BÁO CÁO THỰC HÀNH

**Bài thực hành số 06: Review of Encryption Algorithms**

**Môn học:** An toàn mạng

1. **Lớp:** NT101.Q11.1

**THÀNH VIÊN THỰC HIỆN (Nhóm xx):**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** |
| 1 | Nguyễn Anh Tuấn | 23521717 |

|  |
| --- |
| **Điểm tự đánh giá** |
|  |

**ĐÁNH GIÁ KHÁC:**

|  |  |
| --- | --- |
| Tổng thời gian thực hiện |  |
| Phân chia công việc |  |
| Ý kiến *(nếu có)*  + Khó khăn  + Đề xuất, kiến nghị |  |

Phần bên dưới của báo cáo này là báo cáo chi tiết của nhóm thực hiện

MỤC LỤC

[**A.** BÁO CÁO CHI TIẾT 2](#_Toc216216471)

[1. Task 1: Caesar cipher: (1.5 điểm - 2 điểm) 2](#_Toc216216472)

[a. Ý tưởng 2](#_Toc216216473)

[b. Kết quả: 3](#_Toc216216474)

[2. Task 2: Mono-alphabetic substitution cipher: (1.5 điểm - 2 điểm) 3](#_Toc216216475)

[a. Ý tưởng: 3](#_Toc216216476)

[b. Kết quả: 5](#_Toc216216477)

[3. Task 3: Vigenère cipher (1.5 điểm - 2 điểm) 6](#_Toc216216478)

[Mã Vigenère: Là mã thay thế đa bảng (Polyalphabetic Substitution Cipher). Khác với mã đơn bảng, Vigenère sử dụng một khóa (ví dụ: "CRYPT") để thay thế chữ cái. Mỗi chữ cái của bản rõ được mã hóa bằng một chữ cái khóa khác nhau lặp đi lặp lại. 6](#_Toc216216479)

[a. Ý tưởng: 6](#_Toc216216480)

[b. Kết quả 8](#_Toc216216481)

[4. Task 4: DES — Cài đặt các chế độ hoạt động (2.0 điểm - 2.5 điểm) 8](#_Toc216216482)

[a. Ý tưởng: 8](#_Toc216216483)

[b. Kết quả 10](#_Toc216216484)

[5. Task 5: AES — Cài đặt các chế độ hoạt động (2.0 điểm - 2.5 điểm) 10](#_Toc216216485)

[Advanced Encryption Standard (AES) hay còn được gọi là tiêu chuẩn mã hóa nâng cao theo phương pháp mật mã khối. Với ưu thế bảo mật cao nó đã được chính phủ Hoa Kỳ lựa chọn để bảo vệ dữ liệu, thông tin cho các tổ chức, doanh nghiệp mà người dùng. 10](#_Toc216216486)

[a. Ý tưởng: 10](#_Toc216216487)

[b. Kết quả: 13](#_Toc216216488)

[**B.** TÀI LIỆU THAM KHẢO 13](#_Toc216216489)

