ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

Môn học: **MẬT MÃ HỌC**

BÁO CÁO CUỐI KỲ

ĐỀ TÀI: HYBRID DESIGN FOR CLOUD DATA SECURITY USING COMBINATION OF AES, ECC AND LSB STEGANOGRAPHY

Lớp: **NT219.M21.ATCL**

Giảng viên hướng dẫn: **Nguyễn Ngọc Tự**

Nhóm 5

Nguyễn Mạnh Cường	20520421
Trương Thị Hoàng Hảo	20520191
Hoàng Văn Anh Đức	20520890

Thành phố Hồ Chí Minh

Ngày 8 tháng 6 năm 2022

MỤC LỤC

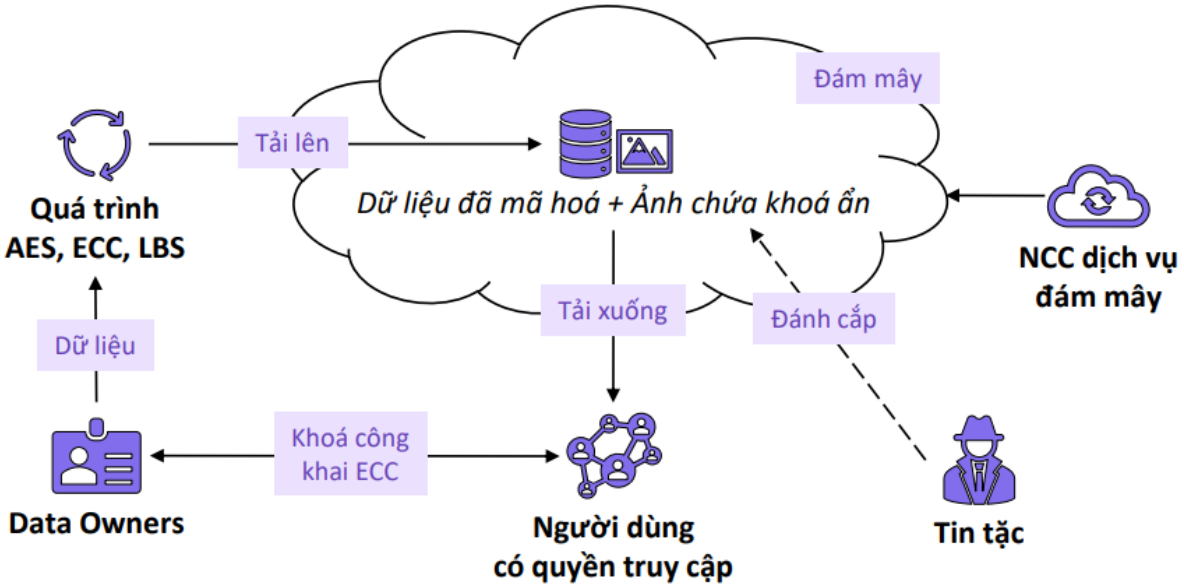
PHÂN CÔNG CÔNG VIỆC

1. Nguyễn Mạnh Cường: LSB
2. Trương Thị Hoàng Hảo: ECC
3. Hoàng Văn Anh Đức: AES

A. NGỮ CẢNH

- Một người dùng A có nhu cầu chia sẻ dữ liệu cho những người dùng xác định thông qua cloud.
- A upload dữ liệu lên cloud và sử dụng mã hoá AES để mã hoá dữ liệu trước khi upload.
- Việc quản lý và chia sẻ key mã hoá AES 256- bit là không được bảm bảo trên môi trường cloud nên người dùng A áp dụng thêm thuật toán mã hoá bất đối xứng ECC cho việc mã hoá key AES đồng thời sử dụng phương pháp LSB để ẩn key AES trên vào 1 hình ảnh để gửi AES key đã được mã hoá đi kèm file data gốc lên cloud.

B. MÔ HÌNH CÁC BÊN LIÊN QUAN



C. VAI TRÒ CỦA CÁC THUẬT TOÁN TRONG VIỆC BẢO MẬT

1. <u>Steganography - LSB</u>

- Steganography là một kỹ thuật che giấu thông tin bí mật trong các tệp dữ liệu thông thường như tệp hình ảnh, tệp âm thanh và tập video. Steganography sử

dụng một số kỹ thuật để che giấu thông tin bí mật trong những phần dữ liệu không quan trọng của các tập tin.

