**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

--------o0o--------

A blue and white logo

Description automatically generated with medium confidence

**BÁO CÁO MÔN HỌC MẬT MÃ HỌC**

**-&-**

**ĐỀ TÀI: MÃ HÓA VÀ KIỂM SOÁT TRUY CẬP CƠ SỞ DỮ LIỆU**

**SỬ DỤNG AES VÀ CP-ABE**

**PROJECT: Encryption and Access Control Database**

**using AES and CP-ABE**

Giảng viên hướng dẫn : **TS. NGUYỄN NGỌC TỰ**

Sinh viên thực hiện:

1. **NGUYỄN ĐỨC TÀI** **MSSV: 21521395**
2. **TRẦN MINH DUY** **MSSV: 21522010**
3. **NGUYỄN HOÀI PHƯƠNG** **MSSV: 21520408**

Lớp : **NT219.N21.ANTT**

Khoá : **16**

***TP. Hồ Chí Minh, tháng 06 năm 2023***

**MỞ ĐẦU**

Trong thời đại số hóa hiện nay, bảo vệ thông tin cá nhân và dữ liệu trở nên ngày càng quan trọng hơn bao giờ hết. Việc lưu trữ và truy xuất các thông tin nhạy cảm trong cơ sở dữ liệu đòi hỏi một cơ chế bảo mật mạnh mẽ để đảm bảo tính bảo mật và sự toàn vẹn của dữ liệu.

Với các vụ việc lộ thông tin cá nhân và dữ liệu gần đây, việc bảo vệ thông tin cá nhân và dữ liệu càng trở nên cấp bách. Trong ngành bán lẻ, dữ liệu càng trở nên quan trọng hơn trong việc phân tích xu hướng và vận hành kinh doanh. Vì vậy, bảo vệ cơ sở dữ liệu trở thành một trong những ưu tiên hàng đầu của các doanh nghiệp bán lẻ.

Để đáp ứng nhu cầu bảo mật cơ sở dữ liệu, nhóm đồ án của chúng em đã chọn đề tài "Mã hóa và kiểm soát truy cập cơ sở dữ liệu sử dụng AES và CP-ABE". Mục tiêu của đề tài là nghiên cứu và xây dựng một hệ thống bảo mật mạnh mẽ cho cơ sở dữ liệu của chuỗi cửa hàng bán lẻ, sử dụng kỹ thuật mã hóa AES và CP-ABE.

Kết hợp AES và CP-ABE, hệ thống bảo mật cơ sở dữ liệu sẽ cung cấp một lớp bảo mật mạnh mẽ cho dữ liệu. AES được sử dụng để mã hóa dữ liệu trước khi được lưu trữ trong cơ sở dữ liệu, đảm bảo rằng chỉ những người được ủy quyền mới có thể giải mã dữ liệu. CP-ABE cho phép quản lý truy cập dựa trên các thuộc tính của người dùng, đảm bảo rằng chỉ những người có đủ quyền mới có thể giải mã và xem dữ liệu.

Với hệ thống này, chuỗi cửa hàng bán lẻ sẽ có thể đảm bảo tính bảo mật và sự toàn vẹn của dữ liệu trong cơ sở dữ liệu và quản lý truy cập của người dùng một cách chính xác và hiệu quả. Chúng em hy vọng rằng đề tài này sẽ đóng góp tích cực vào việc nâng cao tính bảo mật của các cơ sở dữ liệu và giúp các doanh nghiệp bán lẻ tăng cường bảo vệ thông tin cá nhân và dữ liệu của khách hàng.

**MỤC LỤC**

[Chương 1. TỔNG QUAN 1](#_Toc138615701)

[1.1. Tổng quan đề tài 1](#_Toc138615702)

[1.2. Nội dung báo cáo 1](#_Toc138615703)

[1.3. Phân công công việc 2](#_Toc138615704)

[Chương 2. PHÂN TÍCH HỆ THỐNG 3](#_Toc138615705)

[2.1. Ngữ cảnh 3](#_Toc138615706)

[2.2. Các bên liên quan 3](#_Toc138615707)

[2.2.1. TA (Trusted Authority): 4](#_Toc138615708)

[2.2.2. Data owners 4](#_Toc138615709)

[2.2.3. Data users 4](#_Toc138615710)

[2.2.4. Cloud 4](#_Toc138615711)

[2.2.5. Threats 4](#_Toc138615712)

[2.3. Mối liên hệ 4](#_Toc138615713)

[2.4. Vấn đề bảo mật và giải pháp cho hệ thống 5](#_Toc138615714)

[2.4.1. Đối với tấn công đường truyền 5](#_Toc138615715)

[2.4.2. Đối với tấn công giả mạo các biên liên quan: 5](#_Toc138615716)

[2.4.3. Đối với rủi ro lộ thông tin nội bộ: 6](#_Toc138615717)

[2.5. Tổng quan giải pháp 6](#_Toc138615718)

[2.5.1. Cipher text policy – attribute based encryption (BSW07) 8](#_Toc138615719)

[2.5.2. Về thuật toán CP-ABE(BSW07) 8](#_Toc138615720)

[Chương 3. GIẢI PHÁP ĐƯỢC TRIỂN KHAI THỰC HIỆN TRONG ĐỀ TÀI 12](#_Toc138615721)

[3.1. Sơ lược cách giải quyết các vấn đề bảo mật 12](#_Toc138615722)

[3.2. Vấn đề dữ liệu trên Cloud 12](#_Toc138615723)

[3.3. Vấn đề truyền dữ liệu trên các module TA, Users, Owners 13](#_Toc138615724)

[Chương 4. TRIỂN KHAI VÀ KẾT QUẢ THỰC NGHIỆM 14](#_Toc138615725)

[4.1. Kiến trúc hệ thống 14](#_Toc138615726)

[4.2. Phân tích hệ thống 14](#_Toc138615727)

[4.2.1. TA (Trusted Authority) 14](#_Toc138615728)

[4.2.2. Data owners 15](#_Toc138615729)

[4.2.3. Data users 15](#_Toc138615730)

[4.2.4. Cloud 15](#_Toc138615731)

[4.2.5. User 1 15](#_Toc138615732)

[4.2.6. User 2 15](#_Toc138615733)

[4.3. Mối liên hệ 16](#_Toc138615734)

[4.4. Vấn đề bảo mật và giải pháp cho hệ thống 16](#_Toc138615735)

[4.5. Kịch bản triển khai 17](#_Toc138615736)

[4.6. Tài nguyên 18](#_Toc138615737)

[4.7. Triển khai thử nghiệm, kết quả ban đầu: 19](#_Toc138615738)

[4.7.1. Node Trusted Authority 19](#_Toc138615739)

[4.7.2. Cấp public key và secret key 20](#_Toc138615740)

[4.7.3. Node Owner 21](#_Toc138615741)

[4.7.4. Node User 24](#_Toc138615742)

[4.8. Testing 26](#_Toc138615743)

[4.8.1. Owners 26](#_Toc138615744)

[4.8.2. Users 30](#_Toc138615745)

[4.8.3. Trusted Authority 33](#_Toc138615746)

[4.8.4. Cloud 33](#_Toc138615747)

[4.8.5. Thiết lập kết nối SSL/TLS 35](#_Toc138615748)

**DANH MỤC HÌNH**

[Hình 2.1 Các bên liên quan trong hệ thống 3](#_Toc138615790)

[Hình 2.2. Mô hình mã hóa và giải mã key AES 7](#_Toc138615791)

[Hình 2.3. Sơ đồ mã hóa và giải mã CP-ABE 7](#_Toc138615792)

[Hình 2.4. CHARM BENCHMARK ON A I7-4710HQ @ 3.40GHZ 8](#_Toc138615793)

[Hình 3.1. Mô hình cấp phát key dùng kết nối mạng an toàn SSL/TLS 13](#_Toc138615794)

[Hình 4.1. Mô hình Kiến trúc hệ thống 14](#_Toc138615795)

[Hình 4.2. Dữ liệu trước khi mã hóa 26](#_Toc138615796)

[Hình 4.3. Mã hóa bảng Products cột Model 27](#_Toc138615797)

[Hình 4.4. Dữ liệu sau khi mã hóa 27](#_Toc138615798)

[Hình 4.5. Mã hóa key AES bằng CP-ABE 29](#_Toc138615799)

