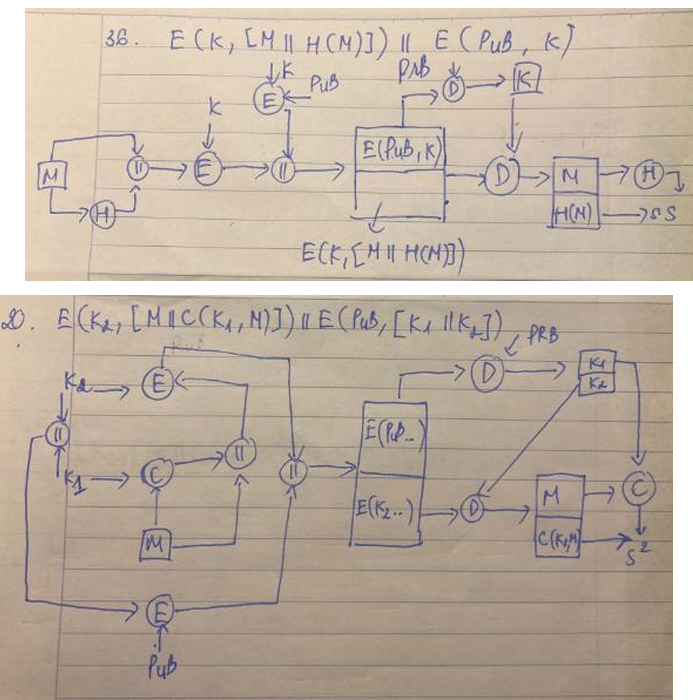
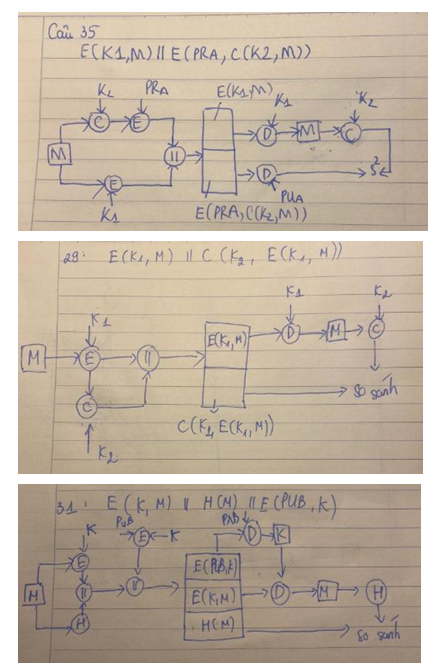
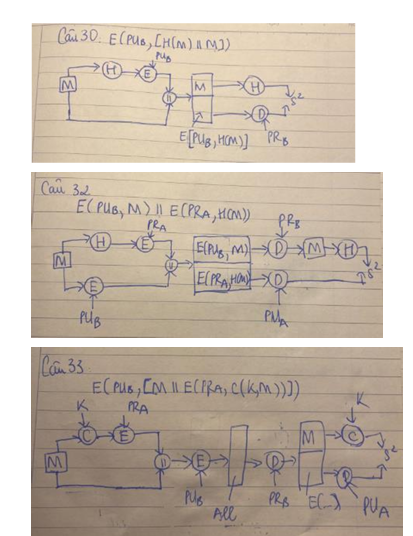
****

****

**** ****

**Bài tập 20 (ChatGPT).** Nêu các bước kỹ thuật brute-force

Brute-force là một phương pháp tấn công mật khẩu bằng cách thử nghiệm tất cả các khả năng có thể của mật khẩu cho đến khi tìm ra mật khẩu đúng. Dưới đây là các bước cơ bản của kỹ thuật brute-force:

1. Xác định mục tiêu: Xác định mục tiêu của tấn công brute-force, chẳng hạn như tài khoản người dùng, mật khẩu hoặc khóa mã hóa.
2. Xác định không gian khóa: Xác định không gian khóa, tức là tất cả các khả năng có thể của mật khẩu hoặc khóa mã hóa.
3. Chuẩn bị công cụ tấn công: Sử dụng phần mềm hoặc kịch bản tấn công để tự động thực hiện việc thử nghiệm mật khẩu.
4. Chọn chiến lược thử nghiệm: Chọn chiến lược thử nghiệm mật khẩu, có thể bao gồm thử tất cả các ký tự có thể, thử theo từ điển, hoặc thử theo mô hình có thể dự đoán được.
5. Thực hiện tấn công: Thực hiện tấn công brute-force bằng cách thử nghiệm mật khẩu từ không gian khóa đã xác định. Cứ mỗi mật khẩu được thử, hệ thống sẽ kiểm tra xem mật khẩu đó có đúng không.
6. Ghi lại kết quả: Ghi lại tất cả các mật khẩu đã thử và kết quả (đúng hay sai).
7. Kiểm tra kết quả: Kiểm tra kết quả để xem liệu tấn công đã thành công hay chưa. Nếu tìm thấy mật khẩu đúng, tấn công thành công.
8. Tối ưu hóa và điều chỉnh: Tùy thuộc vào kết quả, tối ưu hóa và điều chỉnh chiến lược thử nghiệm để làm cho tấn công hiệu quả hơn.

Lưu ý rằng brute-force thường là một phương pháp chậm và đòi hỏi nhiều tài nguyên tính toán. Các biện pháp bảo mật như việc sử dụng mật khẩu mạnh, khóa kép, và giới hạn số lần thử mật khẩu được áp dụng để ngăn chặn loại tấn công này.

**Bài tập 21 chương 5 (docs).** Các kỹ thuật lấy mật khẩu, xâm nhập?

Kỹ thuật lấy mật khẩu

1. Thử các mật khẩu từng dùng với các tài khoản chuẩn gắn liền với hệ thống.
2. Thử hết tất cả các mật khẩu ngắn
3. Thử các từ trong từ điển trực tuyến của hệ thống, hoặc danh sách mà rất có khả năng được làm mật khẩu
4. Thu thập thông tin người dùng như họ tên đầy đủ, tên vợ, chồng, con cái, …
5. Thử tất cả số điện thoại người dùng, các số an sinh xã hội, số nhà, phòng làm việc
6. Thử tất cả các biển xe hợp lý của địa phương
7. Dùng trojan horse để ăn cắp mật khẩu
8. Đấu nối một đường truyền giữa người dùng từ xa với host hệ thống.

Kỹ thuật xâm nhập

1. Tìm cách lấy được mật khẩu người dùng
2. Khai thác các lỗ hổng an ninh của hệ thống (ví dụ lỗi tràn bộ đệm..)
3. Mua chuộc, dụ dỗ…

**Bài tập 22 (ChatGPT).** Phân biệt 2 chuẫn mã hóa dữ liệu AES và DES.

**Bài giải.**

Mật mã AES (Advanced Encryption Standard) và DES (Data Encryption Standard) là hai thuật toán mã hóa khác nhau được sử dụng để bảo vệ thông tin và dữ liệu. Dưới đây là một số điểm khác biệt giữa chúng:

1. Chiều dài khóa:
   1. AES: Hỗ trợ các khóa có độ dài 128, 192, hoặc 256 bit.
   2. DES: Sử dụng khóa 56 bit.
2. Độ an toàn:
   1. AES: Được coi là mạnh và an toàn, được sử dụng rộng rãi trong nhiều ứng dụng bảo mật.
   2. DES: Đã bị coi là không an toàn do độ dài khóa ngắn và đã bị thay thế bởi các tiêu chuẩn mã hóa mạnh mẽ hơn.
3. Hiệu suất:
   1. AES: Thường có hiệu suất cao hơn so với DES, đặc biệt là với các khối dữ liệu lớn.
   2. DES: Có thể trở nên chậm đối với các tập tin hoặc dữ liệu lớn.
4. Kiểu thuật toán:
   1. AES: Sử dụng thuật toán thay thế và hoán vị để mã hóa dữ liệu.
   2. DES: Sử dụng mô hình Substitution-Permutation Network (SPN).
5. Tiêu chuẩn:
   1. AES: Được chấp nhận rộng rãi và được sử dụng trong nhiều ứng dụng và giao thức bảo mật.
   2. DES: Đã bị thay thế bởi các tiêu chuẩn mới hơn như AES và 3DES.

Tóm lại, AES thường được ưa chuộng hơn DES trong các ứng dụng hiện đại do khả năng an toàn và hiệu suất cao hơn.

**Bài tập. Hàm HASH, SHA512, MAC là gì?**

**Bài giải.**

(\*) Hàm hash:

* Hàm Hash là một công cụ quan trọng trong chứng thực.
* Nhiệm vụ của hàm hash H là tác động lên thông điệp đầu vào M (có kích thước bất kì) để thu được một khối dữ liệu nhỏ H(M) có kích thước cố định ở đầu ra. H(M) được gọi là mã hash của M.
* H(M) đặc trưng cho M, nhưng thường có kích thước nhỏ hơn M, sẽ được sử dụng làm giá trị chứng thực.

Đặc điểm hàm hash

* H(M) là một hàm của tất cả các bít trong thông điệp M, chỉ cần thay đổi 1 bít của M cũng khiến H(M) bị thay đổi theo.
* Việc tính toán để tìm một giá trị N ≠ M thoả mãn điều kiện H(N) = H(M) là rất khó khăn (hoặc không thể).
* Không thể tính ngược để thu được M từ H(M).

