0BÁO CÁO THỰC HÀNH

**Môn học: CRYPTOGRAPHY – Mật mã học**

**Tên chủ đề: BLOCK CIPHERS**

*GVHD: Tô Trọng Nghĩa*

1. **THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: **NT219.N21.ANTT.1**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ và tên | MSSV | Email |
| 1 | Lê Đoàn Trà My | 21521149 | 21521149@gm.uit.edu.vn |

1. **NỘI DUNG THỰC HIỆN:[[1]](#footnote-1)**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Nội dung | Tình trạng | Trang |
| 1 | Modes of operation of block cipher | 100% | 2 – 8 |
| 2 | Encryption Mode – ECB vs. CBC | 100% | 9 – 12 |
| 3 | Padding | 100% | 12 – 14 |
| 4 | Error Propagation | 100% | 14 – 18 |
| 5 | Programming using the Crypto Library | 50% | 19 |
| Điểm tự đánh giá | | | **8.5-9/10** |

**Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.**

BÁO CÁO CHI TIẾT

## Modes of operation of block cipher (các chế độ hoạt động của mã khối)

**1.1. Liệt kê và xác định ngắn gọn các chế độ hoạt động của mã khối, ưu và nhược điểm của các chế độ:**

***a, ECB – Electronic Codebook***

***Diagram

Description automatically generated***

*Hình 1. Quy trình ECB*

*- Định nghĩa:* Là chế độ mà bản rõ (plaintext) được chia thành các khối bit độc lập, mỗi khối được mã hóa độc lập với các khối khác 𝐶𝑖 = 𝐸𝐾 (𝑃𝑖), các khối sử dụng chung 1 khoá (key).

*- Công dụng:* truyền an toàn các giá trị đơn lẻ, đoạn thông tin ngắn.

*- Ưu điểm:*

+ Đơn giản.

+ Các khối được mã hoá song song → mã hoá nhanh hơn, tăng tốc độ xử lý; nếu một khối dữ liệu bị lỗi thì chỉ ảnh hưởng mã và giải mã khối đó, các khối khác bình thường.

*- Nhược điểm:*

+ Cùng một đoạn dữ liệu được lặp đi lặp lại trong plaintext được mã hoá → luôn tạo ra cùng một bản mã (ciphertext).

+ Dữ liệu lặp lại chứa trong bản rõ có thể hiển thị trong bản mã, nếu được căn chỉnh theo các khối.

+ Dễ bị phá vì có mối quan hệ trực tiếp giữa bản rõ và bản mã.

***b, CBC – Cipher Block Chaining***

***Diagram

Description automatically generated***

*Hình 2. Quy trình CBC*

*- Định nghĩa:* Là một cải tiến được thực hiện trên ECB, giải quyết được lỗ hổng bảo mật trong ECB (Lặp đi lặp lại cùng một khối bản rõ dẫn đến khối bản mã khác nhau); Trong CBC, mỗi ciphertext từ khối trước đó được XOR với plaintext của khối hiện tại để trở thành đầu vào cho việc thực thi thuật toán mã hoá với khoá K.

Sử dụng Vector ban đầu (IV) để bắt đầu quá trình:

Ci = EK (Pi ⊕ Ci - 1); C0 = IV

*(Pi: khối plaintext thứ i; Ci: khối ciphertext thứ i; EK là thuật toán encryption sử dụng key K; IV nghĩa là Initialization Vector, không được sử dụng lại chung cho nhiều tin nhắn.)*

*- Công dụng:* Mã hóa dữ liệu hàng loạt, xác thực.

*- Ưu điểm:*

+ Khả năng bảo mật cao hơn ECB.

+ Các khối bản rõ lặp đi lặp lại được mã hóa khác nhau.

+ Quá trình giải mã vẫn có thể thực hiện song song nhiều khối dữ liệu.

*- Nhược điểm:*

+ Không thể mã hoá song song, mã hóa của một khối phụ thuộc vào khối hiện tại và **tất cả** các khối trước nó.