[Hình 4.6. Owner thực hiện set policy cho từng Key 30](#_Toc138615800)

[Hình 4.7. User dùng attribute để nhận được key và giải mã được những Key AES tương ứng 31](#_Toc138615801)

[Hình 4.8. Dùng key vừa nhận được tiến hành giải mã 32](#_Toc138615802)

[Hình 4.9. Dữ liệu nhận được sau khi giải mã 33](#_Toc138615803)

[Hình 4.10. Trusted Authority ghi lại các phiên hoạt động 33](#_Toc138615804)

[Hình 4.11. Key AES trước khi mã 34](#_Toc138615805)

[Hình 4.12. Key AES sau khi mã được lưu trên Cloud 34](#_Toc138615806)

[Hình 4.13. Những trường dữ liệu quan trọng đều được mã và lưu trên Cloud 34](#_Toc138615807)

[Hình 4.14. Tạo certificate cho server 35](#_Toc138615808)

**NỘI DUNG BÁO CÁO**

# TỔNG QUAN

## Tổng quan đề tài

* Đề tài:  Encrypt and Control Access Database using AES and CP-ABE
* Thời gian thực hiện: từ tháng 03/2023 đến tháng 06/2023

## Nội dung báo cáo

* Chủ đề
* Giới thiệu
* Ngữ cảnh
* Các bên liên quan
* Các yêu cầu về bảo mật
* Tổng quan các giải pháp
* Đề xuất các hướng nghiên cứu cho đồ án và các kết quả dự kiến
* Lựa chọn tài liệu tham khảo chính: 2 bài báo
* Trình bày sơ lược cách giải quyết
* Triển khai và kết quả thực nghiệm:
  + Phân tích hệ thống
  + Kịch bản triển khai
  + Tài nguyên
  + Code và kết quả demo

## Phân công công việc

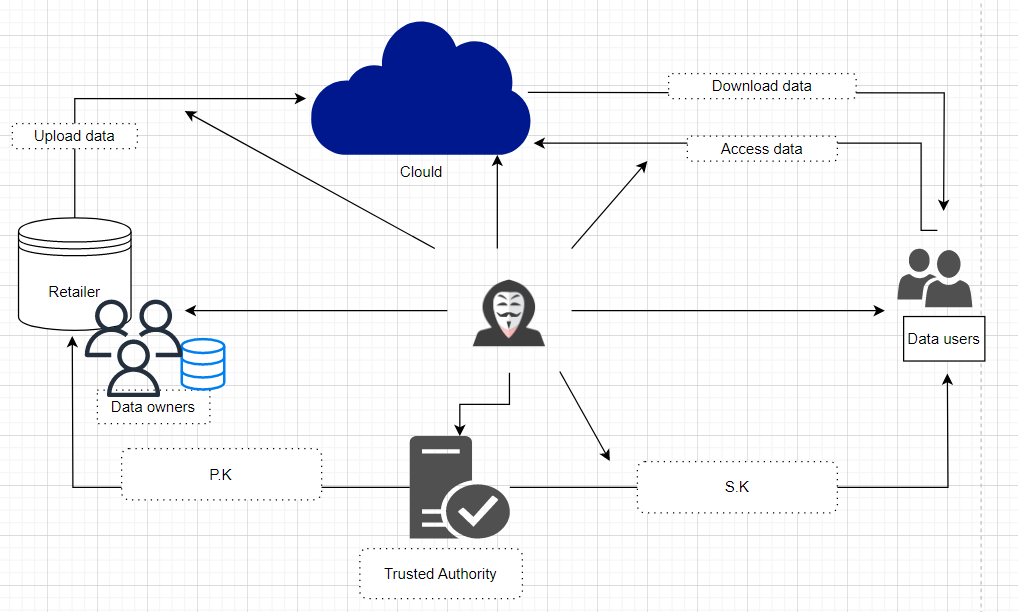
|  |  |
| --- | --- |
| **Công việc** | **Sinh viên thực hiện** |
| Xác định đề tài, lên ý tưởng, hướng nghiên cứu | Cả nhóm |
| Chọn bài báo học thuật phù hợp | Nguyễn Hoài Phương  Trần Minh Duy |
| Tìm ngữ cảnh, các bên liên quan,..  Nghiên cứu giải thuật và cách thức hoạt động của giao thức xác thực người dùng được đề xuất | Cả nhóm |
| Triển khai mã hóa và giải mã CP-ABE triển khai User, triển khai Owner trên Virtual Machine | Trần Minh Duy |
| Triển khai máy chủ cloud trên Microsoft Azure, thiết kế Cơ sở dữ liệu, Thực hiện mã hóa, giải mã CSDL bằng AES | Nguyễn Đức Tài |
| Triển khai Trusted Authority CP-ABE trên Virtual Machine, thiết lập kết nối giữa các node, thực hiện mã hóa key AES | Nguyễn Hoài Phương |
| Tìm kiếm thông tin, tài liệu, thư viện cần thiết để viết kịch bản triển khai demo | Cả nhóm |
| Viết báo cáo word, powerpoint | Cả nhóm |

**BÁO CÁO CHI TIẾT**

# PHÂN TÍCH HỆ THỐNG

## **Ngữ cảnh**

Chuỗi cửa hàng bán lẻ muốn lưu trữ dữ liệu trong database của mình lên cloud, để đảm bảo an toàn cho những dữ liệu này họ cần mã hóa nó và kiểm soát quyền truy cập, chỉ những ai có quyền mới được sử dụng dữ liệu trong CSDL



Hình 2.1 Các bên liên quan trong hệ thống

## Các bên liên quan

Có 5 bên liên quan trong mô hình ta đề cập ở trên: Data-users, Data-owners, Trusted Authority, Cloud và Threats**.**

### TA (Trusted Authority):

Đây là bên đáng tin cậy trong hệ thống, chịu trách nhiệm cung cấp các khóa và chứng chỉ cho các bên khác. TA cung cấp public key cho Data-owners để mã hóa dữ liệu và tạo secret key cho Data-users để giải mã dữ liệu trong hệ thống. TA cũng đồng thời chịu trách nhiệm xác thực cho các bên.

### **Data owners**

Chủ sở hữu dữ liệu, là người thực hiện mã hóa dữ liệu trong cơ sở dữ liệu bằng AES trước khi đưa lên cloud. Sau đó, Data-owners sử dụng public key của TA để mã hóa key AES, key sau khi mã hóa cũng được lưu trên cloud cùng với cơ sở dữ liệu đã mã hóa.

### Data users

Người dùng dữ liệu trong cơ sở dữ liệu. Vì dữ liệu được lưu trữ trên cloud đã được mã hóa, Data-users muốn truy cập và giải mã dữ liệu cần phải có secret key, key này được tạo bởi TA khi Data-users yêu cầu và cung cấp thuộc tính cho TA.

### Cloud

Đây là nơi lưu trữ cơ sở dữ liệu và khóa AES sau khi đã được mã hóa, đồng thời cung cấp khả năng truy xuất dữ liệu cho người dùng.

### Threats

Tập hợp các mối đe dọa cho hệ thống – là bên tìm cách gây hại, tấn công vào hệ thống nhằm mục đích đánh cắp, phá hoại dữ liệu.

## Mối liên hệ

Khi cần đưa dữ liệu lên cloud, Data-owners thực hiện mã hóa dữ liệu bằng AES, rồi yêu cầu TA (Trusted Authority) cung cấp public key, sau khi có public key Data owners dùng nó kết hợp chính sách truy cập (access policies) để mã hóa key AES. Dữ liệu đã mã hóa và bản mã của key sau đó được đưa lên cloud.

Khi một Data-user nào có nhu cầu truy cập vào database thì cần truy cập vào hệ thống bằng email và password, nếu chưa có secret key để giải mã dữ liệu, Data-user cần gửi yêu cầu đến TA đồng thời cung cấp Attributes để tạo secret key. Lúc này Data user lên cloud để tải dữ liệu cần giải mã và bản mã của key tương ứng, lần lượt giải mã key AES và dữ liệu bằng secret key và key AES vừa giải mã được.

Threat có thể là hacker, các công ty đối thủ,... sẽ tấn công vào đường truyền để đánh cấp secret key hoặc cũng có thể đóng giả làm các thực thể Data Owner, Users hoặc Trusted Authority.

