



ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN MẬT MÃ HỌC

**ĐỀ TÀI: Ứng dụng Ciphertext-Policy Attribute-Based Encryption
Thực thi Kiểm soát Truy cập cho Hồ sơ Sức khỏe Điện tử**

Môn học: NT219_Mật mã học

Lớp: NT219.P21.ANTT

Giảng viên hướng dẫn: Nguyễn Ngọc Tự

Sinh viên thực hiện:

23520543_Trần Việt Hoàng

23520385_Phan Nguyễn Huy Duy

23520247_Hoàng Quốc Đạt

MỤC LỤC

LỜI CẢM ƠN	5
LỜI MỞ ĐẦU	6
Chương 1. GIỚI THIỆU	7
1.1. Tổng quan đề tài.....	7
1.2. Đặt vấn đề.....	7
1.2.1. Bối cảnh	8
1.2.2. Các bên liên quan.....	8
1.3. Các tài sản chính cần đảm bảo	9
1.4. Các rủi ro bảo mật	9
1.5. Mục tiêu đề tài.....	10
1.6. Phạm vi Nghiên cứu và Triển khai.....	11
Chương 2: GIẢI PHÁP	12
2.1. Xác thực người dùng	12
2.2. AES (Advanced Encryption Standard):	12
2.3. CP-ABE (Ciphertext-Policy Attribute-Based Encryption):	13
2.4. ABAC (Attribute-based Access Control):.....	14
2.5. Quản lý Khóa, Mã hóa Dữ liệu và Bảo mật Truyền thông	15
2.5.1. Các Loại Khóa và Vai Trò:	15
2.5.2. Bảo mật Truyền tải:	15
Chương 3: TÀI NGUYÊN CỦA MÔ HÌNH	16
3.1. Công cụ và tài nguyên.....	16
3.1.1. Ngôn ngữ lập trình Python và framework Django.....	16
3.1.2. Môi trường triển khai.....	16
3.1.3. Các thư viện hỗ trợ chính.....	16
3.1.3.1. Charm-Crypto.....	16
3.1.3.2. JWT	16
3.1.3.3. Pyodide	17
3.1.3.4. OpenSSL, GMP, PBC	17
3.2. Kiến trúc hệ thống	17
3.3. Luồng hoạt động.....	18
3.3.1. Đăng nhập	18
3.3.2. Upload Dữ liệu:.....	18
3.3.3. Truy cập/Tải Dữ liệu:.....	18
Chương 4: TRIỂN KHAI.....	19
4.1. Khởi tạo hệ thống	19
4.1.1. Gọi CPABEHndler.run_system_setup():	19

4.1.2. Thực thi <code>f_cpabe_setup()</code> :	19
4.1.3. Kết quả của Quá trình Khởi tạo:	20
4.2. Cấp phát access token gồm thuộc tính	20
4.2.1. Tùy chỉnh Payload của JWT để nhúng Thuộc tính Người dùng	21
4.2.2. Thực hiện đăng nhập qua API	22
4.3. Tạo và phân phối Public, secret key	23
4.3.1. Cung cấp Khóa Công Khai Hệ thống (Public Key - PK)	23
4.3.2. Tạo và Cấp phát Khóa Bí Mật Người Dùng (Secret Key - SK)	24
4.4. Mã hóa	26
4.4.1. Mã hóa ở phía Data Creators	26
4.4.2. Quy trình mã hóa tại phía Data Creators	30
4.5. Giải mã tại phía Data Users	34
4.5.1. Tìm các mảnh hồ sơ của bệnh nhân dựa vào <code>patient_id</code>	34
4.5.2. Truy cập vào một mảnh bản ghi sức khỏe	35
4.5.3. Tiến hành giải mã	37
4.5.4. Quy trình kiểm tra ABAC	39
4.6. Quy trình giải mã dữ liệu tại phía Data Users	41
Chương 5: TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN	45
5.1. Kết quả Đạt được	45
5.2. Hạn chế của Đề tài	46
5.3. Hướng Phát triển trong Tương lai	46
5.4. Kết luận	47
Chương 6: THAM KHẢO	48

LỜI CẢM ƠN

Lời đầu tiên, tập thể nhóm 14 thuộc lớp NT219.P21.ANTT xin gửi lời cảm ơn chân thành và tri ân sâu sắc đến thầy Nguyễn Ngọc Tự – giảng viên môn Mật mã học, Khoa Mạng máy tính và Truyền thông, Trường Đại học Công nghệ Thông tin, ĐHQG-HCM.

Chúng em xin chân thành cảm ơn thầy vì sự tận tâm chỉ dạy, hỗ trợ và hướng dẫn xuyên suốt quá trình thực hiện đồ án. Những buổi thảo luận cùng thầy không chỉ giúp chúng em giải quyết những khó khăn gặp phải mà còn mở rộng kiến thức chuyên môn, giúp chúng em có được định hướng rõ ràng và tự tin hơn trong việc triển khai đề tài.

Chúng em cũng xin gửi lời cảm ơn đến các thành viên trong nhóm đã cùng nhau làm việc với tinh thần nghiêm túc, trách nhiệm và hỗ trợ lẫn nhau trong suốt quá trình triển khai đề tài. Từ việc thu thập tài liệu, đề xuất ý tưởng đến hiện thực hóa giải pháp, mỗi cá nhân đều đã góp phần tạo nên kết quả chung.

Mặc dù đã nỗ lực hết mình, nhưng do hạn chế về mặt thời gian và kinh nghiệm thực tiễn, đồ án khó tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự góp ý, phản hồi từ thầy để hoàn thiện đề tài một cách tốt hơn trong tương lai.

Một lần nữa, chúng em xin chân thành cảm ơn thầy vì sự đồng hành và hỗ trợ quý báu trong suốt quá trình học tập và thực hiện đồ án này.

TP. Hồ Chí Minh, tháng 06 năm 2025
Nhóm 14, Lớp NT219.P21.ANTT

LỜI MỞ ĐẦU

Sự phát triển của Hồ sơ Sức khỏe Điện tử (EHR) mang lại nhiều lợi ích cho ngành y tế, tuy nhiên, việc bảo vệ thông tin y tế cá nhân nhạy cảm là một thách thức lớn. Các cơ chế kiểm soát truy cập truyền thống thường không đủ linh hoạt để quản lý quyền truy cập chi tiết trong môi trường chia sẻ dữ liệu y tế phức tạp, nơi thông tin được tạo ra từ nhiều nguồn và cần được tiếp cận bởi nhiều đối tượng người dùng với các vai trò khác nhau.

Trước yêu cầu cấp thiết đó, đề tài "**Ứng dụng Ciphertext-Policy Attribute-Based Encryption thực thi kiểm soát truy cập cho Hồ sơ Sức khỏe Điện tử**" được thực hiện. Dự án tập trung vào việc nghiên cứu và triển khai một giải pháp bảo mật sử dụng mã hóa dựa trên thuộc tính CP-ABE (scheme Waters11) kết hợp với JSON Web Tokens (JWT) ký bất đối xứng. Mục tiêu là xây dựng một hệ thống cho phép mã hóa dữ liệu EHR với các chính sách truy cập động do người tạo dữ liệu định nghĩa, đảm bảo chỉ những người dùng sở hữu đúng các thuộc tính cần thiết mới có thể giải mã thông tin.

Báo cáo này sẽ trình bày kiến trúc, các giải pháp kỹ thuật đã triển khai, kết quả đạt được và những định hướng phát triển tiếp theo của dự án, nhằm góp phần giải quyết bài toán bảo mật và kiểm soát truy cập cho dữ liệu y tế điện tử.

Chương 1. GIỚI THIỆU

1.1. Tổng quan đề tài

- Tên đề tài: Ứng dụng Ciphertext-Policy Attribute-Based Encryption thực thi kiểm soát truy cập cho Hồ sơ Sức khỏe Điện tử
- Thời gian thực hiện: từ tháng 4/2025 đến khoảng tháng 6/2025
- Mã nguồn: [Github](#)
- Giảng viên hướng dẫn: Nguyễn Ngọc Tụ
- Sinh viên thực hiện và mức độ đóng góp:

Mã số sinh viên	Tên	Mức độ đóng góp
23520543	Trần Việt Hoàng	70%
23520247	Hoàng Quốc Đạt	20%
23520385	Phan Nguyễn Huy Duy	10%

1.2. Đặt vấn đề

Trong thời đại chuyển đổi số diễn ra mạnh mẽ trên toàn cầu, lĩnh vực y tế đang chứng kiến sự thay đổi đáng kể thông qua việc ứng dụng công nghệ thông tin vào quá trình quản lý và chăm sóc sức khỏe. Một trong những thành tựu quan trọng nhất là sự ra đời và phát triển của **Hồ sơ sức khỏe điện tử (EHR - Electronic Health Record)**. EHR thay thế cho hồ sơ bệnh án giấy truyền thống, lưu trữ toàn diện thông tin y tế của bệnh nhân một cách có hệ thống, từ tiền sử bệnh, kết quả xét nghiệm, thông tin tiêm chủng, đơn thuốc, đến các ghi chú của bác sĩ và hình ảnh chẩn đoán. Việc số hóa này mang lại nhiều lợi ích to lớn như cải thiện chất lượng chẩn đoán và điều trị, giảm thiểu sai sót y khoa, tối ưu hóa quy trình làm việc của nhân viên y tế, và tạo điều kiện cho việc phân tích dữ liệu y tế quy mô lớn phục vụ nghiên cứu và hoạch định chính sách.

Tuy nhiên, việc lưu EHR, vốn chứa đựng những thông tin cá nhân và y tế cực kỳ nhạy cảm, cũng đặt ra những thách thức vô cùng lớn về **bảo mật và quyền riêng tư**. Dữ liệu EHR trở thành mục tiêu hấp dẫn cho các cuộc tấn công mạng, và bất kỳ sự rò rỉ hay truy cập trái phép nào cũng có thể gây ra những hậu quả nghiêm trọng cho bệnh nhân, làm suy giảm niềm tin vào hệ thống y tế số và vi phạm các quy định pháp lý nghiêm ngặt về bảo vệ dữ liệu cá nhân.

Do đó, việc nghiên cứu và triển khai các giải pháp bảo mật dữ liệu toàn diện và cơ chế kiểm soát truy cập chi tiết, linh hoạt cho hệ thống EHR là một yêu cầu cấp thiết, nhằm đảm bảo rằng lợi ích của công nghệ có thể được phát huy tối đa mà không làm tổn hại đến quyền riêng tư và an toàn thông tin của người bệnh.

1.2.1. Bối cảnh

Trong hệ sinh thái y tế hiện đại, việc quản lý và chia sẻ Hồ sơ Sức khỏe Điện tử (EHR) không còn giới hạn trong phạm vi một cơ sở y tế đơn lẻ. Thay vào đó, một bệnh nhân thường nhận được sự chăm sóc từ nhiều nguồn khác nhau: bác sĩ gia đình, các bác sĩ chuyên khoa tại các bệnh viện khác nhau, phòng xét nghiệm, trung tâm chẩn đoán hình ảnh, và thậm chí cả các thiết bị theo dõi sức khỏe cá nhân. Dữ liệu y tế của một bệnh nhân do đó được tạo ra và đóng góp từ nhiều Người Tạo Dữ Liệu này, hình thành nên một bức tranh sức khỏe toàn diện nhưng phân mảnh.