- Steganography là một kỹ thuật khó nhận dạng và ít người biết đến, thông qua đó dữ liệu bí mật có thể được truyền trong tình trạng bảo vệ an toàn. Steganography cũng tránh làm ảnh hưởng đến tính bảo mật của các thuật toán mã hóa khác.

⇨ Trong bài này, sử dụng Least Significant Bit (LSB) tương tác với hình ảnh. Khi khóa được lưu trữ trong bức ảnh thì các khóa bí mật có thể được bảo mật cho dù có người xâm nhập được vào nơi lưu trữ dữ liệu (Google Drive). Những bit kém quan trọng nhất của mỗi pixel trong một hình ảnh có thể được sử dụng để ẩn một bit của thông tin bí mật.

2. <u>Elliptic curve cryptography</u>

- Các thuật toán bất đối xứng đóng một vai trò quan trọng trong việc lấp đầy các khuyết điểm bởi các thuật toán đối xứng. Chữ ký số và trao đổi khóa được phân phối bởi các thuật toán bất đối xứng đã được chứng minh hiệu quả.
- Lợi thế chính của việc sử dụng các kỹ thuật xác thực bất đối xứng là nó thường yêu cầu một lần vượt qua để xác thực. Mặc dù nó yêu cầu tính toán mở rộng hơn so với thuật toán đối xứng, nhưng thời gian tính toán thường mất ít thời gian hơn.
- Trong kỹ thuật bất đối xứng việc lựa chọn thuật toán rất khó vì tất cả các thuật toán đều dựa trên các vấn đề toán học. Trong mật mã khóa công khai, lựa chọn thuật toán dựa trên kích thước khóa, cũng ảnh hưởng đến tốc độ của thuật toán. Dưới đây là bảng so sánh những thuật toán dựa trên khóa của chúng.

AES	*Diffie-Hellman*	*RSA*	*ECC*
80	1,024	1,024	160
112	2,048	2,048	224
128	3,072	3,072	256
192	7,680	7,680	384
256	15,360	15,360	521

- Sức mạnh của một thuật toán phụ thuộc vào số lượng tài nguyên hoặc thời gian hoặc cả 2 thứ này để có thể phá lớp bảo mật.
- Các giá trị trong bảng trên cho thấy ECC cung cấp bảo mật tốt hơn với độ dài khóa nhỏ hơn nhiều.

⇨ Trong bài báo cáo này, ECC được sử dụng để mã hóa và bảo mật khóa AES. Tổng thể bảo mật của cơ chế được trình bày trong này dựa trên sức mạnh mật mã của ECC.

3. Advanced encryption standard

- Hiện tại AES là ứng cử viên chính cho việc đảm bảo an ninh trong hệ thống kỹ thuật số. Hầu hết các bảo mật dữ liệu có sẵn trong đám mây dựa trên AES. Dễ thực hiện và tốc độ cao làm cho nó trở thành một lựa chọn lý tưởng để mã hóa các khối lớn dữ liệu được lưu trữ, xử lý và truyền bằng cách sử dụng đám mây.
- Thuật toán AES là một mật mã khối đối xứng sử dụng một khóa để mã hóa và giải mã thông tin cần bảo mật. Việc mã hóa dữ liệu có nghĩa sang một dạng " khó hiểu " gọi là bản mã ; giải mã bản mã sẽ chuyển dữ liệu về dạng nguyên bản gọi là bản rõ .
- AES cung cấp các lợi ích của khả năng thiết kế khác nhau và kiến trúc cùng với bảo mật đáng tin cậy. Bảo mật thiết kế do AES cung cấp yêu cầu cẩn thận triển khai để phù hợp với các yêu cầu đám mây

- AES là một mật mã khối làm việc với các khối dữ liệu (đầu vào và đầu ra) 128 bit - 16 bytes và thực hiện một số lần lặp để gây ra sự khó hiểu. Thuật toán mật mã này có thể cung cấp bảo mật với kích thước khóa thay đổi là 128, 192 và 256 bit. Các bước liên quan đến AES là:

  • SubBytes

  • ShiftRows

  • MixColumns

  • XORRoundKey

⇨ AES là lựa chọn tốt nhất để mã hóa các khối dữ liệu lớn. Các kỹ thuật steganography và mã hóa bất đối xứng không phù hợp với hiệu suất do AES cung cấp. Bất kỳ giải pháp nào cho đám mây bảo mật phải xem xét lượng lớn dữ liệu được lưu trữ và được xử lý bằng công nghệ đám mây.