# Link REPO: [LAB06](https://github.com/blastvious/NT101)

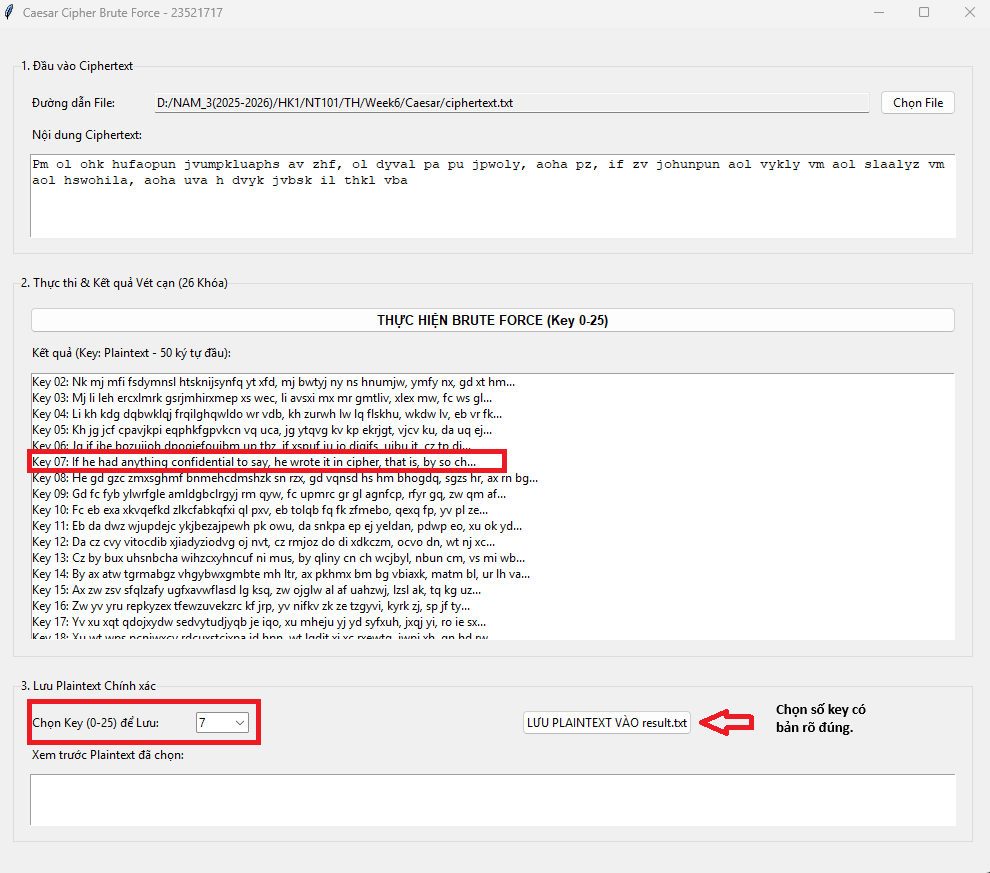
# BÁO CÁO CHI TIẾT

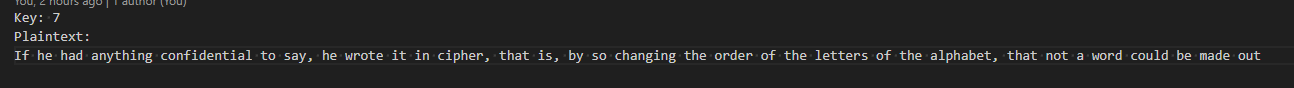
## Task 1: Caesar cipher: (1.5 điểm - 2 điểm)

### Ý tưởng

Thuật toán mã hóa Caesar có ý tưởng là thay thế mỗi ký tự trong văn bản gốc bằng môt ký tự khác, cách nó một số bước cố định trong bảng chữ cái (26). Vậy ý tưởng để crack mã hóa này ta cũng sẽ lặp 26 lần từ a - > z sau đó sẽ từ danh sách giải mã chọn mã hóa phù hợp.

### Kết quả:





## Task 2: Mono-alphabetic substitution cipher: (1.5 điểm - 2 điểm)

### Ý tưởng:

**Bước 1:** phân tích tần số xuất hiện của từng letter trong ciphertext.txt.

**Bước 2:**

Dựa vào các bảng COMMON\_WORDS, BIGRAM\_DATA\_RAW, TRIGRAM\_DATA\_LIST

**Bước 3:**

* Sử dụng các thuật toán Simulated Annealing (SA), và State (mapping) dùng để ánh xạ giữa 26 ký tự của bản mã và 26 ký tự của bản rõ.
* Neighbor một trạng thái láng giềng được tạo ra bằng cách hoán đổi giữa 2 vị trí chữ cái bất kỳ trong ánh xạ hiện tại.

**Bước 4:** Hàm đánh giá (Fitness Funtion) dùng để đo mức độ mà bản rõ giống TA nhất.

* **Nguyên tắc:** Các chuỗi ký tự phổ biến (N-gram) trong Tiếng Anh (như **TH, HE, AND, THE**) phải xuất hiện thường xuyên trong bản rõ được giải mã, tương ứng với xác suất thống kê.
* **Công thức:** Hàm $calculate\\_fitness$ sử dụng tổng logarit xác suất của các **bigram** và **trigram** xuất hiện trong bản rõ:



Trong đó:

* PNgram là xác suất của N-gram tương ứng trong Tiếng Anh.
* **Trọng số W2GRAM=0.4, W3GRAM=0.6):** Dùng để điều chỉnh mức độ quan trọng của bigram và trigram (trigram thường có tính phân biệt cao hơn).
* **Chuẩn hóa:** Kết quả được chia cho độ dài bản mã (N) để tránh thiên vị bản mã dài hơn.