Bài toán cốt lõi mà dự án này giải quyết là làm thế nào để cho phép các bên liên quan hợp pháp – từ bác sĩ điều trị, y tá, đến chính bệnh nhân và các nhà nghiên cứu được ủy quyền – truy cập đúng và chỉ đúng những phần thông tin trong EHR mà họ được phép xem, dựa trên vai trò, chuyên môn, mối quan hệ với bệnh nhân, và tính chất của dữ liệu. Các cơ chế kiểm soát truy cập dựa trên vai trò (RBAC) truyền thống thường tỏ ra không đủ linh hoạt để đáp ứng yêu cầu kiểm soát truy cập chi tiết, theo ngữ cảnh và thay đổi động này. Ví dụ, một chính sách cần cho phép bác sĩ A xem toàn bộ bệnh sử nhưng chỉ cho phép y tá B xem các ghi chú điều dưỡng gần đây, hoặc cho phép nhà nghiên cứu C truy cập dữ liệu thống kê đã được ẩn danh hoàn toàn.

Do đó, dự án này đề xuất một hệ thống EHR với cơ chế kiểm soát truy cập dựa trên thuộc tính mật mã (Ciphertext-Policy Attribute-Based Encryption - CP-ABE). Trong đó, mỗi mảnh dữ liệu EHR được mã hóa với một chính sách truy cập riêng do người tạo dữ liệu định nghĩa. Chỉ những người dùng sở hữu một tập hợp các thuộc tính (ví dụ: vai trò, chuyên khoa, chứng chỉ) thỏa mãn chính sách đó mới có thể giải mã và xem được nội dung. Hệ thống này được thiết kế với các thành phần chuyên biệt: một Auth Center để quản lý danh tính, thuộc tính và khóa CP-ABE; và một Resource Server để lưu trữ an toàn các bản ghi EHR đã mã hóa, đồng thời thực thi kiểm soát truy cập API dựa trên các thuộc tính được chứng thực qua JSON Web Tokens (JWT). Điều này nhằm đảm bảo dữ liệu nhạy cảm được bảo vệ mạnh mẽ, đồng thời cho phép chia sẻ thông tin một cách linh hoạt và có kiểm soát giữa các bên liên quan trong quá trình chăm sóc sức khỏe và nghiên cứu.

1.2.2. Các bên liên quan

1.2.2.1 Auth Center

Là bên trung tâm xác thực, chịu trách nhiệm xác thực người dùng, phân phối JWT token cho việc truy cập API, và phân phối public key cho Data Creators để mã hóa và secret key dựa trên thuộc tính của người dùng để giải mã dữ liệu trong hệ thống.

1.2.2.2 Resource Server

Là nơi lưu trữ dữ liệu, bao gồm dữ liệu nhạy cảm đã mã hóa, và cả khóa mã hóa (đã được mã hóa), cung cấp khả năng truy xuất dữ liệu cho người dùng.

1.2.2.3 Data Creators

Là những người tạo ra các bản ghi dữ liệu, định nghĩa chính sách truy cập cho bản ghi này và thực hiện mã hóa dữ liệu trên chính máy của họ dựa trên chính sách đó trước khi đăng tải dữ liệu đã mã hoá lên cơ sở dữ liệu nhằm lưu trữ.

1.2.2.4 Data Users

Là các người dùng, những người đã được xác thực, có các quyền hợp lệ để lấy dữ liệu về và giải mã nếu thỏa mãn các chính sách được áp dụng cho dữ liệu.

1.3. Các tài sản chính cần đảm bảo

Trong hệ thống Hồ sơ Sức khỏe Điện tử (EHR) được đề xuất, việc xác định và bảo vệ các tài sản thông tin cốt lõi là ưu tiên hàng đầu. Các tài sản chính cần được đảm bảo an toàn bao gồm:

- **Dữ liệu Hồ sơ Sức khỏe Điện tử:** Đây là tài sản quan trọng nhất, chứa thông tin y tế cá nhân và cực kỳ nhạy cảm của bệnh nhân. Tính bí mật, toàn vẹn và sẵn sàng của dữ liệu này phải được bảo vệ ở mức cao nhất. Dữ liệu này tồn tại dưới dạng đã mã hóa khi lưu trữ và chỉ được giải mã bởi người dùng có thẩm quyền.
- **Khóa Bí Mật Chủ CP-ABE:** Là "chìa khóa gốc" của toàn bộ hệ thống mã hóa CP-ABE. Nếu MSK bị lộ, kẻ tấn công có thể tạo ra bất kỳ Khóa Bí Mật Người Dùng nào và giải mã toàn bộ dữ liệu được bảo vệ bằng CP-ABE. MSK phải được lưu trữ và quản lý với các biện pháp an ninh nghiêm ngặt nhất tại Auth Center.
- **Khóa Bí Mật của Người Dùng:** Mỗi người dùng được cấp một khóa riêng dựa trên thuộc tính của họ. Khóa này cho phép người dùng giải mã các phần dữ liệu EHR mà họ được phép truy cập. Việc lộ khóa của một người dùng sẽ làm lộ dữ liệu mà người đó có quyền.
- **Khóa Riêng Tư Ký JWT:** Được Auth Center sử dụng để ký các JSON Web Tokens. Nếu khóa này bị lộ, kẻ tấn công có thể tạo ra các JWT giả mạo, mạo danh người dùng và truy cập trái phép vào các API của hệ thống. Khóa này cũng cần được bảo vệ nghiêm ngặt tại Auth Center.
- **Thông tin Định danh và Xác thực Người dùng:** Bao gồm tên đăng nhập, mật khẩu đã được hash, thông tin liên lạc, và các thuộc tính được gán cho người dùng.
- **Chính sách Truy cập:** Bao gồm cả các chuỗi chính sách CP-ABE được nhúng khi mã hóa dữ liệu và các quy tắc ABAC được định nghĩa trên Resource Server để kiểm soát truy cập API.

1.4. Các rủi ro bảo mật

Có thể chia thành 2 nhóm chính:

- Bên ngoài (tấn công đường truyền, giả mạo các bên liên quan)
- Bên trong (do nội bộ, do vi phạm từ phía cung cấp cloud)

1.4.1 Đối với tấn công đường truyền

Giải pháp là thiết lập kết nối bằng TLS chỉ sử dụng 1.2 hoặc 1.3.

1.4.2 Đối với giả mạo các bên liên quan

Xác thực và phân quyền chi tiết cho từng người, ai có thể truy cập phần nào của từng mảnh dữ liệu.

1.4.3 Đối với rủi ro nội bộ

Việc phân quyền chi tiết sẽ hạn chế tối đa việc bị xâm phạm dữ liệu đối với nội bộ, không phải người nào cũng có quyền truy cập các phần dữ liệu.

1.4.4 Đối với rủi ro từ phía cung cấp cloud

Cloud được xem là một đối tượng semi-trusted, nghĩa là chỉ có thể tin tưởng ở một mức độ nhất định. Mặc dù có thể coi là bên đáng tin cậy, nhưng vẫn có xác suất xảy ra rò rỉ dữ liệu do đơn vị cung cấp bị tấn công hoặc nhân viên nhà cung cấp cố ý xem dữ liệu. Giải pháp ở đây là mã hóa dữ liệu trước khi lưu trữ dữ liệu trên cloud.

1.5. Mục tiêu đề tài

Đề tài này nhằm giải quyết các thách thức về bảo mật và kiểm soát truy cập chi tiết trong việc quản lý và chia sẻ Hồ sơ Sức khỏe Điện tử (EHR) trong môi trường có nhiều bên tham gia. Các mục tiêu cụ thể bao gồm:

- **Nghiên cứu và ứng dụng Ciphertext-Policy Attribute-Based Encryption (CP-ABE):** Tìm hiểu sâu về lý thuyết và các scheme CP-ABE (cụ thể là Waters11) để áp dụng vào việc mã hóa dữ liệu EHR, cho phép người tạo dữ liệu định nghĩa chính sách truy cập động dựa trên thuộc tính.
- **Thiết kế và xây dựng hệ thống EHR phân tán:** Phát triển một kiến trúc bao gồm Auth Center(AC) độc lập và Resource Server(RS), trong đó AC quản lý danh tính, thuộc tính và khóa CP-ABE, còn RS quản lý việc lưu trữ và truy xuất dữ liệu EHR đã mã hóa.
- **Triển khai cơ chế xác thực mạnh mẽ và kiểm soát truy cập API:** Sử dụng JWT ký bằng thuật toán bất đối xứng (ES256) để xác thực người dùng khi truy cập API. Kết hợp với ABAC trên RS để kiểm soát quyền truy cập vào các API lấy ciphertext, dựa trên thuộc tính của người dùng được nhúng trong JWT.
- **Hiện thực hóa việc mã hóa và giải mã phía Client:** Cho phép Data Creators mã hóa dữ liệu EHR (sử dụng mã hóa lai CP-ABE + AES) trên thiết bị của họ trước khi upload, và Data Users giải mã dữ liệu trên thiết bị của họ sau khi tải về, tăng cường tính bảo mật đầu cuối.
- **Đảm bảo các tính chất an toàn cơ bản:** Tập trung vào tính bí mật (Confidentiality) của dữ liệu EHR thông qua mã hóa, tính toàn vẹn (Integrity) thông qua các cơ chế như AES-GCM và chữ ký JWT, và tính sẵn sàng (Availability) của hệ thống cho người dùng hợp pháp.
- **Đánh giá giải pháp:** Phân tích tính an toàn, hiệu quả và những hạn chế của mô hình đề xuất, đặc biệt trong môi trường triển khai không tin cậy(zero-trust).

1.6. Phạm vi Nghiên cứu và Triển khai

Do giới hạn về thời gian và nguồn lực của một đồ án môn học, đề tài sẽ tập trung vào các phạm vi chính sau:

- **Scheme CP-ABE:** Nghiên cứu và triển khai chủ yếu dựa trên scheme Waters (2011).
- **Kiến trúc hệ thống:** Xây dựng hai thành phần backend chính là Auth Center và Resource Server bằng Django framework. Phát triển các API cần thiết cho việc xác thực, quản lý khóa, upload và truy xuất dữ liệu mã hóa.
- **Mã hóa/Giải mã phía Client:** Tập trung vào việc triển khai logic mã hóa lai (CP-ABE cho KEK, AES-GCM cho dữ liệu) và giải mã tương ứng ở phía client sử dụng JavaScript và Pyodide để chạy thư viện Charm-Crypto. Giao diện người dùng cho việc định nghĩa chính sách CP-ABE sẽ được xây dựng ở mức độ cơ bản, cho phép chọn và kết hợp các thuộc tính đã định nghĩa trước.
- **Quản lý Thuộc tính:** Việc tạo và gán thuộc tính (cả ABAC và CP-ABE) cho người dùng sẽ được thực hiện chủ yếu thông qua giao diện quản trị của Django trên Auth Center.
- **Các tính năng EHR đầy đủ:** Đề tài không nhằm mục đích xây dựng một hệ thống EHR hoàn chỉnh với đầy đủ các chức năng nghiệp vụ y tế, mà tập trung vào khía cạnh bảo mật và kiểm soát truy cập dữ liệu EHR.
- **Bảo mật hạ tầng và vận hành nâng cao:** Các biện pháp bảo mật ở tầng hạ tầng vật lý, mạng lưới phức tạp, hay các quy trình vận hành an ninh chi tiết trên môi trường thực tế sẽ không được triển khai sâu.