## Vấn đề bảo mật và giải pháp cho hệ thống

Threats đối với hệ thống có thể chia thành: bên ngoài gồm có tấn công đường truyền, giả mạo các biên liên quan, bên trong có rủi ro lộ thông tin nội bộ

### Đối với tấn công đường truyền

Giải pháp là thiết lập kết nối bảo mật bằng TLS

### Đối với tấn công giả mạo các biên liên quan:

#### Giả mạo Trusted Authority

Để cấp public key cho Data-owners, nếu Data-owners sử dụng key đó để mã hóa dữ liệu thì kẻ giả mạo có thể giải mã những dữ liệu đó bằng cách tự tạo secret key cho bản thân: giải pháp là tạo Certificate để xác thực TA, Certificate của TA được ký bởi chính nó, mỗi khi TA cung cấp key nó sẽ gửi kèm Certificate, bên xác thực sẽ lấy public key của TA đã được lưu ở nơi an toàn để xác thực.

#### Giả mạo Data-owners

Để đưa dữ liệu rác, không chính xác, làm tốn bộ nhớ,… : TA xác thực Data-owners trước khi cho phép vào hệ thống, tạo cơ chế để chỉ những Data-owners mới được đưa dữ liệu lên cloud.

#### Giả mạo Data-users

Trong hệ thống để lấy secret key, từ đó lấy được dữ liệu… : TA xác thực Data-users trước khi tạo secret key.

### Đối với rủi ro lộ thông tin nội bộ:

Cloud được coi là một đối tượng semi-trusted (tức là có sự tin tưởng đến một mức độ nhất định), mặc dù có thể coi là một đối tượng đáng tin cậy nhưng vẫn có khả năng thực hiện các hành động không mong muốn đến dữ liệu như làm rò rỉ thông tin hay sử dụng dữ liệu cho mục đích khác: giải pháp là mã hóa những dữ liệu quan trọng không muốn bên cloud biết, khi đó dù dữ liệu bị rò rỉ thì thiệt hại ở mức nhỏ vì những dữ liệu đó đã mã hóa không thể đọc được.

## Tổng quan giải pháp

* Mã hóa bản rõ database bằng thuật toán AES-CBC-256 (mỗi key sẽ mã hóa 1 cột trong database)
* Khóa AES sẽ được mã hóa tiếp bằng CP-ABE (scheme BSW07)

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, hàng

Mô tả được tạo tự động

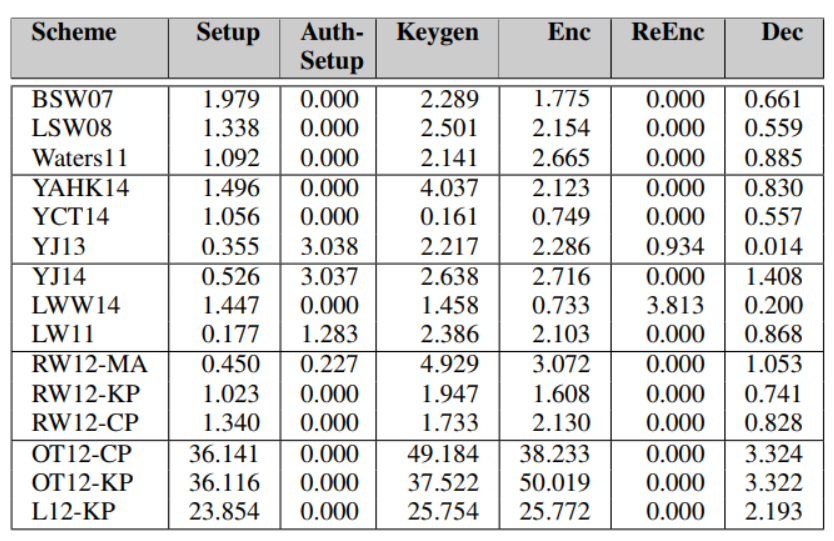
Hình 2.2. Mô hình mã hóa và giải mã key AES

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, Phông chữ

Mô tả được tạo tự động

Hình 2.3. Sơ đồ mã hóa và giải mã CP-ABE

### Cipher text policy – attribute based encryption (BSW07)



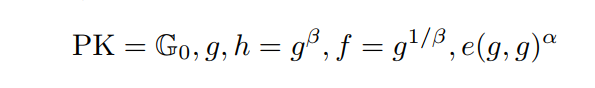
Hình 2.4. CHARM BENCHMARK ON A I7-4710HQ @ 3.40GHZ

Dựa vào hình trên, đối với những nhà phát triển làm việc với CP-ABE (Attribute-Based Encryption), thì hệ thống cần có sự cân đối tốt giữa ba hoạt động tạo setup, mã hóa và giải mã. Với việc sử dụng BSW07, có thể tận dụng được hiệu suất mã hóa và giải mã tốt hơn trong hệ thống CP-ABE. Điều này đảm bảo rằng quá trình mã hóa và giải mã AES sẽ được thực hiện một cách hiệu quả và nhanh chóng.

### Về thuật toán CP-ABE(BSW07)

#### Setup

Trong thuật toán Setup, ta sẽ chọn một nhóm bilinear G0 có thứ tự nguyên tố p với phần tử sinh g. Tiếp theo, ta sẽ chọn hai số mũ ngẫu nhiên α và β thuộc Zp. Public key và master key sẽ tính bằng công thức:



Trong đó:

G0 là nhóm bilinear có thứ tự nguyên tố p.

g là phần tử sinh của nhóm G0.

h là gβ, nơi β là một số mũ ngẫu nhiên.

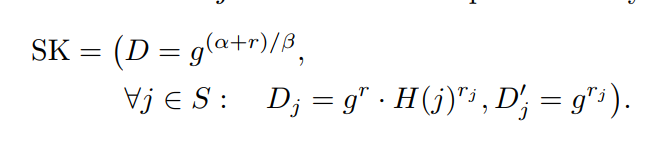
f là g1/β, 1/β là nghịch đảo của β trong trường số nguyên modulo p.

e(g, g) là một hàm kết hợp hai phần tử của nhóm G0 và trả về một phần tử của nhóm đa thức.

PK = G0, g, h = gβ, f = g1/β, e(g, g)α và khóa chủ (master key) MK là (β, gα).

#### Key Gen(MS,S):

Thuật toán đầu tiên chọn một số ngẫu nhiên r thuộc Zp và sau đó chọn các số ngẫu nhiên rj thuộc Zp cho mỗi thuộc tính j thuộc S. Sau đó, thuật toán tính toán khóa như sau:



Ví dụ với mỗi thuộc tính j thuộc S:

Dj = gr . H(j) rj,

D’j = grj,

Trong đó:

SK là khóa kết quả được tạo ra.

D là thành phần khóa chứa các giá trị để truy cập tới dữ liệu.

g, α và β là các thành phần của khóa công khai PK.

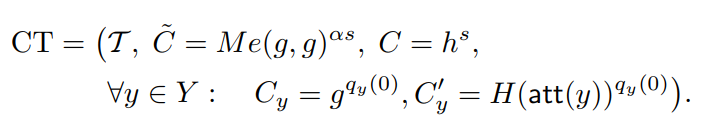
H(j) là hàm băm được áp dụng cho thuộc tính j.

#### Encrypt(PK,M.T):

Thuật toán đầu tiên chọn một đa thức qx cho mỗi nút x (bao gồm cả các lá) trong cây T. Những đa thức này được chọn theo cách sau theo hướng từ trên xuống, bắt đầu từ nút gốc R. Đối với mỗi nút x trong cây, đặt bậc dx của đa thức qx là một nhỏ hơn một đơn vị so với giá trị ngưỡng kx của nút đó, tức là dx = kx - 1.

Bắt đầu với nút gốc R, thuật toán chọn một số ngẫu nhiên s thuộc Zp và đặt qR(0) = s. Sau đó, nó chọn dR điểm khác của đa thức qR một cách ngẫu nhiên để định nghĩa nó hoàn toàn. Đối với bất kỳ nút x nào khác, nó đặt qx(0) = qparent(x)(index(x)) và chọn dxđiểm khác ngẫu nhiên để định nghĩa qx

Cho Y là tập các nút lá trong T. Bản mã được tạo bằng cách cung cấp cấu trúc truy cập cây T và tính toán như sau:



Trong đó:

CT là cấu trúc truy cập cây.