**Bước 5: Cơ chế Chấp nhận và Lịch Giảm Nhiệt**

Thuật toán lặp qua các bước sau:

1. Tính thay đổi điểm (Delta Score): Delta Score = Scoreneighbor – Scorecurrent.
2. Chấp nhận láng giềng:
   * Nếu Delta Score > 0 (láng giềng tốt hơn): Luôn chấp nhận.
   * Nếu Delta Score <= 0 (láng giềng tệ hơn): Chấp nhận với xác suất:

**A black text on a white background

AI-generated content may be incorrect.Công thức:**

Trong đó, T là nhiệt độ hiện tại.

1. Giảm nhiệt: Nhiệt độ T được giảm dần theo mỗi lần lặp theo công thức:

Công thức Tnew = Told x CoolingRate

Tham số Nhiệt độ ban đầu (initial\_temp = 10.0) và Tốc độ làm lạnh (cool\_rate = 0.9999). Tốc độ làm lạnh chậm giúp thuật toán tìm kiếm kỹ

Đây là toàn bộ ý tưởng cho bài này

### Kết quả:

A screenshot of a computer

AI-generated content may be incorrect.

A black background with many small colored lines

AI-generated content may be incorrect.

Kết quả thu được gần đúng 99% đang nhầm lân giũa X và Y. Yêu cầu để giải chính xác phải là một ciphertext trên 5000 từ.

## Task 3: Vigenère cipher (1.5 điểm - 2 điểm)

### Mã Vigenère: Là mã thay thế đa bảng (Polyalphabetic Substitution Cipher). Khác với mã đơn bảng, Vigenère sử dụng một khóa (ví dụ: "CRYPT") để thay thế chữ cái. Mỗi chữ cái của bản rõ được mã hóa bằng một chữ cái khóa khác nhau lặp đi lặp lại.

### Ý tưởng:

Mặc dù Vigenère là đa bảng, việc lặp lại khóa khiến nó thực chất là $L$ mã Caesar lồng vào nhau, trong đó L là độ dài khóa.

**Bước 1: Ước tính Độ dài Khóa L bằng Index of Coincidence (IC)**

* + - **Nguyên tắc:** Nếu chúng ta nhóm các chữ cái bản mã được mã hóa bởi **cùng một chữ cái khóa** (nghĩa là cách nhau $L$ vị trí), chuỗi con đó chỉ là một **mã Caesar** (mã đơn bảng) của bản rõ.
      * Tần suất chữ cái của một mã Caesar vẫn giữ nguyên hình dạng, chỉ bị dịch chuyển, do đó IC của nó vẫn cao (gần IC\_english ~ 0.666).
      * Ngược lại, tần suất tổng hợp của Vigenere bị làm phẳng, IC của nó thấp gần (0.0385).
      * Thực hiện hàm find\_key\_length:
        + **Giả định độ dài khóa** Ltest = 1, 2, 3, …., MAX\_KEY\_LENGTH.
        + **Chia bản mã** thành Ltest chuỗi con C0, C1, …, CLtest-1) bằng cách lấy các ký tự cách nhau Ltest vị trí (cipher\_alpha[i::L\_test]).
        + **Tính IC trung bình:** Tính IC cho từng chuỗi con và lấy trung bình.
        + **Lựa chọn:** Độ dài Ltest nào cho IC trung bình **gần ICEnglish nhất** sẽ là độ dài khóa chính xác.

**A mathematical equation with numbers and symbols

AI-generated content may be incorrect.Công thức:**

**Bước 2: Tìm từng ký tự Khóa bằng Chi-Squared Text**

* + - Sau khi xác định được độ dài khóa $L$, chúng ta tìm **độ dịch chuyển** (shift) cho từng chuỗi con Ci độc lập.
      * **Nguyên tắc:** Mã Caesar được giải mã chính xác khi phân bố tần suất chữ cái trong bản rõ giải mã giống nhất với phân bố tần suất chuẩn của Tiếng Anh.
      * **Kiểm tra Chi-Squared (X^2):** Công cụ thống kê này đo lường sự khác biệt giữa tần suất quan sát được Oi và tần suất lý thuyết Ei của Tiếng Anh. Giá trị X^2 càng nhỏ, bản rõ giải mã càng giống Tiếng Anh.