Chương 2: GIẢI PHÁP

Nhằm hiện thực hóa các mục tiêu đã đề ra, ứng dụng được phát triển bằng ngôn ngữ lập trình Python, sử dụng Django framework để triển khai một hệ thống mã hóa và kiểm soát truy cập dữ liệu, kết hợp với các kỹ thuật và công nghệ hiện đại để đảm bảo các yêu cầu về bảo mật, toàn vẹn và sẵn sàng của hệ thống.

2.1. Xác thực người dùng

Để đảm bảo an toàn và khả năng mở rộng, hệ thống áp dụng mô hình xác thực và ủy quyền phân tán. Trung tâm của mô hình này là **Auth Center (AC)**, một dịch vụ độc lập chịu trách nhiệm quản lý danh tính người dùng và cấp phát JSON Web Tokens (JWT).

Khi người dùng yêu cầu đăng nhập, họ sẽ tương tác với AC. Sau khi xác thực thành công thông tin đăng nhập, AC sẽ tạo và ký một JWT (sử dụng thuật toán ký bất đối xứng ES256 với cặp khóa private/public riêng của AC). JWT này chứa các thông tin định danh (user_id) và các thuộc tính (ví dụ: vai trò, phòng ban, mức độ can thiệp,...) cần thiết cho việc kiểm soát truy cập của người dùng.

Các dịch vụ nghiệp vụ khác, cụ thể là **Resource Server (RS)** – nơi xử lý các API liên quan đến hồ sơ bệnh án điện tử (tìm kiếm, quản lý, cung cấp dữ liệu đã mã hóa) – sẽ yêu cầu JWT hợp lệ cho mỗi yêu cầu truy cập. Khi nhận được yêu cầu từ client kèm theo JWT, RS sẽ tự xác minh tính hợp lệ của token này bằng cách sử dụng Public Key của AC (đã được RS lưu trữ an toàn). Quá trình xác minh này bao gồm kiểm tra chữ ký, thời gian hết hạn, và các claim quan trọng khác (như issuer, audience nếu có).

Ưu điểm của kiến trúc này là sự tách biệt rõ ràng giữa dịch vụ xác thực và dịch vụ tài nguyên, tăng cường bảo mật (RS không cần lưu trữ thông tin nhạy cảm như mật khẩu), giảm tải cho AC (RS xác minh token cục bộ), và phù hợp với các nguyên tắc của kiến trúc Zero Trust. Auth Center cũng là nơi quản lý và cấp phát Khóa Công Khai (PK) và Khóa Bí Mật (SK) cho hệ thống mã hóa CP-ABE.

2.2. AES (Advanced Encryption Standard):

AES là thuật toán mã hóa đối xứng tiêu chuẩn hiện nay và được chuẩn hóa bởi NIST, được thiết kế để cung cấp cả tính bảo mật và toàn vẹn cho dữ liệu và sử dụng rộng rãi trong các hệ thống lớn yêu cầu bảo mật cao như ngân hàng, y tế, doanh nghiệp, chính phủ,... Với mode AES256-GCM mà nhóm sử dụng, AES có thể mang lại:

- Tính bảo mật: AES cung cấp khả năng mã hóa mạnh, bảo vệ dữ liệu khỏi việc bị truy cập trái phép. Với việc triển khai AES256-GCM có độ dài khóa là 256-bit cung cấp mức độ bảo mật cao, tránh bị các cuộc tấn công nhằm bẻ khóa mật khẩu, phù hợp với dữ liệu nhạy cảm trong hệ thống PHR.
- Tính toàn vẹn: Mode GCM của AES cung cấp một tag giúp xác nhận dữ liệu không bị thay đổi

- Tốc độ: AES-GCM kết hợp việc mã hóa và xác thực trong một bước, mang lại tốc độ nhanh và chi phí tính toán thấp, vô cùng lý tưởng cho việc mã hóa một lượng lớn dữ liệu mà không làm hệ thống bị chậm.
- Phổ biến: AES được hỗ trợ bởi nhiều thư viện mật mã và có thể dễ dàng tích hợp vào trong các nền tảng Cloud.

Tuy nhiên, chỉ sử dụng mỗi AES là không thể vì bên cạnh những ưu điểm mà nó mang lại, cũng có những nhược điểm như:

- Vấn đề quản lý khóa: Vì AES là thuật toán mã hóa đối xứng, nó sẽ sử dụng một khóa bí mật (private key) cho việc mã hóa và giải mã. Khóa bí mật này phải được chia sẻ an toàn giữa người mã hóa (Data owner: bác sĩ) và người giải mã (Data user: bệnh nhân, bác sĩ khác), nếu khóa này bị rò rỉ thì dữ liệu sẽ bị lộ
- Phụ thuộc vào kênh truyền tải: Khóa AES phải được truyền tải qua kênh an toàn như HTTPS, TLS để không bị các cuộc tấn công xen giữa (Man in the middle) dẫn tới lộ khóa
- Không hỗ trợ việc kiểm soát truy cập chi tiết: Vì bất kỳ ai có khóa bí mật cũng có thể xem được thông tin mà không có cơ chế nào để kiểm tra xem người dùng có quyền để đọc dữ liệu hay không. Không phù hợp với hệ thống PHR với nhiều loại người dùng như bác sĩ, y tá, nhà nghiên cứu, bệnh nhân. Mỗi vai trò chỉ có thể được xem một số dữ liệu nhất định

Dựa vào những ưu điểm mà AES mang lại, ta có thể sử dụng để mã hóa dữ liệu của hệ thống trước khi lưu trữ trên đám mây. Điều này giúp mang lại tốc độ lưu trữ và trích xuất thông tin nhanh chóng, đồng thời cũng đảm bảo được tính an toàn, bảo mật của thông tin, ngăn chặn được việc thông tin nhạy cảm bị rò rỉ, kể cả bên thứ ba (nhà cung cấp đám mây) cũng không thể đọc được dữ liệu nhạy cảm, khi dữ liệu bị rò rỉ cũng không thể đọc được.

Để khắc phục những hạn chế của AES hiện có, có thể triển khai thêm CP-ABE.

2.3. CP-ABE (Ciphertext-Policy Attribute-Based Encryption):

Đây là một thuật toán mã hóa tiên tiến, cho phép mã hóa dữ liệu dựa trên các thuộc tính của người dùng và chính sách truy cập, đặc biệt hiệu quả trong các hệ thống phân tán như cloud.

Ý tưởng: Trong CP-ABE,

- Bản mã (ciphertext) được liên kết với một chính sách truy cập, định nghĩa những người dùng với thuộc tính cần phải có để được phép giải mã.
- Mã hóa: Trong mô hình mã hóa lai của hệ thống, CP-ABE (sử dụng Public Key hệ thống) được dùng để mã hóa Khóa Mã Hóa Dữ Liệu đối xứng (KEK - ví dụ, một khóa AES). Người dùng sau đó sử dụng Secret Key CP-ABE cá nhân của họ (được tạo từ Master Secret Key và tập thuộc tính của họ) để giải mã KEK này, rồi dùng KEK để giải mã dữ liệu chính.
- Giải mã: Chỉ những người dùng có chính sách truy cập phù hợp với phần trong bản mã mới có thể giải mã.

Do đó, CP-ABE cung cấp khả năng:

- Kiểm soát truy cập chi tiết: CP-ABE cung cấp mức độ bảo mật cao với việc mã hóa dữ liệu dựa trên các chính sách truy cập phức tạp, linh hoạt dựa trên thuộc tính của người dùng, chứ không chỉ dựa trên vai trò (Role-based) của người dùng.
- Quản lý khóa: CP-ABE tập trung vào việc quản lý thuộc tính thay vì quản lý từng cặp khóa riêng cho người dùng hoặc nhóm người dùng, giảm chi phí so với việc quản lý từng cặp khóa riêng lẻ trong hệ thống lớn với nhiều dữ liệu.
- Hỗ trợ mã hóa lai: CP-ABE được sử dụng để mã hóa khóa AES, tận dụng hiệu suất cao của AES cho dữ liệu lớn và khả năng kiểm soát truy cập của CP-ABE, tối ưu hóa vấn đề bảo mật và hiệu suất.

Tuy nhiên, các sơ đồ ABE vẫn còn tồn tại hạn chế:

- Hiệu suất tính toán: Các sơ đồ CP-ABE có thời gian tính toán lâu, tăng tỉ lệ thuận với độ phức tạp của cây thuộc tính.
- Kích thước: Kích thước khóa và bản mã lớn, gây khó khăn khi truyền tải dữ liệu trên thiết bị với tài nguyên hạn chế.
- Phức tạp trong triển khai và tích hợp: Việc xây dựng cây chính sách, triển khai và tích hợp vào dự án phức tạp, đòi hỏi kiến thức.

Hệ thống được triển khai CP-ABE theo scheme Waters11, có nhiều ưu điểm so với CP-ABE cổ điển.

- Truy cập linh hoạt: scheme Waters11 cho phép định nghĩa các phép định nghĩa các phép toán AND, OR và threshold để biểu diễn chính sách.
- Hiệu suất: Waters11 cải thiện hiệu suất mã hóa và giải mã so với CP-ABE cổ điển.
- Khả năng chống thông đồng (Collusion Resistance): Nhiều người dùng với các tập thuộc tính khác nhau không thể kết hợp khóa bí mật của họ để giải mã một bản mã mà không ai trong số họ có đủ thuộc tính để giải mã riêng lẻ.

Tuy nhiên, scheme này cũng có một số điểm yếu:

- Kích thước ciphertext phụ thuộc vào độ phức tạp của chính sách truy cập.
- Kích thước secret key phụ thuộc vào số lượng thuộc tính của người dùng.
- Kích thước public key phụ thuộc vào tổng số thuộc tính(uni_size)

2.4. ABAC (Attribute-based Access Control):

CP-ABE cung cấp khả năng kiểm soát truy cập chi tiết bằng cách mã hóa khóa AES với chính sách truy cập dựa trên thuộc tính, nhưng việc sử dụng CP-ABE một mình có một số hạn chế, đặc biệt về hiệu suất và quản lý truy cập. ABAC được tích hợp để bổ sung và khắc phục hạn chế này.

Hệ thống EHR này triển khai ABAC có nhiệm vụ như một lớp bảo mật bổ sung cho phần CP-ABE, kiểm tra quyền truy cập của người dùng dựa vào thuộc tính trước khi người dùng được phép lấy bản dữ liệu đã mã hóa về và thử giải mã nó bằng khóa CP-ABE của người dùng, đồng thời cũng giúp giảm tải hành động giải mã, cho phép kiểm soát truy cập linh hoạt hơn thay vì chỉ là giải mã được hay không.