**Ứng dụng**

* Mã hash thường được đính kèm và gửi cùng với thông điệp, nhằm mục đích chứng thực cho thông điệp đó là chính xác về mặt nội dung.
* Mã hash thường được dùng kết hợp với các mật mã khoá công khai hoặc đối xứng.

A diagram of a diagram

AI-generated content may be incorrect.

A diagram of a block diagram

Description automatically generated

* Mục đích: Đảm bảo rằng thông điệp M đến từ A và không bị thay đổi trên đường truyền (do lỗi đường truyền hoặc bị tấn công)
* Hình (b) giảm được khối lượng tính toán so với hình (a), nhưng không có khả năng giữ bí mật nội dung thông điệp như hình (a).

A diagram of a block diagram

Description automatically generated

Ký hiệu:

* PR: Private key – Khóa riêng (giữ bí mật)
* PU: Public key – Khóa công khai

Phân tích

* Hình (c): Đảm bảo rằng thông điệp M đến từ A và không bị thay đổi trên đường truyền, nhưng không có khả năng giữ bí mật nội dung thông điệp. Đây chính là cơ sở của chữ kí số.
* Hình (d) có thêm khả năng bảo mật cho thông điệp.

**(\*) Phân tích SHA-512**

Khái niệm

* Thông điệp đầu vào có chiều dài tối đa nhỏ hơn 2128 bít, chia thành N khối 1024 bít
* Đầu ra là một giá trị hash 512 bít

Quá trình thực hiện

* **Bước 1: Nối các bit đệm**. Thông điệp được độn thêm các bit sao cho chiều dài của nó khi chia cho chiều dài khối 1024 thì dư 896
* **Bước 2: Nối giá trị chiều dài**. Một giá trị 128 bit nhị phân được nối thêm vào cuối thông điệp. Khối này là một số nguyên không dấu, nó chứa chiều dài thực của thông điệp ban đầu (128+896=1024)
* Kết quả của hai bước đầu là thu được thông điệp có chiều dài là bội số của 1024 bit. Tổng chiều dài của thông điệp mới này là *N* × 1024 bit. Các khối của thông điệp được kí hiệu là *M*1, *M*2,..., *M*N.
* **Bước 3: Khởi tạo bộ đệm giá trị hash**. Một bộ đệm 512 bit được sử dụng để lưu trữ các kết quả trung gian và cuối chu trình sẽ là kết quả của hàm hash. Bộ đệm gồm tám thanh ghi 64-bit (*a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*). Các thanh ghi này được khởi tạo với các số nguyên (thập lục phân) 64-bit tương ứng như sau: a = 6A09E667F3BCC908, b = BB67AE8584CAA73B, c = 3C6EF372FE94F82B, d = A54FF53A5F1D36F1, e = 510E527FADE682D1, f = 9B05688C2B3E6C1F, g = 1F83D9ABFB41BD6B, h = 5BE0CDI9137E2179
* **Bước 4: Xử lý thông điệp theo các khối 1024-bit (128-word)**. Trọng tâm của thuật toán là một module chứa 80 vòng; module này được kí hiệu là F. Cấu trúc logic của thuật toán được miêu tả trong hình sau:

A diagram of a schedule

Description automatically generated

* Trong đó:
  + Wi là các giá trị 64 bít được lấy ra từ 1024 bít của khối.
  + Ki là các hằng số ngẫu nhiên dài 64 bít, chúng được cộng vào mỗi vòng nhằm loại bỏ tính đều đặn của dữ liệu đầu vào.
* **Bước 5: Đầu ra.** Sau khi tất cả *N* khối 1024-bit đã được xử lý, đầu ra của giai đoạn thứ *N* là một thông điệp sắp xếp (message digest) hay mã hash, là một giá trị 512-bit.

**(\*) MAC**

* MAC (Message Authentication Code - Mã chứng thực thông điệp) là một kĩ thuật mã hoá nhằm biến đổi thông điệp đầu vào M (có kích thước bất kì) thành một khối dữ liệu nhỏ có kích thước cố định ở đầu ra.
* Khối dữ liệu đầu ra đó được gọi là mã MAC của thông điệp M. Nếu M bị thay đổi thì mã MAC của nó cũng sẽ thay đổi theo.
* Giống như mã hash, mã MAC thường được đính kèm và gửi cùng với thông điệp, nhằm mục đích chứng thực cho thông điệp đó là chính xác về mặt nội dung.
* Sự khác biệt cơ bản của MAC với các hàm hash là: Kĩ thuật mã hoá của MAC có sử dụng mật khoá *K*:
  + ***MAC* = C(*K*, *M*)**
  + *M* = Thông điệp đầu vào.
  + C = Hàm MAC.
  + *K* = Mật khoá chia xẻ giữa bên gửi và bên nhận.
  + *MAC* = Mã chứng thực thông điệp (Mã MAC).

A diagram of a graph

Description automatically generated with medium confidence

* Nếu có kẻ tấn công thay đổi thông điệp *M*, hắn sẽ không thể tạo ra mã *MAC* phù hợp vì không biết mật khoá *K*.
* Người nhận sẽ phát hiện ra sự giả mạo khi giá trị *MAC* nhận được và giá trị *MAC* mà họ tính toán từ *M* có sự khác nhau.

Một số cách dùng khác của MAC

A diagram of a block diagram

Description automatically generated

**Bài tập (ChatGPT). Phân biệt Hash và Mac.**

**Bài giải.**

Hash (hàm băm) và MAC (Message Authentication Code - mã xác thực tin nhắn) đều là các cơ chế mật mã được sử dụng trong bảo mật thông tin, nhưng chúng có mục đích và ứng dụng khác nhau. Dưới đây là sự phân biệt giữa Hash và MAC:

Hash (Hàm Băm):

* Mục Đích: Mục đích chính của hàm băm là chuyển đổi dữ liệu đầu vào thành một giá trị băm (hash) cố định với kích thước cố định. Nó không dùng để xác thực tính toàn vẹn hay nguồn gốc của dữ liệu.
* Đặc Điểm Cơ Bản:
  + Hàm băm không sử dụng khóa. Mỗi lần đầu vào giống nhau sẽ tạo ra một giá trị băm duy nhất.
  + Không có khả năng giải mã, nghĩa là từ giá trị băm, rất khó để xác định ngược lại dữ liệu đầu vào.
* Ứng Dụng:
  + Thường được sử dụng để lưu trữ mật khẩu (mật khẩu không được lưu trữ, chỉ lưu trữ giá trị băm).
  + Được sử dụng trong việc kiểm tra tính toàn vẹn của dữ liệu (ví dụ: checksum).

MAC (Message Authentication Code):

* Mục Đích: Mục đích chính của MAC là xác thực tính toàn vẹn và xác thực nguồn gốc của dữ liệu. Nó thường sử dụng một khóa để tạo ra mã xác thực.
* Đặc Điểm Cơ Bản:
  + Sử dụng khóa để tạo ra mã xác thực, điều này làm cho MAC có thể kiểm tra nguồn gốc của dữ liệu.
  + Cho phép xác thực và ngăn chặn sự thay đổi của dữ liệu trên đường truyền.
* Ứng Dụng:
  + Thường được sử dụng để xác thực tính toàn vẹn và nguồn gốc của tin nhắn hoặc dữ liệu truyền tải giữa các bên (ví dụ: trong giao thức TLS/SSL).

Có thể được sử dụng để xác thực tính toàn vẹn của dữ liệu lưu trữ.

* Tóm Lược:
  + Hash: Chuyển đổi dữ liệu thành giá trị băm, không sử dụng khóa, không thể giải mã, thường được sử dụng để kiểm tra tính toàn vẹn và lưu trữ mật khẩu.
  + MAC: Sử dụng khóa để tạo ra mã xác thực, giúp xác thực tính toàn vẹn và nguồn gốc của dữ liệu, thường được sử dụng trong việc xác thực giao tiếp và bảo vệ tính toàn vẹn.

**Bài tập 3. Khóa riêng, khóa công khai là gì?**

**Bài giải.**

Khóa riêng (private key) và khóa công khai (public key) là hai thành phần quan trọng trong hệ thống mật mã đối xứng (asymmetric cryptography). Đây là một phương pháp mật mã sử dụng cặp khóa này để thực hiện các quá trình mã hóa và giải mã thông tin.

Khóa Riêng (Private Key)

* Là một khóa bí mật, được giữ bởi người sử dụng hoặc hệ thống.
* Được sử dụng để giải mã thông tin đã được mã hóa bằng khóa công khai tương ứng của nó.
* Việc bảo mật và giữ an toàn khóa riêng là quan trọng, vì nếu ai đó có được khóa riêng, họ có thể giải mã thông tin mà chỉ người giữ khóa đó mới có thể thực hiện được.

Khóa Công Khai (Public Key)

* Là một khóa được công bố rộng rãi và có thể được chia sẻ với bất kỳ ai.
* Được sử dụng để mã hóa thông tin và tạo chữ ký số.
* Thông tin được mã hóa bằng khóa công khai chỉ có thể được giải mã bởi khóa riêng tương ứng. Do đó, người gửi thông tin có thể sử dụng khóa công khai của người nhận để mã hóa thông tin, đảm bảo rằng chỉ người nhận mới có thể giải mã thông tin đó.