**A mathematical equation with numbers and symbols

AI-generated content may be incorrect.**

**Công thức:**

* + - **Thực hiện (Hàm find\_key):**
      * **Lặp qua mỗi chuỗi con** Ci (từ i=0 đến L - 1).
      * **Thử 26 lần dịch chuyển** (stest=0 đến 25, tương ứng với khóa 'A' đến 'Z').
      * **Giải mã giả định** Ci bằng stest để tạo ra Pi, stest.
      * **Tính X^2** cho Pi, stest với tần suất Tiếng Anh chuẩn ENGLISH\_FREQ).
      * **Lựa chọn:** Độ dịch chuyển stest cho X^2 **nhỏ nhất** là ký tự khóa (key character) chính xác cho vị trí i.

**Bước 3: Giải mã và Xác nhận Cuối cùng**

* + - Giải mã (Hàm vigenere\_decrypt): Sau khi tìm được khóa k (ví dụ: "CRYPT"), toàn bộ bản mã được giải mã, áp dụng phép trừ modulo 26.

**Công thức:**

* + - **Xác nhận (Hàm validate\_plaintext\_with\_words):** Để tăng độ tin cậy, bản rõ giải mã được kiểm tra bằng một heuristic: đếm **tỷ lệ xuất hiện** các từ phổ biến trong Tiếng Anh (như *the, and, to*).
      * Tỷ lệ từ phổ biến cao (> 0.05) là bằng chứng mạnh mẽ rằng khóa và bản rõ là chính xác.

### Kết quả

A screenshot of a computer

AI-generated content may be incorrect.

## Task 4: DES — Cài đặt các chế độ hoạt động (2.0 điểm - 2.5 điểm)

**DES** là một thuật toán mã hóa khối (Block Cipher) hoạt động trên các khối dữ liệu 64-bit và sử dụng khóa 56-bit (khóa đầu vào 64-bit, nhưng 8 bit là bit kiểm tra chẵn lẻ, không dùng trong mã hóa).

### Ý tưởng:

Dựa vào gợi ý của LAB

Ta triển khai các yếu tố và hàm quan trọng

* + - Quản lý Khóa Vòng (Key Schedule): Khóa 64-bit đầu vào được xử lý để tạo ra 16 khóa vòng (Round Keys) K1 đến K16, mỗi khóa có độ dài 48-bit.

Thể hiện qua hàm generate\_round\_keys(key64\_bits)

* + - * **PC-1 (Permuted Choice 1):** Khóa 64-bit được hoán vị và giảm xuống 56-bit bằng cách loại bỏ các bit kiểm tra chẵn lẻ.
      * **Chia và Dịch vòng:** Khóa 56-bit được chia thành hai nửa C và D (mỗi nửa 28-bit).
      * **Dịch trái (Shift Left):** Trong mỗi vòng, C và D được dịch trái một lượng bit xác định bởi mảng SHIFT (1 hoặc 2 bit).
      * **PC-2 (Permuted Choice 2):** Kết hợp C và D sau khi dịch, sau đó thực hiện hoán vị và giảm từ 56-bit xuống **48-bit** để tạo thành khóa vòng Ki.
      * Hàm Feistel Cốt lõi (Feistel Function f())
      * Hàm f là trái tim của DES, nơi sự nhầm lẫn (confusion) và lan truyền (diffusion) được tạo ra. Nó ánh xạ nửa phải của khối 32-bit (R) và khóa vòng 48-bit (Ki) thành đầu ra 32-bit.
      * **XOR:** Kết quả 48-bit được XOR với khóa vòng Ki (48-bit)
      * **S-box (Substitution):** Kết quả XOR (48-bit) được chia thành 8 khối 6-bit. Mỗi khối 6-bit đi vào một S-box riêng.
      * 6 bit đầu vào được dùng để xác định **hàng** (bit 1 và bit 6) và **cột** (4 bit giữa).
      * S-box thay thế 6 bit đầu vào bằng 4 bit đầu ra (đầu ra tổng cộng là 8 x 4 = 32 bit). Đây là thành phần **phi tuyến tính** duy nhất của DES, tạo ra sự nhầm lẫn.
      * **P-box (Permutation):** Kết quả 32-bit từ S-box được hoán vị bằng bảng $P$. Mục đích là xáo trộn các bit, tăng độ khuếch tán.
    - **Chế độ hoạt động của Block Cipher Modes**
      * Hàm des\_encrypt(mode=’<mode>’):
        + ECB

Mã hóa từng khối 64 – bit một cách độc lập với cùng một khóa.

**Nhược điểm:** Mẫu lặp trong plaintext sẽ tạo ra mẫu lập trong ciphertext, khiến nó dễ bị tấn công.

* + - * + **CBC**

Mỗi khối bản rõ được XOR với khối bản mã trước đó (hoặc Vector IV) trước khi mã hóa.