2.5. Quản lý Khóa, Mã hóa Dữ liệu và Bảo mật Truyền thông

An toàn cho việc quản lý khóa và dữ liệu là trọng tâm của hệ thống. Quy trình này bao gồm việc tạo, bảo vệ, phân phối các loại khóa khác nhau và mã hóa dữ liệu cả khi lưu trữ lẫn khi truyền tải.

2.5.1. Các Loại Khóa và Vai Trò:

Hệ thống sử dụng nhiều loại khóa với các mục đích riêng biệt:

- **Khóa Bí Mật Chủ CP-ABE (Master Secret Key - MSK):** Đây là khóa tối quan trọng của hệ thống CP-ABE. Nó được tạo ra một lần duy nhất trong quá trình setup hệ thống và được lưu trữ cực kỳ an toàn tại Auth Center (AC). MSK không bao giờ được phân phối ra bên ngoài AC. Nó được AC sử dụng để tạo ra Khóa Bí Mật cá nhân cho từng người dùng.
 - Lưu trữ MSK: Cần các giải pháp như HSM hoặc KMS khi triển khai thực tế.
 - Khóa Công Khai CP-ABE (Public Key - PK): Cũng được tạo ra trong quá trình setup bởi AC. PK được sử dụng để mã hóa Khóa Mã Hóa Dữ Liệu (KEK). PK này được AC cung cấp công khai (qua API) cho các thành phần cần thực hiện mã hóa, ví dụ như Resource Server (RS) hoặc Client Application của người tạo dữ liệu.
- **Khóa Bí Mật CP-ABE Người Dùng (Secret Key - SK):** Mỗi người dùng được AC cấp một SK duy nhất dựa trên tập thuộc tính CP-ABE của họ và MSK. Người dùng sử dụng SK này để giải mã các KEK mà họ có quyền truy cập (tức là thuộc tính của họ thỏa mãn chính sách CP-ABE của KEK đó). SK được cấp phát an toàn cho người dùng và người dùng có trách nhiệm tự bảo quản.
- **Khóa Mã Hóa Dữ Liệu (Data Encryption Key - DEK, hay Key Encryption Key - KEK cho khóa AES):** Đây là một khóa đối xứng (ví dụ: khóa AES-256) được tạo ngẫu nhiên mỗi khi một "mảnh thông tin EHR" cần được mã hóa.
 - DEK này được dùng để mã hóa nội dung thực sự của mảnh thông tin EHR bằng thuật toán AES-256-GCM.
 - Sau đó, chính DEK này lại được mã hóa bằng CP-ABE (sử dụng PK hệ thống và một chính sách truy cập do người tạo dữ liệu định nghĩa). Bản mã của DEK được lưu trữ cùng với bản mã của dữ liệu.
- **Khóa Ký JWT :**
 - Auth Center sở hữu một cặp khóa ECC.. Private Key ECC được AC giữ bí mật tuyệt đối và dùng để ký tất cả các JWT được phát hành.
 - Public Key ECC được AC chia sẻ cho Resource Server để có thể xác minh chữ ký của JWT.

2.5.2. Bảo mật Truyền tải:

- Tất cả các giao tiếp mạng giữa Client Application, Auth Center, và Resource Server phải được thực hiện qua HTTPS(chỉ TLS 1.3 hoặc 1.2)
- Điều này đảm bảo rằng tất cả dữ liệu truyền đi, bao gồm thông tin đăng nhập, JWTs, Khóa Công Khai CP-ABE, Khóa Bí Mật CP-ABE, và ciphertext EHR, đều được mã hóa trên đường truyền.
- Việc này giúp bảo vệ chống lại các cuộc tấn công nghe lén và tấn công xen giữa.

Chương 3: TÀI NGUYÊN CỦA MÔ HÌNH

3.1. Công cụ và tài nguyên

3.1.1. Ngôn ngữ lập trình Python và framework Django

- **Auth Center (AC):** Được xây dựng bằng Django, chịu trách nhiệm xác thực người dùng, quản lý thuộc tính, tạo các khóa của hệ thống CP-ABE (PK, MSK), cấp phát Khóa Bí Mật CP-ABE (SK) cho người dùng, và cấp phát JWT.
- **Resource Server (RS):** Cũng được xây dựng bằng Django, chịu trách nhiệm cung cấp API để upload dữ liệu đã mã hóa ở phía Data Creators và API để truy xuất ciphertext. RS cũng xác minh JWT từ AC.
- **Client-Side (JavaScript với Pyodide/Charm-Crypto):** Chịu trách nhiệm chính cho việc mã hóa trước khi đưa lên CSDL và giải mã sau khi lấy dữ liệu đã mã hóa.

3.1.2. Môi trường triển khai

Trong giai đoạn hiện tại của đề án môn học, hệ thống được xây dựng, kiểm thử và vận hành chủ yếu trên môi trường phát triển cục bộ

Auth Center và Resource Server : Hai thành phần này được triển khai dưới dạng các project Django riêng biệt. Mỗi project được xây dựng độc lập để mô phỏng sự tách biệt của các dịch vụ trong một kiến trúc microservices, đảm bảo nguyên tắc Zero-trust khi triển khai trên thực tế.

3.1.3. Các thư viện hỗ trợ chính

3.1.3.1. Charm-Crypto

Charm-Crypto là một framework mã nguồn mở dành cho nghiên cứu và phát triển các giao thức mật mã. Nó được thiết kế để dễ dàng mô phỏng và triển khai các thuật toán mật mã phức tạp, hỗ trợ nhiều loại nhóm mật mã và cung cấp các công cụ để nhanh chóng triển khai các thuật toán mới. Charm-Crypto hỗ trợ nhiều giao thức mật mã bao gồm các giao thức dựa trên cặp (pairing-based) và mật mã đường cong elliptic (elliptic curve cryptography), và là một công cụ quan trọng cho các nhà nghiên cứu trong lĩnh vực mật mã.

3.1.3.2. JWT

JSON Web Token (JWT) là một tiêu chuẩn mở (RFC 7519) định nghĩa một cách nhỏ gọn và khép kín để truyền thông tin an toàn giữa các bên dưới dạng một đối tượng JSON. Thông tin này có thể được xác minh và tin cậy vì nó được ký điện tử. JWT thường được sử dụng để xác thực và ủy quyền trong các ứng dụng web và API. Một JWT bao gồm ba phần được phân tách bằng dấu chấm: Header (chứa thuật toán ký), Payload (chứa các "claims" - thông tin về người dùng và token), và Signature (chữ ký để xác minh tính toàn vẹn).

Trong dự án này, JWT có thêm trường `user_attributes`, được cấp bởi `dj_rest_auth` khi người dùng đăng nhập thành công qua `/api/auth/login` của Auth Center. Auth Center và Resource Server đều sử dụng thư viện `jwt` nhưng với cấu hình khác nhau. Auth Center cấu hình để ký JWT bằng private key và thuật toán bất đối xứng ES256. Resource Server cấu hình để xác minh JWT bằng public key tương ứng của Auth Center và cùng thuật toán. Khi người

dùng truy cập một API yêu cầu cần xác minh danh tính, Resource server sẽ xác minh token dựa trên public-key có được từ Auth_Center. Cách tiếp cận này cho phép xác minh danh tính mà không cần phải truy cập tới Auth Center từ Resource Server, khiến thiết kế trở nên an toàn.

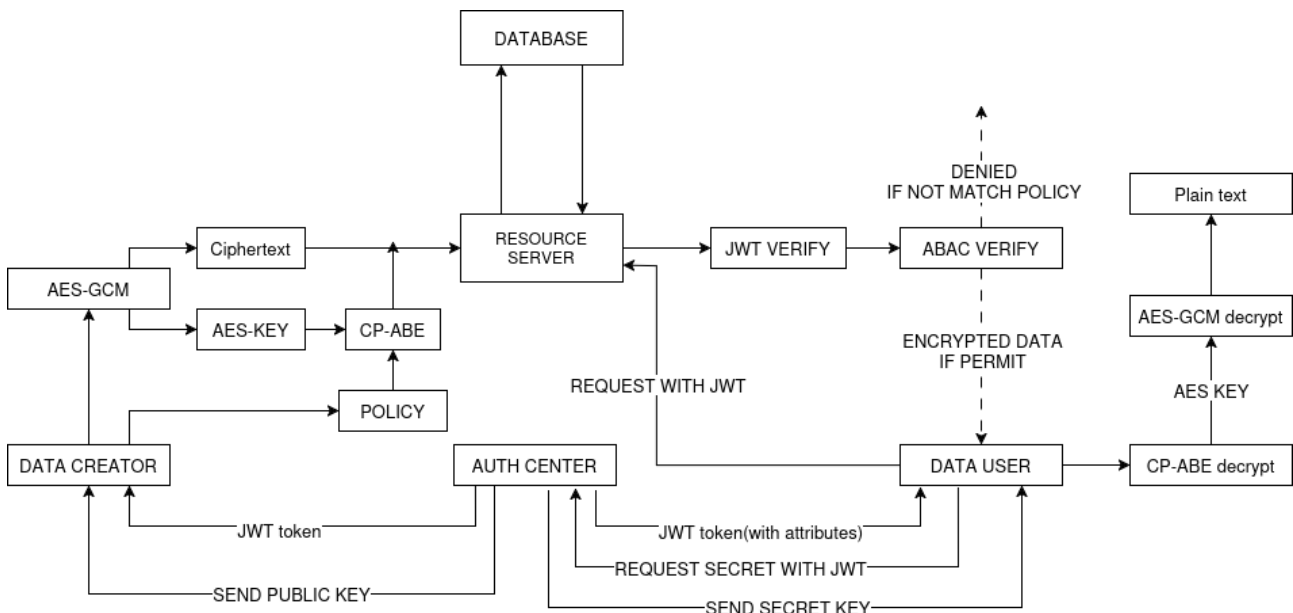
3.1.3.3. Pyodide

Thư viện cho phép build các thư viện python thành các wheel package. Thư viện này không được sử dụng trực tiếp trong đề tài mà dùng để build Charm-Crypto thành wheel package, giúp sử dụng trực tiếp trong code Javascript ở phía client, phục vụ việc mã hóa và giải mã hoàn toàn ở phía client.

3.1.3.4. OpenSSL, GMP, PBC

OpenSSL, Pairing-Based Cryptography, GNU Multiple Precision là các thư viện mã nguồn mở, cung cấp các chức năng cần thiết cho việc tính toán, ghép cặp, mã hóa. Đây là các thư viện phụ thuộc, cần thiết để có thể cài đặt Charm-Crypto.

3.2. Kiến trúc hệ thống



Hệ thống thiết kế gồm 3 thành phần chính::

- **Người Dùng Cuối (End Users):** Bao gồm hai vai trò chính:
 - **Data Creators:** Những cá nhân hoặc bộ phận (bác sĩ, phòng lab, y tá) có trách nhiệm nhập và tạo ra các phần của Hồ sơ Sức khỏe Điện tử (EHR). Họ định nghĩa chính sách truy cập CP-ABE cho dữ liệu họ tạo và mã hóa trên máy của họ trước khi đưa lên lưu trữ.
 - **Data Users:** Những cá nhân (bác sĩ, y tá, bệnh nhân, nhà nghiên cứu,...) có nhu cầu truy cập và xem thông tin EHR cần phải trải qua lớp kiểm tra ABAC để có quyền lấy dữ liệu đã mã hóa dựa trên các thuộc tính và quyền hạn của họ và giải mã dữ liệu đó ngay trên máy của họ.