**Bài tập 10. Mã hóa đa kí tự là gì? Ưu và nhược điểm của loại mã hóa này.**

Mã hóa đa ký tự (Polyalphabetic encryption) là một phương pháp mã hóa mà mỗi ký tự trong văn bản được ánh xạ với nhiều ký tự mã hóa tương ứng, thay đổi theo ngữ cảnh hoặc theo quy luật nào đó. Một số thuật toán mã hóa đa ký tự nổi tiếng bao gồm Vigenère, Autokey, và Playfair.

Ưu điểm của mã hóa đa ký tự

* Chống lại tấn công tần suất ký tự: Mã hóa đa ký tự làm cho mối quan hệ giữa các ký tự gốc và ký tự mã hóa trở nên phức tạp, khó dự đoán, do đó khó bị tấn công bằng cách phân tích tần suất xuất hiện của các ký tự.
* Tăng độ phức tạp: So với một số phương pháp mã hóa đơn giản như mã hóa Caesar, mã hóa đa ký tự có độ phức tạp cao hơn, giúp làm tăng độ khó khăn của quá trình giải mã.
* Khả năng chống lại các phương pháp tấn công thông thường: Mã hóa đa ký tự thường khó bị tấn công bằng cách sử dụng phương pháp tần suất chữ cái, bigram hoặc trigram, nếu thực hiện đúng.

Nhược điểm của mã hóa đa ký tự:

* Khó quản lý khóa: Nếu khóa không được quản lý cẩn thận, có thể dẫn đến việc giảm tính an toàn của hệ thống mã hóa đa ký tự.
* Khả năng bị tấn công Kasiski: Một số thuật toán mã hóa đa ký tự, như Vigenère, có thể bị tấn công bằng cách sử dụng phương pháp tìm kiếm khoảng cách giữa các trùng lặp trong văn bản mã hóa.
* Không thay đổi theo ngữ cảnh: Một số phương pháp mã hóa đa ký tự không thay đổi dựa vào ngữ cảnh của văn bản, điều này có thể tạo ra các mô hình dự đoán và tấn công.
* Không an toàn nếu không được sử dụng đúng cách: Nếu khóa được sử dụng một cách dễ đoán hoặc nếu quy luật của mô hình mã hóa được phát hiện, mô hình mã hóa đa ký tự có thể trở nên yếu đuối.

**Bài tập chương 3.4.** Chữ ký số và giao thức chứng thực

**Giao thức chứng thực**

* Giao thức là tập hợp các quy tắc để các bên tham gia liên lạc trao đổi thông tin với nhau
* Các giao thức chứng thực là giao thức giúp các bên tham gia liên lạc tự nhận dạng lẫn nhau và chuyển giao khoá phiên

Giao thức chứng thực dùng mã hóa đối xứng

* Giữa hai bên liên lạc A và B cần có một bên thứ ba gọi là KDC (Key Distribution Center - Trung tâm phân phối khóa tin cậy)
* Bên A sẽ chia sẻ khoá Ka với KDC
* Bên B sẽ chia sẻ khoá Kb với KDC
* KDC sẽ sinh ra khoá phiên Ks (session key). Ks chỉ tồn tại trong một thời gian ngắn để A và B thực hiện phiên liên lạc.

Giao thức Needham-Schroeder

1. A 🡪 KDC: IDA || IDB || N1
2. KDC 🡪 A: E(Ka, [Ks || IDB || N1 || E(Kb, [Ks || IDA])])
3. A 🡪 B: E(Kb, [Ks || IDA])
4. B 🡪 A: E(Ks, N2)
5. A 🡪 B: E(Ks, f(N2))

Nhận xét

* N1, N2 là các giá trị nonce (lời gọi), có tác dụng chống giả mạo
* Khi A gửi cho KDC một nonce (N1), theo quy ước KDC sẽ phải trả lời A bằng một giá trị tương ứng với N1, nếu không có nghĩa là thông điệp trả lời đã bị giả mạo

Quá trình

* KDC sinh ra khóa phiên Ks rồi gửi cho A
* A gửi thông điệp đã mã hóa (có chứa Ks) cho B
* B gửi lại A một nonce (N2), chứng tỏ B đã nhận được Ks, và muốn kết nối với A
* A trả lời B bằng một thông điệp có chứa N2. Phiên kết nối giữa A và B hình thành. Sau đó A và B có thể trao đổi các thông tin được mã hóa bởi Ks

Nhược điểm

* Chưa có cơ chế kiểm tra thời hạn sử dụng của khóa phiên Ks.
* Nếu kẻ tấn công (bằng cách nào đó) có được một khóa phiên Ks cũ, anh ta có thể dùng nó để đóng giả A nhằm lừa gạt B

Cụ thể

* Kẻ tấn công (X) lắng nghe và lấy được một thông điệp A gửi cho B ở bước 3 (có chứa khóa phiên cũ mà anh ta đã biết), sao chép thông điệp đó lại để dùng sau này.
* Khi cần tấn công, X chặn một thông điệp A gửi cho B ở bước 3, thay thế nó bằng thông điệp cũ đã lưu, và gửi nhại lại cho B
* Khi B trả lời ở bước 4 (dùng Ks cũ), X chặn thông điệp này và lấy được N2. Từ đó X có thể đóng giả A để nói chuyện với B

Khắc phục

* Cần gắn thêm nhãn thời gian T để thiết lập thời hạn sử dụng cho khóa phiên (xem giao thức Denning sau đây)

Giao thức Denning

1. A 🡪 KDC: *ID*A||*ID*B
2. KDC 🡪 A: E(*K*a, [*K*s||*ID*B||*T*||E(*K*b, [*K*s||*ID*A||*T*])])
3. A 🡪 B: E(*K*b, [*K*s||*ID*A||*T*])
4. B 🡪A: E(*K*s, *N*1)
5. A 🡪 B: E(*K*s, *f*(*N*1))

Nhận xét

* Khóa phiên Ks được gắn kèm với nhãn thời gian T
* B chỉ chấp nhận thông điệp (chứa khóa phiên Ks) ở bước 3 nếu giá trị của nhãn thời gian T đủ gần với thời gian hiện hành của B, bao gồm độ trễ mạng và sai số cho phép
* Đồng hồ của các bên tham gia liên lạc phải được đồng bộ hóa với nhau.

Lưu ý:

* Các giao thức trên chỉ áp dụng cho trường hợp các bên tham gia liên lạc đồng thời online trên mạng (ví dụ dịch vụ Chat)
* Nếu một bên tham gia liên lạc không online (ví dụ dịch vụ Email) thì cần có giải pháp khác

Trường hợp A, B không đồng thời trực tuyến (Chứng thực một chiều)

1. A 🡪 KDC: *ID*A||*ID*B||*N*1
2. KDC 🡪 A: E(*K*a, [*K*s||*ID*B||*N*1||E(*K*b, [*K*s||*ID*A])])
3. A 🡪 B: E(*K*b, [*K*s||*ID*A])||E(*K*s, *M*)

Nhận xét:

* Phương pháp này không sử dụng nhãn thời gian T nên vẫn có nguy cơ bị tấn công nhại.
* Tuy nhiên, việc xử lý email thường không tức thời, nên người nhận B chủ yếu dùng khoá phiên Ks để giải mã thông điệp M, chứ không dùng Ks để tiếp tục giao tiếp với A.

Giao thức chứng thực dùng mã hoá khoá công khai (của Denning)

* Giữa hai bên liên lạc A và B cần có một bên thứ ba gọi là AS (Authentication Server – Server chứng thực)
* AS không sinh ra khoá phiên Ks mà sẽ đưa ra “chứng chỉ” khoá công khai
* Khóa phiên được chọn và mã hóa bởi A

Quá trình

1. A 🡪 AS: *IDA*||*IDB*
2. AS 🡪 A: E(*PRas*, [*IDA*||*PUa*||*T*])||E(*PRas*, [*IDB*||*PUb*||*T*])
3. A 🡪 B: E(*PRas*, [*IDA*||*PUa*||*T*])||E(*PRas*, [*IDB*||*PUb*||*T*])||E(*PUb*, E(*PRa*, [*Ks*||*T*]))

Nhận xét

* AS sinh ra chứng chỉ khoá công khai (chứa khoá công khai của người dùng), mỗi chứng chỉ chỉ có hiệu lực trong khoảng thời gian T
* Khi A nhận được chứng chỉ do AS gửi, A sẽ biết được khoá công khai của B, và dùng nó để chuyển giao khoá phiên Ks cho B
* Khoá phiên được mã hoá 2 lần: Lần 1 bởi PRa để chứng thực, lần 2 bởi PUb để giữ bí mật

**Chữ ký số là**

* Chữ ký số là một cơ chế chứng thực cho phép tác giả thông điệp gắn thêm một đoạn mã vào thông điệp. Đoạn mã này đóng vai trò như một chữ ký.
* Chữ ký được tạo ra bằng cách tính giá trị hash của thông điệp và mã hóa nó bằng khóa riêng của tác giả.
* Chữ ký bảo đảm về nguồn gốc và tính toàn vẹn của thông điệp

**Tính chất chữ ký số**

* Phải xác minh được tác giả cùng thời gian của chữ ký.
* Phải chứng thực được nội dung thực sự vào thời gian ký.
* Phải kiểm tra được bởi một bên thứ ba, để giải quyết các tranh chấp.

