**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: NHẬP MÔN AN TOÀN BẢO MẬT THÔNG TIN**

**ĐỀ TÀI: GAME “GIẢI MÃ KHO BÁO”**

**Giáo viên hướng dẫn: TS. Trần Đăng Công**

**Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Stt** | **Mã sv** | **Họ và tên** | **Lớp** |
| 1 | 1771020256 | Vũ Tuấn Hiệp | CNTT 17-01 |
| 2 | 1771020311 | Trần Hậu Huân | CNTT 17-01 |
| 3 | 1771020092 | Nguyễn Thanh Bình | CNTT 17-01 |

**Hà Nội, năm 2024**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: NHẬP MÔN AN TOÀN BẢO MẬT THÔNG TIN**

**ĐỀ TÀI: GAME “GIẢI MÃ KHO BÁU”**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Mã Sinh Viên | Họ và Tên | Ngày Sinh | Điểm | |
| Bằng Số | Bằng Chữ |
| 1 | 1771020256 | Vũ Tuấn Hiệp | 13/8/2005 |  |  |
| 2 | **1771020311** | **Trần Hậu Huân** | **05/04/2005** |  |  |
| 3 | **1771020092** | **Nguyễn Thanh Bình** | 05/02/2005 |  |  |

CÁN BỘ CHẤM THI 1 CÁN BỘ CHẤM THI 2

**Hà Nội, năm 2024**

**LỜI NÓI ĐẦU**

Trong thời đại công nghệ số phát triển mạnh mẽ, an toàn bảo mật thông tin đã trở thành một trong những lĩnh vực quan trọng và cấp thiết, ảnh hưởng trực tiếp đến mọi khía cạnh của đời sống, từ cá nhân, doanh nghiệp đến các tổ chức chính phủ. Môn học "Nhập môn An toàn Bảo mật Thông tin" được thiết kế nhằm cung cấp cho sinh viên những kiến thức nền tảng về các khái niệm, nguyên lý cơ bản và các kỹ thuật bảo đảm an toàn thông tin trong môi trường số.

Khóa học sẽ giới thiệu các nội dung cốt lõi như mã hóa, xác thực, quản lý truy cập, bảo vệ dữ liệu, cũng như các mối đe dọa an ninh mạng phổ biến như tấn công mạng, phần mềm độc hại và lừa đảo trực tuyến. Bên cạnh đó, môn học cũng nhấn mạnh vai trò của ý thức bảo mật và các quy định pháp lý trong việc xây dựng một môi trường số an toàn.

Với sự kết hợp giữa lý thuyết và thực hành, môn học không chỉ trang bị kiến thức mà còn giúp sinh viên phát triển kỹ năng phân tích, đánh giá và giải quyết các vấn đề liên quan đến an toàn thông tin. Đây sẽ là nền tảng vững chắc để các bạn tiếp tục khám phá các lĩnh vực chuyên sâu hơn trong ngành bảo mật thông tin, đồng thời nâng cao ý thức trách nhiệm trong việc bảo vệ thông tin cá nhân và tổ chức.

Hy vọng rằng môn học này sẽ khơi dậy niềm đam mê và sự tò mò của các bạn đối với lĩnh vực an toàn bảo mật thông tin, đồng thời tạo động lực để các bạn trở thành những chuyên gia trong việc xây dựng một thế giới số an toàn và đáng tin cậy.

**MỤC LỤC**

**[LỜI NÓI ĐẦU](#_Toc202278451)** [3](#_Toc202278451)

**[CHƯƠNG 1: THUẬT TOÁN CAESAR CIPHER 5](#_Toc202278452)**

[1.1 Giới thiệu Caesar Cipher 5](#_Toc202278453)

[1.2 Nguyên lý hoạt động 5](#_Toc202278454)

[1.3 Thuật toán mã hóa và giải mã 5](#_Toc202278455)

[1.3.1 Thuật toán mã hóa 5](#_Toc202278456)

[1.3.2 Thuật toán giải mã 6](#_Toc202278457)

[1.4 Ví dụ minh họa 7](#_Toc202278458)

[1.5 Ưu điểm và nhược điểm 8](#_Toc202278459)

[1.5.1 Ưu điểm 8](#_Toc202278460)

[1.5.2 Nhược điểm 8](#_Toc202278461)

[1.6 Ứng dụng trong trò chơi 8](#_Toc202278462)

[1.6.1 Thiết kế cấp độ 8](#_Toc202278463)

[1.6.2 Giao diện người chơi 8](#_Toc202278464)

[1.6.3 Hệ thống đánh giá 9](#_Toc202278465)

**[CHƯƠNG 2: THUẬT TOÁN VIGENÈRE CIPHER 10](#_Toc202278466)**

[2.1 Giới thiệu Vigenère Cipher 10](#_Toc202278467)

[2.2 Nguyên lý hoạt động 10](#_Toc202278468)

[2.3 Thuật toán mã hóa và giải mã 10](#_Toc202278469)

[2.3.1 Thuật toán mã hóa 10](#_Toc202278470)

[2.3.2 Thuật toán giải mã 11](#_Toc202278471)

[2.4 Ví dụ minh họa 12](#_Toc202278472)

[2.5 Phân tích độ an toàn 13](#_Toc202278473)

[2.5.1 Ưu điểm về bảo mật 13](#_Toc202278474)

[2.5.2 Nhược điểm và điểm yếu 13](#_Toc202278475)

[2.5.3 Các phương pháp tấn công 13](#_Toc202278476)

[2.6 Ứng dụng trong trò chơi 14](#_Toc202278477)

[2.6.1 Thiết kế progression 14](#_Toc202278478)

[2.6.2 Công cụ hỗ trợ người chơi 14](#_Toc202278479)

[2.6.3 Hệ thống gợi ý phân tầng 15](#_Toc202278480)

**[CHƯƠNG 3: THUẬT TOÁN RSA 16](#_Toc202278481)**

[3.1 Giới thiệu RSA 16](#_Toc202278482)

[3.2 Cơ sở toán học 16](#_Toc202278483)

[3.2.1 Lý thuyết số cơ bản 16](#_Toc202278484)

[3.2.2 Bài toán phân tích thừa số 17](#_Toc202278485)

[3.3 Thuật toán tạo khóa 17](#_Toc202278486)

[3.3.1 Quy trình tạo cặp khóa RSA 17](#_Toc202278487)

[3.3.2 Ví dụ tạo khóa với số nhỏ 18](#_Toc202278488)

[3.4 Thuật toán mã hóa và giải mã 18](#_Toc202278489)

[3.4.1 Quá trình mã hóa 18](#_Toc202278490)

[3.4.2 Quá trình giải mã 18](#_Toc202278491)

[3.4.3 Ví dụ mã hóa/giải mã 19](#_Toc202278492)

[3.5 Tối ưu hóa và cải tiến 19](#_Toc202278493)

[3.5.1 Thuật toán lũy thừa nhanh 19](#_Toc202278494)

[3.5.2 Đệm (Padding Schemes) 20](#_Toc202278495)

[3.6 Tính bảo mật và ứng dụng 21](#_Toc202278496)

[3.6.1 Tính bảo mật của RSA 21](#_Toc202278497)

[3.6.2 Ứng dụng của RSA 22](#_Toc202278498)

[3.7 Ứng dụng trong trò chơi 23](#_Toc202278499)

[3.7.1 Thiết kế cấp độ 23](#_Toc202278500)

[3.7.2 Giao diện người chơi 24](#_Toc202278501)

[3.7.3 Hệ thống đánh giá 24](#_Toc202278502)

**[CHƯƠNG 4: THUẬT TOÁN AES 25](#_Toc202278503)**

[4.1 Giới thiệu AES 25](#_Toc202278504)

[4.2 Nguyên lý hoạt động 26](#_Toc202278505)

[4.3 Thuật toán mã hóa và giải mã 27](#_Toc202278506)

[4.3.1 Thuật toán mã hóa 27](#_Toc202278507)

[4.3.2 Thuật toán giải mã 28](#_Toc202278508)

[4.4 Ví dụ minh họa 29](#_Toc202278509)

[4.5 Ưu điểm và nhược điểm 30](#_Toc202278510)

[4.5.1 Ưu điểm 30](#_Toc202278511)

[4.5.2 Nhược điểm 30](#_Toc202278512)

[4.6 Ứng dụng trong trò chơi 31](#_Toc202278513)

[4.6.1 Thiết kế cấp độ 31](#_Toc202278514)

[4.6.2 Giao diện người chơi 32](#_Toc202278515)

[4.6.3 Hệ thống đánh giá 32](#_Toc202278516)

**[CHƯƠNG 5: GIỚI THIỆU ĐỀ TÀI - GAME “GIẢI MÃ KHO BÁU” 32](#_Toc202278517)**

[5.1 Mục tiêu của đề tài 32](#_Toc202278518)

[5.2 Ý nghĩa của đề tài 33](#_Toc202278519)

[5.3 Phạm vi của đề tài 34](#_Toc202278520)

[5.4 Cấu trúc của trò chơi 34](#_Toc202278521)

[5.4.1 Cốt truyện và bối cảnh 34](#_Toc202278522)

[5.4.2 Các cấp độ trong trò chơi 35](#_Toc202278523)

[5.4.3 Giao diện và công cụ hỗ trợ 35](#_Toc202278524)

[5.5 Tích hợp các thuật toán mật mã học 35](#_Toc202278525)

[5.6 Kết quả kỳ vọng 36](#_Toc202278526)

[5.7 Kết luận 36](#_Toc202278527)

**[CHƯƠNG 6: PHÂN TÍCH ĐỀ TÀI 37](#_Toc202278528)**

[6.1 Tổng quan về đề tài 37](#_Toc202278529)

[6.2 Mục tiêu và yêu cầu chức năng 38](#_Toc202278530)

[6.2.1 Mục tiêu 38](#_Toc202278531)

[6.2.2 Yêu cầu chức năng 38](#_Toc202278532)

[6.3 Thiết kế hệ thống 39](#_Toc202278533)

[6.3.1 Kiến trúc tổng thể 39](#_Toc202278534)

[6.3.2 Tích hợp thuật toán mật mã 39](#_Toc202278535)

[6.3.3 Cơ sở dữ liệu 40](#_Toc202278536)

[6.4 Phân tích mã nguồn 40](#_Toc202278537)

[6.4.1 File cipher\_utils.py 40](#_Toc202278538)

[6.4.2 File app.py 41](#_Toc202278539)

[6.4.3 File index.html 41](#_Toc202278540)

[6.5 Đánh giá ưu điểm và hạn chế 42](#_Toc202278541)

[6.5.1 Ưu điểm 42](#_Toc202278542)

[6.5.2 Hạn chế 42](#_Toc202278543)

[Chương 7: Kết luận 43](#_Toc202278544)

[7.1 Điểm mạnh 43](#_Toc202278545)

[7.2 Điểm yếu 44](#_Toc202278546)

[7.3 Hướng cải tiến và cải thiện 44](#_Toc202278547)

[7.4 Hướng phát triển 45](#_Toc202278548)

# CHƯƠNG 1: THUẬT TOÁN CAESAR CIPHER

## 1.1 Giới thiệu Caesar Cipher

Caesar Cipher, còn được gọi là mã dịch chuyển (shift cipher), là một trong những phương pháp mã hóa cổ điển và đơn giản nhất trong lịch sử mật mã học. Thuật toán này được đặt tên theo Gaius Julius Caesar, hoàng đế La Mã nổi tiếng, người đã sử dụng phương pháp này để bảo vệ các thông tin quân sự quan trọng trong thời kỳ chinh phạt Gaul (khoảng 50-58 TCN).