C~ là phần tử mã hóa của thông điệp M.

C là phần tử mã hóa cho cây T.

Cy và Cy0 là các phần tử liên quan đến nút y trong cây T.

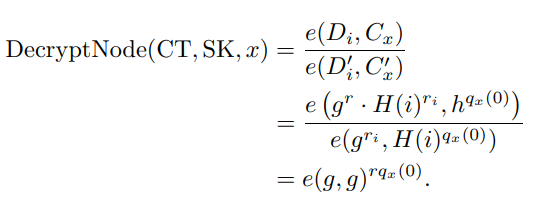
qy  là đa thức tương ứng với nút y, và qy(0) là giá trị của đa thức qy tại điểm 0.

H(att(y)) là hàm băm được áp dụng cho thuộc tính att(y) của nút y.

#### Decrypt(CT,SK,x)

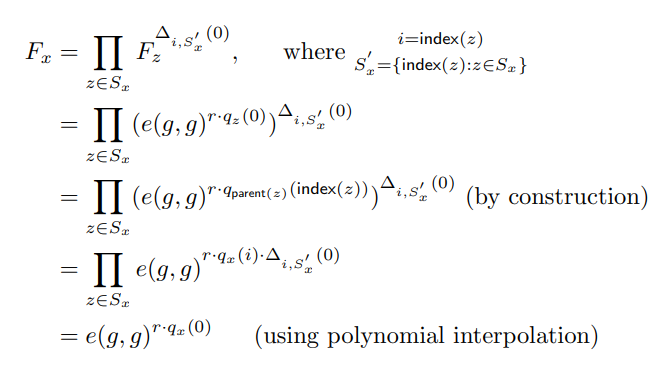
Lấy i = att(x) là chỉ số của nút lá.

Nếu i thuộc S, ta định nghĩa DecryptNode(CT, SK, x) như sau:



Nếu i không thuộc S, ta định nghĩa DecryptNode(CT, SK, x) = ⊥ (ký hiệu cho việc không giải mã được).

Tiếp theo, ta xem xét trường hợp đệ quy khi x là một nút không phải lá. Thuật toán DecryptNode(CT, SK, x) tiếp tục như sau:



Trả về kết quả Fx

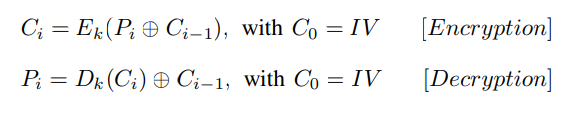
# **GIẢI PHÁP ĐƯỢC TRIỂN KHAI THỰC HIỆN TRONG ĐỀ TÀI**

## Sơ lược cách giải quyết các vấn đề bảo mật

## **Vấn đề dữ liệu trên Cloud**

Đối với Cloud, mã hóa dữ liệu bằng thuật toán AES - 256 (mode CBC) trước khi lưu lên database:

* Dữ liệu sẽ được mã hóa theo cột để đảm bảo số lượng key tạo ra không quá nhiều và đảm bảo rằng chỉ những cột dữ liệu nhạy cảm sẽ được mã hóa, đồng thời hỗ trợ truy vấn dữ liệu theo cột.
* Chế độ CBC làm giảm khả năng xuất hiện các mẫu lặp lại trong bản mã. Trong chế độ CBC, trước khi mã hóa một khối bản rõ, nó được XOR với khối bản mã trước đó. Bản rõ đầu tiên khối được XOR với một vectơ khởi tạo (IV). Trong đồ án này, nhóm chúng em đề xuất cách tạo IV dựa trên key đã tạo điều này nhằm dễ dàng lưu trữ khóa, ta chỉ cần lưu trữ Key và IV sẽ được sinh tương ứng với Key đó. Chúng em tiến hành băm Key đã tạo qua SHA-256 và lấy 16 Byte đầu tiên tương ứng với 128 bits sẽ là IV.



## Vấn đề truyền dữ liệu trên các module TA, Users, Owners

Thiết lập sử dụng kết nối mạng an toàn (SSL/TLS) giữa người dùng và TA để ngăn chặn các cuộc tấn công giả mạo hoặc đánh cắp thông tin trên đường truyền. Chứng chỉ SSL/TLS được tạo bằng thuật toán mã hóa ECC (Elliptic Curve Cryptography) và sử dụng thuật toán băm SHA256 trong quá trình ký và xác thực chứng chỉ

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, Sơ đồ

Mô tả được tạo tự động

Hình 3.1. Mô hình cấp phát key dùng kết nối mạng an toàn SSL/TLS

# TRIỂN KHAI VÀ KẾT QUẢ THỰC NGHIỆM

## Kiến trúc hệ thống

Ảnh có chứa văn bản, biểu đồ, bản đồ, Kế hoạch

Mô tả được tạo tự động

Hình 4.1. Mô hình Kiến trúc hệ thống

## Phân tích hệ thống

### **TA (Trusted Authority)**

Đây là bên đáng tin cậy trong hệ thống, chịu trách nhiệm cung cấp các khóa và chứng chỉ cho các bên khác. TA cung cấp public key cho Data-owners để mã hóa dữ liệu và tạo secret key cho Data-users để giải mã dữ liệu trong hệ thống. TA cũng đồng thời chịu trách nhiệm xác thực cho các bên.

### **Data owners**

Chủ sở hữu dữ liệu, là người thực hiện mã hóa dữ liệu trong cơ sở dữ liệu bằng AES trước khi đưa lên cloud. Sau đó, Data-owners sử dụng public key của TA để mã hóa key AES, key sau khi mã hóa cũng được lưu trên cloud cùng với cơ sở dữ liệu đã mã hóa.

### Data users

Người dùng dữ liệu trong cơ sở dữ liệu. Vì dữ liệu được lưu trữ trên cloud đã được mã hóa, Data-users muốn truy cập và giải mã dữ liệu cần phải có secret key, key này được tạo bởi TA khi Data-users yêu cầu và cung cấp thuộc tính cho TA.

### **Cloud**

Đây là nơi lưu trữ cơ sở dữ liệu và khóa AES sau khi đã được mã hóa, đồng thời cung cấp khả năng truy xuất dữ liệu cho người dùng. Với từng loại dữ liệu thì cloud sẽ có cách lưu trữ và xử lý riêng:

* + Dữ liệu dạng text: lưu trực tiếp bản mã trong database, khi truy vấn trả về đoạn text đã mã hóa.
  + Dữ liệu dạng hình ảnh: lưu bản mã trong một thư mục, địa chỉ nào đó, khi truy vấn chỉ trả về đường link để tải hình ảnh đã mã hóa xuống.
  + Dữ liệu dạng file: xử lý tương tự như dữ liệu dạng ảnh.

### User 1

Gửi yêu cầu đến Trusted Authority (TA) và gửi các thuộc tính của mình để TA tạo secret key. User1 trong hệ thống trên có các thuộc tính thỏa với chính sách truy cập của Data Owners nên User 1 có thể dùng secert key của mình để giải mã key AES và dùng key AES để giải mã những cột dữ liệu database mà mình có thể truy cập được.

### User 2

Ngược lại với User 1, User 2 cũng yêu cầu TA cấp secret key có mình. Tuy nhiên User 2 có những thuộc tính không thỏa mãn với chính sách của Data Owners nên User 2 không thể giải mã để lấy bản rõ của AES key.

## Mối liên hệ

Khi cần đưa dữ liệu lên cloud, Data-owners thực hiện mã hóa dữ liệu bằng AES, rồi yêu cầu TA (Trusted Authority) cung cấp public key, sau khi có public key Data owners dùng nó kết hợp chính sách truy cập (access policies) để mã hóa key AES. Dữ liệu đã mã hóa và bản mã của key sau đó được đưa lên cloud.

Khi một Data-user nào có nhu cầu truy cập vào database thì cần truy cập vào hệ thống bằng email và password, nếu chưa có secret key để giải mã dữ liệu, Data-user cần gửi yêu cầu đến TA đồng thời cung cấp Attributes để tạo secret key. Lúc này Data user lên cloud để tải dữ liệu cần giải mã và bản mã của key tương ứng, lần lượt giải mã key AES và dữ liệu bằng secret key và key AES vừa giải mã được.