**Phân loại**

* Chữ kí số trực tiếp
* Chữ kí số trọng tài

A diagram of a block diagram

Description automatically generated

A diagram of a diagram

Description automatically generated

**Phân tích chữ ký số trực tiếp**

* Loại chữ ký này chỉ liên quan đến hai bên tham gia (nguồn và đích).
* Một chữ ký số có thể tạo ra bằng cách mã hoá toàn bộ thông điệp bằng khoá riêng của người gửi (hình a), hoặc mã hoá mã hash của thông điệp với khoá riêng của người gửi (hình b).
* Có thể tăng cường tính bảo mật bằng cách thêm một lần mã hoá toàn bộ thông điệp và chữ ký bằng mật khoá chia sẻ (hình c) , hoặc bằng khoá-công-khai của của người nhận (hình d)
* Nhược điểm:
  + Chưa ghi lại thời gian phát sinh chữ kí
  + Thiếu cơ chế để xử lý khi xảy ra tranh chấp.
  + Người gửi có thể phủ nhận chữ kí của mình

**Phân tích chữ ký số trọng tài**

* Những nhược điểm của chữ ký số trực tiếp có thể được giải quyết nhờ vào bên thứ ba tin cậy (trọng tài).
* Giả sử cần gửi thông điệp từ X đến người nhận Y. Trước hết X phải kí vào thông điệp rồi gửi tới trọng tài A.
* A sẽ tiến hành kiểm tra thông điệp và chữ kí, nếu thấy hợp lệ thì sẽ gắn thêm thời gian và nhãn “đã kiểm tra” vào thông điệp, rồi gửi cho Y.
* Trọng tài đóng một vài trò chủ yếu và nhạy cảm trong hình thức này
* Tất cả các bên tham gia đều phải thỏa thuận về sự tin cậy tuyệt đối để cơ chế phân xử thực sự có hiệu quả.

**Ví dụ chữ ký số trọng tài**

Kí hiệu: X = người gửi, Y = người nhận, A = trọng tài, M = thông điệp, T = timestamp, nhãn thời gian để xác định thời gian phát sinh chữ kí, IDX = Tên người gửi X, Kxa = Khóa mà X và A biết, Kay = Khóa mà A và Y biết

**Bài tập. Kỹ thuật lây lan Virus, ngụy trang, cách phòng tránh.**

**(\*) Kỹ thuật lây lan virus**

* Muốn lây lan thì virus phải được kích hoạt. Khi virus được kích hoạt, mã lệnh của nó sẽ tìm cách tự sao chép tới các file khác hoặc đĩa khác.
* Có nhiều cách để kích hoạt một virus, chủ yếu là lợi dụng các sơ hở của hệ thống, hoặc tìm cách “dụ” người dùng chạy file nhiễm virus.
* Hacker viết virus, gắn nó vào một chương trình (do anh ta tự viết hoặc ăn cắp), rồi phát tán chương trình đó (ví dụ bằng cách gửi nó lên một website cung cấp phần mềm miễn phí). Ai download và chạy chương trình này sẽ kích hoạt virus
* Có thể đính kèm file nhiễm virus vào email, rồi gửi cho mọi người. Khi mở file đính kèm thì virus sẽ được kích hoạt
* Virus “đóng giả” các file tài liệu hoặc Folder (bằng cách đổi tên, đổi icon…) để “lừa” người dùng bấm chuột vào
* Lợi dụng tính năng Auto Run của các ổ đĩa USB, CD để kích hoạt virus

**(\*) Kỹ thuật ngụy trang virus**

* Để chống lại các phần mềm diệt virus, virus phải có giải pháp nguỵ trang, che dấu bản thân
* Sau khi lây vào một file mới, virus thường sửa đổi lại ngày, giờ, thuộc tính file… cho giống với file chưa bị lây nhiễm
* Có thể áp dụng phương pháp nén để giữ nguyên kích thước của file sau khi lây
* Dấu mã lệnh virus vào các vùng an toàn trên đĩa, và đánh dấu chúng là các sector hỏng (đối với virus boot hoặc virus lai).
* Liên tục biến đổi mã lệnh virus để thay đổi bản thân, tránh bị nhận dạng (virus đa hình).

**(\*) Kỹ thuật lây lan worm**

* **Tìm cách thi hành chương trình từ xa:** Một số máy cho phép chạy chương trình shell ở xa mà không yêu cầu xác thực. Nếu thành công, shell ở xa sẽ tải chương trình worm về và tiếp tục lây vào các máy khác kết nối với nó
* **Tìm cách đăng nhập từ xa:** Worm có thể sử dụng chương trình đoán mật khẩu, lần lượt thử đăng nhập với một danh sách mật khẩu được chuẩn bị từ trước. Nếu đoán mật khẩu thành công, Worm có thể đăng nhập được vào bất cứ máy tính nào người dùng đó có tài khoản.
* **Tận dụng lỗi tràn bộ đệm:** Worm Morris đã tận dụng lỗi tràn bộ đệm của chương trình finger (có trên một số website) để kích hoạt Worm trên máy bị tấn công.
* **Tận dụng lỗi của hệ thống email:** Một số hệ thống email cho phép chương trình worm gửi thư có chưa phần khởi động cho người khác và thi hành nó

**(\*) Các bước lây lan của worm**

* Tìm kiếm các hệ thống khác để lây nhiễm bằng cách khảo sát các bảng host hay những nơi chứa các địa chỉ các hệ thống từ xa.
* Thiết lập kết nối tới hệ thống từ xa.
* Tự sao chép đến hệ thống từ xa và kích hoạt để tự thi hành.

**(\*) Phòng tránh**

* Thứ nhất, hãy chọn một hệ điều hành có độ an toàn cao, chế độ kernel và chế độ người dùng có ranh giới rõ ràng, người dùng và người quản trị hệ thống phải có mật khẩu đăng nhập riêng. Khi đó, nếu virus có lẻn vào thì cũng khó có thể lây vào hệ thống nhị phân.
* Thứ hai, chỉ cài đặt các phần mềm chính thống mua từ các nhà sản xuất đáng tin cậy. Không download các phần mềm không rõ nguồn gốc từ Internet.
* Thứ ba, hãy mua một chương trình diệt virus tốt và sử dụng nó thường xuyên. Hãy luôn cập nhật phiên bản mới nhất của nó từ Website của nhà sản xuất.
* Thứ tư, đừng bao giờ bấm chuột vào các file thi hành đính kèm trong email, hay các file thi hành tự nhiên xuất hiện trong ổ đĩa USB của bạn. Nói chung, phải cận thận với tất cả các file có thể chứa mã lệnh!
* Thứ năm, thường xuyên sao lưu các file quan trọng ra bộ nhớ ngoài (như đĩa USB, đĩa CD), đề phòng trường hợp máy tính của bạn bị nhiễm virus, bạn vẫn có thể khôi phục lại các file về trạng thái ban đầu.
* Thứ sáu, thường xuyên cập nhật các bản vá lỗi mới nhất của hệ điều hành, của trình duyệt web, và các phần mềm khác.
* **Các bước thực hiện SSL**

A screenshot of a chat

Description automatically generated

1. Client Gửi Yêu Cầu Kết Nối: Client gửi yêu cầu kết nối tới server thông qua giao thức SSL/TLS.
2. Server Gửi Chứng Chỉ (Certificate): Server gửi chứng chỉ kỹ thuật số của mình tới client. Chứng chỉ chứa khóa công khai của server và thông tin xác minh.
3. Client Xác Nhận Chứng Chỉ: Client kiểm tra chứng chỉ của server để đảm bảo rằng nó được ký bởi một CA (Certificate Authority) tin cậy và không bị thu hồi.
4. Client Tạo Khóa Phiên (Session Key): Client tạo một khóa phiên (session key) mới, được sử dụng cho việc mã hóa và giải mã dữ liệu trong suốt phiên kết nối.
5. Client Gửi Khóa Phiên Đã Mã Hóa đến Server: Client sử dụng khóa công khai của server để mã hóa khóa phiên, sau đó gửi nó đến server.
6. Server Giải Mã Khóa Phiên: Server sử dụng khóa riêng tư của mình để giải mã khóa phiên đã nhận từ client.
7. Kết Nối Bảo Mật Được Thiết Lập: Với khóa phiên đã chia sẻ, kết nối SSL/TLS được thiết lập giữa client và server. Từ bước này trở đi, dữ liệu được truyền qua lại giữa client và server sẽ được mã hóa và giải mã bằng khóa phiên.
8. Gửi Xác Nhận Cho Client: Server gửi một xác nhận đến client để thông báo rằng quá trình thiết lập kết nối đã hoàn tất và dữ liệu có thể bắt đầu được truyền tải an toàn.

