FINAL REPORT

CRYPTOGRAPHY

June 11TH, 2025

—

Multimedia Product Service Platform

—

Instructor: Nguyễn Ngọc Tự

—

Group's member:

—

Nguyễn Đình Hưng – 23520564
Trần Quang Huy – 23520648
Trịnh Nhật Duy – 23520394

Lời cảm ơn

Để hoàn thành bài báo cáo này, chúng em đã nhận được sự hỗ trợ và hướng dẫn quý báu từ nhiều nguồn tài liệu và cá nhân. Chúng em xin chân thành cảm ơn những đóng góp đã giúp chúng em có cái nhìn sâu sắc hơn về các khái niệm mật mã học, kiến trúc mạng và cách thức triển khai bảo mật trong thực tế.

Đặc biệt, chúng em xin gửi lời cảm ơn sâu sắc đến giảng viên môn mật mã học của chúng em, thầy **Nguyễn Ngọc Tự** đã cung cấp những kiến thức nền tảng vững chắc, những bài giảng tâm huyết và định hướng nghiên cứu quý báu. Nhờ sự hướng dẫn tận tình, chúng em đã có thể tiếp cận và phân tích sâu hơn về cơ chế mã hóa truyền nhận video của các nền tảng trực tuyến hàng đầu, cũng như xây dựng và đánh giá hiệu quả của mô hình mã hóa AES_256 do nhóm chúng em triển khai.

Những kiến thức và kinh nghiệm thu được trong quá trình này là vô cùng giá trị, góp phần không nhỏ vào sự thành công của đồ án môn học.

Mục lục

DANH MỤC HÌNH ẢNH, BIỂU ĐỒ

I. INTRODUCTION

1. Bối cảnh

Trong thời đại các dịch vụ truyền thông đa phương tiện thời gian thực ngày càng phát triển mạnh mẽ, như hội nghị trực tuyến, livestream hay VoIP, việc đảm bảo tính bảo mật cho luồng dữ liệu truyền đi trở thành một yêu cầu thiết yếu.
Các giải pháp mã hóa truyền thống như AES (Advanced Encryption Standard) rất hiệu quả với dữ liệu tĩnh nhưng thường không phù hợp cho các ứng dụng thời gian thực do độ trễ mã hóa.

Do đó, nhóm chúng em đã đề xuất thực hiện một đề tài xây dựng hệ thống mã hóa luồng, kết hợp giữa Chaotic Stream Cipher (dùng cho việc mã hóa dữ liệu thời gian thực như video/audio, nhằm giảm độ trễ) và AES-256 dùng cho bảo mật dữ liệu lưu trữ, metadata và các thông tin nhạy cảm). Dự án sẽ phát triển một nguyên mẫu (prototype) gồm client và server, đảm bảo các nguyên lý bảo mật thông tin như confidentiality, integrity, availability và authenticity.

2. Tài sản, Bên liên quan và Chủ sở hữu

Tài sản (Asset)

Các tài sản chính (Core Assets):

1. **Nội dung Video:**
    - **Định nghĩa:** Các file video gốc, các phiên bản đã được mã hóa (AES-256), và các phiên bản đã được giải mã.
    - **Tại sao quan trọng:** Đây là tài sản chính cần được bảo vệ khỏi việc truy cập trái phép, sửa đổi, hoặc sao chép.
2. **Khóa mã hóa (Encryption Keys):**
    - **Định nghĩa:** Khóa AES-256 được sử dụng để mã hóa và giải mã video.
    - **Tại sao quan trọng:** Nếu khóa bị lộ, quá trình mã hóa trở nên vô nghĩa. Đây là một tài sản cực kỳ nhạy cảm.

Các tài sản hỗ trợ (Supporting Assets):

3. **Dữ liệu người dùng (User Data):**
    - **Định nghĩa:** Thông tin đăng nhập (nếu có), lịch sử xem, cài đặt cá nhân của người dùng trên hệ thống client.
    - **Tại sao quan trọng:** Bảo vệ quyền riêng tư và ngăn chặn truy cập tài khoản trái phép.
4. **Cơ sở hạ tầng phần cứng và phần mềm (Hardware & Software Infrastructure):**
    - **Server:** Máy chủ lưu trữ video, thực hiện mã hóa và truyền tải.
    - **Client:** Ứng dụng/thiết bị của người dùng để nhận và giải mã/phát video.
    - **Mã nguồn ứng dụng:** Mã nguồn của server và client.
    - **Tại sao quan trọng:** Đảm bảo hệ thống hoạt động ổn định, không bị tấn công làm gián đoạn dịch vụ hoặc lợi dụng để truy cập tài sản khác.
5. **Kênh truyền tải mạng (Network Communication Channel):**
    - **Định nghĩa:** Kết nối giữa server và client (có thể là HTTP/HTTPS).
    - **Tại sao quan trọng:** Kênh truyền tải là nơi dữ liệu dễ bị đánh chặn, cần được bảo vệ bằng TLS/SSL.

Asset Owners (Chủ sở hữu tài sản)

1. **Nhóm phát triển/Người tạo nội dung (Content Creators):**

- **Định nghĩa:** Chính nhà sáng tạo nội dung và các thành viên trong nhóm. Ví dụ, chúng ta là người tạo ra video, mã hóa nó và triển khai hệ thống.
- **Trách nhiệm:** Quyết định chính sách bảo mật cho video, đảm bảo khóa được quản lý an toàn, kiểm soát truy cập vào server và mã nguồn.

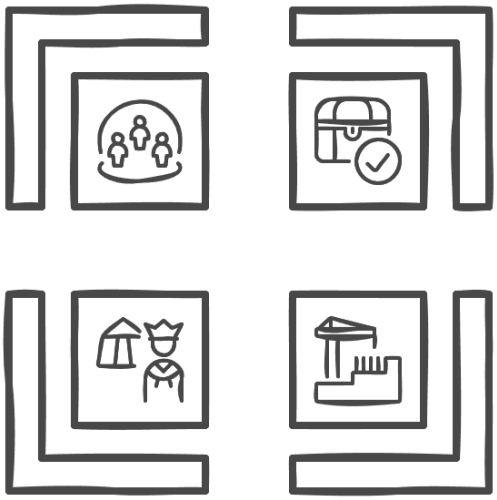
2. **Người dùng cuối (End-Users/Viewers):**
    - **Định nghĩa:** Những người sử dụng ứng dụng client để xem video đã giải mã.
    - **Trách nhiệm:** Kiểm soát quyền riêng tư của dữ liệu cá nhân (nếu có), đảm bảo thiết bị của họ an toàn để nhận và giải mã video. Trong một hệ thống thực tế, họ là chủ sở hữu của *quyền truy cập* nội dung chứ không phải nội dung đó.

Related Stakeholders (Các bên liên quan)

1. **Người dùng cuối (End-Users/Viewers):** (Lặp lại từ Owners)
    - **Vai trò:** Tiêu thụ nội dung, cung cấp phản hồi, và là đối tượng mà hệ thống bảo mật hướng đến (đảm bảo họ xem được nội dung an toàn).
2. **Các bên cung cấp dịch vụ (Service Providers) - nếu có:**
    - **Ví dụ:** Nếu sử dụng các dịch vụ hosting (VPS, Cloud) để triển khai server, nhà cung cấp dịch vụ Internet (ISP).
    - **Vai trò:** Cung cấp cơ sở hạ tầng mạng và máy chủ, đảm bảo kết nối ổn định và an toàn (ở cấp độ của họ).

Các bên liên quan

Các bên liên quan như người dùng cuối và nhà cung cấp dịch vụ.



Tài sản chính

Các tài sản quan trọng như nội dung video và khóa mã hóa cần được bảo vệ.

Chủ sở hữu tài sản

Các cá nhân, nhóm sáng tạo tài sản, người dùng(viewer)

Tài sản hỗ trợ

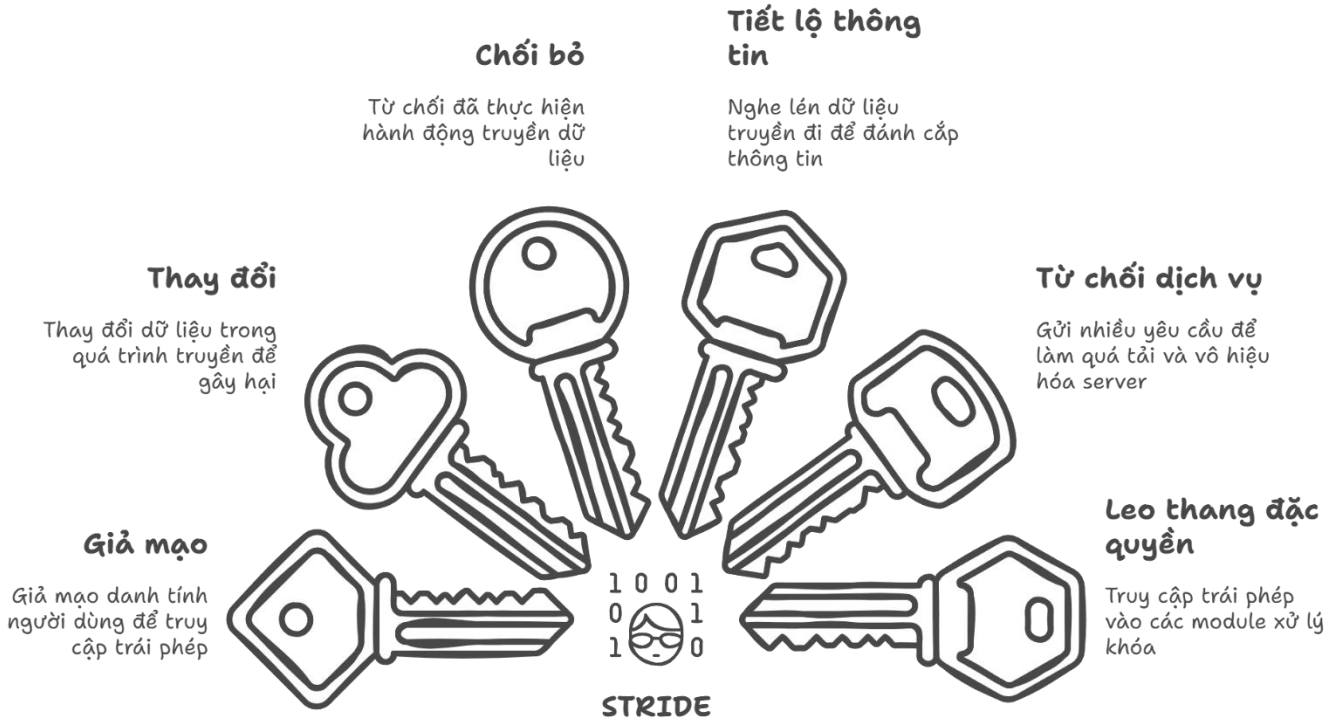
Các yếu tố hỗ trợ như dữ liệu người dùng và cơ sở hạ tầng phần cứng.