Threat có thể là hacker, các công ty đối thủ,... sẽ tấn công vào đường truyền để đánh cấp secret key hoặc cũng có thể đóng giả làm các thực thể Data Owner, Users hoặc Trusted Authority.

## Vấn đề bảo mật và giải pháp cho hệ thống

Threats đối với hệ thống có thể chia thành: bên ngoài gồm có tấn công đường truyền, giả mạo các biên liên quan, bên trong có rủi ro lộ thông tin nội bộ

* Đối với tấn công đường truyền: giải pháp là thiết lập kết nối bảo mật bằng TLS
* Đối với tấn công giả mạo các biên liên quan:
  + Giả mạo Trusted Authority để cấp public key cho Data-owners, nếu Data-owners sử dụng key đó để mã hóa dữ liệu thì kẻ giả mạo có thể giải mã những dữ liệu đó bằng cách tự tạo secret key cho bản thân: giải pháp là tạo Certificate để xác thực TA, Certificate của TA được ký bởi chính nó, mỗi khi TA cung cấp key nó sẽ gửi kèm Certificate, bên xác thực sẽ lấy public key của TA đã được lưu ở nơi an toàn để xác thực.
  + Giả mạo Data-owners để đưa dữ liệu rác, không chính xác, làm tốn bộ nhớ,… : TA xác thực Data-owners trước khi cho phép vào hệ thống, tạo cơ chế để chỉ những Data-owners mới được đưa dữ liệu lên cloud.
  + Giả mạo Data-users trong hệ thống để lấy secret key, từ đó lấy được dữ liệu… : TA xác thực Data-users trước khi tạo secret key.
* Đối với rủi ro lộ thông tin nội bộ: Cloud được coi là một đối tượng semi-trusted (tức là có sự tin tưởng đến một mức độ nhất định), mặc dù có thể coi là một đối tượng đáng tin cậy nhưng vẫn có khả năng thực hiện các hành động không mong muốn đến dữ liệu như làm rò rỉ thông tin hay sử dụng dữ liệu cho mục đích khác: giải pháp là mã hóa những dữ liệu quan trọng không muốn bên cloud biết, khi đó dù dữ liệu bị rò rỉ thì thiệt hại ở mức nhỏ vì những dữ liệu đó đã mã hóa không thể đọc được.

## Kịch bản triển khai

* Trusted Authority, Data owners và Data users được triển khai trên các laptop khác nhau với môi trường Linux
* Xây dựng 1 cloud server database trên Microsoft Azure là nơi lưu trữ bản mã của key AES và dữ liệu đã mã hóa.
* TA sẽ là bên xác thực cho Data owners và Data users trước khi cho phép vào hệ thống nhưng hiện nhóm chúng em chưa làm được nên bọn em sử dụng dịch vụ firebase của Google để thay thế.
* Sau khi cài đặt xong các thiết bị, Data owner sẽ tiến hành đăng nhập vào hệ thống bằng cách cung cấp email và password để xác thực, sau khi xác thực thành công Data owner sẽ vào giao diện chính của chương trình; lúc này Data owner thao tác để chọn dữ liệu cần mã hóa và chính sách truy cập, chính sách truy cập được yêu cầu để Data user giải mã được một số cột trong database còn một số cột khác thì không nhằm mục đích mô phỏng, các công việc như tạo key AES, mã hóa dữ liệu, mã hóa key AES rồi đưa tất cả lên Cloud sẽ được thực hiện tự động.
* Data user ở một máy khác sẽ tiến hành đăng nhập hệ thống bằng cách cung cấp email và password, sau khi xác thực thành công Data user tiến hành tải dữ liệu trên database về và tiến hành giải mã để lấy key AES, các thao tác như kiểm tra đã có secret key hay chưa, gửi yêu cầu đến TA, giải mã lấy key sẽ được thực hiện, đối với nhưng key mà user không có quyền truy cập sẽ hiện kết quả là “Decryption failed” trong file key đã được giải mã; sau khi có key AES, user sử dụng key này để giải mã dữ liệu trong cột tương ứng.

## Tài nguyên

Sử dụng 3 thiết bị làm 3 thành phần chính:

* TA (Trusted Authority): dùng 1 laptop để triển khai. Viết chương trình triển khai:
  + - IDE: Visual Studio
    - Program Language: python
    - Library: Charm
  + User: dùng 1 laptop để triển khai. Viết chương trình triển khai:
    - IDE: Visual Studio
    - Program Language: python
    - Library: Charm
  + Owner: dùng 1 laptop:
    - IDE: Visual Studio
    - Program Language: python
    - Library: Charm
  + Cloud: tạo một SQL Server trên Microsoft Azure, triển khai SQL database trên đó.

## Triển khai thử nghiệm, kết quả ban đầu:

### Node Trusted Authority

|  |
| --- |
| **class** **TA**:  **def** **\_\_init\_\_**(self, groupObj):   self.cpabe = CPabe\_BSW07(groupObj)  self.hyb\_abe = HybridABEnc(self.cpabe, groupObj)  self.pk = None  self.mk = None   **def** **setup**(self):  (self.pk, self.mk) = self.hyb\_abe.setup()  # return (self.pk, self.mk)   **def** **keygen**(self, attributes):  **return** self.hyb\_abe.keygen(self.pk, self.mk, attributes)   **def** **get\_pk**(self):  """  Get the public key.  """  **return** self.pk   **def** **get\_mk**(self):  """  Get the master key.  """  **return** self.mk |

### Cấp public key và secret key

|  |
| --- |
| **def** **handle\_client**(conn, addr):  **print**('client connected:', addr)  **while** True:  msg = conn.recv(**1024**).decode(FORMAT)  **if** msg == 'get\_pubKey':  pk = ta.get\_pk()  group = PairingGroup('SS512')  data = objectToBytes(pk, group)  conn.sendall(data)  conn.sendall('\_\_end\_\_'.encode(FORMAT))  **print**('Sent public key to client', addr)   **elif** msg == 'gen\_secretKey':  att\_list = receive\_att(conn)  **print**('Received attributes list', att\_list, 'from', addr)  sk = ta.keygen(att\_list)  group = PairingGroup('SS512')  data = objectToBytes(sk, group)  conn.sendall(data)  conn.sendall('\_\_end\_\_'.encode(FORMAT))  **print**('Generate secret key for client', addr)   **else**:  **if** msg == 'x':  **break**  **print**('client', str(addr)+":", msg)    **print**("client", addr, "disconnected")  conn.close() |