**Bài tập chương 4.1. Ứng dụng chứng thực chứng chỉ khóa công khai là gì?**

Việc phân phối khoá công khai cũng cần phải có phương pháp. Một số cách để phân phối khoá công khai như:

* Yết thị công khai
  + Người sở hữu khoá (A) có thể gửi khoá công khai của mình cho đối tác, hoặc công bố rộng rãi cho cộng đồng
  + Nhược điểm: Do không có cơ chế xác thực, bất cứ ai cũng có thể đóng giả làm A và gửi khoá công khai cho đối tác của A, khiến đối tác nhầm tưởng là đang liên lạc với A.
* Đặt trong một thư mục công cộng
  + Các khoá công khai được đặt trong một thư mục công cộng do một tổ chức đáng tin cậy quản lý.
  + Người dùng muốn công bố khoá công khai của mình thì phải đăng kí với tổ chức đó.
  + Nhược điểm: Nếu kẻ tấn công (bằng cách nào đó) chiếm được quyền truy cập của tổ chức, anh ta có thể thay thế các khoá công khai trong thư mục bằng các khoá giả mạo.
* Dùng chứng chỉ khoá công khai
  + Khoá công khai không được đặt trực tiếp trong thư mục công cộng, mà được đặt trong “chứng chỉ khoá công khai”
  + Người dùng A muốn công bố khoá công khai của mình thì phải đăng kí với một tổ chức đáng tin cậy có thẩm quyền cấp chứng chỉ, tổ chức này sẽ cấp cho A một chứng chỉ bao gồm khoá công khai của A (PUa), định danh của A (IDA), và chữ kí của tổ chức.
  + A có thể công bố rộng rãi chứng chỉ này hoặc gửi chứng chỉ cho đối tác, mà không phải lo lắng bị kẻ khác làm giả chứng chỉ.
  + Đối tác của A có thể kiểm tra tính hợp lệ của chứng chỉ dựa trên chữ kí của tổ chức.

Mô hình

A diagram of a certificate authority

Description automatically generated

* CA: Certificate Authority (Tổ chức cấp phát chứng chỉ)
* CA: là chứng chỉ khoá công khai của A. *C*A = E(*PR*auth, [*T*||*ID*A||*PU*a])
* *PR*auth là khóa riêng của tổ chức cấp phát chứng chỉ.
* *T* là một nhãn thời gian cho biết thời gian hiệu lực của chứng chỉ
* Đối tác của A có thể dùng *PU*auth để giải mã CA, nhằm kiểm tra tính hợp lệ của CA và thu được PUa:

D(*PUauth*, *CA*) = D(*PUauth*, E(*PRauth*, [*T*||*IDA*||*PUa*]))

= [*T*||*IDA*||*PUa*]

* Sau đó đối tác có thể liên lạc với A nhờ PUa.

**Bài tập chương 4.1. Hạ tầng khoá công khai là gì?**

Hạ tầng khoá công khai (PKI - Public Key Infrastructure) là một tập phần cứng, phần mềm, con người, chính sách và các thủ tục cần thiết để tạo ra, quản lý, lưu trữ, phân phối và thu hồi các chứng chỉ số dựa trên kỹ thuật mật mã bất đối xứng.

Hạ tầng khoá công khai dựa trên tiêu chuẩn X.509 được kí hiệu là PKIX

Mô hình

A diagram of a flowchart

Description automatically generated

* **CA (Certification Authority**): Cơ quan cấp phát chứng chỉ, quản lý, thu hồi và công bố danh sách các chứng chỉ bị thu hồi (CRL)
* **RA (Registration Authority**): Cơ quan đăng kí cấp phát chứng chỉ, có nhiệm vụ tiếp nhận và xác minh các yêu cầu về chứng chỉ số của người dùng, rồi gửi các yêu cầu đã xác minh cho CA để thực hiện yêu cầu đó.
* **Thực thể cuối (End Entity)**: là người dùng hay các bên sử dụng dịch vụ. Người dùng có thể gửi yêu cầu xin cấp chứng chỉ trực tiếp cho CA hoặc gián tiếp qua RA
* **Bên ban hành CRL (CRL issuer)**: CA có thể trực tiếp công bố danh sách các chứng chỉ bị thu hồi (CRL), hoặc giao cho CRL issuer công bố

**Bài tập 19. Khi mua bán thì người bán gửi cho người mua những thông tin gì (Liên quan đến OI và PI)?**

**Bài giải.**

Thông Tin Đặt Hàng (Order Information - OI):

* Số đơn hàng (Order Number): Để theo dõi và xác nhận đơn hàng.
* Sản phẩm và Dịch vụ: Mô tả chi tiết về sản phẩm hoặc dịch vụ mà người mua đang mua.
* Số lượng và Giá trị đơn hàng: Để xác định số lượng sản phẩm và giá trị cả đơn hàng.
* Phí vận chuyển và Thuế: Thông tin về chi phí vận chuyển và thuế, nếu có.
* Địa chỉ giao hàng và Thông tin liên lạc: Để xác định nơi gửi hàng đến và liên lạc với người mua.

Thông Tin Cá Nhân (Personal Information - PI):

* Tên và Địa chỉ của Người Mua: Để xác định người nhận hàng và thông tin liên lạc.
* Phương thức thanh toán: Thông tin về phương thức thanh toán được sử dụng, ví dụ như số thẻ tín dụng hoặc thông tin tài khoản ngân hàng.
* Thông tin Liên lạc: Địa chỉ email, số điện thoại để người bán có thể liên lạc với người mua nếu cần thiết.

Thông tin xác thực: Trong trường hợp cần, thông tin để xác minh danh tính người mua, ví dụ như chứng minh nhân dân hoặc giấy phép lái xe.

# Bài tập câu 3.

**Bài tập 1 (2202).** Mã hóa bằng phép XOR

Lập một chuỗi ký tự plaintext từ bàn phím. Nhập khóa K (16 bit) từ bàn phím. Chia chuỗi thành từng khối 16 bit (nếu không chia hết cho 16 bit thì thêm kí tự quy ước).

Mã hóa chuỗi ban đầu bằng cách xor từng khối của chuỗi với K. Hiện chuỗi sau khi mã hóa màn hình. Viết chương trình giải mã.

A screenshot of a computer code

Description automatically generated

**Bài tập 2 (2205).** Mã hóa bằng công thức C = 7\*P + K

Lập một chuỗi ký tự plaintext từ bàn phím (Các ký tự nằm trong khoảng từ ‘A’ tới ‘Z’). Nhập số nguyên K thuộc Z26 từ bàn phím. Mã hóa chuỗi ban đầu theo công thức C = 7\*P + K. Các phép tính được thực hiện trong Z26.

Hiện chuỗi sau khi mã hóa màn hình. Viết chương trình giải mã.

A screenshot of a computer program

Description automatically generated

**Bài tập 3 (1805).** Mã hóa bằng công thức C = P/a + b

Lập một chuỗi ký tự plaintext từ bàn phím (Các ký tự nằm trong khoảng từ ‘A’ tới ‘Z’). Nhập số nguyên a, b thuộc Z26 từ bàn phím. Mã hóa chuỗi ban đầu theo công thức C = P/a + b. Các phép tính được thực hiện trong Z26.

Hiện chuỗi sau khi mã hóa màn hình. Viết chương trình giải mã.

A computer code with numbers and symbols

Description automatically generated

A screenshot of a computer code

Description automatically generated

A computer code with black text

Description automatically generated

A computer code with black text

Description automatically generated**}**

**Bài tập 4.** Mã hóa bằng phép cộng (Giống Caesar?)

Lập trình nhập một chuỗi ký tự từ bàn phím, cộng mỗi phần tử của chuỗi với K (K là một số nguyên nhập từ bàn phím). Hiện chuỗi mới ra màn hình. Lập trình khôi phục lại chuỗi ban đầu.

A screenshot of a computer program

Description automatically generated

**Bài tập 5.** Mã hóa Caesar tổng quát

Lập trình nhập một chuỗi ký tự từ bàn phím, mã hóa chuỗi bằng thuật toán Caesar tổng quát với khóa K nhập từ bàn phím. Hiện chuỗi mới. Lập trình giải mã để khôi phục lại chuỗi ban đầu.

**Bài tập 6.** Mã hóa Caesar, giải mã Brute-force

Lập trình bẻ khóa mật mã Caesar bằng phương pháp Brute-force. Đầu vào chương trình là chuỗi ký tự cipher text thu được từ bài trên.

A computer screen shot of text

Description automatically generated

**Bài tập 7.** Mã hóa Affine

Lập trình nhập một chuỗi ký tự từ bàn phím, mã hóa chuỗi bằng thuật toán Affine với cặp số {a, b} nhập từ bàn phím. Hiện chuỗi mới ra màn hình.

Lập trình giải mã Affine để khôi phục lại chuỗi ban đầu:

* Đầu vào chương trình là cặp số {a, b} và chuỗi ký tự cipher text từ câu trên.
* Đầu ra chương trình là chuỗi ký tự plain text.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer code

Description automatically generated

**Bài tập 8.** Mã hóa Monoalphabetic

**Phần 1.** Lập trình nhập một chuỗi ký tự từ bàn phím, mã hóa chuỗi bằng thuật toán Monoalphabetic với khóa K nhập từ bàn phím (Khóa K là một chuỗi gồm 26 chữ cái có trật tự bất kỳ). Hiện chuỗi mới ra màn hình.

**Phần 2.** Lập trình giải mã Monoalphabetic để khôi phục lại chuỗi ban đầu:

* Đầu vào chương trình là khóa K và chuỗi ký tự cipher text từ phần 1.
* Đầu ra chương trình là chuỗi ký tự plain text

A computer code with text

AI-generated content may be incorrect.

**Bài tập 9.** Mã hóa Playfair

**Phần 1.** Mã hóa tay

Chọn một từ khóa bất kỳ rồi xây dựng ma trận khóa Playfair. Chọn một plaintext bất kỳ, áp dụng ma trận khóa trên để tạo ra ciphertext.

Thử giải mã ciphertext rồi so sánh với plaintext ban đầu.