Về bản chất, Caesar Cipher là một dạng mã hóa thay thế đơn âm (monoalphabetic substitution cipher), trong đó mỗi chữ cái trong văn bản gốc được thay thế bằng một chữ cái khác cách nó một khoảng cố định trong bảng chữ cái. Đây là một ví dụ điển hình của mã hóa đối xứng, trong đó cùng một khóa được sử dụng cho cả quá trình mã hóa và giải mã.

## 1.2 Nguyên lý hoạt động

Nguyên lý cơ bản của Caesar Cipher dựa trên việc dịch chuyển các chữ cái trong bảng chữ cái theo một hướng cố định. Thuật toán hoạt động theo các bước sau:

Bước 1: Thiết lập khóa Khóa trong Caesar Cipher là một số nguyên k (thường từ 1 đến 25 đối với bảng chữ cái tiếng Anh), biểu thị số vị trí dịch chuyển. Theo truyền thống, Caesar sử dụng k = 3.

Bước 2: Ánh xạ bảng chữ cái Mỗi chữ cái được gán một giá trị số từ 0 đến 25 (A=0, B=1, C=2, ..., Z=25).

Bước 3: Thực hiện dịch chuyển Đối với mã hóa: mỗi chữ cái x được thay thế bằng chữ cái ở vị trí (x + k) mod 26. Đối với giải mã: mỗi chữ cái y được thay thế bằng chữ cái ở vị trí (y - k) mod 26.

## 1.3 Thuật toán mã hóa và giải mã

### 1.3.1 Thuật toán mã hóa

Đầu vào: Văn bản gốc P, khóa k

Đầu ra: Văn bản mã hóa C

FOR mỗi ký tự p\_i trong P DO

     IF p\_i là chữ cái THEN

         Chuyển p\_i thành số (A=0, B=1, ..., Z=25)

         c\_i = (p\_i + k) mod 26

         Chuyển c\_i thành chữ cái tương ứng

     ELSE

         c\_i = p\_i (giữ nguyên ký tự không phải chữ cái)

     END IF

END FOR

### 1.3.2 Thuật toán giải mã

Đầu vào: Văn bản mã hóa C, khóa k

Đầu ra: Văn bản gốc P

FOR mỗi ký tự c\_i trong C DO

     IF c\_i là chữ cái THEN

        Chuyển c\_i thành số (A=0, B=1, ..., Z=25)

        p\_i = (c\_i - k) mod 26

        Chuyển p\_i thành chữ cái tương ứng

    ELSE

        p\_i = c\_i (giữ nguyên ký tự không phải chữ cái)

    END IF

END FOR

## 1.4 Ví dụ minh họa

Ví dụ 1: Mã hóa cơ bản

* Văn bản gốc: "HELLO WORLD"
* Khóa: k = 3
* Quá trình mã hóa:
  + H (7) → K (10)
  + E (4) → H (7)
  + L (11) → O (14)
  + L (11) → O (14)
  + O (14) → R (17)
  + W (22) → Z (25)
  + O (14) → R (17)
  + R (17) → U (20)
  + L (11) → O (14)
  + D (3) → G (6)
* Văn bản mã hóa: "KHOOR ZRUOG"

Ví dụ 2: Xử lý trường hợp wrap-around

* Văn bản gốc: "XYZ"
* Khóa: k = 3
  + Quá trình mã hóa:
    - X (23) → A ((23+3) mod 26 = 0)
    - Y (24) → B ((24+3) mod 26 = 1)
    - Z (25) → C ((25+3) mod 26 = 2)
* Văn bản mã hóa: "ABC"

## 1.5 Ưu điểm và nhược điểm

### 1.5.1 Ưu điểm

* Đơn giản: Dễ hiểu và triển khai, không cần kiến thức toán học phức tạp
* Nhanh chóng: Tốc độ xử lý rất nhanh, phù hợp cho các ứng dụng thời gian thực
* Không yêu cầu tài nguyên: Không cần bộ nhớ hay sức mạnh tính toán lớn
* Giáo dục: Excellent cho việc giảng dạy các khái niệm cơ bản về mật mã học

### 1.5.2 Nhược điểm

* Bảo mật thấp: Chỉ có 25 khóa có thể (cho bảng chữ cái tiếng Anh), dễ bị tấn công brute force
* Dễ phân tích: Dễ bị phá bằng phân tích tần suất ký tự
* Không ẩn pattern: Không thay đổi cấu trúc của văn bản gốc
* Không phù hợp thực tế: Không đủ an toàn cho các ứng dụng hiện đại

## 1.6 Ứng dụng trong trò chơi

Trong game "Giải mã kho báu", Caesar Cipher được sử dụng như là cấp độ đầu tiên - cấp độ học việc để người chơi làm quen với các khái niệm cơ bản về mật mã học.

### 1.6.1 Thiết kế cấp độ

* Cấp độ 1: Giải mã đơn giản với k = 3 (theo truyền thống Caesar)
* Cấp độ 2: Người chơi phải thử các giá trị k khác nhau
* Cấp độ 3: Tấn công brute force - thử tất cả 25 khóa có thể

### 1.6.2 Giao diện người chơi

* Hiển thị văn bản mã hóa
* Cho phép người chơi nhập giá trị k
* Hiển thị kết quả giải mã ngay lập tức
* Cung cấp gợi ý về phân tích tần suất

### 1.6.3 Hệ thống đánh giá

* Điểm cơ bản cho việc giải mã đúng
* Điểm thưởng cho việc giải nhanh
* Giảm điểm cho mỗi lần thử sai

# CHƯƠNG 2: THUẬT TOÁN VIGENÈRE CIPHER

## 2.1 Giới thiệu Vigenère Cipher

Vigenère Cipher, được phát triển bởi nhà ngoại giao và mật mã học người Pháp Blaise de Vigenère vào thế kỷ 16, đại diện cho một bước tiến quan trọng trong lịch sử mật mã học. Khác với Caesar Cipher sử dụng một khóa dịch chuyển cố định, Vigenère Cipher sử dụng một từ khóa (keyword) để tạo ra một chuỗi các khóa dịch chuyển khác nhau, tạo nên tính phức tạp và bảo mật cao hơn đáng kể.

Thuật toán này được coi là "le chiffre indéchiffrable" (mã không thể giải được) trong hơn 300 năm cho đến khi Friedrich Kasiski phát triển phương pháp phá mã vào năm 1863. Vigenère Cipher là một dạng mã hóa thay thế đa âm (polyalphabetic substitution cipher), trong đó cùng một chữ cái có thể được mã hóa thành các chữ cái khác nhau tùy thuộc vào vị trí của nó trong văn bản.

## 2.2 Nguyên lý hoạt động

Nguyên lý cơ bản của Vigenère Cipher dựa trên việc sử dụng một từ khóa để tạo ra một chuỗi các Caesar Cipher khác nhau. Quá trình hoạt động như sau:

Bước 1: Chuẩn bị từ khóa Từ khóa được lặp lại để có độ dài bằng với văn bản cần mã hóa. Ví dụ, nếu từ khóa là "KEY" và văn bản là "ATTACKATDAWN", từ khóa mở rộng sẽ là "KEYKEYKEYKEYKEY".

Bước 2: Chuyển đổi thành số Cả văn bản và từ khóa đều được chuyển đổi thành các giá trị số (A=0, B=1, ..., Z=25).

Bước 3: Thực hiện phép toán modulo Mỗi ký tự được mã hóa bằng cách áp dụng công thức tương tự Caesar Cipher nhưng với khóa thay đổi theo từ khóa.

## 2.3 Thuật toán mã hóa và giải mã

### 2.3.1 Thuật toán mã hóa

Đầu vào: Văn bản gốc P, từ khóa K

Đầu ra: Văn bản mã hóa C

Bước 1: Chuẩn hóa từ khóa

expanded\_key = lặp lại K để có độ dài bằng P

Bước 2: Mã hóa từng ký tự

FOR i = 0 to length(P)-1 DO

     IF P[i] là chữ cái THEN

         p\_val = giá trị số của P[i] (A=0, B=1, ..., Z=25)

         k\_val = giá trị số của expanded\_key[i]

         c\_val = (p\_val + k\_val) mod 26

         C[i] = chữ cái tương ứng với c\_val

     ELSE

         C[i] = P[i] (giữ nguyên ký tự không phải chữ cái)

     END IF

END FOR

### 2.3.2 Thuật toán giải mã

Đầu vào: Văn bản mã hóa C, từ khóa K

Đầu ra: Văn bản gốc P

Bước 1: Chuẩn hóa từ khóa

expanded\_key = lặp lại K để có độ dài bằng C

Bước 2: Giải mã từng ký tự

FOR i = 0 to length(C)-1 DO

     IF C[i] là chữ cái THEN

         c\_val = giá trị số của C[i] (A=0, B=1, ..., Z=25)

         k\_val = giá trị số của expanded\_key[i]

         p\_val = (c\_val - k\_val + 26) mod 26

         P[i] = chữ cái tương ứng với p\_val

     ELSE

         P[i] = C[i] (giữ nguyên ký tự không phải chữ cái)

     END IF

END FOR

## 2.4 Ví dụ minh họa

Ví dụ chi tiết: Mã hóa văn bản "ATTACKATDAWN" với từ khóa "KEY"

Bước 1: Chuẩn bị

* Văn bản gốc: A T T A C K A T D A W N
* Từ khóa gốc: K E Y
* Từ khóa mở rộng: K E Y K E Y K E Y K E Y

Bước 2: Chuyển đổi thành số

* Văn bản: [0, 19, 19, 0, 2, 10, 0, 19, 3, 0, 22, 13]
* Từ khóa: [10, 4, 24, 10, 4, 24, 10, 4, 24, 10, 4, 24]