- Cả Data Creators và Data Users đều tương tác với hệ thống thông qua một cùng 1 giao diện.
- **Auth Center (AC):** Một server độc lập thực hiện các chức năng:
 - Xác thực tất cả người dùng.
 - Quản lý thuộc tính người dùng.
 - Tạo và quản lý Khóa Công Khai (PK) và Khóa Bí Mật Chủ (MSK) của hệ thống CP-ABE.
 - Cấp phát Khóa Bí Mật CP-ABE (SK) cá nhân cho người dùng.
 - Cấp phát JSON Web Tokens (JWT) cho các phiên làm việc đã xác thực.
- **Resource Server (RS):** Server chính nơi:
 - Lưu trữ an toàn các bản ghi EHR đã được mã hóa.
 - Cung cấp API cho Data Creators để upload dữ liệu đã mã hóa.
 - Cung cấp API cho Data Users để truy xuất các bản ghi EHR (dưới dạng ciphertext) sau khi đã xác thực JWT và vượt qua các kiểm tra ABAC để tiến hành giải mã.

3.3. Luồng hoạt động

3.3.1. Đăng nhập

- End Users (từ trang do RS phục vụ) gọi API login của AC.
- AC nhận thông tin và tiến hành xác thực định danh.
- Nếu đúng, AC trả về cho end-user một JWT access token chứa các thông tin và thuộc tính của người dùng, dùng để xác nhận danh tính khi truy cập các API của AC và RS.

3.3.2. Upload Dữ liệu:

- Data Creator tải về public key của hệ thống.
- Data Creator chuẩn bị dữ liệu và chính sách CP-ABE.
- Sử dụng PK và Pyodide/Charm để mã hóa KEK bằng CP-ABE, sau đó mã hóa dữ liệu bằng AES với KEK trên chính máy Data Creator
- Data Creator gửi JWT và bundle dữ liệu đã mã hóa (bao gồm metadata, policy đã dùng, encrypted KEK, encrypted data) đến API upload của RS.
- RS xác minh JWT, kiểm tra ABAC (dựa trên claim trong JWT).
- Nếu được phép, RS lưu trữ dữ liệu đã mã hóa.

3.3.3. Truy cập/Tải Dữ liệu:

- Data User gửi JWT đến API của RS để yêu cầu một bản ghi EHR.
- RS xác minh JWT, kiểm tra ABAC trước khi cho phép lấy dữ liệu đã mã hóa.
- Nếu được phép, RS trả về ciphertext bundle.
- Client sử dụng SK cá nhân, PK hệ thống và Pyodide/Charm để giải mã dữ liệu.

Chương 4: TRIỂN KHAI

4.1. Khởi tạo hệ thống

(*auth_center_project/cpabe_service_app/cpabe_handler.py*)

4.1.1. Gọi CPABEHandler.run_system_setup():

- View sẽ khởi tạo một instance của lớp CPABEHandler.
- Phương thức run_system_setup() trong CPABEHandler được gọi để điều phối quá trình setup.
- Kiểm tra sự tồn tại của khóa: Trước khi tiến hành tạo khóa mới, CPABEHandler kiểm tra xem các file Khóa Công Khai (public_key.bin) và Khóa Bí Mật Chủ (master_key.bin) đã tồn tại trong thư mục(settings.CPABE_CONFIG['KEYS_DIR']) hay chưa. Nếu cả hai file đã tồn tại, hệ thống coi như đã được setup trước đó và sẽ trả về thông báo thành công mà không thực hiện lại quá trình tạo khóa, tránh việc ghi đè không mong muốn lên các khóa quan trọng.
- Nếu các khóa chưa tồn tại, CPABEHandler sẽ gọi hàm f_cpabe_setup() (được import từ f_cpabe.py) và truyền vào các tham số cần thiết:
 - self.actual_scheme_instance: Một instance của lớp Waters11 (từ charm.schemes.abenc.waters11) đã được khởi tạo trong CPABEHandler.__init__ với PairingGroup và uni_size được cấu hình.
 - self.keys_dir: Đường dẫn đến thư mục sẽ lưu trữ các file khóa.
 - pk_filename: Tên file cho Khóa Công Khai (ví dụ: "public_key.bin").
 - msk_filename: Tên file cho Khóa Bí Mật Chủ (ví dụ: "master_key.bin").

```
45
46 def run_system_setup(self):
47     if os.path.exists(self.pk_file_path) and os.path.exists(self.msk_file_path):
48         msg = "Hệ thống CP-ABE (PK, MSK) đã được thiết lập trước đó."
49         logger.info(msg)
50         return True, msg
51
52     try:
53         logger.info(f"Đang thực hiện setup hệ thống CP-ABE vào thư mục: {self.keys_dir}")
54         f_cpabe_setup_util(
55             self.actual_scheme_instance,
56             self.keys_dir,
57             pk_filename=self.pk_filename,
58             msk_filename=self.msk_filename
59         )
60         if not (os.path.exists(self.pk_file_path) and os.path.exists(self.msk_file_path)):
61             error_msg = (f"Lỗi: Hàm setup không tạo ra các file khóa như mong đợi. "
62                         f"PK dự kiến: {self.pk_file_path}, MSK dự kiến: {self.msk_file_path}")
63             logger.error(error_msg)
64             return False, error_msg
65
66         msg = "Thiết lập hệ thống CP-ABE thành công."
67         logger.info(msg)
68         return True, msg
69     except Exception as e:
70         error_msg = f"Lỗi trong quá trình thiết lập hệ thống CP-ABE: {e}"
71         logger.exception(error_msg)
72         return False, error_msg
73
```

4.1.2. Thực thi f_cpabe_setup():

(*auth_center_project/cpabe_service_app/f_cpabe.py*)

- Bên trong hàm `f_cpabe_setup()`: **Gọi `waters11_scheme_instance.setup()`**: Đây là hàm cốt lõi của scheme Waters11 trong thư viện Charm-Crypto. Nó thực hiện các phép toán mật mã trên nhóm ghép cặp để tạo ra:
 - **public_key_dict**: Một dictionary chứa các thành phần của PK hệ thống
 - **master_key_dict**: Một dictionary chứa các thành phần của MSK.
 - **Serialize Khóa**: Các dictionary `public_key_dict` và `master_key_dict` (chứa các đối tượng phần tử nhóm của Charm) được serialize thành chuỗi bytes bằng hàm `objectToBytes()` của Charm-Crypto. Quá trình này chuyển đổi các cấu trúc dữ liệu phức tạp thành một định dạng nhị phân có thể lưu trữ được.
 - **Lưu Khóa vào File**: Các chuỗi bytes đã serialize của PK và MSK được ghi vào các file tương ứng (`public_key.bin`, `master_key.bin`) trong thư mục `output_directory_path` (chính là `self.keys_dir` được truyền vào).

```

11
12 def setup(actual_waters11_scheme_instance, output_directory_path, pk_filename="public_key.bin", msk_filename="master_key.bin"):
13     if not os.path.exists(output_directory_path):
14         os.makedirs(output_directory_path)
15     public_key_dict, master_key_dict = actual_waters11_scheme_instance.setup()
16     serialized_public_key = objectToBytes(public_key_dict, actual_waters11_scheme_instance.group)
17     serialized_master_key = objectToBytes(master_key_dict, actual_waters11_scheme_instance.group)
18     full_pk_path = os.path.join(output_directory_path, pk_filename)
19     full_msk_path = os.path.join(output_directory_path, msk_filename)
20     _save_bytes_to_file(serialized_public_key, full_pk_path)
21     _save_bytes_to_file(serialized_master_key, full_msk_path)
22     return full_pk_path, full_msk_path
23

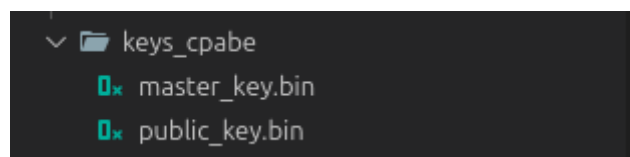
```

4.1.3. Kết quả của Quá trình Khởi tạo:

(*auth_center_project/keys_cpabe*)

Sau khi quá trình setup hoàn tất thành công, hệ thống sẽ có:

- Khóa Công Khai (PK - `public_key.bin`): Được lưu trữ trên server của Auth Center. PK này sẽ được cung cấp cho Resource Server (hoặc Client Application của Data Creator) để sử dụng trong quá trình mã hóa Khóa Mã Hóa Dữ Liệu (KEK) bằng CP-ABE.
- Khóa Bí Mật Chủ (MSK - `master_key.bin`): Được lưu trữ cực kỳ an toàn trên server của Auth Center. MSK này không bao giờ rời khỏi Auth Center và chỉ được sử dụng nội bộ bởi AC để tạo ra các Khóa Bí Mật (SK) cá nhân cho người dùng. Việc bảo vệ MSK là tối quan trọng cho an toàn của toàn bộ hệ thống CP-ABE.



4.2. Cấp phát access token gồm thuộc tính

Sau khi người dùng được xác thực thành công thông qua thông tin đăng nhập, Auth Center (AC) sẽ cấp phát JSON Web Tokens (JWT). Các token này không chỉ đóng vai trò là "vé thông hành" cho phép người dùng truy cập các API được bảo vệ trên cả AC và Resource Server (RS), mà còn mang theo các thuộc tính quan trọng của người dùng, phục vụ cho việc kiểm soát truy cập dựa trên thuộc tính (ABAC) ở phía RS.