Biểu đồ 1. Assets, owner assets and related stockholders

3. Phân tích rủi ro bảo mật

Phân tích rủi ro bắt đầu bằng việc xác định tài sản và các điểm truy cập vào chúng. Các điểm truy cập chính bao gồm socket mạng, file cấu hình, và giao diện nhập liệu của người dùng. Ranh giới tin cậy được thiết lập giữa client và server, cũng như giữa dữ liệu đã mã hóa và dữ liệu thô.

- **Mô hình hóa mối đe dọa (STRIDE)**
  - **(1) S - Spoofing (Giả mạo)**
    - (i) Kẻ tấn công giả mạo danh tính người dùng.
  - **(2) T - Tampering (Thay đổi)**
    - (i) Thay đổi dữ liệu trong quá trình truyền.
  - **(3) R - Repudiation (Chối bỏ)**
    - (i) Từ chối đã thực hiện hành động truyền dữ liệu.
  - **(4) I - Information Disclosure (Tiết lộ thông tin)**
    - (i) Nghe lén dữ liệu truyền đi.
  - **(5) D - Denial of Service (Từ chối dịch vụ)**
    - (i) Gửi nhiều yêu cầu khiến server ngừng hoạt động.
  - **(6) E - Elevation of Privilege (Leo thang đặc quyền)**
    - (i) Truy cập trái phép vào module xử lý khóa.



Hình 1. Mô hình STRIDE

➢ **Đánh giá rủi ro**

Đánh giá rủi ro	Khả năng xảy ra	Mức độ ảnh hưởng	Ưu tiên xử lý
Man-in-the-middle	**Trung bình**	**Cao**	**Cao**
Replay attack	**Thấp**	**Trung bình**	**Trung bình**
Key leakage	**Thấp**	**Cao**	**Cao**

➢ **Biện pháp giảm thiểu**

- Sử dụng kênh truyền an toàn (TLS hoặc SSH) cho các luồng điều khiển.
- Áp dụng token có thời hạn để ngăn chặn replay attack.
- Quản lý khóa chỉ trong bộ nhớ RAM, không ghi xuống đĩa.

4. Mục tiêu bảo mật và kiến trúc giải pháp

Mục tiêu Bảo mật (CIA Mở rộng)

Mục tiêu	Nội dung
Confidentiality	Bảo mật nội dung dữ liệu qua AES và Chaotic Cipher.
Integrity	Đảm bảo dữ liệu không bị sửa đổi qua authenticated encryption (AES-GCM).
Availability	Hệ thống truyền dữ liệu ổn định, đáp ứng thời gian thực.
Authenticity	Xác thực người dùng qua hệ thống đăng nhập và quản lý phiên.
Non-repudiation	Ghi log đầy đủ các sự kiện nhằm ngăn chặn việc từ chối hành vi.

Kiến trúc Giải pháp

Thuật toán và giao thức mật mã:

- **AES-256 (GCM):** Bảo vệ thông tin người dùng, metadata.
- **Chaotic Stream Cipher:** Logistic map hoặc Chebyshev map tạo keystream XOR với dữ liệu.
- **Key derivation:** PBKDF2 từ mật khẩu.
- **Thư viện:** pycryptodome, socket, hashlib.

Client	Server
• Chaotic Cipher engine. • AES encryption cho metadata. • Giao diện xác thực. • Module truyền video/audio.	• AES/Chaotic decryption engine. • Module xác thực và ghi log.

Quản lý khóa:

- AES/Chaotic decryption engine.
- Module xác thực và ghi log.

1) CLIENT tải trang demo:

- GET / → server trả về templates/index.html
- Browser tải index.html, parse và thực thi script

2) TẢI ASSET (Client-side):

- GET /static/js/script.js → tải logic decrypt + play
- GET /static/wasm/chaotic_wasm.js → tải wrapper WASM
- GET /static/wasm/chaotic_wasm.wasm → tải binary WASM

3) KHỞI TẠO RSA (Client-side):

- script.js gọi registerPubKey('sample1') → sinh cặp RSA-OAEP
- Client POST public key PEM → /register_pubkey/sample1
- Server nhận → lưu file static/pubkeys/sample1.pem

4) LẤY AES-KEY (Client-side):

- Client GET /get_key_rsa/sample1?token=abc123
- Server:
  - Đọc public key PEM của client
  - Đọc raw AES-key từ static/keys/sample1.key
  - Mã hóa AES-key bằng RSA-OAEP (SHA-256)
  - Trả JSON { key_rsa_b64 }
- Client nhận Base64, giải RSA decrypt → raw AES-key
- Client importKey → CryptoKey AES-GCM

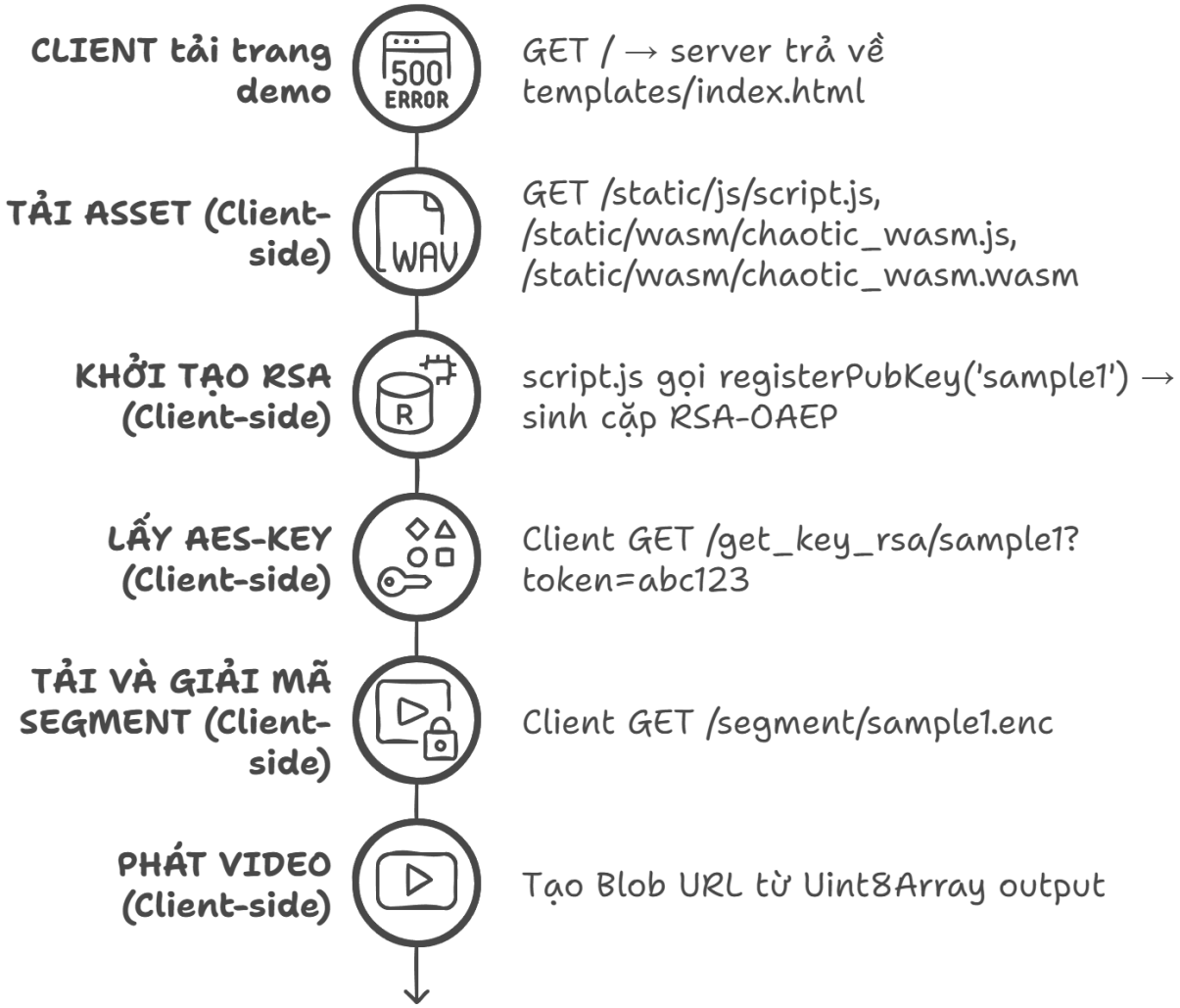
5) TẢI VÀ GIẢI MÃ SEGMENT (Client-side):