Bước 3: Mã hóa từng ký tự

* A + K: (0 + 10) mod 26 = 10 → K
* T + E: (19 + 4) mod 26 = 23 → X
* T + Y: (19 + 24) mod 26 = 17 → R
* A + K: (0 + 10) mod 26 = 10 → K
* C + E: (2 + 4) mod 26 = 6 → G
* K + Y: (10 + 24) mod 26 = 8 → I
* A + K: (0 + 10) mod 26 = 10 → K
* T + E: (19 + 4) mod 26 = 23 → X
* D + Y: (3 + 24) mod 26 = 1 → B
* A + K: (0 + 10) mod 26 = 10 → K
* W + E: (22 + 4) mod 26 = 0 → A
* N + Y: (13 + 24) mod 26 = 11 → L

Kết quả: "KXRKGIKXBKAL"

## 2.5 Phân tích độ an toàn

### 2.5.1 Ưu điểm về bảo mật

* Kháng phân tích tần suất: Do cùng một chữ cái có thể được mã hóa thành các chữ cái khác nhau, phân tích tần suất truyền thống trở nên khó khăn
* Khóa dài: Với từ khóa dài, số lượng khóa có thể rất lớn
* Phân tán pattern: Các pattern trong văn bản gốc được che giấu hiệu quả

### 2.5.2 Nhược điểm và điểm yếu

* Kasiski examination: Nếu từ khóa ngắn, các pattern lặp lại trong văn bản có thể tiết lộ độ dài từ khóa
* Index of coincidence: Phân tích thống kê có thể xác định độ dài từ khóa
* Phụ thuộc vào từ khóa: Nếu từ khóa bị lộ hoặc có thể đoán được, toàn bộ hệ thống bị compromised

### 2.5.3 Các phương pháp tấn công

1. Tấn công Kasiski

* Tìm các đoạn văn bản lặp lại
* Tính khoảng cách giữa các lần lặp
* Sử dụng ước chung lớn nhất để xác định độ dài từ khóa

2. Phân tích Index of Coincidence

* Tính toán độ trùng lặp ký tự cho các độ dài từ khóa khác nhau
* So sánh với giá trị mong đợi cho ngôn ngữ tự nhiên

## 2.6 Ứng dụng trong trò chơi

### 2.6.1 Thiết kế progression

Vigenère Cipher được đặt ở cấp độ trung bình của trò chơi, sau khi người chơi đã thành thạo Caesar Cipher:

Cấp độ 4: Giới thiệu Vigenère với từ khóa đơn giản

* Từ khóa ngắn (3-4 ký tự)
* Cung cấp từ khóa trực tiếp
* Văn bản ngắn để dễ xử lý

Cấp độ 5: Thử thách với từ khóa ẩn

* Người chơi phải đoán từ khóa dựa trên gợi ý
* Từ khóa có thể là tên địa điểm, nhân vật trong cốt truyện

Cấp độ 6: Tấn công mật mã học

* Văn bản dài hơn
* Không cung cấp từ khóa
* Người chơi phải sử dụng các kỹ thuật phân tích

### 2.6.2 Công cụ hỗ trợ người chơi

* Frequency analyzer: Hiển thị tần suất xuất hiện của các ký tự
* Pattern detector: Tìm các pattern lặp lại trong văn bản
* Key length guesser: Gợi ý về độ dài có thể của từ khóa
* Keyword dictionary: Cơ sở dữ liệu các từ khóa phổ biến

### 2.6.3 Hệ thống gợi ý phân tầng

1. Gợi ý cơ bản: "Hãy tìm các đoạn văn bản lặp lại"
2. Gợi ý trung bình: "Khoảng cách giữa các pattern có thể cho biết độ dài từ khóa"
3. Gợi ý nâng cao: "Thử phân tích từng nhóm ký tự theo chu kỳ của từ khóa"

# CHƯƠNG 3: THUẬT TOÁN RSA

## 3.1 Giới thiệu RSA

RSA (Rivest-Shamir-Adleman) là một trong những thuật toán mã hóa công khai quan trọng nhất trong lịch sử mật mã học hiện đại. Được phát triển vào năm 1977 bởi Ron Rivest, Adi Shamir và Leonard Adleman tại MIT, RSA đánh dấu một cuộc cách mạng trong lĩnh vực bảo mật thông tin bằng cách giải quyết vấn đề phân phối khóa - một trong những thách thức lớn nhất của các hệ thống mã hóa truyền thống.

Điểm đột phá của RSA nằm ở việc sử dụng một cặp khóa bất đối xứng: khóa công khai (public key) và khóa riêng tư (private key). Khóa công khai có thể được chia sẻ công khai mà không ảnh hưởng đến tính bảo mật, trong khi khóa riêng tư được giữ bí mật tuyệt đối. Tính bảo mật của RSA dựa trên độ khó của việc phân tích một số nguyên lớn thành các thừa số nguyên tố - một bài toán được chứng minh là rất khó giải với các công nghệ tính toán hiện tại.

## 3.2 Cơ sở toán học

### 3.2.1 Lý thuyết số cơ bản

Định lý Euler Đối với hai số nguyên dương a và n, nếu gcd(a,n) = 1, thì: a^φ(n) ≡ 1 (mod n)

Trong đó φ(n) là hàm Euler's totient, đếm số lượng các số nguyên dương nhỏ hơn n và nguyên tố cùng nhau với n.

Hàm Euler's Totient

* Nếu n = p × q (với p, q là số nguyên tố), thì φ(n) = (p-1)(q-1)
* Đây là cơ sở cho việc tính toán khóa trong RSA

Định lý Fermat nhỏ Nếu p là số nguyên tố và a không chia hết cho p, thì: a^(p-1) ≡ 1 (mod p)

### 3.2.2 Bài toán phân tích thừa số

Tính bảo mật của RSA dựa trên độ khó của việc phân tích một số nguyên lớn n = p × q thành hai thừa số nguyên tố p và q. Với các số có độ dài 2048 bit trở lên, việc phân tích này đòi hỏi thời gian tính toán hàng triệu năm với công nghệ hiện tại**.**

## 3.3 Thuật toán tạo khóa

### 3.3.1 Quy trình tạo cặp khóa RSA

Bước 1: Chọn hai số nguyên tố lớn

- Chọn hai số nguyên tố p và q khác nhau

- p và q nên có độ dài xấp xỉ nhau (để tối ưu bảo mật)

- p và q nên đủ lớn (ít nhất 1024 bit mỗi số)

Bước 2: Tính n và φ(n)

- n = p × q (modulus công khai)

- φ(n) = (p-1) × (q-1) (Euler's totient của n)

Bước 3: Chọn số mũ công khai e

- Chọn e sao cho 1 < e < φ(n) và gcd(e, φ(n)) = 1

- Thường chọn e = 65537 (2^16 + 1) vì hiệu quả tính toán

Bước 4: Tính số mũ riêng tư d

- Tính d sao cho (e × d) ≡ 1 (mod φ(n))

- d là nghịch đảo modular của e theo φ(n)

- Sử dụng thuật toán Euclidean mở rộng

Bước 5: Hình thành khóa

- Khóa công khai: (n, e)

- Khóa riêng tư: (n, d)

- Lưu ý: p, q, và φ(n) phải được giữ bí mật tuyệt đối

### 3.3.2 Ví dụ tạo khóa với số nhỏ

Bước 1: Chọn p = 17, q = 19 Bước 2: n = 17 × 19 = 323, φ(n) = 16 × 18 = 288 Bước 3: Chọn e = 5 (vì gcd(5, 288) = 1) Bước 4: Tìm d sao cho 5d ≡ 1 (mod 288)

* Sử dụng thuật toán Euclidean mở rộng: d = 173
* Kiểm tra: 5 × 173 = 865 = 3 × 288 + 1 ≡ 1 (mod 288) ✓ Bước 5:
* Khóa công khai: (323, 5)
* Khóa riêng tư: (323, 173)

## 3.4 Thuật toán mã hóa và giải mã

### 3.4.1 Quá trình mã hóa

Đầu vào: Thông điệp M, khóa công khai (n, e)

Đầu ra: Bản mã C

Điều kiện: 0 ≤ M < n

Thuật toán:

C = M^e mod n

### 3.4.2 Quá trình giải mã

Đầu vào: Bản mã C, khóa riêng tư (n, d)

Đầu ra: Thông điệp gốc M

Thuật toán:

M = C^d mod n

### 3.4.3 Ví dụ mã hóa/giải mã

Sử dụng khóa đã tạo ở trên:

* Khóa công khai: (323, 5)
* Khóa riêng tư: (323, 173)

Mã hóa thông điệp M = 123: C = 123^5 mod 323 = 28,372,757,913 mod 323 = 225

Giải mã bản mã C = 225: M = 225^173 mod 323

Để tính toán hiệu quả, sử dụng thuật toán lũy thừa nhanh (fast exponentiation): M = 123 (khôi phục đúng thông điệp gốc)

## 3.5 Tối ưu hóa và cải tiến

### 3.5.1 Thuật toán lũy thừa nhanh

Trong cả quá trình mã hóa (C=Me(modn)) và giải mã (M=Cd(modn)) của RSA, chúng ta cần tính toán lũy thừa với số mũ rất lớn (e và d) và lấy modulo một số nguyên lớn (n). Việc tính toán trực tiếp Me hoặc Cd rồi mới lấy modulo sẽ dẫn đến những con số khổng lồ, vượt quá khả năng lưu trữ và tính toán của máy tính.

Để giải quyết vấn đề này, thuật toán lũy thừa nhanh (còn gọi là lũy thừa theo mô-đun, exponentiation by squaring) được sử dụng. Thuật toán này giúp tính ab(modm) một cách hiệu quả bằng cách chia nhỏ số mũ b thành các lũy thừa của 2.

**Nguyên lý hoạt động:**

Thuật toán lũy thừa nhanh dựa trên các tính chất sau:

* (a×b)(modm)=((a(modm))×(b(modm)))(modm)
* a2k=(ak)2
* a2k+1=a×(ak)2

### 3.5.2 Đệm (Padding Schemes)

Các thuật toán RSA cơ bản mã hóa dữ liệu theo khối (block cipher), với kích thước khối giới hạn bởi kích thước của n. Điều này có thể dẫn đến một số vấn đề bảo mật nếu không được xử lý đúng cách, chẳng hạn như:

* Tấn công chữ ký giả mạo (Forged Signatures): Kẻ tấn công có thể tạo ra các bản mã có ý nghĩa nếu không có đệm.
* Tấn công lựa chọn bản mã (Chosen-ciphertext Attacks): Kẻ tấn công có thể lợi dụng các tính chất toán học của RSA để thu thập thông tin về bản rõ.
* Mã hóa lặp lại: Nếu một thông điệp nhỏ được mã hóa nhiều lần, bản mã có thể dễ dàng bị tấn công.