Ví dụ. PT = DAIHOCTHUYLOI, K = CONGNGHE

**Bài giải.**

**Ma trận khóa K**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C | O | N | G | H |
| E | A | B | D | F |
| I/J | K | L | M | P |
| Q | R | S | T | U |
| V | W | X | Y | Z |

P = DAIHOCTHUYLOI

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P | DA | IH | OC | TH | UY | LO | IX |
| C | FB | PC | NO | UG | TZ | KN | LV |

**Phần 2.** Code

Chọn một từ khóa bất kỳ rồi xây dựng ma trận khóa Playfair. Chọn một plaintext bất kỳ, áp dụng ma trận khóa trên để tạo ra ciphertext.

Thử giải mã ciphertext rồi so sánh với plaintext ban đầu

A screenshot of a computer code

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer code

Description automatically generated

A computer code with black and white text

Description automatically generated

**Bài tập 10.** Mật mã Hill

Với plaintext là “paymoremoney” và sử dụng với mật khóa K nói trên, hãy xác định ciphertext. Giải mã ciphertext thu đươc bằng cách nhân nó với inv(K), rồi so sánh kết quả với plaintext ban đầu.





Mã hóa

A table with numbers and symbols

Description automatically generated

NVPKCTLFUTTL

Giải mã

A black background with numbers and equations

Description automatically generated

PAYMOREMONEY

**Bài tập 13.** Mã hóa Rain-Fence

Nhập một chuỗi plaintext. Mã hóa chuỗi bằng kỹ thuật Rain-Fence, hiện ciphertext ra màn hình. Giải mã ciphertext.

**Hướng giải.** Plaintext được viết dịch xuống tuần tự theo các đường chéo rồi đọc trình tự theo các hàng.

A screenshot of a computer code

Description automatically generated

**Bài tập 14.** Code đổi hàng cột

Nhập một chuỗi plaintext(dài không quá 25 ký tự). Sắp xếp các ký tự của chuỗi vào một ma trận 5x5(lần lượt theo từng hàng). Đọc các phần tử của ma trận theo từng cột để tạo thành ciphertext.

Giải mã ciphertext, so sánh kết quả với chuỗi ban đầu.

A screenshot of a computer program

AI-generated content may be incorrect.

**Bài tập 15.** Mật mã Feistel

Lập trình mô phỏng hoạt động của mật mã Feistel đơn giản với 2 vòng xử lý:

* Nhập một khối plaintext từ bàn phím, khối có độ dài m = 2w = 16 bít.
* Nhập khóa K(dài 8 bít) từ bàn phím. Khóa K\_i được sinh ra từ khóa K nhờ phép dịch trái K\_i lần.

Hàm F thực hiện phép cộng giữa R\_{i-1} với K\_i. Hiện khối ciphertext ra màn hình.

A computer code on a white background

Description automatically generated

**Bài tập 16.** Mật mã RSA

* Chọn một cặp số nguyên tố p, q
* Thực hiện các bước tạo khóa, mã hóa và giải mã theo thuật toán RSA

**Bài giải.**

Công thức mã hoá: *C* = *Pa* mod *n* (1)

Công thức giải mã: *P* = *Cb* mod *n* (2)

Trong đó:

* *n = p.q* là tích của hai số nguyên tố lớn
* *P, C* thuộc *Zn*
* *a* và *b* là các số nguyên thoả mãn điều kiện: *Pab* mod *n = P*
* Có thể đạt được điều đó nếu: *ab mod Φ = 1 (\*),* với *Φ = (p* − *1)(q* − *1)*

Điều đó chỉ đúng nếu *a* (và *b)* có quan hệ nguyên tố với *Φ*: *ƯSCLN(Φ, a) = 1*

Công thức *(\*)* tương đương với: *b = a-1 mod Φ*

Tạo khóa

* Chọn 2 số nguyên tố lớn *p*, *q*
* Tính *n* = *pq*
* Tính *Φ = (p − 1)(q − 1)*
* Chọn *a* sao cho *ƯSCLN(Φ,a) = 1*; *a* < *Φ*
* Tính *b = a-1 mod Φ*

Để mã hoá cần phải biết *a* và *n*. Khoá công khai chính là *K1 = (a, n)*

Để giải mã cần biết *b* và *n*. Khoá bí mật là *K2 = (b, n)*

**Bài tập 17.** Mật mã Vigenere tay to + Code

Cho chuỗi Plaintext sau : **KHONGBENHLATIENSUONGTUYETTRAN**

Hãy mã hóa chuỗi nói trên bằng mật mã Vigenere, với khóa K là họ tên của sinh viên, m là độ dài họ tên.

Giải mã ciphertext thu được và so sánh plaintext ban đầu.

**Bài giải.**

**(\*) Phần mã hóa**

Khóa K: **HOANGTHIMAI** 🡪 m = 11

Plaintext: **KHONGBENHLATIENSUONGTUYETTRAN**

Do m = 11, ta sẽ tách thành plaintext thành từng nhóm 11 ký tự:

**KHONGBENHLA** / **TIENSUONGTU** / **YETTRAN**

Viết theo dạng số là:

10 7 14 13 6 1 4 13 7 11 0 / 19 8 4 13 18 20 14 13 6 19 20 / 24 4 19 19 17 0 13

Từ khóa HOANGTHIMAI tương ứng với K = (7, 14, 0, 13, 6, 19, 7, 8, 12, 0, 8)

Cộng từng nhóm 11 ký tự của plaintext với K ta có:

10 7 14 13 6 1 4 13 7 11 0 / 19 8 4 13 18 20 14 13 6 19 20 / 24 4 19 19 17 0 13

7 14 0 13 6 19 7 8 12 0 8 / 7 14 0 13 6 19 7 8 12 0 8 / 7 14 0 13 6 19 7

17 21 14 0 12 20 11 21 19 11 8 / 0 22 4 0 24 13 21 21 18 19 2 / 5 18 19 6 23 19 20

* Ciphertext tương ứng là: RVOAMULVTLI / AWEAYNVVSTC / FSTGXTU

**(\*) Phần giải mã**

Khóa K: HOANGTHIMAI 🡪 m =11

Ciphertext: RVOAMULVTLI / AWEAYNVVSTC / FSTGXTU

Do m = 11, ta sẽ tách ciphertext thành từng nhóm 11 ký tự:

RVOAMULVTLI / AWEAYNVVSTC / FSTGXTU

Viết theo dạng số là:

17 21 14 0 12 20 11 21 19 11 8 / 0 22 4 0 24 13 21 21 18 19 2 / 5 18 19 6 23 19 20

Trừ từng nhóm 16 ký tự của Ciphertext với K ta có:

17 21 14 0 12 20 11 21 19 11 8 / 0 22 4 0 24 13 21 21 18 19 2 / 5 18 19 6 23 19 20

7 14 0 13 6 19 7 8 12 0 8 / 7 14 0 13 6 19 7 8 12 0 8 / 7 14 0 13 6 19 7

10 7 14 13 6 1 4 13 7 11 0 / 19 8 4 13 18 20 14 13 6 19 20 / 24 4 19 19 17 0 13

* Plaintext : KHONGBENHLATIENSUONGTUYETTRAN

**(\*) Code**

A screenshot of a computer program

Description automatically generated

**Bài tập 18.** Mật mã Vernman tay to + Code

Giả sử plaintext là “**TIEN VE HA NOI 1945**” (các ký tự trong chuỗi có thể được biểu diễn dưới dạng nhị phân bảng mã chuẩn ASCII), khóa K là họ và tên sinh viên dưới dạng mã nhị phân. Hãy mã hóa chuỗi ban đầu bằng phương pháp Vernman.

Giải mã chuỗi thu được rồi so sánh với chuỗi ban đầu.

**Phương pháp giải.** C = P xor K

**Bài giải.**

Khóa K = “HOANG THI MAI” = 48 4F 41 4E 47 20 54 48 49 20 4D 41 49

= 0100 1000 0100 1111 0100 0001 0100 1110 0100 0111 0010 0000 0101 0100 0100 1000 0100 1001 0010 0000 0100 1101 0100 0001 0100 1001

PT = “TIEN VE HA NOI 1945” = 54 49 45 4E 20 56 45 20 48 41 20 4E 4F 49 20 31 39 34 35

= 0101 0100 0100 1001 0100 0101 0100 1110 0010 0000 0101 0110 0100 0101 0010 0000 0100 1000 0100 0001 0010 0000 0100 1110 0100 1111 0100 1001 0011 0001 0011 1001 0011 0100 0011 0101

**(\*) Phần mã hóa**

Plaintext = 0101 0100 0100 1001 0100 0101 0100 1110 0010 0000 0101 0110 0100 0101 0010 0000 0100 1000 0100 0001 0010 0000 0100 1110 0100 1111 0100 1001 0010 0000 0011 0001 0011 1001 0011 0100 0011 0101

Khóa K = 0100 1000  0100 1111 0100 0001 0100 1110 0100 0111 0010 0000 0101 0100

0100 1000 0100 1001 0010 0000 0100 1101 0100 0001 0100 1001

Ciphertext = 0001 1100 0000  0110 0000 0100 0000 0000 0110 0111 0111 0110 0001 0001 0110 1000 0000 0001 0110 0001 0110 1101 0000 1111 0000 0110 0000 0001  0110 1111 0111  0000 0111  0111 0111 0011 0001 0101

**(\*) Phần giải mã**

Ciphertext = 0001 1100 0000  0110 0000 0100 0000 0000 0110 0111 0111 0110 0001 0001 0110 1000 0000 0001 0110 0001 0110 1101 0000 1111 0000 0110 0000 0001  0110 1111 0111  0000 0111  0111 0111 0011 0001 0101

Khóa K = 0100 1000  0100 1111 0100 0001 0100 1110 0100 0111 0010 0000 0101 0100

0100 1000 0100 1001 0010 0000 0100 1101 0100 0001 0100 1001

Plaintext = 0101 0100 0100 1001 0100 0101 0100 1110 0010 0000 0101 0110 0100 0101

0010 0000 0100 1000 0100 0001 0010 0000 0100 1110 0100 1111 0100 1001 0010 0000 0011 0001 0011 1001 0011 0100 0011 0101

**(\*) Code**

**A computer screen shot of a code

Description automatically generated**

**Bài tập 19.** Mật mã Affine (mã hóa bằng tay)

Cho chuỗi plaintext sau “**KHOACONGNGHETHONGTIN**”

Hãy mã hóa chuỗi nói trên bằng mật mã Affine, với khóa K là cặp số {a,b} lấy từ 2 số cuối của mã sinh viên. Trong đó, a là số hàng chục (nếu a chẵn thì cộng thêm 1), b là số hàng đơn vị.