+ Nếu lỗi bit xuất hiện trên ciphertext của một khối dữ liệu thì nó sẽ làm sai kết quả giải mã của các khối dữ liệu đó và khối dữ liệu tiếp theo.

+ Không đảm bảo tính toàn vẹn của dữ liệu.

+ Cần bộ tạo giá trị ngẫu nhiên cho IV, sử dụng lại IV làm rò rỉ một số thông tin.

***c, CFB – Cipher Feedback***

***Diagram, schematic

Description automatically generated***

*Hình 3. Quy trình CFB*

*- Định nghĩa:* Là chế độ mã hoá mà đầu vào được xử lý s bit tại một thời điểm. Bản mã ciphertext của khối trước được đưa đến đầu vào của thuật toán mã hoá khối tiếp theo, nó kết hợp với khối bản rõ bằng XOR để tạo ra bản mã hiện tại.

Text

Description automatically generated

*- Công dụng:* Mã hóa dữ liệu dòng, xác thực.

*- Ưu điểm:*

+ Khả năng bảo mật cao hơn ECB.

+ Quá trình giải mã có thể song song và chịu được tổn thất.

+ Tùy biến được độ dài khối dữ liệu mã hóa, giải mã thông qua thông số s.

*- Nhược điểm:*

+ Một đoạn bản mã phụ thuộc vào đoạn văn bản rõ hiện tại và **tất cả** các đoạn trước đó, không thể mã hoá song song.

+ Một đoạn bản mã bị hỏng trong quá trình truyền sẽ ảnh hưởng đến một số đoạn văn bản rõ hiện tại và tiếp theo.

+ Cần bộ tạo giá trị ngẫu nhiên cho IV, sử dụng lại IV làm rò rỉ một số thông tin.

***d, OFB – Output Feedback***

***Diagram

Description automatically generated***

*Hình 4. Quy trình OFB*

*- Định nghĩa:* Tương tự CFB nhưng trong chế độ mã hoá này giá trị ngõ ra của khối thực thi thuật toán mã hoá (không phải ciphertext) của lần hiện tại sẽ được phản hồi đến ngõ vào của lần mã hoá kế tiếp. OFB xử lý trên một khối dữ liệu với độ dài bit đầy đủ như thuật toán mã hóa quy định chứ không xử lý trên một phần hay một vài bit của khối dữ liệu. C i = P i ⊕ O i ; O i = E K (O i-1 ); O 0 = IV

*- Công dụng*: Truyền mã hóa qua các kênh có nhiễu.

*- Ưu điểm:*

+ Khả năng bảo mật cao hơn ECB.

+ Mỗi bit trong bản mã độc lập với bit hoặc các bit trước đó→ tránh lan truyền lỗi.

+ Có thể tính toán trước mật mã chuyển tiếp.

*- Nhược điểm:*

+ Vấn đề an ninh: Khi bản rõ thứ j được biết, đầu ra thứ j của hàm mật mã chuyển tiếp sẽ được biết; dễ dàng che đậy khối văn bản rõ thứ j của thông điệp khác có cùng khoá IV.

+ Dễ bị tấn công sửa đổi luồng thông báo hơn CFB.

+ Không thể thực hiện mã hóa/giải mã song song nhiều khối dữ liệu vì lần mã hóa/giải mã sau phụ thuộc vào khối ngõ ra của lần mã hóa/giải mã liền trước nó.

+ Tái sử dụng IV phá hủy hoàn toàn bảo mật

***e, CTR – Counter***

***Diagram

Description automatically generated***

*Hình 5. Quy trình CTR*

*- Định nghĩa:* Là chế độ mã hóa sử dụng một tập các khối counter, để sinh ra một tập các giá trị ngõ ra thông qua một thuật toán mã hóa. Các giá trị này được cung cấp làm đầu vào cho XOR cùng với plaintext để tạo ra ciphertext.

C i = P i ⊕ O i ; O i = E K (i)

*- Công dụng:* Mã hoá mạng tốc độ cao.

*- Ưu điểm:*

+ Khả năng bảo mật cao hơn ECB.