Để khắc phục những điểm yếu này, các **sơ đồ đệm** (padding schemes) được sử dụng trước khi mã hóa. Đệm thêm dữ liệu ngẫu nhiên hoặc có cấu trúc vào thông điệp gốc, đảm bảo rằng mỗi lần mã hóa, bản mã sẽ khác nhau ngay cả khi thông điệp giống nhau, và làm cho các cuộc tấn công nhất định trở nên khó khăn hơn.

Các sơ đồ đệm phổ biến:

* PKCS#1 v1.5 (Public-Key Cryptography Standard #1, version 1.5): Đây là một trong những sơ đồ đệm đầu tiên và vẫn được sử dụng rộng rãi. PKCS#1 v1.5 thêm một chuỗi byte cấu trúc bao gồm byte 0x00, loại đệm (0x01 cho chữ ký, 0x02 cho mã hóa), một chuỗi byte ngẫu nhiên khác 0, và byte 0x00 kết thúc trước thông điệp. Mặc dù đã có những lỗ hổng được phát hiện trong các tình huống cụ thể (như tấn công Bleichenbacher), nó vẫn được coi là an toàn nếu được triển khai đúng cách.
* OAEP (Optimal Asymmetric Encryption Padding): OAEP là một sơ đồ đệm hiện đại hơn, được thiết kế để chống lại các cuộc tấn công lựa chọn bản mã. OAEP sử dụng một hàm băm và một trình tạo số ngẫu nhiên mặt nạ (MGF) để tạo ra đệm ngẫu nhiên hơn và phức tạp hơn. Nó cung cấp một mức độ bảo mật cao hơn đáng kể so với PKCS#1 v1.5 và được khuyến nghị cho các ứng dụng mới.

Việc sử dụng đệm là cực kỳ quan trọng để đảm bảo tính an toàn của RSA trong các ứng dụng thực tế.

## 3.6 Tính bảo mật và ứng dụng

### 3.6.1 Tính bảo mật của RSA

Tính bảo mật của RSA dựa trên hai bài toán toán học khó:

1. Bài toán phân tích thừa số nguyên tố (Factoring Problem): Độ khó của việc phân tích một số nguyên lớn n (được tạo ra từ tích của hai số nguyên tố lớn p và q) thành các thừa số nguyên tố p và q. Cho đến nay, không có thuật toán hiệu quả nào được biết đến có thể giải quyết bài toán này trong thời gian đa thức đối với các số có độ dài đủ lớn.
2. Bài toán nghịch đảo rời rạc (RSA Problem): Độ khó của việc tính M từ C, e, và n mà không biết d. Mặc dù bài toán này không hoàn toàn tương đương với bài toán phân tích thừa số, nhưng việc giải quyết nó thường liên quan đến khả năng phân tích n.

Các yếu tố ảnh hưởng đến bảo mật:

* Độ dài khóa (Key Length): Đây là yếu tố quan trọng nhất. Khóa càng dài (ví dụ: 2048 bit, 3072 bit, 4096 bit), bài toán phân tích thừa số càng khó và RSA càng an toàn. Các khóa ngắn hơn (ví dụ: 512 bit, 1024 bit) đã bị phá vỡ hoặc có nguy cơ bị phá vỡ trong tương lai gần.
* Chất lượng của số nguyên tố: Các số nguyên tố p và q phải được chọn ngẫu nhiên và phải đủ lớn để tránh các cuộc tấn công dựa trên các số nguyên tố nhỏ. Ngoài ra, p−1 và q−1 không nên chỉ có các thừa số nguyên tố nhỏ.
* Triển khai (Implementation): Ngay cả với các tham số khóa mạnh, việc triển khai RSA không đúng cách (ví dụ: lỗi trong tạo số ngẫu nhiên, không sử dụng đệm thích hợp) có thể dẫn đến các lỗ hổng bảo mật nghiêm trọng.

Thách thức trong tương lai:

Sự phát triển của máy tính lượng tử đặt ra một mối đe dọa tiềm tàng đối với RSA. Thuật toán Shor trên máy tính lượng tử có khả năng phân tích các số nguyên lớn thành thừa số nguyên tố một cách hiệu quả, làm cho RSA không còn an toàn. Do đó, các nhà nghiên cứu đang tích cực phát triển các thuật toán mật mã học hậu lượng tử (post-quantum cryptography) để thay thế RSA trong tương lai.

### 3.6.2 Ứng dụng của RSA

RSA được ứng dụng rộng rãi trong nhiều lĩnh vực của bảo mật thông tin do khả năng cung cấp cả mã hóa và chữ ký số.

1. Mã hóa dữ liệu:

* Mã hóa khóa phiên (Session Key Encryption): RSA thường được sử dụng để mã hóa các khóa đối xứng nhỏ (khóa phiên) mà sau đó được dùng để mã hóa dữ liệu thực tế. Điều này là do mã hóa đối xứng nhanh hơn nhiều so với RSA đối với dữ liệu lớn. Đây là cách RSA được sử dụng trong giao thức SSL/TLS để thiết lập kết nối an toàn trên Internet (ví dụ: khi bạn truy cập một trang web HTTPS).
* Mã hóa dữ liệu nhạy cảm nhỏ: Đối với các thông tin nhạy cảm có kích thước nhỏ như mật khẩu hoặc thông tin thẻ tín dụng (trong một số trường hợp nhất định), RSA có thể được sử dụng trực tiếp.

2. Chữ ký số (Digital Signatures):

Đây là một trong những ứng dụng quan trọng nhất của RSA, cung cấp tính xác thực, toàn vẹn và chống chối bỏ.

* Xác thực nguồn gốc: Đảm bảo rằng thông điệp thực sự đến từ người gửi tuyên bố.
* Toàn vẹn dữ liệu: Đảm bảo rằng thông điệp không bị thay đổi sau khi được ký.
* Chống chối bỏ: Ngăn người gửi phủ nhận việc đã gửi thông điệp.

Cách hoạt động của chữ ký số RSA:

1. Người gửi tạo một hàm băm (hash) của thông điệp.
2. Người gửi mã hóa hàm băm này bằng khóa riêng tư của mình. Đây chính là chữ ký số.
3. Người gửi đính kèm chữ ký số vào thông điệp và gửi đi.
4. Người nhận tính toán hàm băm của thông điệp nhận được.
5. Người nhận giải mã chữ ký số bằng khóa công khai của người gửi.
6. Nếu hàm băm giải mã được khớp với hàm băm người nhận tự tính toán, chữ ký được xác minh là hợp lệ.

Ví dụ ứng dụng:

* Chứng chỉ số (Digital Certificates): Được sử dụng trong cơ sở hạ tầng khóa công khai (PKI) để xác minh danh tính của trang web, máy chủ và người dùng. RSA là thuật toán chính được sử dụng để ký và xác minh các chứng chỉ X.509.
* Ký mã (Code Signing): Đảm bảo tính toàn vẹn của phần mềm và xác định nhà phát triển.
* Giao dịch tài chính: Bảo mật các giao dịch trực tuyến và xác thực người dùng.

## 3.7 Ứng dụng trong trò chơi

Trong game "Giải mã kho báu", RSA được tích hợp vào các cấp độ nâng cao, giới thiệu cho người chơi về khái niệm mã hóa bất đối xứng và tầm quan trọng của nó trong bảo mật hiện đại.

### 3.7.1 Thiết kế cấp độ

* Cấp độ 7: Giới thiệu RSA cơ bản
  + Người chơi được cung cấp các tham số khóa công khai (n, e) và một bản mã C.
  + Nhiệm vụ là tìm khóa riêng tư d bằng cách phân tích n thành p và q (được đơn giản hóa bằng cách cho n là một số nhỏ hơn, ví dụ, có thể phân tích bằng tay hoặc máy tính đơn giản).
  + Sau khi tìm được d, người chơi sẽ giải mã C để nhận được thông điệp.
  + Trò chơi có thể cung cấp công cụ tính toán modulo và lũy thừa nhanh.
* Cấp độ 8: Thách thức với chữ ký số
  + Người chơi được giao một thông điệp đã ký và khóa công khai của người ký.
  + Nhiệm vụ là xác minh chữ ký, hoặc phát hiện xem thông điệp có bị giả mạo hay không.
  + Cấp độ này giúp người chơi hiểu về vai trò của chữ ký số trong việc đảm bảo tính toàn vẹn và xác thực.
* Cấp độ 9: Mô phỏng tấn công RSA
  + Cung cấp một kịch bản nơi kẻ tấn công cố gắng phá vỡ RSA với các tham số yếu (ví dụ: p và q quá gần nhau, e nhỏ không an toàn).
  + Người chơi phải nhận diện các điểm yếu này và đề xuất cách khắc phục hoặc thực hiện một cuộc tấn công đơn giản để khai thác lỗ hổng.

### 3.7.2 Giao diện người chơi

* Bộ tạo khóa RSA mô phỏng: Cho phép người chơi tự tạo cặp khóa RSA với các tham số p,q do họ chọn và xem các giá trị n,ϕ(n),e,d được tính toán. Điều này giúp trực quan hóa quá trình tạo khóa.
* Công cụ mã hóa/giải mã tương tác: Cung cấp giao diện để người chơi nhập thông điệp, chọn khóa và xem kết quả mã hóa/giải mã.
* Trình phân tích số nguyên tố: Một công cụ đơn giản để giúp người chơi phân tích các số nhỏ thành thừa số nguyên tố, hỗ trợ giải các cấp độ đầu tiên của RSA.
* Mô phỏng chữ ký số: Giao diện cho phép người chơi ký thông điệp và xác minh chữ ký, hiển thị các bước của quá trình.

### 3.7.3 Hệ thống đánh giá

* Điểm cơ bản: Dành cho việc giải mã chính xác thông điệp hoặc xác minh chữ ký thành công.
* Điểm thưởng: Cho việc giải quyết các cấp độ nhanh chóng hoặc sử dụng các phương pháp tối ưu.
* Giảm điểm: Khi thử quá nhiều lần hoặc đưa ra các giải pháp không hiệu quả.

# CHƯƠNG 4: THUẬT TOÁN AES

## 4.1 Giới thiệu AES

Thuật toán AES (Advanced Encryption Standard - Tiêu chuẩn Mã hóa Nâng cao) là một thuật toán mã hóa đối xứng được công nhận rộng rãi và sử dụng phổ biến trong các hệ thống bảo mật thông tin hiện đại. Được phát triển bởi hai nhà mật mã học người Bỉ, Joan Daemen và Vincent Rijmen, AES được chọn làm tiêu chuẩn mã hóa bởi Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST) vào năm 2001 sau một quá trình đánh giá cạnh tranh.