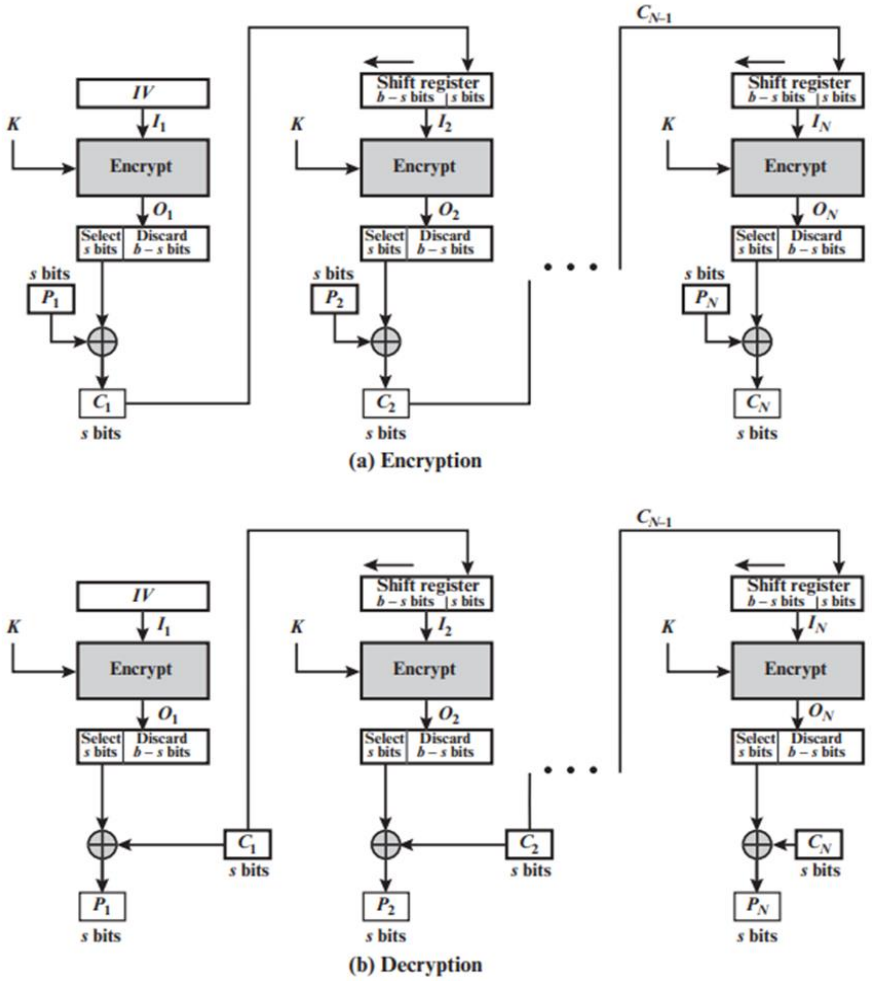
D. CỤ THỂ TỪNG THUẬT TOÁN

1. AES

❖ Chế độ mã hóa và giải mã sẽ là CFB:

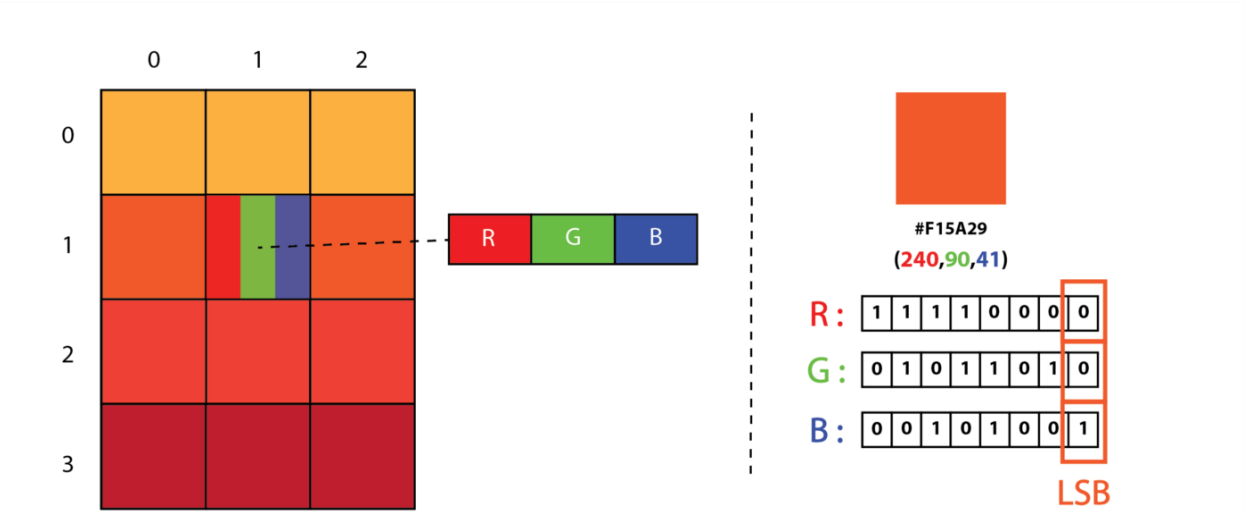
- Xử lý từng phần nhỏ của bản rõ thành bản mã, thay vì xử lý toàn bộ khối cùng một lúc. Chế độ này sử dụng một thanh ghi shift có độ dài là một khối và được chia thành nhiều phần.Ví dụ , nếu kích thước khối là 8 byte, với một byte được xử lý tại một thời điểm, thanh ghi shift được chia thành tám phần.
- Nếu một bit trong văn bản mật mã bị xáo trộn thì một bit trong văn bản thuần túy bị xáo trộn và thanh ghi shift bị hỏng . Kết quả là một số văn bản thuần túy tiếp theo bị xáo trộn cho đến khi bit này được chuyển ra khỏi thanh ghi shift. Kích thước phản hồi mặc định có thể thay đổi theo thuật toán, nhưng thường là 8 bit hoặc số bit của blocksize.

- Bạn có thể thay đổi số lượng bit phản hồi bằng cách sử dụng *System.Security.Cryptography.SymmetricAlgorithm.FeedbackSize.property*. Các thuật toán hỗ trợ CFB sử dụng thuộc tính này để cài đặt feedback.
- Đầu vào được xử lý s bit (thường là 8) tại một thời điểm. Bản mã trước đó được sử dụng như đầu vào cho thuật toán mã hóa để tạo ra đầu ra giả ngẫu nhiên, được XOR với văn bản rõ để tạo ra đơn vị tiếp theo của bản mã.
- Ưu điểm chính của chế độ CFB là nó không sử dụng thuật toán giải mã, nó thường nhanh hơn chế độ CBC. Mã hóa CFB không tiết lộ bất kỳ mẫu nào mà bản rõ có thể có.
- Không thể mã hóa song song nhiều khối.



2. LSB

- Trong bài báo cáo này, chúng tôi sử dụng các ảnh tự chọn ở định dạng bitmat (.bmp .dib .png) ở hệ màu RGB, với điều kiện kích thước phải đủ để chứa thông điệp hoặc file cần nhúng. Chúng tôi tránh sử dụng các định dạng ảnh JPEG do thuật toán nén sẽ làm mất những dữ liệu quan trọng.
- Mỗi một pixel của ảnh bao gồm 3 bytes lưu trữ mức hiển thị của 3 màu trong hệ RGB (đỏ, xanh lá, xanh dương). Mỗi một byte của tập tin cần nhúng sẽ được ẩn trong 1 pixel của ảnh.



- Mã hóa:
    - Mỗi 8 bits của file dữ liệu sẽ được nhúng vào những bits kém quan trọng trong 1 pixel (24 bits) với quy tắc như sau:

```
bits[0] = redBits[5];
bits[1] = greenBits[5];
bits[2] = redBits[6];
bits[3] = redBits[7];
bits[4] = greenBits[6];
bits[5] = greenBits[7];
bits[6] = blueBits[6];
bits[7] = blueBits[7];
```

- Trực tiếp thay đổi 8 bits kém quan trọng trong 1 pixel, nhìn bằng mắt thường sẽ khó phân biệt được ảnh trước và sau khi nhúng dữ liệu.
- Một số bits khác sẽ được ẩn 1 số các ký tự cờ nhằm mục đích xác định ảnh đã được nhúng hay chưa, thuận tiện cho việc giải mã.
- Những bits còn lại sẽ được giữ nguyên và tạo thành một bản sao ảnh với định dạng .bmp đã được nhúng dữ liệu.