+ Quá trình mã hóa/giải mã của mỗi khối là độc lập.

+ Vì có một giá trị bộ đếm khác nhau cho mỗi khối nên tránh được mối quan hệ giữa bản rõ và bản mã trực tiếp → cùng một văn bản thuần túy có thể ánh xạ tới các bản mã khác nhau.

+ Đơn giản, có thể thực hiện mã hóa song song → mã và giải mã nhanh hơn.

*- Nhược điểm:*

+ Yêu cầu bộ đếm đồng bộ.

+ Tái sử dụng IV phá hủy hoàn toàn bảo mật.

+ Phần cứng cần thiết kế thêm các bộ đếm counter hoặc giải thuật tạo các giá trị counter không lặp lại.

**1.2. Làm cách nào để mã hóa song song bản rõ trên nhiều khối ở chế độ CBC?**

**Làm thế nào về giải mã?**

- Như đã trình bày ở phần 1.1.b ở trên, Trong CBC, mỗi ciphertext từ khối trước đó được XOR với plaintext của khối hiện tại để trở thành đầu vào cho việc thực thi thuật toán mã hoá với khoá K.

Do đó, chế độ CBC bình thường không thể được song song trong quá trình mã hoá.

- Chế độ mã hoá CBC được định nghĩa:

Ci = EK (Pi ⊕ Ci - 1); C0 = IV

*(Pi: khối plaintext thứ i; Ci: khối ciphertext thứ i; EK là thuật toán encryption sử dụng key K; IV nghĩa là Initialization Vector, không được sử dụng lại chung cho nhiều tin nhắn.)*

- Chế độ Cipher Block Chaining (CBC) có thể được giải mã song song các khối dữ liệu.

Công thức để giải mã CBC là:

Pi = DK (Ci )⊕ Ci - 1; C0 = IV

*(Pi: khối plaintext thứ i; Ci: khối ciphertext thứ i; DK là thuật toán decryption sử dụng key K; IV nghĩa là Initialization Vector, không được sử dụng lại chung cho nhiều tin nhắn.)*

**1.3. Tạo một tệp văn bản 100 bytes bắt đầu bằng tên và MSSV. Mã hóa tệp này dưới dạng văn bản gốc với ít nhất ba chế độ sử dụng OpenSSL (khoá và IV (nếu cần) do bạn chọn).**

- Tạo một text file 100 bytes với tên input.txt chứa thông tin tên và MSSV cùng một số thông tin khác:

Graphical user interface, text, application

Description automatically generated

*Hình 6. Nội dung và kích thước file input.txt*

- Mã hoá tệp này, với: ++ Key: 010307009AB04078ADC056402FF0263FE

++ IV: 080504AFC6670809DEF0102

+ Chế độ AES 128-bit CBC:

Text

Description automatically generated

*Hình 7. Cipher.txt theo chế độ mã hoá CBC*

+ Chế độ AES 128-bit CFB:

Graphical user interface, text

Description automatically generated

*Hình 8. Cipher.txt theo chế độ mã hoá CFB*

+ Chế độ AES 128-bit OFB:

Text

Description automatically generated

*Hình 9. Cipher.txt theo chế độ mã hoá OFB*

+ Chế độ AES 128-bit ECB:

Text

Description automatically generated

*Hình 10. Cipher.txt theo chế độ mã hoá ECB*

1. **Encryption Mode – ECB vs. CBC**

- Mã hoá tấm hình dưới bằng chế độ EBC và CBC

**Shape, circle

Description automatically generated**

*Hình 11.* pic\_original.bmp

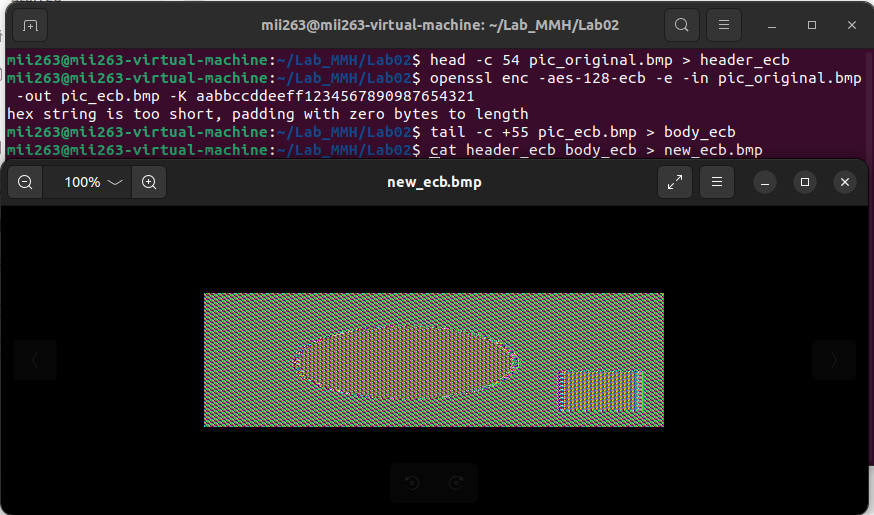
**2.1. Mã hoá dưới chế độ ECB và CBC**

- Ta có, 54 bytes đầu tiên chứa thông tin tiêu đề về hình ảnh. Nên ta tách file header riêng và nội dung riêng, mã hoá xong thì ghép 2 file lại với nhau.

++ Key: aabbccddeeff1234567890987654321

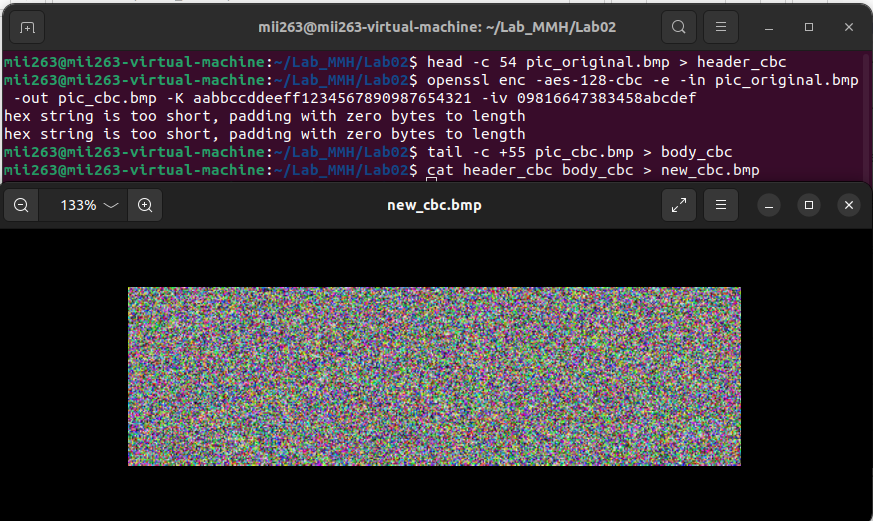
++ IV: 09816647383458abcdef

- AES 128-bit ECB:



*Hình 12.* *Đoạn lệnh thực thi và hình ảnh thu được*

- AES 128-bit CBC:



*Hình 13.* *Đoạn lệnh thực thi và hình ảnh thu được*

**2.2. Hiển thị ảnh đã mã hóa bằng chương trình xem ảnh. Có thể lấy được bất kỳ thông tin hữu ích nào về ảnh gốc từ ảnh được mã hóa không? Hãy giải thích những quan sát.**

- Với ECB: Hình ảnh sau khi mã hoá có nét tương đồng rất lớn so với hình ảnh gốc, vẫn nhìn thấy rõ hình dạng và vị trí của các hình vẽ, tuy nhiên về màu thì có phần khác.