AES được thiết kế để thay thế cho thuật toán DES (Data Encryption Standard) đã lỗi thời, cung cấp mức độ bảo mật cao hơn, hiệu suất tốt hơn và khả năng triển khai trên nhiều nền tảng phần cứng và phần mềm. Thuật toán này được sử dụng trong nhiều ứng dụng, từ mã hóa dữ liệu trên đĩa cứng, giao thức SSL/TLS cho các kết nối an toàn trên Internet, đến các hệ thống VPN và mã hóa cơ sở dữ liệu.

AES hỗ trợ ba độ dài khóa: 128 bit, 192 bit và 256 bit, tương ứng với các cấp độ bảo mật khác nhau. Với tính linh hoạt và độ an toàn cao, AES đã trở thành tiêu chuẩn vàng trong mã hóa đối xứng.

## 4.2 Nguyên lý hoạt động

AES là một thuật toán mã hóa khối (block cipher), xử lý dữ liệu theo các khối có kích thước cố định là 128 bit (16 byte). Dữ liệu đầu vào được chia thành các khối, và mỗi khối được mã hóa hoặc giải mã bằng cách sử dụng một khóa bí mật chung cho cả hai quá trình. AES hoạt động thông qua một chuỗi các phép biến đổi toán học được lặp lại qua nhiều vòng (rounds), với số lượng vòng phụ thuộc vào độ dài khóa:

* Khóa 128 bit: 10 vòng
* Khóa 192 bit: 12 vòng
* Khóa 256 bit: 14 vòng

Quá trình mã hóa AES bao gồm bốn bước biến đổi chính trong mỗi vòng:

1. **SubBytes**: Thay thế từng byte trong khối dữ liệu bằng một byte khác theo một bảng thay thế (S-box) được định nghĩa trước, dựa trên phép thay thế phi tuyến để tăng tính bảo mật.
2. **ShiftRows**: Dịch chuyển các hàng của khối dữ liệu (được biểu diễn dưới dạng ma trận 4x4) theo các khoảng cách khác nhau để trộn dữ liệu.
3. **MixColumns**: Thực hiện phép nhân ma trận trong trường Galois GF(2^8) để trộn các cột của khối dữ liệu, tăng tính khuếch tán.
4. **AddRoundKey**: Kết hợp khối dữ liệu với một khóa con (subkey) được tạo ra từ khóa chính thông qua quá trình mở rộng khóa.

Quá trình giải mã thực hiện các bước ngược lại với thứ tự ngược, sử dụng các phép biến đổi nghịch đảo (InvSubBytes, InvShiftRows, InvMixColumns).

## 4.3 Thuật toán mã hóa và giải mã

### 4.3.1 Thuật toán mã hóa

**Đầu vào**: Khối dữ liệu gốc P (128 bit), khóa K (128, 192 hoặc 256 bit)  
**Đầu ra**: Khối dữ liệu mã hóa C (128 bit)

**Bước 1: Mở rộng khóa**

* Tạo ra một tập hợp các khóa con (round keys) từ khóa chính K bằng thuật toán mở rộng khóa (Key Expansion).
* Số lượng khóa con bằng số vòng cộng thêm một khóa ban đầu (Nr + 1, với Nr là số vòng).

**Bước 2: Vòng khởi tạo**

* Thực hiện phép **AddRoundKey** đầu tiên: kết hợp khối dữ liệu P với khóa con đầu tiên.

**Bước 3: Các vòng chính (từ 1 đến Nr-1)**  
FOR i = 1 to Nr-1 DO  
    1. **SubBytes**: Thay thế từng byte trong khối dữ liệu theo bảng S-box.  
    2. **ShiftRows**: Dịch chuyển hàng thứ i của ma trận trạng thái sang trái i-1 vị trí.  
    3. **MixColumns**: Thực hiện phép nhân ma trận trong trường Galois để trộn các cột.  
    4. **AddRoundKey**: Kết hợp khối dữ liệu với khóa con thứ i.  
END FOR

**Bước 4: Vòng cuối (vòng Nr)**

* Thực hiện **SubBytes**, **ShiftRows**, và **AddRoundKey** (bỏ qua **MixColumns**).

**Kết quả**: Khối dữ liệu mã hóa C.

### 4.3.2 Thuật toán giải mã

**Đầu vào**: Khối dữ liệu mã hóa C (128 bit), khóa K (128, 192 hoặc 256 bit)  
**Đầu ra**: Khối dữ liệu gốc P (128 bit)

**Bước 1: Mở rộng khóa**

* Sử dụng cùng tập hợp khóa con như trong quá trình mã hóa.

**Bước 2: Vòng khởi tạo**

* Thực hiện phép **AddRoundKey** với khóa con cuối cùng.

**Bước 3: Các vòng chính (từ 1 đến Nr-1)**  
FOR i = 1 to Nr-1 DO  
    1. **InvShiftRows**: Dịch chuyển hàng thứ i của ma trận trạng thái sang phải i-1 vị trí.  
    2. **InvSubBytes**: Thay thế từng byte trong khối dữ liệu theo bảng S-box nghịch đảo.  
    3. **AddRoundKey**: Kết hợp khối dữ liệu với khóa con thứ (Nr-i).  
    4. **InvMixColumns**: Thực hiện phép nhân ma trận nghịch đảo trong trường Galois.  
END FOR

**Bước 4: Vòng cuối (vòng Nr)**

* Thực hiện **InvShiftRows**, **InvSubBytes**, và **AddRoundKey** (bỏ qua **InvMixColumns**).

**Kết quả**: Khối dữ liệu gốc P.

## 4.4 Ví dụ minh họa

**Ví dụ mã hóa đơn giản với AES-128**

**Đầu vào**:

* Khối dữ liệu gốc (plaintext): 00112233445566778899aabbccddeeff (16 byte, hex).
* Khóa: 000102030405060708090a0b0c0d0e0f (128 bit, hex).

**Bước 1: Mở rộng khóa**

* Khóa được mở rộng để tạo ra 11 khóa con (cho 10 vòng + 1 vòng khởi tạo).
* (Quá trình mở rộng khóa phức tạp, sử dụng phép xoay từ, thay thế byte, và hằng số vòng; chi tiết được bỏ qua để đơn giản hóa).

**Bước 2: Vòng khởi tạo**

* Kết hợp khối dữ liệu với khóa con đầu tiên (AddRoundKey).

**Bước 3: Các vòng chính (9 vòng)**

* Thực hiện tuần tự các bước SubBytes, ShiftRows, MixColumns, và AddRoundKey.
* Ví dụ: Sau vòng đầu tiên, trạng thái ma trận có thể thay đổi như sau (giá trị giả định để minh họa):
  + **SubBytes**: Thay thế từng byte theo bảng S-box.
  + **ShiftRows**: Dịch chuyển các hàng (hàng 1 giữ nguyên, hàng 2 dịch trái 1, hàng 3 dịch trái 2, hàng 4 dịch trái 3).
  + **MixColumns**: Nhân ma trận trạng thái với ma trận cố định trong trường Galois.
  + **AddRoundKey**: Kết hợp với khóa con tiếp theo.

**Bước 4: Vòng cuối**

* Thực hiện SubBytes, ShiftRows, và AddRoundKey.

**Kết quả**: Khối mã hóa (ciphertext): 69c4e0d86a7b0430d8cdb78070b4c55a (hex).

**Giải mã**:

* Sử dụng cùng khóa và các bước nghịch đảo để khôi phục khối dữ liệu gốc 00112233445566778899aabbccddeeff.

(Lưu ý: Các giá trị trên là minh họa; kết quả thực tế phụ thuộc vào triển khai cụ thể và bảng S-box chuẩn của AES).

## 4.5 Ưu điểm và nhược điểm

### 4.5.1 Ưu điểm

* **Bảo mật cao**: Với độ dài khóa 128, 192 hoặc 256 bit, AES được coi là an toàn trước các cuộc tấn công brute force với công nghệ hiện tại.
* **Hiệu suất tốt**: AES được tối ưu hóa cho cả phần cứng và phần mềm, phù hợp với các thiết bị từ máy tính đến thiết bị nhúng.
* **Tính linh hoạt**: Hỗ trợ nhiều độ dài khóa và dễ dàng tích hợp vào các giao thức bảo mật như SSL/TLS, IPsec, và VPN.
* **Kháng tấn công phân tích**: Thiết kế với các bước SubBytes, MixColumns đảm bảo tính khuếch tán và nhầm lẫn, làm khó các cuộc tấn công phân tích tần suất hoặc tuyến tính.
* **Tiêu chuẩn hóa**: Được NIST công nhận và sử dụng rộng rãi trên toàn cầu.

### 4.5.2 Nhược điểm

* **Phụ thuộc vào khóa**: Nếu khóa bị lộ hoặc quản lý không an toàn, toàn bộ hệ thống mã hóa sẽ bị xâm phạm.
* **Độ phức tạp triển khai**: Mặc dù thuật toán được tối ưu hóa, việc triển khai sai (ví dụ: lỗi trong mở rộng khóa) có thể dẫn đến lỗ hổng.
* **Không phù hợp với dữ liệu lớn trực tiếp**: Mã hóa khối yêu cầu đệm và chế độ vận hành (như CBC, CTR) để xử lý dữ liệu dài, làm tăng độ phức tạp.
* **Mối đe dọa từ máy tính lượng tử**: Trong tương lai, thuật toán Grover trên máy tính lượng tử có thể giảm độ an toàn của AES, yêu cầu tăng độ dài khóa.

## 4.6 Ứng dụng trong trò chơi

Trong game "Giải mã kho báu", AES được tích hợp vào các cấp độ cao nhất, giúp người chơi làm quen với khái niệm mã hóa đối xứng hiện đại và các ứng dụng thực tế của nó trong bảo mật thông tin.

### 4.6.1 Thiết kế cấp độ

**Cấp độ 10: Giới thiệu AES cơ bản**

* + Người chơi được cung cấp một khối dữ liệu mã hóa và khóa 128 bit.
  + Nhiệm vụ: Giải mã khối dữ liệu bằng cách mô phỏng các bước SubBytes, ShiftRows, MixColumns, và AddRoundKey.
  + Trò chơi cung cấp bảng S-box và ma trận MixColumns đơn giản hóa để người chơi thực hiện các phép toán thủ công.

**Cấp độ 11: Thử thách chế độ vận hành**

* + Giới thiệu các chế độ như ECB (Electronic Codebook) và CBC (Cipher Block Chaining).
  + Người chơi phải nhận diện sự khác biệt giữa các chế độ, ví dụ: ECB không an toàn vì lặp lại các mẫu trong dữ liệu gốc.
  + Nhiệm vụ: Giải mã một chuỗi khối bằng chế độ CBC với vector khởi tạo (IV) được cung cấp.