- Client GET /segment/sample1.enc
- Server trả file .enc (gồm IV||TAG||CIPHERTEXT)
- Client đọc ArrayBuffer:
  - AES-GCM decrypt bằng crypto.subtle
  - Sinh keystream chaotic từ WASM
  - XOR với plaintext để phục hồi video gốc

6) PHÁT VIDEO (Client-side):

• Tạo Blob URL từ Uint8Array output

• Gán vào <video.src> và gọi video.play()



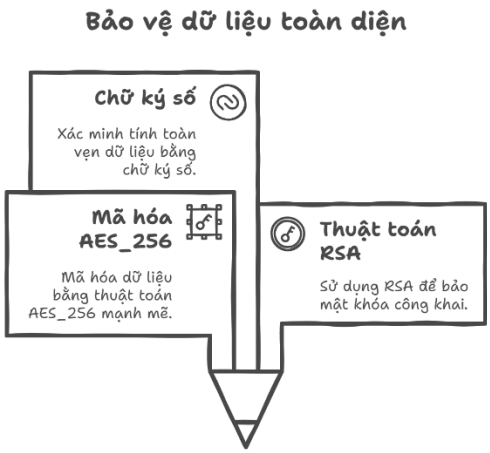
Biểu đồ 2. Sơ đồ luồng hoạt động

II. SOLUTION ARCHITECTURE

1. Các thuật toán mã hóa sử dụng trong đồ án

Thuật toán mã hóa khối AES_256, Chaotic Map
Thuật toán mật mã khóa công khai RSA
Tạo và xác minh chữ ký số.

Dựa trên những phân tích về rủi ro và giải pháp đã nêu trên, nhóm chúng em đã quyết định sử dụng thuật toán Chaotic Map và AES_256 để thực hiện việc mã hóa video. Để có thể trao đổi khóa một cách an toàn, chúng em chọn thuật toán về Mã hóa Bất đối xứng (Asymmetric Cryptography) RSA. Tiếp theo đó, để an toàn hơn trong việc truyền dữ liệu, nhóm chúng em đã chọn dùng chữ ký số để mã hóa đường truyền HTTPS đã thực hiện ở bài Lab.



2. Thuật toán mã hóa khối AES_256

Nguyên lý hoạt động cơ bản của AES:

AES hoạt động trên các khối dữ liệu có kích thước **128 bit** (16 byte), được biểu diễn dưới dạng một ma trận 4x4 gồm các byte, gọi là **State**. Khóa mã hóa có thể có độ dài **128, 192, hoặc 256 bit**, và độ dài khóa sẽ quyết định số lượng vòng lặp (rounds) trong quá trình mã hóa:

- Khóa 128 bit: 10 vòng
- Khóa 192 bit: 12 vòng
- Khóa 256 bit: 14 vòng

Mỗi vòng mã hóa trong AES bao gồm một chuỗi các phép biến đổi đơn giản nhưng được kết hợp lại để tạo ra hiệu ứng phức tạp (confusion và diffusion).

Các bước chính trong quá trình mã hóa AES:

1. **Chuyển đổi bản rõ thành State:**
    - Bản rõ 128 bit được đưa vào và sắp xếp thành một ma trận 4x4 byte gọi là State.
2. **Mở rộng khóa (Key Expansion):**
    - Từ khóa chính (master key), một thuật toán mở rộng khóa sẽ tạo ra các khóa con (round keys) cho mỗi vòng. Mỗi khóa con có kích thước 128 bit.
3. **Vòng khởi tạo (Initial Round): AddRoundKey**
    - Trước khi vào các vòng lặp chính, ma trận State được XOR với khóa con đầu tiên (tạo từ khóa chính).
    - Phép toán này kết hợp các byte của State với các byte của khóa con một cách tuyến tính.
4. **Các vòng lặp chính (Main Rounds - Nr-1 vòng):** Mỗi vòng lặp (trừ vòng cuối cùng) bao gồm 4 phép biến đổi tuần tự:
    - **SubBytes (Thay thế Byte):**
        - Đây là một phép thay thế phi tuyến tính, cung cấp tính "confusion".
        - Mỗi byte trong ma trận State được thay thế bằng một byte khác dựa trên một bảng tra cố định (S-box) 8x8 bit. S-box này được tạo ra từ nghịch đảo trên trường hữu hạn GF(28) và một phép biến đổi affine. Điều này làm cho mối quan hệ giữa bản rõ và bản mã trở nên phi tuyến tính và phức tạp.
    - **ShiftRows (Dịch hàng):**
        - Đây là phép hoán vị bit, cung cấp tính "diffusion" (khuếch tán) theo chiều ngang.
        - Các hàng của ma trận State được dịch vòng sang trái một số vị trí nhất định:
            - Hàng 0 (đầu tiên): Giữ nguyên (dịch 0 vị trí).

        - Hàng 1: Dịch vòng trái 1 byte.
        - Hàng 2: Dịch vòng trái 2 byte.
        - Hàng 3: Dịch vòng trái 3 byte.
    - Điều này đảm bảo rằng các byte từ cùng một cột ban đầu sẽ được phân tán ra các cột khác nhau, làm cho mỗi cột đầu ra của bước tiếp theo phụ thuộc vào tất cả các cột đầu vào.
- **MixColumns (Trộn cột):**
    - Đây là một phép biến đổi tuyến tính, cung cấp thêm tính "diffusion" theo chiều dọc.
    - Mỗi cột của ma trận State được xem là một đa thức trên trường hữu hạn GF(28) và được nhân với một ma trận cố định.
    - Kết quả là mỗi byte mới trong cột phụ thuộc vào tất cả 4 byte của cột đó trước khi trộn.
- **AddRoundKey (Cộng khóa vòng):**
    - Ma trận State được XOR với khóa con tương ứng của vòng hiện tại.
    - Phép XOR này đơn giản và hiệu quả, kết hợp khóa vào dữ liệu đã được biến đổi.

5. **Vòng cuối cùng (Final Round - Vòng thứ Nr):**
    - Vòng cuối cùng hơi khác một chút: nó không thực hiện phép biến đổi **MixColumns**.
    - Các bước trong vòng cuối cùng là: **SubBytes**, **ShiftRows**, và **AddRoundKey**.
6. **Đầu ra:**
    - Ma trận State cuối cùng sau vòng cuối cùng chính là bản mã 128 bit.

Quá trình giải mã AES:

Quá trình giải mã trong AES cũng bao gồm các bước tương tự nhưng theo thứ tự ngược lại và sử dụng các phép biến đổi nghịch đảo (inverse operations) của từng bước, cùng với các khóa con theo thứ tự đảo ngược.

- Inverse AddRoundKey
- Inverse MixColumns (trừ vòng cuối cùng)
- Inverse ShiftRows
- Inverse SubBytes

