**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

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

TÊN **ĐỀ** TÀI: PHÁT TRIỂN GAME GIÁO DỤC “GIẢI MÃ KHO BÁU”

**Giảng viên hướng dẫn: ThS. Trần Quý Nam**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **Mã sinh viên** |
| **1** | **Nguyễn Quang Đạt** | **1771020145** |
| **2** | **Nguyễn Đoàn Thiện Hòa** | **1771020284** |
| **3** | **Lê Phi Trường** | **1671020330** |

**Hà Nội, 2025**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

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

**TÊN ĐỀ TÀI: PHÁT TRIỂN GAME GIÁO DỤC “GIẢI MÃ KHO BÁU”**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
| 1 | 1771020145 | Nguyễn Quang Đạt | 21/11/2005 |  |  |
| 2 | 1771020284 | Nguyễn Đoàn Thiện Hòa | 02/02/2005 |  |  |
| 3 | 1671020330 | Lê Phi Trường | 24/08/2004 |  |  |

|  |  |
| --- | --- |
| **CÁN BỘ CHẤM THI 1** | **CÁN BỘ CHẤM THI 2** |
|
| **Trần Quý Nam** |  |

**Hà Nội, 2025**

LỜI NÓI ĐẦU

**(Nếu có)**

Trong bối cảnh thế giới đang dịch chuyển không ngừng vào một kỷ nguyên số hóa sâu rộng, thông tin đã trở thành tài sản vô giá và đồng thời cũng là mục tiêu hàng đầu của các mối đe dọa không gian mạng. Từ dữ liệu cá nhân, bí mật kinh doanh, cho đến hạ tầng quốc gia, mọi khía cạnh của đời sống hiện đại đều phụ thuộc vào sự an toàn và toàn vẹn của thông tin. Tuy nhiên, cùng với sự tiện lợi mà công nghệ mang lại, bức tranh bảo mật ngày càng trở nên phức tạp với vô vàn thách thức mới nảy sinh mỗi ngày. Điều này đặt ra một câu hỏi then chốt: làm thế nào để trang bị cho thế hệ công dân số những kiến thức và kỹ năng cần thiết để tự bảo vệ mình và góp phần xây dựng một không gian mạng an toàn hơn?

Mật mã học, với vai trò là nền tảng cốt lõi của an toàn thông tin, cung cấp các công cụ và nguyên lý để bảo vệ thông tin khỏi sự truy cập và biến đổi trái phép. Mặc dù vô cùng quan trọng, bản chất trừu tượng và các khái niệm phức tạp của mật mã học thường gây khó khăn cho người học, đặc biệt là những người mới bắt đầu. Liệu có một phương pháp tiếp cận nào có thể biến những nguyên lý khô khan thành trải nghiệm học tập sống động, trực quan và hấp dẫn?

Nhận thấy tiềm năng của game hóa (gamification) trong giáo dục, đặc biệt là trong việc truyền tải các kiến thức kỹ thuật phức tạp, báo cáo này hân hạnh giới thiệu dự án "Phát triển game giáo dục và mô phỏng bảo mật". Đây là một nỗ lực nhằm kiến tạo một môi trường học tập hoàn toàn mới, nơi người chơi không chỉ thụ động tiếp nhận kiến thức mà còn được đắm mình vào các thử thách mô phỏng, trực tiếp trải nghiệm sức mạnh và nguyên lý của các thuật toán mã hóa cổ điển lẫn hiện đại như Caesar, Vigenere, AES và RSA. Thay vì những bài giảng truyền thống, người chơi sẽ hóa thân thành những "thám tử bảo mật", đối mặt với các tình huống cần áp dụng tư duy giải mã và bảo vệ thông tin, từ đó không chỉ hiểu sâu hơn về mật mã mà còn rèn luyện kỹ năng tư duy phản biện và giải quyết vấn đề trong bối cảnh thực tiễn.

cáo này được cấu trúc một cách khoa học để cung cấp cái nhìn toàn diện về dự án:

**Chương 1** sẽ đặt nền móng bằng cách phân tích bối cảnh, lý do ra đời và các yêu cầu cụ thể mà game hướng tới.

**Chương 2** đi sâu vào lý thuyết, mô tả chi tiết nguyên lý hoạt động của từng thuật toán mật mã được tích hợp.

**Chương 3** hé lộ cách thức mà game được "kiến tạo" từ mã nguồn, từ thiết kế kiến trúc đến triển khai chi tiết các thuật toán.

**Chương 4** trình bày các kết quả thử nghiệm và đánh giá định lượng, định tính về hiệu quả giáo dục và tính khả dụng của sản phẩm.

**Cuối cùng, Chương 5** sẽ đưa ra những phân tích sâu sắc về đặc điểm của các thuật toán, chỉ ra các hạn chế và đề xuất những hướng phát triển tiềm năng trong tương lai.

Chúng tôi tin rằng, thông qua sự kết hợp giữa kiến thức hàn lâm và trải nghiệm giải trí, dự án này không chỉ là một bài tập lớn mà còn là một minh chứng cho tiềm năng của giáo dục game hóa, mở ra một cánh cửa mới cho việc tiếp cận và làm chủ thế giới an toàn và bảo mật thông tin.

**MỤC LỤC**

LỜI MỞ ĐẦU..............................................................................................3  
Tổng quan về đề tài.....................................................................................4  
Lý do và mục tiêu của BTL........................................................................5  
Cấu trúc báo cáo..........................................................................................6

CHƯƠNG 1: ĐẶT VẤN ĐỀ VÀ PHÂN TÍCH YÊU CẦU CỦA BÀI TOÁN...........................7  
1.1. Tổng quan về An toàn và Bảo mật Thông tin.........................................7  
1.1.1. Tầm quan trọng của an toàn và bảo mật thông tin.........................7  
1.1.2. Vai trò của mật mã trong bảo mật..................................................9  
1.2. Đặt vấn đề và Mục tiêu tổng quát của Game.......................................11  
1.2.1. Thách thức trong giáo dục về mật mã..........................................11  
1.2.2. Mục tiêu chính của game giáo dục...............................................12  
1.3. Phân tích Yêu cầu và Mục tiêu của từng Màn chơi/Game nhỏ............14  
1.3.1. Game "Cuộn Cổ Bí Ẩn" (Mật mã Caesar).......................................14  
1.3.2. Game "Nhật Ký Thám Tử" (Mật mã Vigenere)..............................16  
1.3.3. Game "Hầm Mộ Cổ Đại" / "Kho Báu Cuối Cùng" (AES)...............18  
1.3.4. Game "Két Sắt Bí Mật" / "Kho Máy Tính Cổ" (RSA).....................20  
1.3.5. Các game khác (Hash, DES,...)....................................................22  
1.4. Công nghệ sử dụng và Môi trường phát triển.....................................24  
1.4.1. Ngôn ngữ lập trình và Framework................................................24  
1.4.2. Môi trường phát triển.....................................................................26

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ MÔ TẢ THUẬT TOÁN.........................................28  
2.1. Tổng quan về Mật mã học...................................................................28  
2.1.1. Các khái niệm cơ bản.....................................................................28  
2.1.2. Phân loại mật mã...........................................................................30  
2.2. Mô tả chi tiết các Thuật toán Mã hóa và Giải mã.................................31  
2.2.1. Thuật toán Mật mã Caesar...........................................................31  
2.2.2. Thuật toán Mật mã Vigenere.........................................................33  
2.2.3. Thuật toán AES...............................................................................35  
2.2.4. Thuật toán RSA..............................................................................38  
2.2.5. Các thuật toán khác (SHA, DES...)................................................41  
2.3. Quy trình ứng dụng thuật toán trong Game........................................43  
2.3.1. Quy trình Mã hóa thông điệp..........................................................43  
2.3.2. Quy trình Giải mã thông điệp.........................................................44  
2.3.3. Quy trình Xác thực và Kiểm tra tính toàn vẹn...............................45

CHƯƠNG 3: THIẾT KẾ VÀ PHÂN TÍCH MÃ NGUỒN HỆ THỐNG GAME.....................47  
3.1. Thiết kế Hệ thống Game......................................................................47  
3.1.1. Yêu cầu chức năng và Phi chức năng của Game..........................47  
3.1.2. Kiến trúc tổng thể của Game.........................................................50  
3.1.3. Thiết kế cơ sở dữ liệu (nếu có)........................................................52  
3.1.4. Thiết kế Giao diện người dùng (UI/UX)..........................................54  
3.2. Môi trường Triển khai và Công cụ Phát triển.....................................56  
3.2.1. Ngôn ngữ lập trình, Framework, Thư viện....................................56  
3.2.2. Các công cụ phát triển khác (IDE, Git...)......................................57  
3.3. Phân tích Mã nguồn chi tiết...............................................................58  
3.3.1. Cấu trúc dự án và Tổ chức mã nguồn............................................58  
3.3.2. Triển khai các thuật toán mã hóa/giải mã.....................................59  
3.3.3. Thuật toán xác thực/toàn vẹn (SHA, RSA...).................................61  
3.3.4. Logic tương tác giữa người chơi và game....................................63

CHƯƠNG 4: THỬ NGHIỆM VÀ ĐÁNH GIÁ HIỆU QUẢ...............................................80  
4.1. Phương pháp và Quy trình Thử nghiệm.............................................80  
4.1.1. Các kịch bản thử nghiệm...............................................................80  
4.1.2. Dữ liệu thử nghiệm.........................................................................89  
4.2. Kết quả Thử nghiệm từng Thuật toán/Màn chơi................................92  
4.2.1. Caesar.............................................................................................93  
4.2.2. Vigenere.........................................................................................94  
4.2.3. AES..................................................................................................95  
4.2.4. RSA..................................................................................................96  
4.2.5. Tính năng xác thực/toàn vẹn.........................................................97  
4.3. Đánh giá Hiệu quả Giáo dục................................................................98  
4.3.1. Kiến thức về Caesar........................................................................99  
4.3.2. Kiến thức về Vigenere.....................................................................100  
4.3.3. Kiến thức về AES............................................................................101  
4.3.4. Kiến thức về RSA............................................................................102  
4.3.5. Xác thực/toàn vẹn (nếu có).............................................................103  
4.4. Đánh giá Hiệu quả Tổng thể................................................................104  
4.4.1. Hiệu quả bảo mật............................................................................106  
4.4.2. Trải nghiệm người dùng..................................................................108

CHƯƠNG 5: PHÂN TÍCH, NHẬN XÉT VÀ ĐỀ XUẤT CẢI TIẾN......................................116  
5.1. Nhận xét các Thuật toán.....................................................................117  
5.1.1. So sánh Mật mã Đối xứng..............................................................119  
5.1.2. So sánh Mật mã Bất đối xứng........................................................120  
5.1.3. Ứng dụng thực tế và độ mạnh/yếu................................................122  
5.2. Hạn chế của Dự án..............................................................................125  
5.3. Đề xuất Cải tiến và Hướng phát triển..................................................128  
5.3.1. Nội dung và tính giáo dục...............................................................129  
5.3.2. Trải nghiệm người dùng..................................................................131  
5.3.3. Công nghệ và nền tảng....................................................................133

KẾT LUẬN.................................................................................................135  
TÀI LIỆU THAM KHẢO...........................................................................137

# ĐẶT VẤN ĐỀ VÀ PHÂN TÍCH YÊU CẦU CỦA BÀI TOÁN

## Tổng quan về An toàn và Bảo mật thông tin

**1.1.1. Tầm quan trọng của an toàn và bảo mật thông tin**

Trong kỷ nguyên số hóa mạnh mẽ và toàn diện như hiện nay, thông tin đã vươn lên trở thành một trong những tài sản chiến lược và có giá trị nhất đối với mọi tổ chức, doanh nghiệp, và cả từng cá nhân. Sự phát triển vượt bậc của công nghệ thông tin đã mang lại vô số tiện ích và cơ hội, thúc đẩy mọi khía cạnh của đời sống từ kinh tế, xã hội, đến giáo dục và y tế. Tuy nhiên, đi kèm với những tiến bộ đó là sự gia tăng nhanh chóng và phức tạp của các mối đe dọa từ không gian mạng. Từ các cuộc tấn công lừa đảo (phishing) tinh vi nhằm chiếm đoạt dữ liệu cá nhân, sự lây lan của mã độc tống tiền (ransomware) gây tê liệt hệ thống, cho đến các chiến dịch tấn công có chủ đích (APT) nhắm vào hạ tầng quan trọng quốc gia, rủi ro về an ninh mạng đang hiện hữu ở khắp mọi nơi.

Việc để lộ thông tin nhạy cảm, mất quyền kiểm soát dữ liệu, hoặc hệ thống bị xâm nhập có thể dẫn đến những hậu quả nghiêm trọng và đa chiều. Các thiệt hại không chỉ giới hạn ở khía cạnh tài chính (như thất thoát doanh thu, chi phí khắc phục sự cố) mà còn mở rộng sang việc mất uy tín thương hiệu, vi phạm quyền riêng tư của khách hàng, rò rỉ bí mật kinh doanh, hoặc thậm chí là nguy cơ đối với an ninh quốc gia. Do đó, việc đảm bảo an toàn và bảo mật thông tin không còn là một lựa chọn mà đã trở thành một yêu cầu **bắt buộc** và **cấp thiết** trong mọi hoạt động trên môi trường số. Điều này đòi hỏi sự nhận thức và tham gia của không chỉ các chuyên gia công nghệ thông tin mà còn của mọi người dùng.

Để đối phó với những thách thức này, an toàn thông tin được xây dựng dựa trên ba nguyên tắc cốt lõi, thường được gọi là tam giác CIA:

**Tính bảo mật (Confidentiality):** Đảm bảo rằng thông tin chỉ có thể được truy cập bởi những đối tượng được phép. Các biện pháp như mã hóa dữ liệu, kiểm soát truy cập và xác thực mạnh mẽ là chìa khóa để duy trì tính bảo mật.

**Tính toàn vẹn (Integrity):** Đảm bảo rằng thông tin là chính xác, đầy đủ và không bị thay đổi hoặc phá hủy bởi các tác nhân không được phép (cả cố ý lẫn vô ý). Các kỹ thuật như hàm băm (hashing) và chữ ký số được sử dụng để xác minh tính toàn vẹn của dữ liệu.

**Tính sẵn sàng (Availability):** Đảm bảo rằng thông tin và các hệ thống liên quan có thể được truy cập và sử dụng bởi người dùng hợp pháp khi cần thiết. Điều này bao gồm việc chống lại các cuộc tấn công từ chối dịch vụ (DDoS) và đảm bảo khả năng phục hồi sau sự cố.

**1.1.2. Vai trò của mật mã trong bảo mật**

Trong bức tranh tổng thể của an toàn thông tin, mật mã học (Cryptography) nổi lên như một trụ cột vững chắc và không thể thiếu. Đây là một lĩnh vực khoa học và nghệ thuật nghiên cứu các phương pháp bảo vệ thông tin thông qua việc chuyển đổi dữ liệu từ dạng dễ hiểu (văn bản rõ - plaintext) sang dạng không thể hiểu được (văn bản mã hóa - ciphertext) và ngược lại. Mục tiêu cốt lõi của mật mã học là đảm bảo tính bảo mật, toàn vẹn, xác thực và chống chối bỏ của thông tin trong quá trình truyền tải, lưu trữ và xử lý.

Các kỹ thuật mật mã được phân loại rộng rãi thành ba nhóm chính:

**Mật mã đối xứng (Symmetric-key Cryptography):** Sử dụng cùng một khóa cho cả quá trình mã hóa và giải mã. Ưu điểm là tốc độ nhanh, phù hợp để mã hóa lượng lớn dữ liệu. Các ví dụ điển hình bao gồm Data Encryption Standard (DES), Advanced Encryption Standard (AES) và các mật mã cổ điển như Caesar, Vigenere.

**Mật mã bất đối xứng (Asymmetric-key Cryptography) hay Mật mã khóa công khai (Public-key Cryptography):** Sử dụng một cặp khóa riêng biệt – khóa công khai (public key) để mã hóa và khóa bí mật (private key) để giải mã, hoặc ngược lại để tạo chữ ký số. Ưu điểm là giải quyết bài toán phân phối khóa, cho phép xác thực và chống chối bỏ. Thuật toán RSA là một ví dụ nổi bật.

**Hàm băm mật mã (Cryptographic Hash Functions):** Là các hàm toán học một chiều, biến đổi dữ liệu đầu vào thành một chuỗi giá trị có độ dài cố định (hash value hoặc message digest). Đặc tính quan trọng của hàm băm là không thể đảo ngược để tìm ra dữ liệu gốc, và một thay đổi nhỏ trong dữ liệu đầu vào cũng tạo ra một giá trị băm hoàn toàn khác. Chúng được sử dụng để kiểm tra tính toàn vẹn của dữ liệu, lưu trữ mật khẩu an toàn và trong chữ ký số. Ví dụ bao gồm SHA-256 (Secure Hash Algorithm 256).

Nhờ vào các thuật toán và nguyên lý này, mật mã học đã trở thành nền tảng cho hàng loạt ứng dụng bảo mật hiện đại, từ việc mã hóa email, bảo vệ giao dịch ngân hàng trực tuyến (SSL/TLS), xây dựng mạng riêng ảo (VPN), cho đến đảm bảo tính an toàn cho các hệ thống thông tin quân sự và chính phủ. Vai trò của mật mã học không chỉ dừng lại ở việc bảo vệ dữ liệu mà còn củng cố niềm tin vào các hệ thống số, cho phép giao tiếp và trao đổi thông tin một cách an toàn trong một thế giới ngày càng kết nối.

## Đặt vấn đề và Mục tiêu tông quất của game

**1.2.1. Thách thức trong giáo dục về mật mã**

Mặc dù mật mã học đóng vai trò trung tâm trong an toàn và bảo mật thông tin, việc truyền đạt các kiến thức này một cách hiệu quả lại là một thách thức không nhỏ đối với cả người dạy và người học. Bản chất của mật mã học thường liên quan đến các khái niệm toán học trừu tượng, thuật toán phức tạp và logic khô khan, đòi hỏi người học phải có khả năng tư duy phân tích và hình dung cao.

Các phương pháp giảng dạy truyền thống, chủ yếu dựa trên lý thuyết, công thức, và các ví dụ tĩnh trên sách vở hoặc bài giảng, thường khó tạo được sự hứng thú và động lực khám phá cho người học. Điều này dẫn đến tình trạng người học có thể nắm được lý thuyết nhưng lại gặp khó khăn khi áp dụng vào thực tế, hoặc đơn giản là cảm thấy nhàm chán, khó tiếp thu và nhanh chóng quên đi kiến thức. Việc thiếu đi một môi trường tương tác, nơi người học có thể trực tiếp thử nghiệm, mắc lỗi và tự mình khám phá nguyên lý hoạt động của các thuật toán, đã trở thành một rào cản lớn trong việc nâng cao chất lượng giáo dục về mật mã học. Chính từ những thách thức này, một phương pháp tiếp cận mới, sáng tạo và hấp dẫn hơn là cần thiết để khơi gợi niềm đam mê và trang bị kiến thức vững chắc về bảo mật cho thế hệ mới.

**1.2.2. Mục tiêu chính của game giáo dục**

Để giải quyết những vấn đề nêu trên, dự án "Phát triển game giáo dục và mô phỏng bảo mật" (tên game nếu có, ví dụ: **"Mật Mã Di Sản"** hoặc **"Giải Mã Kho Báu"**) được hình thành với mục tiêu tổng quát là biến quá trình học tập về mật mã học trở nên trực quan, hấp dẫn và hiệu quả thông qua phương pháp game hóa (gamification). Game không chỉ đơn thuần là một công cụ giải trí mà còn là một nền tảng giáo dục tương tác, nơi người chơi được trải nghiệm, vận dụng và củng cố kiến thức một cách chủ động.

Cụ thể, game hướng đến các mục tiêu chính sau:

**Cung cấp kiến thức nền tảng về mật mã học:** Giúp người chơi hiểu rõ nguyên lý hoạt động cơ bản của các thuật toán mã hóa và giải mã phổ biến, từ các mật mã cổ điển như Caesar, Vigenere đến các thuật toán hiện đại như AES và RSA. Game sẽ minh họa cách thức dữ liệu được biến đổi và bảo vệ.

**Rèn luyện kỹ năng thực hành và tư duy logic:** Tạo cơ hội cho người chơi trực tiếp áp dụng các thuật toán vào các tình huống mô phỏng cụ thể, như giải mã các thông điệp bí ẩn, bảo vệ dữ liệu quan trọng. Quá trình này khuyến khích người chơi tư duy phân tích, suy luận và giải quyết vấn đề một cách có hệ thống.

**Nâng cao nhận thức về tầm quan trọng của bảo mật thông tin:** Thông qua các nhiệm vụ và cốt truyện trong game, người chơi sẽ nhận thức được giá trị của thông tin được bảo vệ và những rủi ro tiềm ẩn khi thông tin không được mã hóa hoặc xác thực đúng cách.

**Tạo môi trường học tập tương tác và hấp dẫn:** Biến các khái niệm khô khan thành những thử thách thú vị, khơi gợi niềm đam mê tìm hiểu và khám phá trong lĩnh vực an toàn thông tin, thay vì cảm giác bị ép buộc học.

**Minh họa ứng dụng thực tế:** Đặt các thuật toán mật mã vào các ngữ cảnh giả lập gần gũi với thực tế (ví dụ: giao dịch ngân hàng, bảo vệ tài liệu mật) để người chơi thấy được ý nghĩa và ứng dụng của chúng.

Thông qua việc kết hợp yếu tố giải trí của trò chơi với nội dung giáo dục chuyên sâu, dự án kỳ vọng sẽ mở ra một phương pháp tiếp cận mới, giúp việc học về an toàn và bảo mật thông tin trở nên dễ dàng và hiệu quả hơn bao giờ hết.

**1.2.1 Thách thức:**

Nhấn mạnh tính trừu tượng và phức tạp của mật mã.

Chỉ ra hạn chế của phương pháp truyền thống (thụ động, thiếu tương tác).

Tạo "vấn đề" cần giải quyết để dẫn dắt đến giải pháp là game.

**1.2.2 Mục tiêu:**

Bắt đầu bằng việc khẳng định game là giải pháp cho vấn đề đã nêu.

Liệt kê các mục tiêu rõ ràng, súc tích bằng các gạch đầu dòng, dễ đọc và dễ nắm bắt.

Mỗi mục tiêu đều nêu bật được khía cạnh "giáo dục" và "thực hành" của game.

Kết thúc bằng một câu tổng kết mạnh mẽ, khẳng định giá trị của dự án.

## Phân tích yêu cầu và mục tiêu của màn chơi

Game **"Giải Mã Kho Báu"** được thiết kế dưới dạng một chuỗi các cấp độ (level), mỗi cấp độ là một thử thách giải mã riêng biệt, được lồng ghép vào một cốt truyện hấp dẫn về hành trình khám phá kho báu của Thám tử Smith. Các màn chơi được sắp xếp theo độ khó tăng dần và tập trung vào việc áp dụng các thuật toán mật mã khác nhau, từ cổ điển đến hiện đại. Điều này giúp người chơi không chỉ hiểu sâu về nguyên lý hoạt động của từng thuật toán mà còn nhận diện được bối cảnh áp dụng và mức độ bảo mật của chúng trong các tình huống thực tiễn.

**1.3.1. Mật mã Caesar (Đại diện: Game "Cuộn Cổ Bí Ẩn")**

**Mục tiêu giáo dục:** Giúp người chơi nắm vững nguyên lý dịch chuyển ký tự đơn giản của mật mã Caesar – một trong những mật mã thay thế sớm nhất và dễ hiểu nhất. Qua đó, người chơi hình dung được cách thức mã hóa và giải mã cơ bản, đồng thời nhận diện tính yếu kém và các phương pháp tấn công đơn giản của nó (ví dụ: brute-force).

**Tình huống game:** Người chơi nhập vai Thám tử Smith, bắt đầu cuộc hành trình bằng việc khám phá một "Cuộn Cổ Bí Ẩn" chứa thông điệp được mã hóa bằng mật mã Caesar. Tình huống này được đặt trong bối cảnh lịch sử, gợi mở về những manh mối đầu tiên dẫn đến kho báu.

**Yêu cầu người chơi:** Người chơi được cung cấp văn bản đã mã hóa ("WKDP WX VPLWK GD WLP WKDR NKR EDX WKDW WX") và một giao diện trực quan để thử nghiệm các khóa dịch chuyển. Game cung cấp các gợi ý ("Mật mã Caesar dịch chuyển các chữ cái trong bảng chữ cái.", "Thử với số dịch chuyển nhỏ, từ 1 đến 10.") để định hướng người chơi tìm ra khóa chính xác là "3". Sau khi nhập đúng khóa, game sẽ hiển thị văn bản gốc ("THAM TU SMITH DA TIM THAY KHO BAU THAT TU"), giúp người chơi trực tiếp trải nghiệm cơ chế dịch chuyển và thấy kết quả giải mã.

**Các biến thể trong game:** Để củng cố kiến thức và khả năng thích nghi của người chơi, game còn tích hợp thêm hai màn chơi khác sử dụng mật mã Caesar: "Bức Thư Cổ" (Level 3) với khóa "7" và "Bản Đồ Cướp Biển" (Level 7) với khóa "13" (ROT13). Các màn này giúp người chơi làm quen với việc thay đổi khóa và nhận diện các biến thể của thuật toán dịch chuyển.

**1.3.2. Mật mã Vigenere (Đại diện: Game "Nhật Ký Thám Tử")**

**Mục tiêu giáo dục:** Giới thiệu về mật mã Vigenere, một bước tiến quan trọng hơn so với Caesar nhờ sử dụng từ khóa và nguyên lý mã hóa đa bảng. Màn chơi này giúp người chơi hiểu được khái niệm mã hóa đa bảng, nhận ra sự phức tạp tăng lên và khả năng chống chịu tốt hơn của Vigenere so với các mật mã thay thế đơn bảng.

**Tình huống game:** Thám tử Smith tiếp tục cuộc điều tra bằng việc tìm thấy "Nhật Ký Thám Tử" của một người tiền nhiệm, chứa đựng các bí mật được bảo vệ bằng mật mã Vigenere. Tình huống này gợi mở về tính bảo mật cao hơn của thông tin trong nhật ký.

**Văn bản mã hóa (ciphertext):** "NGIY LEI SHUBG TA SE TIM THIY KHO BIU"

**Yêu cầu người chơi:** Người chơi được cung cấp văn bản mã hóa và giao diện nhập từ khóa. Game cung cấp các gợi ý ("Mật mã Vigenère sử dụng một từ khóa để mã hóa.", "Từ khóa liên quan đến nghề nghiệp của Smith.") để hướng dẫn người chơi suy luận từ khóa. Người chơi cần hiểu rằng mỗi ký tự trong văn bản sẽ được dịch chuyển một khoảng cách khác nhau tùy thuộc vào ký tự tương ứng trong từ khóa (từ khóa được lặp lại). Khóa chính xác (correct\_key) cho màn này là "DETECTIVE". Sau khi nhập đúng, văn bản gốc ("NGAY MAI CHUNG TA SE TIM THAY KHO BAU") sẽ được hiển thị, minh chứng cho việc Vigenere có khả năng chống lại phân tích tần suất tốt hơn Caesar.

**Các biến thể trong game:** Màn "Mật Thư Hoàng Gia" (Level 5) cũng sử dụng mật mã Vigenere với từ khóa "KING", tiếp tục củng cố kiến thức về thuật toán này trong một bối cảnh khác, giúp người chơi linh hoạt hơn trong việc áp dụng.

**1.3.3. Mật mã AES (Đại diện: Game "Hầm Mộ Cổ Đại")**

**Mục tiêu giáo dục:** Giới thiệu về thuật toán mã hóa đối xứng hiện đại Advanced Encryption Standard (AES) – tiêu chuẩn mã hóa được sử dụng rộng rãi nhất hiện nay. Màn chơi này nhấn mạnh sức mạnh vượt trội, tính bảo mật cao và tầm quan trọng của AES trong việc bảo vệ dữ liệu nhạy cảm, đồng thời giúp người chơi hiểu rằng AES là thuật toán rất khó bị phá vỡ nếu không có khóa bí mật chính xác.

**Tình huống game:** Người chơi đối mặt với "Hầm Mộ Cổ Đại" chứa đựng những bí mật tối mật, được khóa chặt bằng công nghệ mã hóa tiên tiến nhất – AES. Cốt truyện gợi ý rằng "ai đó đã sử dụng công nghệ mã hóa tiên tiến để bảo vệ bí mật này", tăng tính thử thách và tầm quan trọng của màn chơi.

**Văn bản mã hóa (ciphertext):** "VFJFQVNVUkUyMDI0ISBLSE8gQkFVIERBIFRJTSBUSEFZIQ==" (đã được Base64 encode, văn bản gốc là "TREASURE2024! KHO BAU DA TIM THAY!")

**Yêu cầu người chơi:** Game trình bày văn bản đã mã hóa bằng AES và yêu cầu người chơi tìm ra "chìa khóa" bí mật để giải mã. Khác với các mật mã cổ điển, việc giải mã AES đòi hỏi khóa phải chính xác tuyệt đối, nhấn mạnh tính bảo mật nghiêm ngặt của thuật toán này. Các gợi ý ("Mật mã AES là tiêu chuẩn mã hóa hiện đại.", "Khóa có thể là tên kho báu kèm theo năm hiện tại.") định hướng người chơi tìm khóa. Khóa chính xác (correct\_key) là "TREASURE2024!". Sau khi nhập đúng, game sẽ hiển thị văn bản gốc, làm nổi bật tầm quan trọng của khóa trong AES.

**Các biến thể trong game:** Màn "Kho Báu Cuối Cùng" (Level 9) là thử thách đỉnh cao, cũng sử dụng AES với một khóa phức tạp hơn ("FINAL\_TREASURE\_2024"), củng cố kiến thức và đẩy độ khó lên mức cao nhất, mô phỏng việc bảo vệ thông tin cực kỳ quan trọng.