→ Giải thích: Mỗi khối dữ liệu được mã hóa độc lập với các khối khác, không có sự tương tác giữa các khối. Các đặc trưng của ảnh ban đầu vẫn được giữ lại trong ảnh sau khi mã hóa. Hình dạng và vị trí của các hình vẽ trong ảnh được giữ nguyên và có thể nhìn thấy rõ ràng trong ảnh sau khi mã hóa. Tuy nhiên, các giá trị pixel cụ thể của các hình vẽ sẽ bị thay đổi và không giống hệt với giá trị pixel trong ảnh gốc.

- Với CBC: Hình ảnh sau khi mã hoá không có những nét đặc trưng liên hệ giống với hình ảnh ban đầu.

→ Giải thích: Mỗi khối dữ liệu được xử lý bằng cách thực hiện phép XOR với khối trước đó trước khi được mã hóa. Điều này làm cho các khối dữ liệu khác nhau được mã hóa thành các khối mã khác nhau, ngay cả khi chúng có một số phần giống nhau làm cho các đặc trưng của ảnh ban đầu không còn được giữ lại trong ảnh sau khi mã hóa. Hơn nữa, trong CBC, một khối dữ liệu bị thay đổi sẽ ảnh hưởng đến các khối dữ liệu tiếp theo trong quá trình mã hóa

* 1. **Chọn một hình ảnh, lặp lại thí nghiệm trên và báo cáo kết quả quan sát.**

**Icon

Description automatically generated**

*Hình 14.* *Hình ảnh tự chọn*

*- Chế độ ECB:*

*Graphical user interface

Description automatically generated with low confidence*

*Hình 15.* *Đoạn lệnh thực thi và hình ảnh thu được*

→ Quan sát: Hình ảnh vẫn có mối liên kết với hình ảnh gốc, vẫn có thể nhìn và đoán được hình ảnh ban đầu chỉ có các pixel màu của hình thay đổi.

*- Chế độ CBC:*

*Graphical user interface, text

Description automatically generated*

*Hình 16.* *Đoạn lệnh thực thi và hình ảnh thu được*

→ Quan sát: Hình ảnh không có nét tương đồng.

1. **Padding**

**3.1. Dùng ECB, CBC, CFB, OFB để mã hoá tệp. Chế độ nào có sử dụng padding và chế dộ nào không sử dụng padding? Với những chế dộ không sử dụng padding hãy giải thích vì sao chúng không sử dụng.**

- Các chế độ sử dụng padding là: ECB, CBC.

- Các chế độ không sử dụng padding là: CFB, OFB (và cả CTR).

- Giải thích: Vì CFB, OFB và CTR đều là mật mã dòng chúng mã hóa dữ liệu trong một luồng liên tục các bit, chứ không phải theo từng khối như block cipher. Các chế độ này tạo ra một dòng dữ liệu giả ngẫu nhiên có thể hoặc không phụ thuộc vào bản rõ, dòng giả ngẫu nhiên được tạo ra được XOR với bản rõ để tạo ra bản mã. Vì có thể sử dụng bao nhiêu bit của luồng ngẫu nhiên tùy thích nên không cần sử dụng padding vì quá trình mã hóa hoạt động độc lập trên mỗi bit hoặc byte của dữ liệu đầu vào, không cần đảm bảo rằng dữ liệu đầu vào phải là bội số của kích thước khối.

**3.2. Thực hành về padding:**

*- Giai đoạn 1:* Tạo 3 file chứa 5 bytes, 10 bytes, 16 bytes. Ít nhất một trong những file này có MSSV và tất cả có thể là file văn bản hoặc file nhị nhân

*A picture containing graphical user interface

Description automatically generated*

*Graphical user interface, text, application

Description automatically generated*

*Graphical user interface, application

Description automatically generated*

*Graphical user interface, text, application

Description automatically generated*

*Hình 17-20. Tạo 3 file có kích thước 5, 10, 16 bytes và nội dung của chúng*

*- Giai đoạn 2:* Sử dụng openssl enc -aes-128-cbc -e để mã hóa các tệp này bằng AES 128-bit với chế độ CBC. Mô mô tả kích thước của các file được mã hóa.