**Cấp độ 12: Mô phỏng tấn công**

* + Người chơi được giao một kịch bản với triển khai AES không an toàn (ví dụ: sử dụng chế độ ECB hoặc khóa yếu).
  + Nhiệm vụ: Phân tích lỗ hổng và đề xuất cách khắc phục (như chuyển sang chế độ CBC với IV ngẫu nhiên).

### 4.6.2 Giao diện người chơi

* **Công cụ mô phỏng AES**: Cho phép người chơi nhập khối dữ liệu, khóa, và xem từng bước của quá trình mã hóa/giải mã (SubBytes, ShiftRows, v.v.).
* **Trình trực quan hóa ma trận trạng thái**: Hiển thị dữ liệu dưới dạng ma trận 4x4, giúp người chơi dễ dàng theo dõi các phép biến đổi.
* **Hỗ trợ tính toán trường Galois**: Cung cấp công cụ để thực hiện phép nhân trong trường GF(2^8) cho bước MixColumns.
* **Mô phỏng chế độ vận hành**: Cho phép người chơi chọn chế độ (ECB, CBC) và xem tác động đến dữ liệu mã hóa.

### 4.6.3 Hệ thống đánh giá

* **Điểm cơ bản**: Dành cho việc giải mã đúng khối dữ liệu hoặc nhận diện chế độ vận hành.
* **Điểm thưởng**: Cho việc hoàn thành cấp độ nhanh chóng hoặc giải thích chính xác các lỗ hổng trong triển khai AES.
* **Giảm điểm**: Khi sử dụng quá nhiều lần thử sai hoặc không nhận ra các điểm yếu bảo mật.

# CHƯƠNG 5: GIỚI THIỆU ĐỀ TÀI - GAME “GIẢI MÃ KHO BÁU”

## 5.1 Mục tiêu của đề tài

Đề tài “Game Giải mã kho báu” được xây dựng với mục tiêu mang lại một phương pháp học tập sáng tạo, kết hợp giữa lý thuyết và thực hành, nhằm giúp sinh viên nắm bắt các khái niệm cơ bản về an toàn bảo mật thông tin thông qua một trải nghiệm trò chơi tương tác. Các mục tiêu cụ thể bao gồm:

* **Giới thiệu các thuật toán mật mã học**: Cung cấp kiến thức về các thuật toán mã hóa quan trọng như Caesar Cipher, Vigenère Cipher, RSA và AES, giúp sinh viên hiểu rõ cách chúng hoạt động và vai trò của chúng trong bảo mật thông tin.
* **Phát triển kỹ năng phân tích**: Khuyến khích người chơi áp dụng tư duy logic, phân tích và giải quyết vấn đề thông qua các thử thách giải mã.
* **Tăng cường ý thức bảo mật**: Nâng cao nhận thức về tầm quan trọng của an toàn thông tin trong đời sống và công việc, đặc biệt trong môi trường số hóa.
* **Tạo hứng thú học tập**: Biến các khái niệm trừu tượng của mật mã học thành các thử thách thú vị, khơi dậy sự tò mò và đam mê với lĩnh vực bảo mật.

## 5.2 Ý nghĩa của đề tài

Trong bối cảnh các mối đe dọa an ninh mạng ngày càng gia tăng, việc hiểu biết và áp dụng các kỹ thuật bảo mật thông tin trở nên cấp thiết. Đề tài “Game Giải mã kho báu” mang ý nghĩa quan trọng cả về mặt học thuật và thực tiễn:

* **Ý nghĩa học thuật**: Trò chơi giúp sinh viên tiếp cận các thuật toán mật mã học một cách trực quan, từ các phương pháp cổ điển (Caesar, Vigenère) đến các thuật toán hiện đại (RSA, AES). Điều này tạo nền tảng vững chắc cho việc học các môn chuyên sâu hơn trong lĩnh vực an toàn bảo mật thông tin.
* **Ý nghĩa thực tiễn**: Các thử thách trong trò chơi mô phỏng các tình huống thực tế, như giải mã thông điệp bí mật hay bảo vệ dữ liệu, giúp người chơi hiểu cách các thuật toán được ứng dụng trong các hệ thống bảo mật như SSL/TLS, VPN, hoặc chữ ký số.
* **Khuyến khích sáng tạo**: Thông qua việc giải các câu đố mật mã, người chơi được khuyến khích sáng tạo trong việc tìm kiếm giải pháp và nhận diện các lỗ hổng bảo mật, từ đó phát triển tư duy phản biện.
* **Tính giáo dục và giải trí**: Kết hợp giữa học tập và giải trí, trò chơi tạo động lực cho sinh viên học hỏi một cách tự nhiên, giảm bớt sự khô khan của các khái niệm lý thuyết.

## 5.3 Phạm vi của đề tài

Đề tài tập trung vào việc thiết kế và triển khai một trò chơi giáo dục với các nội dung chính sau:

* **Các thuật toán mật mã học**: Bao gồm Caesar Cipher (mã hóa thay thế đơn âm), Vigenère Cipher (mã hóa thay thế đa âm), RSA (mã hóa bất đối xứng), và AES (mã hóa đối xứng). Mỗi thuật toán được tích hợp vào các cấp độ khác nhau của trò chơi.
* **Các cấp độ thử thách**: Trò chơi được chia thành các cấp độ từ dễ đến khó, bắt đầu từ các thuật toán đơn giản (Caesar) đến các thuật toán phức tạp hơn (AES), phù hợp với trình độ của sinh viên mới tiếp cận môn học.
* **Ứng dụng thực tế**: Các thử thách được thiết kế để mô phỏng các tình huống thực tế, như giải mã thông điệp bí mật, xác minh chữ ký số, hoặc nhận diện các lỗ hổng bảo mật.
* **Công cụ hỗ trợ**: Cung cấp các công cụ trực quan như trình mô phỏng mã hóa/giải mã, phân tích tần suất, hoặc tính toán số học để hỗ trợ người chơi trong việc giải quyết thử thách.

Phạm vi của trò chơi giới hạn trong việc giới thiệu các khái niệm cơ bản và không đi sâu vào các kỹ thuật tấn công mật mã phức tạp hoặc các ứng dụng chuyên sâu trong môi trường thực tế.

## 5.4 Cấu trúc của trò chơi

Game “Giải mã kho báu” được thiết kế dưới dạng một hành trình thám hiểm, trong đó người chơi đóng vai trò là một nhà thám hiểm tìm kiếm kho báu ẩn giấu. Để đạt được mục tiêu, người chơi phải vượt qua các cấp độ thử thách, mỗi cấp độ yêu cầu giải mã một thông điệp hoặc thực hiện một nhiệm vụ liên quan đến mật mã học. Cấu trúc trò chơi bao gồm:

### 5.4.1 Cốt truyện và bối cảnh

* **Bối cảnh**: Người chơi được đưa vào một thế giới giả tưởng, nơi một kho báu huyền thoại được bảo vệ bởi các thông điệp mã hóa. Mỗi thông điệp chứa manh mối dẫn đến vị trí kho báu.
* **Cốt truyện**: Người chơi bắt đầu hành trình với tư cách là một “học viên mật mã”, dần dần nâng cấp kỹ năng qua các cấp độ để trở thành “bậc thầy mật mã”. Mỗi cấp độ là một thử thách liên quan đến một thuật toán cụ thể, với độ khó tăng dần.

### 5.4.2 Các cấp độ trong trò chơi

* **Cấp độ 1-3 (Caesar Cipher)**: Làm quen với khái niệm mã hóa thay thế đơn âm, từ việc giải mã với khóa cố định (k=3) đến thử nghiệm các khóa khác nhau và tấn công brute force.
* **Cấp độ 4-6 (Vigenère Cipher)**: Tìm hiểu mã hóa thay thế đa âm, từ việc sử dụng từ khóa đơn giản đến đoán từ khóa và phân tích tần suất ký tự.
* **Cấp độ 7-9 (RSA)**: Làm quen với mã hóa bất đối xứng, từ việc giải mã thông điệp với khóa công khai đến xác minh chữ ký số và nhận diện các điểm yếu của RSA.
* **Cấp độ 10-12 (AES)**: Khám phá mã hóa đối xứng hiện đại, từ việc thực hiện các bước mã hóa/giải mã cơ bản đến hiểu các chế độ vận hành (ECB, CBC) và nhận diện lỗ hổng triển khai.

### 5.4.3 Giao diện và công cụ hỗ trợ

* **Giao diện người chơi**: Giao diện trực quan hiển thị văn bản mã hóa, ô nhập khóa, và kết quả giải mã. Các công cụ như bảng S-box (cho AES), trình phân tích tần suất (cho Vigenère), hoặc trình tính toán modulo (cho RSA) được tích hợp để hỗ trợ người chơi.
* **Hệ thống gợi ý**: Cung cấp các gợi ý phân cấp (cơ bản, trung bình, nâng cao) để hướng dẫn người chơi qua các bước giải mã mà không làm mất đi tính thách thức.
* **Hệ thống đánh giá**: Điểm số dựa trên độ chính xác, thời gian hoàn thành, và số lần thử sai. Điểm thưởng được trao cho các giải pháp tối ưu hoặc sáng tạo.

## 5.5 Tích hợp các thuật toán mật mã học

Trò chơi “Giải mã kho báu” được xây dựng dựa trên việc tích hợp bốn thuật toán mật mã học chính, mỗi thuật toán đại diện cho một giai đoạn phát triển của mật mã học:

* **Caesar Cipher**: Giới thiệu khái niệm mã hóa cổ điển, đơn giản và dễ hiểu, giúp người chơi làm quen với ý tưởng dịch chuyển ký tự.
* **Vigenère Cipher**: Tăng độ phức tạp với mã hóa đa âm, yêu cầu người chơi hiểu về từ khóa và phân tích tần suất để phá mã.
* **RSA**: Giới thiệu mã hóa bất đối xứng, nhấn mạnh vai trò của khóa công khai và khóa riêng tư trong các ứng dụng như chữ ký số và bảo mật giao tiếp.
* **AES**: Trình bày mã hóa đối xứng hiện đại, với các bước biến đổi phức tạp và ứng dụng trong các hệ thống bảo mật thực tế.

Mỗi thuật toán được thiết kế thành một chuỗi các cấp độ, từ cơ bản (hiểu thuật toán) đến nâng cao (phân tích và tấn công). Điều này giúp người chơi không chỉ học cách mã hóa/giải mã mà còn hiểu được các điểm yếu và cách cải thiện bảo mật.