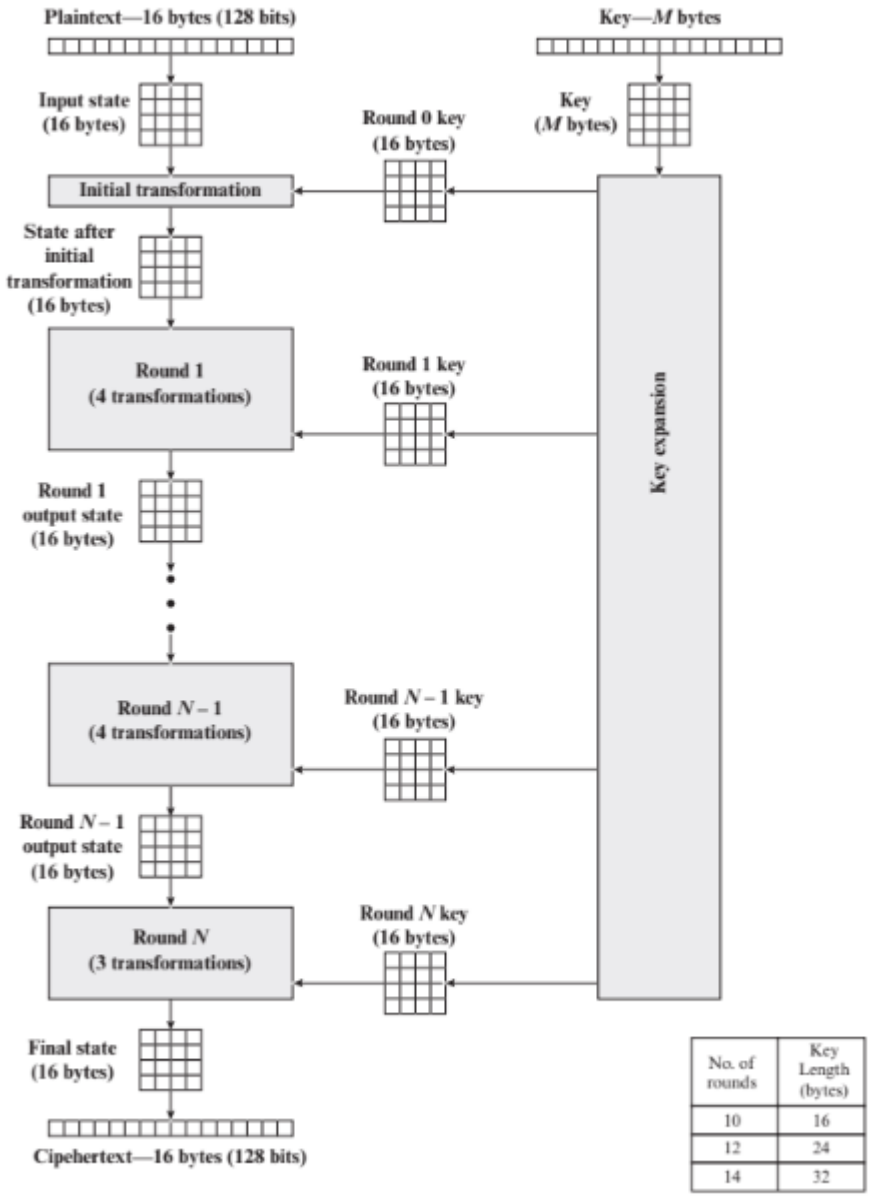


Figure 6.1 AES Encryption Process

Hình 2. Mã hóa AES_126

3. Thuật toán mật mã khóa công khai RSA

Nguyên lý hoạt động của RSA

RSA là một trong những thuật toán mật mã khóa công khai đầu tiên và được sử dụng rộng rãi nhất, được đặt tên theo tên của ba nhà khoa học đã phát minh ra nó: Ron **R**ivest, Adi **S**hamir, và Leonard **A**dleman vào năm 1977.

Sức mạnh của RSA dựa trên bài toán toán học khó khăn là **phân tích thừa số nguyên tố của các số nguyên lớn (integer factorization problem)**. Rất dễ dàng để nhân hai số nguyên tố lớn để có được một số lớn, nhưng cực kỳ khó khăn (tốn rất nhiều thời gian tính toán) để đảo ngược quá trình đó – tức là tìm ra hai số nguyên tố ban đầu khi chỉ biết tích của chúng.

Các bước chính trong RSA:

RSA bao gồm ba giai đoạn: **Tạo khóa, Mã hóa, và Giải mã.**

1. Tạo khóa (Key Generation)

Đây là bước quan trọng nhất và tạo ra cặp khóa công khai và khóa riêng.

- **Bước 1: Chọn hai số nguyên tố lớn (p và q)**
    - Chọn hai số nguyên tố ngẫu nhiên, phân biệt, rất lớn, p và q.
    - Kích thước của p và q (thường là hàng trăm chữ số) quyết định độ an toàn của khóa. Ví dụ, nếu p và q có 1024 bit, thì khóa RSA sẽ có 2048 bit.
- **Bước 2: Tính n (modulus)**
    - Tính $n=p\times q$.
    - n là thành phần công khai của cả khóa công khai và khóa riêng.
- **Bước 3: Tính $\phi(n)$ (Hàm Euler's Totient Function)**
    - Tính $\phi(n)=(p-1)\times(q-1)$.
    - $\phi(n)$ là số lượng các số nguyên dương nhỏ hơn n và nguyên tố cùng nhau với n.
    - Giá trị $\phi(n)$ này **phải được giữ bí mật** vì nó được sử dụng để tìm khóa riêng.
- **Bước 4: Chọn số mũ mã hóa e (public exponent)**
    - Chọn một số nguyên e sao cho:
        - $1<e<\phi(n)$
        - e và $\phi(n)$ là nguyên tố cùng nhau (tức là ƯCLN$(e,\phi(n))=1$).
    - Một số giá trị phổ biến cho e là 3,17,65537.
- **Bước 5: Tính số mũ giải mã d (private exponent)**
    - Tính d sao cho:

        - $d \times e \equiv 1 (\text{mod} \phi(n))$
    - d là nghịch đảo nhân của e modulo $\phi(n)$. d có thể được tìm thấy bằng thuật toán Euclid mở rộng.
- **Kết quả:**
    - **Khóa công khai (Public Key):** (e,n)
    - **Khóa riêng (Private Key):** (d,n) (hoặc (d,p,q) để tối ưu hóa việc giải mã)

2. Mã hóa (Encryption)

Giả sử Alice muốn gửi tin nhắn M (plaintext) cho Bob.

- Alice lấy **khóa công khai của Bob**: (e,n).
- Chuyển tin nhắn M thành một số nguyên (M) sao cho $0 \leq M < n$. (Nếu M lớn hơn n, nó sẽ được chia thành nhiều khối).
- Alice tính bản mã C (ciphertext) bằng công thức: $C = M^e (\text{mod} n)$
- Alice gửi C cho Bob.

3. Giải mã (Decryption)

Bob nhận bản mã C từ Alice.

- Bob sử dụng **khóa riêng của chính mình**: (d,n).
- Bob tính bản rõ M bằng công thức: $M = C^d (\text{mod} n)$
- Bob đã khôi phục được tin nhắn M ban đầu.

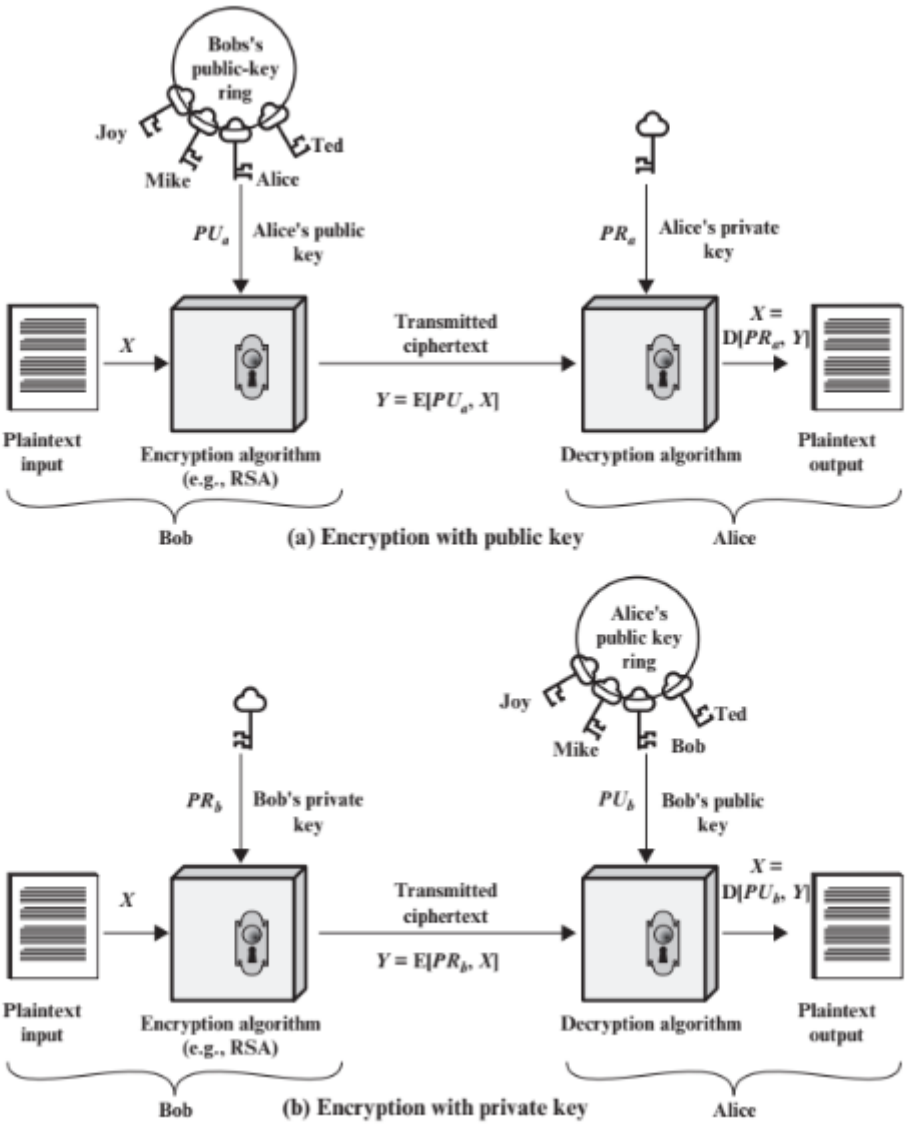
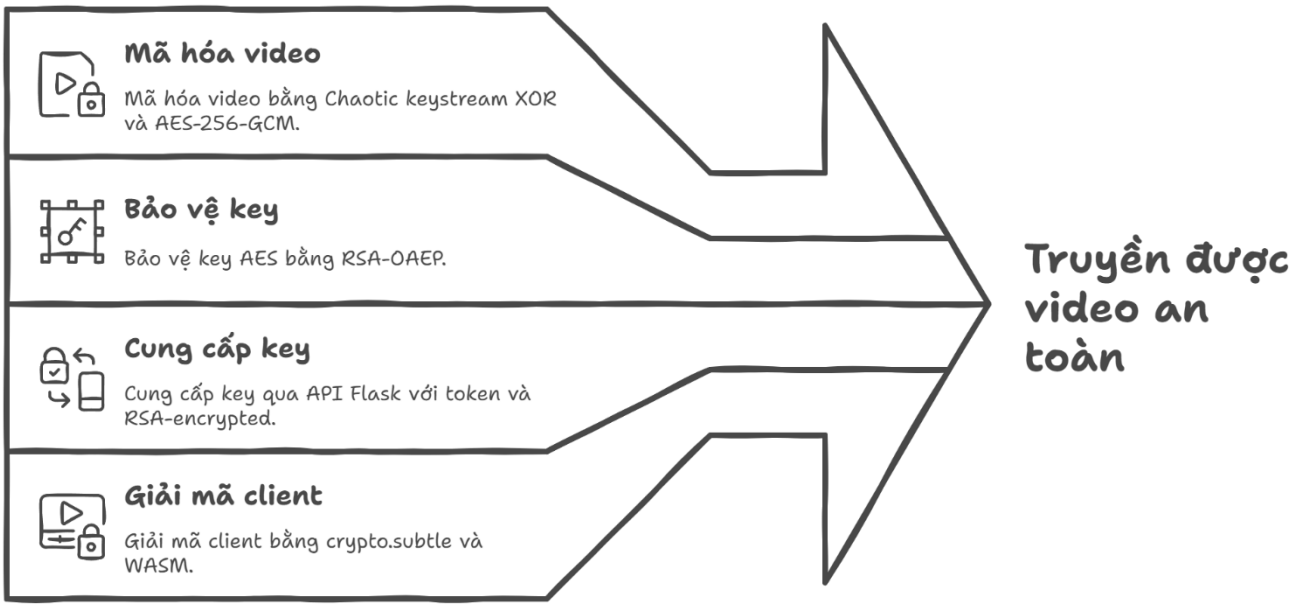