++ Key: aabbccddeeff1234567890987654321

++ IV: 09816647383458abcdef

Graphical user interface, text

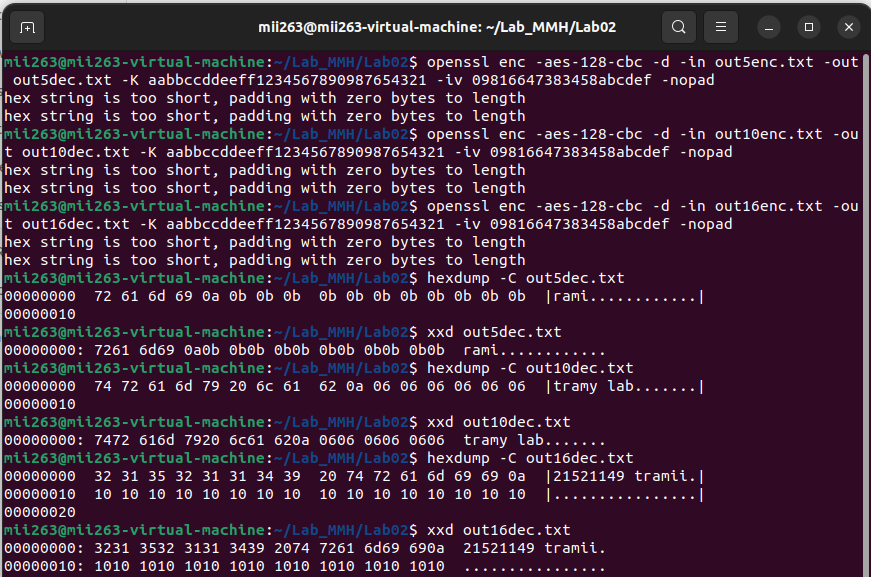
Description automatically generated

*A picture containing graphical user interface

Description automatically generated*

*Hình 21,22. Câu lệnh mã hoá và sự thay đổi kích thước của các file sau khi mã*

*- Giai đoạn 3:* Sử dụng openssl enc -aes-128-cbc -d để giải mã và -nopad để xem phần đệm

**

*Hình 23. Giải mã các file và hiển thị nội dung bằng công cụ hex*

1. **Error Propagation**

**- Có thể khôi phục bao nhiêu thông tin bằng cách giải mã tệp bị hỏng, nếu chế độ mã hóa lần lượt là ECB, CBC hoặc OFB?**

+ ECB: Có thể khôi phục các thông tin trong các khối không bị hỏng/lỗi.

+ CBC: Có thể khôi phục các thông tin trong các khối trước khối bị hỏng/lỗi.

+ OFB: Có thể khôi phục được một số phần trừ các bit tương ứng của ciphertext trong plaintext.

**- Thực hiện bài tập:**

*Bước 1:* Tạo một file văn bản dài ít nhất 1000 bytes.

**Graphical user interface, text, application

Description automatically generated**

*Hình 24. File văn bản dài 100 bytes*

*Bước 2:* Mã hoá tệp bằng mật mã AES-128.

++ Key: aabbccddeeff1234567890987654321

++ IV: 09816647383458abcdef

- Mã hoá file input.txt theo 3 mode ECB, CBC, OFB:

Text

Description automatically generated

*Hình 25. Lệnh thực thi mã hoá file input.txt theo 3 mode*

*Bước 3:* Giải sử 1 bit ở bytes thứ 55 file mã hoá bị lỗi/hỏng.

- Mở và thay thế lần lượt byte thứ 55 trong bless hex editor trong các mode:

+ EBC: đổi “89” thành “88”

Table

Description automatically generated

Table

Description automatically generated

*Hình 26, 27. Trước và sau khi thay đổi byte thứ 55*

+ CBC: đổi “96” thành “97”

Table

Description automatically generated

Table

Description automatically generated

*Hình 28, 29. Trước và sau khi thay đổi byte thứ 55*

+ OFB: đổi “7C” thành “6C”

Table

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

*Hình 30, 31. Trước và sau khi thay đổi byte thứ 55*

- Tiến hành giải mã lại các file:

Graphical user interface, text

Description automatically generated

*Hình 32. Giải mã các file*

+ Các kết quả giải mã:

++ ECB: Do byte thứ 55 nằm trong block thứ 4 (mỗi block là 16 bytes) nên khi byte thứ 55 bị thay đổi dẫn đến việc toàn bộ block thứ 4 bị thay đổi (từ byte thứ 49 đến 64).