Giải mã ciphertext thu được và so sánh với plaintext ban đầu.

**Bài giải.**

K = {a,b} = {3, 9},  inv(a)= 9

P = KHOACONGNGHETHONGTIN

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P\_i | K | H | O | A | C | O | N | G | N | G | H | E | T | H | O | N | G | T | I | N |
|  | 10 | 7 | 14 | 0 | 2 | 14 | 13 | 6 | 13 | 6 | 7 | 4 | 19 | 7 | 14 | 13 | 6 | 19 | 8 | 13 |

**Công thức mã hóa**: C\_i = E(P\_i, {a,b}) = (aP\_i + b) mod 26 = (3P\_i + 9) mod 26

Áp dụng công thức trên ta có

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ci | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 |
|  | 13 | 4 | 25 | 9 | 15 | 25 | 22 | 1 | 22 | 1 | 4 | 21 | 14 | 4 | 25 | 22 | 1 | 14 | 7 | 22 |
|  | N | E | Z | J | P | Z | W | B | W | B | E | V | O | E | Z | W | B | O | H | W |

**Công thức giải mã:** P\_i = D(C\_i, {a,b}) = inv(a)(C\_i – b) mod 26 = 9(C\_i – 9) mod 26

Áp dụng công thức trên ta có

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ci | N | E | Z | J | P | Z | W | B | W | B | E | V | O | E | Z | W | B | O | H | W |
| Pi | 10 | 7 | 14 | 0 | 2 | 14 | 13 | 6 | 13 | 6 | 7 | 4 | 19 | 7 | 14 | 13 | 6 | 19 | 8 | 13 |
|  | K | H | O | A | C | O | N | G | N | G | H | E | T | H | O | N | G | T | I | N |

* KL: Giải mã được kết quả trùng với plaintext ban đầu

**Bài tập 20**. Tính mã Hash 1

Nhập một chuỗi ký tự từ bàn phím, chia chuỗi thành từng khối 64 bit (ứng với 8 ký tự) rồi tính mã hash của chuỗi theo phương pháp:

* Giả sử thông điệp đầu vào được chia thành N khối, mối khối dài m bít
* Mã hash dài m bít được tính bằng cách XOR các bít tương ứng của tất cả các khối: H\_i = b\_i1 XOR b\_i2 XOR ... XOR b\_iN. Trong đó: H\_i là bit thứ i của mã hash, 1 <= i <= m, b\_ij là bit thứ i của khối thứ j.

(Nếu độ dài chuỗi không phải là bội số của 8 thì có thể chèn thêm các ký tự quy ước)

**Bài giải.**

**A computer screen shot of a program

Description automatically generated**

**Bài tập 21.** Tính mã Hash 2

Giả sử thông điệp đầu vào được chia thành N khối, mỗi khối dài m bít. Để cải thiện tính ngẫu nhiên của đầu vào so với bài trên, ta sẽ thực hiện phép quay phải trước khi tiến hành XOR:

* Giá trị hash ban đầu được đặt bằng 0
* Thực hiện N vòng lặp (i từ 0 → N-1), mỗi vòng làm các công việc sau:
  + Quay phải một bít giá trị hash hiện tại
  + XOR giá trị hash với khối dữ liệu thứ i

A computer screen shot of text

Description automatically generated

**Bài tập 22.** Mã hóa Rain-Fence hoán vị cột

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

Description automatically generated

A computer code with text

Description automatically generated

### ✅ 1. ****Cộng****

char F\_add(char R, char K) {

return R + K; // Cộng đơn giản

}

### ✅ 2. ****AND****

char F\_and(char R, char K) {

return R & K; // Phép AND bitwise

}

### ✅ 3. ****NOT****

char F\_not(char R, char K) {

return ~(R ^ K); // Đảo bit của XOR, tăng độ rối

}

### ✅ 4. ****Nhân****

char F\_mul(char R, char K) {

return (R \* K) % 256; // Tránh tràn, giữ trong 8 bit

}

### ✅ 5. ****Modulo tổng****

char F\_mod(char R, char K) {

return (R + K) % 128; // Chỉ lấy kết quả trong khoảng 0–127

}

### ✅ 6. ****XOR kết hợp quay phải****

char F\_xor\_rotate(char R, char K, int round) {

return R ^ ((K >> round) | (K << (8 - round))); // Kết hợp xoay phải

}

### ✅ 7. ****Kết hợp nhiều phép****

char F\_mix(char R, char K, int round) {

char rotK = (K >> round) | (K << (8 - round)); // rotateRight

return ((R + rotK) ^ (R | K)) % 256;

}

### ✅ 8. ****Ngẫu nhiên có kiểm soát (pseudo-random)****

char F\_prng(char R, char K, int round) {

return (R ^ ((K \* round + 31) % 256)) + round;

}

**✅ 1. XOR + Quay phải**

**char F(char R, char Ki) {**

**return R ^ Ki;**

**}**

**char rotateRight(char ch, int i) {**

**return (char)((ch >> i) | (ch << (8 - i)));**

**}**

**✅ 2. XOR + Quay trái**

**char F(char R, char Ki) {**

**return R ^ Ki;**

**}**

**char rotateLeft(char ch, int i) {**

**return (char)((ch << i) | (ch >> (8 - i)));**

**}**

**✅ 3. OR + Quay phải**

**char F(char R, char Ki) {**

**return R | Ki;**

**}**

**char rotateRight(char ch, int i) {**

**return (char)((ch >> i) | (ch << (8 - i)));**

**}**

**✅ 4. AND + Quay trái**

**char F(char R, char Ki) {**

**return R & Ki;**

**}**

**char rotateLeft(char ch, int i) {**

**return (char)((ch << i) | (ch >> (8 - i)));**

**}**

**✅ 5. Cộng + Quay phải**

**char F(char R, char Ki) {**

**return R + Ki;**

**}**

**char rotateRight(char ch, int i) {**

**return (char)((ch >> i) | (ch << (8 - i)));**

**}**

**✅ 6. Nhân + Quay trái (và mod 256 để tránh tràn)**

**char F(char R, char Ki) {**

**return (R \* Ki) % 256;**

**}**

**char rotateLeft(char ch, int i) {**

**return (char)((ch << i) | (ch >> (8 - i)));**

**}**

**✅ 7. Modulo tổng + Quay phải**

**char F(char R, char Ki) {**

**return (R + Ki) % 128;**

**}**

**char rotateRight(char ch, int i) {**

**return (char)((ch >> i) | (ch << (8 - i)));**

**}**

**✅ 8. NOT XOR + Quay trái**

**char F(char R, char Ki) {**

**return ~(R ^ Ki);**

**}**

**char rotateLeft(char ch, int i) {**

**return (char)((ch << i) | (ch >> (8 - i)));**

**}**

**Câu 3 - Đề 5 – Trang 14:**

#**include** <iostream>

#**include** <string>

**using** **namespace** std;

char **F**(char R, char Ki) {

**return** R - Ki;

}

char **rotateRight**(char ch, int i) {

**return** (char)((ch >> i) | (ch << (8 - i)));

}

string **MHKhoi**(char P0, char P1, char k) {

char K[6], L[6], R[6];

string C = " "; // Độ dài 2 ký tự

R[0] = P0;

L[0] = P1;

K[0] = k;

**for** (int i = 1; i <= 5; i++) {

K[i] = rotateRight(K[0], i); // Quay phải i lần

R[i] = L[i - 1] ^ F(R[i - 1], K[i]); // Feistel function

L[i] = R[i - 1];

}

C[0] = L[5];

C[1] = R[5];

**return** C;

}

string **GiaiMaKhoi**(char C0, char C1, char k) {

char K[6], L[6], R[6];

string P = " ";

L[5] = C0;

R[5] = C1;

K[0] = k;

**for** (int i = 1; i <= 5; i++) {

K[i] = rotateRight(K[0], i); // Quay phải i lần

}

**for** (int i = 5; i >= 1; i--) {

R[i - 1] = L[i];

L[i - 1] = R[i] ^ F(R[i - 1], K[i]);

