giá trị này trong quá trình khởi chạy chương trình, có nghĩa là khác nhau Số nguyên tố và giá trị số mũ cũng có thể được sử dụng.

Chúng em đã vấp ngã một số khó khăn về kỹ thuật trong quá trình phát triển chương trình, chẳng hạn như thực tế là khóa riêng D trở nên cực kỳ lớn khi số nguyên tố lớn được cung cấp và giá trị số mũ lớn được sử dụng. Chúng em không thể lưu các giá trị này bên trong một số nguyên trong PYTHON, không phải là một float hoặc các loại dữ liệu khác. Chúng em đã kết thúc bằng cách sử dụng "số nguyên dài không dấu", có thể lưu trữ ít nhất 64 bit dữ liệu, nhưng tại một số điểm, điều này cũng không đủ tốt.

Chúng em cũng quyết định mã hóa từng nhân vật vào thời điểm đó, thay vì toàn bộ bản rõ trong một đợt. Điều này là để ngăn chặn các số lượng lớn gây ra tràn bộ đệm, và để xử lý và lưu trữ từng ký tự từng cái một. Tất cả các ký tự được lưu trữ bên trong bộ đệm (mảng) và các giá trị tương tự này được tải ra khỏi bộ đệm khi quá trình giải mã bắt đầu lại. Chúng em tin rằng đây là một cách tốt để xử lý các văn bản lớn, vì chương trình chỉ quan tâm đến một nhân vật tại một thời điểm và không quan tâm đến việc toàn bộ câu nào.

Hình ảnh 8: Xem trước chương trình

Chúng em đã chạy chương trình bằng các tham số khác nhau mỗi lần: lần đầu tiên Chúng em sử dụng p = 53, q = 59, e = 3 và được mã hóa văn bản "abc" đã trả về bản mã "018". Sau đó, bản mã được giải mã trở lại "ABC". Lần thứ hai Chúng em đã sử dụng p = 101, q = 109, e = 7 và được mã hóa văn bản "baba" đã trả về bản mã "1010". Sau đó, bản mã được giải mã trở lại "Baba".

## Source code

# Kết luận

Thuật toán RSA là một thuật toán mã hóa rất thú vị, và nó chắc chắn là một trong những thuật toán tốt nhất và an toàn nhất có sẵn vào hôm nay. Nó cung cấp mã hóa tuyệt vời và đáng tin cậy về mặt bảo mật và hiệu suất. Bảo mật mã hóa phụ thuộc vào thực tế là các số nguyên tố được sử dụng trong quá trình tạo quan trọng phải đủ lớn để không thể phá vỡ và điều này khá thú vị. Có thể có một thời gian trong tương lai khi siêu máy tính có thể phá vỡ những thứ này, nhưng điều đó sẽ không xảy ra bất cứ lúc nào sớm nhất.

Mặc dù thuật toán cung cấp mã hóa tuyệt vời và nó đáng tin cậy, bảo mật tổng thể thực sự dựa trên nhà phát triển chương trình hoặc người dùng thuật toán. Nếu người dùng chọn số nguyên tố nhỏ, nó sẽ ảnh hưởng đến rất nhiều bảo mật mà thuật toán RSA cung cấp và do đó rất dễ bị mất điện tử và các cuộc tấn công Brutegforce. Vì vậy, nói cách khác, chỉ cần sử dụng thuật toán RSA là không đủ - nó phải được sử dụng đúng cách để đạt được mức mã hóa, ban đầu nó cung cấp, vì nó phải được sử dụng chính xác về quy trình tạo quan trọng và chuẩn bị ban đầu của thuật toán.

Chúng em đã học được vô cùng trong giai đoạn thi nhà này và Chúng em không bao giờ nghĩ RSA sẽ thú vị, cho đến khi Chúng em thực sự bắt đầu đọc về nó. Các toán học đằng sau thuật toán là không thể tin được và Chúng em thấy nó hấp dẫn rằng các tính toán toán học đơn giản như vậy có thể tạo ra một thuật toán mã hóa lớn như vậy, được sử dụng trên toàn thế giới.

Chúng em cũng đánh giá cao sự thật rằng chúng Chúng em có cơ hội thực sự mã hóa và thực hiện thuật toán. Đó là một kinh nghiệm thú vị để sử dụng các kỹ năng lập trình của Chúng em để tạo một thuật toán, và Chúng em đã học được rất nhiều về mặt lý thuyết và thực tế. Nhiệm vụ này có tổng thể là một trải nghiệm học tập tuyệt vời đối với Chúng em và Chúng em hy vọng bạn hài lòng với công việc của mình trong suốt nhiệm vụ này!

# Tài liệu tham khảo

1. ”Discrete Mathematics and Its Applications”

Kenneth H. Rosen, 7th Edition (2012)

1. ”Cryptography and Network Security: Principles and Practice”

William Stallings, 7th Edition (2016)

1. ”Introduction to Cybersecurity (DCS3101): Mandatory Assignments”

Sirajuddin Asjad, University of South-Eastern Norway

1. Comparitech: ”What is AES encryption and how does it work?” <https://www.comparitech.com/blog/information-security/what-is-aes-encryption>
2. GeeksforGeeks: ”RSA Algorithm in Cryptography” <https://www.geeksforgeeks.org/rsa-algorithm-cryptography/>
3. JohnDCook: ”Three applications of Euler’s theorem” <https://www.johndcook.com/blog/2018/09/23/eulers-theorem/>
4. Binance Academy: ”Symmetric vs. Asymmetric Encryption” <https://www.binance.vision/security/symmetric-vs-asymmetric-encryption>
5. WolframMathWorld: ”Euclidean Algorithm” <http://mathworld.wolfram.com/EuclideanAlgorithm.html>
6. GeeksforGeeks: ”Euclidean algorithms (Basic and Extended)” <https://www.geeksforgeeks.org/euclidean-algorithms-basic-and-extended/>
7. WolframMathWorld: ”Totient Function” <http://mathworld.wolfram.com/TotientFunction.html>
8. Image sources:

<https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences>