BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO BÀI TẬP LỚN LẬP TRÌNH JAVA**

**CÀI ĐẶT THUẬT TOÁN MÃ HÓA AES, HÀM BĂM SHA-256 VÀ ỨNG DỤNG**

**Sinh viên thực hiện: Trần Mai Ngọc Duy**

**Mã số sinh viên: 65130650**

**Giảng viên hướng dẫn: TS.Phạm Văn Nam**

**Học kỳ 2, năm học 2024-2025**

Khánh Hòa – 2025

**MỤC LỤC**

**DANH MỤC HÌNH ẢNH**

**Chương 1. GIỚI THIỆU ĐỀ TÀI**

**1.1 LÝ DO CHỌN ĐỀ TÀI**

Là một sinh viên năm hai ngành công nghệ thông tin, tôi nhận thức rõ tầm quan trọng của bảo mật dữ liệu trong thời đại số hóa hiện nay. Những sự cố rò rỉ thông tin, tấn công mạng, và đánh cắp dữ liệu ngày càng phổ biến, gây ra những hậu quả nghiêm trọng không chỉ đối với cá nhân mà còn cho các tổ chức, doanh nghiệp lớn. Chính vì vậy, tôi thấy việc nghiên cứu về các thuật toán bảo mật như AES (Advanced Encryption Standard) và SHA-256 không chỉ cần thiết mà còn rất thú vị. Đây là cơ hội để tôi khám phá sâu hơn lĩnh vực bảo mật thông tin, một trong những mảng công nghệ thông tin đang phát triển mạnh mẽ.

Thuật toán AES đóng vai trò quan trọng trong việc mã hóa dữ liệu, đảm bảo rằng thông tin được bảo vệ an toàn trước khi truyền tải hoặc lưu trữ. Trong khi đó, hàm băm SHA-256 giúp bảo toàn tính toàn vẹn của dữ liệu, đảm bảo rằng dữ liệu không bị chỉnh sửa hoặc giả mạo. Việc kết hợp hai thuật toán này tạo ra một giải pháp mạnh mẽ cho nhiều ứng dụng bảo mật thực tế như bảo mật giao tiếp mạng, lưu trữ thông tin nhạy cảm, và kiểm tra tính toàn vẹn của file.

Tôi chọn đề tài này vì nó không chỉ cho phép tôi áp dụng lý thuyết đã học vào thực tế mà còn giúp tôi phát triển kỹ năng lập trình và tư duy giải quyết vấn đề. Qua việc triển khai các thuật toán và xây dựng ứng dụng, tôi có cơ hội rèn luyện khả năng làm việc với các công nghệ bảo mật hiện đại. Hơn nữa, tôi tin rằng đây là một bước đệm quan trọng để tôi tiến xa hơn trong việc theo đuổi sự nghiệp liên quan đến lĩnh vực an ninh mạng.

Ngoài ra, tôi cũng rất hào hứng với việc khám phá các ứng dụng thực tế của AES và SHA-256, chẳng hạn như bảo mật giao dịch tài chính, xác thực dữ liệu trong blockchain, và bảo vệ dữ liệu cá nhân. Tôi tin rằng những kiến thức và kỹ năng tích lũy được trong quá trình thực hiện đề tài này sẽ không chỉ hỗ trợ tốt cho việc học tập hiện tại mà còn là nền tảng vững chắc cho tương lai.

Với những lý do này, tôi mong muốn thông qua đề tài “**Cài đặt thuật toán mã hóa AES, hàm băm SHA-256 và ứng dụng**”, tôi có thể nâng cao kiến thức, thử thách bản thân, và chuẩn bị hành trang để tiến xa hơn trong ngành công nghệ thông tin.

**1.2 MỤC TIÊU NGHIÊN CỨU**

**1.2.1 Mục tiêu đề tài**

* Cài đặt và triển khai thuật toán:

+ Thực hiện cài đặt thuật toán mã hóa AES và hàm băm SHA-256 bằng ngôn ngữ lập trình Java.

+ Đảm bảo quá trình mã hóa và băm dữ liệu hoạt động chính xác và hiệu quả.

* Hiểu rõ nguyên lý hoạt động:

+ Nghiên cứu sâu về cấu trúc và cách thức hoạt động của các thuật toán AES và SHA-256.

+ Phân tích các bước xử lý dữ liệu trong từng thuật toán để hiểu rõ cơ chế bảo mật.

* Xây dựng ứng dụng minh họa:

+ Phát triển một ứng dụng đơn giản, có giao diện trực quan để minh họa cách các thuật toán được sử dụng trong việc mã hóa và kiểm tra tính toàn vẹn của dữ liệu.

+ Đảm bảo ứng dụng thể hiện rõ ràng kết quả hoạt động của các thuật toán.

* Đánh giá hiệu suất:

+ Đánh giá hiệu suất của các thuật toán trong việc xử lý dữ liệu, từ tốc độ đến mức độ an toàn.

**1.2.2 Đối tượng và phạm vi nghiên cứu**

* Đối tượng nghiên cứu: Thuật toán mã hóa AES, hàm băm SHA-256, và ứng dụng của chúng trong việc bảo mật dữ liệu và đảm bảo tính toàn vẹn thông tin.
* Phạm vi nghiên cứu: Việc tìm hiểu lý thuyết, cài đặt và minh họa thuật toán mã hóa AES, hàm băm SHA-256. Đề tài sẽ triển khai trong môi trường lập trình Java và xây dựng ứng dụng minh họa nhỏ, với phạm vi dữ liệu đầu vào và đầu ra cơ bản, không đi sâu vào các hệ thống lớn hoặc ứng dụng phức tạp.

**1.2.3 Công nghệ sử dụng**

* Ngôn ngữ lập trình: Java
* Thư viện hỗ trợ GUI: java.swing
* Công cụ lập trình: IntelliJ IDEA

**CHƯƠNG 2. CƠ SỞ LÝ THUYẾT**

**2.1 MÃ HÓA ĐỐI XỨNG AES**

**2.1.1 Giới thiệu**

Mã hóa AES (Advanced Encryption Standard) là một thuật toán mã hóa đối xứng được tiêu chuẩn hóa bởi Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST) vào năm 2001. AES được phát triển dựa trên thuật toán Rijndael do hai nhà mật mã học người Bỉ là Vincent Rijmen và Joan Daemen đề xuất.

Với đặc điểm mã hóa đối xứng, AES sử dụng cùng một khóa cho cả quá trình mã hóa và giải mã. Điều này giúp cho AES có tốc độ xử lý nhanh, hiệu quả cao và phù hợp với nhiều ứng dụng thực tế như bảo mật dữ liệu, truyền thông an toàn, và các hệ thống nhúng.

AES hiện là một trong những chuẩn mã hóa phổ biến và an toàn nhất, được sử dụng rộng rãi trong các hệ thống chính phủ, doanh nghiệp và cá nhân trên toàn cầu. Thuật toán này hỗ trợ các độ dài khóa 128 bit, 192 bit và 256 bit, tương ứng với số vòng lặp xử lý là 10, 12 và 14 vòng.

Nhờ vào cấu trúc linh hoạt, khả năng bảo mật cao, cùng với khả năng chống lại hầu hết các phương pháp tấn công mật mã hiện đại, AES đã thay thế thuật toán DES (Data Encryption Standard) trước đó và trở thành tiêu chuẩn mã hóa dữ liệu chính thức tại nhiều tổ chức.

**2.1.2 Nguyên lý hoạt động**

Thuật toán AES hoạt động dựa trên cơ chế mã hóa khối (block cipher), trong đó dữ liệu đầu vào được chia thành các khối có độ dài cố định là 128 bit (tương đương 16 byte). Quá trình mã hóa diễn ra thông qua nhiều vòng lặp (round), mỗi vòng bao gồm một tập hợp các phép biến đổi toán học nhằm đảm bảo độ an toàn và tính không thể đảo ngược của dữ liệu.

*a) Độ dài khóa và số vòng lặp*

**AES-128:** Khóa dài 128 bit, có 10 vòng lặp.

**AES-192:** Khóa dài 192 bit, có 12 vòng lặp.

**AES-256:** Khóa dài 256 bit, có 14 vòng lặp.

*b) Các bước xử lý trong một vòng lặp AES*

Mỗi vòng lặp (trừ vòng đầu và vòng cuối) gồm bốn bước chính:

Bước 1: SubBytes

Mỗi byte trong khối dữ liệu được thay thế bằng một byte khác dựa vào bảng S-box phi tuyến. Bước này tạo ra tính phi tuyến cho thuật toán.

Bước 2: ShiftRows

Các hàng trong ma trận dữ liệu được dịch vòng sang trái:

Hàng thứ 1: giữ nguyên.

Hàng thứ 2: dịch trái 1 byte.

Hàng thứ 3: dịch trái 2 byte.

Hàng thứ 4: dịch trái 3 byte.

Bước 3: MixColumns

Mỗi cột được biến đổi thông qua phép nhân ma trận trong trường hữu hạn GF(28). Bước này giúp trộn dữ liệu giữa các byte trong một cột.

Bước 4: AddRoundKey

Kết quả sau ba bước trên sẽ được XOR với khóa con tương ứng. Các khóa con này được sinh ra từ khóa chính nhờ vào quá trình mở rộng khóa (Key Expansion).

*c) Quy trình tổng thể mã hóa thuật toán AES*

**Vòng khởi tạo:** AddRoundKey

**Các vòng chính:** SubBytes → ShiftRows → MixColumns → AddRoundKey

**Vòng cuối cùng:** SubBytes → ShiftRows → AddRoundKey (Không có bước Mixcolumns trong vòng cuối)

*d) Giải mã trong thuật toán AES*

Quá trình giải mã của AES bao gồm các bước ngược lại:

* InvShiftRows
* InvSubBytes
* InvMixColumns
* AddRoundKey

Các khóa con được sử dụng theo thứ tự ngược lại so với quá trình mã hóa.

**2.1.3 Cấu trúc thuật toán AES**

Thuật toán AES hoạt động trên một cấu trúc ma trận gọi là State, kích thước 4x4 byte (tương đương 128 bit). Dữ liệu đầu vào (plaintext) được chia thành các khối 128 bit, và mỗi khối sẽ được biểu diễn dưới dạng ma trận State để thực hiện các phép biến đổi qua các vòng lặp.

*a) Cấu trúc khối dữ liệu*

Mỗi khối dữ liệu đầu vào được tổ chức như sau:

A graph of numbers and symbols

AI-generated content may be incorrect.

**Hinh 2.1: Ma trận State trong thuật toán AES**

Trong đó Si,j là một byte (8 bit) tại vị trí hàng i, cột j trong ma trận.

*b) Quá trình mở rộng khóa (Key Expansion)*

Từ khóa ban đầu, thuật toán AES sinh ra nhiều khóa con (round key) để sử dụng cho từng vòng mã hóa. Tổng số khóa con cần tạo ra tùy vào độ dài khóa:

* AES-128: 10 + 1 = 11 round keys
* AES-192: 12 + 1 = 13 round keys
* AES-256: 14 + 1 = 15 round keys

Quá trình mở rộng khóa bao gồm các bước:

* Lấy khóa gốc chia thành các từ (word) 32 bit.
* Dựa vào các từ trước đó và hàm **SubWord**, **RotWord**, kết hợp với hằng số **Rcon** để tạo ra từ mới.
* Mỗi khóa con là tập hợp 4 từ (tổng 128 bit).

*c) Bảng S-Box và Inverse S-Box*

*A table of numbers and letters

AI-generated content may be incorrect.*

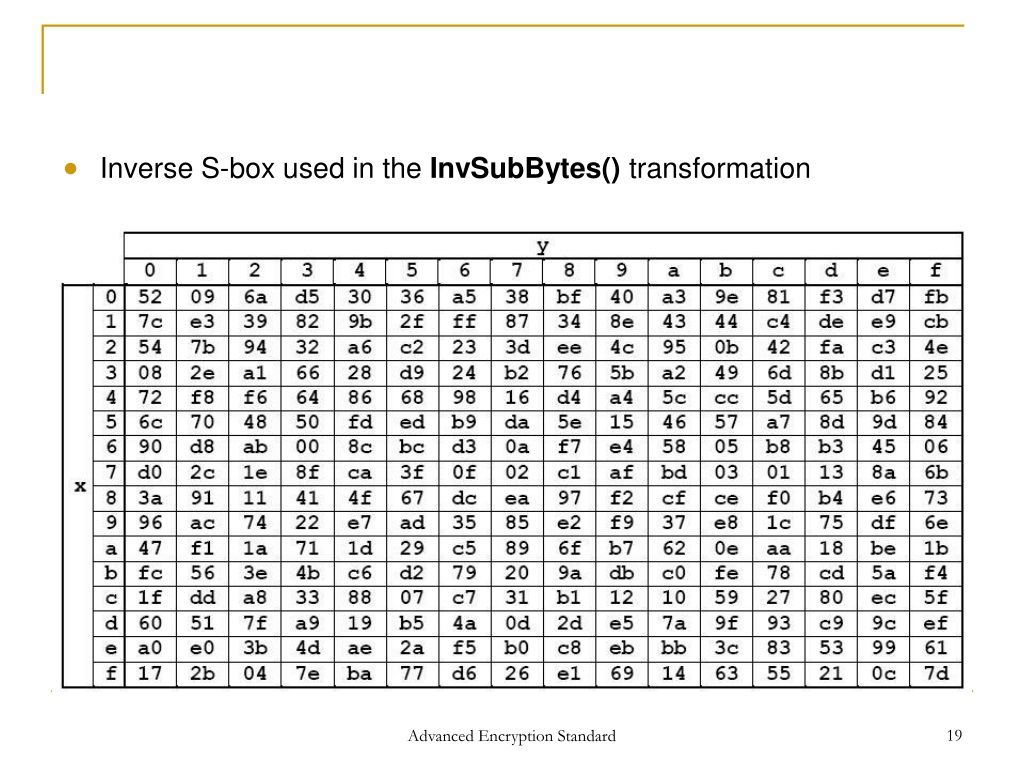
**Hình 2.2** - **Bảng tra cứu 256 giá trị dùng trong bước SubBytes**

S-Box (Substitution Box) là một bảng tra cứu gồm 256 giá trị, tương ứng với tất cả các giá trị có thể của 1 byte (từ 0x00 đến 0xFF).

Bảng này được sử dụng trong bước SubBytes() – một trong các bước biến đổi chính của thuật toán AES.

Mỗi byte trong ma trận trạng thái (State) sẽ được thay thế bởi một giá trị tương ứng trong bảng S-Box. Quá trình này tạo ra tính phi tuyến cho thuật toán, giúp chống lại các cuộc tấn công như phân tích tuyến tính và vi sai.

S-Box được xây dựng từ phép nghịch đảo trong trường hữu hạn GF(28), kết hợp với phép biến đổi tuyến tính (Affine Transformation).



**Hình 2.3** - **Được sử dụng trong quá trình giải mã, là bảng ngược của S-Box**

Inverse S-Box là bảng đảo ngược của S-Box, chứa các giá trị được sử dụng để hoàn tác quá trình SubBytes trong giai đoạn giải mã.

Được sử dụng trong bước InvSubBytes(), khi mỗi byte của bản mã được thay thế bằng giá trị tương ứng trong bảng Inverse S-Box.

Bảng này giúp khôi phục lại dữ liệu gốc từ bản mã trong quá trình giải mã AES.

**2.1.4 Độ an toàn của thuật toán AES**

Thuật toán AES (Advanced Encryption Standard) được đánh giá là một trong những thuật toán mã hóa đối xứng an toàn và hiệu quả nhất hiện nay. Một trong những yếu tố cốt lõi tạo nên độ an toàn của AES là kích thước khóa lớn, với ba mức độ tương ứng là 128 bit, 192 bit và 256 bit. Việc thử tất cả các khóa khả thi (brute-force) với AES-128 đã là bất khả thi với công nghệ hiện tại, bởi số lượng khóa có thể lên đến 2¹²⁸, tức khoảng 3.4 × 10³⁸ khóa. Với AES-256, độ an toàn còn tăng lên gấp nhiều lần với không gian khóa là 2²⁵⁶.

Bên cạnh đó, thiết kế của AES theo mô hình mạng hoán vị-thay thế (Substitution–Permutation Network) giúp tăng cường độ phức tạp và khả năng khuếch tán dữ liệu. Các bước xử lý như SubBytes, ShiftRows, MixColumns và AddRoundKey phối hợp chặt chẽ để đảm bảo rằng chỉ một thay đổi nhỏ ở đầu vào cũng làm thay đổi toàn bộ kết quả mã hóa (hiệu ứng avalanche). Đặc biệt, bảng S-box trong AES được thiết kế dựa trên các phép biến đổi phi tuyến học nhằm tăng khả năng kháng cự trước các dạng tấn công vi sai (differential cryptanalysis) và tấn công tuyến tính (linear cryptanalysis).

AES cũng đã được phân tích sâu rộng trong cộng đồng mật mã học và chưa phát hiện lỗ hổng nghiêm trọng nào tính đến thời điểm hiện tại. Thuật toán này đã được Viện Tiêu chuẩn và Công nghệ Hoa Kỳ (NIST) chuẩn hóa vào năm 2001 và hiện đang được sử dụng rộng rãi trong các lĩnh vực yêu cầu mức độ bảo mật cao như chính phủ, quốc phòng, ngân hàng và hệ thống thông tin công nghiệp.

Với không gian khóa khổng lồ, thiết kế bảo mật chặt chẽ và sự kiểm định rộng rãi từ các tổ chức uy tín, AES vẫn là một lựa chọn hàng đầu trong việc bảo vệ dữ liệu trước các mối đe dọa hiện đại.

**2.2 THUẬT TOÁN BĂM SHA-256**

**2.2.1 Giới thiệu**

SHA-256 (Secure Hash Algorithm 256-bit) là một thuật toán băm mật mã thuộc họ SHA-2, được phát triển bởi Cơ quan An ninh Quốc gia Hoa Kỳ (NSA) và được Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST) công bố như là một phần của tiêu chuẩn FIPS PUB 180-4. Thuật toán SHA-256 được thiết kế để tạo ra một giá trị băm (digest) cố định có độ dài 256 bit từ một thông điệp đầu vào có độ dài tùy ý. Đầu ra này được gọi là “mã băm” hoặc “giá trị băm”, đóng vai trò như một “dấu vân tay” số của dữ liệu gốc.

Hàm băm SHA-256 đóng vai trò quan trọng trong nhiều lĩnh vực của an ninh mạng và mật mã học hiện đại như xác thực dữ liệu, lưu trữ mật khẩu an toàn, kiểm tra toàn vẹn tệp tin, chữ ký số, giao thức SSL/TLS và đặc biệt là trong công nghệ blockchain. Điều quan trọng cần lưu ý là SHA-256 là một hàm băm một chiều, nghĩa là rất dễ để tính giá trị băm từ dữ liệu đầu vào, nhưng gần như không thể khôi phục được dữ liệu gốc từ giá trị băm đó. Ngoài ra, xác suất để hai thông điệp khác nhau cho cùng một giá trị băm là cực kỳ thấp (gọi là khả năng chống va chạm).

Một số đặc tính nổi bật của SHA-256 có thể kể đến:

**Tính xác định:** Cùng một đầu vào luôn cho ra một đầu ra giống nhau.

**Khó đảo ngược:** Không thể tìm được thông điệp ban đầu từ giá trị băm.

**Chống va chạm:** Xác suất hai đầu vào khác nhau tạo ra cùng một giá trị băm là cực kỳ nhỏ.

**Khả năng khuếch đại:** Chỉ cần thay đổi một bit trong đầu vào thì đầu ra thay đổi hoàn toàn, khiến việc dự đoán rất khó khăn.

SHA-256 sử dụng cấu trúc **Merkle–Damgård** và dựa trên nguyên lý **hàm nén** (compression function) để xử lý dữ liệu theo từng khối 512 bit. Sau mỗi vòng xử lý, dữ liệu được biến đổi thông qua một loạt các phép toán logic như dịch vòng, phép XOR, AND, cộng modulo 232, cùng với các hằng số được định nghĩa trước.

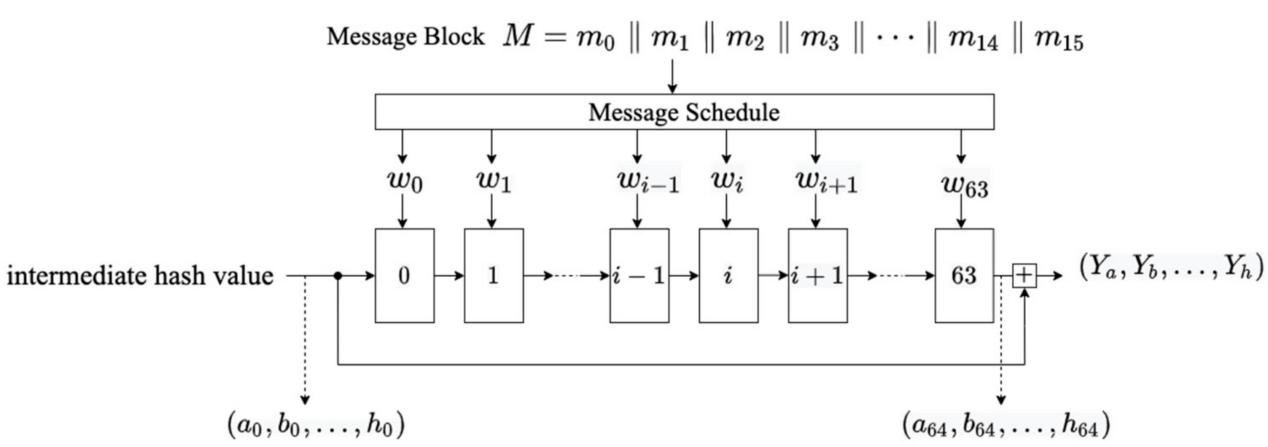
Hiện nay, SHA-256 vẫn được xem là an toàn trước hầu hết các kỹ thuật tấn công mật mã đã biết, và là lựa chọn tiêu chuẩn trong các hệ thống đòi hỏi bảo mật cao. Tuy nhiên, với sự phát triển nhanh chóng của công nghệ tính toán và các tiến bộ trong lĩnh vực điện toán lượng tử, việc nghiên cứu và chuyển sang các thuật toán băm thế hệ tiếp theo như SHA-3 cũng đang được quan tâm.

**2.2.2 Nguyên lý hoạt động của thuật toán SHA-256**

SHA-256 hoạt động dựa trên 2 bước là xử lý dữ liệu đầu vào và băm dữ liệu. Một số kí hiệu được sử dụng ở phần này bao gồm:

1. biểu diễn cho toán tử XOR.
2. biểu diễn cho toán tử AND.
3. biểu diễn cho toán tử NOT.
4. Phép cộng ở đây là cộng nhị phân modulo 232.

**2.2.2.1 *Xử lý dữ liệu đầu vào***



**Hình 2.4** - **Quá trình xử lý dữ liệu đầu vào của SHA-256**

Ban đầu, 16 từ đầu tiên w0,…,w15 được lấy trực tiếp từ các từ đầu vào m0,…, m15. Các từ còn lại w16,…,w63 được tạo ra bằng công thức lặp sử dụng các phép toán logic như xoay vòng (rotate), dịch phải (shift right), và XOR theo định nghĩa của chuẩn SHA-256.

Mục đích của bước này là tạo ra 64 từ w\_i được sử dụng ở mỗi vòng của hàm nén, nhằm đảm bảo rằng toàn bộ thông điệp ảnh hưởng sâu sắc đến giá trị băm cuối cùng (đặc tính khuếch đại).

Trước khi bắt đầu xử lý khối dữ liệu đầu vào, SHA-256 sử dụng một tập hợp các giá trị khởi tạo ban đầu (initial hash values), gồm 8 từ 32-bit, ký hiệu là a0,b0,..., h0. Đây là các hằng số được định nghĩa trước trong chuẩn SHA-2, thường được suy ra từ phần thập phân của các căn bậc hai của các số nguyên tố đầu tiên.

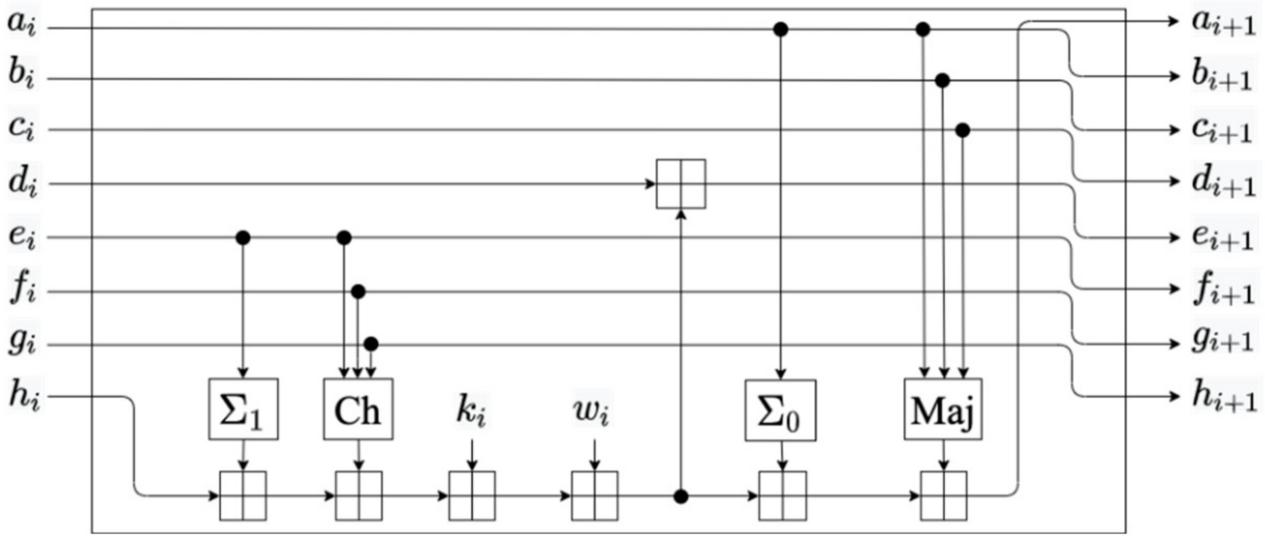
Khối chính của sơ đồ là 64 vòng xử lý (được biểu diễn từ w0​ đến w63​). Trong mỗi vòng, thuật toán sử dụng các từ thông điệp wiw\_iwi​ cùng với các hằng số vòng kj​, kết hợp với giá trị băm tạm thời hiện tại (các thanh ghi a, b, ..., h) để tính toán lại 8 giá trị băm mới.

Các phép toán được sử dụng bao gồm:

* Phép XOR
* Phép AND, OR
* Phép quay trái/quay phải (rotate)
* Hàm lựa chọn (Ch)
* Hàm đa số (Maj)
* Cộng modulo 232

Sau khi kết thúc 64 vòng, giá trị băm trung gian ban đầu (a0, b0,…, h0) sẽ được cộng với kết quả của vòng cuối cùng (a64, b64,…, h64) để tạo ra giá trị băm trung gian mới. Giá trị này sẽ được sử dụng làm đầu vào cho khối dữ liệu tiếp theo (nếu có), hoặc là kết quả cuối cùng nếu đây là khối cuối cùng.

**2.2.2.2 *Mã hóa dữ liệu***



**Hình 2.4 – Quá trình mã hóa dữ liệu của SHA-256**

Trong mỗi vòng i của thuật toán SHA-256, 8 thanh ghi tạm thời ai,bi,...,hi, đại diện cho giá trị băm hiện tại sẽ được cập nhật thông qua các phép toán logic, cộng modulo và hàm phi tuyến. Các phép toán này đảm bảo tính hỗn loạn và khuếch đại – hai tính chất cốt lõi trong các hàm băm mật mã.

**Các thành phần chính:**

**Σ0 và Σ1​:**

Là hai hàm hoán vị phi tuyến được định nghĩa như sau

Σ0​(x) = ROTR2(x) ⊕ ROTR13(x) ⊕ ROTR22(x)

Σ1​(x) = ROTR6(x) ⊕ ROTR11(x) ⊕ ROTR25(x)

**Ch (Choose function):**

Hàm chọn bit từ hai thanh ghi dựa trên giá trị điều kiện:

Ch(e,f,g) = (e ∧ f) ⊕ (¬e∧g)

→ Nếu e là 1 thì chọn bit từ f, ngược lại chọn từ g.

**Maj (Majority function)**:

Trả về giá trị chiếm đa số trong 3 bit:

Maj(a,b,c) = (a∧b) ⊕ (a∧c) ⊕ (b∧c)

**Ki:** Là hằng số vòng, được định nghĩa sẵn theo chuẩn SHA-256

**Wi:** Là từ thông điệp tại vòng thứ iii, được tạo ra từ bước “Message Schedule” đã mô tả ở **Hình 2.4**

***Quá trình tính toán***

Tại mỗi vòng i, ta thực hiện:

T1 = hi + Σ1(ei) + Ch(ei,fi,gi) + ki + wi

T2 = Σ0(ai) +Maj(ai,bi,ci)

Sau đó, các thanh ghi được cập nhật theo quy tắc:

hi+1 = gi, gi+1 = fi, fi+1 = ei, ei+1 = di+T1

di+1 = ci, ci+1 = bi, bi+1 = ai, ai+1 = T1 + T2

Sơ đồ trong hình thể hiện **một vòng cập nhật trạng thái nội bộ** trong thuật toán SHA-256. Đây là phần trung tâm của hàm nén, nơi mà các giá trị băm được biến đổi qua nhiều vòng để tăng tính hỗn loạn và đảm bảo rằng mỗi bit của thông điệp ảnh hưởng đến kết quả cuối cùng.

Thông qua các phép toán logic như **Ch**, **Maj**, và các hàm hoán vị Σ, kết hợp với giá trị từ message schedule wi​ và hằng số vòng ki​, thuật toán SHA-256 đạt được **tính không thể đảo ngược, tính chống va chạm và tính phân tán mạnh mẽ**, làm cho nó phù hợp để sử dụng trong bảo mật dữ liệu, xác thực và nhiều ứng dụng mật mã khác.

**2.2.3 Độ an toàn của thuật toán SHA-256**

SHA-256 (Secure Hash Algorithm 256-bit) được thiết kế bởi Cơ quan An ninh Quốc gia Hoa Kỳ (NSA) và được công bố như một phần của chuẩn FIPS PUB 180-4 do Viện Tiêu chuẩn và Công nghệ Hoa Kỳ (NIST) ban hành. Cho đến nay, SHA-256 vẫn được xem là một trong những hàm băm mật mã mạnh mẽ và an toàn nhất được sử dụng rộng rãi trong thực tiễn.

*Không gian đầu ra lớn (256 bit)*

SHA-256 sinh ra một giá trị băm dài 256 bit, tương đương 22562^{256}2256 khả năng giá trị đầu ra. Điều này khiến việc **tấn công vét cạn (brute-force)** để tìm đầu vào tương ứng gần như không khả thi với khả năng tính toán hiện tại, vì nó đòi hỏi một lượng tài nguyên tính toán vượt quá khả năng của tất cả các hệ thống hiện nay trong thời gian hợp lý.

*Khả năng chống va chạm (Collision Resistance)*

Một hàm băm được coi là an toàn nếu rất khó để tìm ra hai thông điệp khác nhau x ≠ y sao cho SHA256(x) = SHA256(y). Với không gian 256 bit, theo phân tích lý thuyết, để tìm va chạm bằng tấn công sinh nhật (birthday attack), kẻ tấn công cần thực hiện khoảng 2128 phép tính băm một con số cực kỳ lớn và chưa thực tế.

*Khả năng chống tìm ngược (Preimage Resistance)*

SHA-256 cũng đảm bảo rằng **không thể tìm được đầu vào** nếu chỉ biết giá trị băm h = SHA256(m). Việc tìm ra mmm sao cho SHA256(m) = h hiện vẫn chưa có thuật toán khả thi nào đạt hiệu quả tốt hơn brute-force, vốn yêu cầu trung bình 2256 bước thử.

*Khả năng chống tìm ngược lần hai (Second Preimage Resistance)*

SHA-256 còn ngăn chặn việc tìm một thông điệp khác m′ sao cho SHA256(m) = SHA256(m’) dù đã biết trước một thông điệp mmm. Để phá vỡ đặc tính này cũng cần trung bình 2256 phép tính tương đương với mức độ an toàn rất cao.

*Chưa bị phá bởi tấn công thực tiễn*

Dù có một số nghiên cứu lý thuyết cố gắng rút ngắn quá trình phá SHA-2 (bao gồm SHA-256), **cho đến nay chưa có bất kỳ cuộc tấn công nào thành công trên SHA-256 toàn bộ**. Các biến thể có chiều dài ngắn hơn như SHA-1 đã bị phá bởi va chạm, nhưng SHA-256 vẫn chưa có lỗ hổng thực tiễn.

*Được kiểm định và sử dụng rộng rãi*

SHA-256 được dùng trong nhiều ứng dụng bảo mật như xác thực mật khẩu, chữ ký số, blockchain (ví dụ: Bitcoin), và nhiều giao thức truyền thông bảo mật (TLS/SSL). Tính phổ biến và được chấp nhận rộng rãi là minh chứng cho mức độ tin cậy và an toàn cao của thuật toán.

**Chương 3. PHÂN TÍCH YÊU CẦU**

**3.1 YÊU CẦU CHỨC NĂNG**

**3.1.1 Chức năng mã hóa và giải mã**

1. Mã hóa và giải mã văn bản sử dụng thuật toán AES
2. Mã hóa và giải mã file sử dụng thuật toán AES
3. Hỗ trợ chế độ CBC (Cipher Block Chaining) với IV (Initialization Vector)

**3.1.2 Quản lý khóa và IV**

1. Cho phép nhập khóa thủ công (32 ký tự hex)
2. Cho phép nhập IV thủ công (32 ký tự hex)
3. Tự động tạo khóa ngẫu nhiên
4. Tự động tạo IV ngẫu nhiên

**3.1.3 Xử lý file**

1. Cho phép chọn file để mã hóa/giải mã
2. Lưu file đã mã hóa với phần mở rộng .encrypted
3. Hỗ trợ ghi đè file khi giải mã
4. Lưu kết quả mã hóa/giải mã vào file mới

**3.1.4 Giao diện người dùng**

1. Chuyển đổi giữa chế độ mã hóa văn bản và file
2. Hiển thị thời gian thực hiện mã hóa/giải mã
3. Hiển thị thời gian hiện tại khi thực hiện
4. Hiển thị kết quả mã hóa/giải mã
5. Xóa nội dung đầu vào và kết quả
6. Thoát chương trình

**3.2 YÊU CẦU PHI CHỨC NĂNG**

**3.2.1 Bảo mật**

1. Sử dụng thuật toán AES - một thuật toán mã hóa đối xứng mạnh
2. Sử dụng IV ngẫu nhiên để tăng tính bảo mật
3. Khóa được tạo ngẫu nhiên an toàn (SecureRandom)
4. Kiểm tra tính hợp lệ của khóa và IV

**3.2.2 Hiệu suất**

1. Đo lường và hiển thị thời gian thực hiện mã hóa/giải mã
2. Lưu trữ lịch sử thời gian thực hiện
3. Tính toán thời gian trung bình

**3.2.3 Giao diện người dùng**

1. Thiết kế giao diện hiện đại với màu sắc hài hòa
2. Sử dụng font chữ Segoe UI cho dễ đọc
3. Các nút có hiệu ứng hover
4. Icon trực quan cho các chức năng
5. Tooltip hướng dẫn người dùng
6. Thông báo lỗi và cảnh báo rõ ràng

**3.2.4 Khả năng sử dụng**

1. Giao diện trực quan, dễ sử dụng
2. Hỗ trợ cả chế độ văn bản và file
3. Kiểm tra đầu vào để tránh lỗi
4. Thông báo rõ ràng khi có lỗi
5. Xác nhận trước khi ghi đè file

**3.2.5 Khả năng bảo trì**

1. Code được tổ chức theo cấu trúc rõ ràng
2. Sử dụng các hằng số cho màu sắc và cấu hình
3. Tách biệt logic xử lý và giao diện
4. Comment code đầy đủ

**3.2.6 Tương thích**

1. Hỗ trợ mã hóa/giải mã file có kích thước lớn
2. Hỗ trợ các ký tự Unicode trong văn bản
3. Tương thích với các hệ điều hành khác nhau (thông qua Java)

**3.2.7 Độ tin cậy**

1. Xử lý các trường hợp lỗi
2. Kiểm tra tính hợp lệ của đầu vào
3. Xác nhận trước khi thực hiện các thao tác quan trọng
4. Bảo vệ dữ liệu người dùng

**3.3 USE-CASE DIAGRAM VÀ MÔ TẢ**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Hình 3.1 - Sơ đồ Use-Case**

Đăng nhập: Khi truy cập vào hệ thống, người dùng cần thực hiện đăng nhập bằng tài khoản của mình. Chức năng này giúp đảm bảo chỉ những người dùng hợp lệ mới có thể sử dụng các tính năng mã hóa, giải mã và băm dữ liệu của hệ thống.

Nhập dữ liệu: Người dùng có thể nhập dữ liệu đầu vào cho quá trình mã hóa bằng hai cách: nhập trực tiếp văn bản vào ô nhập liệu trên giao diện hoặc lựa chọn một file có sẵn từ máy tính. Điều này giúp người dùng linh hoạt trong việc cung cấp dữ liệu cần mã hóa hoặc giải mã.

Chọn cách mã hóa: Sau khi nhập dữ liệu, người dùng có thể lựa chọn một trong hai phương pháp: mã hóa/giải mã bằng thuật toán AES (mã hóa đối xứng) hoặc sử dụng hàm băm SHA-256 để tạo ra giá trị băm của dữ liệu. Việc lựa chọn này giúp người dùng chủ động kiểm soát cách thức bảo vệ và xử lý dữ liệu.

Hiển thị kết quả: Người dùng sẽ thực hiện các thao tác nhập dữ liệu (văn bản hoặc file), chọn phương pháp xử lý (AES hoặc SHA-256), và xem kết quả (bao gồm cả các bước thực hiện của thuật toán) thông qua giao diện của hệ thống. Các chức năng này đảm bảo tính linh hoạt, minh bạch và dễ sử dụng cho người dùng khi làm việc với các thuật toán mã hóa và băm dữ liệu.

**3.4 SEQUENCE DIAGRAM CHO CÁC CHỨC NĂNG CHÍNH**

**A computer screen with white text

AI-generated content may be incorrect.**

**Hình 3.2 - Quy trình chính của hệ thống mã hóa AES và SHA-256 có đăng nhập**

Hình trên mô tả sequence diagram (sơ đồ trình tự) cho quy trình chính của hệ thống mã hóa dữ liệu sử dụng AES và SHA-256, có tích hợp chức năng đăng nhập. Quy trình bắt đầu khi người dùng nhập thông tin đăng nhập trên giao diện. Giao diện sẽ gửi thông tin này đến hệ thống xác thực để kiểm tra. Sau khi xác thực thành công hoặc thất bại, kết quả sẽ được trả về cho người dùng.

Khi đăng nhập thành công, người dùng tiếp tục nhập hoặc chọn dữ liệu cần xử lý, sau đó lựa chọn phương pháp mã hóa (AES hoặc SHA-256). Giao diện sẽ kiểm tra lựa chọn của người dùng. Nếu người dùng chọn AES, giao diện sẽ truyền dữ liệu, khóa và IV đến module AES để thực hiện mã hóa hoặc giải mã, sau đó nhận lại kết quả. Nếu người dùng chọn SHA-256, giao diện sẽ truyền dữ liệu đến module SHA-256 để thực hiện băm và nhận lại kết quả băm.

Cuối cùng, kết quả (và từng bước thực hiện nếu có) sẽ được hiển thị cho người dùng trên giao diện. Sơ đồ này thể hiện rõ luồng tương tác giữa các thành phần chính của hệ thống, đảm bảo tính bảo mật, linh hoạt và minh bạch trong quá trình xử lý dữ liệu.

**Chương 4. THIẾT KẾ**

**4.1 KIẾN TRÚC TỔNG THỂ**

**4.1.1 Lớp giao diện người dùng**

**Chức năng:** Hiển thị giao diện cho người dùng thao tác (đăng nhập, nhập/chọn dữ liệu, chọn thuật toán, xem kết quả).

Tiếp nhận dữ liệu đầu vào từ người dùng và hiển thị kết quả đầu ra.

**Thành phần tiêu biểu:** Các class như AESPanel, các form đăng nhập, các nút chức năng, các ô nhập liệu, vùng hiển thị kết quả.

**4.1.2 Lớp xử lý nghiệp vụ**

**Chức năng:**

Xử lý các yêu cầu từ giao diện: xác thực đăng nhập, mã hóa/giải mã dữ liệu bằng AES, băm dữ liệu bằng SHA-256.

Kiểm tra tính hợp lệ của dữ liệu, khóa, IV.

Điều phối luồng dữ liệu giữa các lớp.

**Thành phần tiêu biểu:**

Các class như AES, SHA256, FileEncryption, Authenticator (nếu có).

Các phương thức thực hiện mã hóa, giải mã, băm, xác thực.

**4.1.3 Lớp dữ liệu**

**Chức năng:**

Lưu trữ và truy xuất dữ liệu người dùng, file đầu vào/đầu ra, thông tin đăng nhập (nếu có).

Đọc/ghi file khi người dùng chọn file hoặc lưu kết quả.

**Thành phần tiêu biểu:**

Các thao tác với file hệ thống (File I/O), có thể là các class hoặc phương thức hỗ trợ đọc/ghi file, lưu trữ thông tin đăng nhập (nếu có).

**4.2 CLASS DIAGRAM**

A diagram of a computer

AI-generated content may be incorrect.

**4.3 GIAO DIỆN NGƯỜI DÙNG**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 4.1 – Giao diện ban đầu**

Giao diện trên là màn hình đăng nhập của hệ thống. Người dùng sẽ nhập tên đăng nhập và mật khẩu vào hai ô tương ứng. Giao diện được thiết kế thân thiện, trực quan với các biểu tượng minh họa cho từng trường nhập liệu, giúp người dùng dễ dàng nhận biết chức năng của từng ô. Hai nút chức năng chính ở phía dưới là “Đăng nhập” và “Đăng ký”, cho phép người dùng thực hiện đăng nhập vào hệ thống hoặc chuyển sang màn hình đăng ký tài khoản mới. Màu sắc chủ đạo là xanh dương tạo cảm giác hiện đại, chuyên nghiệp và dễ nhìn. Giao diện này đảm bảo tính bảo mật và thuận tiện cho người dùng khi truy cập vào các chức năng mã hóa, giải mã và băm dữ liệu của hệ thống.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 4.2 – Giao diện thuật toán mã hóa AES**

Giao diện trên là màn hình chính của chức năng mã hóa/giải mã AES trong ứng dụng. Người dùng có thể lựa chọn giữa hai chế độ: mã hóa văn bản hoặc mã hóa file thông qua các nút radio ở phía trên. Giao diện cung cấp các trường để nhập khóa (Key) và IV (Initialization Vector), đồng thời hỗ trợ tạo khóa và IV ngẫu nhiên bằng các nút “Tạo khóa” và “Tạo IV” tiện lợi. Phần trung tâm của giao diện được chia thành hai vùng lớn: “Dữ liệu đầu vào” để người dùng nhập hoặc chọn dữ liệu cần mã hóa/giải mã, và “Kết quả” để hiển thị kết quả sau khi xử lý. Dưới cùng là các nút chức năng: Mã hóa, Giải mã, Chọn File, Lưu File, Xóa và Thoát, giúp người dùng thao tác dễ dàng với dữ liệu và file. Giao diện được thiết kế trực quan, hiện đại với các biểu tượng minh họa rõ ràng cho từng chức năng, giúp người dùng dễ dàng sử dụng và thao tác nhanh chóng trong quá trình mã hóa hoặc giải mã dữ liệu bằng thuật toán AES.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 4.3 – Giao diện thuật toán băm SHA-256**

Giao diện trên là màn hình chức năng băm SHA-256 trong ứng dụng. Người dùng có thể lựa chọn giữa nhiều chế độ thao tác như: băm văn bản, băm file, so sánh băm 2 file, hoặc kiểm tra file thông qua các nút radio ở phía trên. Giao diện được chia thành hai vùng chính: “Dữ liệu đầu vào” để người dùng nhập văn bản hoặc chọn file cần băm, và “Kết quả” để hiển thị giá trị băm hoặc kết quả so sánh/kiểm tra. Phía dưới là các nút chức năng: Băm, Lưu File, Xóa và Thoát, giúp người dùng dễ dàng thực hiện thao tác băm dữ liệu, lưu lại kết quả, xóa dữ liệu hoặc thoát khỏi chương trình. Giao diện được thiết kế đơn giản, trực quan, giúp người dùng thao tác nhanh chóng và thuận tiện khi làm việc với thuật toán băm SHA-256 trong việc kiểm tra tính toàn vẹn và bảo mật dữ liệu.

**CHƯƠNG 5. CÀI ĐẶT**

**5.1 CẤU TRÚC SOURCE CODE**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 5.1 - Cấu trúc của source code**

Project được tổ chức theo các package và thư mục rõ ràng, đảm bảo tính dễ bảo trì, mở rộng và thuận tiện cho việc phát triển. Cụ thể:

1. Thư mục src/: Chứa toàn bộ mã nguồn của ứng dụng.

Chứa các lớp giao diện người dùng (GUI) như:

1. LoginFrame.java: Giao diện đăng nhập.
2. RegisterFrame.java: Giao diện đăng ký tài khoản.
3. AESPanel.java: Giao diện chức năng mã hóa/giải mã AES.
4. SHA\_256Panel.java: Giao diện chức năng băm SHA-256.
5. src/algorithm/

Chứa các lớp hiện thực thuật toán:

1. AES.java: Thuật toán mã hóa/giải mã AES.
2. SHA\_256.java: Thuật toán băm SHA-256.
3. src/db/

Chứa các lớp quản lý dữ liệu người dùng:

1. User.java: Định nghĩa đối tượng người dùng.
2. UserDatabase.java: Quản lý danh sách người dùng, xác thực đăng nhập/đăng ký.
3. src/utils/

Chứa các lớp tiện ích hỗ trợ:

1. FileEncryption.java: Hỗ trợ mã hóa/giải mã file.
2. Các class/hàm tiện ích khác (nếu có) như chuyển đổi dữ liệu, kiểm tra định dạng, v.v.
3. src/images/

Chứa các file hình ảnh, icon sử dụng cho giao diện (nút bấm, logo).

**5.2 CÁC THUẬT TOÁN VÀ KỸ THUẬT QUAN TRỌNG**

**5.2.1 Các thuật toán**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Hình 5.2 - Thuật toán mã hóa AES**

**Thuật toán mã hóa đối xứng AES (Advanced Encryption Standard)**

1. Mục đích: Bảo vệ dữ liệu bằng cách chuyển đổi dữ liệu gốc thành dạng không thể đọc được nếu không có khóa giải mã.
2. Kỹ thuật sử dụng:
3. Chế độ hoạt động CBC (Cipher Block Chaining) giúp tăng cường bảo mật bằng cách sử dụng một vector khởi tạo (IV) ngẫu nhiên cho mỗi lần mã hóa.
4. Quản lý khóa và IV dưới dạng chuỗi hex, kiểm tra độ dài và tính hợp lệ của khóa/IV.
5. Hỗ trợ mã hóa và giải mã cả văn bản và file.

A screenshot of a computer

AI-generated content may be incorrect.

**Hình 5.3 - Thuật toán băm SHA-256**

**Thuật toán băm SHA-256 (Secure Hash Algorithm 256-bit)**

1. Mục đích: Tạo ra một giá trị băm (hash) duy nhất cho mỗi dữ liệu đầu vào, dùng để kiểm tra tính toàn vẹn dữ liệu hoặc xác thực thông tin.
2. Kỹ thuật sử dụng:
3. Hiện thực đầy đủ các bước của thuật toán SHA-256: padding dữ liệu, chia khối, xử lý từng khối, cập nhật giá trị hash.
4. Hỗ trợ băm văn bản, băm file, so sánh băm giữa hai file và kiểm tra tính toàn vẹn file.

**5.2.2 Các kỹ thuật**

**Kỹ thuật xác thực người dùng**

1. Mục đích: Đảm bảo chỉ người dùng hợp lệ mới có thể truy cập và sử dụng các chức năng của hệ thống.

**Kỹ thuật sử dụng:**

1. Lưu trữ mật khẩu dưới dạng giá trị băm SHA-256 thay vì lưu mật khẩu gốc, tăng cường bảo mật thông tin người dùng.
2. Kiểm tra thông tin đăng nhập và đăng ký thông qua lớp quản lý người dùng (UserDatabase).

**Kỹ thuật xử lý file:**

1. Mục đích: Cho phép người dùng mã hóa, giải mã, băm và lưu trữ dữ liệu dưới dạng file.

**Kỹ thuật sử dụng:**

1. Đọc và ghi file nhị phân an toàn, đảm bảo dữ liệu không bị mất mát hoặc lỗi định dạng.
2. Hỗ trợ chọn file, lưu file, xác nhận ghi đè file khi cần thiết.
3. Kỹ thuật giao diện người dùng (GUI)
4. Mục đích: Đem lại trải nghiệm thân thiện, trực quan cho người dùng.

**Kỹ thuật sử dụng:**

1. Sử dụng Java Swing để xây dựng các cửa sổ, panel, nút bấm, trường nhập liệu, và các thành phần giao diện khác.
2. Sử dụng icon, màu sắc, hiệu ứng hover để tăng tính thẩm mỹ và dễ sử dụng.

**5.3 CODE MẪU CHO CÁC CHỨC NĂNG CHÍNH**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Hình 5.4 - Kiểm tra đăng nhập trong LoginFrame**

A screen shot of a computer program

AI-generated content may be incorrect.

**Hình 5.5 – Mã hóa văn bản bằng thuật toán AES**

**A computer screen shot of a program

AI-generated content may be incorrect.**

**Hình 5.6 – Giải mã văn bản bằng thuật toán AES**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Hình 5.7 – Mã hóa file bằng thuật toán AES**

**A computer screen shot of a program code

AI-generated content may be incorrect.**

**Hình 5.8 – Giải mã file bằng thuật toán AES**

**A computer screen shot of a program

AI-generated content may be incorrect.**

**Hình 5.9 – Băm văn bản bằng SHA-256**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Hình 5.10 – Băm file bằng SHA-256**

**A computer screen shot of a program

AI-generated content may be incorrect.**

**Hình 5.11 – So sánh file bằng thuật toán SHA-256**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Hình 5.12 – Kiểm tra file bằng thuật toán SHA-256**

**5.4 HƯỚNG DẪN BUILD VÀ CHẠY CHƯƠNG TRÌNH**

**5.4.1 Yêu cầu hệ thống**

1. JDK: Cài đặt Java Development Kit (JDK) phiên bản 8 trở lên.
2. IDE (khuyến nghị): IntelliJ IDEA, Eclipse, NetBeans, hoặc có thể dùng dòng lệnh.

**5.4.2 Buil và chạy bằng IDE**

*a) Mở project*

1. Mở IDE, chọn Open Project hoặc Import Project và trỏ đến thư mục chứa project.

*b) Thiết lập cấu hình chạy*

1. Đảm bảo JDK đã được chọn cho project.
2. Đặt class chứa hàm main (thường là class khởi động giao diện, ví dụ: Main.java hoặc LoginFrame.java) làm entry point.

*c) Build và chạy*

1. Nhấn Run (hoặc Shift+F10 với IntelliJ, F11 với NetBeans, Ctrl+F11 với Eclipse).
2. Giao diện chương trình sẽ hiện ra, bạn có thể đăng nhập và sử dụng các chức năng.

**CHƯƠNG 6. KIỂM THỬ**

**6.1 Kế hoạc kiểm thử**

**6.1.1 Mục tiêu kiểm thử**

1. Đảm bảo tất cả các chức năng chính của hệ thống hoạt động đúng như yêu cầu.
2. Phát hiện và sửa các lỗi phát sinh trong quá trình sử dụng.
3. Đảm bảo giao diện thân thiện, dễ sử dụng và không phát sinh lỗi khi thao tác.

**6.1.2 Phạm vi kiểm thử**

1. Kiểm thử chức năng đăng nhập, đăng ký.
2. Kiểm thử chức năng mã hóa/giải mã AES (văn bản và file).
3. Kiểm thử chức năng băm SHA-256 (văn bản, file, so sánh, kiểm tra).
4. Kiểm thử các chức năng phụ: lưu file, xóa dữ liệu, thoát chương trình.
5. Kiểm thử giao diện người dùng.

**6.1.3 Các loại kiểm thử**

*a) Kiểm thử chức năng*

1. Đăng nhập với tài khoản đúng/sai.
2. Đăng ký tài khoản mới (kiểm tra trùng tên, xác nhận mật khẩu).
3. Mã hóa/giải mã văn bản với khóa và IV hợp lệ/không hợp lệ.
4. Mã hóa/giải mã file với file hợp lệ/không hợp lệ.
5. Băm văn bản, băm file, so sánh băm 2 file, kiểm tra file.
6. Lưu kết quả ra file, kiểm tra nội dung file lưu.
7. Xóa dữ liệu, kiểm tra dữ liệu đã được xóa.
8. Thoát chương trình, kiểm tra chương trình đóng đúng.

*b) Kiểm thử giao diện*

1. Kiểm tra hiển thị đầy đủ các thành phần giao diện.
2. Kiểm tra các nút bấm, trường nhập liệu, thông báo lỗi/thành công.
3. Kiểm tra icon, màu sắc, font chữ, hiệu ứng hover.

*c) Kiểm thử phi chức năng (Non-functional Testing)*

1. Kiểm tra hiệu suất mã hóa/giải mã với dữ liệu lớn.
2. Kiểm tra bảo mật: mật khẩu lưu dưới dạng băm, không lưu plain text.
3. Kiểm tra tính ổn định khi thao tác liên tục, nhập dữ liệu lớn, thao tác sai.

**6.2 Kết quả kiểm thử**

Sau quá trình kiểm thử, tất cả các chức năng chính của hệ thống đều hoạt động đúng như mong đợi. Cụ thể, chức năng đăng nhập và đăng ký đều cho kết quả chính xác: hệ thống cho phép đăng nhập với tài khoản hợp lệ và thông báo lỗi khi nhập sai thông tin hoặc trùng tên tài khoản khi đăng ký.

Chức năng mã hóa và giải mã AES hoạt động ổn định, đảm bảo dữ liệu sau khi mã hóa có thể giải mã lại chính xác về văn bản gốc, đồng thời hệ thống cũng kiểm tra và thông báo lỗi khi người dùng nhập khóa hoặc IV không hợp lệ.Chức năng băm SHA-256 cho kết quả băm đúng với cả văn bản và file, đồng thời hỗ trợ so sánh băm giữa hai file và kiểm tra tính toàn vẹn file một cách chính xác. Các chức năng phụ như lưu file, xóa dữ liệu và thoát chương trình đều thực hiện đúng vai trò, không phát sinh lỗi trong quá trình sử dụng. Giao diện người dùng hiển thị đầy đủ các thành phần, các nút chức năng hoạt động tốt, thông báo rõ ràng, dễ hiểu.

**6.3 Demo các chức năng**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 6.1 – Thông báo lỗi đăng nhập thiếu thông tin**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 6.2 – Cảnh báo thiếu dữ liệu đầu vào mã hóa**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 6.3 – Kết quả mã hóa văn bản**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 6.4 – Kết quả mã hóa được lưu file txt**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 6.5 – Kết quả mã hóa file thành công**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 6.6 – Giải mã file thành công**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 6.7 – Kết quả băm SHA-256 lưu vào file txt**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 6.8 – Kết quả băm file thành công**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 6.9 – So sánh băm hai file khác nhau**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 6.10 – Kết quả kiểm tra file không bị thay đổi**

**CHƯƠNG 7. KẾT LUẬN**

**7.1 KẾT QUẢ ĐẠT ĐƯỢC**

1. Nắm vững lý thuyết về hai thuật toán quan trọng trong lĩnh vực mật mã hiện đại là AES và SHA-256, bao gồm: nguyên lý hoạt động, cấu trúc thuật toán, các bước xử lý dữ liệu, và vai trò trong bảo mật thông tin.
2. Cài đặt và mô phỏng thành công thuật toán AES và SHA-256 bằng ngôn ngữ lập trình phù hợp (ví dụ: Java hoặc Python), qua đó củng cố kiến thức và hiểu rõ từng giai đoạn của các thuật toán.
3. Phân tích chi tiết mức độ an toàn của hai thuật toán, bao gồm khả năng chống lại các dạng tấn công phổ biến như brute-force, va chạm, tìm ngược, từ đó khẳng định độ tin cậy khi triển khai trong thực tế.
4. Trình bày được mối quan hệ và ứng dụng thực tiễn của AES và SHA-256 trong các lĩnh vực như: bảo vệ dữ liệu, xác thực thông tin, blockchain, lưu trữ mật khẩu, chữ ký số,…

**7.2 HẠN CHẾ VÀ HƯỚNG PHÁT TRIỂN**

**7.2.1 Hạn chế**

1. Chưa triển khai trên quy mô lớn hoặc tích hợp hệ thống thực tế, do giới hạn về thời gian và môi trường triển khai.
2. Chưa đánh giá được hiệu năng (thời gian xử lý, độ tối ưu) của các thuật toán khi so sánh với các phương pháp khác hoặc khi xử lý dữ liệu lớn.
3. Thiếu phần thử nghiệm với dữ liệu thực tế trong môi trường mạng có yếu tố an ninh, như trong giao tiếp giữa máy chủ - máy khách, hoặc trong hệ thống lưu trữ phân tán.

**7.2.2 Hướng phát triển**

1. Tích hợp AES và SHA-256 vào hệ thống bảo mật hoàn chỉnh, ví dụ như ứng dụng đăng nhập bảo mật hoặc hệ thống quản lý dữ liệu có mã hóa và xác thực.
2. Nghiên cứu các thuật toán nâng cao hơn như AES-GCM (vừa mã hóa vừa xác thực), SHA-3 (phiên bản mới hơn của SHA-2), hoặc các phương pháp mật mã hậu lượng tử.
3. Đánh giá hiệu suất trên phần cứng chuyên dụng (như FPGA hoặc GPU), qua đó hiểu rõ hơn về khả năng triển khai thực tế và hiệu quả của từng thuật toán.
4. Mở rộng nghiên cứu sang các vấn đề liên quan đến bảo mật toàn vẹn dữ liệu, bảo vệ quyền riêng tư người dùng và các ứng dụng thực tiễn như chữ ký số, xác thực blockchain hoặc hệ thống phi tập trung.