Graphical user interface, text

Description automatically generated

*Hình 33. Giải mã ECB*

++ CBC: Do byte thứ 55 nằm trong block thứ 4 (mỗi block là 16 bytes) nên khi byte thứ 55 bị thay đổi dẫn đến việc toàn bộ block thứ 4 trở về sau bị thay đổi.

Text

Description automatically generated

*Hình 34. Giải mã CBC*

++ OFB: Đối với mode OFB gần giống như mode ECB vì quá trình giải mã sử dụng các block cipher độc lập để giải cho nên chỉ có dữ liệu trong block thứ 4 bị thay đổi.

Text

Description automatically generated

*Hình 35. Giải mã OFB*

- Giải thích:

+ ECB: Trong chế độ ECB, mỗi khối của plaintext được mã hóa độc lập với cùng một khóa, vì vậy nếu một khối bị hỏng, chỉ khối đó sẽ bị ảnh hưởng. Tuy nhiên, nếu nhiều khối bị hỏng, có thể khó hoặc không thể khôi phục lại dữ liệu gốc.

+ CBC: Trong chế độ CBC, mỗi khối của plaintext được XOR với khối ciphertext trước đó trước khi được mã hóa, điều này có nghĩa là bất kỳ sự hỏng hóc nào trong một khối sẽ ảnh hưởng đến tất cả các khối tiếp theo. Rất khó để khôi phục lại bất kỳ dữ liệu gốc nào kể từ khối bị hỏng.

+ OFB: Trong chế độ OFB, một keystream được tạo ra độc lập với plaintext và được sử dụng để mã hóa plaintext. Sự hỏng hóc trong ciphertext chỉ ảnh hưởng đến các bit tương ứng của plaintext đã giải mã, vì vậy có thể khôi phục được một số phần của dữ liệu gốc, nhưng không phải toàn bộ tệp.

1. **Programming using the Crypto Library**

- Code và các file text được gửi kèm theo báo cáo.

- Code chưa hoàn thiện (chưa giải quyết trường hợp encrypt với key và iv được nhập từ bàn phím/ chưa hỗ trợ tiếng việt / chưa giải quyết được mode CCM)

- Lập bảng của trường hợp Encrypt và mặc định là đọc file input và key cùng iv được tạo ngẫu nhiên:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | ECB | CBC | OFB | CFB | CTR | XTS | CCM |
| 9.8KB | 0.00017s | 0.00016s | 0.00015s | 0.00014s | 0.00014s | 0.00013s | - |
| 999KB | 0.00624s | 0.00627s | 0.00614s | 0.00592s | 0.00525s | 0.0058s | - |
| 97.6MB | 0.5747s | 0.64838s | 0.6682s | 0.64423s | 0.59235s | 1.07556s | - |

**Graphical user interface, text

Description automatically generated**

*Hình 36. Thời gian chạy trung bình của 100 lần chạy với file input 9.8KB*

**Text

Description automatically generated**

*Hình 37. Thời gian chạy trung bình của 100 lần chạy với file input 999KB*

**Text

Description automatically generated**

*Hình 38. Thời gian chạy trung bình của 100 lần chạy với file input 97.6MB*

**\*THAM KHẢO:**

1. Slide bài giảng, lab thực hành môn Mật mã học Trường ĐH Công nghệ Thông tin.

2. William Stallings. Cryptography and Network Security - Principles and Practice (7th Edition).

3. KattamuriMeghna, Block Cipher modes of Operation, link bài viết: <https://www.geeksforgeeks.org/block-cipher-modes-of-operation/>.

1. Ghi nội dung công việc, các kịch bản trong bài Thực hành [↑](#footnote-ref-1)