}

P[0] = R[0];

P[1] = L[0];

**return** P;

}

int **main**() {

string P, C, decryptedP;

char k;

cout << "Nhap chuoi plaintext: ";

getline(cin, P);

cout << "Nhap khoa K: ";

cin >> k;

**if** (P.size() % 2 == 1) P += 'X'; // Thêm 'X' nếu độ dài lẻ

**for** (int i = 0; i < P.size(); i += 2) {

C += MHKhoi(P[i], P[i + 1], k);

}

cout << "Chuoi ma hoa: " << C << endl;

**for** (int i = 0; i < C.size(); i += 2) {

decryptedP += GiaiMaKhoi(C[i], C[i + 1], k);

}

**if** (!decryptedP.empty() && decryptedP[decryptedP.size() - 1] == 'X') {

decryptedP.erase(decryptedP.size() - 1); // Bỏ 'X' cuối nếu có

}

cout << "Chuoi giai ma: " << decryptedP << endl;

**return** 0;

}

**Câu 3 – Đề 6 – T15:**

#**include** <iostream>

#**include** <string>

**using** **namespace** std;

// Hàm F đơn giản dùng phép XOR thay vì phép trừ

char **F**(char R, char Ki) {

**return** R ^ Ki; // XOR

}

// Hàm quay phải ký tự k đi n bit

char **rotateRight**(char k, int n) {

**return** (k >> n) | (k << (8 - n)); // Quay phải n lần

}

// Mã hóa một khối gồm 2 ký tự P0, P1 với khóa k

string **MHKhoi**(char P0, char P1, char k) {

char K[8], L[8], R[8];

string C = " "; // 2 ký tự

// Khởi tạo

R[0] = P0;

L[0] = P1;

K[0] = k;

// Tạo 7 khóa con và thực hiện 7 vòng Feistel

**for** (int i = 1; i <= 7; i++) {

K[i] = rotateRight(K[0], i); // Quay phải i lần tạo khóa con

R[i] = L[i - 1] ^ F(R[i - 1], K[i]); // R mới

L[i] = R[i - 1]; // L mới

}

// Kết quả sau 7 vòng: L7 và R7 là ciphertext

C[0] = L[7];

C[1] = R[7];

**return** C;

}

// Giải mã một khối gồm 2 ký tự C0, C1 với khóa k

string **GiaiMaKhoi**(char C0, char C1, char k) {

char K[8], L[8], R[8];

string P = " "; // plaintext 2 ký tự

// Gán lại giá trị L7, R7 từ ciphertext

L[7] = C0;

R[7] = C1;

K[0] = k;

// Tạo khóa con

**for** (int i = 1; i <= 7; i++) {

K[i] = rotateRight(K[0], i);

}

// Giải mã ngược từ vòng 7 → 1

**for** (int i = 7; i >= 1; i--) {

R[i - 1] = L[i]; // R trước = L sau

L[i - 1] = R[i] ^ F(R[i - 1], K[i]); // L trước

}

// Kết quả ban đầu

P[0] = R[0];

P[1] = L[0];

**return** P;

}

int **main**() {

string P, C, decryptedP;

char k;

cout << "Nhap chuoi plaintext: ";

getline(cin, P);

cout << "Nhap khoa K: ";

cin >> k;

// Nếu độ dài lẻ → thêm 'X'

**if** (P.size() % 2 == 1) P += 'X';

// Mã hóa từng cặp ký tự

**for** (int i = 0; i < P.size(); i += 2) {

C += MHKhoi(P[i], P[i + 1], k);

}

cout << "Chuoi ma hoa: " << C << endl;

// Giải mã từng cặp ký tự

**for** (int i = 0; i < C.size(); i += 2) {

decryptedP += GiaiMaKhoi(C[i], C[i + 1], k);

}

// Xoá ký tự 'X' nếu có (dùng để đệm)

**if** (!decryptedP.empty() && decryptedP[decryptedP.size() - 1] == 'X') {

decryptedP.erase(decryptedP.size() - 1);

}

cout << "Chuoi giai ma: " << decryptedP << endl;

**return** 0;

}

**Câu 2 – Đề 7: So sánh sự giống và khác nhau của Hash và MAC? Hàm nào sử dụng chữ ký số tốt hơn? Tại sao?**

**Giống:**

* **Tính toán giá trị cố định:** Cả hàm Hash và hàm MAC đều nhận đầu vào có độ dài bất kỳ và trả về một giá trị cố định đã định.
* **Đảm bảo tính toàn vẹn:** Cả hai đều dùng để đảm bảo tính toàn vẹn của dữ liệu, nghĩa là một thay đổi nhỏ trong đầu vào sẽ dẫn đến sự thay đổi lớn trong giá trị đầu ra.
* **Ứng dụng trong bảo mật:** Cả hai đều là công cụ mật mã học quan trọng và được sử dụng rộng rãi trong các ứng dụng bảo mật hiện đại.

|  | **Mac** | **Hash** |
| --- | --- | --- |
| **Khóa bí mật** | Sử dụng khóa bí mật cùng với đầu vào để tính toán giá trị MAC, đảm bảo cả tính toàn vẹn và xác thực nguồn gốc dữ liệu. | Không sử dụng khóa bí mật. Giá trị được tính toán chỉ dựa vào dữ liệu. |
| **Mục đích sử dụng** | Dùng để đảm bảo tính toàn vẹn và xác thực nguồn gốc dữ liệu, thường được sử dụng trong truyền thông mạng và các giao thức bảo mật. | Chủ yếu để kiểm tra tính toàn vẹn của dữ liệu. Dùng trong lưu trữ mật khẩu, chữ ký số, và kiểm tra tính toàn vẹn của các tập tin. |

### ****Hàm Hash sử dụng tốt hơn cho chữ ký số:****

1. **Tính toàn vẹn và hiệu quả:**  
   Hàm hash nhanh chóng tạo ra một giá trị băm nhỏ gọn đại diện cho dữ liệu, giúp phát hiện thay đổi trong dữ liệu một cách hiệu quả.
2. **Hệ thống khóa công khai:**  
   Chữ ký số dựa trên hàm băm giá trị băm bằng khóa riêng và xác minh bằng khóa công khai, phù hợp với cách hoạt động của hàm hash.
3. **Không yêu cầu khóa bí mật:**  
   Hàm hash không cần khóa bí mật để tính toán giá trị, đơn giản hóa quá trình tạo và xác minh chữ ký số.

**Câu 3 – Đề 7:** Giống 2 đề ở trên, chỉ khác ở chỗ: Số vòng xử lý là n=4. Hàm F thực hiện phép OR

// ----------------- feistel OR -------------------

#**include** <iostream>

#**include** <string>

**using** **namespace** std;

// Hàm F sử dụng phép OR giữa R và Ki

char **F**(char R, char Ki) {

**return** R | Ki; // OR logic

}

// Hàm quay phải 1 ký tự k đi n bit

char **rotateRight**(char k, int n) {

**return** (k >> n) | (k << (8 - n)); // Quay phải n lần

}

// Mã hóa một khối gồm 2 ký tự với khóa k

string **MHKhoi**(char P0, char P1, char k) {

char K[5], L[5], R[5]; // Dùng 4 vòng → mảng 5 phần tử

string C = " ";

R[0] = P0;

L[0] = P1;

K[0] = k;

// 4 vòng mã hóa

**for** (int i = 1; i <= 4; i++) {

K[i] = rotateRight(K[0], i); // Tạo khóa con bằng quay phải

R[i] = L[i - 1] ^ F(R[i - 1], K[i]); // R mới = L cũ ^ F(R cũ, K con)

L[i] = R[i - 1]; // L mới = R cũ

}

C[0] = L[4];

C[1] = R[4];

**return** C;

}

// Giải mã một khối gồm 2 ký tự với khóa k

string **GiaiMaKhoi**(char C0, char C1, char k) {

char K[5], L[5], R[5];

string P = " ";

L[4] = C0;

R[4] = C1;

K[0] = k;

// Tạo khóa con như khi mã hóa

**for** (int i = 1; i <= 4; i++) {

K[i] = rotateRight(K[0], i);

}

// Giải mã ngược từ vòng 4 về 1

**for** (int i = 4; i >= 1; i--) {

R[i - 1] = L[i];

L[i - 1] = R[i] ^ F(R[i - 1], K[i]);

}

P[0] = R[0];

P[1] = L[0];

**return** P;

}

int **main**() {

string P, C, decryptedP;

char k;

cout << "Nhap chuoi plaintext: ";

getline(cin, P);

cout << "Nhap khoa K: ";

cin >> k;

// Nếu độ dài lẻ → thêm 'X' để đủ cặp

**if** (P.size() % 2 == 1) P += 'X';

// Mã hóa từng cặp ký tự

**for** (int i = 0; i < P.size(); i += 2) {

C += MHKhoi(P[i], P[i + 1], k);

}

cout << "Chuoi ma hoa: " << C << endl;

// Giải mã

**for** (int i = 0; i < C.size(); i += 2) {

decryptedP += GiaiMaKhoi(C[i], C[i + 1], k);

}

// Bỏ 'X' nếu là ký tự đệm

**if** (!decryptedP.empty() && decryptedP[decryptedP.size() - 1] == 'X') {

decryptedP.erase(decryptedP.size() - 1);

}

cout << "Chuoi giai ma: " << decryptedP << endl;

**return** 0;

}