### Node Owner

* Coding phần mã hóa

|  |
| --- |
| **class** **Owner**:  **def** **\_\_init\_\_**(self, groupObj):  self.groupObj = groupObj  self.cpabe = CPabe\_BSW07(groupObj)  self.hyb\_abe = HybridABEnc(self.cpabe, groupObj)  self.policies = None   **def** **set\_access\_policy**(self, policy):  self.access\_policy = policy    **def** **encrypt\_file**(self, plaintext\_path, ciphertext\_path, pk):  **try**:  **with** open(plaintext\_path, 'rb') **as** plaintext\_file:  plaintext = plaintext\_file.read()    ciphertext = self.hyb\_abe.encrypt(pk, plaintext, self.access\_policy)  cipher\_bytes = objectToBytes(ciphertext, self.groupObj)   **with** open(ciphertext\_path, 'wb') **as** ciphertext\_file:  pickle.dump(cipher\_bytes, ciphertext\_file)   **print**('...')  **print**("The file '" + ciphertext\_path + "' created with policies: ", self.access\_policy)     **except** **Exception** **as** e:  **print**("Failed to ecrypt file: ", str(e))    **def** **encrypt\_message**(self, message, pk, access\_policy, ciphertext\_path):  **try**:  ciphertext = self.hyb\_abe.encrypt(pk, message, access\_policy)  cipher\_bytes = objectToBytes(ciphertext, self.groupObj)      **with** open(ciphertext\_path, 'ab') **as** ciphertext\_file:  ciphertext\_file.write(cipher\_bytes)  ciphertext\_file.write(b'**\n**')   **return** cipher\_bytes   **except** **Exception** **as** e:  **print**("Failed to encrypt message:", str(e))  **return** None    **def** **set\_attributes**(self, attributes):  self.att = attributes   **def** **set\_secretKey**(self, secretKey):  self.sk = secretKey   **def** **decrypt\_file**(self, ciphertext\_path, sk, pk, plaintext\_path):    **try**:  **with** open(ciphertext\_path, 'rb') **as** ciphertext\_file:  cipher\_bytes = pickle.load(ciphertext\_file)    ciphertext = bytesToObject(cipher\_bytes, self.groupObj)  plaintext = self.hyb\_abe.decrypt(pk, sk, ciphertext)   **with** open(plaintext\_path, 'wb') **as** plaintext\_file:  plaintext\_file.write(plaintext)   **print**("...")  **print**("File decrypted successfully !!")    **except** **Exception** **as** e:  **print**("Failed to decrypt file:", str(e))    **def** **decrypt\_message**(self, ciphertext\_path, sk, pk, plaintext\_path):  **try**:  **with** open(ciphertext\_path, 'rb') **as** ciphertext\_file:  **with** open(plaintext\_path, 'a') **as** plaintext\_file:  **while** True:  **try**:  cipher\_bytes = ciphertext\_file.readline()  **if** **not** cipher\_bytes:  **break**    # Loại bỏ ký tự xuống dòng ở cuối mỗi dòng  # print(cipher\_bytes)  cipher\_bytes = cipher\_bytes.rstrip(b'**\n**')   ciphertext = bytesToObject(cipher\_bytes, self.groupObj)   plaintext = self.hyb\_abe.decrypt(pk, sk, ciphertext).decode('utf-8')   **print**(plaintext, '...success')  plaintext\_file.write(plaintext)  plaintext\_file.write('**\n**')   **except** **Exception** **as** e:  **print**("Failed to decrypt key:", str(e))  plaintext\_file.write('Decryption failed**\n**')  **continue**     **except** **Exception** **as** e:  **print**("Failed to open ciphertext file:", str(e)) |

### Node User

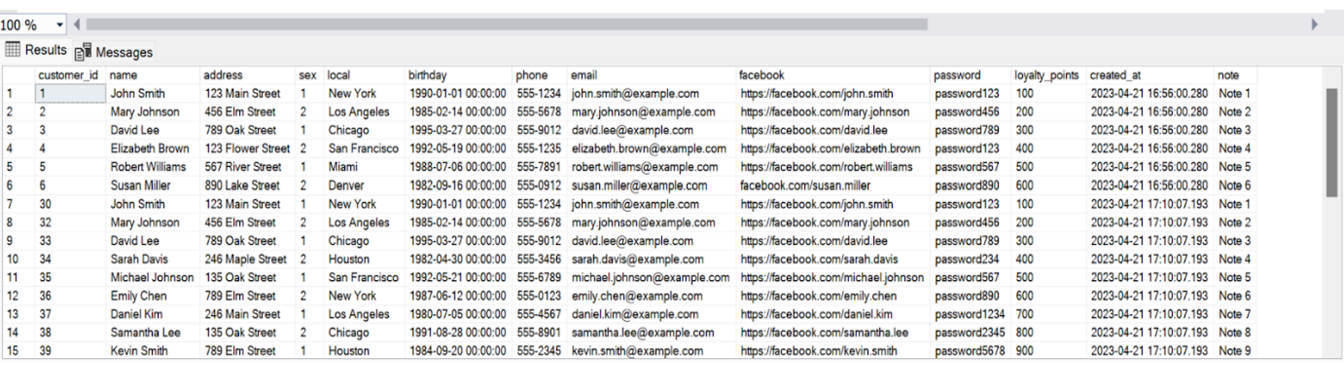
* Coding phần giải mã

|  |
| --- |
| **class** **Client**:  **def** **\_\_init\_\_**(self, groupObj):  self.groupObj = groupObj  self.cpabe = CPabe\_BSW07(groupObj)  self.hyb\_abe = HybridABEnc(self.cpabe, groupObj)  self.att = None  self.sk = None  **def** **set\_attributes**(self, attributes):  self.att = attributes  **def** **set\_secretKey**(self, secretKey):  self.sk = secretKey  **def** **decrypt\_file**(self, ciphertext\_path, sk, pk, plaintext\_path):    **try**:  **with** open(ciphertext\_path, 'rb') **as** ciphertext\_file:  cipher\_bytes = pickle.load(ciphertext\_file)    ciphertext = bytesToObject(cipher\_bytes, self.groupObj)  plaintext = self.hyb\_abe.decrypt(pk, sk, ciphertext)  **with** open(plaintext\_path, 'wb') **as** plaintext\_file:  plaintext\_file.write(plaintext)  **print**("...")  **print**("File decrypted successfully !!")    **except** **Exception** **as** e:  **print**("Failed to decrypt file:", str(e))  **def** **decrypt\_message**(self, ciphertext\_path, sk, pk, plaintext\_path):  **try**:  **with** open(ciphertext\_path, 'rb') **as** ciphertext\_file:  **with** open(plaintext\_path, 'a') **as** plaintext\_file:  **while** True:  **try**:  cipher\_bytes = ciphertext\_file.readline()  **if** **not** cipher\_bytes:  **break**    # Loại bỏ ký tự xuống dòng ở cuối mỗi dòng  # print(cipher\_bytes)  cipher\_bytes = cipher\_bytes.rstrip(b'**\n**')  ciphertext = bytesToObject(cipher\_bytes, self.groupObj)  plaintext = self.hyb\_abe.decrypt(pk, sk, ciphertext).decode('utf-8')  **print**(plaintext, '...success')  plaintext\_file.write(plaintext)  plaintext\_file.write('**\n**')  **except** **Exception** **as** e:  **print**("Failed to decrypt key:", str(e))  plaintext\_file.write('Decryption failed**\n**')  **continue**    **except** **Exception** **as** e:  **print**("Failed to open ciphertext file:", str(e)) |

## Testing

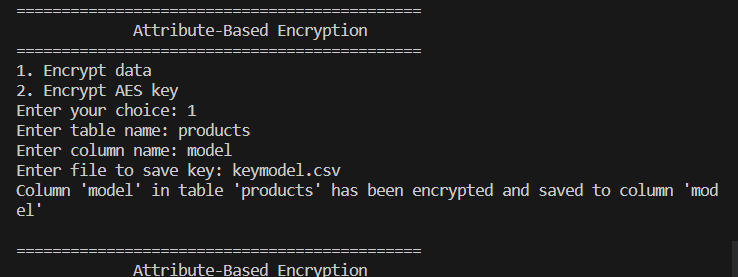
### Owners

* Mã hóa Database bằng AES-CBC-256.
* Dữ liệu trước khi mã hóa:



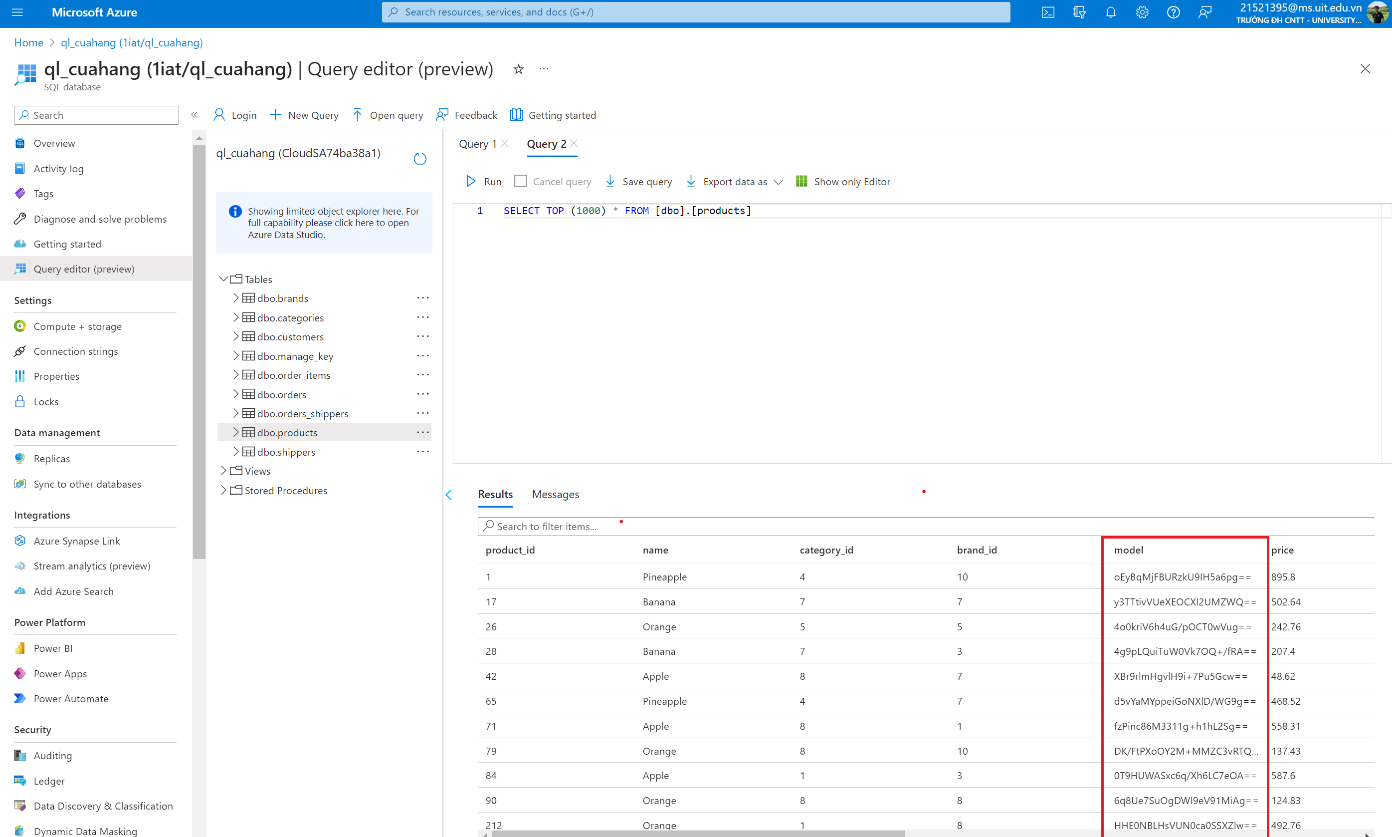
Hình 4.2. Dữ liệu trước khi mã hóa

* Demo mã hóa bảng products cột model:



Hình 4.3. Mã hóa bảng Products cột Model

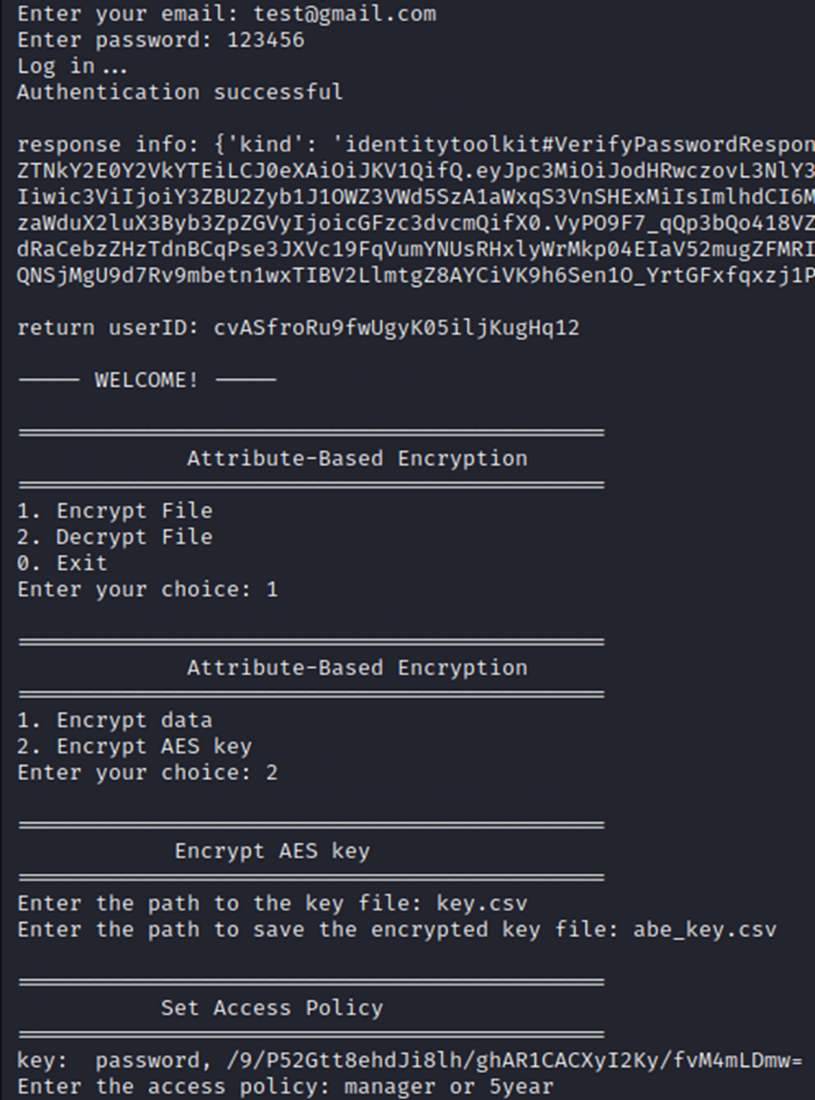
* Dữ liệu sau khi mã hóa, được đưa lên Cloud:



Hình 4.4. Dữ liệu sau khi mã hóa



* Mã hóa Key AES bằng CP-ABE



Hình 4.5. Mã hóa key AES bằng CP-ABE

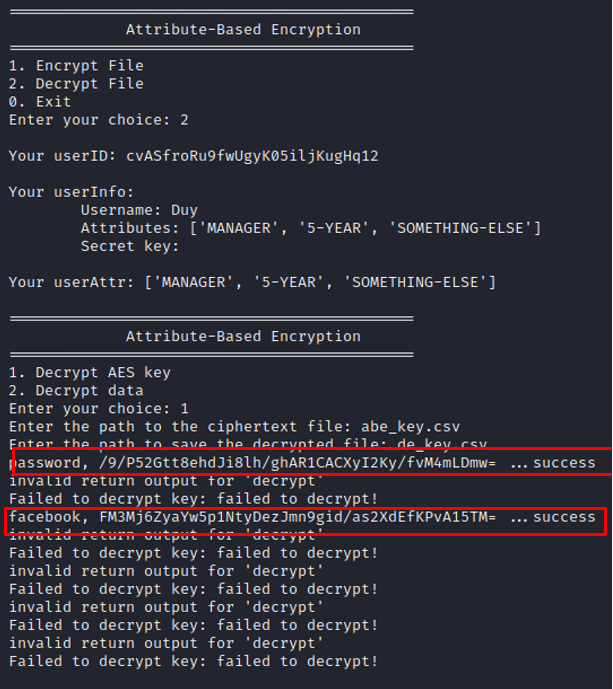
* Owner thực hiện set policies cho từng key