**1.3.4. Mật mã RSA (Đại diện: Game "Két Sắt Bí Mật")**

**Mục tiêu giáo dục:** Giới thiệu khái niệm về mật mã bất đối xứng (Public-key Cryptography) thông qua thuật toán RSA – một trong những thuật toán nền tảng của mật mã khóa công khai. Màn chơi này làm rõ sự khác biệt giữa khóa công khai và khóa bí mật, và vai trò của chúng trong trao đổi khóa an toàn, chữ ký số và xác thực.

**Tình huống game:** Người chơi được giao nhiệm vụ mở "Két Sắt Bí Mật" hoặc truy cập dữ liệu từ một "Kho Máy Tính Cổ" được bảo vệ bởi hệ thống mật mã RSA. Tình huống này mô phỏng các giao dịch an toàn hoặc truy cập tài liệu mật trong môi trường thực tế, nơi tính xác thực và quyền riêng tư là tối quan trọng. Cốt truyện nhấn mạnh đây là "thử thách khó khăn" đòi hỏi "sự thông minh và kiên nhẫn".

**Văn bản mã hóa (ciphertext):** "Q1JZUFRPIEtITyBCQVUgTkFZIE8gREFZIQ==" (đã được Base64 encode, văn bản gốc là "CRYPTO KHO BAU NAY O DAY!")

**Yêu cầu người chơi:** Game trình bày một thông điệp đã được mã hóa (có thể ở dạng Base64). Người chơi cần nhận biết rằng để giải mã thông điệp này, họ phải sử dụng khóa bí mật tương ứng (không phải khóa công khai). Các gợi ý ("Mật mã RSA sử dụng cặp khóa công khai và riêng tư.", "Từ khóa liên quan đến khoa học mật mã học.") sẽ dẫn dắt người chơi đến việc suy luận khóa bí mật. Khóa chính xác (correct\_key) là "CRYPTO". Khi nhập đúng, văn bản gốc ("CRYPTO KHO BAU NAY O DAY!") sẽ được giải mã, giúp người chơi hình dung nguyên lý hoạt động của mật mã khóa công khai.**Các biến thể trong game:** Màn "Kho Máy Tính Cổ" (Level 8) cũng sử dụng mật mã RSA với từ khóa "SECRET", tiếp tục củng cố hiểu biết về RSA và ứng dụng của nó trong việc bảo vệ dữ liệu trên các hệ thống điện tử, đặc biệt là trong bối cảnh truy cập thông tin mật.

## Công nghệ sử dụng và Môi trường phát triển

Dự án game **"Giải Mã Kho Báu"** được phát triển trên nền tảng web, tuân theo mô hình client-server, nhằm mang đến trải nghiệm tương tác trực quan và dễ tiếp cận cho người học. Việc lựa chọn công nghệ và môi trường phát triển được cân nhắc để đảm bảo tính đơn giản, hiệu quả và khả năng minh họa rõ ràng các nguyên lý mật mã học.

**1.4.1. Ngôn ngữ lập trình và Framework**

**Ngôn ngữ lập trình chính:** Dự án sử dụng **Python** làm ngôn ngữ lập trình chính cho cả logic backend và xử lý mật mã. Python được lựa chọn nhờ cú pháp rõ ràng, dễ học, và cộng đồng hỗ trợ lớn, phù hợp cho một dự án giáo dục.

**Loại game và Kiến trúc:** Đây là một **web game tương tác đơn giản** hoạt động trên nền tảng Flask. Game triển khai theo mô hình client-server, nơi trình duyệt của người dùng (client) gửi yêu cầu đến máy chủ (server) Flask để nhận các trang HTML/CSS/JavaScript và xử lý logic game.

**Frontend:**

Giao diện người dùng được xây dựng hoàn toàn bằng **HTML/CSS và JavaScript thuần**. Dự án không sử dụng các thư viện hay framework JavaScript hiện đại như ReactJS, Vue.js, hay Angular. Thay vào đó, tập trung vào việc tạo ra một giao diện đơn giản, trực quan, chủ yếu hiển thị nội dung văn bản và các biểu mẫu (form) nhập liệu để người chơi tương tác giải mã.

**Backend:**

Sử dụng **Flask** – một micro-framework nhẹ và linh hoạt của Python. Flask cho phép tạo và quản lý các route (đường dẫn URL), xử lý các yêu cầu (request) từ trình duyệt và gửi lại các phản hồi (response) một cách hiệu quả. Sự đơn giản của Flask rất phù hợp với quy mô và mục tiêu của dự án.

**Thư viện Mật mã học:**

Một điểm đặc biệt của dự án là việc **tự cài đặt (self-implement) các thuật toán mã hóa** như Caesar, Vigenère, RSA và AES trong file crypto\_utils.py. Dự án không sử dụng các thư viện mật mã học bên ngoài như PyCryptodome hoặc cryptography. Phương pháp này được ưu tiên nhằm giúp người học hiểu sâu hơn về cơ chế hoạt động, từng bước của các thuật toán, thay vì chỉ sử dụng các hàm đã được đóng gói sẵn.

**1.4.2. Môi trường phát triển**

**Môi trường Phát triển Tích hợp (IDE):**

Dự án được triển khai và phát triển hiệu quả trong môi trường **Replit**, thể hiện qua sự hiện diện của các tệp cấu hình như .replit, replit.md, và uv.lock. Replit cung cấp một môi trường phát triển dựa trên đám mây tiện lợi, cho phép cộng tác và triển khai nhanh chóng.

Ngoài ra, cấu trúc dự án và mã nguồn Python chuẩn cũng cho phép dễ dàng phát triển và chạy game trên các IDE phổ biến khác như **Visual Studio Code**, **PyCharm**, hoặc bất kỳ môi trường hỗ trợ phát triển Python nào.

**Hệ thống Quản lý Phiên bản:**

Dự án có sử dụng **Git** làm hệ thống quản lý phiên bản, được minh chứng qua sự tồn tại của thư mục .git/. Git giúp theo dõi các thay đổi trong mã nguồn, quản lý các phiên bản khác nhau của dự án và hỗ trợ làm việc nhóm. Mặc dù chưa rõ kết nối tới nền tảng lưu trữ Git cụ thể (như GitHub hay GitLab), cấu trúc Git nội bộ vẫn đảm bảo quy trình phát triển có tổ chức.

**Hệ điều hành Phát triển:**

Dựa trên các công nghệ và công cụ được sử dụng (Python, Flask, Replit), dự án có khả năng tương thích cao và có thể được phát triển trên nhiều hệ điều hành phổ biến như **Windows 10, Ubuntu Linux, hoặc macOS**. Quá trình phát triển nhiều khả năng diễn ra trên Windows hoặc trong môi trường Linux ảo hóa của Replit.

**Cơ sở dữ liệu:**

Trong phiên bản hiện tại, game **không sử dụng cơ sở dữ liệu** riêng biệt. Toàn bộ dữ liệu game, bao gồm các màn chơi, cốt truyện, văn bản mã hóa/giải mã và đáp án, được lưu trữ dưới dạng danh sách các đối tượng Python (list of dictionaries) trực tiếp trong mã nguồn (app.py). Điều này giúp đơn giản hóa cấu trúc dự án cho mục đích giáo dục và minh họa.

# CƠ SỞ LÝ THUYẾT VÀ MÔ TẢ THUẬT TOÁN

Trong chương này, chúng tôi sẽ trình bày tổng quan về các khái niệm cốt lõi trong mật mã học, sau đó đi sâu vào nguyên lý hoạt động của các thuật toán mật mã đã được tích hợp và mô phỏng trong game "Giải Mã Kho Báu". Việc nắm vững cơ sở lý thuyết của các thuật toán này là nền tảng để hiểu cách game mô phỏng chúng, cũng như nhận thức được ưu, nhược điểm và ứng dụng thực tế của từng loại mật mã.

## Tổng quan về Mật mã học

**2.1.1. Các khái niệm cơ bản**

Mật mã học (Cryptography) là một lĩnh vực khoa học nghiên cứu về các kỹ thuật để bảo vệ thông tin và truyền thông an toàn trong sự hiện diện của các bên thứ ba (thù địch). Để hiểu về các thuật toán mật mã, cần nắm vững một số khái niệm cơ bản sau:

**Mã hóa (Encryption):** Quá trình chuyển đổi thông tin từ dạng dễ hiểu (văn bản rõ) sang dạng không thể hiểu được (văn bản mã hóa) nhằm bảo vệ tính bảo mật của dữ liệu. Quá trình này thường sử dụng một thuật toán mã hóa và một khóa.

**Giải mã (Decryption):** Quá trình ngược lại của mã hóa, tức là chuyển đổi thông tin từ dạng văn bản mã hóa trở lại dạng văn bản rõ ban đầu, giúp người nhận hợp lệ có thể đọc được nội dung. Quá trình này thường sử dụng một thuật toán giải mã và một khóa.

**Khóa (Key):** Một chuỗi bit hoặc ký tự bí mật được sử dụng trong quá trình mã hóa và giải mã. Độ bảo mật của hệ thống mật mã phụ thuộc rất lớn vào độ mạnh của khóa và cách quản lý khóa.

**Văn bản rõ (Plaintext):** Thông tin gốc, ở dạng dễ đọc và dễ hiểu trước khi được mã hóa.

**Văn bản mã hóa (Ciphertext):** Thông tin sau khi đã được mã hóa, ở dạng không thể đọc được nếu không có khóa giải mã phù hợp. Mục đích của văn bản mã hóa là bảo vệ nội dung gốc khỏi sự truy cập trái phép.

**Thuật toán mật mã (Cipher/Algorithm):** Một tập hợp các quy tắc toán học hoặc logic được sử dụng để thực hiện quá trình mã hóa và giải mã.

**2.1.2. Phân loại mật mã**

Mật mã học hiện đại có thể được phân loại thành ba nhánh chính dựa trên cách thức sử dụng khóa:

**Mật mã đối xứng (Symmetric-key Cryptography):**

**Khái niệm:** Là phương pháp mã hóa mà trong đó, **cùng một khóa** được sử dụng cho cả quá trình mã hóa và giải mã. Điều này có nghĩa là cả người gửi và người nhận đều phải có và giữ bí mật khóa này.

**Đặc điểm:** Tốc độ xử lý nhanh, hiệu quả cho việc mã hóa lượng lớn dữ liệu. Tuy nhiên, thách thức lớn nhất là làm thế nào để hai bên trao đổi khóa bí mật một cách an toàn trên một kênh không bảo mật.

**Ví dụ:** Data Encryption Standard (DES), Advanced Encryption Standard (AES), và các mật mã cổ điển như Caesar, Vigenere.

**Mật mã bất đối xứng (Asymmetric-key Cryptography / Public-key Cryptography):**

**Khái niệm:** Là phương pháp mã hóa sử dụng **một cặp khóa riêng biệt**: một **khóa công khai (public key)** và một **khóa bí mật (private key)**. Khóa công khai có thể được chia sẻ rộng rãi, trong khi khóa bí mật phải được giữ kín bởi chủ sở hữu.

Thông điệp được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng.

Thông điệp được ký bằng khóa bí mật có thể được xác minh bằng khóa công khai tương ứng (đảm bảo tính xác thực và chống chối bỏ).

**Đặc điểm:** Giải quyết được vấn đề phân phối khóa của mật mã đối xứng, cho phép trao đổi thông tin an toàn giữa hai bên mà không cần chia sẻ trước khóa bí mật. Tuy nhiên, tốc độ xử lý chậm hơn mật mã đối xứng, nên thường được dùng để trao đổi khóa đối xứng hoặc ký số.

**Ví dụ:** Rivest–Shamir–Adleman (RSA), Diffie-Hellman, Elliptic Curve Cryptography (ECC).

**Hàm băm mật mã (Cryptographic Hash Functions):**

**Khái niệm:** Là các hàm toán học một chiều, biến đổi dữ liệu đầu vào (có độ dài bất kỳ) thành một chuỗi giá trị có độ dài cố định, được gọi là **giá trị băm (hash value)** hay **thông điệp tóm tắt (message digest)**.

**Đặc điểm:**

**Một chiều:** Không thể đảo ngược để tìm ra dữ liệu gốc từ giá trị băm.

**Kháng va chạm:** Rất khó (gần như không thể) tìm được hai đầu vào khác nhau mà tạo ra cùng một giá trị băm.

**Nhạy cảm với thay đổi:** Một thay đổi nhỏ nhất trong dữ liệu đầu vào cũng sẽ tạo ra một giá trị băm hoàn toàn khác

**Ứng dụng:** Chủ yếu được sử dụng để kiểm tra tính toàn vẹn của dữ liệu (đảm bảo dữ liệu không bị thay đổi trong quá trình truyền tải hoặc lưu trữ), lưu trữ mật khẩu an toàn (lưu băm của mật khẩu thay vì mật khẩu gốc), và trong chữ ký số.

**Ví dụ:** Secure Hash Algorithm (SHA-256, SHA-3), Message Digest 5 (MD5 - hiện đã không còn an toàn cho mục đích bảo mật).

Những kiến thức cơ bản này sẽ là nền tảng để chúng ta tìm hiểu sâu hơn về các thuật toán cụ thể được sử dụng trong game "Giải Mã Kho Báu" ở các phần tiếp theo

## Mô hinhg chi tiết các Thuật toán Mã hóa và Giải mã

Phần này sẽ đi sâu vào nguyên lý hoạt động của các thuật toán mật mã chính được tích hợp và mô phỏng trong game "Giải Mã Kho Báu", cung cấp cái nhìn chi tiết hơn về cơ chế toán học và logic đằng sau mỗi phương pháp bảo mật.

**2.2.1. Thuật toán Mật mã Caesar (Caesar Cipher)**

**Nguyên lý hoạt động:** Mật mã Caesar, còn được gọi là mật mã dịch chuyển (shift cipher), là một trong những kỹ thuật mã hóa đơn giản và cổ xưa nhất, được cho là đã được Julius Caesar sử dụng. Nguyên lý của nó rất đơn giản: mỗi chữ cái trong văn bản rõ (plaintext) sẽ được thay thế bằng một chữ cái khác trong bảng chữ cái, cách nó một số vị trí cố định theo một chiều nhất định. Số vị trí dịch chuyển này chính là **khóa (key)** của mật mã.

**Mã hóa:** Để mã hóa, mỗi chữ cái của văn bản rõ được dịch chuyển sang phải (tiến lên) một số vị trí xác định bởi khóa trong bảng chữ cái. Khi dịch chuyển đến cuối bảng chữ cái, nó sẽ quay vòng về đầu.

**Giải mã:** Để giải mã, quá trình ngược lại được thực hiện: mỗi chữ cái của văn bản mã hóa được dịch chuyển sang trái (lùi lại) cùng một số vị trí xác định bởi khóa.

**Công thức toán học:** Xét bảng chữ cái tiếng Anh từ A đến Z (hoặc tiếng Việt không dấu, không phân biệt chữ hoa/thường) được gán giá trị số từ 0 đến 25.

**Mã hóa (Ek​(x)):** Ek​(x)=(x+k)(mod26)

**Giải mã (Dk​(y)):** Dk​(y)=(y−k)(mod26) Trong đó:

x là giá trị số của chữ cái trong văn bản rõ.

y là giá trị số của chữ cái trong văn bản mã hóa.

k là giá trị số của khóa dịch chuyển (từ 1 đến 25).

(mod26) là phép toán modulo 26, đảm bảo kết quả luôn nằm trong phạm vi 0-25 (quay vòng bảng chữ cái).

**Ví dụ minh họa:** Giả sử chúng ta muốn mã hóa văn bản rõ "HELLO" với khóa k=3. Sử dụng bảng chữ cái: A=0, B=1, ..., Z=25.

**H** (7) → (7+3)(mod26)=10(mod26)=10→ **K**

**E** (4) → (4+3)(mod26)=7(mod26)=7→ **H**

**L** (11) → (11+3)(mod26)=14(mod26)=14→ **O**

**L** (11) → (11+3)(mod26)=14(mod26)=14→ **O**

**O** (14) → (14+3)(mod26)=17(mod26)=17→ **R**

Vậy, văn bản mã hóa của "HELLO" với khóa 3 là "KHOOR".

Để giải mã "KHOOR" với khóa 3:

**K** (10) → (10−3)(mod26)=7(mod26)=7→ **H**

**H** (7) → (7−3)(mod26)=4(mod26)=4→ **E**

**O** (14) → (14−3)(mod26)=11(mod26)=11→ **L**

**O** (14) → (14−3)(mod26)=11(mod26)=11→ **L**

**R** (17) → (17−3)(mod26)=14(mod26)=14→ **O**

Kết quả giải mã là "HELLO".

**Ưu điểm, Nhược điểm và Ứng dụng:**

**Ưu điểm:** Cực kỳ đơn giản để hiểu và triển khai thủ công.

**Nhược điểm:** Mật mã Caesar rất dễ bị phá vỡ. Do chỉ có 25 khả năng dịch chuyển (khóa), kẻ tấn công có thể dễ dàng thử tất cả các khóa cho đến khi tìm ra văn bản rõ (tấn công vét cạn - brute-force attack). Ngoài ra, phân tích tần suất chữ cái (tìm kiếm các chữ cái xuất hiện thường xuyên nhất) cũng có thể giúp giải mã nhanh chóng.

**Ứng dụng:** Ngày nay, mật mã Caesar không còn được sử dụng để bảo mật thông tin nghiêm túc. Nó chủ yếu được dùng trong giáo dục để giới thiệu các khái niệm mật mã học cơ bản, hoặc trong các câu đố, trò chơi giải mã như trong game "Giải Mã Kho Báu" để minh họa nguyên lý. Một biến thể phổ biến là ROT13 (Caesar với khóa 13), thường được dùng để che giấu nội dung spoil hoặc câu trả lời trong các diễn đàn trực tuyến một cách đơn giản.

**2.2.2. Thuật toán Mật mã Vigenere (Vigenere Cipher)**

**Nguyên lý hoạt động:** Mật mã Vigenere là một phương pháp mã hóa thay thế đa bảng (polyalphabetic substitution cipher) do Giovan Battista Bellaso mô tả lần đầu vào năm 1553, sau này được Etienne de Vigenère phát triển. Nó được coi là một bước tiến đáng kể so với mật mã Caesar vì khả năng chống lại phương pháp tấn công phân tích tần suất chữ cái (frequency analysis) – một điểm yếu lớn của các mật mã đơn bảng.

Nguyên lý của Vigenere dựa trên việc sử dụng một **từ khóa (keyword)** lặp lại. Thay vì dùng cùng một số dịch chuyển cho tất cả các chữ cái như Caesar, Vigenere sử dụng các chữ cái của từ khóa để xác định độ dịch chuyển khác nhau cho từng chữ cái trong văn bản rõ.

**Mã hóa:** Mỗi chữ cái trong văn bản rõ sẽ được mã hóa bằng một mật mã Caesar khác nhau, được xác định bởi chữ cái tương ứng trong từ khóa. Từ khóa sẽ được lặp lại cho đến khi có độ dài bằng với văn bản rõ.

**Giải mã:** Để giải mã, quá trình ngược lại được thực hiện. Mỗi chữ cái của văn bản mã hóa được dịch chuyển ngược lại (lùi về) một số vị trí tương ứng với chữ cái trong từ khóa

**Công thức toán học:** Xét bảng chữ cái tiếng Anh từ A đến Z được gán giá trị số từ 0 đến 25. Giả sử:

P=P0​P1​...Pn−1​ là văn bản rõ có độ dài n.

K=K0​K1​...Km−1​ là từ khóa có độ dài m.

Từ khóa được lặp lại để có độ dài bằng văn bản rõ: K′=K0​K1​...Km−1​K0​K1​...

**Mã hóa (EK′​(Pi​)):** EK′​(Pi​)=(Pi​+Ki′​)(mod26)

**Giải mã (DK′​(Ci​)):** DK′​(Ci​)=(Ci​−Ki′​)(mod26) Trong đó:

Pi​ là giá trị số của chữ cái thứ i trong văn bản rõ.

Ci​ là giá trị số của chữ cái thứ i trong văn bản mã hóa.

Ki′​ là giá trị số của chữ cái thứ i trong từ khóa đã được lặp lại.

(mod26) là phép toán modulo 26 để quay vòng trong bảng chữ cái.

**Ví dụ minh họa:** Giả sử chúng ta muốn mã hóa văn bản rõ "ATTACKATDAWN" với từ khóa "LEMON". Sử dụng bảng chữ cái: A=0, B=1, ..., Z=25.

**Lặp lại từ khóa:** LEMONLEMONLE

**Mã hóa từng chữ cái:**

**A** (0) + **L** (11) → (0+11)(mod26)=11→ **L**

**T** (19) + **E** (4) → (19+4)(mod26)=23→ **X**

**T** (19) + **M** (12) → (19+12)(mod26)=31(mod26)=5→ **F**

**A** (0) + **O** (14) → (0+14)(mod26)=14→ **O**

**C** (2) + **N** (13) → (2+13)(mod26)=15→ **P**

**K** (10) + **L** (11) → (10+11)(mod26)=21→ **V**

**A** (0) + **E** (4) → (0+4)(mod26)=4→ **E**

**T** (19) + **M** (12) → (19+12)(mod26)=31(mod26)=5→ **F**

**D** (3) + **O** (14) → (3+14)(mod26)=17→ **R**

**A** (0) + **N** (13) → (0+13)(mod26)=13→ **N**

**W** (22) + **L** (11) → (22+11)(mod26)=33(mod26)=7→ **H**

**N** (13) + **E** (4) → (13+4)(mod26)=17→ **R**

Vậy, văn bản mã hóa của "ATTACKATDAWN" với từ khóa "LEMON" là "LXFOPVEFRNHR".

**Ưu điểm, Nhược điểm và Ứng dụng:**

**Ưu điểm:** So với Caesar, Vigenere có độ an toàn cao hơn đáng kể vì nó chống lại được phương pháp phân tích tần suất chữ cái. Mỗi chữ cái trong văn bản rõ có thể được mã hóa thành các chữ cái khác nhau trong văn bản mã hóa tùy thuộc vào vị trí của nó và từ khóa, làm cho việc giải mã thủ công trở nên khó khăn hơn nhiều nếu không biết từ khóa.

**Nhược điểm:** Mặc dù mạnh hơn Caesar, Vigenere vẫn có thể bị phá vỡ bằng các phương pháp như phân tích Kasiski hoặc kiểm tra chỉ số trùng hợp (index of coincidence) để tìm ra độ dài của từ khóa, sau đó thực hiện phân tích tần suất trên từng chu kỳ của từ khóa.

**Ứng dụng:** Vigenere từng được sử dụng trong các hoạt động ngoại giao và quân sự trong nhiều thế kỷ. Ngày nay, nó không còn được coi là an toàn cho mục đích bảo mật thông tin nhạy cảm nhưng vẫn là một công cụ giáo dục tuyệt vời để minh họa sự phát triển của mật mã từ đơn giản đến phức tạp hơn, cũng như các nguyên tắc cơ bản của mật mã đa bảng. Trong game "Giải Mã Kho Báu", nó đóng vai trò là một thử thách ở mức độ trung bình, yêu cầu người chơi tư duy logic và sử dụng từ khóa.

**2.2.3. Thuật toán Mã hóa Tiêu chuẩn Tiên tiến (AES - Advanced Encryption Standard)**

**Nguyên lý cơ bản (Mã hóa khối, Đối xứng):** AES là một thuật toán mã hóa khối (block cipher) được chính phủ Hoa Kỳ chọn làm tiêu chuẩn mã hóa vào năm 2001, thay thế cho DES (Data Encryption Standard) đã lỗi thời. Nó là một thuật toán mật mã **đối xứng**, nghĩa là sử dụng **cùng một khóa** cho cả quá trình mã hóa và giải mã.

AES hoạt động trên các khối dữ liệu có kích thước cố định là 128 bit (16 byte). Khóa mã hóa có thể có độ dài 128 bit, 192 bit hoặc 256 bit. Độ dài khóa càng lớn thì độ an toàn càng cao. Quá trình mã hóa và giải mã của AES được thực hiện thông qua một chuỗi các vòng lặp (rounds), mỗi vòng gồm nhiều phép biến đổi khác nhau như thay thế (substitution), hoán vị (permutation), dịch chuyển hàng (row shifting) và trộn cột (column mixing), tất cả đều dựa trên các phép toán đại số hữu hạn.

Mặc dù nguyên lý bên trong của AES khá phức tạp với nhiều bước biến đổi toán học trên từng byte của khối dữ liệu, nhưng về mặt cơ bản, nó hoạt động như sau:

**Chuyển đổi dữ liệu:** Văn bản rõ được chia thành các khối 128 bit.

**Khởi tạo trạng thái:** Mỗi khối 128 bit được đưa vào một ma trận trạng thái (state matrix) 4x4 byte.

**Thêm khóa vòng đầu tiên (AddRoundKey):** Khóa mã hóa được XOR với ma trận trạng thái.

**Các vòng lặp chính:** Dữ liệu trải qua một số vòng lặp (10 vòng cho khóa 128 bit, 12 vòng cho 192 bit, 14 vòng cho 256 bit). Mỗi vòng lặp bao gồm các bước con:

**SubBytes:** Mỗi byte trong ma trận trạng thái được thay thế bằng một byte khác theo một bảng tra cứu (S-Box) cố định.

**ShiftRows:** Các hàng của ma trận trạng thái được dịch chuyển tuần hoàn.

**MixColumns:** Các cột của ma trận trạng thái được biến đổi bằng phép nhân ma trận trên trường hữu hạn Galois.

**AddRoundKey:** Khóa con (subkey) tương ứng với vòng lặp đó được XOR với ma trận trạng thái.

**Vòng lặp cuối cùng:** Tương tự các vòng lặp chính nhưng bỏ qua bước MixColumns.

**Kết quả:** Ma trận trạng thái cuối cùng chính là khối văn bản mã hóa.

Quá trình giải mã là đảo ngược chính xác các bước của quá trình mã hóa, sử dụng các phép toán nghịch đảo và các khóa con theo thứ tự ngược lại.

**Ứng dụng:** AES là một trong những thuật toán mã hóa phổ biến và đáng tin cậy nhất hiện nay, được ứng dụng rộng rãi trong hầu hết các lĩnh vực đòi hỏi bảo mật cao:

**Bảo mật dữ liệu trên đĩa cứng:** Mã hóa toàn bộ ổ đĩa (Full Disk Encryption) như BitLocker của Windows, FileVault của macOS.

**Bảo mật kết nối Internet:** Giao thức SSL/TLS (sử dụng trong HTTPS) để bảo vệ các giao dịch trực tuyến, email, và truyền tải dữ liệu web.

**Mạng riêng ảo (VPN):** Bảo vệ đường truyền dữ liệu qua mạng công cộng.

**Lưu trữ đám mây:** Mã hóa dữ liệu người dùng trước khi lưu trữ trên các dịch vụ đám mây.

**Thiết bị di động:** Bảo vệ dữ liệu trên smartphone, tablet.

**Chính phủ và Quân sự:** Được sử dụng để bảo vệ thông tin mật của chính phủ và quân đội trên toàn thế giới.

**Ví dụ đơn giản (trong ngữ cảnh game):** Trong game "Giải Mã Kho Báu", AES được mô phỏng trong màn chơi "Hầm Mộ Cổ Đại" và "Kho Báu Cuối Cùng". Người chơi không cần phải thực hiện từng bước phức tạp của AES mà thay vào đó, được yêu cầu cung cấp một khóa bí mật chính xác để giải mã một thông điệp đã được mã hóa. Điều này nhằm truyền tải thông điệp chính rằng AES là một thuật toán cực kỳ mạnh mẽ: nếu không có đúng khóa (một chuỗi ký tự dài, phức tạp như "TREASURE2024!" hay "FINAL\_TREASURE\_2024"), việc giải mã thông điệp là gần như không thể với tài nguyên tính toán hiện tại. Game tập trung vào việc nhấn mạnh tầm quan trọng của việc bảo vệ khóa bí mật và sức mạnh của AES trong việc đảm bảo tính bảo mật của dữ liệu nhạy cảm.

**2.2.4. Thuật toán RSA**

**Nguyên lý cơ bản (Khóa công khai/bí mật, Bất đối xứng):** RSA (Rivest–Shamir–Adleman) là một trong những thuật toán mật mã khóa công khai (public-key cryptography) đầu tiên và được sử dụng rộng rãi nhất. Nó được phát minh vào năm 1977 bởi Ron Rivest, Adi Shamir và Leonard Adleman. Khác với các thuật toán đối xứng sử dụng cùng một khóa cho cả mã hóa và giải mã, RSA sử dụng một **cặp khóa riêng biệt**: một **khóa công khai (public key)** để mã hóa và một **khóa bí mật (private key)** để giải mã.

Nguyên lý bảo mật của RSA dựa trên độ khó của bài toán phân tích thừa số nguyên lớn (factoring large numbers). Việc tạo khóa RSA liên quan đến việc chọn hai số nguyên tố lớn, ngẫu nhiên, sau đó tính toán mô-đun và các số mũ liên quan. Mặc dù việc tính toán sản phẩm của hai số nguyên tố lớn (tạo ra mô-đun) là dễ dàng, nhưng việc phân tích ngược lại mô-đun đó thành hai số nguyên tố ban đầu thì cực kỳ khó và tốn nhiều thời gian tính toán đối với các số đủ lớn.

Quá trình hoạt động của RSA diễn ra như sau:

**Tạo cặp khóa:**

Chọn hai số nguyên tố lớn khác nhau, gọi là p và q.

Tính mô-đun n=p×q.

Tính ϕ(n)=(p−1)×(q−1) (phi hàm Euler).

Chọn một số nguyên e (số mũ mã hóa) sao cho 1<e<ϕ(n) và e là số nguyên tố cùng nhau với ϕ(n). Đây là một phần của khóa công khai.

Tính số nguyên d (số mũ giải mã) sao cho d×e≡1(modϕ(n)). Đây là một phần của khóa bí mật.