Figure 9.1 Public-Key Cryptography

Hình 3.Trao đổi khóa công khai RSA

4. Mục tiêu đặt ra

- Mã hóa video sử dụng Chaotic keystream XOR và AES-256-GCM
- Bảo vệ key AES bằng RSA-OAEP: server mã hóa key với public key client, client giải mã bằng private key
- Cung cấp key từ server Flask qua API /get_key_rsa có token và RSA-encrypted
- Giải mã tại client bằng crypto.subtle (RSA + AES) và WASM để phát lại video

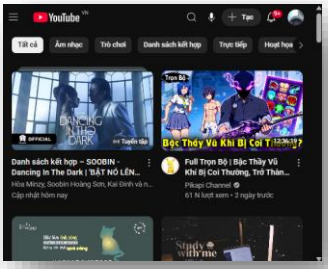


III. IMPLEMENTATION

1. Kế hoạch tìm hiểu phân tích và triển khai

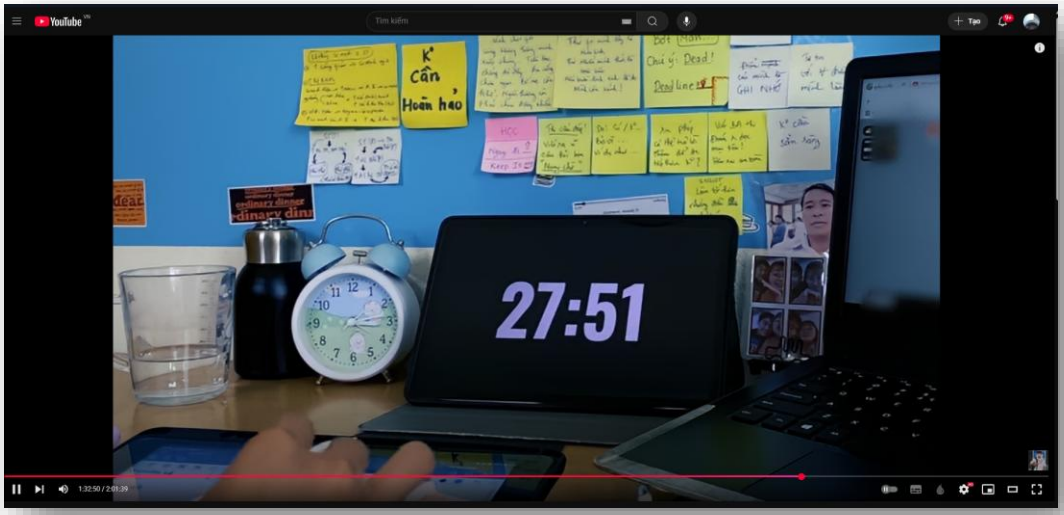
Tìm hiểu về các thuật toán mà kênh truyền thông lớn Youtube mã hóa, truyền và nhận video.
Hiện thực đồ án với mô hình demo client – server.

Dựa trên những lý thuyết đã học và nghiên cứu thực hành. Nhóm chúng em sẽ tiến hành nghiên cứu, tìm hiểu các thuật toán mã hóa và truyền dữ liệu của Youtube với khả năng của nhóm. Sau đó hiện thực đồ án để mô phỏng lại những thuật toán đã học, sau đó so sánh với mô hình của Youtube.



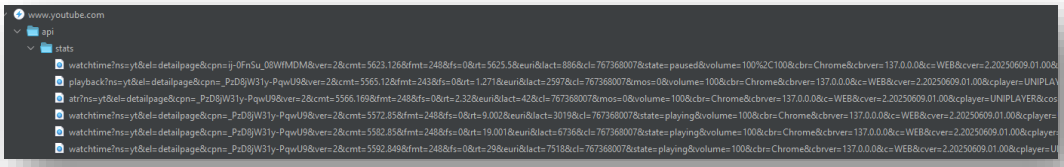
2. Tìm hiểu về Youtube với CharlesProxy

Bước 1: Chúng ta truy cập một trang Youtube để xem một video bất kỳ.



Hình 4. Mở một video Youtube bất kỳ.

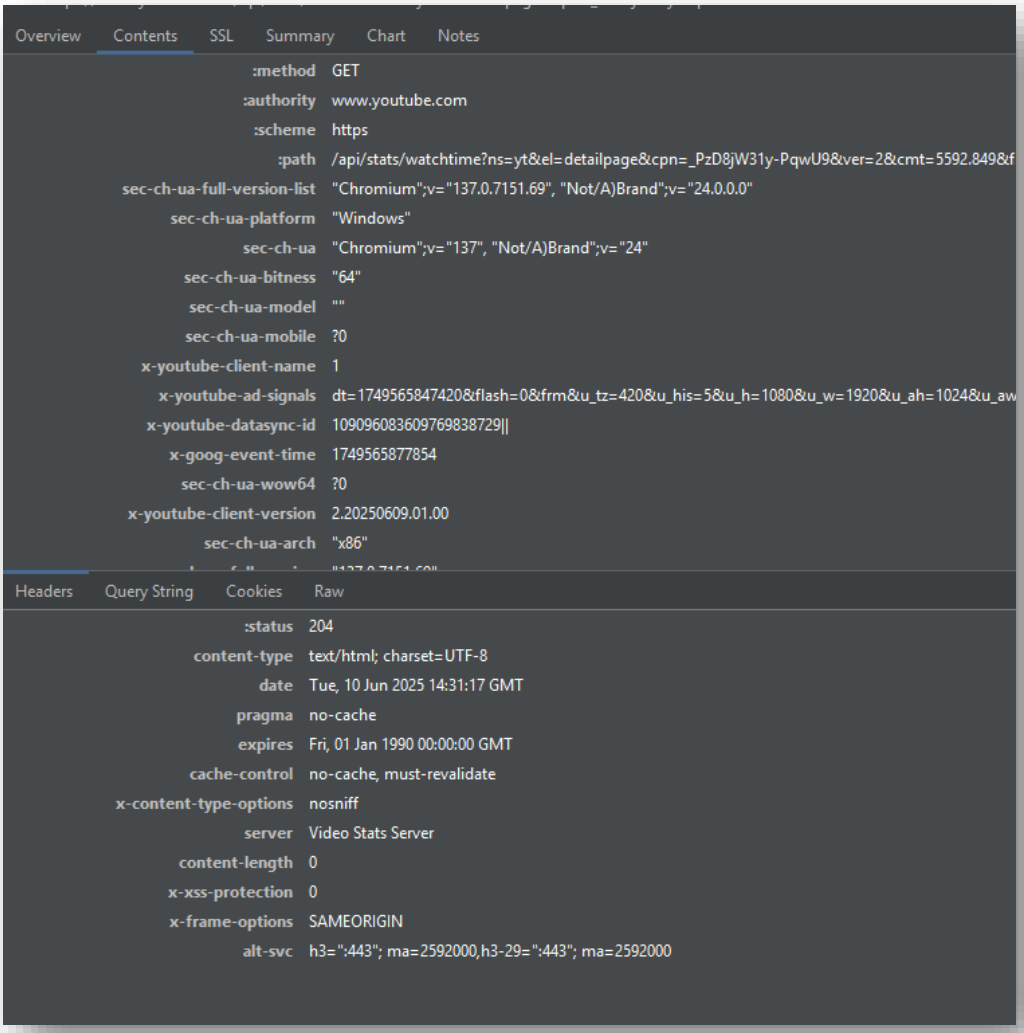
Bước 2: Truy cập CharlesProxy để xem các lưu lượng mạng của URL Youtube. Ở đây, ta có thể thấy được, cứ cách một khoảng thời gian nhất định, Youtube sẽ gửi một yêu cầu cho Server của Youtube.