## 5.6 Kết quả kỳ vọng

Đề tài “Game Giải mã kho báu” kỳ vọng đạt được các kết quả sau:

* **Kiến thức**: Sinh viên hiểu rõ các nguyên lý cơ bản của bốn thuật toán mật mã học và vai trò của chúng trong an toàn bảo mật thông tin.
* **Kỹ năng**: Phát triển khả năng phân tích, giải quyết vấn đề và áp dụng các thuật toán vào các tình huống thực tế.
* **Thái độ**: Tăng cường ý thức về tầm quan trọng của bảo mật thông tin và khuyến khích sự tò mò, khám phá trong lĩnh vực này.
* **Sản phẩm**: Một trò chơi giáo dục hoàn chỉnh, có thể được sử dụng như một công cụ giảng dạy trong môn học “Nhập môn An toàn Bảo mật Thông tin” hoặc các khóa học liên quan.

## 5.7 Kết luận

“Game Giải mã kho báu” không chỉ là một bài tập lớn mà còn là một cách tiếp cận sáng tạo để kết nối lý thuyết mật mã học với thực tiễn. Thông qua việc tích hợp các thuật toán từ cổ điển đến hiện đại, trò chơi mang đến một hành trình học tập thú vị, giúp sinh viên nắm bắt kiến thức một cách tự nhiên và hiệu quả. Đề tài này hy vọng sẽ góp phần nâng cao chất lượng giảng dạy và học tập trong lĩnh vực an toàn bảo mật thông tin, đồng thời khơi dậy niềm đam mê cho thế hệ tương lai của các chuyên gia bảo mật.

# CHƯƠNG 6: PHÂN TÍCH ĐỀ TÀI

## 6.1 Tổng quan về đề tài

Đề tài “Game Giải mã kho báu” là một dự án giáo dục kết hợp giữa lý thuyết mật mã học và thực hành phát triển phần mềm, nhằm cung cấp một cách tiếp cận tương tác để học các khái niệm cơ bản về an toàn bảo mật thông tin. Trò chơi được thiết kế như một hành trình thám hiểm, trong đó người chơi giải mã các thông điệp sử dụng các thuật toán mật mã học (Caesar Cipher, Vigenère Cipher, RSA, AES) để tìm ra kho báu. Đề tài không chỉ giúp sinh viên hiểu rõ các thuật toán mà còn rèn luyện kỹ năng lập trình, phân tích, và tư duy bảo mật.

Phân tích đề tài tập trung vào các khía cạnh sau:

* Mục tiêu và yêu cầu chức năng của trò chơi.
* Thiết kế hệ thống và cách tích hợp các thuật toán mật mã.
* Phân tích mã nguồn và các thành phần kỹ thuật.
* Đánh giá ưu điểm, hạn chế và đề xuất cải tiến.

## 6.2 Mục tiêu và yêu cầu chức năng

### 6.2.1 Mục tiêu

* **Giáo dục**: Giới thiệu các thuật toán mật mã học từ cổ điển (Caesar, Vigenère) đến hiện đại (RSA, AES) thông qua các cấp độ trò chơi.
* **Tương tác**: Tạo trải nghiệm học tập hấp dẫn bằng cách kết hợp cốt truyện, gợi ý, và giao diện thân thiện.
* **Thực hành**: Phát triển kỹ năng lập trình và áp dụng các thuật toán vào các tình huống thực tế.
* **Nâng cao nhận thức**: Khuyến khích người chơi nhận thức về các vấn đề bảo mật thông tin, như quản lý khóa và các lỗ hổng tiềm ẩn.

### 6.2.2 Yêu cầu chức năng

* **Đăng ký và đăng nhập**: Người chơi có thể tạo tài khoản, đăng nhập, và lưu tiến độ trò chơi.
* **Quản lý cấp độ**: Hệ thống hỗ trợ nhiều cấp độ, mỗi cấp độ sử dụng một thuật toán mật mã và có độ khó tăng dần.
* **Giải mã thông điệp**: Người chơi nhập tham số (khóa, từ khóa) để giải mã thông điệp và nhận điểm thưởng khi giải đúng.
* **Hỗ trợ người chơi**: Cung cấp gợi ý, cốt truyện, và công cụ trực quan để hỗ trợ quá trình giải mã.
* **Đánh giá và tiến độ**: Lưu trữ điểm số và cấp độ hiện tại của người chơi, cho phép đặt lại trò chơi nếu cần.

## 6.3 Thiết kế hệ thống

Hệ thống của trò chơi “Giải mã kho báu” được xây dựng dựa trên mô hình client-server, với backend xử lý logic mật mã và quản lý dữ liệu, và frontend cung cấp giao diện tương tác. Các thành phần chính bao gồm:

### 6.3.1 Kiến trúc tổng thể

* **Backend (Flask)**: Xử lý đăng ký/đăng nhập, quản lý cấp độ, giải mã thông điệp, và lưu trữ dữ liệu người chơi. File app.py là trung tâm của backend, sử dụng Flask để tạo API RESTful và SQLAlchemy để quản lý cơ sở dữ liệu SQLite.
* **Frontend (HTML, JavaScript, Bootstrap)**: File index.html cung cấp giao diện người dùng với các màn hình đăng nhập, đăng ký, chơi game, và hiển thị kết quả. Bootstrap đảm bảo giao diện thân thiện và responsive.
* **Thư viện mật mã (cipher\_utils.py)**: File cipher\_utils.py chứa các hàm mã hóa/giải mã cho các thuật toán Caesar, Vigenère, RSA, và AES, được tích hợp vào backend để xử lý logic giải mã.

### 6.3.2 Tích hợp thuật toán mật mã

* **Caesar Cipher**: Được triển khai trong cipher\_utils.py với hàm caesar\_cipher\_encrypt và caesar\_cipher\_decrypt. Thuật toán này đơn giản, sử dụng phép dịch chuyển ký tự với khóa là một số nguyên.
* **Vigenère Cipher**: Được triển khai với hàm vigenere\_cipher\_encrypt và vigenere\_cipher\_decrypt, sử dụng từ khóa để tạo chuỗi dịch chuyển động.
* **RSA**: Sử dụng thư viện Crypto.PublicKey.RSA với đệm OAEP (Optimal Asymmetric Encryption Padding) để mã hóa/giải mã. Khóa công khai và khóa riêng được lưu trữ trong file .pem.
* **AES**: Sử dụng thư viện Crypto.Cipher.AES với chế độ CBC và đệm PKCS7. Khóa AES được lưu trữ trong file .bin và được sử dụng để mã hóa/giải mã các thông điệp.

### 6.3.3 Cơ sở dữ liệu

* Cơ sở dữ liệu SQLite (game.db) lưu trữ thông tin người dùng (tên, mật khẩu băm, điểm số, cấp độ hiện tại) thông qua model User trong app.py.
* Dữ liệu cấp độ được lưu trong biến LEVELS trong app.py, chứa thông tin về thuật toán, văn bản mã hóa, gợi ý, và giải pháp đúng.

## 6.4 Phân tích mã nguồn

### 6.4.1 File cipher\_utils.py

* **Mục đích**: Cung cấp các hàm mã hóa/giải mã cho bốn thuật toán và quản lý khóa.
* **Phân tích**:
  + **Khởi tạo khóa**: Hàm generate\_and\_save\_keys tạo và lưu trữ khóa RSA (2048 bit) và AES (128 bit) vào file nếu chúng chưa tồn tại. Điều này đảm bảo tính nhất quán của khóa giữa các phiên chạy chương trình, nhưng việc lưu trữ trực tiếp vào file hệ thống có thể gây rủi ro bảo mật trong môi trường sản phẩm.
  + **Caesar Cipher**: Hàm caesar\_cipher\_encrypt và caesar\_cipher\_decrypt xử lý văn bản chữ cái (a-z, A-Z) và giữ nguyên các ký tự không phải chữ cái. Thuật toán đơn giản và dễ triển khai, phù hợp với cấp độ đầu tiên.
  + **Vigenère Cipher**: Hàm vigenere\_cipher\_encrypt và vigenere\_cipher\_decrypt mở rộng từ khóa để khớp độ dài văn bản, sử dụng phép modulo 26 để dịch chuyển ký tự. Đây là bước nâng cấp từ Caesar, tăng tính phức tạp.
  + **RSA**: Hàm rsa\_encrypt và rsa\_decrypt sử dụng đệm OAEP để đảm bảo bảo mật, phù hợp với các thông điệp ngắn trong trò chơi. Việc sử dụng thư viện Crypto giúp đơn giản hóa việc triển khai.
  + **AES**: Hàm aes\_encrypt và aes\_decrypt sử dụng chế độ CBC với vector khởi tạo (IV) ngẫu nhiên, kết hợp đệm PKCS7 để xử lý văn bản có độ dài không chia hết cho kích thước khối (16 byte).
* **Ưu điểm**: Mã nguồn được tổ chức rõ ràng, dễ bảo trì, và sử dụng thư viện Crypto để đảm bảo tính chính xác của các thuật toán hiện đại (RSA, AES).
* **Hạn chế**: Lưu trữ khóa trực tiếp vào file hệ thống không an toàn trong môi trường sản phẩm. Thiếu cơ chế kiểm tra lỗi chi tiết cho các tham số đầu vào (ví dụ: kiểm tra từ khóa Vigenère có hợp lệ).

### 6.4.2 File app.py

* **Mục đích**: Xây dựng backend của trò chơi, xử lý API, quản lý người dùng, và logic giải mã.
* **Phân tích**:
  + **Quản lý người dùng**: Các endpoint /register, /login, /logout, và /get\_user\_data sử dụng Flask-SQLAlchemy và Flask-Bcrypt để quản lý tài khoản, băm mật khẩu, và duy trì phiên đăng nhập qua session.
  + **Quản lý cấp độ**: Biến LEVELS định nghĩa bốn cấp độ, mỗi cấp độ chứa thông tin về thuật toán, văn bản mã hóa, gợi ý, và giải pháp đúng. Endpoint /level/<int:level> cung cấp thông tin cấp độ, đảm bảo người chơi chỉ truy cập các cấp độ đã mở khóa.
  + **Xử lý giải mã**: Endpoint /decode/<int:level> kiểm tra đầu vào của người chơi (văn bản mã hóa và tham số), gọi các hàm giải mã từ cipher\_utils.py, và cập nhật điểm số/cấp độ nếu giải mã thành công.
  + **Bảo mật**: Sử dụng regex để kiểm tra tên người dùng, băm mật khẩu bằng bcrypt, và kiểm tra quyền truy cập cấp độ. Tuy nhiên, SECRET\_KEY được tạo ngẫu nhiên bằng os.urandom(24) nếu không được đặt trong biến môi trường, điều này không phù hợp với môi trường sản phẩm.
* **Ưu điểm**: Backend được tổ chức tốt, hỗ trợ CORS cho frontend, và có cơ chế logging để theo dõi lỗi và hành động người dùng. Hệ thống điểm số và cấp độ được tích hợp chặt chẽ với cơ sở dữ liệu.
* **Hạn chế**: Thiếu cơ chế giới hạn số lần thử giải mã để ngăn chặn brute force. CORS hiện cho phép tất cả các nguồn gốc (\*), cần giới hạn trong sản phẩm. Thiếu xử lý ngoại lệ chi tiết trong một số trường hợp.