4.2.1. Tùy chỉnh Payload của JWT để nhúng Thuộc tính Người dùng

(auth_center_project/cpabe_service_app/serializers.py)

```
7
8 class MyTokenObtainPairSerializer(TokenObtainPairSerializer):
9     @classmethod
10     def get_token(cls, user):
11         logger.info(f"MyTokenObtainPairSerializer.get_token called for user: {user.username}")
12         token = super().get_token(user) # Lấy token gốc từ simplejwt
13
14         # Thêm các claim tùy chỉnh vào payload của Access Token
15         try:
16             profile = UserProfile.objects.get(user=user)
17             # Sử dụng method có sẵn để lấy attributes
18             user_attributes = profile.get_attributes_string()
19
20             logger.info(f"User {user.username} attributes: {user_attributes}")
21
22             # Thêm attributes vào token
23             token['user_attributes'] = user_attributes
24
25             # Có thể thêm thêm thông tin khác
26             token['username'] = user.username
27             token['email'] = user.email
28
29             logger.info(f"Token claims added for user {user.username}")
30
31         except UserProfile.DoesNotExist:
32             # Xử lý trường hợp user không có profile
33             logger.warning(f"UserProfile not found for user: {user.username}")
34             token['user_attributes'] = ""
35             token['username'] = user.username
36             token['email'] = user.email
37
38         return token
39
40 class CustomTokenSerializer(TokenSerializer):
```

Nội dung token:

- **Header:** thuật toán ký là ES256 và loại token là JWT.
- **Payload:** Phần payload chứa các "claims" (thông tin):
 - **Claims tiêu chuẩn:** token_type ("access"), exp , iat , jti, user_id.
 - **Claims tùy chỉnh:** user_attributes: Chuỗi các ID thuộc tính CP-ABE của người dùng.

```

162 # Cấu hình djangorestframework-simplejwt
163 from datetime import timedelta
164 SIMPLE_JWT = {
165     'ACCESS_TOKEN_LIFETIME': timedelta(minutes=60),
166     'REFRESH_TOKEN_LIFETIME': timedelta(days=7),
167     'ROTATE_REFRESH_TOKENS': True,
168     'BLACKLIST_AFTER_ROTATION': True,
169     'UPDATE_LAST_LOGIN': True,
170
171     'ALGORITHM': 'ES256',
172     'SIGNING_KEY': JWT_EC_PRIVATE_KEY,
173     'VERIFYING_KEY': JWT_EC_PUBLIC_KEY,
174     'AUDIENCE': None,
175     'ISSUER': None,
176     'JWK_URL': None,
177     'LEEWAY': 0,
178
179     'AUTH_HEADER_TYPES': ('Bearer',),
180     'AUTH_HEADER_NAME': 'HTTP_AUTHORIZATION',
181     'USER_ID_FIELD': 'id',
182     'USER_ID_CLAIM': 'user_id',
183     'USER_AUTHENTICATION_RULE': 'rest_framework_simplejwt.authentication.default_user_authentication_rule',
184
185     'AUTH_TOKEN_CLASSES': ('rest_framework_simplejwt.tokens.AccessToken',),
186     'TOKEN_TYPE_CLAIM': 'token_type',
187     'TOKEN_USER_CLASS': 'rest_framework_simplejwt.models.TokenUser',
188
189     'JTI_CLAIM': 'jti',
190
191     'SLIDING_TOKEN_REFRESH_EXP_CLAIM': 'refresh_exp',
192     'SLIDING_TOKEN_LIFETIME': timedelta(minutes=60),
193     'SLIDING_TOKEN_REFRESH_LIFETIME': timedelta(days=1),
194
195     # Custom serializer để thêm attributes vào JWT
196     'TOKEN_OBTAIN_SERIALIZER': 'cpabe_service_app.serializers.MyTokenObtainPairSerializer',
197 }
198

```

Ký JWT bằng Thuật toán Bất đối xứng (ES256):

- Sau khi payload hoàn chỉnh (bao gồm cả claim tiêu chuẩn và tùy chỉnh) được tạo, djangorestframework-simplejwt sẽ tiến hành ký token.
- Hệ thống được cấu hình để sử dụng thuật toán ký bất đối xứng ES256 (ECDSA với SHA-256).
- Auth Center sử dụng Private Key ECC để tạo ra chữ ký số cho mỗi JWT. Public Key tương ứng sẽ được Resource Server sử dụng để xác minh chữ ký.
- **Signature:** Phần chữ ký được tạo bằng Private Key ES256 của Auth Center và được RS sử dụng để xác minh tính toàn vẹn, nguồn gốc của token.

4.2.2. Thực hiện đăng nhập qua API

Khi users thực hiện đăng nhập thông qua API `api/auth/login`, sẽ sử dụng custom token xây dựng ở trên để cấp phát và ký JWT token cho người dùng.

```

69
70 class CustomTokenObtainPairView(TokenObtainPairView):
71     """
72     Custom JWT token view để sử dụng serializer có attributes
73     """
74     serializer_class = MyTokenObtainPairSerializer

```

[illegible]

4.3. Tao và phân phối Public, secret key

(auth center project/cpabe service app/[views.py](#))

Một trong những vai trò trung tâm của AC là quản lý các khóa mật mã cho hệ thống CP-ABE. Điều này bao gồm việc tạo ra Khóa Công Khai của toàn hệ thống và cấp phát Khóa Bí Mật cá nhân cho từng người dùng dựa trên các thuộc tính của họ. Các khóa này là nền tảng cho việc mã hóa và giải mã dữ liệu EHR một cách an toàn và có kiểm soát

4.3.1. Cung cấp Khóa Công Khai Hệ thống (Public Key - PK)

Khóa Công Khai của hệ thống CP-ABE là thành phần cần thiết cho bất kỳ bên nào muốn mã hóa dữ liệu theo một chính sách thuộc tính. Data Creators sẽ cần PK này để thực hiện mã hóa phía client.

- **API Endpoint:** GET /api/cpabe/public-key/ trên Auth Center.
- **Logic Thực thi:** API này được xử lý bởi PublicKeyView (đặt tại `auth_center_project/cpabe_service_app/views.py`).
 - View này không yêu cầu xác thực (`permission_classes = [AllowAny]`), cho phép bất kỳ ai cũng có thể truy cập và tải về PK. Điều này là hợp lý vì PK, theo đúng bản chất của mật mã khóa công khai.
 - PublicKeyView sẽ gọi phương thức `get_public_key_content()` của CPABEHandler.
 - View trả về nội dung file này dưới dạng `HttpResponse` với `Content-Type: application/octet-stream` và header `Content-Disposition: attachment; filename="public_key.bin"`. Điều này khiến trình duyệt của người dùng tự động tải file PK về.


```

class PublicKeyView(APIView):
    permission_classes = [AllowAny] # Khóa công khai cho phép mọi người truy cập

    def get(self, request, *args, **kwargs):
        handler = CPABHandler()
        pk_content, error_msg = handler.get_public_key_content()

        if error_msg:
            return Response({"error": error_msg}, status=status.HTTP_404_NOT_FOUND)

        # Trả về file Khóa Công Khai
        response = HttpResponse(pk_content, content_type='application/octet-stream')
        # Lấy tên file từ settings để client có thể lưu đúng tên
        pk_filename = settings.CPABE_CONFIG['PUBLIC_KEY_FILENAME']
        response['Content-Disposition'] = f'attachment; filename="{pk_filename}"'
        return response

```

4.3.2. Tạo và Cấp phát Khóa Bí Mật Người Dùng (Secret Key - SK)

Khóa Bí Mật CP-ABE là khóa cá nhân của mỗi người dùng, cho phép họ giải mã các KEK (và do đó là dữ liệu EHR) mà thuộc tính của họ thỏa mãn chính sách CP-ABE tương ứng. Việc tạo và cấp phát SK phải được thực hiện một cách an toàn và chỉ cho những người dùng đã được xác thực.

- **API Endpoint:** POST /api/cpabe/generate-secret-key/ trên Auth Center.
- **Logic Thực thi:** API này được xử lý bởi GenerateSecretKeyView (đặt tại auth_center_project/cpabe_service_app/views.py).
 - **Xác thực:** View này yêu cầu người dùng phải được xác thực bằng JWT hợp lệ (permission_classes = [IsAuthenticated]). JWTAuthentication sẽ xác minh token và cung cấp thông tin request.user.
 - **Lấy Thuộc tính Người dùng:** View truy vấn UserProfile của request.user để lấy chuỗi các ID thuộc tính CP-ABE thông qua phương thức get_attributes_string().
 - **Gọi CPABHandler.generate_secret_key_content():** View truyền chuỗi thuộc tính này vào CPABHandler.
 - CPABHandler kiểm tra sự tồn tại của PK và MSK.
 - Nó gọi hàm f_cpabe.gen_secret_key() (từ f_cpabe.py), truyền vào instance của scheme Waters11, đường dẫn đến file PK và MSK, và chuỗi thuộc tính của người dùng.

```

22     return fute_pk_path, fute_msk_path
23
24
25 def gen_secret_key(actual_waters11_scheme_instance, public_key_file_path, master_key_file_path,
26                  user_attributes_string, output_sk_file_path):
27     (parameter) actual_waters11_scheme_instance: Any
28     pk_dict = bytesToObject(load_bytes_from_file(public_key_file_path), actual_waters11_scheme_instance.group)
29     msk_dict = bytesToObject(load_bytes_from_file(master_key_file_path), actual_waters11_scheme_instance.group)
30     attr_list = [attr.strip().upper() for attr in user_attributes_string.split(',') if attr.strip()] # Chuẩn hóa và upper
31     if not attr_list:
32         raise ValueError("Attribute list cannot be empty for key generation.")
33     user_secret_key_dict = actual_waters11_scheme_instance.keygen(pk_dict, msk_dict, attr_list)
34     serialized_user_secret_key = objectToBytes(user_secret_key_dict, actual_waters11_scheme_instance.group)
35     save_bytes_to_file(serialized_user_secret_key, output_sk_file_path)
36     return output_sk_file_path

```

- Hàm f_cpabe.gen_secret_key() sẽ:
 1. Deserialize PK và MSK từ file.
 2. Chuyển đổi chuỗi thuộc tính thành danh sách các ID thuộc tính.