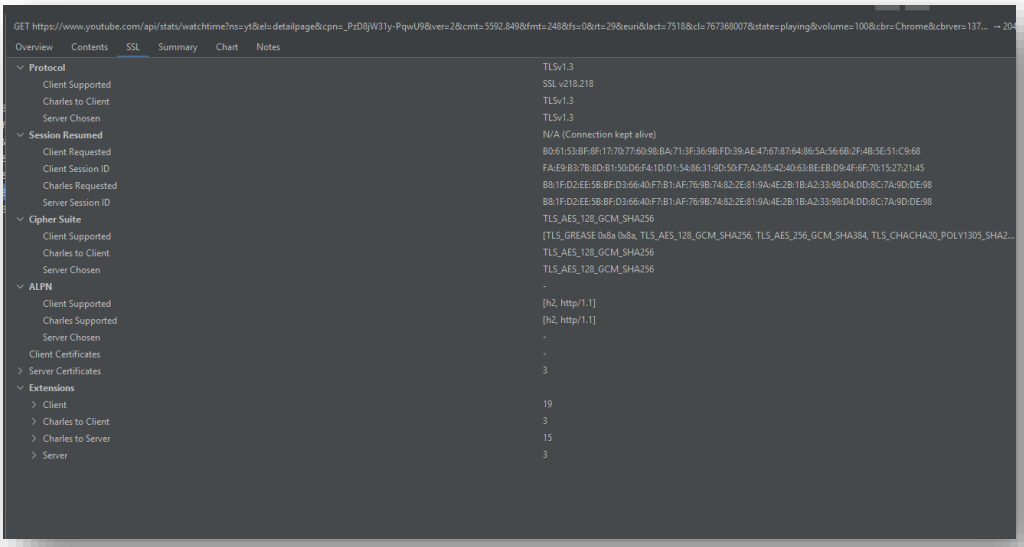
Hình 5. Truy cập CharlesProxy và Enable Proxying cho URL của Youtube

Toàn bộ URL request như sau:
https://www.youtube.com/api/stats/watchtime?ns=yt&el=detailpage&cpn=_PzD8jW31y-PqwU9&ver=2&cmt=5592.849&fmt=248&fs=0&rt=29&euri&lact=7518&cl=767368007&state=playing&volume=100&cbr=Chrome&cbrver=137.0.0.0&c=WEB&cver=2.20250609.01.00&cplayer=UNIPLAYER&cos=Windows&cosver=10.0&cplatform=DESKTOP&hl=vi_VN&cr=VN&uga=m24&len=7299.001&rtn=71&afmt=251&idpj=-3&ldpj=-1&rti=29&st=5582.85&et=5592.849&muted=0&vis=10&docid=1YaSvQqwPpI&ei=lkFIaJaHI4TZ1d8PtMqQ4Aw&plid=AAY3OIrMXcBvhEqG&of=BYCh469UR_dSvfqRtKB5tQ&vm=CAEQABgEOjJBSHFpSlRLby1qU2syakEyWTZBMWNHYTR3WG96WFFwR2ZDSlBPbmdCalVDRE5BNmV0Z2JYQUZVQTZSVDVRcnRROXVKeWhZYUVQMmZxN2FselBpTnp2a0k2NGRoODB0bnE4akkzYjVWMm03aDQ5MExTZm11SEdZTFVIM0JVcXBzbmpFeUlpUE1FcGFLMQ

Bước 3: Để hiểu rõ hơn nó là gì, chúng ta sẽ dùng CharlesProxy để giải mã kết nối TLS (sử dụng TLS_AES_128_GCM_SHA256). Điều này cho phép chúng ta thấy các HTTP header và URL.

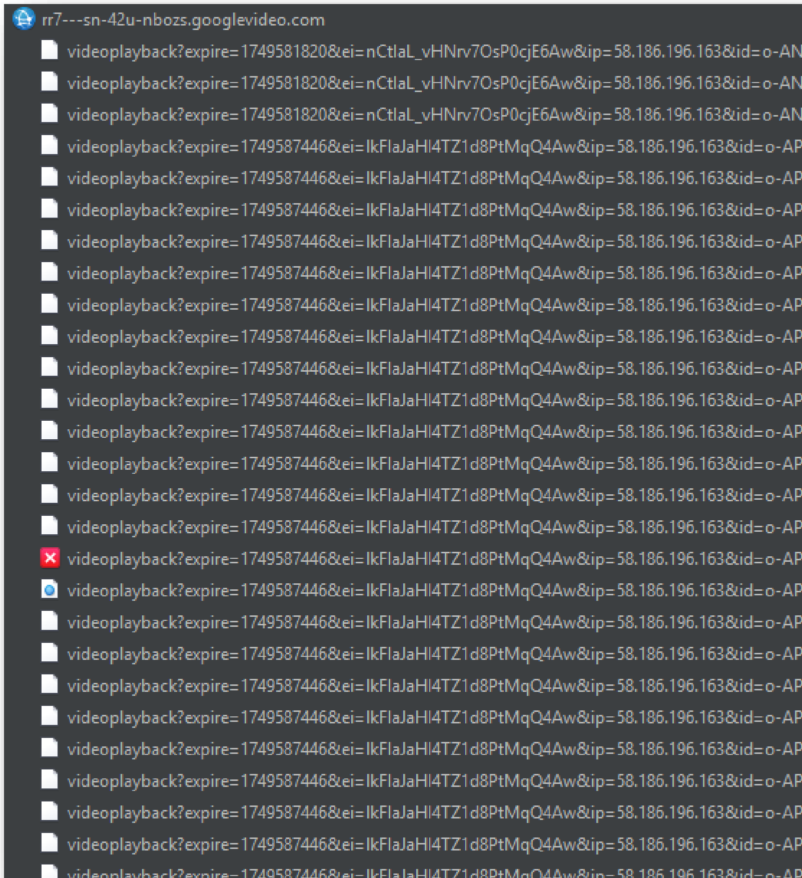


Hình 6. Chọn một gói tin bất kỳ và xem được Header

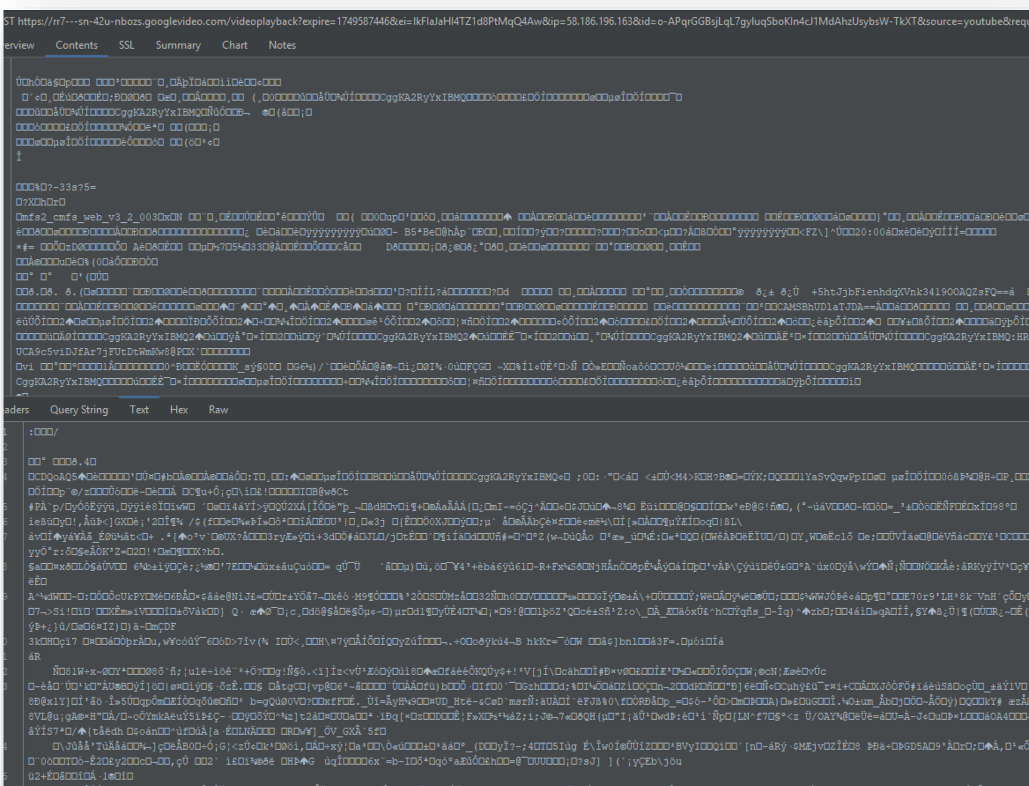


Hình 7. Chứng chỉ SSL của Youtube (TLS_AES_128_GCM_SHA256)