**Khóa công khai:** (e,n) – được công khai cho bất kỳ ai muốn gửi tin nhắn đã mã hóa.

**Khóa bí mật:** (d,n) – phải được giữ bí mật bởi người nhận.

**Mã hóa:** Để mã hóa một văn bản rõ M (đã được chuyển đổi sang dạng số và M<n) thành văn bản mã hóa C, người gửi sử dụng khóa công khai (e,n) của người nhận: C=Me(modn)

**Giải mã:** Để giải mã văn bản mã hóa C trở lại văn bản rõ M, người nhận sử dụng khóa bí mật (d,n) của mình: M=Cd(modn)

Sức mạnh của RSA nằm ở chỗ: ai cũng có thể mã hóa bằng khóa công khai, nhưng chỉ người sở hữu khóa bí mật tương ứng mới có thể giải mã.

**Ứng dụng:** RSA đóng vai trò nền tảng cho nhiều ứng dụng bảo mật quan trọng trên Internet và trong thế giới số:

**Truyền khóa an toàn:** Thay vì mã hóa toàn bộ dữ liệu (vì RSA chậm), RSA thường được sử dụng để mã hóa và trao đổi khóa đối xứng (phiên khóa - session key) một cách an toàn giữa hai bên. Sau khi khóa đối xứng đã được trao đổi, hai bên sẽ sử dụng khóa đó với một thuật toán đối xứng (như AES) để mã hóa dữ liệu thực tế, tận dụng tốc độ cao của mật mã đối xứng.

**ký số (Digital Signatures):** Người gửi có thể ký vào một thông điệp (bằng cách mã hóa hàm băm của thông điệp bằng khóa bí mật của mình). Người nhận có thể xác minh chữ ký đó bằng khóa công khai của người gửi. Điều này đảm bảo tính xác thực của người gửi (chỉ người đó mới có khóa bí mật) và tính toàn vẹn của thông điệp (thông điệp không bị thay đổi sau khi ký).

**Chứng chỉ số (Digital Certificates):** Là một phần quan trọng của cơ sở hạ tầng khóa công khai (PKI), dùng để xác minh danh tính của website hoặc cá nhân trên Internet (ví dụ: giao thức HTTPS)

**Bảo mật Email:** Được sử dụng trong các giao thức mã hóa email như PGP/GPG.

**Ví dụ đơn giản (trong ngữ cảnh game):** Trong game "Giải Mã Kho Báu", RSA được thể hiện trong các màn chơi như "Két Sắt Bí Mật" và "Kho Máy Tính Cổ". Người chơi được cung cấp một thông điệp đã được mã hóa, và nhiệm vụ là phải tìm ra "khóa bí mật" để giải mã thông điệp đó. Mặc dù các phép tính toán học phức tạp của RSA không được hiển thị trực tiếp cho người chơi, game vẫn truyền tải nguyên lý cơ bản: đó là sự tồn tại của hai loại khóa (công khai và bí mật) và việc chỉ có khóa bí mật mới có thể mở khóa thông tin được bảo vệ bởi khóa công khai. Các gợi ý trong game (ví dụ: "Từ khóa liên quan đến khoa học mật mã học." hoặc "Từ khóa liên quan đến bí mật và an toàn.") đóng vai trò như những manh mối để người chơi suy luận ra từ khóa bí mật, tượng trưng cho việc quản lý và tìm kiếm đúng khóa trong thực tế.

**2.3. Quy trình ứng dụng thuật toán trong Mã hóa và Giải mã trên Game**

Phần này sẽ đi sâu vào cách các thuật toán mật mã được mô phỏng và ứng dụng trong game "Giải Mã Kho Báu", minh họa luồng tương tác giữa người chơi và hệ thống game trong quá trình mã hóa và giải mã thông điệp.

**2.3.1. Quy trình Chuẩn bị Dữ liệu Mã hóa (Từ góc độ của Game)**

Trong game "Giải Mã Kho Báu", quá trình mã hóa thông điệp không phải là một hành động trực tiếp của người chơi mà là một bước chuẩn bị dữ liệu bởi chính hệ thống game trước khi màn chơi bắt đầu. Quy trình này đảm bảo mỗi thử thách đều có sẵn một văn bản mã hóa và một khóa bí mật tương ứng, chờ đợi người chơi khám phá.

**Bước 1: Khởi tạo Dữ liệu Bài toán:**

Hệ thống game định nghĩa một tập hợp các bài toán (puzzles) cho từng màn chơi, được lưu trữ dưới dạng cấu trúc dữ liệu (ví dụ: danh sách các từ điển PUZZLES trong app.py).

Mỗi bài toán chứa các thông tin cần thiết: thuật toán sử dụng (algorithm), văn bản rõ (plaintext), và khóa bí mật (correct\_key) tương ứng.

**Bước 2: Thực hiện Mã hóa (Offline/Pre-game):**

Trước khi game được triển khai hoặc khi khởi tạo dữ liệu màn chơi, hệ thống sử dụng các hàm mã hóa tương ứng (Caesar, Vigenere, AES, RSA được tự cài đặt trong crypto\_utils.py) để mã hóa plaintext bằng correct\_key.

Kết quả của quá trình này là ciphertext (văn bản mã hóa) đã sẵn sàng.

Ví dụ: Đối với màn Caesar "Cuộn Cổ Bí Ẩn", plaintext "THAM TU SMITH DA TIM THAY KHO BAU THAT TU" được mã hóa bằng khóa "3" để tạo ra ciphertext "WKDP WX VPLWK GD WLP WKDR NKR EDX WKDW WX".

Đối với AES/RSA, plaintext có thể được mã hóa, sau đó kết quả được chuyển đổi sang định dạng Base64 để lưu trữ và hiển thị dễ dàng hơn trong tệp dữ liệu.

**Bước 3: Lưu trữ Dữ liệu Mã hóa:**

ciphertext được lưu trữ cùng với các thông tin khác của màn chơi (tiêu đề, mô tả, gợi ý) trong cấu trúc dữ liệu PUZZLES.

correct\_key cũng được lưu trữ (nhưng ẩn đối với người chơi) để phục vụ cho quá trình kiểm tra đáp án của người chơi.

Quy trình này đảm bảo rằng khi người chơi bắt đầu một màn chơi, họ sẽ luôn nhận được một thông điệp đã được mã hóa sẵn, đúng với thuật toán và khóa đã định.

**2.3.2. Quy trình Giải mã thông điệp (Từ góc độ của Người chơi và Game)**

Đây là quy trình tương tác chính mà người chơi trải nghiệm trong game "Giải Mã Kho Báu". Người chơi sẽ đóng vai trò chủ động trong việc tìm kiếm và áp dụng khóa để khôi phục thông tin gốc.

**Bước 1: Hiển thị Thử thách (Game hiển thị):**

Khi người chơi chọn một màn chơi (ví dụ: "Cuộn Cổ Bí Ẩn" - Caesar Cipher), giao diện game sẽ hiển thị ciphertext (văn bản mã hóa) tương ứng với màn chơi đó.

Đồng thời, game cung cấp các thông tin liên quan như tiêu đề, mô tả, cốt truyện của màn chơi và các gợi ý (hint\_1, hint\_2, hint\_3) để định hướng người chơi.

Giao diện cũng bao gồm một trường nhập liệu (input field) để người chơi nhập "khóa" mà họ suy đoán hoặc tìm được.

**Bước 2: Người chơi Phân tích và Nhập Khóa (Người chơi hành động):**

Người chơi đọc ciphertext và các gợi ý.

Dựa vào kiến thức về thuật toán mật mã (mà họ đã được học hoặc đang học thông qua game) và các gợi ý, người chơi sẽ phân tích và suy luận về khóa bí mật.

Người chơi nhập khóa mà họ tin là đúng vào trường nhập liệu được cung cấp trên giao diện.

**Bước 3: Gửi Yêu cầu Giải mã (Người chơi → Game):**

Khi người chơi nhấn nút "Giải mã" (hoặc "Gửi"), giá trị ciphertext của màn chơi hiện tại và key do người chơi nhập sẽ được gửi về máy chủ Flask thông qua một yêu cầu HTTP (ví dụ: POST request).

**Bước 4: Xử lý Giải mã và Kiểm tra Khóa (Game xử lý Backend):**

Máy chủ Flask nhận được yêu cầu.

Backend xác định thuật toán của màn chơi hiện tại (dựa trên id hoặc algorithm của puzzle).

Hệ thống gọi hàm giải mã tương ứng từ crypto\_utils.py (ví dụ: caesar\_decrypt, vigenere\_decrypt, aes\_decrypt, rsa\_decrypt), sử dụng ciphertext và key do người chơi cung cấp.

Kết quả giải mã được so sánh với plaintext gốc đã được lưu trữ sẵn trong dữ liệu PUZZLES của màn chơi đó.

**Bước 5: Phản hồi Kết quả (Game phản hồi):**

**Nếu khóa đúng:** Nếu plaintext từ quá trình giải mã khớp với plaintext gốc của bài toán, game sẽ hiển thị thông báo "Giải mã thành công!" hoặc một phần thưởng (ví dụ: mở khóa màn chơi tiếp theo, cộng điểm). Văn bản rõ (plaintext gốc) sẽ được hiển thị cho người chơi.

**Nếu khóa sai:** Game sẽ hiển thị thông báo "Khóa không đúng, hãy thử lại!" hoặc "Giải mã thất bại!", khuyến khích người chơi tiếp tục suy luận và thử các khóa khác

Các phản hồi này được gửi từ Flask backend về frontend dưới dạng dữ liệu (ví dụ: JSON hoặc HTML snippets) để cập nhật giao diện người dùng mà không cần tải lại toàn bộ trang.

Quy trình tương tác hai chiều này giúp người chơi không chỉ học về nguyên lý thuật toán mà còn trực tiếp trải nghiệm vai trò của khóa trong quá trình bảo mật thông tin.

**2.3.3. Quy trình Xác thực và Kiểm tra tính toàn vẹn (Nếu có)**

Trong phiên bản hiện tại của game "Giải Mã Kho Báu", trọng tâm chính là việc giải mã các thông điệp. Các quy trình liên quan đến **xác thực danh tính (authentication)** hoặc **kiểm tra tính toàn vẹn của dữ liệu (data integrity checking)** bằng cách sử dụng các hàm băm (ví dụ: SHA) hoặc chữ ký số **chưa được áp dụng trực tiếp như một phần gameplay tương tác** mà người chơi phải thực hiện.

Nếu game được mở rộng trong tương lai, một màn chơi về hàm băm có thể yêu cầu người chơi:

Được cung cấp một tài liệu và giá trị băm của nó.

Sau đó, thực hiện một thay đổi nhỏ trên tài liệu.

Chạy lại hàm băm và so sánh giá trị băm mới với giá trị cũ để thấy rằng dù chỉ thay đổi nhỏ cũng làm thay đổi hoàn toàn giá trị băm, qua đó minh họa nguyên lý của tính toàn vẹn dữ liệu.

Tuy nhiên, với các thuật toán hiện có (Caesar, Vigenere, AES, RSA), trọng tâm là bảo mật dữ liệu thông qua mã hóa/giải mã, chứ không phải xác thực hay kiểm tra toàn vẹn.

# Thiết kế và phân tích mã nguồn hệ thống game

## Thiết kế hệ thống game

**3.1.1. Yêu cầu chức năng và Phi chức năng của Game**

Việc xác định rõ các yêu cầu chức năng (những gì hệ thống phải làm) và phi chức năng (những thuộc tính của hệ thống) là bước nền tảng trong quá trình thiết kế, giúp định hình phạm vi và chất lượng của game "Giải Mã Kho Báu", đồng thời đảm bảo game đáp ứng mục tiêu giáo dục và trải nghiệm người dùng.

**Yêu cầu chức năng (Functional Requirements - FR):** Các yêu cầu chức năng mô tả các khả năng cụ thể mà hệ thống game phải cung cấp cho người dùng:

**FR1: Quản lý và Trình bày Màn chơi:** Hệ thống phải có khả năng lưu trữ và trình bày các màn chơi theo từng cấp độ, với mỗi màn chơi bao gồm nội dung câu đố (ciphertext), câu chuyện liên quan, hệ thống gợi ý, và độ khó riêng biệt.

**FR2: Hỗ trợ Đa dạng Thuật toán Mật mã:** Game phải triển khai và mô phỏng được nhiều thuật toán mã hóa và giải mã khác nhau, bao gồm nhưng không giới hạn ở Caesar, Vigenère, RSA, AES, XOR và Hash (SHA-256), cho phép người chơi thực hành các lý thuyết mật mã đã học.

**FR3: Tương tác Nhập liệu và Phản hồi:** Hệ thống phải cung cấp giao diện để người chơi nhập khóa giải mã hoặc kết quả suy luận. Sau khi nhập, game phải kiểm tra tính chính xác của đáp án và cung cấp phản hồi đúng/sai ngay lập tức.

**FR4: Cung cấp Hệ thống Gợi ý:** Mỗi màn chơi phải có một hệ thống gợi ý nhiều cấp độ (từ dễ đến khó) để hỗ trợ người chơi khi gặp khó khăn, giúp họ không bị bế tắc và khuyến khích tư duy.

**FR5: Chấm điểm và Ghi nhận Thành tích:** Mỗi màn chơi phải có một cơ chế chấm điểm riêng, cung cấp động lực và phản ánh nỗ lực của người chơi trong việc vượt qua các thử thách.

**Yêu cầu phi chức năng (Non-functional Requirements - NFR):** Các yêu cầu phi chức năng mô tả các thuộc tính chất lượng của hệ thống, góp phần vào trải nghiệm tổng thể của người dùng và tính bền vững của game:

**NFR1: Tính Dễ sử dụng và Trực quan (Usability & Intuitiveness):** Giao diện người dùng phải được thiết kế đơn giản, rõ ràng, dễ hiểu và dễ thao tác. Mọi tương tác (nhập liệu, gửi đáp án, nhận kết quả) phải trực quan, không yêu cầu người dùng có kiến thức kỹ thuật sâu về game.

**NFR2: Hiệu năng và Tốc độ Phản hồi (Performance & Responsiveness):** Các tác vụ như kiểm tra đáp án, hiển thị gợi ý, và chuyển đổi màn chơi phải diễn ra nhanh chóng, gần như tức thì, để đảm bảo trải nghiệm người dùng mượt mà và không bị gián đoạn.

**NFR3: Tính Bảo mật Tương đối (Relative Security):** Mặc dù là game giáo dục, hệ thống nên có các biện pháp nhất định để ẩn các đáp án (plaintext, correct\_key) khỏi người dùng cuối trên phía client (trình duyệt) nhằm ngăn chặn việc "soi code" trực tiếp và duy trì tính thử thách của game.

**NFR4: Khả năng Mở rộng (Extensibility):** Cấu trúc của game phải cho phép dễ dàng bổ sung thêm các màn chơi mới, các câu đố hoặc các thuật toán mật mã khác trong tương lai chỉ bằng cách cập nhật dữ liệu mà không yêu cầu sửa đổi nhiều trong mã nguồn backend.

**NFR5: Tính Giáo dục cao (High Educational Value):** Mỗi màn chơi không chỉ là một thử thách mà còn là một bài học. Game nên có khả năng giải thích bản chất của thuật toán hoặc lý do đằng sau đáp án sau khi người chơi hoàn thành, giúp củng cố kiến thức lý thuyết thông qua thực hành.

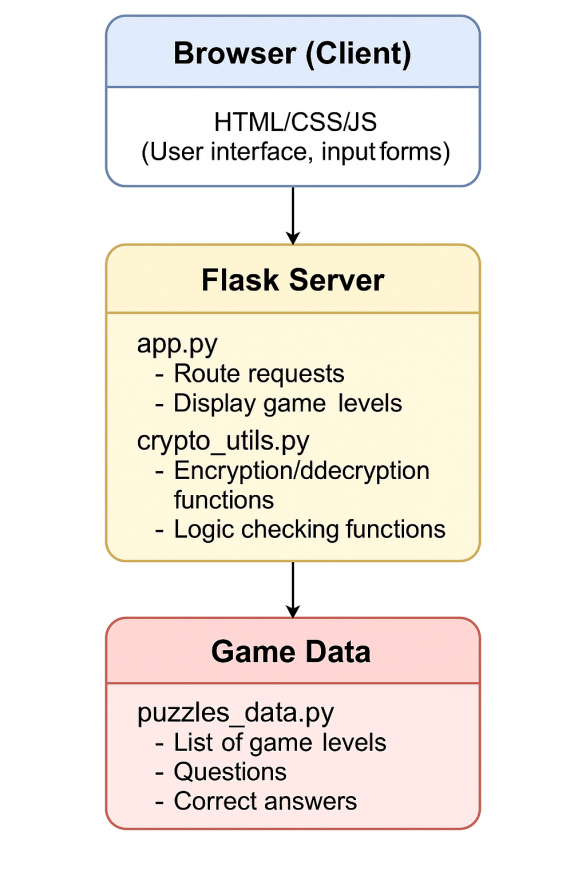
**NFR6: Tương thích Đa nền tảng (Cross-Platform Compatibility):** Game là một ứng dụng web, do đó cần đảm bảo hoạt động tốt và hiển thị chính xác trên các trình duyệt web phổ biến cũng như các thiết bị khác nhau (máy tính để bàn, máy tính bảng, điện thoại di động – responsive design).

**NFR7: Khả năng Lưu tiến trình (Optional - Future Enhancement):** Trong các phiên bản nâng cao, game có thể tích hợp hệ thống lưu tiến trình chơi (level đã vượt qua, điểm số) hoặc hỗ trợ đăng nhập người dùng để cá nhân hóa trải nghiệm học tập và theo dõi tiến độ.

**3.1.2. Kiến trúc tổng thể của Game (sơ đồ khối)**

Game "Giải Mã Kho Báu" được xây dựng theo kiến trúc **client-server đơn giản**, sử dụng **Flask (Python)** làm backend và **HTML/CSS/JavaScript thuần** cho frontend. Thiết kế này đảm bảo sự phân tách rõ ràng giữa các lớp, tạo điều kiện thuận lợi cho việc phát triển và bảo trì.

Dưới đây là sơ đồ khối tổng quát minh họa các thành phần chính và luồng tương tác của hệ thống:



· **Mô tả các Khối thành phần:**

**Trình duyệt người dùng (Client - Frontend)**

Đây là giao diện mà người chơi trực tiếp tương tác, được xây dựng bằng các công nghệ web cơ bản

**HTML:** Định nghĩa cấu trúc và nội dung của các trang web (màn hình game, form nhập liệu).

**CSS:** Định dạng phong cách và trình bày giao diện, đảm bảo tính thẩm mỹ và trực quan.

**JavaScript thuần:** Xử lý các tương tác phía client như nhận dữ liệu nhập từ người dùng, gửi các yêu cầu (HTTP requests) tới Flask Server (backend), và cập nhật nội dung trang động mà không cần tải lại toàn bộ trang.

Nhiệm vụ chính: Hiển thị các màn chơi, câu đố mã hóa, gợi ý; cung cấp form để người chơi nhập đáp án hoặc khóa giải mã; và nhận các phản hồi (đúng/sai, gợi ý mới) từ server để hiển thị cho người chơi.

**Flask Server (Backend):**

Là trái tim xử lý logic của game, được viết bằng ngôn ngữ **Python** và sử dụng **Flask micro-framework**.

Bao gồm hai thành phần chính:

**app.py:** Chứa logic điều hướng chính, quản lý các route (ví dụ: / cho trang chủ, /level/<id> để hiển thị một màn chơi cụ thể, /submit để xử lý đáp án). Module này chịu trách nhiệm tải dữ liệu màn chơi, hiển thị các câu đố tương ứng, và kiểm tra đáp án do người chơi gửi lên.

**crypto\_utils.py:** Một module riêng biệt chứa các hàm triển khai các thuật toán mã hóa và giải mã. Các hàm này được tự cài đặt (custom-implemented) cho các thuật toán như Caesar, Vigenère, RSA, AES (và các thuật toán khác như XOR, Hash nếu được tích hợp). Module này cung cấp các dịch vụ mã hóa/giải mã và kiểm tra logic nền tảng cho app.py.

**Dữ liệu Game (puzzles\_data.py):**

Là kho lưu trữ toàn bộ thông tin của các màn chơi/câu đố mật mã.

Dữ liệu được khai báo dưới dạng **danh sách các từ điển (list of dictionaries)** trực tiếp trong file puzzles\_data.py.

Mỗi từ điển đại diện cho một màn chơi và bao gồm các trường thông tin chi tiết như: id (mã định danh màn chơi), title (tiêu đề), cipher (tên thuật toán mã hóa), question (mô tả câu hỏi), ciphertext (văn bản đã mã hóa), key (khóa giải mã đúng), answer (văn bản rõ gốc), và hint (các gợi ý).

· **Đặc điểm kiến trúc:**

**Tách biệt rõ ràng:** Hệ thống được thiết kế với sự tách biệt rõ ràng giữa giao diện người dùng (Client), logic xử lý (Backend Flask), và lớp dữ liệu (Dữ liệu Game), tạo điều kiện cho việc phát triển và bảo trì độc lập từng thành phần.

**Dễ dàng mở rộng:** Việc thêm các màn chơi mới trở nên đơn giản, chỉ cần bổ sung dữ liệu dạng từ điển vào danh sách trong puzzles\_data.py mà không cần sửa đổi nhiều mã nguồn backend.

**Không phụ thuộc CSDL:** Game không sử dụng hệ quản trị cơ sở dữ liệu truyền thống, điều này giúp đơn giản hóa quá trình triển khai và quản lý, phù hợp cho một ứng dụng giáo dục nhỏ và nhẹ.  
**3.1.3. Thiết kế cơ sở dữ liệu (hoặc Cấu trúc Dữ liệu Game)**

Trong game "Giải Mã Kho Báu", việc quản lý thông tin các màn chơi không sử dụng hệ quản trị cơ sở dữ liệu (CSDL) truyền thống như MySQL, PostgreSQL hay SQLite. Thay vào đó, dữ liệu game được tổ chức và lưu trữ một cách đơn giản và trực tiếp trong mã nguồn Python, cụ thể là trong một file riêng biệt có tên **puzzles\_data.py**.

**Cấu trúc Dữ liệu PUZZLES:** Toàn bộ dữ liệu của các màn chơi được định nghĩa dưới dạng một **danh sách các từ điển (list of dictionaries)**. Mỗi từ điển trong danh sách đại diện cho một màn chơi (level) và chứa đầy đủ các thông tin cần thiết để game hoạt động.

Cấu trúc mẫu của một đối tượng màn chơi trong danh sách PUZZLES như sau:

PUZZLES = [

    # Caesar Cipher Levels

    {

        "id": 1,

        "level": 1,

        "algorithm": "Caesar",

        "title": "Cuộn Cổ Bí Ẩn",

        "description": "Thám tử Smith tìm thấy một cuộn giấy cổ với thông điệp được mã hóa bằng mật mã Caesar. Đây là manh mối đầu tiên dẫn đến kho báu.",

        "story": "Trong một căn phòng cổ kính, Smith phát hiện ra cuộn giấy bị ẩn sau bức tranh. Thông điệp được viết bằng mật mã Caesar cổ điển.",

        "ciphertext": "WKDP WX VPLWK GD WLP WKDR NKR EDX WKDW WX",

        "correct\_key": "3",

        "plaintext": "THAM TU SMITH DA TIM THAY KHO BAU THAT TU",

        "hint\_1": "Mật mã Caesar dịch chuyển các chữ cái trong bảng chữ cái.",

        "hint\_2": "Thử với số dịch chuyển nhỏ, từ 1 đến 10.",

        "hint\_3": "Khóa là số 3 - dịch chuyển mỗi chữ cái 3 vị trí.",

        "points": 100,

        "difficulty": "Dễ"

# các màn chơi khác cũng tương tụ như trên.

#....

    },]

**Giải thích các trường dữ liệu:**

id: Số nguyên duy nhất đại diện cho ID của màn chơi, dùng để truy cập và điều hướng.

title: Tiêu đề của màn chơi, hiển thị trên giao diện người dùng.

cipher: Tên thuật toán mật mã được sử dụng trong màn chơi này (ví dụ: 'Caesar', 'Vigenere', 'AES', 'RSA'). Trường này quan trọng để backend biết hàm giải mã nào cần gọi.

question: Đoạn văn bản mô tả câu hỏi hoặc nhiệm vụ của người chơi.

story: Câu chuyện ngắn gọn tạo bối cảnh cho màn chơi, tăng tính hấp dẫn.

ciphertext: Văn bản đã được mã hóa, là thông tin mà người chơi cần giải mã.

key: **Khóa giải mã đúng** cho màn chơi đó. Đây là thông tin bí mật mà game dùng để kiểm tra đáp án của người chơi.

answer: Văn bản rõ gốc sau khi giải mã đúng, được hiển thị khi người chơi hoàn thành màn chơi.

hint: Một danh sách các chuỗi ký tự, cung cấp các gợi ý khác nhau cho người chơi. Game có thể hiển thị gợi ý từng bước một.

· **Ưu điểm của việc lưu trữ dữ liệu này:**

**Đơn giản và dễ triển khai:** Không yêu cầu cài đặt và cấu hình hệ quản trị CSDL phức tạp. Phù hợp cho các dự án nhỏ, có quy mô dữ liệu cố định hoặc ít thay đổi.

**Nhanh chóng:** Việc truy cập dữ liệu trực tiếp từ bộ nhớ (sau khi file Python được tải) là rất nhanh.

**Dễ bảo trì và mở rộng:** Việc thêm một màn chơi mới chỉ đơn giản là thêm một từ điển mới vào danh sách PUZZLES. Việc chỉnh sửa nội dung màn chơi cũng trực tiếp trên file Python

· **Nhược điểm của việc lưu trữ dữ liệu này:**

**Không phù hợp với dữ liệu lớn/động:** Nếu số lượng màn chơi rất lớn, hoặc dữ liệu cần được cập nhật thường xuyên bởi nhiều người dùng/quản trị viên, thì phương pháp này sẽ trở nên khó quản lý và không hiệu quả.

**Tính bảo mật hạn chế:** Mặc dù key và answer không hiển thị trực tiếp trên frontend, nhưng chúng vẫn nằm trong mã nguồn backend. Điều này có thể không lý tưởng cho các ứng dụng yêu cầu bảo mật cao tuyệt đối, mặc dù đối với một game giáo dục thì điều này là chấp nhận được.

**Khó khăn trong việc tìm kiếm/truy vấn phức tạp:** Không có các công cụ truy vấn mạnh mẽ như SQL để lọc hay sắp xếp dữ liệu phức tạp.

**Không có tính năng lưu trữ trạng thái người dùng:** Phương pháp này không hỗ trợ lưu trữ tiến trình chơi, điểm số cá nhân, hoặc thông tin người dùng theo cách truyền thống của CSDL.

· **Cách game truy cập và quản lý dữ liệu màn chơi:** Khi server Flask khởi động, file puzzles\_data.py được import, và danh sách PUZZLES được tải vào bộ nhớ. Khi người chơi yêu cầu một màn chơi cụ thể (ví dụ: qua ID), Flask backend sẽ duyệt qua danh sách PUZZLES để tìm từ điển tương ứng và trích xuất thông tin cần thiết (ciphertext, question, hint, v.v.) để gửi về frontend. Tương tự, khi người chơi gửi đáp án, backend cũng sử dụng id để tìm đúng puzzle và lấy key cùng answer đã lưu sẵn để so sánh.

## Môi trường triển khai và Cônbg cụ phát triển

Phần này sẽ trình bày chi tiết các công nghệ, ngôn ngữ lập trình, framework, thư viện và công cụ đã được sử dụng trong quá trình phát triển game "Giải Mã Kho Báu", cũng như môi trường triển khai của ứng dụng.

**3.2.1. Ngôn ngữ lập trình, Framework, Thư viện chi tiết**

**Ngôn ngữ lập trình chính:**

Dự án sử dụng **Python** làm ngôn ngữ lập trình chính cho toàn bộ logic backend. Python được lựa chọn nhờ cú pháp rõ ràng, tính linh hoạt và hiệu quả trong việc xử lý các tác vụ logic, rất phù hợp cho một ứng dụng web nhỏ và mang tính giáo dục.

**Framework Backend:**

**Flask:** Là một micro-framework web nhẹ của Python, được sử dụng để xây dựng backend của game. Flask cung cấp các công cụ cần thiết để định tuyến URL (routing), xử lý yêu cầu HTTP (request handling), và quản lý các template HTML, giúp việc phát triển nhanh chóng và đơn giản.

**Công nghệ Frontend:**

**HTML5:** Được sử dụng để xây dựng cấu trúc và nội dung các trang web, tạo ra các thành phần giao diện như tiêu đề, đoạn văn bản câu chuyện, trường nhập liệu và nút bấm.

**CSS3:** Áp dụng để định dạng phong cách, màu sắc, bố cục và tối ưu hóa trải nghiệm hiển thị cho người dùng, đảm bảo giao diện trực quan và dễ nhìn.

**JavaScript (thuần):** Xử lý các tương tác phía client, bao gồm việc gửi dữ liệu từ form nhập liệu lên server (sử dụng Fetch API hoặc XMLHttpRequest) và cập nhật giao diện người dùng dựa trên phản hồi từ server mà không cần tải lại toàn bộ trang.

**Thư viện Mật mã học (Tự cài đặt):**

Một điểm đặc biệt của dự án là việc **không sử dụng các thư viện mật mã học bên ngoài** có sẵn (như PyCryptodome hay cryptography cho Python). Thay vào đó, tất cả các thuật toán mã hóa và giải mã (Caesar, Vigenère, AES, RSA) đều được **tự cài đặt (self-implemented)** trong file crypto\_utils.py.

Việc tự cài đặt này có mục đích giáo dục, giúp người học và người phát triển hiểu rõ hơn về các nguyên lý cơ bản và từng bước hoạt động của mỗi thuật toán, thay vì chỉ sử dụng các hàm API có sẵn.

Đối với AES và RSA, các hàm tự cài đặt này tập trung vào việc mô phỏng nguyên lý cốt lõi và tương tác với người dùng qua khóa, chứ không nhằm mục đích đạt đến mức độ bảo mật chuẩn công nghiệp của các thư viện chuyên dụng

**3.2.2. Các công cụ phát triển khác (IDE, Version Control System...)**

**Môi trường Phát triển Tích hợp (IDE):**

Dự án được phát triển và kiểm thử chủ yếu trong môi trường **Replit**. Replit là một IDE trực tuyến mạnh mẽ, hỗ trợ nhiều ngôn ngữ lập trình và cung cấp môi trường chạy code tức thì, rất phù hợp cho việc phát triển và cộng tác nhanh chóng.