Điều này đảm bảo cùng một khối bản rõ sẽ tạo ra các kối bản mã khác nhau, khắc phục nhuọc điểm của ECB. Cần IV ngẫu nhiên.

* + - **Xử lý dữ liệu và Padding**
      * Đệm PKCS#7: Vì DES hoạt động trên khối 8-byte (64-bit), bản rõ được đệm bằng hàm pad và loại bỏ đệm bằng unpad (đảm bảo độ dài dữ liệu là bội số của 8).
      * **Chuyển đổi Dữ liệu:** Các hàm tiện ích như bytes\_to\_bitstring, bitstring\_to\_bytes, hex\_to\_bitstring được sử dụng để chuyển đổi dữ liệu giữa các định dạng **bytes**, **chuỗi bit**, và **hex/base64**.

### Kết quả

A screenshot of a computer

AI-generated content may be incorrect.

## Task 5: AES — Cài đặt các chế độ hoạt động (2.0 điểm - 2.5 điểm)

### Advanced Encryption Standard (AES) hay còn được gọi là tiêu chuẩn mã hóa nâng cao theo phương pháp mật mã khối. Với ưu thế bảo mật cao nó đã được chính phủ Hoa Kỳ lựa chọn để bảo vệ dữ liệu, thông tin cho các tổ chức, doanh nghiệp mà người dùng.

### Ý tưởng:

Mã nguồn triển khai một hệ thống mã hóa AES hoàn chỉnh, được chia thành 4 phần chính:

* **Hằng số & Bảng tra (Constants & Look-up Tables):** Chứa các hằng số tiêu chuẩn của AES (như BLOCK\_SIZE, S\_BOX, INV\_S\_BOX, RCON).
* **Các Hàm Cơ Sở của AES (Core Functions):** Triển khai các bước xử lý cơ bản trong mỗi vòng của AES: gmul (phép nhân GF(2^8)), SubBytes, ShiftRows, MixColumns, AddRoundKey, và các hàm nghịch đảo tương ứng.
* **Mở rộng Khóa & Thuật toán Khối (Key Expansion & Block Algorithm):** Triển khai thuật toán mở rộng khóa (key\_expansion) và logic mã hóa/giải mã cho **một khối 16 byte** (aes\_encrypt\_block, aes\_decrypt\_block).
* **Chế độ Hoạt động & Padding (Modes of Operation & Padding):** Triển khai việc đóng gói dữ liệu (padding PKCS#7) và xử lý dữ liệu nhiều khối theo các chế độ **ECB** và **CBC** (encrypt, decrypt).

**1. Phép Toán trong Trường Galois (GF($2^8$))**

Hàm cốt lõi: gmul(a, b)

* **Ý tưởng:** AES sử dụng các phép toán số học trên trường hữu hạn **GF(2^8)** (còn gọi là trường Galois). Phép nhân này khác với phép nhân số học thông thường, nó được thực hiện modulo một đa thức bất khả quy là x^8 + x^4 + x^3 + x + 1 (tương đương với 0x1B).
* **Chức năng:** Hàm gmul thực hiện phép nhân a. b (mod m(x)), nơi mà a và b là các byte 8-bit. Việc này là **bắt buộc** cho bước **MixColumns** và **InvMixColumns**.

**2. Mở Rộng Khóa (Key Expansion)**

Hàm cốt lõi: key\_expansion(key)

* **Ý tưởng:** Thuật toán AES cần nhiều **khóa vòng (Round Keys)**, mỗi khóa dài 16 byte, được tạo ra từ khóa gốc (key). Quá trình này được gọi là Key Expansion.
* **Cơ chế:**
  + Khóa gốc được chia thành các **Word (4 byte)**.
  + Các Word mới được tạo ra bằng cách XOR Word trước đó với Word cách nó **Nk** vị trí (với Nk là số Word của khóa gốc, ví dụ: Nk=4 cho AES-128).
  + Cứ mỗi **Nk** Word, một phép biến đổi phức tạp (bao gồm RotWord, SubWord dùng S\_BOX, và XOR với hằng số vòng RCON) được áp dụng lên Word trước đó để đảm bảo tính khuếch tán và chống lại các cuộc tấn công liên quan đến khóa.
  + Hàm trả về một danh sách các khóa vòng, bao gồm Khóa gốc và N\_r Khóa vòng (với N\_r là số vòng: 10, 12, hoặc 14).