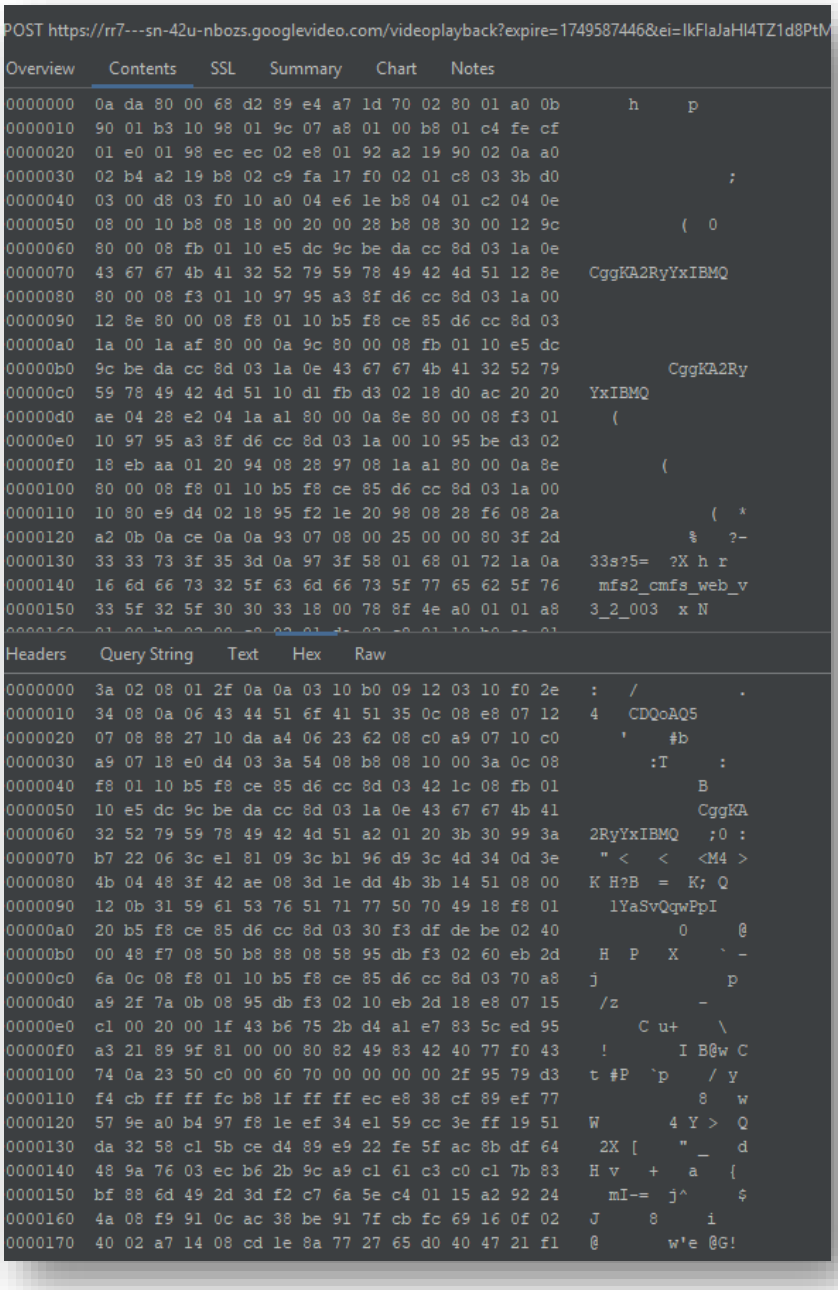
Ta có thể thấy ở đây, các loại mã hóa mà Youtube sử dụng.
Bước 4: Sau đó, chúng ta sẽ thử tìm hiểu một gói tin của Youtube.



Hình 8. Các segment nhận được mỗi 3s



Hình 9. Không thể xem được nội dung vì không có Key



Hình 10. Kiểu Hex

Cứ mỗi 3s, sẽ có một request về Youtube với phương thức POST, tuy nhiên chúng ta không thể giải mã được vì không có Key.
Tổng kết, sau khi tìm hiểu về các thức truyền, nhận video của Youtube và các tài liệu chúng em có góc nhìn như sau:

Về thuật toán mã hóa nội dung video của YouTube

1. **Hệ thống DRM (Digital Rights Management):**
   - YouTube, giống như hầu hết các dịch vụ streaming lớn (Netflix, Disney+, Spotify, v.v.), sử dụng các hệ thống **DRM** để bảo vệ nội dung của họ. DRM không phải là một thuật toán mã hóa duy nhất mà là một bộ các công nghệ bao gồm mã hóa, quản lý khóa, xác thực thiết bị và kiểm soát quyền truy cập.
   - YouTube tích hợp chặt chẽ với **Widevine DRM**, một công nghệ DRM thuộc sở hữu của Google. Widevine được sử dụng rộng rãi trên các trình duyệt Chrome, Android, và nhiều thiết bị thông minh khác. Các hệ thống DRM khác bao gồm PlayReady của Microsoft và FairPlay của Apple.
2. **Mã hóa nội dung thực tế (Content Encryption):**

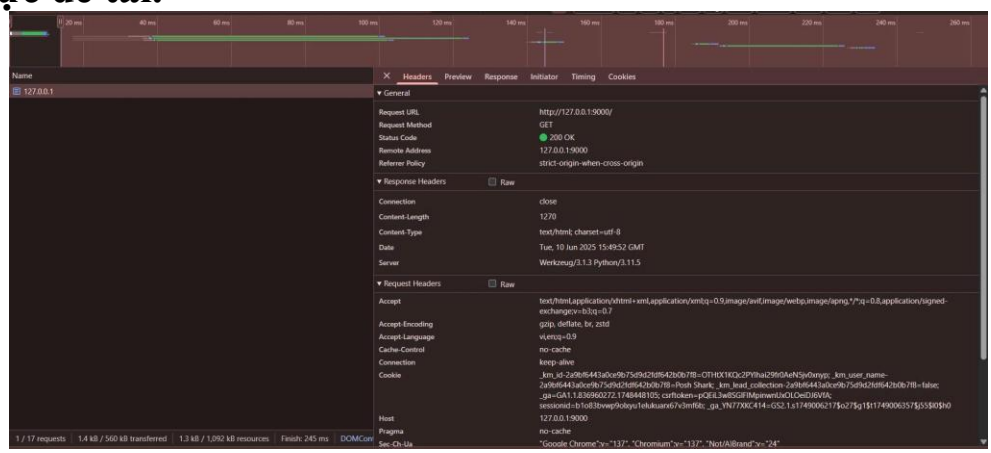
- Các hệ thống DRM như Widevine thường sử dụng **AES-128** (Advanced Encryption Standard với khóa 128 bit) làm thuật toán mã hóa cho bản thân dữ liệu video/âm thanh.
- Tuy nhiên, điểm quan trọng là **khóa giải mã (decryption key)** không được gửi cùng với video mà được quản lý và phân phối một cách rất an toàn thông qua một "license server".

3. **Cách thức hoạt động cơ bản (Đơn giản hóa):**
   - Khi ta bắt đầu xem video, trình phát (player) trên thiết bị của user sẽ gửi một yêu cầu tới **license server** của Widevine (thường là một dịch vụ backend của Google).
   - Yêu cầu này bao gồm thông tin về thiết bị của user, phiên bản DRM, và một "thử thách" mã hóa.
   - License server sẽ xác thực thiết bị và quyền truy cập của user(ví dụ: user có đăng ký premium không, nội dung này có được phép xem ở vùng của user không).
   - Nếu mọi thứ hợp lệ, license server sẽ gửi lại **khóa giải mã (content key)** cho thiết bị của user. Khóa này thường được mã hóa bằng một khóa khác (key encryption key) mà chỉ thiết bị của user mới có thể giải mã được.
   - Trình phát trên thiết bị sau đó sử dụng khóa giải mã đó để giải mã các phân đoạn video đã được mã hóa bằng AES-128 mà nó tải về từ googlevideo.com.

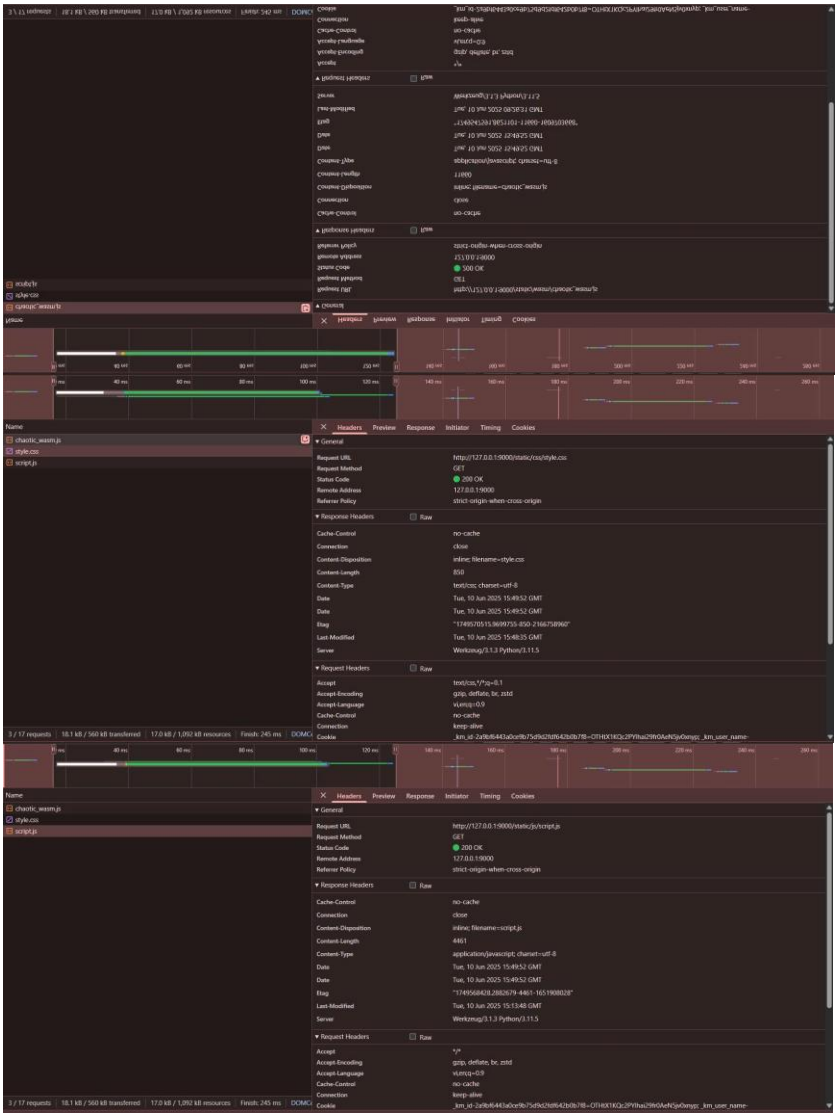
Tại sao chúng ta không thể thấy khóa hoặc nội dung được giải mã bằng Charles Proxy?