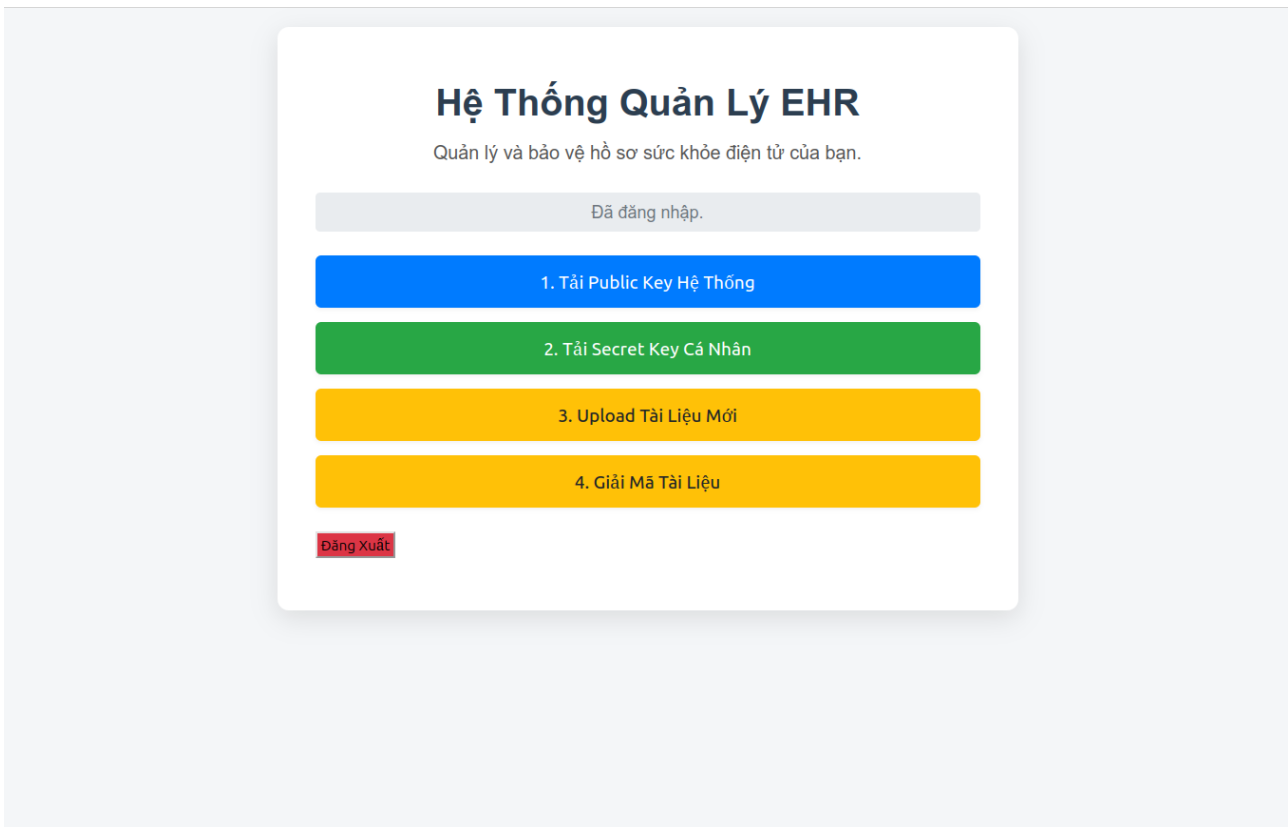
3. Gọi phương thức keygen() của scheme Waters11 (sử dụng PK, MSK và danh sách thuộc tính) để tạo ra đối tượng SK (là một dictionary chứa các thành phần khóa).
 4. Serialize toàn bộ đối tượng SK này thành chuỗi bytes bằng objectToBytes().
 5. Lưu chuỗi bytes SK vào một file tạm thời.
- CPABEHandler đọc nội dung từ file SK tạm thời và sau đó xóa file tạm đó.
 - **Trả về SK:** GenerateSecretKeyView trả về nội dung SK để người dùng có thể tải về và sử dụng để giải mã.

```

44 class GenerateSecretKeyView(APIView):
45     permission_classes = [IsAuthenticated]
46
47     def post(self, request, *args, **kwargs):
48         user = request.user
49         try:
50             user_profile = UserProfile.objects.get(user=user)
51         except UserProfile.DoesNotExist:
52             return Response({"error": "Không tìm thấy hồ sơ người dùng CP-ABE."}, status=status.HTTP_404_NOT_FOUND)
53
54         user_attributes_str = user_profile.get_attributes_string()
55
56         if not user_attributes_str:
57             return Response({"error": "Người dùng chưa được gán thuộc tính nào."}, status=status.HTTP_400_BAD_REQUEST)
58
59         print(f"Đang tạo Khóa Bí Mật cho người dùng '{user.username}' với thuộc tính: '{user_attributes_str}'")
60         handler = CPABEHandler()
61         sk_content, error_msg = handler.generate_secret_key_content(user_attributes_str)
62
63         if error_msg:
64             return Response({"error": error_msg}, status=status.HTTP_500_INTERNAL_SERVER_ERROR)
65
66         response = HttpResponse(sk_content, content_type='application/octet-stream')
67         response['Content-Disposition'] = 'attachment; filename="secret_key.bin"'
68         return response
69

```

Giao diện:



4.4. Mã hóa

4.4.1. Mã hóa ở phía Data Creators

(*main_server_project/resource_api_app/templates/upload_document.html*)

4.4.1.1. Nhập thông tin cơ bản

Nhập thông tin cơ bản về hồ sơ bệnh án này, các thông tin này chỉ có các thông tin không nhạy cảm của bệnh nhân (mã bệnh nhân, người khác sẽ không biết đây là ai), và mô tả chung chung cho mảnh hồ sơ này.

Nhập và Mã Hóa Dữ Liệu Y Tế

Thông Tin Cơ Bản (Không mã hóa)

Mã Bệnh Nhân:

P1234

Mô Tả Chung:

Đơn thuốc tái khám lần 4

Loại Dữ Liệu:

Kế hoạch điều trị

4.4.1.2. Tải lên dữ liệu cần mã hóa

Tải lên file là nội dung mảnh hồ sơ sức khỏe điện tử này.

Nội Dung Chi Tiết (Sẽ được mã hóa)

Chọn một trong hai cách nhập nội dung:

☐ Nhập văn bản trực tiếp ☒ Tải lên file

Chọn file để mã hóa:

đơn thuốc tái khám lần 4.pdf

Hỗ trợ: Text (.txt), Documents (.doc, .docx, .pdf), Images (.jpg, .png, .gif), Excel (.xlsx), PowerPoint (.pptx)

Kích thước tối đa: 10MB (5MB cho ảnh)

☒ File sẵn sàng: đơn thuốc tái khám lần 4.pdf

4.4.1.3. Định nghĩa chính sách truy cập

Data Creators sử dụng các checkbox để thực hiện việc định nghĩa chính sách truy cập. Các chính sách trong cùng một nhóm sẽ được ghép với nhau bởi phép OR. Và các nhóm khác

nhau được chọn ít nhất một ô sẽ được ghép với nhau bởi phép AND. Các chính sách truy cập này sẽ được serialize thành "policy" với các thuộc tính là integer để có thể thực hiện được việc mã hóa với scheme Waters11.

Định Nghĩa Chính Sách Truy Cập (CP-ABE)

Chọn các thuộc tính để định nghĩa ai có thể giải mã nội dung chi tiết ở trên.

Vai trò người truy cập (chọn một hoặc nhiều):

- ☒ DOCTOR
- ☒ NURSE
- ☐ SPECIALIST
- ☐ PHARMACIST
- ☒ LAB_TECHNICIAN
- ☐ RECEPTIONIST
- ☐ PATIENT
- ☐ RESEARCHER
- ☒ INSURANCE_STAFF

Chuyên khoa/Phòng ban người truy cập (chọn một hoặc nhiều):

- ☒ CARDIOLOGY

Chuyên khoa/Phòng ban người truy cập (chọn một hoặc nhiều):

- ☒ CARDIOLOGY
- ☐ ONCOLOGY
- ☐ PEDIATRICS
- ☐ SURGERY
- ☐ RADIOLOGY
- ☐ EMERGENCY
- ☐ INTERNAL_MEDICINE
- ☐ LABORATORY
- ☐ PHARMACY

Quyền truy cập loại thông tin (chọn một hoặc nhiều):

- ☐ MEDICAL_HISTORY
- ☐ LAB_RESULTS
- ☐ PRESCRIPTIONS
- ☐ IMAGING_REPORTS
- ☐ BILLING

Ngữ cảnh truy cập đặc biệt (chọn một hoặc nhiều):

- ☐ EMERGENCY_ACCESS
- ☐ RESEARCH_ANONYMIZED

Mối quan hệ với bệnh nhân (chọn một hoặc nhiều):

- ☐ TREATING_PHYSICIAN
- ☐ CONSULTING_PHYSICIAN

Mức độ truy cập của người xem (chọn một hoặc nhiều):

- ☐ SENSITIVE

Địa điểm làm việc của người xem (chọn một hoặc nhiều):

- ☒ HOSPITAL_A
- ☐ CLINIC_B

Điều kiện thời gian làm việc (chọn một hoặc nhiều):

- ☐ ON_DUTY

Chính sách sẽ được tạo:

Policy (readable):

(ROLE:DOCTOR or ROLE:NURSE or ROLE:LAB_TECHNICIAN or ROLE:INSURANCE_STAFF) and DEPARTMENT:CARDIOLOGY and ACTION:VIEW and LOCATION:HOSPITAL_A

Policy (IDs):

(1 or 2 or 5 or 10) and 11 and 26 and 35

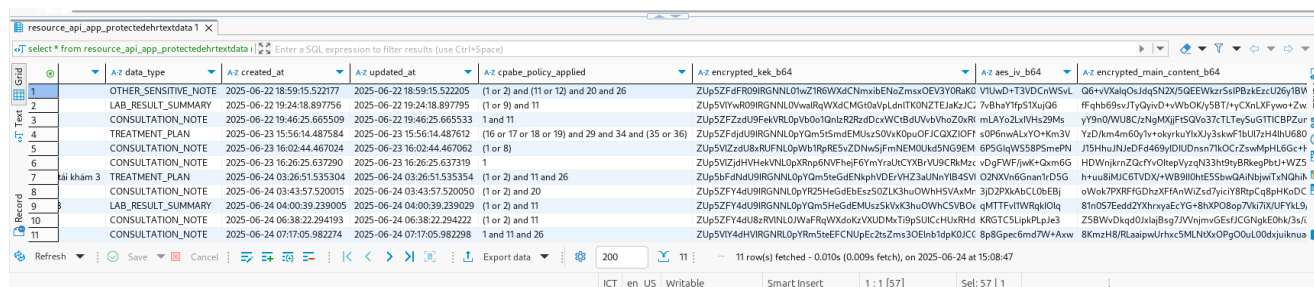
Khóa Công Khai CP-ABE

Tải lên file Public Key CP-ABE (.bin hoặc .json):

public_key.bin

Đã tải Public Key: public_key.bin

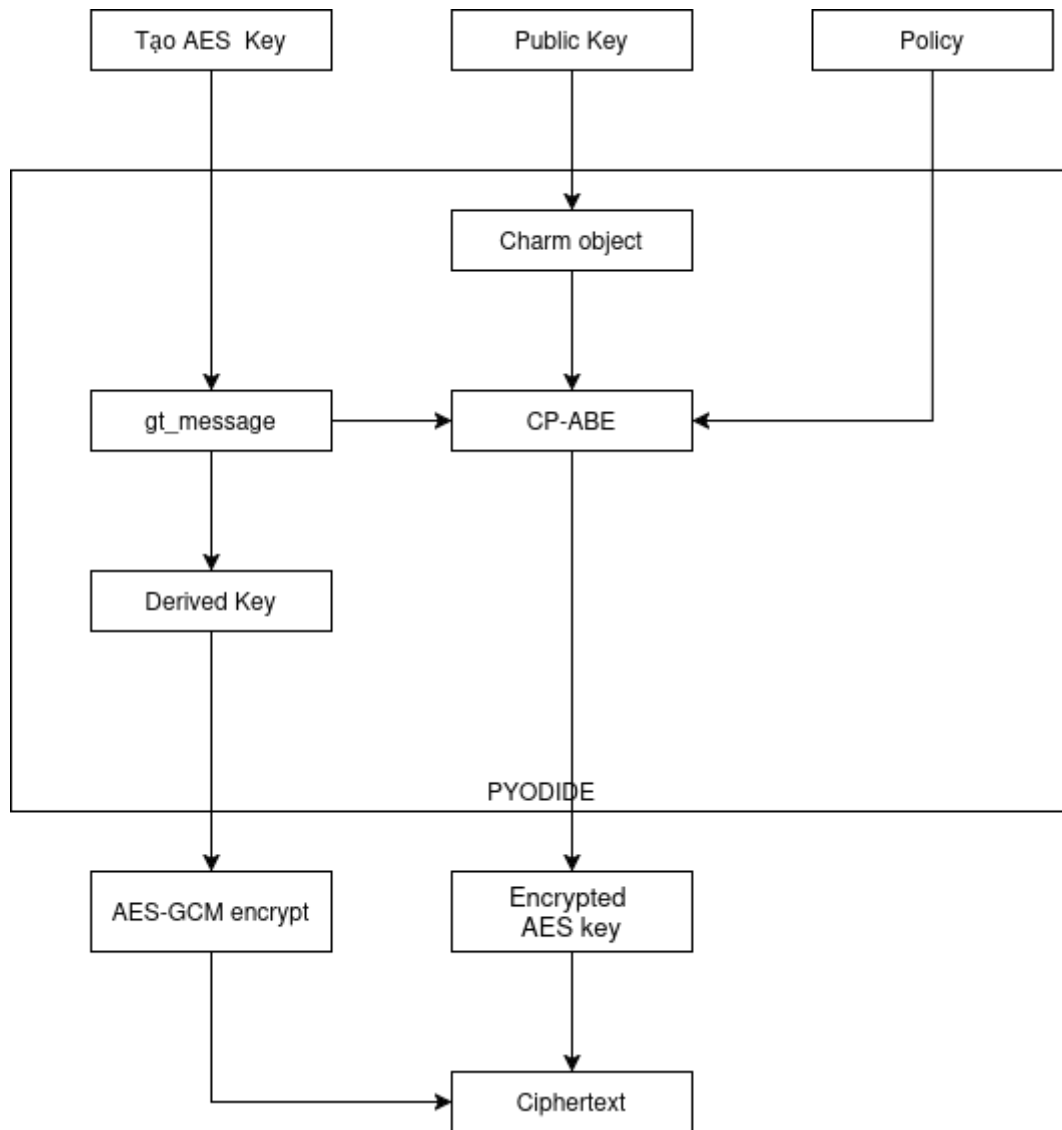
Các chính sách được chọn được hiển thị để có thể kiểm tra. Data Creators tiến hành tải lên public key để thực hiện việc mã hóa với mã hóa khóa lai(CP-ABE+ AES256-GCM) trên chính máy của Data Creator. Sau đó dữ liệu này được gửi đến controller xử lý việc upload dữ liệu lên CSDL, tiến hành lưu trữ dữ liệu đã mã hóa.