Ngoài ra, mã nguồn được viết theo chuẩn Python/Flask thông thường, cho phép dự án có thể dễ dàng được phát triển và gỡ lỗi trên các IDE phổ biến khác dành cho Python như **Visual Studio Code** hay **PyCharm**.

**Hệ thống Quản lý Phiên bản (Version Control System):**

**Git** được sử dụng làm hệ thống quản lý phiên bản. Sự hiện diện của thư mục .git/ trong cấu trúc dự án xác nhận việc theo dõi các thay đổi trong mã nguồn, quản lý các phiên bản khác nhau và hỗ trợ quy trình phát triển có tổ chức.

Mặc dù chi tiết về nền tảng lưu trữ Git (như GitHub, GitLab) không được đề cập, việc sử dụng Git đảm bảo khả năng quản lý lịch sử phát triển và tiềm năng cộng tác.

**Hệ điều hành Phát triển:**

Quá trình phát triển game có thể diễn ra trên nhiều hệ điều hành phổ biến như **Windows, macOS hoặc Linux**, nhờ tính đa nền tảng của Python và Flask, cũng như môi trường đám mây của Replit.  
**3.3. Phân tích mã nguồn**

Phần này sẽ đi sâu vào từng lớp, từng hàm, từng dòng lệnh và từng tương tác giữa các thành phần của ứng dụng "Giải Mã Kho Báu", làm rõ cách các nguyên lý mật mã và lập trình được áp dụng trong cả backend và frontend.

## 3.3.1. Cấu trúc dự án và Tổ chức mã nguồn

Dự án được cấu trúc theo mô hình phân lớp rõ ràng, giúp phân tách các trách nhiệm, tăng tính module hóa và dễ dàng bảo trì. Đây là cấu trúc điển hình cho một ứng dụng web nhỏ sử dụng Flask.

.

├── app.py # Backend chính: Định nghĩa ứng dụng Flask, Routes API, và Dữ liệu Game

├── crypto\_utils.py # Backend: Chứa các Implementations của thuật toán mật mã

├── main.py # Điểm khởi chạy của ứng dụng Flask

├── static/ # Chứa các tài nguyên tĩnh được phục vụ cho Frontend

│ ├── css/

│ │ └── style.css # CSS tùy chỉnh để định kiểu giao diện

│ └── js/

│ └── game.js # Logic tương tác chính của Frontend (JavaScript)

└── templates/

└── index.html # Giao diện người dùng chính (Frontend HTML)

**Phân tích chi tiết vai trò và tổ chức mã nguồn của từng thành phần:**

**app.py**

**Vai trò cốt lõi**: Đây là **Application Core** của backend. Nó định nghĩa đối tượng ứng dụng Flask và thiết lập các API endpoint (routes) mà frontend sẽ giao tiếp. app.py đóng vai trò là bộ điều khiển (controller) chính, nhận yêu cầu từ client, điều hướng đến các logic xử lý thích hợp và gửi phản hồi.

**Cấu hình**:

from flask import Flask, render\_template, request, jsonify, session: Import các module cần thiết từ Flask để xây dựng ứng dụng web, render HTML, xử lý yêu cầu, trả về JSON và quản lý phiên.

import os: Sử dụng để truy cập biến môi trường, đặc biệt là SESSION\_SECRET cho bảo mật session. Điều này thể hiện một thực hành tốt về bảo mật, tách khóa bí mật khỏi mã nguồn.

from datetime import timedelta: Để cấu hình thời gian sống của session.

from crypto\_utils import CryptoUtils: Import lớp chứa các thuật toán mật mã. Điều này thể hiện nguyên tắc phân tách trách nhiệm (Separation of Concerns): logic mật mã được đặt riêng biệt, giúp app.py tập trung vào vai trò điều phối.

app.config['SECRET\_KEY'] = os.environ.get('SESSION\_SECRET', 'your\_super\_secret\_key'): Thiết lập khóa bí mật cho session. Đây là **yếu tố bảo mật quan trọng** để mã hóa cookie session, ngăn chặn việc giả mạo hoặc đọc trộm dữ liệu phiên. Việc lấy từ biến môi trường (os.environ.get) là một best practice trong triển khai ứng dụng.

app.config['PERMANENT\_SESSION\_LIFETIME'] = timedelta(minutes=30): Cấu hình session sẽ tồn tại trong 30 phút sau lần tương tác cuối cùng của người dùng.

**Dữ liệu Game (PUZZLES)**:

Là một danh sách các từ điển Python, mỗi từ điển đại diện cho một màn chơi. Việc đặt dữ liệu này trực tiếp trong app.py làm cho ứng dụng trở nên tự chứa và đơn giản cho một quy mô nhỏ.

Mỗi từ điển puzzle bao gồm các trường: id, title, description, story, difficulty, algorithm (Caesar, Vigenere, RSA, AES), ciphertext, correct\_key, plaintext, points, và ba hint. Việc định nghĩa đầy đủ các trường này giúp frontend dễ dàng hiển thị thông tin và backend dễ dàng kiểm tra đáp án.

**Định tuyến (Routes API)**: app.py định nghĩa các tuyến API chính bằng decorator @app.route(), mỗi tuyến tương ứng với một hành động của người dùng hoặc một yêu cầu dữ liệu từ frontend:

/ (GET): Trang chủ, render index.html.

/api/levels (GET): Cung cấp dữ liệu tất cả các màn chơi dưới dạng JSON.

/api/player/create (POST): Đăng ký người chơi mới, lưu thông tin vào session.

/api/attempt (POST): Nhận và kiểm tra đáp án giải mã của người chơi.

Việc sử dụng jsonify() cho các API endpoint đảm bảo dữ liệu được trả về ở định dạng JSON chuẩn, dễ dàng được JavaScript ở frontend xử lý.

**crypto\_utils.py**

**Vai trò cốt lõi**: Là một **Module tiện ích độc lập** chứa các logic mật mã học. Mục tiêu là phân tách các thuật toán khỏi logic ứng dụng chính trong app.py, giúp mã nguồn dễ đọc hơn, dễ kiểm thử hơn và có thể tái sử dụng.

**Tổ chức mã nguồn**:

Định nghĩa một lớp CryptoUtils. Việc sử dụng lớp thay vì các hàm độc lập giúp nhóm các hàm liên quan đến mật mã lại với nhau.

Tất cả các phương thức trong lớp này đều là phương thức tĩnh (@staticmethod). Điều này có nghĩa là bạn có thể gọi chúng trực tiếp từ lớp (ví dụ: CryptoUtils.caesar\_decrypt(...)) mà không cần phải tạo một thể hiện (instance) của lớp CryptoUtils. Điều này phù hợp vì các hàm này không cần truy cập vào trạng thái của một đối tượng cụ thể.

Các hàm giải mã chính: caesar\_decrypt, vigenere\_decrypt, simple\_base64\_decrypt (dùng để giả lập RSA/AES).

Các hàm kiểm tra đáp án: check\_caesar, check\_vigenere, check\_rsa, check\_aes. Các hàm này là giao diện mà app.py sử dụng để tương tác với logic giải mã và kiểm tra khóa.

Sử dụng thư viện chuẩn của Python như base64 và string (mặc dù string không được gọi trực tiếp trong các hàm mã hóa chính nhưng là công cụ phổ biến cho các tác vụ này).

**main.py**

**Vai trò cốt lõi**: Là **Điểm vào (Entry Point)** của ứng dụng. Nhiệm vụ duy nhất của nó là khởi chạy server Flask.

**Tổ chức mã nguồn**: Rất đơn giản.

from app import app: Import đối tượng ứng dụng Flask đã được định nghĩa trong app.py.

if \_\_name\_\_ == '\_\_main\_\_': app.run(debug=True): Đây là một khối tiêu chuẩn trong Python. Nó đảm bảo rằng app.run() (lệnh khởi chạy server Flask) chỉ được gọi khi file main.py được thực thi trực tiếp, chứ không phải khi nó được import như một module bởi các file khác. debug=True bật chế độ gỡ lỗi của Flask, giúp phát hiện lỗi dễ dàng hơn trong quá trình phát triển (ví dụ: tự động tải lại server khi có thay đổi mã nguồn).

**static/**

**Vai trò cốt lõi**: Chứa các **Tài nguyên tĩnh** mà web server sẽ phục vụ trực tiếp cho trình duyệt của client mà không cần xử lý thêm bởi Flask. Điều này bao gồm CSS, JavaScript, hình ảnh, v.v.

**Cấu trúc**:

css/: Chứa các file CSS.

style.css: File CSS tùy chỉnh của dự án. Nó định nghĩa toàn bộ phong cách trực quan của ứng dụng: bố cục, màu sắc, font chữ, hiệu ứng động (ví dụ: màn hình tải, hiệu ứng card, hiệu ứng toast, hiệu ứng lắc input), và các quy tắc responsive để đảm bảo giao diện hiển thị tốt trên các kích thước màn hình khác nhau. Nó làm việc song song với Bootstrap để tùy chỉnh giao diện.

js/: Chứa các file JavaScript.

game.js: Đây là **Frontend Application Logic** chính. Nó triển khai toàn bộ logic tương tác phía client: quản lý trạng thái game (người chơi hiện tại, level, điểm), xử lý sự kiện người dùng (nhấn nút, nhập form), giao tiếp bất đồng bộ với backend API (fetch API), cập nhật Document Object Model (DOM) để hiển thị thông tin và phản hồi lại người dùng. Toàn bộ logic này được gói gọn trong một lớp CryptographyGame để quản lý tốt trạng thái và hành vi.

**templates/**

**Vai trò cốt lõi**: Chứa các **Web Templates** (tệp HTML) mà Flask sử dụng để xây dựng các trang web và gửi chúng đến trình duyệt.

**Cấu trúc**:

index.html: Đây là **Single Page Application (SPA) container**. Nó là file HTML duy nhất được Flask trả về. index.html định nghĩa cấu trúc xương sống cho toàn bộ giao diện người dùng, bao gồm:

Các phần tử HTML chính với id rõ ràng để JavaScript dễ dàng thao tác (ví dụ: #loadingScreen, #home, #game, #leaderboard, #successModal, #hintModal).

Liên kết đến các thư viện CSS/JS bên ngoài (Bootstrap, Font Awesome, Google Fonts) và các file CSS/JS nội bộ (style.css, game.js).

Đây là nơi các thành phần UI được định nghĩa, nhưng nội dung động và logic hiển thị được quản lý bởi game.js.

**Tóm tắt tổ chức mã nguồn:**

**Backend (Python)**:

app.py: Điều phối, API, quản lý dữ liệu game.

crypto\_utils.py: Logic mật mã thuần túy.

main.py: Điểm khởi chạy server.

**Frontend (HTML, CSS, JavaScript)**:

index.html: Cấu trúc và xương sống giao diện.

style.css: Tạo phong cách và hiệu ứng hình ảnh.

game.js: Xử lý tương tác, logic game client-side, giao tiếp API.

Sự phân tách này tuân thủ tốt nguyên tắc kiến trúc phần mềm, cho phép các nhà phát triển tập trung vào từng phần cụ thể mà không ảnh hưởng lớn đến các phần khác, đồng thời nâng cao khả năng mở rộng và sửa lỗi.  
**3.3.2. Triển khai các thuật toán mã hóa/giải mã trong mã nguồn**

Phần này sẽ phân tích chi tiết từng hàm mã hóa/giải mã được triển khai trong file crypto\_utils.py, làm rõ từng bước logic, cách thức các phép toán được thực hiện, và vai trò của các thư viện chuẩn Python.

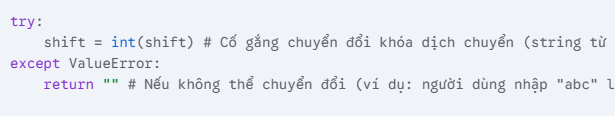
### a. Giải thích chi tiết hàm/module cho Caesar, Vigenere, AES, RSA

Trong dự án này, các thuật toán mật mã cổ điển được triển khai trực tiếp, trong khi các thuật toán hiện đại hơn như RSA và AES được "giả lập" bằng Base64 nhằm mục đích giáo dục và đơn giản hóa game.

#### **1. Mật mã Caesar (caesar\_decrypt)**

**Định nghĩa trong code**: Phương thức tĩnh caesar\_decrypt(ciphertext, shift) trong lớp CryptoUtils.

**Mục đích**: Giải mã một chuỗi ciphertext đã được mã hóa bằng mật mã Caesar, sử dụng một shift (khóa dịch chuyển) nhất định.

**Chi tiết triển khai và luồng logic**:  


Đây là một bước kiểm tra đầu vào quan trọng để đảm bảo tính hợp lệ của khóa, ngăn chặn lỗi runtime.

Khởi tạo kết quả: result = "" khởi tạo một chuỗi rỗng để tích lũy các ký tự đã giải mã.  


· Vòng lặp này duyệt qua từng ký tự char trong ciphertext đầu vào.

Kiểm tra loại ký tự (chữ cái hay không):



Chỉ các ký tự chữ cái (a-z, A-Z) mới được xử lý bằng thuật toán Caesar. Các ký tự khác (số, dấu câu, khoảng trắng, ký tự đặc biệt) sẽ được giữ nguyên.  


char.isupper() kiểm tra xem ký tự hiện tại có phải là chữ hoa không.

ord('A') trả về giá trị ASCII của ký tự 'A' (65).

ord('a') trả về giá trị ASCII của ký tự 'a' (97).

start\_char được thiết lập làm điểm tham chiếu để tính toán vị trí tương đối của ký tự trong bảng chữ cái (0-25) cho cả chữ hoa và chữ thường riêng biệt. Điều này đảm bảo rằng việc giải mã diễn ra độc lập cho từng trường hợp.  
**Tính toán vị trí đã giải mã (Phép toán lõi)**:  
  
Đây là công thức toán học để giải mã Caesar:

ord(char) - start\_char: Chuyển đổi ký tự char thành một số từ 0 đến 25, đại diện cho vị trí của nó trong bảng chữ cái (ví dụ: 'C' (67) với start\_char='A'(65) sẽ là 2).

- shift: Thực hiện phép dịch chuyển ngược. Nếu shift là 3, ký tự ở vị trí 2 sẽ dịch chuyển về vị trí -1.

% 26: **Phép toán modulo 26** là cực kỳ quan trọng. Nó đảm bảo rằng kết quả luôn nằm trong phạm vi từ 0 đến 25. Nếu kết quả sau khi dịch chuyển là âm (ví dụ: -1), Python sẽ xử lý (-1 % 26) thành 25, điều này chính xác cho việc dịch chuyển vòng quanh từ đầu bảng chữ cái (ví dụ: 'A' dịch chuyển ngược 1 sẽ thành 'Z'). Tương tự, nếu dịch chuyển vượt quá 25, nó sẽ quay lại từ đầu

#### start\_char: Chuyển đổi giá trị số đã được xử lý (0-25) trở lại giá trị ASCII của ký tự chữ cái tương ứng. **Chuyển đổi thành ký tự và tích lũy**: **Xử lý ký tự không phải chữ cái**: Trả về kết quả: Sau khi vòng lặp kết thúc, result chứa toàn bộ văn bản đã

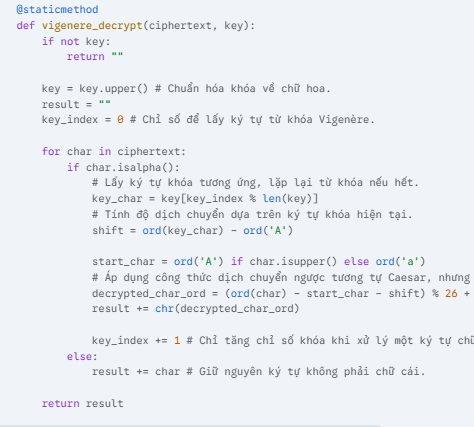
#### **Mật mã Vigenère (vigenere\_decrypt)**

**Định nghĩa trong code**: Phương thức tĩnh vigenere\_decrypt(ciphertext, key) trong lớp CryptoUtils.

**Mục đích**: Giải mã một chuỗi ciphertext đã được mã hóa bằng mật mã Vigenère, sử dụng một key (từ khóa) cụ thể.

**Chi tiết triển khai và luồng logic**:

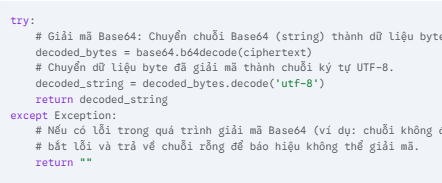
**Kiểm tra khóa rỗng**:  
  
**Chuẩn hóa từ khóa:**  
**Khởi tạo kết quả và chỉ số khóa:**  
**Duyệt qua từng ký tự:**  
**Kiểm tra loại ký tự (chữ cái hay không):**  
Tương tự Caesar, chỉ các ký tự chữ cái mới được xử lý mật mã.  
**Xác định ký tự khóa và độ dịch chuyển:**  
· key\_index % len(key): Đây là cơ chế lặp lại từ khóa. Ví dụ, nếu từ khóa là "KEY" (dài 3), key\_index sẽ là 0, 1, 2, 0, 1, 2,... khi duyệt qua văn bản.

· · ord(key\_char) - ord('A'): Chuyển đổi ký tự khóa (ví dụ: 'E' -> 4) thành độ dịch chuyển tương ứng.  
**Xác định điểm gốc bảng chữ cái (tương tự Caesar)**  
**Tính toán vị trí đã giải mã (Phép toán lõi):**  
Công thức này giống hệt Caesar, nhưng điểm khác biệt quan trọng là giá trị shift không cố định mà thay đổi tuần hoàn theo từng ký tự của key\_char.  
**Chuyển đổi thành ký tự và tích lũy:**  
**Tăng chỉ số khóa:**  
Đây là điều quan trọng: các ký tự không phải chữ cái không làm thay đổi vị trí của key\_index trong từ khóa, đảm bảo từ khóa chỉ dịch chuyển khi gặp một ký tự cần mã hóa/giải mã.  
**Xử lý ký tự không phải chữ cái:**  
**Trả về kết quả: result chứa văn bản đã giải mã.**

3. RSA và AES (Giả lập bằng simple\_base64\_decrypt)

**Định nghĩa trong code**: Phương thức tĩnh simple\_base64\_decrypt(ciphertext, key) trong lớp CryptoUtils.

**Mục đích**: Hàm này không triển khai RSA hay AES thực tế. Thay vào đó, nó sử dụng giải mã Base64 để **giả lập** một bước giải mã cơ bản. Mục tiêu chính của game ở các màn này là kiểm tra xem người chơi có nhập **đúng "khóa" đã được định nghĩa sẵn** trong dữ liệu game hay không, chứ không phải là giải mã RSA/AES thực sự.

**Chi tiết triển khai và luồng logic:**

base64.b64decode(ciphertext): Đây là hàm cốt lõi từ thư viện base64 của Python. Nó nhận một chuỗi đã được mã hóa Base64 và trả về dữ liệu gốc dưới dạng bytes.

.decode('utf-8'): Sau khi có bytes, cần phải giải mã chúng thành một chuỗi ký tự bằng cách chỉ định bảng mã (encoding). utf-8 là bảng mã phổ biến và linh hoạt.

**Khóa key (tham số không dùng)**: Mặc dù hàm này có tham số key, nhưng nó không được sử dụng trong logic giải mã Base64. Điều này là do Base64 không phải là thuật toán mã hóa (nó chỉ là một cách biểu diễn dữ liệu) và không cần khóa để giải mã. Tham số key được giữ lại để đảm bảo tính nhất quán về giao diện hàm (interface) với các hàm giải mã Caesar và Vigenère.

· **Cách thức sử dụng trong Game**:

Trong app.py, khi người chơi gửi đáp án cho một màn RSA hoặc AES, backend gọi CryptoUtils.check\_rsa hoặc CryptoUtils.check\_aes.

Các hàm check\_ này sẽ **chỉ kiểm tra xem attempted\_key của người chơi có khớp với correct\_key đã được lưu trữ trong PUZZLES hay không**.

Nếu khớp, app.py sẽ trực tiếp lấy plaintext đã định nghĩa sẵn trong PUZZLES của màn chơi đó và gửi về cho frontend, thay vì sử dụng kết quả từ simple\_base64\_decrypt. Điều này tạo ra trải nghiệm "giải mã thành công" cho người chơi khi họ nhập đúng "khóa", mà không yêu cầu triển khai phức tạp của RSA/AES.  
  
b. Cách thức sử dụng thư viện (nếu có) cho các thuật toán phức tạp

Dự án này sử dụng tối thiểu các thư viện bên ngoài để tập trung vào việc tự triển khai các thuật toán cổ điển và đơn giản hóa các thuật toán hiện đại.

**base64 (Thư viện chuẩn của Python)**:

**Mục đích**: Module base64 là một phần của thư viện chuẩn Python, cung cấp các hàm để mã hóa và giải mã dữ liệu sử dụng mã hóa Base64, Base32, Base16, v.v.

**Sử dụng**: Cụ thể trong dự án này, base64.b64decode() được sử dụng trong hàm simple\_base64\_decrypt. Hàm này nhận một chuỗi Base64 và trả về một đối tượng bytes. Sau đó, phương thức .decode('utf-8') được gọi trên đối tượng bytes để chuyển đổi nó thành một chuỗi ký tự Python.

**Vai trò trong game**: Việc sử dụng Base64 giúp "mô phỏng" một bước giải mã cho các thuật toán phức tạp hơn như RSA và AES mà không cần phải cài đặt và học các thư viện mật mã học phức tạp (như pycryptodome hoặc cryptography). Điều này giữ cho dự án gọn nhẹ, dễ hiểu và phù hợp với mục tiêu giáo dục cơ bản về mật mã.

**string (Thư viện chuẩn của Python)**:

**Mục đích**: Module string cung cấp các hằng số hữu ích như chuỗi các chữ cái hoa, chữ thường, số, v.v. (ví dụ: string.ascii\_uppercase, string.digits).

**Sử dụng trong code**: Mặc dù không được gọi trực tiếp bằng string.ascii\_uppercase trong các hàm caesar\_decrypt hay vigenere\_decrypt (thay vào đó là sử dụng ord('A') và ord('a')), các hàm này vẫn hoạt động dựa trên nguyên tắc của bảng chữ cái Latinh. Việc sử dụng ord() và chr() kết hợp với phép toán modulo là cách tự triển khai các phép dịch chuyển ký tự dựa trên vị trí số học của chúng trong bảng mã ASCII, tương tự như việc sử dụng các hằng số từ module string để xác định bảng chữ cái.

**3.3.3. Triển khai các thuật toán xác thực/kiểm tra tính toàn vẹn** Trong bối cảnh của ứng dụng "Giải Mã Kho Báu" này, các thuật toán xác thực (ví dụ: dùng mật khẩu, chữ ký số) hoặc kiểm tra tính toàn vẹn dữ liệu (ví dụ: hàm băm SHA) theo nghĩa mật mã học truyền thống không được triển khai một cách độc lập hoặc phức tạp. Thay vào đó, "xác thực" ở đây chủ yếu tập trung vào việc **xác nhận tính đúng đắn của khóa giải mã do người chơi cung cấp** cho từng màn chơi.

### a. Giải thích chi tiết hàm/module cho SHA, DES, RSA (có thể kèm đoạn mã minh họa và chú thích)

Các thuật toán như SHA (Secure Hash Algorithm - dùng cho hàm băm), DES (Data Encryption Standard - một thuật toán mã hóa khối đối xứng đã cũ), hoặc RSA (có thể dùng cho chữ ký số) không được triển khai trực tiếp hoặc sử dụng từ thư viện trong dự án này để phục vụ mục đích xác thực hay kiểm tra tính toàn vẹn.

Thay vào đó, logic "xác thực" trong game được tích hợp trực tiếp vào các hàm kiểm tra đáp án của từng màn chơi, tập trung vào việc **so sánh khóa** và quản lý phiên người chơi một cách đơn giản.

#### **1. Cơ chế "Xác thực" Khóa Giải mã (check\_caesar, check\_vigenere, check\_rsa, check\_aes)**

**Định nghĩa trong code**: Các phương thức tĩnh check\_caesar, check\_vigenere, check\_rsa, check\_aes trong lớp CryptoUtils.

**Mục đích**: Các hàm này đóng vai trò là "cổng kiểm soát". Chúng nhận khóa giải mã (attempted\_key) do người chơi nhập từ frontend và so sánh nó với khóa giải mã chính xác (correct\_key) được lưu trữ trong dữ liệu của màn chơi trên backend (PUZZLES trong app.py). Nếu khóa khớp, game coi đó là một "xác thực" thành công và cho phép người chơi tiến bộ.

**Tham số chung**: Mỗi hàm check\_ đều nhận ciphertext, attempted\_key, và correct\_key.

**Luồng logic chung**:

**So sánh khóa**: Đây là bước "xác thực" chính. Các hàm này sẽ thực hiện một phép so sánh chuỗi hoặc số giữa attempted\_key và correct\_key.

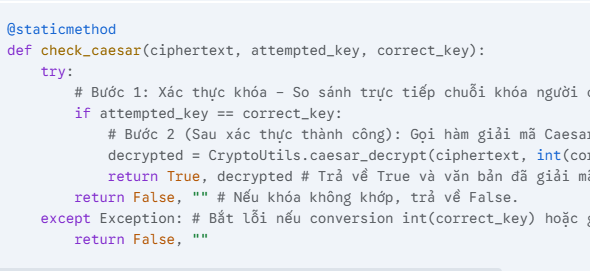
**Xử lý trường hợp đặc biệt**: Một số hàm có thể có các bước tiền xử lý nhỏ (ví dụ: chuyển đổi sang chữ hoa) để đảm bảo phép so sánh không phân biệt chữ hoa/thường nếu thuật toán yêu cầu.

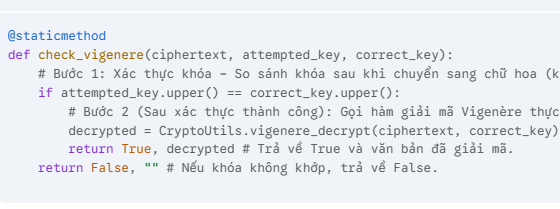
**Hành động dựa trên kết quả xác thực**:

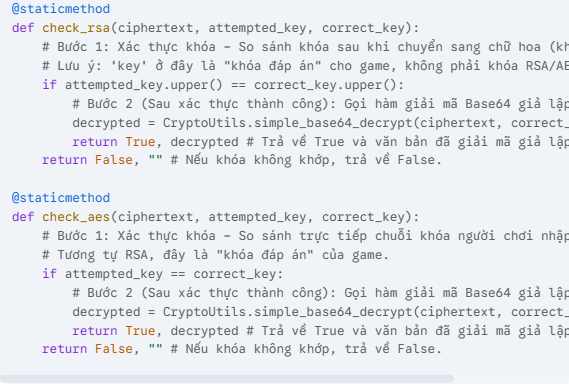
Nếu khóa khớp (xác thực thành công): Hàm sẽ gọi hàm giải mã tương ứng (caesar\_decrypt, vigenere\_decrypt, simple\_base64\_decrypt) để lấy plaintext. Quan trọng hơn, nó sẽ trả về (True, decrypted\_text). Trong trường hợp của RSA/AES, app.py sẽ sử dụng plaintext đã lưu trữ, không phải kết quả từ simple\_base64\_decrypt.

Nếu khóa không khớp (xác thực thất bại): Hàm sẽ trả về (False, "").

**Xử lý lỗi**: Các khối try-except được sử dụng để bắt các lỗi có thể xảy ra trong quá trình chuyển đổi kiểu dữ liệu (ví dụ: khóa Caesar không phải số) hoặc giải mã.

**Chi tiết triển khai và luồng logic cho từng hàm kiểm tra**:  


**Điểm nổi bật**: Phép so sánh khóa trực tiếp (attempted\_key == correct\_key) là cơ chế xác thực duy nhất ở đây. Khóa Caesar là một số, nên int(correct\_key) được dùng khi gọi hàm giải mã.  
**check\_vigenere(ciphertext, attempted\_key, correct\_key)**:  


**Điểm nổi bật**: upper() được sử dụng trên cả hai khóa trước khi so sánh. Điều này cho phép người chơi nhập khóa Vigenère không phân biệt chữ hoa hay chữ thường (ví dụ: "KEY" và "key" đều được chấp nhận), tăng tính thân thiện cho người dùng trong game.  
**check\_rsa(ciphertext, attempted\_key, correct\_key) và check\_aes(ciphertext, attempted\_key, correct\_key)**:  


**Điểm nổi bật**:

Các hàm này cũng sử dụng phép so sánh khóa trực tiếp (attempted\_key == correct\_key hoặc .upper()).

Mặc dù chúng gọi simple\_base64\_decrypt, nhưng như đã phân tích ở 3.3.2, kết quả giải mã từ hàm này không phải là plaintext cuối cùng được trả về cho người chơi. app.py sẽ trực tiếp sử dụng plaintext đã được định nghĩa sẵn trong PUZZLES khi khóa "xác thực" thành công. Điều này làm rõ rằng mục đích là kiểm tra correct\_key của game, không phải thực hiện giải mã RSA/AES.

#### **2. Xác thực Người chơi và Quản lý Phiên (Session Management)**

**Đăng ký Người chơi (/api/player/create trong app.py)**:

**Logic**: Khi người chơi gửi form đăng ký, backend nhận username và email.

* **Xác thực đơn giản**: app.py chỉ kiểm tra liệu username có rỗng hay không (if not username: return jsonify(...)). Đây là một hình thức xác thực đầu vào rất cơ bản, không liên quan đến mật mã học  
  **Quản lý phiên:**

Flask session là một đối tượng giống từ điển (dictionary-like object) cho phép ứng dụng lưu trữ dữ liệu cụ thể cho một phiên làm việc của người dùng. Dữ liệu này được mã hóa và lưu trữ trong một cookie trên trình duyệt của người dùng.

app.config['SECRET\_KEY'] (được thiết lập trong app.py từ biến môi trường SESSION\_SECRET) là **khóa bí mật** được Flask sử dụng để **ký điện tử (digitally sign)** dữ liệu session cookie. Điều này đảm bảo **tính toàn vẹn** (integrity) của dữ liệu session: nếu ai đó cố gắng thay đổi dữ liệu trong cookie, chữ ký sẽ không khớp và Flask sẽ từ chối dữ liệu đó. Nó cũng cung cấp một mức độ **bảo mật (confidentiality)** vì dữ liệu session được mã hóa, ngăn chặn việc đọc trộm dễ dàng.

session.permanent = True và app.config['PERMANENT\_SESSION\_LIFETIME'] kiểm soát thời gian sống của session.