- **Charles Proxy giải mã lớp HTTPS (TLS/SSL), không phải lớp DRM.** Như chúng đã thấy trong ảnh **Hình 10**, Charles đã giải mã thành công kết nối TLS (sử dụng TLS_AES_128_GCM_SHA256). Điều này cho phép chúng ta thấy các HTTP header và URL.
- **Dữ liệu video vẫn bị mã hóa bởi DRM.** Các byte mà chúng ta thấy trong **Hình 10** (0a da 01 00 ...) là dữ liệu video đã được mã hóa bằng Widevine (thường là AES-128), **trước khi** nó được truyền qua kênh HTTPS.
- **Khóa giải mã không đi qua Charles Proxy một cách rõ ràng.** Khóa giải mã được trao đổi giữa thiết bị của user và license server thông qua các giao thức bảo mật riêng của DRM, và nó được thiết kế để không thể bị chặn hoặc trích xuất dễ dàng bởi các công cụ proxy. Ngay cả khi chúng ta thấy các yêu cầu tới license server, payload của chúng thường được mã hóa hoặc là định dạng nhị phân độc quyền, khiến chúng ta không thể hiểu được.

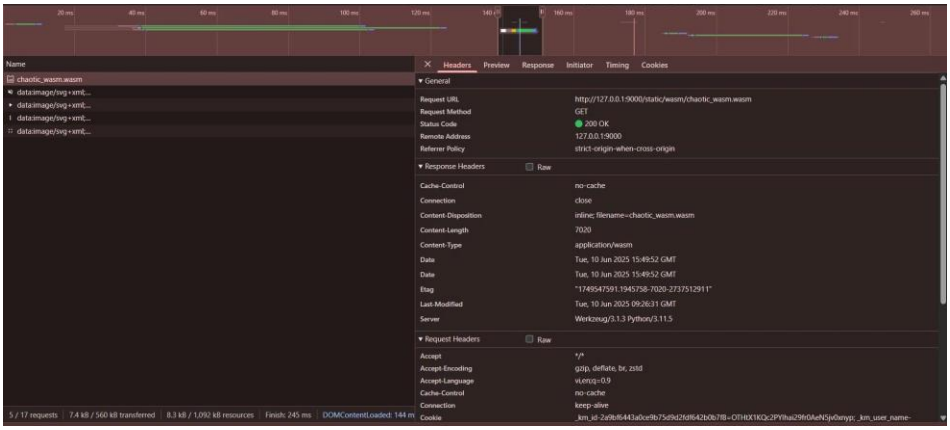
3. Hiện thực đề tài.



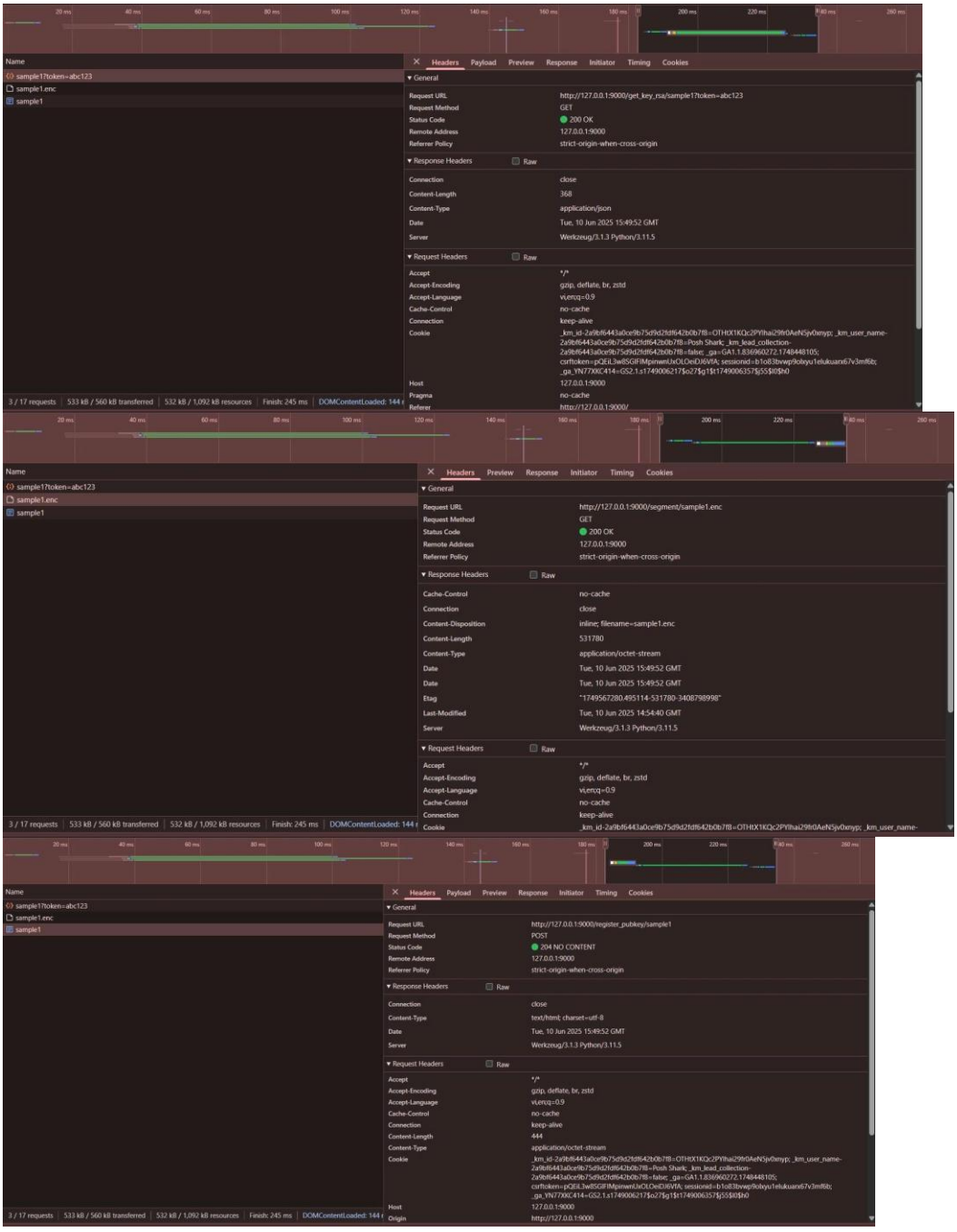
Khi truy cập trang web, trình duyệt gửi GET request đến server và server trả về index.html



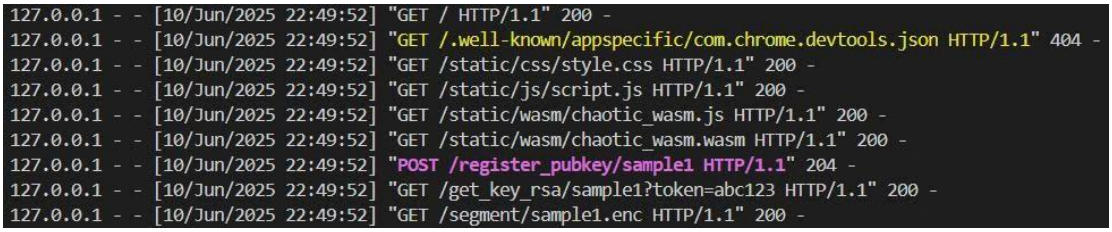
sau đó trình duyệt tiếp tục tiến hành tải các file js và css liên quan



trình duyệt tiếp tục gửi request để GET file web assembly để nhúng vào js



tiếp tục trình duyệt gửi một POST request để server nhận public key sau đó gửi một GET resquest với token là abc123 để nhận AES-key đã được mã hoá bằng public key client tiếp tục gửi một GET request để lấy file sample1.enc



Đây là console truyền và nhận video khi user thực hiện phát video.

Chi tiết các gói tin, lưu lượng mạng sẽ trình bày tại lớp.

So sánh với hệ thống truyền nhận của Youtube

- **Đồ án của nhóm:** chúng em đang tập trung vào việc mã hóa nội dung video bằng AES-256 và gửi khóa giải mã cho người dùng.

- **YouTube (và các nền tảng thương mại):** Họ cũng mã hóa nội dung bằng AES-128 (hoặc tương tự), nhưng sự khác biệt chính nằm ở **cơ chế quản lý và phân phối khóa (key management and distribution)**. Các hệ thống DRM rất phức tạp, liên quan đến các mô-đun phần cứng/phần mềm bảo mật trên thiết bị, các thỏa thuận cấp phép và quy trình xác thực chặt chẽ để đảm bảo chỉ những thiết bị được cấp phép và người dùng có quyền mới có thể lấy được khóa giải mã.

- **Báo cáo:** đồ án của nhóm tập trung vào việc hiểu và triển khai thuật toán mã hóa (AES-256, RSA,…), trong khi các hệ thống thương mại như YouTube không chỉ sử dụng các thuật toán mạnh mẽ mà còn có một hệ thống DRM phức tạp để quản lý quyền truy cập và phân phối khóa một cách an toàn, điều này vượt xa khả năng của nhóm chúng em để phân tích chi tiết.

Tài liệu tham khảo:

1) **Digital Rights Management (DRM) - Widevine (Google):**

https://developers.google.com/widevine

2) **Adaptive Bitrate Streaming (ABS) - HLS & MPEG-DASH:**

https://developer.apple.com/streaming/

3) **Cryptography and Networking Security_Principles and Practice:**

01_Cryptography and Network Security_ Principles and Practice.pdf

THE END