**3. Các Bước Biến Đổi Khối (AES Cipher Block)**

Các bước này biến đổi một ma trận trạng thái 4 x 4 (tương đương 16 byte).

**Mã hóa (Encrypt)**

* **SubBytes:** Thay thế mỗi byte trong ma trận trạng thái bằng giá trị tương ứng trong **S-Box**. Điều này cung cấp tính phi tuyến tính cho thuật toán.
* **ShiftRows:** Xoay vòng các hàng của ma trận trạng thái. Hàng 0 không xoay, hàng 1 xoay trái 1 byte, hàng 2 xoay trái 2 byte, hàng 3 xoay trái 3 byte. Điều này cung cấp tính khuếch tán theo hàng.
* **MixColumns:** Nhân ma trận trạng thái (các cột) với một ma trận cố định trong trường GF(2^8). Điều này cung cấp tính khuếch tán theo cột (ảnh hưởng lẫn nhau giữa các cột).
* **AddRoundKey:** XOR ma trận trạng thái với Khóa vòng hiện tại.

**Giải mã (Decrypt)**

Quá trình giải mã là nghịch đảo của mã hóa, thực hiện các bước theo thứ tự ngược lại và sử dụng các phép biến đổi nghịch đảo:

* InvShiftRows (Xoay phải)
* InvSubBytes (Dùng **Inv-S-Box**)
* InvMixColumns (Dùng ma trận nhân nghịch đảo)
* AddRoundKey (Phép XOR là nghịch đảo của chính nó)

**4. Padding PKCS#7**

Hàm cốt lõi: pkcs7\_pad(data) và pkcs7\_unpad(data)

* **Ý tưởng:** AES chỉ xử lý các khối 16 byte. Nếu dữ liệu đầu vào không phải là bội số của 16 byte, cần phải thêm vào các byte đệm (padding) để đảm bảo độ dài là bội số.
* **Cơ chế PKCS#7:** Số byte đệm được thêm vào bằng chính giá trị độ dài của padding đó. Ví dụ: Nếu cần 5-byte đệm, sẽ thêm 0x05, 0x05, 0x05, 0x05, 0x05. Hàm pkcs7\_unpad sẽ kiểm tra byte cuối cùng để xác định độ dài padding và loại bỏ chúng.

**5. Chế độ Hoạt động (Modes of Operation)**

Hàm cốt lõi: encrypt, decrypt

* + - **ECB (Electronic Codebook)**
* **Cơ chế:** Mỗi khối Plaintext được mã hóa độc lập với các khối khác bằng cùng một Khóa vòng

Công thức:

* **Ưu điểm:** Đơn giản, tốc độ cao, có thể song song hóa.
* **Nhược điểm:** **Không an toàn** cho dữ liệu có cấu trúc lặp lại. Nếu hai khối Plaintext giống nhau, hai khối Ciphertext tương ứng sẽ **giống nhau**, tiết lộ mẫu lặp lại.
  + - **CBC (Cipher Block Chaining)**
* **Cơ chế (Mã hóa):** Mỗi khối Plaintext được XOR với khối Ciphertext **trước đó** trước khi mã hóa. Khối đầu tiên sử dụng một **Vector Khởi tạo (IV)** ngẫu nhiên.

****

**Công thức:**

* **Cơ chế (Giải mã):** Khối Ciphertext được giải mã, sau đó kết quả được XOR với khối Ciphertext **trước đó** để khôi phục Plaintext.
* ****

**Công thức:**

* **Ưu điểm:** **An toàn hơn ECB**. Việc sử dụng IV và "xích" các khối với nhau đảm bảo rằng cùng một Plaintext, khi mã hóa 2 lần (hoặc 2 khối giống nhau), sẽ tạo ra các Ciphertext khác nhau.
* **Nhược điểm:** Cần IV (phải được truyền đi cùng Ciphertext) và không thể song song hóa quá trình mã hóa.

### Kết quả:

A screenshot of a computer

AI-generated content may be incorrect.

* + - **Lưu ý:**

**Ở bày này chỉ mới triển khai được phần mã hóa, giải mã đang bị bug.**

# TÀI LIỆU THAM KHẢO

Bài 3: Vigenere [Index of Coincidence](https://www.youtube.com/watch?v=kty-dCB4AAk&t=250s)

Bài 5: [AES (Advanced Encryption Standard)](https://www.youtube.com/watch?v=LiFcxxRRLSk&t=947s)