Hình 4.6. Owner thực hiện set policy cho từng Key

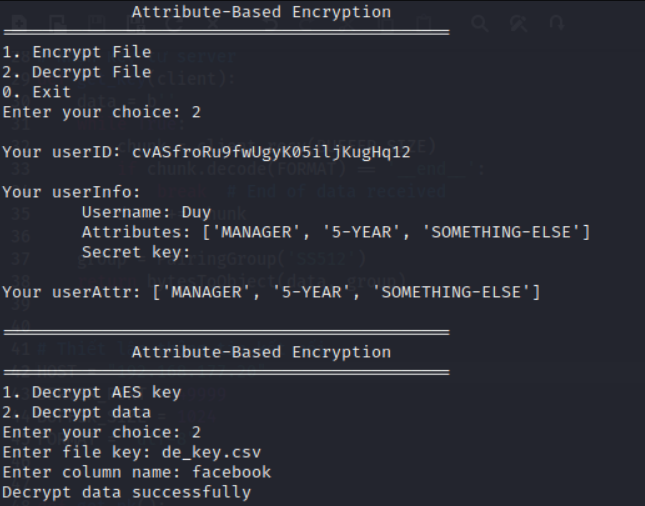
### Users

* User thực hiện giải mã key AES và nhận kết quả



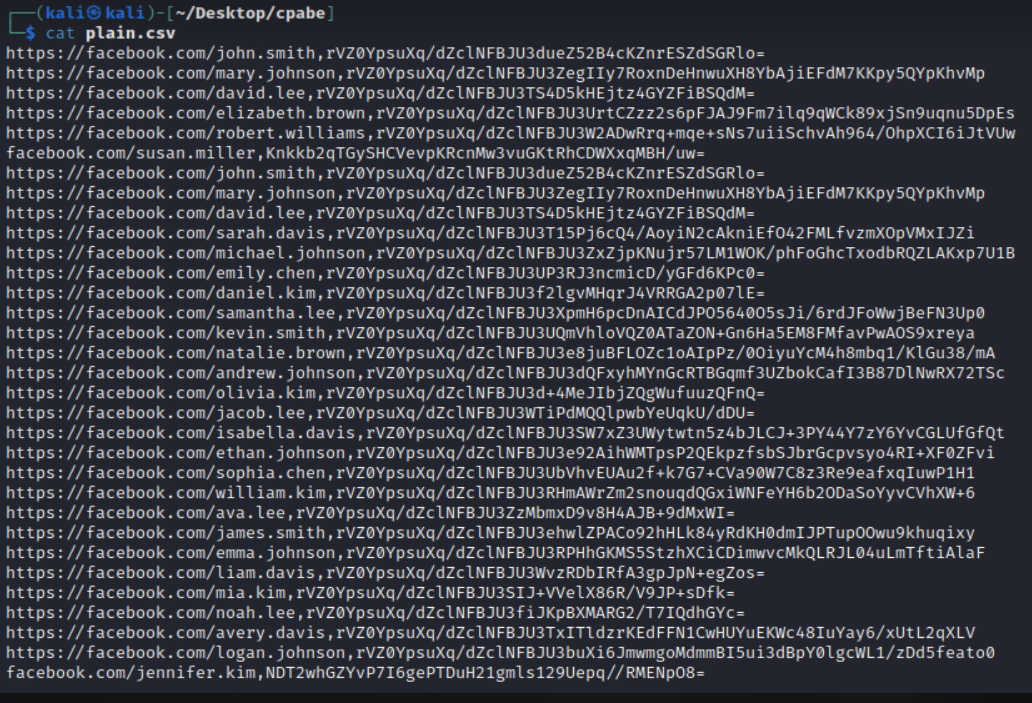
Hình 4.7. User dùng attribute để nhận được key và giải mã được những Key AES tương ứng

* Sau khi nhận được key user sẽ dùng để giải mã các cột dữ liệu tương ứng



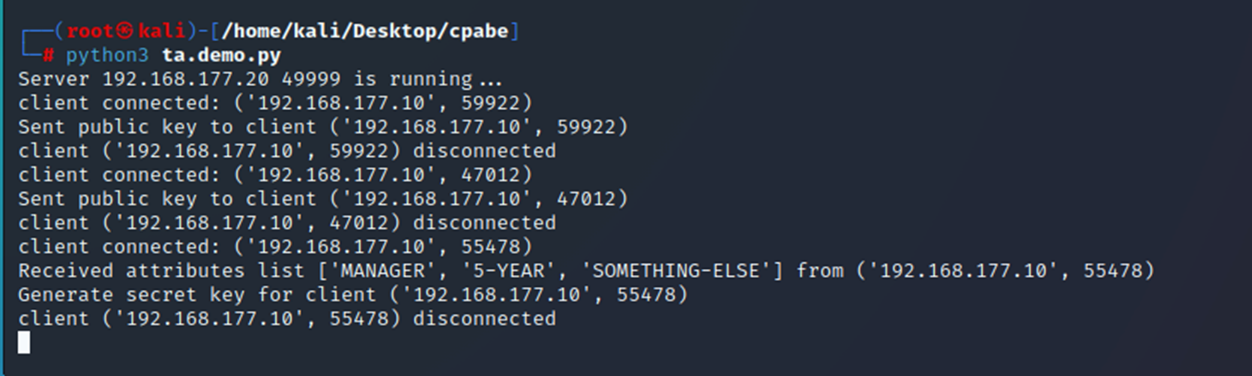
Hình 4.8. Dùng key vừa nhận được tiến hành giải mã

* Dữ liệu nhận được:



Hình 4.9. Dữ liệu nhận được sau khi giải mã

### Trusted Authority

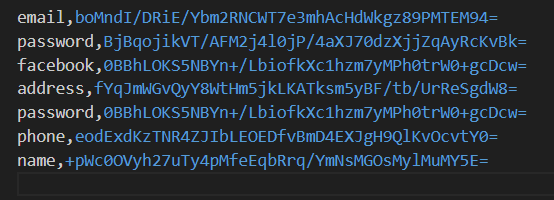


Hình 4.10. Trusted Authority ghi lại các phiên hoạt động

* Dòng lệnh Sent public key: Gửi public key cho Owners khi thực hiện mã hóa
* Dòng lệnh Generate secret key: Gửi secret key cho Users khi thực hiện giải hóa

### Cloud

* Key AES trước khi mã:



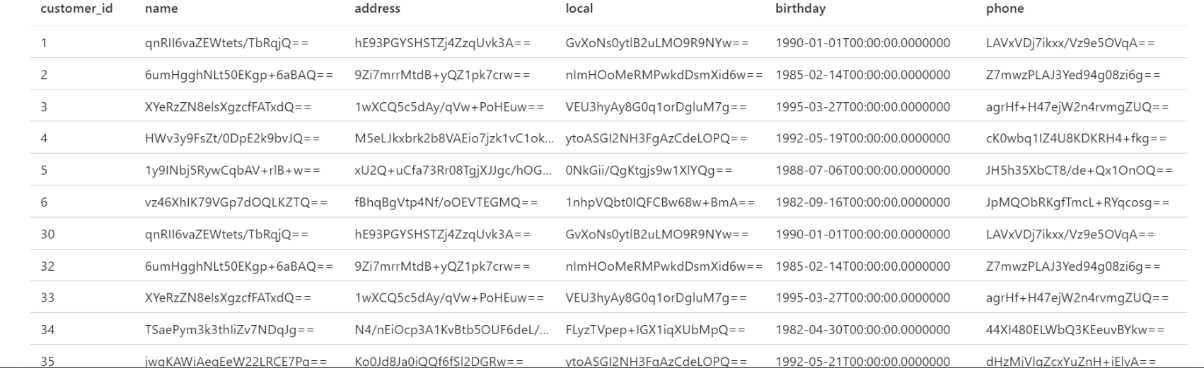
Hình 4.11. Key AES trước khi mã

* Key AES sau khi mã hóa:



Hình 4.12. Key AES sau khi mã được lưu trên Cloud

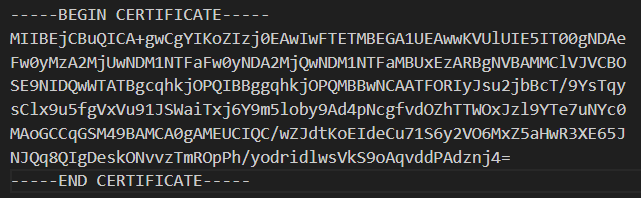
* Dữ liệu lưu trên cloud sau khi mã hóa:



Hình 4.13. Những trường dữ liệu quan trọng đều được mã và lưu trên Cloud

### Thiết lập kết nối SSL/TLS

* Certificate cho server:



Hình 4.14. Tạo certificate cho server

**DANH MỤC TÀI LIỆU THAM KHẢO**

1. *Survey on Revocation in Ciphertext‐Policy Attribute‐ Based Encryption* Trích dẫn: Al-Dahhan, R. R., Shi, Q., Lee, G. M., & Kifayat, K. (2019). Survey on revocation in ciphertext-policy attribute-based encryption. Sensors, 19(7), 1695.
2. *Performance Analysis of the Encryption Algorithms as Solution to Cloud Database Security* Baba, M. A., Yusuf, A., Ahmad, A., & Maijama’a, L. (2014). Performance analysis of the encryption algorithms as solution to cloud database security. International Journal of Computer Applications, 99(14), 24-31.
3. Zickau, S., Thatmann, D., Butyrtschik, A., Denisow, I., & Küpper, A. (2016, March). Applied attribute-based encryption schemes. In 19th International ICIN Conference-Innovations in Clouds, Internet and Networks (pp. 1-3).