### 6.4.3 File index.html

* **Mục đích**: Cung cấp giao diện người dùng cho trò chơi, sử dụng Bootstrap 5 và JavaScript để tương tác với backend.
* **Phân tích**:
  + **Giao diện**: Bao gồm các màn hình đăng nhập, đăng ký, chơi game, và chiến thắng. Sử dụng Bootstrap để tạo giao diện responsive, với các phần tử như form, button, và alert.
  + **Tương tác**: Các phần tử HTML sử dụng data-bs-toggle để hiển thị/ẩn gợi ý, và JavaScript (không được cung cấp trong mã) dự kiến sẽ xử lý các yêu cầu AJAX tới backend để lấy thông tin cấp độ và gửi kết quả giải mã.
  + **Hạn chế**: Mã HTML hiện tại thiếu phần JavaScript để xử lý logic giao tiếp với backend (ví dụ: gửi yêu cầu tới /decode/<int:level>). Điều này cần được bổ sung để hoàn thiện chức năng trò chơi.
* **Ưu điểm**: Giao diện đơn giản, dễ sử dụng, và phù hợp với mục đích giáo dục. Sử dụng Bootstrap giúp đảm bảo tính tương thích trên nhiều thiết bị.
* **Hạn chế**: Thiếu mã JavaScript để xử lý logic phía client, làm giảm tính tương tác. Thiếu các biện pháp bảo mật phía client, như kiểm tra đầu vào trước khi gửi yêu cầu.

## 6.5 Đánh giá ưu điểm và hạn chế

### 6.5.1 Ưu điểm

* **Tính giáo dục**: Trò chơi tích hợp bốn thuật toán mật mã học, từ đơn giản đến phức tạp, giúp người chơi hiểu rõ sự phát triển của mật mã học.
* **Tính tương tác**: Giao diện thân thiện, kết hợp cốt truyện và gợi ý, tạo trải nghiệm học tập thú vị.
* **Tính thực tiễn**: Sử dụng các thư viện tiêu chuẩn (Crypto) và framework phổ biến (Flask, Bootstrap) để triển khai các thuật toán và giao diện.
* **Khả năng mở rộng**: Thiết kế mô-đun (cipher\_utils.py tách biệt logic mã hóa) cho phép dễ dàng thêm các thuật toán hoặc cấp độ mới.

### 6.5.2 Hạn chế

* **Bảo mật**: Lưu trữ khóa trong file hệ thống (rsa\_private\_key.pem, aes\_key.bin) không an toàn. Trong môi trường thực tế, cần sử dụng hệ thống quản lý khóa như HashiCorp Vault.
* **Hiệu suất**: SQLite phù hợp cho phát triển, nhưng có thể không đáp ứng được trong môi trường sản phẩm với nhiều người chơi đồng thời.
* **Tính hoàn thiện**: Mã JavaScript phía client chưa được cung cấp, dẫn đến giao diện chưa đầy đủ chức năng.
* **Giới hạn thử thách**: Chỉ có bốn cấp độ, có thể không đủ thách thức với người chơi có kinh nghiệm. Thiếu các cấp độ phân tích tấn công (ví dụ: phân tích tần suất cho Vigenère).

.

# Chương 7: Kết luận

## 7.1 Điểm mạnh

* Dự án "Giải mã Kho báu" đã đạt được một số điểm mạnh nổi bật như sau:
* **Tính giáo dục và tương tác cao**: Trò chơi kết hợp các thuật toán mã hóa (Caesar, Vigenère, RSA, AES) với cốt truyện hấp dẫn, giúp người chơi học hỏi kiến thức về mật mã học một cách thú vị và trực quan.
* **Giao diện thân thiện**: Thiết kế giao diện sử dụng HTML, CSS, và JavaScript với phong cách hiện đại, dễ sử dụng, hỗ trợ responsive cho các thiết bị di động (media query trong style.css).
* **Tính bảo mật**: Hệ thống sử dụng Flask và SQLAlchemy để quản lý người dùng, lưu trữ mật khẩu được mã hóa bằng bcrypt, đảm bảo an toàn dữ liệu người dùng.
* **Cấu trúc mô-đun**: Các hàm mã hóa/giải mã được tách riêng trong cipher\_utils.py, giúp mã nguồn dễ bảo trì và mở rộng.
* **Tính năng đa dạng**: Hỗ trợ đăng ký, đăng nhập, đặt lại trò chơi, và theo dõi tiến độ (điểm số, cấp độ), mang lại trải nghiệm liền mạch cho người chơi.
* **Hỗ trợ nhiều thuật toán mã hóa**: Trò chơi tích hợp bốn thuật toán mã hóa khác nhau, từ đơn giản (Caesar) đến phức tạp (RSA, AES), phù hợp với người chơi ở nhiều cấp độ.

## 7.2 Điểm yếu

* Mặc dù có nhiều ưu điểm, dự án vẫn tồn tại một số hạn chế cần khắc phục:
* **Phụ thuộc vào server backend**: Các chức năng như đăng ký, đăng nhập, và giải mã đều yêu cầu kết nối với server Flask (app.py), dẫn đến việc trò chơi không thể chơi offline.
* **Hạn chế về số lượng cấp độ**: Hiện tại chỉ có 4 cấp độ, có thể khiến người chơi nhanh chóng hoàn thành trò chơi và giảm tính thử thách lâu dài.
* **Xử lý lỗi chưa tối ưu**: Một số thông báo lỗi (ví dụ: "Lỗi kết nối server!") còn chung chung, thiếu chi tiết để người chơi hiểu rõ vấn đề.
* **Hiệu suất xử lý RSA và AES**: Với các thiết bị có cấu hình thấp, việc giải mã RSA và AES có thể gây chậm trễ do tính toán phức tạp.
* **Thiếu tính năng lưu tiến độ tạm thời**: Nếu người chơi thoát giữa chừng, không có cơ chế lưu trạng thái hiện tại của cấp độ (ngoài điểm số và cấp độ đã lưu trong database).
* **Bảo mật khóa chưa tối ưu**: Khóa RSA và AES được lưu trong file (rsa\_private\_key.pem, aes\_key.bin), có nguy cơ bị truy cập trái phép nếu không được bảo vệ tốt trong môi trường sản xuất.

## 7.3 Hướng cải tiến và cải thiện

Để nâng cao chất lượng dự án, các cải tiến sau đây được đề xuất:

* **Hỗ trợ chơi offline**: Tích hợp các thuật toán mã hóa/giải mã trực tiếp trên client-side sử dụng JavaScript (trong game.js), giảm sự phụ thuộc vào server backend.
* **Mở rộng số lượng cấp độ**: Thêm nhiều cấp độ mới với các thuật toán mã hóa khác (ví dụ: DES, Blowfish) hoặc các biến thể của thuật toán hiện tại để tăng tính đa dạng và thử thách.
* **Cải thiện thông báo lỗi**: Cung cấp thông báo lỗi chi tiết hơn, ví dụ: chỉ rõ định dạng khóa RSA hoặc AES không hợp lệ thay vì thông báo chung chung.
* **Tối ưu hóa hiệu suất**: Sử dụng các thư viện mã hóa nhẹ hơn hoặc tối ưu hóa thuật toán RSA và AES để giảm thời gian xử lý trên các thiết bị yếu.
* **Lưu trạng thái tạm thời**: Thêm cơ chế lưu trữ tạm thời (ví dụ: sử dụng localStorage hoặc sessionStorage trong trình duyệt) để lưu trạng thái của cấp độ hiện tại khi người chơi thoát ra.
* **Tăng cường bảo mật khóa**: Thay vì lưu khóa RSA và AES trong file, sử dụng các giải pháp quản lý khóa an toàn như HashiCorp Vault hoặc AWS KMS trong môi trường sản xuất.
* **Thêm hướng dẫn chi tiết**: Cung cấp tài liệu hoặc hướng dẫn trong trò chơi về cách hoạt động của các thuật toán mã hóa để hỗ trợ người chơi mới.

## 7.4 Hướng phát triển

Dự án có tiềm năng phát triển thêm trong tương lai với các hướng sau:

* **Tích hợp chế độ nhiều người chơi**: Cho phép người chơi cạnh tranh hoặc hợp tác giải mã thông điệp qua mạng, sử dụng WebSocket hoặc API thời gian thực.
* **Phát triển ứng dụng di động**: Chuyển đổi trò chơi thành ứng dụng di động (sử dụng React Native hoặc Flutter) để tiếp cận nhiều người chơi hơn, tận dụng các nền tảng như iOS và Android.
* **Tích hợp AI hỗ trợ**: Sử dụng AI (ví dụ: Grok 3 của xAI) để cung cấp gợi ý thông minh hoặc phân tích cách giải mã của người chơi, cải thiện trải nghiệm học tập.
* **Mở rộng nội dung cốt truyện**: Phát triển thêm các câu chuyện và bối cảnh mới cho mỗi cấp độ, kết hợp với đồ họa hoặc âm thanh để tăng tính hấp dẫn.
* **Hỗ trợ đa ngôn ngữ**: Thêm các ngôn ngữ khác (Anh, Pháp, Tây Ban Nha, v.v.) để mở rộng đối tượng người chơi trên toàn cầu.
* **Tích hợp API của xAI**: Kết nối với API của xAI (https://x.ai/api) để thêm các tính năng như phân tích dữ liệu người chơi hoặc tạo nội dung cấp độ động.
* **Gamification nâng cao**: Thêm bảng xếp hạng, thành tích, hoặc phần thưởng trong trò chơi để khuyến khích người chơi tiếp tục tham gia.

**Link Githup:** https://github.com/NTBinh205/Giai\_Ma.git

**Danh mục tham khảo**

[1] Stallings, W. (2017). Mật mã học và bảo mật mạng: Nguyên tắc và thực hành (ấn bản thứ 7). Pearson Education.

[2] Katz, J., & Lindell, Y. (2020). Giới thiệu về mật mã học hiện đại (ấn bản lần thứ 3). CRC Press.

[3] Schneier, B. (1996). Mật mã ứng dụng: Giao thức, thuật toán và mã nguồn trong C (ấn bản lần 2). Wiley.