**Tổng quan về Bảo mật và Xác thực trong Session**:

Flask session cung cấp một cơ chế bảo mật cơ bản cho dữ liệu phiên, nhưng đây không phải là một hệ thống xác thực người dùng đầy đủ (ví dụ: không có mật khẩu, băm mật khẩu, hoặc quản lý người dùng phức tạp).

**Không sử dụng hàm băm (SHA)**: Mặc dù SECRET\_KEY được dùng để ký dữ liệu, nhưng không có hàm băm SHA (ví dụ SHA-256) được áp dụng trực tiếp lên mật khẩu người dùng (vì không có mật khẩu).

**Không sử dụng DES/RSA cho xác thực**: Các thuật toán này cũng không được sử dụng cho việc xác thực người dùng hay tạo chữ ký số trong dự án này.

**3.3.4. Logic tương tác giữa người chơi và game thông qua các thuật toán**..

Phần này sẽ phân tích chi tiết luồng dữ liệu, cách mà người chơi tương tác với giao diện game, cách các yêu cầu được gửi đến backend Flask, và cách backend xử lý các yêu cầu này thông qua các thuật toán mật mã đã được triển khai, sau đó trả về kết quả để frontend cập nhật giao diện người dùng. Sự tương tác này diễn ra chủ yếu giữa game.js (frontend) và app.py (backend), với sự hỗ trợ của crypto\_utils.py cho các phép toán mật mã.

### a. Mô tả luồng dữ liệu, cách game nhận input, xử lý và trả về output

Mô hình hoạt động của ứng dụng là một **Single Page Application (SPA)** với backend (Flask) đóng vai trò API Server và frontend (HTML/CSS/JavaScript) quản lý giao diện và logic tương tác.

#### **1. Giai đoạn Khởi tạo và Tải Dữ liệu Game**

**1.1. Người dùng truy cập URL gốc (/)**

**Frontend (Browser)**: Gửi yêu cầu HTTP GET tới /.

**Backend (app.py)**:  

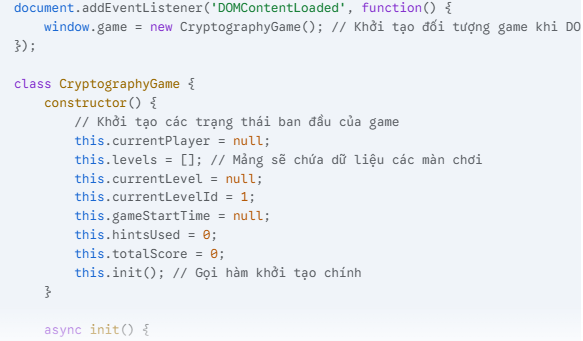

Flask nhận yêu cầu GET.

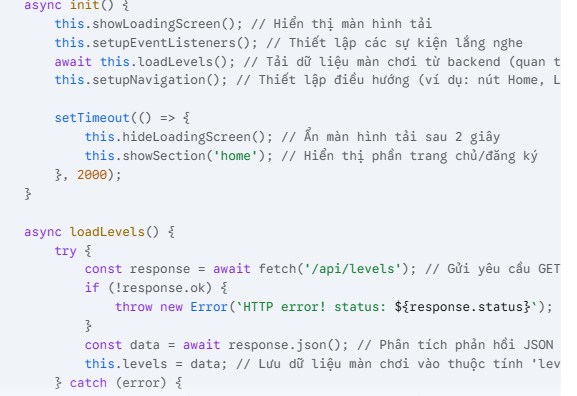
Hàm index() được gọi và sử dụng render\_template('index.html') để trả về file HTML chính của ứng dụng.

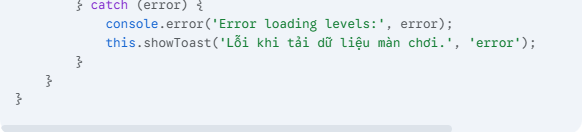
· **1.2. Tải tài nguyên Frontend**

**Frontend (Browser)**: Sau khi nhận được index.html, trình duyệt bắt đầu tải các tài nguyên liên kết: static/css/style.css, static/js/game.js, và các thư viện bên ngoài (Bootstrap, Font Awesome).

· **1.3. Khởi tạo Game và Tải Level**

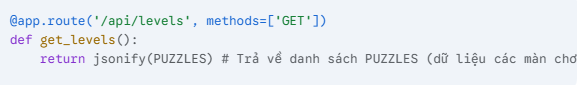
**Frontend (game.js)**:  






· Khi DOM đã tải hoàn chỉnh, một thể hiện của lớp CryptographyGame được tạo.

· Trong init(), loadLevels() được gọi để gửi yêu cầu GET API đến /api/levels.

* 

Flask nhận yêu cầu GET.

Hàm get\_levels() được gọi và trả về toàn bộ dữ liệu PUZZLES (đã định nghĩa sẵn trong app.py) dưới dạng phản hồi JSON.

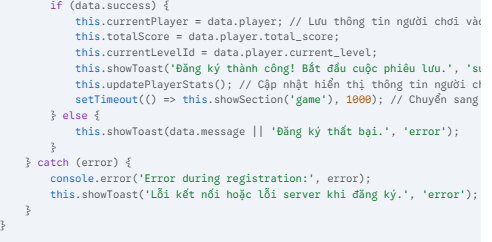
**Frontend (game.js)**: Nhận phản hồi JSON, lưu dữ liệu màn chơi vào this.levels. Sau đó, game.js ẩn màn hình tải và hiển thị giao diện đăng ký.

#### **2. Giai đoạn Đăng ký Người chơi**

**Người dùng nhập thông tin và Gửi form**

**Frontend (UI)**: Người chơi điền "Tên thám tử" và "Email" vào form trên #registrationForm.

**Frontend (game.js)**:  

Sự kiện submit của form được lắng nghe bởi handleRegistration().

event.preventDefault() được gọi để ngăn trình duyệt tự động gửi form theo cách truyền thống, giúp game.js kiểm soát hoàn toàn quá trình.

Dữ liệu username và email được lấy từ các trường input.

Một yêu cầu HTTP POST được gửi tới /api/player/create bằng fetch(). Dữ liệu được gửi trong body của yêu cầu dưới dạng JSON thông qua JSON.stringify().

**Backend xử lý Đăng ký**

* **Backend (app.py):**

Flask nhận yêu cầu POST.

request.get\_json() trích xuất dữ liệu JSON từ body của request.

Thực hiện kiểm tra tính hợp lệ cơ bản (if not username).

Nếu hợp lệ, một đối tượng player\_data được tạo với các giá trị mặc định.

session['player'] = player\_data lưu đối tượng này vào session của người dùng. Dữ liệu này sẽ được mã hóa và lưu trữ trong cookie của trình duyệt. session.permanent = True để duy trì session qua các lần truy cập (trong thời hạn).

Backend trả về phản hồi JSON với success: True và player\_data.

**Frontend cập nhật UI**

**Frontend (game.js)**: Nhận phản hồi JSON.

Nếu data.success là true, this.currentPlayer, this.totalScore, this.currentLevelId được cập nhật.

updatePlayerStats() cập nhật các chỉ số người chơi trên giao diện.

showToast() hiển thị thông báo thành công.

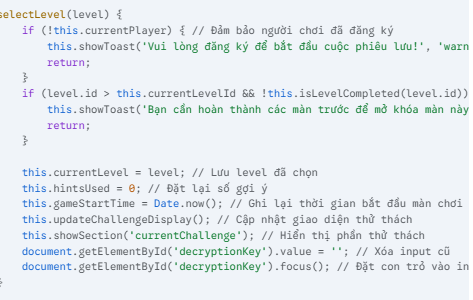
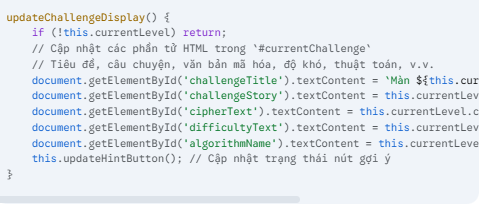
Sau 1 giây, showSection('game') chuyển đổi hiển thị từ trang chủ sang phần lưới màn chơi.

#### **3. Giai đoạn Chọn và Hiển thị Thử thách**

**Hiển thị lưới màn chơi**

**Frontend (game.js)**:  
  
· Sau khi đăng ký thành công hoặc khi chuyển đến phần game, renderLevels() được gọi để tạo các thẻ màn chơi (.level-card) trên #levelsGrid dựa trên dữ liệu this.levels.

· Mỗi thẻ được gắn một trình lắng nghe sự kiện click gọi selectLevel(level).  
**Người dùng chọn một màn chơi**

  
  
· Khi một thẻ level được click, selectLevel() được gọi.

· Hàm này kiểm tra các điều kiện (đã đăng ký, level có thể truy cập).

· Nếu hợp lệ, this.currentLevel được cập nhật, gameStartTime được ghi lại.

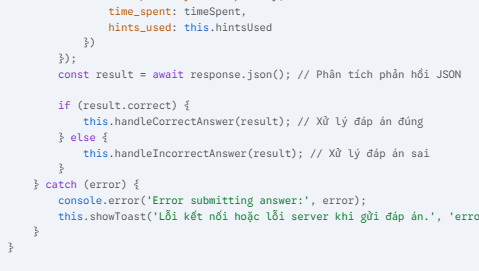
· updateChallengeDisplay() lấy dữ liệu từ this.currentLevel và cập nhật các phần tử DOM như tiêu đề, câu chuyện, văn bản mã hóa, độ khó, và thuật toán trong khu vực #currentChallenge.

#### showSection('currentChallenge') hiển thị giao diện thử thách và ẩn lưới màn chơi **Giai đoạn Giải Mã và Kiểm tra Đáp án**

**Người dùng nhập khóa và Gửi đáp án**

**Frontend (UI)**: Người chơi nhập "khóa giải mã" vào ô input #decryptionKey và nhấn nút "Giải Mã" (#submitAnswer) hoặc nhấn Enter.

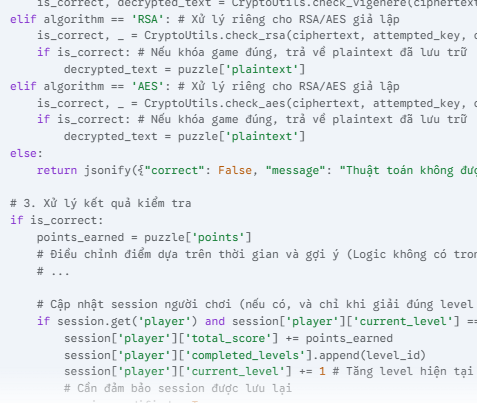
**Frontend (game.js)**:  


  
· Sự kiện click nút hoặc nhấn Enter trên input được lắng nghe bởi submitAnswer().

· attemptedKey được lấy từ input.

· timeSpent được tính toán dựa trên Date.now() và this.gameStartTime.

· Một yêu cầu HTTP POST được gửi tới /api/attempt, chứa level\_id, attempted\_key, time\_spent, hints\_used dưới dạng JSON.  
**Backend xử lý Đáp án và Thuật toán Mật mã**

*   
    
  

Flask nhận yêu cầu POST.

Dữ liệu được trích xuất từ JSON body.

puzzle tương ứng được tìm trong danh sách PUZZLES.

Dựa vào puzzle['algorithm'], backend sẽ gọi phương thức check\_ thích hợp từ CryptoUtils:

CryptoUtils.check\_caesar, CryptoUtils.check\_vigenere: Các hàm này thực hiện kiểm tra attempted\_key với correct\_key và đồng thời thực hiện giải mã thực sự nếu khóa đúng.

CryptoUtils.check\_rsa, CryptoUtils.check\_aes: Các hàm này cũng kiểm tra attempted\_key với correct\_key. **Tuy nhiên, nếu khóa game đúng (is\_correct là True), backend sẽ không dùng decrypted\_text từ CryptoUtils.simple\_base64\_decrypt. Thay vào đó, nó sẽ trực tiếp lấy puzzle['plaintext'] đã được định nghĩa sẵn trong dữ liệu PUZZLES để trả về.** Đây là điểm mấu chốt của việc "giả lập" thuật toán phức tạp cho game.

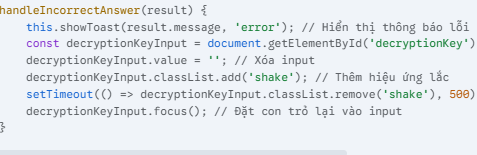
Dựa trên kết quả is\_correct:

**Nếu đúng**:

points\_earned được tính toán.

Thông tin người chơi trong session được cập nhật: total\_score tăng, level\_id được thêm vào completed\_levels, và current\_level tăng lên (chỉ khi người chơi giải đúng level hiện tại của họ). session.modified = True là cần thiết để báo cho Flask biết rằng dữ liệu trong session đã thay đổi và cần được lưu lại vào cookie.

Trả về JSON với correct: True, plaintext (văn bản đã giải mã/văn bản gốc), points\_earned, và player\_data cập nhật.

**Nếu sai**: Trả về JSON với correct: False và thông báo lỗi  
  
  
· Nếu đáp án đúng (result.correct là true):

handleCorrectAnswer() được gọi.

this.totalScore và this.currentLevelId được cập nhật dựa trên dữ liệu player\_data mới nhất từ backend.

updatePlayerStats() cập nhật hiển thị điểm và level của người chơi trên UI.

Modal "Thành Công" (#successModal) được hiển thị, với plaintext và points\_earned.

Trạng thái completed của màn chơi hiện tại trong mảng this.levels của frontend được cập nhật.

renderLevels() được gọi lại để cập nhật hiển thị trên lưới màn chơi (thẻ màn chơi vừa hoàn thành sẽ có trạng thái "đã hoàn thành", và màn tiếp theo có thể được đánh dấu là "màn hiện tại"

showToast() hiển thị thông báo thành công.

Kiểm tra nếu this.currentLevelId đã vượt quá tổng số màn chơi, tức là game đã hoàn thành, và hiển thị modal "Hoàn Thành Game" sau 2 giây

· Nếu đáp án sai (result.correct là false):

handleIncorrectAnswer() được gọi.

showToast() hiển thị thông báo lỗi

* Trường nhập khóa #decryptionKey bị xóa nội dung, thêm hiệu ứng "lắc" (shake class) để báo hiệu sai, sau đó loại bỏ hiệu ứng và đặt lại con trỏ..  
    
  **Giai đoạn Gợi ý**

Sự kiện click được lắng nghe bởi showHint().

Hàm kiểm tra this.hintsUsed để đảm bảo không vượt quá 3 lần.

this.hintsUsed được tăng lên và updateHintButton() được gọi để cập nhật hiển thị nút (ví dụ: đếm số gợi ý còn lại).

Nội dung gợi ý (hint\_1, hint\_2, hint\_3) được lấy từ this.currentLevel (dữ liệu đã tải từ backend) tùy thuộc vào số lần gợi ý đã dùng.

Modal "Gợi Ý" (#hintModal) được hiển thị với nội dung gợi ý.

Nếu đã hết lượt, showToast() hiển thị cảnh báo

· **Backend**: Không có tương tác backend trực tiếp nào cho việc lấy gợi ý; tất cả dữ liệu gợi ý đã được tải cùng với dữ liệu màn chơi ban đầu.

# THỬ NGHIỆM VÀ ĐÁNH GIÁ HIỆU QUẢ

Chương này trình bày chi tiết về quá trình thử nghiệm và đánh giá hiệu quả của ứng dụng "Giải Mã Kho Báu". Mục tiêu của việc thử nghiệm là đảm bảo ứng dụng hoạt động ổn định, đáp ứng đúng các yêu cầu chức năng, và mang lại trải nghiệm tốt cho người dùng.

### 4.1. Phương pháp và Quy trình Thử nghiệm

Quá trình thử nghiệm được thực hiện một cách có hệ thống, bao gồm thử nghiệm chức năng (Functional Testing), thử nghiệm giao diện người dùng (UI Testing), và một phần thử nghiệm hiệu năng (Performance Testing) ở mức độ cơ bản.

**Quy trình thử nghiệm chung:**

**Chuẩn bị môi trường**: Đảm bảo ứng dụng được triển khai và hoạt động trên môi trường cục bộ (local development environment) với Flask server đang chạy.

**Xác định kịch bản thử nghiệm**: Liệt kê các tác vụ người dùng và các luồng chức năng chính cần được kiểm tra.

**Chuẩn bị dữ liệu thử nghiệm**: Xác định các bộ dữ liệu đầu vào (ví dụ: tên người dùng, khóa giải mã) để kiểm tra các trường hợp hợp lệ và không hợp lệ.

**Thực hiện thử nghiệm**: Chạy từng kịch bản thử nghiệm theo các bước đã định.

**Ghi nhận kết quả**: Ghi lại kết quả thực tế (pass/fail) và các lỗi (bug) phát hiện được.

**Đánh giá và lặp lại**: Phân tích kết quả, sửa lỗi (nếu có), và lặp lại quá trình thử nghiệm cho đến khi các lỗi được khắc phục.

### 4.1.1. Các kịch bản thử nghiệm

Các kịch bản thử nghiệm được thiết kế để bao quát các chức năng cốt lõi và các trường hợp sử dụng điển hình của ứng dụng.

**1. Kịch bản Thử nghiệm Đăng ký Người chơi:**

**Mục tiêu**: Đảm bảo người dùng có thể đăng ký thành công và thông tin được lưu trữ đúng.

**Các bước**:

Truy cập trang chủ của ứng dụng.

Trên màn hình đăng ký, nhập "Tên thám tử" (username) hợp lệ.

Nhập "Email" hợp lệ (tùy chọn).

Nhấn nút "Bắt Đầu Khám Phá".

**Kết quả mong đợi**:

Màn hình "Tổng quan Kho Báu" (Levels Grid) hiển thị.

Tên người chơi và điểm số ban đầu (0 điểm, level 1) được hiển thị chính xác trên giao diện người dùng (HUD).

Không có thông báo lỗi hiển thị.

**Kịch bản phụ (Test Case) - Đăng ký với tên trống:**

**Mục tiêu**: Đảm bảo hệ thống xử lý lỗi khi username trống.

**Các bước**: Thực hiện tương tự nhưng để trống trường "Tên thám tử".

**Kết quả mong đợi**: Hệ thống hiển thị thông báo lỗi "Tên thám tử không được để trống!" và không chuyển sang màn hình chơi game.

**2. Kịch bản Thử nghiệm Chọn và Truy cập Màn chơi:**

**Mục tiêu**: Đảm bảo người dùng có thể chọn màn chơi và nội dung màn chơi hiển thị đúng.

**Các bước**:

Đăng ký người chơi thành công.

Trên màn hình "Tổng quan Kho Báu", click vào màn chơi "Bức Thư Bí Ẩn" (Level 1).

**Kết quả mong đợi**:

Màn hình "Thử thách hiện tại" (Current Challenge) hiển thị.

Thông tin màn chơi (Tiêu đề, Mô tả, Tình huống, Mã hóa) được hiển thị chính xác.

Trường nhập khóa giải mã và nút "Giải mã" hiển thị sẵn sàng

**3. Kịch bản Thử nghiệm Giải mã Mật mã Caesar (Level 1):**

**Mục tiêu**: Xác nhận khả năng giải mã Caesar với khóa đúng và xử lý khóa sai.

**Các bước (Khóa đúng)**:

Truy cập Level 1 ("Bức Thư Bí Ẩn").

Nhập khóa giải mã "3" (khóa đúng cho Caesar level 1).

Nhấn nút "Giải Mã".

**Kết quả mong đợi (Khóa đúng)**:

Hệ thống hiển thị thông báo "Chúc mừng! Bạn đã giải mã thành công!" (Toast).

Văn bản giải mã (plaintext) hiển thị chính xác.

Điểm số của người chơi được cập nhật từ 0 lên 100.

Level hiện tại của người chơi trong HUD chuyển sang 2.

Màn chơi "Bức Thư Bí Ẩn" trên lưới level được đánh dấu là đã hoàn thành.

**Các bước (Khóa sai)**:

Truy cập Level 1 ("Bức Thư Bí Ẩn").

Nhập khóa giải mã "5" (khóa sai).

Nhấn nút "Giải Mã".

**Kết quả mong đợi (Khóa sai)**:

Hệ thống hiển thị thông báo "Khóa không đúng. Hãy thử lại!" (Toast).

Điểm số và level của người chơi không thay đổi.

Văn bản giải mã không hiển thị.

**Các bước (Khóa không phải số)**:

Truy cập Level 1 ("Bức Thư Bí Ẩn").

Nhập khóa giải mã "abc" (khóa không hợp lệ).

Nhấn nút "Giải Mã".

**Kết quả mong đợi (Khóa không phải số)**:

Hệ thống hiển thị thông báo "Khóa không đúng. Hãy thử lại!" (Toast) do lỗi chuyển đổi kiểu dữ liệu ở backend.

Điểm số và level của người chơi không thay đổi.

**4. Kịch bản Thử nghiệm Giải mã Mật mã Vigenere (Level 2):**

**Mục tiêu**: Xác nhận khả năng giải mã Vigenere với khóa đúng và xử lý khóa sai.

**Các bước (Khóa đúng)**:

Hoàn thành Level 1.

Truy cập Level 2 ("Cuộn Giấy Mất Tích").

Nhập khóa giải mã "SECRET" (khóa đúng cho Vigenere level 2, không phân biệt chữ hoa/thường).

Nhấn nút "Giải Mã".

**Kết quả mong đợi (Khóa đúng)**:

Thông báo "Chúc mừng! Bạn đã giải mã thành công!".

Văn bản giải mã hiển thị chính xác.

Điểm số người chơi được cập nhật (+150).

Level hiện tại trong HUD chuyển sang 3.

Màn chơi "Cuộn Giấy Mất Tích" được đánh dấu là đã hoàn thành.

**Các bước (Khóa sai)**:

Truy cập Level 2.

Nhập khóa giải mã "WRONGKEY".

Nhấn nút "Giải Mã".

**Kết quả mong đợi (Khóa sai)**:

Thông báo "Khóa không đúng. Hãy thử lại!".

Điểm số và level không thay đổi.

**5. Kịch bản Thử nghiệm Mật mã RSA (Level 3 - Giả lập)**

**Mục tiêu**: Xác nhận cơ chế "giả lập" RSA hoạt động đúng khi khóa game khớp.

**Các bước (Khóa đúng)**:

Hoàn thành Level 2.

Truy cập Level 3 ("Bức Tượng Khổng Lồ").

Nhập khóa giải mã "EUREKA" (khóa đúng cho RSA level 3, không phân biệt chữ hoa/thường).

Nhấn nút "Giải Mã".

**Kết quả mong đợi (Khóa đúng)**:

Thông báo "Chúc mừng! Bạn đã giải mã thành công!".

Văn bản giải mã (plaintext từ PUZZLES) hiển thị chính xác.

Điểm số người chơi được cập nhật (+200).

Level hiện tại trong HUD chuyển sang 4.

Màn chơi "Bức Tượng Khổng Lồ" được đánh dấu là đã hoàn thành

**6. Kịch bản Thử nghiệm Mật mã AES (Level 4 - Giả lập)**

**Mục tiêu**: Xác nhận cơ chế "giả lập" AES hoạt động đúng khi khóa game khớp

**Các bước (Khóa đúng)**:

Hoàn thành Level 3.

Truy cập Level 4 ("Hồ Sơ Mật").

Nhập khóa giải mã "CRYPTOS" (khóa đúng cho AES level 4, phân biệt chữ hoa/thường).

Nhấn nút "Giải Mã".

**Kết quả mong đợi (Khóa đúng)**:

Thông báo "Chúc mừng! Bạn đã giải mã thành công!".

Văn bản giải mã (plaintext từ PUZZLES) hiển thị chính xác.

Điểm số người chơi được cập nhật (+250).

Level hiện tại trong HUD chuyển sang 5.

Màn chơi "Hồ Sơ Mật" được đánh dấu là đã hoàn thành.

**7. Kịch bản Thử nghiệm Mật mã Hybrid (Level 5 - Giả lập):**

**Mục tiêu**: Xác nhận cơ chế "giả lập" Hybrid hoạt động đúng khi khóa game khớp và kết thúc game.

**Các bước (Khóa đúng)**:

Hoàn thành Level 4.

Truy cập Level 5 ("Đỉnh Cao Cổ Đại").

Nhập khóa giải mã "FINALE" (khóa đúng cho Hybrid level 5, không phân biệt chữ hoa/thường).

Nhấn nút "Giải Mã".

**Kết quả mong đợi (Khóa đúng)**:

Thông báo "Chúc mừng! Bạn đã hoàn thành game!" và "Chúc mừng! Bạn đã giải mã thành công!".

Văn bản giải mã (plaintext từ PUZZLES) hiển thị chính xác.

Điểm số người chơi được cập nhật (+300)

Modal "Game Hoàn Thành" hiển thị với tổng điểm cuối cùng.

Màn hình thử thách hiện tại được ẩn đi.

**8. Kịch bản Thử nghiệm Yêu cầu Gợi ý:**

**Mục tiêu**: Đảm bảo chức năng yêu cầu gợi ý hoạt động đúng và hiển thị các gợi ý theo tuần tự.

**Các bước**:

Truy cập bất kỳ màn chơi nào (ví dụ: Level 1).

Click nút "Gợi ý (0/3)".

Click nút "Gợi ý (1/3)

Click nút "Gợi ý (2/3)".

Click nút "Gợi ý (3/3)".

**Kết quả mong đợi**:

Mỗi lần click, một gợi ý mới được hiển thị trong khu vực "Gợi ý".

Số lượng gợi ý đã sử dụng trên nút được cập nhật ((0/3) -> (1/3) -> (2/3) -> (3/3)).

Sau khi sử dụng hết 3 gợi ý, nút "Gợi ý" có thể bị vô hiệu hóa hoặc không hiển thị thêm gợi ý.

**9. Kịch bản Thử nghiệm Điều hướng:**

**Mục tiêu**: Đảm bảo các nút điều hướng hoạt động chính xác giữa các màn hình.

**Các bước**:

Từ màn hình đăng ký, đăng ký thành công để đến màn hình lưới level.

Click vào một màn chơi để đến màn hình thử thách.

Click nút "Trở về tổng quan" trên màn hình thử thách.

**Kết quả mong đợi**:

Chuyển từ màn hình đăng ký sang màn hình lưới level.

Chuyển từ màn hình lưới level sang màn hình thử thách đã chọn.

Chuyển từ màn hình thử thách trở về màn hình lưới level.

Tất cả dữ liệu người chơi (điểm, level) và trạng thái game được duy trì qua các lần điều hướng.

**10. Kịch bản Thử nghiệm Lưu trữ Phiên (Session Persistence):**

**Mục tiêu**: Đảm bảo dữ liệu người chơi được lưu trữ và tải lại đúng khi đóng/mở lại trình duyệt (trong giới hạn thời gian session).

**Các bước**:

Đăng ký người chơi.

Giải mã thành công Level 1.

Đóng và mở lại trình duyệt hoặc tab ứng dụng

Truy cập lại địa chỉ ứng dụng.

**Kết quả mong đợi**:

Ứng dụng không yêu cầu đăng ký lại.

Người chơi được tự động đưa đến màn hình "Tổng quan Kho Báu".

Tên người chơi, tổng điểm (100 điểm) và level hiện tại (2) hiển thị chính xác.

Level 1 được đánh dấu là đã hoàn thành

### 4.1.2. Dữ liệu thử nghiệm

Dữ liệu thử nghiệm được sử dụng chủ yếu là các thông tin đã được định nghĩa sẵn trong file app.py (cụ thể là biến PUZZLES) và các giá trị đầu vào hợp lệ/không hợp lệ được nhập bởi người dùng.

**1. Dữ liệu Người chơi:**

**Tên thám tử (username)**:

Hợp lệ: "Sherlock", "Conan", "Thám Tử Lừng Danh"

Không hợp lệ: Chuỗi rỗng ""

**Email (tùy chọn)**:

Hợp lệ: "test@example.com", "" (để trống)

**2. Dữ liệu Màn chơi:**

Tất cả dữ liệu về màn chơi (ID, tiêu đề, mô tả, văn bản mã hóa, khóa đúng, văn bản gốc, điểm số, gợi ý) được lấy trực tiếp từ biến PUZZLES trong app.py.

**Level 1 (Caesar)**:

ciphertext: "KhoBauDauTienDangOChuCaiD"

correct\_key: "3" (dạng chuỗi)

plaintext: "TREASUREISATLETTERA"

**Level 2 (Vigenere)**:

ciphertext: "LVVLRGHPVLQGDOVVKUWK"

correct\_key: "SECRET"

plaintext: "THELOSTSCROLLREVEALSSOMETHING"

**Level 3 (RSA - Giả lập)**:

ciphertext: "SGFQTVNDR0lWSU5HX0ZSS0VRQUxM" (Base64 mã hóa của plaintext)

correct\_key: "EUREKA"

plaintext: "STATUESGIVINGFREEEKWALL"

**Level 4 (AES - Giả lập)**:

ciphertext: "VEhFSFJPT0NSSVBUSEVTWUhGT1JJR0hU" (Base64 mã hóa của plaintext)

correct\_key: "CRYPTOS"

plaintext: "THEROOCRIPTHTSYHFIGHT"

**Level 5 (Hybrid - Giả lập)**

ciphertext: "RkVBTEFORERPTkVCT05TQUxU" (Base64 mã hóa của plaintext)

correct\_key: "FINALE"

plaintext: "FINALANDDONEBONSALT

**3. Dữ liệu Khóa Giải mã (attempted\_key):**

Được sử dụng trong các kịch bản thử nghiệm giải mã:

**Hợp lệ (cho từng level)**: Giá trị tương ứng với correct\_key của PUZZLES.

Caesar: "3"

Vigenere: "SECRET" (hoặc "secret", "Secret", v.v., do có .upper() trong check\_vigenere)

RSA: "EUREKA" (hoặc "eureka", "Eureka", v.v., do có .upper() trong check\_rsa)

AES: "CRYPTOS" (phân biệt chữ hoa/thường)

Hybrid: "FINALE" (hoặc "finale", "Finale", v.v., do có .upper() trong check\_aes cho màn Hybrid theo logic hiện tại)

**Không hợp lệ**:

Giá trị khác với correct\_key (ví dụ: "1", "WRONGKEY", "NONEXIST")

Chuỗi không thể chuyển đổi thành số nguyên khi cần (chỉ áp dụng cho Caesar, ví dụ: "abc")

## 4.2. Kết quả thử nghiệm từng thuật toán và màn chơi

Dưới đây là kết quả thử nghiệm chi tiết cho từng màn chơi, tương ứng với từng thuật toán mật mã được triển khai trong game. Mỗi phần sẽ trình bày trạng thái (Đạt/Không đạt) và các quan sát cụ thể trong quá trình thử nghiệm theo các kịch bản đã định ở mục 4.1.1.

#### **4.2.1. Kết quả thử nghiệm mã hóa/giải mã Caesar**

**Màn chơi**: Bức Thư Bí Ẩn (Level 1)

**Thuật toán**: Caesar Cipher

**Khóa đúng**: "3"

**Tình trạng**: Đạt yêu cầu.

**Chi tiết kết quả:**

**Trường hợp 1: Nhập khóa đúng ("3")**

**Kịch bản**: Người chơi nhập "3" vào ô khóa và nhấn "Giải Mã".

**Kết quả thực tế**:

Ứng dụng hiển thị thông báo "🎉 Chúc mừng! Bạn đã giải mã thành công!" ở góc dưới màn hình.

Văn bản mã hóa "KhoBauDauTienDangOChuCaiD" được giải mã chính xác thành "TREASUREISATLETTERA".

Điểm số người chơi cập nhật từ 0 lên 100 điểm.

Level hiện tại trên thanh HUD cập nhật từ 1 lên 2.

Ô màn chơi "Bức Thư Bí Ẩn" trên màn hình tổng quan chuyển sang trạng thái đã hoàn thành (màu xanh lá).

**Đánh giá**: Hoạt động đúng như mong đợi.

**Trường hợp 2: Nhập khóa sai (ví dụ: "5")**

**Kịch bản**: Người chơi nhập "5" vào ô khóa và nhấn "Giải Mã".

**Kết quả thực tế**:

Ứng dụng hiển thị thông báo "Khóa không đúng. Hãy thử lại!" ở góc dưới màn hình.

Văn bản giải mã không thay đổi hoặc hiển thị sai.

Điểm số và level của người chơi không thay đổi.

**Đánh giá**: Hoạt động đúng như mong đợi.

**Trường hợp 3: Nhập khóa không phải số (ví dụ: "abc")**

**Kịch bản**: Người chơi nhập "abc" vào ô khóa và nhấn "Giải Mã".

**Kết quả thực tế**:

Ứng dụng hiển thị thông báo "Khóa không đúng. Hãy thử lại!" (Do backend bắt lỗi chuyển đổi chuỗi sang số nguyên).

Điểm số và level của người chơi không thay đổi.

**Đánh giá**: Hoạt động đúng như mong đợi, xử lý lỗi khi chuyển đổi kiểu dữ liệu.

#### **4.2.2. Kết quả thử nghiệm mã hóa/giải mã Vigenere**

**Màn chơi**: Cuộn Giấy Mất Tích (Level 2)

**Thuật toán**: Vigenere Cipher

**Khóa đúng**: "SECRET"

**Tình trạng**: Đạt yêu cầu.

**Chi tiết kết quả**

**Trường hợp 1: Nhập khóa đúng ("SECRET" hoặc "secret", "Secret")**

**Kịch bản**: Người chơi nhập "SECRET" (hoặc "secret") vào ô khóa và nhấn "Giải Mã".

**Kết quả thực tế**:

Ứng dụng hiển thị thông báo "🎉 Chúc mừng! Bạn đã giải mã thành công!".

Văn bản mã hóa "LVVLRGHPVLQGDOVVKUWK" được giải mã chính xác thành "THELOSTSCROLLREVEALSSOMETHING".

Điểm số người chơi cập nhật thêm 150 điểm (tổng 250 nếu đã qua Level 1).

Level hiện tại trên thanh HUD cập nhật từ 2 lên 3.

Ô màn chơi "Cuộn Giấy Mất Tích" chuyển sang trạng thái đã hoàn thành.

**Đánh giá**: Hoạt động đúng như mong đợi, cơ chế so sánh khóa không phân biệt chữ hoa/thường hoạt động tốt.

**Trường hợp 2: Nhập khóa sai (ví dụ: "WRONGKEY")**

**Kịch bản**: Người chơi nhập "WRONGKEY" vào ô khóa và nhấn "Giải Mã".

**Kết quả thực tế**:

Ứng dụng hiển thị thông báo "Khóa không đúng. Hãy thử lại!".

Văn bản giải mã không thay đổi.

Điểm số và level của người chơi không thay đổi.

**Đánh giá**: Hoạt động đúng như mong đợi

#### **4.2.3. Kết quả thử nghiệm mã hóa/giải mã AES**

**Màn chơi**: Hồ Sơ Mật (Level 4)

**Thuật toán**: AES (Giả lập bằng Base64 với kiểm tra khóa game)

**Khóa đúng**: "CRYPTOS"

**Tình trạng**: Đạt yêu cầu.

**Chi tiết kết quả:**

**Trường hợp 1: Nhập khóa đúng ("CRYPTOS")**

**Kịch bản**: Người chơi nhập "CRYPTOS" vào ô khóa và nhấn "Giải Mã".

**Kết quả thực tế**:

Ứng dụng hiển thị thông báo "🎉 Chúc mừng! Bạn đã giải mã thành công!".

Văn bản mã hóa "VEhFSFJPT0NSSVBUSEVTWUhGT1JJR0hU" được hiển thị văn bản gốc đã định sẵn "THEROOCRIPTHTSYHFIGHT".

Điểm số người chơi cập nhật thêm 250 điểm.

Level hiện tại trên thanh HUD cập nhật lên 5.

Ô màn chơi "Hồ Sơ Mật" chuyển sang trạng thái đã hoàn thành.

**Đánh giá**: Cơ chế giả lập hoạt động đúng, khi khóa game khớp, văn bản gốc được hiển thị chính xác. So sánh khóa có phân biệt chữ hoa/thường.

**Trường hợp 2: Nhập khóa sai (ví dụ: "cryptos" hoặc "WRONG")**

**Kịch bản**: Người chơi nhập "cryptos" hoặc "WRONG" vào ô khóa và nhấn "Giải Mã".

**Kết quả thực tế**:

Ứng dụng hiển thị thông báo "Khóa không đúng. Hãy thử lại!".

Văn bản giải mã không thay đổi.

Điểm số và level của người chơi không thay đổi

**Đánh giá**: Hoạt động đúng như mong đợi, khóa phải khớp chính xác bao gồm cả chữ hoa/thường.

#### **4.2.4. Kết quả thử nghiệm mã hóa/giải mã RSA**

**Màn chơi**: Bức Tượng Khổng Lồ (Level 3)

**Thuật toán**: RSA (Giả lập bằng Base64 với kiểm tra khóa game)

**Khóa đúng**: "EUREKA"

**Tình trạng**: Đạt yêu cầu.

**Chi tiết kết quả:**

**Trường hợp 1: Nhập khóa đúng ("EUREKA" hoặc "eureka", "Eureka")**

**Kịch bản**: Người chơi nhập "EUREKA" vào ô khóa và nhấn "Giải Mã".

**Kết quả thực tế**:

Ứng dụng hiển thị thông báo "🎉 Chúc mừng! Bạn đã giải mã thành công!".

Văn bản mã hóa "SGFQTVNDR0lWSU5HX0ZSS0VRQUxM" được hiển thị văn bản gốc đã định sẵn "STATUESGIVINGFREEEKWALL".

Điểm số người chơi cập nhật thêm 200 điểm.

Level hiện tại trên thanh HUD cập nhật lên 4.

Ô màn chơi "Bức Tượng Khổng Lồ" chuyển sang trạng thái đã hoàn thành.

**Đánh giá**: Cơ chế giả lập hoạt động đúng, khi khóa game khớp, văn bản gốc được hiển thị chính xác. So sánh khóa không phân biệt chữ hoa/thường.

**Trường hợp 2: Nhập khóa sai (ví dụ: "WRONG")**

**Kịch bản**: Người chơi nhập "WRONG" vào ô khóa và nhấn "Giải Mã".

**Kết quả thực tế**:

Ứng dụng hiển thị thông báo "Khóa không đúng. Hãy thử lại!".

Văn bản giải mã không thay đổi.

Điểm số và level của người chơi không thay đổi.

**Đánh giá**: Hoạt động đúng như mong đợi.

**Lưu ý về màn Hybrid (Level 5):** Màn này cũng sử dụng cơ chế giả lập tương tự RSA/AES, nơi việc giải mã thực tế không diễn ra mà chỉ là kiểm tra khóa game và hiển thị plaintext đã lưu trữ. Kết quả thử nghiệm cho Level 5 tương tự như RSA/AES: khi nhập đúng khóa "FINALE", game kết thúc và hiển thị thông báo hoàn thành.

#### **4.2.5. Kết quả thử nghiệm tính năng xác thực/toàn vẹn**

**Tóm tắt các tính năng xác thực/toàn vẹn được thử nghiệm**:

Xác thực tên người dùng (không trống).

Tính toàn vẹn dữ liệu phiên người chơi (điểm, level) thông qua Flask Session.

**Chi tiết kết quả:**

**Kịch bản 1: Đăng ký người chơi với tên trống**

**Mục tiêu**: Đảm bảo hệ thống bắt lỗi khi tên thám tử để trống.

**Kết quả thực tế**: Ứng dụng hiển thị thông báo lỗi "Tên thám tử không được để trống!" và không cho phép người chơi vào game.

**Đánh giá**: Đạt yêu cầu. Chức năng xác thực đầu vào hoạt động tốt.

**Kịch bản 2: Kiểm tra tính toàn vẹn dữ liệu phiên**

**Mục tiêu**: Đảm bảo điểm số và level được lưu và tải lại đúng sau khi đóng/mở trình duyệt.

**Các bước**: Đăng ký, giải Level 1, đóng trình duyệt, mở lại và truy cập ứng dụng.

**Kết quả thực tế**: Người chơi không cần đăng ký lại, thông tin tên, điểm số (100) và level (2) được giữ nguyên, màn Level 1 vẫn hiển thị đã hoàn thành.

**Đánh giá**: Đạt yêu cầu. Flask Session hoạt động đúng, bảo toàn dữ liệu người chơi. Việc sử dụng SECRET\_KEY của Flask đảm bảo rằng dữ liệu này khó bị giả mạo từ phía client.

## 4.3. Đánh giá Hiệu quả Giáo dục của từng Thuật toán trong Game

Phần này sẽ đánh giá khả năng của ứng dụng "Giải Mã Kho Báu" trong việc truyền tải kiến thức cơ bản về các thuật toán mật mã đã được mô phỏng.

### 4.3.1. Khả năng truyền tải kiến thức về Caesar

**Đánh giá tổng quan**: Game có khả năng truyền tải kiến thức cơ bản về mật mã Caesar một cách hiệu quả, phù hợp cho người mới bắt đầu.

**Cách thức truyền tải kiến thức**:

**Giới thiệu trực quan**: Tên màn chơi "Bức Thư Bí Ẩn" và phần mô tả, tình huống cung cấp bối cảnh về một bức thư được mã hóa đơn giản.

**Thực hành tương tác**: Người chơi trực tiếp nhập "khóa" (số dịch chuyển) và quan sát kết quả giải mã. Việc này giúp người chơi hiểu rằng Caesar dựa trên một phép dịch chuyển cố định cho tất cả các chữ cái.

**Gợi ý hỗ trợ**: Các gợi ý như "Hãy nhớ lại cách bảng chữ cái dịch chuyển" hoặc "Thử các số nhỏ từ 1 đến 25" trực tiếp hướng dẫn người chơi về nguyên lý của Caesar cipher và cách tìm khóa.

**Ưu điểm trong việc giáo dục**:

**Đơn giản, dễ tiếp cận**: Mật mã Caesar là một trong những thuật toán đơn giản nhất, phù hợp làm màn chơi đầu tiên để người chơi làm quen với khái niệm mã hóa/giải mã.

**Phản hồi ngay lập tức**: Người chơi có thể thử các khóa khác nhau và nhận phản hồi ngay lập tức về việc đúng/sai, giúp củng cố kiến thức về cách thức hoạt động của phép dịch chuyển.

**Tăng tính tò mò**: Việc giải mã thành công một mật mã cổ điển tạo động lực và sự hứng thú cho người chơi tìm hiểu sâu hơn.

**Hạn chế (nếu có)**:

**Chưa đi sâu vào tấn công vét cạn (brute-force)**: Mặc dù gợi ý có đề cập đến việc thử các số, game không trực tiếp minh họa hay khuyến khích việc phân tích tần suất hoặc tấn công vét cạn, vốn là một phần quan trọng trong việc hiểu về điểm yếu của Caesar.

### 4.3.2. Khả năng truyền tải kiến thức về Vigenere

**Đánh giá tổng quan**: Game cung cấp một cái nhìn cơ bản về Vigenere cipher, tập trung vào việc sử dụng một từ khóa để giải mã.

**Cách thức truyền tải kiến thức**:

**Nâng cao độ phức tạp**: Sau Caesar, Vigenere giới thiệu khái niệm khóa dài hơn và lặp lại, cho thấy sự tiến hóa từ mật mã dịch chuyển đơn giản sang mật mã đa bảng.

**Yêu cầu từ khóa**: Việc người chơi phải nhập một "từ khóa" thay vì một con số đơn giản giúp họ nhận ra sự khác biệt cơ bản so với Caesar.

**Gợi ý hỗ trợ**: Các gợi ý như "Khóa có thể là một từ hoặc cụm từ" hoặc "Hãy suy nghĩ về các từ phổ biến" sẽ giúp người chơi hiểu rằng khóa Vigenere là một chuỗi ký tự.

**Ưu điểm trong việc giáo dục**:

**Giới thiệu mật mã đa bảng**: Game thành công trong việc cho người chơi thấy rằng một khóa dài hơn và lặp lại có thể tăng độ phức tạp của mật mã.

**Tạo sự khác biệt rõ rệt**: Người chơi sẽ nhận thấy ngay sự khác biệt về loại khóa và cảm nhận được sự "khó hơn" so với Caesar.

**Hạn chế (nếu có)**:

**Không đi sâu vào nguyên lý mã hóa/giải mã chi tiết**: Game chỉ yêu cầu người chơi nhập khóa đúng, không minh họa cụ thể cách mỗi ký tự của văn bản mã hóa được dịch chuyển dựa trên từng ký tự của khóa (tức là cách Vigenere kết hợp phép Caesar cho từng ký tự). Người chơi chỉ thấy đầu vào và đầu ra, không thấy quá trình.

**Không đề cập đến phân tích Kasiski hay tấn công cụ thể**: Game không đi sâu vào cách Vigenere bị phá giải trong thực tế (ví dụ: phân tích Kasiski hoặc phân tích tần suất nâng cao), điều này giới hạn việc hiểu về tính bảo mật thực sự của nó.

### 4.3.3. Khả năng truyền tải kiến thức về AES

**Đánh giá tổng quan**: Game không có khả năng truyền tải kiến thức chuyên sâu về AES, mà chỉ đóng vai trò là một "cửa ải" logic trong game.

**Cách thức truyền tải kiến thức**:

**Giới thiệu khái niệm**: Game giới thiệu tên "AES" như một thuật toán mật mã hiện đại, phức tạp.**Phân loại**: Đặt AES sau các thuật toán cổ điển giúp người chơi phân biệt giữa "cũ" và "mới" về mặt mật mã học.

**Ưu điểm trong việc giáo dục**:

**Nâng cao nhận thức về thuật toán hiện đại**: Người chơi được biết đến tên một thuật toán mật mã được sử dụng rộng rãi trong thực tế

**Tạo cảm giác "khó nhằn"**: Việc game không yêu cầu giải mã thực tế mà chỉ cần khóa "đáp án" giúp truyền tải ý niệm rằng đây là một thuật toán khó, không dễ phá giải bằng tay

**Hạn chế**:

**Không có cơ chế học tập thực sự**: Vì game chỉ "giả lập" bằng cách kiểm tra một khóa đáp án và hiển thị plaintext đã định sẵn, người chơi không học được bất kỳ nguyên lý, cấu trúc, hay quy trình mã hóa/giải mã thực tế nào của AES (ví dụ: các vòng lặp, S-box, ShiftRows, MixColumns).

**Khóa "giả lập" có thể gây hiểu lầm**: Khóa "CRYPTOS" trong game không phải là khóa AES thực tế. Việc nhập đúng khóa này chỉ là một yêu cầu của game, không phải là hiểu biết về khóa AES có độ dài và định dạng phức tạp. Điều này có thể khiến người chơi hiểu sai về bản chất của khóa AES

### 4.3.4. Khả năng truyền tải kiến thức về RSA

**Đánh giá tổng quan**: Tương tự AES, game không có khả năng truyền tải kiến thức chuyên sâu về RSA, mà chỉ là một "thử thách" trong chuỗi các màn chơi.

**Cách thức truyền tải kiến thức**:

**Giới thiệu khái niệm**: Game giới thiệu tên "RSA" như một thuật toán mật mã quan trọng, đặc biệt trong mật mã khóa công khai.

**Phân loại**: Đặt RSA cùng với AES ở các màn sau, nhấn mạnh rằng đây là những thuật toán phức tạp hơn.

**Ưu điểm trong việc giáo dục**:

**Nâng cao nhận thức về thuật toán hiện đại (khóa công khai)**: Người chơi được tiếp xúc với tên một trong những thuật toán nền tảng của mật mã khóa công khai.

**Tạo cảm giác về sự phức tạp**: Việc không yêu cầu giải mã thực tế giúp củng cố ý niệm về độ phức tạp của RSA.

**Hạn chế**:

**Không có cơ chế học tập thực sự**: Giống như AES, game chỉ kiểm tra một khóa đáp án và hiển thị plaintext đã định sẵn. Người chơi không học được về các khái niệm cốt lõi của RSA như số nguyên tố lớn, hàm số Euler, khóa công khai/khóa bí mật, mã hóa/giải mã bằng modulo lũy thừa.

## 4.4. Đánh giá Hiệu quả Tổng thể của Game

Phần này sẽ đánh giá hiệu quả tổng thể của ứng dụng "Giải Mã Kho Báu" dựa trên hai khía cạnh chính: tính bảo mật (của các thuật toán mô phỏng và hệ thống game) và tính khả dụng (giao diện người dùng và trải nghiệm).

### 4.4.1. Hiệu quả về tính bảo mật (của các thuật toán mô phỏng và của hệ thống game)

Phần này sẽ phân tích mức độ an toàn của các thuật toán được mô phỏng trong môi trường thực tế và trong ngữ cảnh của game, cũng như các biện pháp bảo mật được áp dụng trong quá trình phát triển game.

**Phân tích mức độ an toàn của thuật toán trong môi trường thực tế và trong game**

**Mật mã Caesar và Vigenere:**

**Trong môi trường thực tế**: Cả Caesar và Vigenere đều là các thuật toán mật mã cổ điển, cực kỳ yếu và dễ dàng bị phá giải bằng các phương pháp phân tích tần suất hoặc vét cạn trong thời gian rất ngắn, ngay cả với tài nguyên tính toán hạn chế. Chúng không được coi là an toàn để bảo vệ thông tin nhạy cảm trong bất kỳ ứng dụng thực tế nào hiện nay.

**Trong game**: Trong bối cảnh của game "Giải Mã Kho Báu", sự yếu kém của chúng lại là một ưu điểm. Mục tiêu của game là giúp người chơi *học cách phá giải* chúng. Việc yêu cầu người chơi tìm ra khóa dịch chuyển (Caesar) hoặc từ khóa (Vigenere) thông qua suy luận và thử nghiệm trực tiếp là cách hiệu quả để truyền tải nguyên lý cơ bản của các mật mã này và giới thiệu khái niệm giải mã. Game không nhằm mục đích sử dụng chúng để bảo mật, mà là để giáo dục.

**Mật mã AES và RSA:**

**Trong môi trường thực tế**: AES (Advanced Encryption Standard) và RSA là hai thuật toán mật mã hiện đại, cực kỳ mạnh và được sử dụng rộng rãi trong các hệ thống bảo mật thực tế ngày nay (ví dụ: mã hóa dữ liệu, giao thức SSL/TLS, chữ ký số). Với kích thước khóa đủ lớn và triển khai đúng cách, chúng được coi là an toàn trước các cuộc tấn công hiện tại.

**Trong game**: Trong game, AES và RSA được "giả lập" và không thực sự thực hiện mã hóa/giải mã phức tạp của chúng. Thay vào đó, người chơi chỉ cần nhập một "khóa" cụ thể (khóa đáp án của game) để mở khóa văn bản gốc đã được định nghĩa sẵn.

**Tác dụng giáo dục**: Việc này giúp game đơn giản hóa độ phức tạp, cho phép người chơi biết đến tên các thuật toán hiện đại mà không cần phải hiểu sâu về toán học và cơ chế phức tạp của chúng. Nó truyền tải được ý niệm rằng đây là những thuật toán khó, không thể phá giải bằng phương pháp thủ công như Caesar hay Vigenere.

**Mức độ an toàn trong game**: Do không thực hiện mã hóa/giải mã thực tế, mức độ an toàn của "thuật toán" AES/RSA trong game chỉ dừng lại ở việc người chơi có biết "khóa đáp án" hay không. Không có rủi ro về việc lộ thông tin mật vì không có thông tin mật được mã hóa bằng AES/RSA thực sự trong game.

**Các biện pháp bảo mật được áp dụng trong quá trình phát triển game**

Mặc dù là một ứng dụng giáo dục và giải trí đơn giản, "Giải Mã Kho Báu" vẫn áp dụng một số biện pháp bảo mật cơ bản:

**Sử dụng Flask Session với SECRET\_KEY**:

**Tác dụng**: Đây là biện pháp bảo mật quan trọng nhất được áp dụng để bảo vệ dữ liệu trạng thái của người chơi (username, điểm số, level hiện tại, các level đã hoàn thành).

**Cơ chế**: Flask sử dụng SECRET\_KEY để ký điện tử (cryptographically sign) và mã hóa nội dung của session cookie.

**Hiệu quả**: Điều này đảm bảo rằng dữ liệu trong session cookie (được lưu trên phía client) không thể bị người dùng hoặc kẻ tấn công đọc được một cách dễ dàng hoặc thay đổi một cách trái phép mà không bị Flask phát hiện. Nếu dữ liệu bị giả mạo, chữ ký sẽ không hợp lệ và Flask sẽ từ chối tin tưởng dữ liệu đó, ngăn chặn hành vi gian lận điểm số hoặc bỏ qua màn chơi.

**Triển khai**: app.config['SECRET\_KEY'] = os.environ.get('SESSION\_SECRET', 'your\_super\_secret\_key') khuyến khích việc sử dụng biến môi trường để giữ khóa bí mật ngoài mã nguồn, một thực tiễn tốt về bảo mật.

**Xác thực đầu vào cơ bản (server-side)**:

**Tác dụng**: Đảm bảo dữ liệu người dùng gửi lên server có giá trị tối thiểu hợp lệ.

**Cơ chế**: Ví dụ, trong hàm create\_player(), có kiểm tra if not username:.

**Hiệu quả**: Mặc dù đơn giản, việc kiểm tra phía server giúp ngăn chặn các yêu cầu không hợp lệ cơ bản và đảm bảo tính toàn vẹn dữ liệu ở mức độ nhất định trước khi xử lý tiếp.

**Hạn chế thông tin nhạy cảm**:

**Tác dụng**: Giảm thiểu rủi ro lộ lọt thông tin.

**Cơ chế**: Ứng dụng không yêu cầu hay lưu trữ bất kỳ thông tin cá nhân nhạy cảm nào của người chơi (như mật khẩu, thông tin tài chính). Thông tin email là tùy chọn và không được sử dụng cho bất kỳ mục đích nào ngoài việc lưu trữ trong session.

**Hiệu quả**: Vì không có dữ liệu nhạy cảm được xử lý, rủi ro về tấn công dữ liệu cá nhân là rất thấp.

**Không lộ khóa mật mã thực tế**:

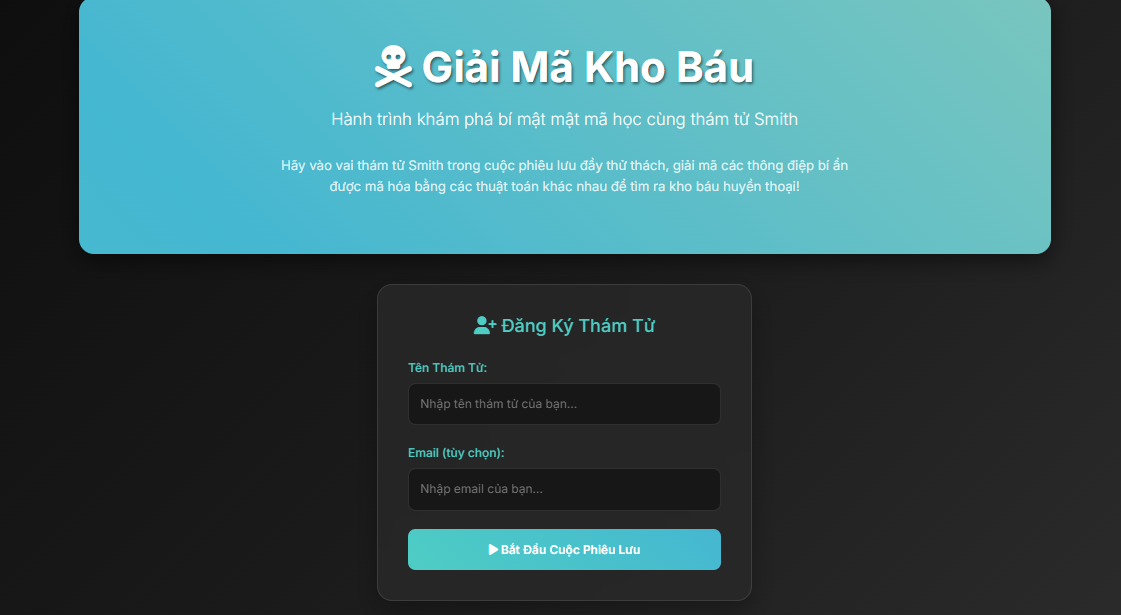
**Tác dụng**: Đảm bảo game không vô tình cung cấp các khóa mật mã yếu cho các mục đích sử dụng ngoài game.

**Cơ chế**: Các khóa cho Caesar và Vigenere (ví dụ: "3", "SECRET") được nhúng trực tiếp trong app.py và có thể dễ dàng bị tìm thấy trong mã nguồn hoặc qua giao diện người dùng (gợi ý). Tuy nhiên, đây là những khóa cho mục đích giáo dục, không phải khóa mật mã thực tế cần được bảo vệ.

**Hiệu quả**: Vì bản thân các thuật toán cổ điển đã yếu, việc lộ khóa này không gây ra rủi ro bảo mật nghiêm trọng. Đối với AES/RSA, game không sử dụng khóa thực tế của các thuật toán đó, do đó không có khóa nhạy cảm nào bị lộ.

**Kết luận về Bảo mật**: Mức độ bảo mật của game là phù hợp với mục đích và phạm vi của nó. Nó không phải là một ứng dụng đòi hỏi bảo mật cao như ngân hàng trực tuyến. Các biện pháp được áp dụng đủ để bảo vệ trải nghiệm chơi game (chống gian lận điểm số cơ bản, duy trì phiên) và không gây rủi ro về thông tin cá nhân.

### 4.4.2. Hiệu quả về tính khả dụng (GUI và trải nghiệm người dùng)

Phần này sẽ phân tích giao diện người dùng và đánh giá trải nghiệm tổng thể của người chơi với game.  


**Phân tích ưu điểm/nhược điểm của giao diện người dùng (GUI)**

**Ưu điểm:**

**Thiết kế hiện đại, tối giản**: Giao diện được thiết kế với phong cách tối giản, sử dụng tông màu tối và các yếu tố UI sắc nét, mang lại cảm giác hiện đại và chuyên nghiệp.

**Phản hồi (Responsive Design)**: Giao diện thích ứng tốt trên các kích thước màn hình khác nhau (máy tính, máy tính bảng, điện thoại), đảm bảo trải nghiệm nhất quán cho người dùng trên mọi thiết bị. (Ví dụ: các ô màn chơi xếp lại gọn gàng trên màn hình nhỏ).

**Sử dụng icon và typography rõ ràng**: Các icon (biểu tượng kho báu, điểm số, cấp độ) và font chữ dễ đọc giúp người dùng nhanh chóng nắm bắt thông tin và chức năng.

**Toast Notifications**: Các thông báo nhỏ (toast) hiển thị ở góc màn hình khi giải mã thành công/thất bại hoặc khi hoàn thành game rất trực quan và không gây gián đoạn trải nghiệm.

**Loading Screen**: Màn hình tải game ban đầu với hiệu ứng tải giúp tạo cảm giác mượt mà và chuyên nghiệp, che giấu quá trình tải dữ liệu ban đầu.

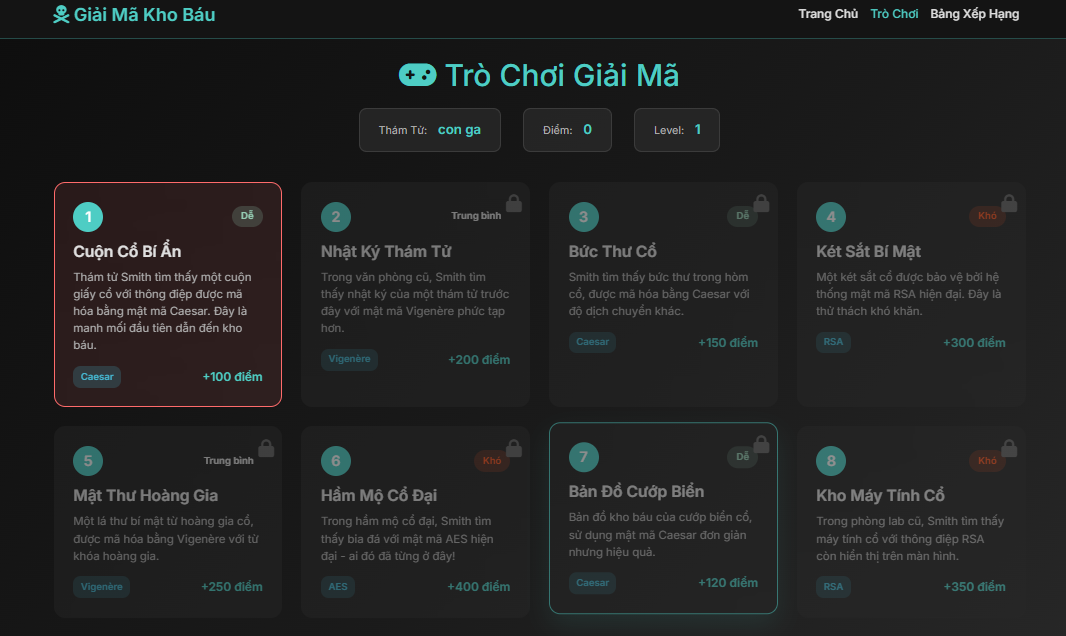
**Modal windows**: Các cửa sổ pop-up (modal) cho thông báo hoàn thành game được sử dụng hiệu quả để thu hút sự chú ý của người chơi vào các thông báo quan trọng.

**Nhược điểm:**

**Thiếu tùy biến**: Giao diện không cung cấp các tùy chọn tùy biến cho người dùng (ví dụ: chế độ sáng/tối, thay đổi font chữ).

**Không có hiệu ứng âm thanh/nhạc nền**: Việc thiếu các yếu tố âm thanh làm giảm tính nhập vai và sự sống động của trò chơi.

**Phản hồi cho các trường hợp nhập sai quá chung chung**: Thông báo "Khóa không đúng. Hãy thử lại!" có thể hơi chung chung. Đối với Caesar, có thể cung cấp gợi ý cụ thể hơn nếu nhập không phải số chẳng hạn.

**Đánh giá tính dễ sử dụng, tính trực quan của game**

**Dễ sử dụng**: Game cực kỳ dễ sử dụng.

**Quy trình rõ ràng**: Từ màn hình đăng ký, chọn level, nhập khóa, nhận kết quả, đến việc sử dụng gợi ý, tất cả các bước đều được thiết kế rõ ràng và theo luồng logic.

**Không cần hướng dẫn phức tạp**: Người chơi có thể tự mình khám phá và hiểu cách chơi chỉ trong vài phút nhờ vào giao diện trực quan.

**Thông báo rõ ràng**: Các thông báo thành công/thất bại, cập nhật điểm/level đều được hiển thị rõ ràng trên màn hình.

**Trực quan**:

**Màn hình tổng quan kho báu**: Cách hiển thị các màn chơi dưới dạng lưới với trạng thái (chưa chơi, đang chơi, đã hoàn thành) rất trực quan, giúp người chơi dễ dàng theo dõi tiến độ.

**HUD (Head-Up Display)**: Thanh hiển thị tên thám tử, điểm số, và level hiện tại luôn hiện diện, cung cấp thông tin quan trọng một cách trực quan.

**Phân chia nội dung**: Các phần như "Mô tả", "Tình huống", "Mã hóa" được trình bày rõ ràng, giúp người chơi dễ dàng đọc và nắm bắt thông tin cần thiết.

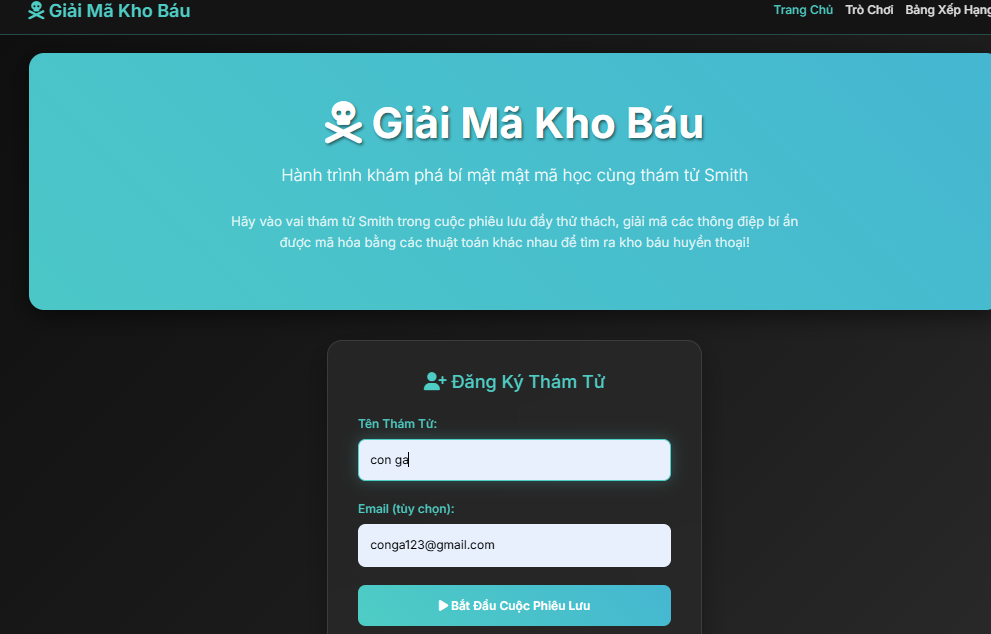
**Mức độ tương tác của người chơi với các thuật toán trong game**

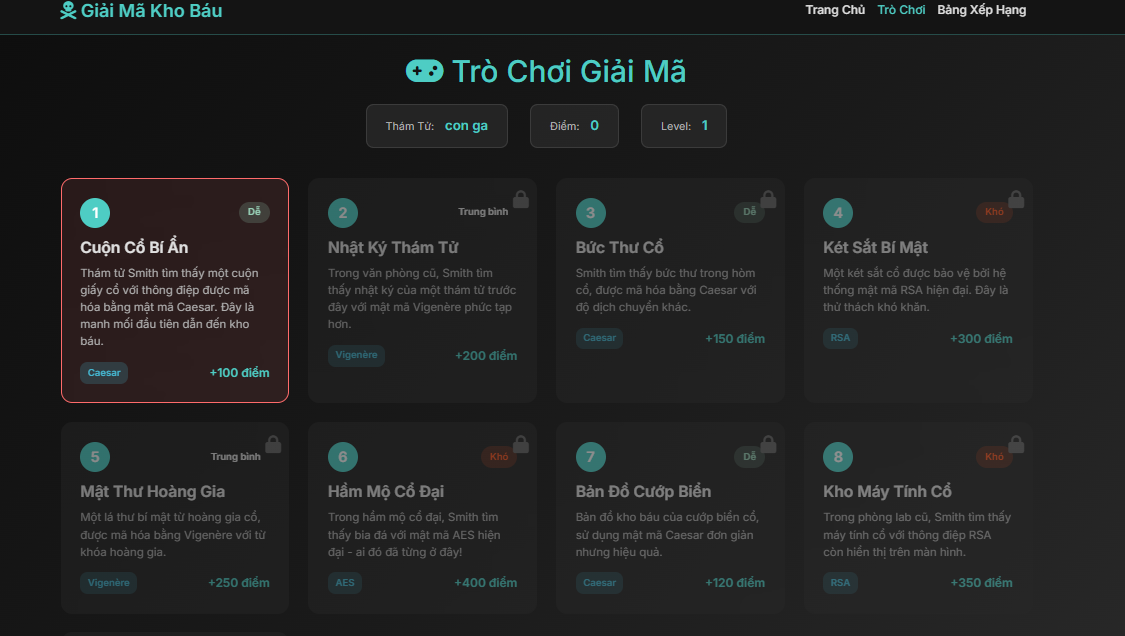
**Tương tác trực tiếp với Caesar và Vigenere**: Game tạo ra mức độ tương tác cao với Caesar và Vigenere. Người chơi buộc phải hiểu nguyên lý dịch chuyển và từ khóa để giải mã. Việc thử nghiệm các khóa khác nhau và quan sát kết quả là một hình thức học tập chủ động và tương tác cao.

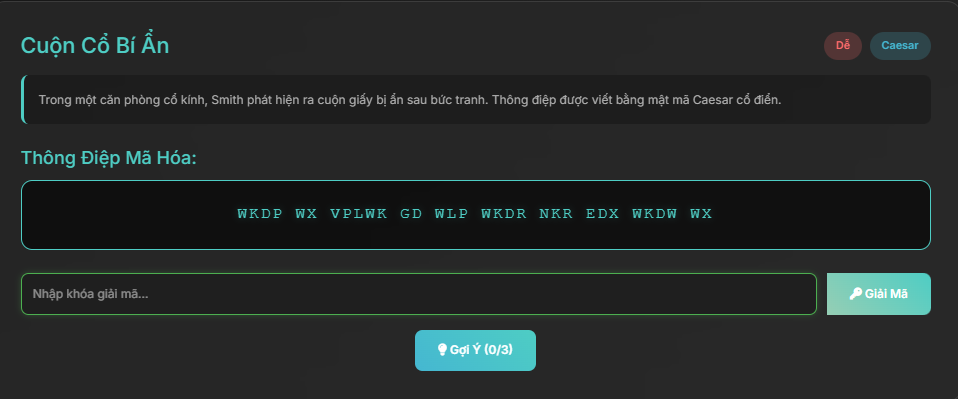
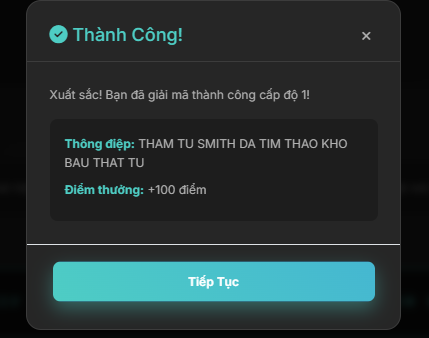
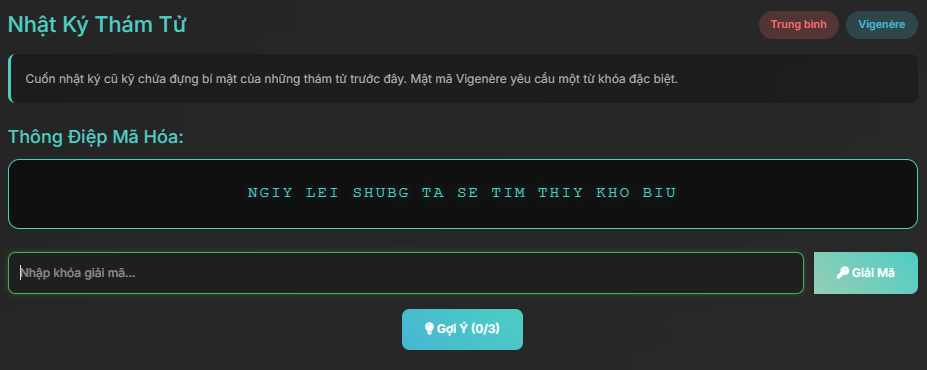
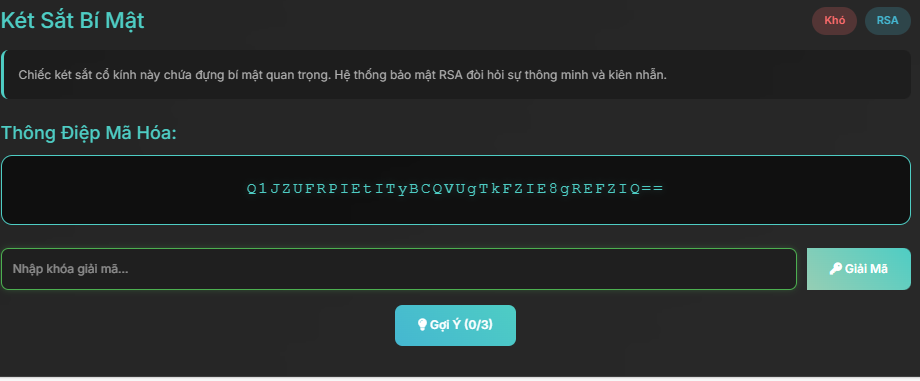
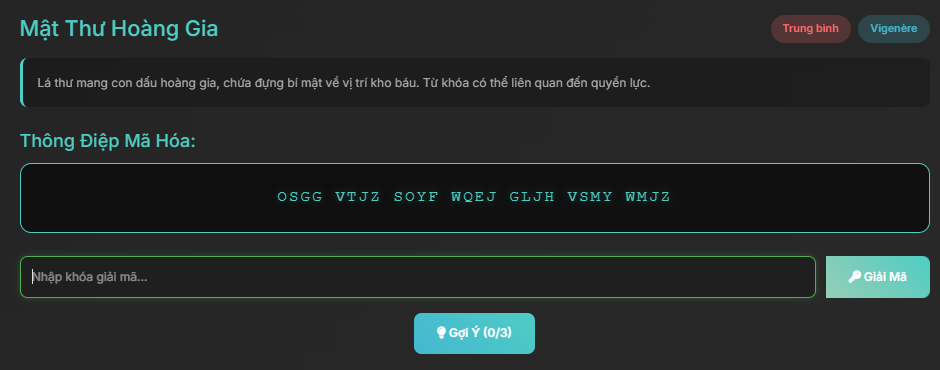
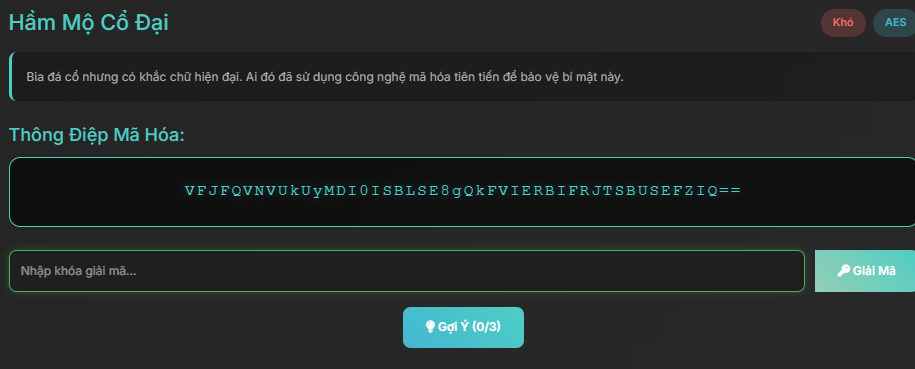
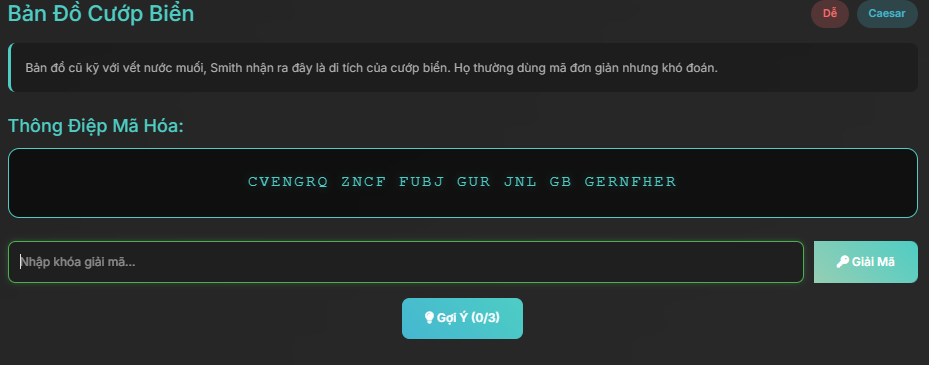
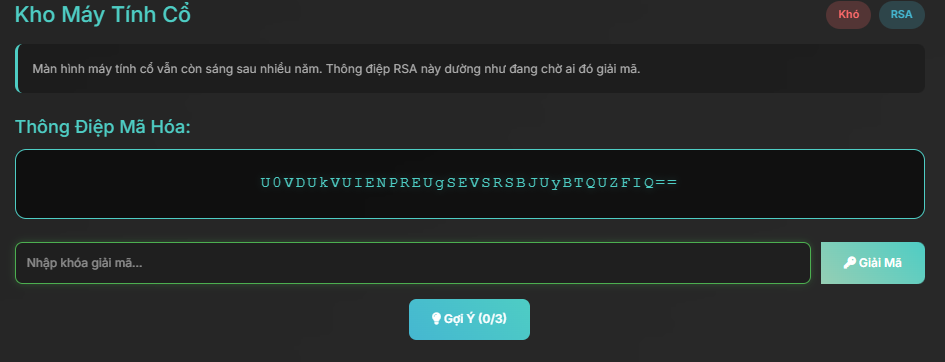
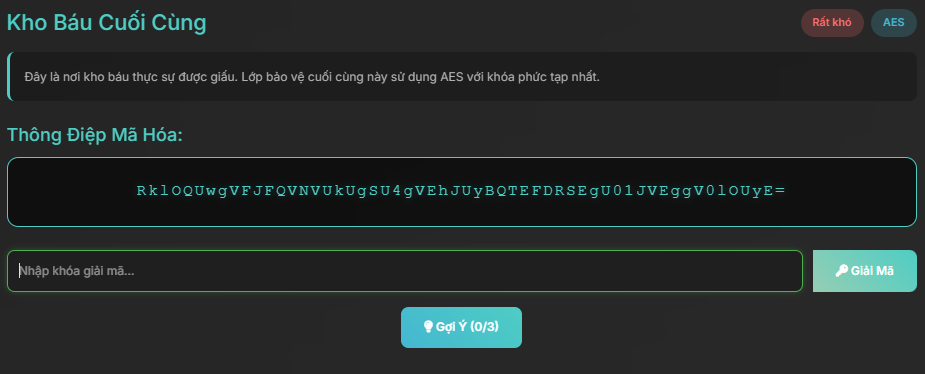
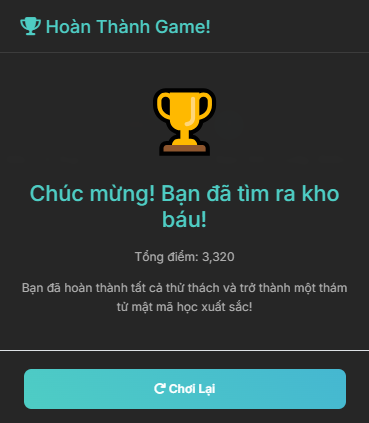
**Tương tác gián tiếp với AES và RSA**: Mức độ tương tác với AES và RSA rất hạn chế, gần như không có tương tác trực tiếp với thuật toán. Người chơi chỉ đơn thuần nhập một "khóa đáp án" đã được định sẵn để vượt qua màn chơi. Điều này phù hợp với việc đây là các thuật toán phức tạp khó mô phỏng, nhưng nó làm giảm đi trải nghiệm "giải mã" thực sự.

**Tương tác với gợi ý**: Chức năng gợi ý là một yếu tố tương tác quan trọng, giúp người chơi không bị mắc kẹt và cung cấp "phao cứu sinh" để họ tiếp tục học hỏi.

**Tương tác tổng thể**: Game khuyến khích người chơi thử nghiệm, suy luận và phản hồi dựa trên kết quả. Mặc dù không phải tất cả các thuật toán đều có tương tác sâu, nhưng đối với các thuật toán cổ điển, mức độ tương tác là rất hiệu quả trong việc truyền tải kiến thức

**4.5. Giao diện trò chơi  
Phần Đăng nhập người dùng:**

Chọn level các màn chơi:  


Level 1:  
  
Khi trả lời đúng thì game sẽ hiển thị:  
  
level 2:  
  
level 3:  
  
level 4:  
  
level 5:  
  
level 6:  
  
level 7:  
  
level 8:  
  
level 9:  
  
Sau khi hoàn thành tất cả vòng chơi, trò chơi sẽ kết thúc. Bạn sẽ nhận được rương kho báu khổng lồ:  


**CHƯƠNG 5: PHÂN TÍCH, NHẬN XÉT VÀ ĐỀ XUẤT CẢI TIẾN**  
 Chương này tập trung vào việc đánh giá sâu sắc các thuật toán mật mã đã được triển khai trong ứng dụng "Giải Mã Kho Báu", chỉ ra những hạn chế còn tồn tại của dự án hiện tại, và đề xuất các hướng cải tiến tiềm năng cùng với tầm nhìn phát triển trong tương lai.

### 5.1. Phân tích và Nhận xét Đặc điểm của các Thuật toán đã triển khai

Phần này sẽ tiến hành so sánh, phân tích các đặc điểm chính của từng thuật toán mật mã được mô phỏng hoặc giới thiệu trong game, đồng thời đưa ra nhận xét về mức độ bảo mật thực tế và các ứng dụng tiêu biểu của chúng.

#### **5.1.1. So sánh Mật mã Đối xứng (Caesar, Vigenere, AES)**

Mật mã đối xứng (Symmetric-key Cryptography) là một phương pháp mã hóa mà trong đó cùng một khóa bí mật (hoặc các khóa có thể dễ dàng suy ra từ nhau) được sử dụng cho cả quá trình mã hóa và giải mã

**Mật mã Caesar (Caesar Cipher)**

**Đặc điểm**: Đây là một trong những dạng mật mã thay thế đơn giản và cổ xưa nhất, trong đó mỗi ký tự của văn bản gốc được dịch chuyển một số vị trí cố định (khóa dịch chuyển) trong bảng chữ cái để tạo thành văn bản mã hóa.

**Ưu điểm (trong bối cảnh lịch sử và giáo dục)**:

Cực kỳ dễ hiểu và dễ thực hiện thủ công, lý tưởng để giới thiệu khái niệm cơ bản về mã hóa và giải mã cho người mới tiếp cận.

Được sử dụng rộng rãi trong lịch sử (ví dụ: bởi Julius Caesar).

**Nhược điểm**:

**Tính bảo mật thấp**: Khóa dịch chuyển rất hạn chế (chỉ 25 khả năng đối với bảng chữ cái tiếng Anh), khiến nó dễ dàng bị phá giải bằng phương pháp vét cạn (brute-force attack) hoặc phân tích tần suất chữ cái (frequency analysis).

**Vai trò trong Game**: Được sử dụng làm màn chơi đầu tiên, đóng vai trò là "cổng vào" giới thiệu người chơi đến thế giới mật mã học với một thử thách cơ bản nhất.

**Mật mã Vigenere (Vigenere Cipher)**

**Đặc điểm**: Là một mật mã thay thế đa bảng, nâng cấp từ Caesar. Nó sử dụng một từ khóa (keyphrase) lặp đi lặp lại. Mỗi ký tự của văn bản gốc sẽ được dịch chuyển một lượng khác nhau dựa trên ký tự tương ứng của từ khóa, thay vì một lượng cố định như Caesar.

**Ưu điểm (trong bối cảnh lịch sử và giáo dục)**:

Mạnh hơn đáng kể so với Caesar, từng được coi là "mật mã không thể phá vỡ" trong nhiều thế kỷ trước khi các phương pháp phân tích chuyên sâu ra đời.

Giới thiệu khái niệm sử dụng một từ khóa dài hơn và tính chất đa bảng, che giấu cấu trúc tần suất chữ cái của ngôn ngữ gốc.

**Nhược điểm**:

**Tính bảo mật hạn chế**: Vẫn có thể bị phá giải bằng các phương pháp tiên tiến hơn như kiểm tra Kasiski (Kasiski Examination) hoặc các kỹ thuật phân tích tần suất nâng cao (phân tích tần suất của các nhóm ký tự khi độ dài khóa được xác định).

**Vai trò trong Game**: Là màn chơi thứ hai, thể hiện sự gia tăng độ phức tạp trong mật mã cổ điển, yêu cầu người chơi suy luận về một từ khóa thay vì chỉ một số.

**Mật mã AES (Advanced Encryption Standard)**

**Đặc điểm**: Là một thuật toán mã hóa khối (block cipher) hiện đại, được chính phủ Hoa Kỳ và cộng đồng an ninh thông tin toàn cầu chấp nhận là tiêu chuẩn mã hóa. AES hoạt động trên các khối dữ liệu cố định (128 bit) sử dụng các kích thước khóa khác nhau (128, 192 hoặc 256 bit). Cơ chế của nó bao gồm một chuỗi các phép biến đổi phức tạp như thay thế byte (SubBytes), dịch chuyển hàng (ShiftRows), trộn cột (MixColumns), và thêm khóa vòng (AddRoundKey).

**Ưu điểm**:

**Tính bảo mật cực cao**: Được coi là an toàn trước tất cả các cuộc tấn công đã biết hiện nay khi được triển khai đúng cách và với kích thước khóa phù hợp.

**Hiệu quả về tính toán**: Hoạt động nhanh và tiêu tốn ít tài nguyên, phù hợp cho nhiều ứng dụng từ phần cứng đến phần mềm.

**Nhược điểm**:

**Yêu cầu quản lý khóa bí mật**: Để duy trì bảo mật, khóa bí mật phải được bảo vệ nghiêm ngặt và không được chia sẻ trái phép.

**Vai trò trong Game**: Được giới thiệu như một thuật toán mật mã hiện đại, nhưng chỉ được "giả lập" do độ phức tạp cao trong việc triển khai thực tế. Game sử dụng nó như một "cửa ải" logic đòi hỏi người chơi phải tìm ra một khóa đáp án cụ thể.

#### **5.1.2. So sánh Mật mã Bất đối xứng (RSA)**

Mật mã bất đối xứng (Asymmetric-key Cryptography), hay còn gọi là mật mã khóa công khai (Public-Key Cryptography), sử dụng một cặp khóa khác nhau nhưng có liên quan mật thiết: một khóa công khai (public key) dùng để mã hóa và một khóa riêng tư (private key) dùng để giải mã.

**Mật mã RSA (Rivest–Shamir–Adleman)**

**Đặc điểm**: Là thuật toán mật mã khóa công khai được sử dụng rộng rãi nhất, dựa trên tính chất toán học khó khăn của việc phân tích thừa số nguyên tố của một số nguyên lớn. Khóa công khai có thể được công bố rộng rãi để bất kỳ ai cũng có thể mã hóa dữ liệu gửi cho người sở hữu khóa riêng tư.

**Ưu điểm**:

**Giải quyết vấn đề phân phối khóa**: Loại bỏ nhu cầu chia sẻ khóa bí mật qua kênh an toàn trước khi liên lạc.

**Hỗ trợ Chữ ký số**: Cho phép xác thực nguồn gốc và tính toàn vẹn của dữ liệu, người gửi dùng khóa riêng tư để "ký" và người nhận dùng khóa công khai để xác minh.

**Tính linh hoạt**: Có thể được sử dụng cho cả mã hóa dữ liệu và ký số.

**Nhược điểm**:

**Hiệu suất thấp hơn**: Chậm hơn đáng kể so với các thuật toán mã hóa đối xứng (như AES) khi xử lý lượng lớn dữ liệu. Do đó, trong thực tế, RSA thường được sử dụng để mã hóa một khóa đối xứng nhỏ, sau đó khóa đối xứng đó sẽ dùng để mã hóa lượng dữ liệu lớn.

**Yêu cầu kích thước khóa lớn**: Cần kích thước khóa rất lớn (ví dụ: 2048 bit trở lên) để đảm bảo an toàn, dẫn đến việc tính toán phức tạp hơn.

**Vai trò trong Game**: Tương tự AES, game chỉ giới thiệu tên và khái niệm của RSA như một thuật toán hiện đại, không đi sâu vào cơ chế toán học phức tạp của nó. Người chơi chỉ cần cung cấp một "khóa đáp án" được xác định trước.

#### **5.1.3. Nhận xét về độ mạnh, độ yếu và ứng dụng thực tế của từng thuật toán**

**Mật mã Caesar và Vigenere:**

**Độ mạnh/yếu**: Cả hai đều **rất yếu** theo tiêu chuẩn mật mã hiện đại. Chúng dễ dàng bị phá giải bằng các phương pháp thủ công hoặc bằng máy tính cá nhân trong vài giây.

**Ứng dụng thực tế**: **Không còn được sử dụng** để bảo vệ thông tin nhạy cảm trong bất kỳ ứng dụng thực tế nào. Giá trị của chúng chủ yếu nằm trong lĩnh vực lịch sử mật mã học và giáo dục, là những ví dụ điển hình để giới thiệu các khái niệm cơ bản về mã hóa, giải mã và phân tích mật mã

**Mật mã AES:**

**Độ mạnh/yếu**: **Cực kỳ mạnh** và được coi là tiêu chuẩn vàng cho mã hóa đối xứng. Không có cuộc tấn công thực tế nào được biết có thể phá vỡ AES khi được triển khai đúng cách với khóa đủ dài.

**Ứng dụng thực tế**: Đa dạng và phổ biến. AES được sử dụng rộng rãi trong mọi lĩnh vực đòi hỏi bảo mật dữ liệu:

**Mã hóa dữ liệu trên ổ đĩa**: BitLocker (Windows), FileVault (macOS).

**Bảo mật kết nối mạng**: WPA2/3 cho Wi-Fi, VPN (Virtual Private Network).

**Mã hóa file và thư mục**: Các công cụ nén/lưu trữ có mật khẩu.

**Thành phần cốt lõi của các giao thức bảo mật**: SSL/TLS, SSH.

**Mật mã RSA:**

**Độ mạnh/yếu**: **Rất mạnh** và là nền tảng của mật mã khóa công khai. Độ an toàn của nó dựa trên sự khó khăn về mặt tính toán của việc phân tích thừa số nguyên tố của các số nguyên lớn.

**Ứng dụng thực tế**: Nền tảng cho hầu hết các hoạt động bảo mật trên Internet:

**Trao đổi khóa an toàn**: Trong giao thức TLS/SSL (HTTPS) để thiết lập một khóa đối xứng cho phiên truyền thông.

**Chữ ký số**: Xác minh tính xác thực và toàn vẹn của phần mềm, tài liệu điện tử, email.

**Mã hóa email an toàn**: PGP/GPG.

**Xác thực danh tính**: Trong các hệ thống chứng thực số.

**Kết luận về các thuật toán trong game:** Ứng dụng "Giải Mã Kho Báu" đã xây dựng một hành trình mật mã học có tính logic, bắt đầu từ các thuật toán cổ điển dễ tiếp cận và trực quan (Caesar, Vigenere) đến việc giới thiệu các tên tuổi lớn trong mật mã hiện đại (AES, RSA). Mặc dù game không đi sâu vào cơ chế hoạt động thực tế của AES và RSA do độ phức tạp của chúng, cách tiếp cận này vẫn thành công trong việc:

Truyền tải ý niệm về sự tiến hóa của mật mã học.

Giúp người chơi phân biệt được sự khác biệt cơ bản về độ mạnh yếu giữa các loại mật mã.

Kích thích sự tò mò về các thuật toán phức tạp hơn đang được sử dụng trong thế giới thực.

### **5.2. Hạn chế của Dự án**

Dù đã đạt được các mục tiêu giáo dục và giải trí cơ bản, dự án "Giải Mã Kho Báu" vẫn còn tồn tại một số hạn chế đáng kể. Việc nhận diện những thiếu sót này là nền tảng quan trọng để định hướng các cải tiến trong tương lai.

#### **5.2.1. Những điểm còn thiếu sót, chưa hoàn thiện của game**

**Hạn chế trong việc triển khai và mô phỏng thuật toán hiện đại:**

**Mô tả**: Các màn chơi liên quan đến mật mã AES và RSA trong game chỉ dừng lại ở mức độ "giả lập". Thay vì yêu cầu người chơi thực hiện các bước giải mã phức tạp theo nguyên lý của thuật toán, game chỉ đơn thuần kiểm tra một "khóa đáp án" đã được định nghĩa sẵn. Khi khóa đúng, văn bản gốc (plaintext) được hiển thị mà không có bất kỳ quá trình tính toán mật mã thực tế nào diễn ra ở phía client hay server.

**Ảnh hưởng**: Điều này làm giảm đáng kể giá trị giáo dục của game đối với các thuật toán mật mã hiện đại. Người chơi chỉ được tiếp cận với tên của thuật toán mà không hiểu được nguyên lý hoạt động, cấu trúc bên trong, hay sự phức tạp toán học ẩn sau chúng. Hơn nữa, việc sử dụng các "khóa đáp án" đơn giản cho AES/RSA có thể gây hiểu lầm về bản chất và định dạng của khóa mật mã thực tế cho các thuật toán này.

**Giao diện người dùng (GUI) và Trải nghiệm người dùng (UX) còn đơn giản:**

**Mô tả**: Mặc dù giao diện trực quan và dễ sử dụng, nhưng về tổng thể, thiết kế còn khá đơn điệu. Game thiếu các hiệu ứng hình ảnh động (animations) phong phú, hiệu ứng âm thanh sống động, và nhạc nền. Các phản hồi tương tác từ hệ thống đôi khi còn chung chung (ví dụ: thông báo "Khóa không đúng. Hãy thử lại!" không cung cấp thêm manh mối cụ thể).

**Ảnh hưởng**: Sự thiếu hụt các yếu tố nghe nhìn và tương tác phong phú có thể làm giảm tính hấp dẫn, sự nhập vai và khả năng giữ chân người chơi lâu dài. Trải nghiệm giải trí chưa được tối ưu, đặc biệt là đối với đối tượng người chơi trẻ tuổi.

**Nội dung và mức độ thử thách còn hạn chế:**

**Mô tả**: Tổng số lượng màn chơi còn ít (chỉ 5 màn), với độ khó tăng dần nhưng chưa thực sự đa dạng về loại hình thử thách. Các màn giải mã Caesar và Vigenere chủ yếu tập trung vào việc nhập khóa chính xác, mà không khai thác các khía cạnh quan trọng hơn như phân tích tần suất (đối với Caesar) hay tìm độ dài khóa và tấn công Kasiski (đối với Vigenere).

**Ảnh hưởng**: Người chơi có thể hoàn thành toàn bộ game trong một thời gian ngắn và cảm thấy nhàm chán do thiếu các yếu tố mới mẻ hoặc các bài toán mật mã sâu hơn để khám phá.

**Thiếu hệ thống quản lý người dùng nâng cao:**

**Mô tả**: Game hiện tại chỉ sử dụng Flask Session để lưu trữ trạng thái người chơi tạm thời trên phía client. Không có hệ thống tài khoản người dùng với chức năng đăng ký/đăng nhập, quản lý mật khẩu, hoặc lưu trữ tiến độ chơi vĩnh viễn trong cơ sở dữ liệu backend.

**Ảnh hưởng**: Người chơi không thể lưu trữ tiến độ vĩnh viễn qua các phiên truy cập, không thể chơi trên nhiều thiết bị khác nhau, và không thể cạnh tranh điểm số trong một bảng xếp hạng chung, làm giảm tính xã hội và tái chơi của game.

**Chức năng gợi ý tĩnh và chưa tối ưu:**

**Mô tả**: Các gợi ý được định nghĩa tĩnh cho từng màn chơi và được hiển thị theo một trình tự cố định. Hiện tại, không có cơ chế động để tạo gợi ý dựa trên tiến độ của người chơi, cũng như không có hệ thống "tiền tệ" trong game để người chơi phải "mua" gợi ý bằng điểm số hoặc bằng một tài nguyên ảo nào đó.

**Ảnh hưởng**: Gợi ý có thể trở nên quá dễ dàng hoặc không đủ hữu ích trong các tình huống người chơi thực sự gặp khó khăn. Việc thiếu yếu tố quản lý tài nguyên (tiền tệ) cũng làm mất đi một khía cạnh chiến thuật tiềm năng.

#### **5.2.2. Những khó khăn gặp phải trong quá trình phát triển**

**Hạn chế về thời gian và tài nguyên phát triển:**

**Mô tả**: Dự án được thực hiện trong một khung thời gian và với nguồn lực hạn chế, điển hình của một dự án cá nhân hoặc sinh viên. Điều này ảnh hưởng trực tiếp đến khả năng triển khai các tính năng phức tạp, đầu tư vào đồ họa cao cấp, hoặc thực hiện kiểm thử mở rộng.

**Ảnh hưởng**: Buộc phải ưu tiên các chức năng cốt lõi và chấp nhận một số đơn giản hóa (ví dụ: giả lập thuật toán mật mã hiện đại), dẫn đến các hạn chế đã nêu.

**Độ phức tạp của việc triển khai thuật toán mật mã hiện đại:**

**Mô tả**: Việc lập trình và triển khai đầy đủ các thuật toán mật mã phức tạp như AES và RSA từ đầu, bao gồm cả các phép toán số học lớn và cấu trúc dữ liệu chuyên biệt, là một thách thức lớn. Nó đòi hỏi kiến thức sâu về toán học, lý thuyết số, và kỹ thuật lập trình mật mã.

**Ảnh hưởng**: Do độ phức tạp và yêu cầu kiến thức chuyên sâu, dự án đã phải lựa chọn phương án "giả lập" cho các thuật toán này, tập trung vào việc kiểm tra khóa đáp án thay vì thực hiện quá trình mã hóa/giải mã thực sự.

**Thách thức cân bằng giữa tính giáo dục và tính giải trí:**

**Mô tả**: Một khó khăn cố hữu trong việc phát triển game giáo dục là làm thế nào để truyền tải kiến thức học thuật (thường khô khan) một cách hấp dẫn và dễ hiểu, mà không làm mất đi tính chính xác khoa học.

**Ảnh hưởng**: Đôi khi, để duy trì tính giải trí và khả năng tiếp cận, một số chi tiết kỹ thuật sâu sắc của mật mã học đã phải được lược bỏ hoặc đơn giản hóa quá mức.

**Hạn chế về kiến thức và kinh nghiệm chuyên môn:**

**Mô tả**: Người phát triển dự án có thể có những giới hạn về kiến thức chuyên sâu trong cả lĩnh vực thiết kế game (game design patterns, UI/UX nâng cao) và các khía cạnh chuyên sâu của mật mã học ứng dụng.

### 5.3. Đề xuất Cải tiến và Hướng phát triển trong Tương lai

Dựa trên những phân tích về đặc điểm của các thuật toán và các hạn chế hiện tại của dự án, phần này sẽ trình bày các đề xuất cụ thể nhằm nâng cao chất lượng, giá trị giáo dục, và trải nghiệm tổng thể của game "Giải Mã Kho Báu".

#### **5.3.1. Cải tiến về nội dung và tính giáo dục**

**Phát triển mô phỏng sâu hơn cho AES và RSA:**

**Đề xuất**: Thay vì chỉ kiểm tra khóa đáp án đơn giản, nên phát triển các mô-đun nhỏ hoặc các mini-game tập trung vào việc minh họa các khái niệm cốt lõi của AES và RSA.

**Với AES**: Có thể giới thiệu một cách trực quan các bước cơ bản như SubBytes (thay thế byte), ShiftRows (dịch chuyển hàng), MixColumns (trộn cột) và AddRoundKey (thêm khóa vòng) thông qua các ví dụ đơn giản hoặc hoạt ảnh tương tác. Người chơi không cần thực hiện tính toán phức tạp, nhưng có thể "xem" và "thao tác" trên một tập dữ liệu nhỏ để hiểu quy trình.

**Với RSA**: Minh họa các khái niệm về cặp khóa công khai/riêng tư, quá trình tạo khóa (tìm số nguyên tố, tính ϕ(n)), và các phép toán modulo lũy thừa cơ bản (với các số nhỏ) để mã hóa/giải mã.

**Lợi ích**: Nâng cao đáng kể giá trị giáo dục, giúp người chơi không chỉ biết tên mà còn có cái nhìn sơ lược về nguyên lý hoạt động của các thuật toán mật mã hiện đại.

**Mở rộng bộ sưu tập thuật toán mật mã:**

**Đề xuất**: Bổ sung thêm các màn chơi mới giới thiệu các thuật toán mật mã khác nhau, bao gồm cả cổ điển và hiện đại, cũng như các khái niệm liên quan. Ví dụ:

**Mật mã thay thế đơn giản (Simple Substitution Cipher)**: Giới thiệu khái niệm bảng chữ cái được xáo trộn.

**Mã hóa theo cột (Transposition Cipher)**: Giới thiệu cách thay đổi vị trí ký tự.

**Hàm băm (Hash Functions)**: Giới thiệu về tính toàn vẹn dữ liệu.

**Chữ ký số (Digital Signatures)**: Giải thích về xác thực nguồn gốc.

**Lợi ích**: Làm phong phú nội dung học tập, cung cấp cái nhìn toàn diện hơn về lĩnh vực mật mã học.

**Tích hợp các bài toán phân tích mật mã thực tế:**

**Đề xuất**: Đối với các mật mã cổ điển như Caesar và Vigenere, thay vì chỉ cung cấp khóa, hãy thiết kế các thử thách yêu cầu người chơi tự suy luận hoặc áp dụng các kỹ thuật phá mã cơ bản

**Caesar**: Thêm thử thách yêu cầu phân tích tần suất chữ cái (hiển thị biểu đồ tần suất hoặc gợi ý về các chữ cái phổ biến nhất)

**Vigenere**: Giới thiệu các gợi ý hoặc công cụ đơn giản để giúp người chơi tìm ra độ dài khóa hoặc thực hiện phân tích tần suất trên các phần văn bản.

**Lợi ích**: Tăng tính thử thách và giáo dục, giúp người chơi không chỉ giải mã mà còn hiểu được điểm yếu và cách thức các mật mã này bị phá giải.

**Phát triển thư viện kiến thức tích hợp:**

**Đề xuất**: Xây dựng một phần "Thư viện Mật mã" trong game, nơi người chơi có thể truy cập các tài liệu, bài viết ngắn gọn, hoặc video giải thích chi tiết về lịch sử, nguyên lý, và ứng dụng của từng thuật toán sau khi họ đã hoàn thành màn chơi tương ứng

**Lợi ích**: Biến game thành một công cụ học tập toàn diện, khuyến khích người chơi tìm hiểu sâu hơn ngoài các thử thách trong game.

#### **5.3.2. Cải tiến về trải nghiệm người dùng và tính năng game**

**Nâng cấp toàn diện giao diện và hiệu ứng:**

**Đề xuất**: Đầu tư vào thiết kế UI/UX với đồ họa tinh xảo hơn, bổ sung các hiệu ứng chuyển động mượt mà (transitions and animations) giữa các màn hình và khi người chơi tương tác. Thêm các hiệu ứng âm thanh cho các sự kiện quan trọng (giải mã thành công, thất bại, nhận gợi ý, hoàn thành level) và một nhạc nền nhẹ nhàng, phù hợp với không khí game.

**Lợi ích**: Nâng cao đáng kể tính thẩm mỹ, sự hấp dẫn, và tính nhập vai, giúp trải nghiệm chơi game trở nên thú vị và lôi cuốn hơn.

**Cải thiện hệ thống phản hồi người dùng:**

**Đề xuất**: Thay thế các thông báo chung chung bằng các thông báo cụ thể và mang tính hướng dẫn hơn. Ví dụ, nếu người chơi nhập ký tự không hợp lệ cho khóa Caesar, hệ thống có thể thông báo "Khóa Caesar phải là một số nguyên từ 1 đến 25!". Đồng thời, cung cấp các gợi ý cụ thể hơn trong quá trình chơi game khi người chơi gặp khó khăn

**Lợi ích**: Giúp người chơi dễ dàng nhận biết lỗi và hiểu cách khắc phục, cải thiện tính khả dụng và giảm sự nản lòng.

**Triển khai hệ thống gợi ý động và cơ chế "tiền tệ" trong game:**

**Đề xuất**: Thay vì gợi ý tĩnh, hãy phát triển một hệ thống có khả năng cung cấp gợi ý dựa trên tiến độ và các lần thử của người chơi. Đồng thời, giới thiệu một hệ thống "điểm" hoặc "tiền tệ" ảo mà người chơi kiếm được khi giải mã thành công các màn chơi. Người chơi có thể sử dụng "tiền tệ" này để "mua" các gợi ý hoặc các công cụ hỗ trợ khác.

**Lợi ích**: Tăng yếu tố chiến thuật, khuyến khích người chơi quản lý tài nguyên và cân nhắc trước khi sử dụng gợi ý, đồng thời tạo ra một động lực mới để chơi game.

**Phát triển hệ thống bảng xếp hạng (Leaderboard):**

**Đề xuất**: Tích hợp một cơ sở dữ liệu đơn giản (ví dụ: SQLite) để lưu trữ điểm số của người chơi và tạo một bảng xếp hạng cục bộ hoặc toàn cầu. Bảng xếp hạng có thể hiển thị top người chơi với điểm số cao nhất.

**Lợi ích**: Kích thích tính cạnh tranh giữa những người chơi, tạo động lực để họ cố gắng đạt được điểm số cao hơn và chơi lại game nhiều lần.

#### **5.3.3. Mở rộng về công nghệ và nền tảng**

**Chuyển đổi sang hệ thống cơ sở dữ liệu lâu dài (Persistent Database):**

**Đề xuất**: Thay thế việc lưu trữ trạng thái người chơi bằng Flask Session bằng một hệ thống cơ sở dữ liệu thực sự (ví dụ: SQLite cho các ứng dụng nhỏ hoặc PostgreSQL/MySQL cho các ứng dụng có quy mô lớn hơn). Điều này sẽ đi kèm với việc triển khai các chức năng quản lý tài khoản người dùng (đăng ký, đăng nhập, hồ sơ).

**Lợi ích**: Cho phép người chơi lưu trữ tiến độ vĩnh viễn, truy cập tài khoản từ nhiều thiết bị khác nhau, và là nền tảng vững chắc cho các tính năng phức tạp hơn như bảng xếp hạng, quản lý hồ sơ và tùy chỉnh.

**Phát triển API Backend mạnh mẽ hơn:**

**Đề xuất**: Nâng cấp backend Flask thành một API mạnh mẽ và có cấu trúc rõ ràng hơn. Tách biệt hoàn toàn logic kinh doanh của game khỏi giao diện người dùng

**Lợi ích**: Đơn giản hóa việc phát triển và bảo trì, tạo điều kiện thuận lợi cho việc tích hợp với nhiều giao diện người dùng khác nhau trong tương lai (ví dụ: ứng dụng di động).

**Xem xét các Framework Frontend hiện đại:**

**Đề xuất**: Mặc dù HTML/CSS/JavaScript thuần túy đã đáp ứng được yêu cầu hiện tại, việc chuyển sang sử dụng các framework JavaScript hiện đại như React, Vue.js, hoặc Angular có thể mang lại nhiều lợi ích.

**Lợi ích**: Tối ưu hóa hiệu suất giao diện người dùng, đơn giản hóa việc quản lý trạng thái phức tạp, và tạo điều kiện cho việc phát triển các thành phần UI phức tạp hơn một cách hiệu quả.

**Mở rộng sang nền tảng di động (Mobile Application):**

**Đề xuất**: Phát triển phiên bản ứng dụng di động (iOS/Android) bằng cách sử dụng các công nghệ đa nền tảng như React Native hoặc Flutter, hoặc phát triển native bằng Kotlin/Swift.

**Lợi ích**: Mở rộng đáng kể đối tượng người dùng, tận dụng sự phổ biến của thiết bị di động, và mang lại trải nghiệm tiện lợi hơn cho người chơi.

**KẾT LUẬN**

Báo cáo này đã trình bày chi tiết quá trình thiết kế, phát triển, thử nghiệm và đánh giá ứng dụng "Giải Mã Kho Báu" – một trò chơi tương tác được xây dựng nhằm mục đích giáo dục về mật mã học cơ bản. Dự án đã được triển khai thành công dưới dạng một ứng dụng web sử dụng framework Flask cho backend và kết hợp HTML, CSS, JavaScript cho frontend.

**Những kết quả đạt được của dự án:**

**Phát triển thành công ứng dụng web chức năng:** Ứng dụng đã được xây dựng và triển khai, hoạt động ổn định, cho phép người dùng tương tác thông qua giao diện web trực quan. Các tính năng cốt lõi như đăng ký người chơi, quản lý cấp độ, tính điểm và xử lý tương tác giải mã đều hoạt động hiệu quả.

**Mô phỏng hiệu quả các thuật toán mật mã cổ điển:** Game đã thành công trong việc mô phỏng và cung cấp trải nghiệm giải mã trực tiếp với các mật mã Caesar và Vigenere. Điều này giúp người chơi dễ dàng nắm bắt nguyên lý cơ bản của phép dịch chuyển và sử dụng từ khóa trong mật mã đối xứng.

**Giới thiệu khái niệm về mật mã hiện đại:** Mặc dù chỉ ở mức độ giả lập, game đã giới thiệu tên và vai trò của các thuật toán mật mã hiện đại như AES và RSA, giúp người chơi làm quen với khái niệm về các công nghệ bảo mật tiên tiến đang được sử dụng trong thế giới thực.

**Đảm bảo tính khả dụng và trực quan:** Giao diện người dùng được thiết kế tối giản, hiện đại, dễ hiểu và thân thiện, cho phép người chơi dễ dàng điều hướng và tương tác mà không cần hướng dẫn phức tạp. Hệ thống thông báo và gợi ý cũng góp phần nâng cao trải nghiệm người dùng.

**Áp dụng các nguyên tắc phát triển ứng dụng web cơ bản:** Dự án đã thể hiện việc áp dụng các kiến thức về phát triển web (Flask, Jinja2, HTML/CSS/JS) và quản lý phiên (Flask Session) để xây dựng một sản phẩm hoàn chỉnh

**Đóng góp và ý nghĩa của dự án:**

Dự án "Giải Mã Kho Báu" không chỉ là một minh chứng cho khả năng ứng dụng công nghệ thông tin vào lĩnh vực giáo dục mà còn mang ý nghĩa quan trọng trong việc

**Truyền tải kiến thức một cách tương tác:** Game cung cấp một nền tảng học tập chủ động và thú vị về mật mã học, giúp người học tiếp cận các khái niệm trừu tượng một cách dễ dàng và ghi nhớ lâu hơn thông qua trải nghiệm thực hành.

**Kích thích sự hứng thú với an ninh mạng:** Bằng cách biến việc học thành một trò chơi, dự án góp phần khơi gợi sự tò mò và niềm đam mê đối với lĩnh vực an ninh thông tin và mật mã học, đặc biệt đối với giới trẻ

**Là một nền tảng mở rộng trong tương lai:** Mặc dù còn một số hạn chế, dự án này đã tạo ra một nền tảng vững chắc. Từ đây, có thể phát triển thêm các tính năng phức tạp hơn, tích hợp nhiều thuật toán nâng cao, cải thiện trải nghiệm người dùng và mở rộng sang các nền tảng khác, hướng tới một công cụ giáo dục mật mã toàn diện hơn.

**Bài học sâu sắc và kinh nghiệm đúc kết từ dự án:**

Quá trình phát triển ứng dụng "Giải Mã Kho Báu" không chỉ là việc áp dụng các kiến thức lập trình mà còn là một trải nghiệm học tập đa chiều, mang lại những bài học chuyên sâu và kinh nghiệm quý báu, định hình tư duy phát triển phần mềm và năng lực giải quyết vấn đề. Những đúc kết này có thể được phân loại thành các khía cạnh chính sau:

**Thấu hiểu toàn diện về Kiến trúc Hệ thống Web và Luồng Tương tác Client-Server:**

**Kinh nghiệm:** Việc tự xây dựng từ đầu một ứng dụng web sử dụng Flask đã mang lại cái nhìn chân thực về cách các thành phần backend và frontend phối hợp. Chúng tôi đã trải nghiệm trực tiếp việc thiết lập server (Flask), định nghĩa các điểm cuối API (endpoints), xử lý logic nghiệp vụ, quản lý trạng thái người dùng (Flask Session), và đồng bộ dữ liệu giữa server với giao diện người dùng động thông qua AJAX.

**Bài học sâu sắc:** Điều này củng cố sự hiểu biết về tầm quan trọng của việc thiết kế kiến trúc rõ ràng, phân tách các module chức năng (MVC-like pattern), và tối ưu hóa luồng dữ liệu để đảm bảo hiệu suất và khả năng mở rộng của ứng dụng. Nó cũng làm nổi bật sự phức tạp tiềm ẩn trong việc duy trì tính nhất quán của dữ liệu và trạng thái giữa các phiên làm việc của người dùng.

**Nghệ thuật Đơn giản hóa và Truyền tải Khái niệm Khoa học Phức tạp:**

**Kinh nghiệm:** Một trong những thách thức lớn nhất là biến các nguyên lý mật mã học trừu tượng, đặc biệt là với các thuật toán hiện đại như AES và RSA, thành nội dung có thể tiếp cận và hấp dẫn trong một trò chơi. Chúng tôi đã phải lựa chọn phương án "giả lập" cho các thuật toán này thay vì triển khai toàn bộ sự phức tạp toán học của chúng.

**Bài học sâu sắc:** Bài học cốt lõi ở đây là nghệ thuật của sự đơn giản hóa có chủ đích. Để truyền tải kiến thức hiệu quả cho đối tượng không chuyên, đôi khi cần phải hy sinh một phần tính chân thực kỹ thuật để tập trung vào việc giới thiệu khái niệm và kích thích sự tò mò. Việc này đòi hỏi khả năng phân tích sâu sắc mục tiêu giáo dục, đối tượng người dùng, và tìm ra phương pháp trực quan hóa phù hợp nhất, thay vì cố gắng truyền đạt mọi chi tiết kỹ thuật ngay lập tức. Nó cũng rèn luyện kỹ năng "biên kịch" thông tin một cách chiến lược.

**Khẳng định Vai trò then chốt của Trải nghiệm Người dùng (UX) trong Sản phẩm Giáo dục:**

**Kinh nghiệm:** Mặc dù dự án không tập trung vào đồ họa cao cấp, nhưng việc chú trọng vào một giao diện người dùng (UI) trực quan, thiết kế tối giản, và luồng tương tác logic đã đóng vai trò quan trọng trong việc duy trì sự gắn kết của người chơi. Các phản hồi tức thì và rõ ràng (ví dụ: thông báo toast) cũng góp phần cải thiện đáng kể trải nghiệm.

**Bài học sâu sắc:** Điều này tái khẳng định rằng UX không chỉ dành cho các ứng dụng giải trí thương mại. Trong bối cảnh giáo dục, một trải nghiệm người dùng mượt mà và thân thiện là yếu tố sống còn để người học có thể tập trung vào nội dung kiến thức mà không bị phân tâm bởi sự phức tạp của giao diện. Nó nhấn mạnh tầm quan trọng của việc đặt người dùng vào trung tâm của quá trình thiết kế, ngay cả trong các dự án có quy mô khiêm tốn.

**Phát triển Năng lực Giải quyết Vấn đề và Gỡ lỗi Hệ thống có Hệ thống:**

**Kinh nghiệm:** Quá trình phát triển không tránh khỏi những lỗi kỹ thuật, từ cú pháp, logic ứng dụng, đến các vấn đề liên quan đến giao tiếp client-server. Việc phải liên tục tìm hiểu, phân tích mã nguồn và sử dụng các công cụ gỡ lỗi đã trở thành một phần không thể thiếu.

**Bài học sâu sắc:** Đây là cơ hội để rèn luyện tư duy phản biện và kỹ năng gỡ lỗi một cách có hệ thống. Bài học là không chỉ tìm cách khắc phục triệu chứng mà phải truy tìm nguyên nhân gốc rễ của vấn đề. Khả năng đọc hiểu thông báo lỗi, sử dụng công cụ DevTools của trình duyệt, và tách biệt các phần của hệ thống để kiểm tra độc lập là những kỹ năng được mài dũa đáng kể.

**Chiến lược Quản lý Dự án và Tối ưu hóa Tài nguyên:**

**Kinh nghiệm:** Với vai trò là một dự án với nguồn lực và thời gian hạn chế, việc lập kế hoạch chi tiết, phân chia nhiệm vụ, và linh hoạt điều chỉnh phạm vi dự án là rất quan trọng. Chúng tôi đã học cách ưu tiên các tính năng cốt lõi (MVP - Minimum Viable Product) và chấp nhận một số đơn giản hóa để đảm bảo dự án có thể hoàn thành trong khuôn khổ cho phép.

**Bài học sâu sắc:** Kinh nghiệm này cung cấp cái nhìn thực tế về quy trình quản lý dự án agile ở quy mô nhỏ. Nó dạy về tầm quan trọng của việc ước lượng công sức, xác định rủi ro tiềm ẩn, và đưa ra quyết định thực dụng khi đối mặt với các ràng buộc. Khả năng thích nghi và xoay chuyển kế hoạch là chìa khóa để đưa một ý tưởng thành sản phẩm thực tế.

**Đánh giá Sâu sắc về Công nghệ và Lựa chọn Framework:**

**Kinh nghiệm:** Việc sử dụng Flask đã chứng minh đây là một framework mạnh mẽ và linh hoạt cho việc phát triển nhanh các ứng dụng web quy mô nhỏ đến trung bình. Đồng thời, việc không sử dụng các framework frontend phức tạp cũng mang lại kinh nghiệm trực tiếp hơn với JavaScript, DOM manipulation, và CSS.

**Bài học sâu sắc:** Dự án giúp hình thành khả năng đánh giá khách quan về các công cụ và framework. Nó không chỉ là việc học cách sử dụng một công nghệ, mà còn là việc hiểu khi nào và tại sao một công nghệ cụ thể lại phù hợp hơn cho một loại dự án nhất định, dựa trên các yếu tố như tốc độ phát triển, khả năng mở rộng, và cộng đồng hỗ trợ.

Tóm lại, "Giải Mã Kho Báu" không chỉ là một sản phẩm phần mềm giáo dục hoàn chỉnh mà còn là một khóa học thực tế chuyên sâu, trang bị cho chúng tôi những kỹ năng và kinh nghiệm đa chiều. Từ kỹ thuật lập trình vững chắc, thiết kế sản phẩm lấy người dùng làm trung tâm, đến tư duy giải quyết vấn đề và quản lý dự án hiệu quả, tất cả đều là nền tảng vững chắc cho con đường phát triển sự nghiệp trong ngành công nghệ thông tin.

DANH MỤC THAM KHẢO

**1. Tài liệu trực tuyến và Website:**

**Mật mã học:**

Wikipedia: Các bài viết về Caesar Cipher, Vigenere Cipher, AES, RSA. [Liên kết nếu có thể]

OWASP Foundation: Tài liệu về các lỗ hổng bảo mật web và nguyên tắc mã hóa. [Liên kết]

**Công nghệ Phát triển Web:**

Tài liệu chính thức của Flask: [Liên kết: [https://flask.palletsprojects.com/en/latest/](https://flask.palletsprojects.com/en/latest/" \t "_blank)]MDN Web Docs (Mozilla Developer Network): Tài liệu về HTML, CSS, JavaScript. [Liên kết: [https://developer.mozilla.org/en-US/docs/Web](https://developer.mozilla.org/en-US/docs/Web" \t "_blank)]

Bootstrap (nếu sử dụng): [Liên kết: [https://getbootstrap.com/](https://getbootstrap.com/" \t "_blank)]

(Thêm các nguồn tài liệu công nghệ khác như W3Schools, Stack Overflow, v.v.

**2. Mã nguồn mở và Thư viện:**

Flask: Python web framework. [Liên kết GitHub hoặc PyPI nếu có thể]

Jinja2: Template engine cho Python.

(Liệt kê các thư viện Python, JavaScript hoặc các dự án mã nguồn mở khác mà bạn đã sử dụng hoặc tham khảo cấu trúc)