- Giải mã:
  - Nhìn chung, việc giải mã đơn giản chỉ là lấy những bits được ẩn theo quy tắc trên và xuất ra file dữ liệu đã nhúng.
  - Việc xác định ảnh đó có phải ảnh chứa dữ liệu ẩn hay không đã được đề cập ở phần mã hóa, chỉ cần xác định vị trí cờ và đọc cờ đã ẩn. Tránh được việc giải mã những ảnh không chứa dữ liệu.

3. ECC

- **Bước 1:** Tạo cặp private – public keys ngẫu nhiên cho mỗi người dùng.
- **Bước 2:** Encryption (mã hoá)

Algorithm 1: ECC encryption

Input: Elliptic curve parameters (E, p, P, n), private key P_r and public key $P_u = P_rP$.

Output: Cipher text (CT_1, CT_2)

Ensure: *Convert message AES key K to point on elliptic curve M.*

1 Select $k \in R^{[1,n-1]}$

2 **Compute** $CT_1 = kP$

3 **Compute** $CT_2 = M + kP_u$

4 **Return** (CT_1, CT_2)

- K: Chọn ngẫu nhiên từ 1 đến n-1.
- P: Base point.

- M: Điểm chứa AES key.
- Ciphertext sẽ là toạ độ 2 điểm CT1 và CT2.
- Ở đây cần đảm bảo rằng khoá của mã hoá AES đã được biến đổi thành 1 điểm hợp lệ trên đường cong (curve).

- **Bước 3:** Decryption (giải mã)

Algorithm 2: ECC decryption

Input: Elliptic curve parameters (E, p, P, n), private key P_r and cipher text (CT_1, CT_2)

Output: AES key K as message M

Ensure:

1 **Compute** $M = CT_2 - dCT_1$

2 **Extract key** K from M

3 **Return** (K)

- Ta có:

  CT2 – dCT1

  mà CT2 = M + kPu, CT1 = kP

  => M + kPu – dkP

  = M + kPu – kdP

  = M + k(Pu – dP)

- Vậy ta lấy được toạ độ điểm M khi và chỉ khi dp = Pu (public key và private key đúng là 1 cặp).
- Từ điểm M, ta thu được khoá của mã hoá AES

⇨ Tất cả các phép cộng, nhân và trừ trong thuật toán 1 và thuật toán 2 đều được thực hiện trên các điểm của cùng 1 đường cong (curve).

E. BẢO MẬT KHI SỬ DỤNG

- 256-bit key của mã hoá AES sẽ được mã hóa và lưu trữ trong 1 bức ảnh trên cloud với dữ liệu được mã hoá mà vẫn đảm bảo thông tin không bị tiết lộ.
- 256-bit key của mã hoá AES chỉ được chia sẻ với những người dùng được chỉ định bởi data owner.
- Việc quản lý khoá không phụ thuộc vào bên thứ 3.

F. CÁC BƯỚC CỦA GIẢI PHÁP

- Khoá của mã hoá AES được tạo ngẫu nhiên bởi openssl.
- Khoá này sẽ được dùng để mã hóa dữ liệu.
- Khoá của AES tiếp đến sẽ được mã hoá bằng mã hoá ECC, rồi được ẩn vào 1 hình ảnh bằng kỹ thuật Least Significant Bit (LSB).
- Tải dữ liệu đã được mã hóa và hình ảnh lên Cloud (Google Drive).
- Khi cần đến dữ liệu này thì ta sẽ dùng LSB để lấy key của AES đã được mã hóa bởi ECC ra. Dùng ECC để giải mã để ra được key của AES và dùng key đó giải mã dữ liệu.

G. SƠ LƯỢC THỰC NGHIỆM

- AES với khoá 256-bit thực hiện bằng C# với thư viện AES256.
- ECC thực hiện bằng C# với thư viện Bouncy Castle, curve sec256k1 của NIST.
- LSB Stegnography thực hiện bằng C# với các class hổ trợ tách bits, tách bytes các file dữ liệu và thay đổi một pixel trong ảnh ở dạng bitmap.
- Sử dụng Google drive để lưu trữ.