	Az data_type	Az created_at	Az updated_at	Az cpabe_policy_applied	Az encrypted_kek_b64	Az aes_iv_b64	Az encrypted_main_content_b64
1	OTHER_SENSITIVE_NOTE	2025-06-22 18:59:15.522177	2025-06-22 18:59:15.522205	(1 or 2) and (11 or 12) and 20 and 26	ZUp5ZF4FR09IRGNL0hwZ1R6Wx4CNmxbtENoZmsxOEY3Y0RaKz V1UwD+T3VDcNvSVL	Q6++VXalqOslJdySN2X/5GEEWkz/SaPBz4EzU26y1Bw	
2	LAB_RESULT_SUMMARY	2025-06-22 19:24:18.897756	2025-06-22 19:24:18.897795	(1 or 9) and 11	ZUp5VYVwR09IRGNL0VwalRwX4CMGtdAvLdnITKONZTEJaKzJC 7vBhaY1p5T1XujQ6	fFqb669svJTyQynD++VbOKy5BT1+yCnLXFywo+Zw	
3	CONSULTATION_NOTE	2025-06-22 19:46:25.665509	2025-06-22 19:46:25.665533	1 and 11	ZUp5ZFZzdU9FekVRL0pVb0nIQnLzR2zdDccvWCiBdVvVbVhoZ0zRI mLAYo2LxIVhs29Ms	yY9n0/WUC8/zNgMXfFISQV637CLTeySuGTTICBPZur	
4	TREATMENT_PLAN	2025-06-23 15:56:14.487584	2025-06-23 15:56:14.487612	(16 or 17 or 18 or 19) and 29 and 34 and (35 or 36)	ZUp5ZFZjdU9IRGNL0pYQm5SmdEMUzS0VvK0puOFJCQXZIOFT s0P6nwaLxYO+Km3V	YzD/km4m60y/++okykuYyXy3skwF1bU7zH4IHU680	
5	CONSULTATION_NOTE	2025-06-23 16:02:44.467024	2025-06-23 16:02:44.467062	(1 or 8)	ZUp5VIZzdU8xRUFNL0pWb1RgREsvZDNwSjFmNEMOUk45NG9EM 6PSGqW558P5mePN	J15HhuJNJeDF4469yDIUDnsn7K1OCrZswMphL6Gc+	
6	CONSULTATION_NOTE	2025-06-23 16:26:25.637290	2025-06-23 16:26:25.637319	1	ZUp5VIZjdHvHekVNL0pXRnp6NVFhejF6YmYraUICYXB+VU9CRkMzc vDgFWFjwK+Qum6G	HDWnjknZQcYvOtepVyzqN33h9yBRkegPbtJ+WZ5	
7	ai khám 3 TREATMENT_PLAN	2025-06-24 03:26:51.535304	2025-06-24 03:26:51.535354	(1 or 2) and 11 and 26	ZUp5BfdHdU9IRGNL0pYQm5SteGdENkphVDErVHZ3aUJnnYB45Vl OZNXVn6GnanTrDSG	h+u08MJC6TVDXH++WB9I0HtE55bwGAINbjwTxNQHk	
8	CONSULTATION_NOTE	2025-06-24 03:43:57.520050	2025-06-24 03:43:57.520050	(1 or 2) and 20	ZUp5ZF4dU9IRGNL0pYRZ5HegdEExz50ZUX3huOWNh5VhA+Mr 3jD29XkACU0eEBj	oWok7PWPFG0h2XFAnWzsd7YicY8RqCq8pHkoDc	
9	LAB_RESULT_SUMMARY	2025-06-24 04:00:39.239005	2025-06-24 04:00:39.239029	(1 or 2) and 11	ZUp5ZF4dU9IRGNL0pYQm5HegdEExz50ZUX3huOWNh5VhA+Mr 3jD29XkACU0eEBj	8In05TEddZ27YbryaEYg+8hXPO8op7A7XUFPfLb5	
10	CONSULTATION_NOTE	2025-06-24 06:38:22.294193	2025-06-24 06:38:22.294222	(1 or 2) and 11	ZUp5ZF4dU8xRVNL0pWb1RgREsvZDNwSjFmNEMOUk45NG9EM 6PSGqW558P5mePN	Z5BWwDkgd0JxlaBsg7V7VnjmVGEsfJJC6NgkE0HnJ3s/	
11	CONSULTATION_NOTE	2025-06-24 07:17:05.982274	2025-06-24 07:17:05.982298	1 and 11 and 26	ZUp5VY4dHVIRGNL0pYRm5SteEFCNUpEcZtsZms3OEbnIdpK0JCC 8p8Spec6md7W+Aw	8KmzH8RLaapwUrhx5cMLN8XoOpG0uL00dxjuikm	

4.4.2. Quy trình mã hóa tại phía Data Creators

(main_server_project/resource_api_app/templates/upload_document.html)



Bước 1: Khởi tạo môi trường mã hóa CP-ABE

Khi truy cập giao diện mã hóa và upload file, Pyodide sẽ tự động được tải, sau đó là charm wheel package để có thể thực thi code Python cho CP-ABE trên giao diện của người dùng. Sau đó bắt đầu khởi tạo các tham số cần thiết cho CP-ABE

```

855     const charmWheelURL =
856         "https://quackusarle.github.io/charm_crypto_wheel_for_pyodide/charm_crypto-0.50-cp312-cp312-pyodide_2024_0_wasm32.whl";
857     await micropip.install(charmWheelURL);
858
859     loadingStatus.textContent = "Đang khởi tạo hệ thống CP-ABE...";
860     // Initialize CP-ABE system như trong medical_upload.js
861     await pyodide.runPythonAsync(`
862
863     from charm.toolbox.pairinggroup import PairingGroup
864     from charm.schemes.abenc.waters11 import Waters11
865
866     group = PairingGroup('SS512')
867     waters_abe = Waters11(group, uni_size=100)
868
869     globals()["_waters11_abe_scheme"] = waters_abe
870     globals()["_waters11_group"] = group
871
872 `);
873
874     // Test initialization - DEBUG LINE 870
875     const testResult = await pyodide.runPythonAsync(`
876     result = "SUCCESS"
877     try:
878         _waters11_abe_scheme
879         _waters11_group
880     except NameError as e:
881         result = f"FAILED: {str(e)}"
882     except Exception as e:
883         result = f"ERROR: {str(e)}"
884     result
885 `);
886

```

Bước 2: Tạo khóa AES ngẫu nhiên

Sử dụng WebCrypto API để tạo khóa AES-GCM ngẫu nhiên cho mỗi lần mã hóa.

```

1097
1098     const aesKey = await crypto.subtle.generateKey(
1099         { name: "AES-GCM", length: 256 },
1100         true,
1101         ["encrypt", "decrypt"]
1102     );
1103     const aesKeyRaw = await crypto.subtle.exportKey("raw", aesKey);
1104
1105     // 2. Convert AES key to hex string like medical_upload.js
1106     const aesKeyHex = arrayBufferToHexString(aesKeyRaw);
1107
1108     // 3. Use the encryptAESKeyWithCPABE logic from medical_upload.js
1109
1110     // Pass data to Pyodide globals to avoid template literal issues
1111     pyodide.globals.set("js_aes_key_hex", aesKeyHex);
1112     pyodide.globals.set("js_policy_string", policyString);
1113

```

Bước 3: Biến đổi khóa AES thành GT message để mã hóa CP-ABE(trong Pyodide)

```

1147
1148     # Convert AES key hex to bytes
1149     aes_key_bytes = bytes.fromhex(js_aes_key_hex)
1150
1151     # Create a consistent seed from the AES key using SHA256
1152     key_hash = hashlib.sha256(aes_key_bytes).digest()
1153     seed = int.from_bytes(key_hash[:4], byteorder='big')
1154
1155     # Create GT message from the seed
1156     base_gt = group.pair_prod(pk['g1'], pk['g1'])
1157     gt_message = base_gt ** group.init(ZR, seed)
1158

```

Bước 4: Mã hóa GT message bằng CP-ABE(trong Pyodide)


```

1162
1163 # Use the policy string with integer IDs
1164 policy_str = js_policy_string
1165
1166 # Encrypt the GT message with CP-ABE
1167 print("Starting CP-ABE encryption...")
1168 ciphertext = waters_abe.encrypt(pk, gt_message, policy_str)
1169 print("✓ CP-ABE encryption completed")
1170 print(f"✓ Ciphertext keys: {list(ciphertext.keys())}")
1171
1172 # Serialize ciphertext components separately to avoid BinNode serialization issues
1173 from charm.core.engine.util import objectToBytes
1174
1175 # Create a serializable version of ciphertext without policy tree
1176 serializable_ct = {}
1177 for key, value in ciphertext.items():
1178     if key == 'policy':
1179         # Convert policy to string representation instead of BinNode
1180         serializable_ct[key] = str(value)
1181     else:
1182         # Keep other components as-is (they are serializable)
1183         serializable_ct[key] = value
1184

```

Bước 5: Mã hóa dữ liệu với khóa AES và chuyển thành Base64

```

1236
1237 // 4. Import derived key for data encryption
1238 const derivedKeyBytes = base64ToArrayBuffer(
1239   parsedResult.web_crypto_aes_key_base64
1240 );
1241 const derivedAesKey = await crypto.subtle.importKey(
1242   "raw",
1243   derivedKeyBytes,
1244   { name: "AES-GCM" },
1245   false,
1246   ["encrypt"]
1247 );
1248
1249 // 5. Encrypt content with derived key
1250 const iv = crypto.getRandomValues(new Uint8Array(12));
1251 const encodedContent = new TextEncoder().encode(content);
1252 const encryptedContentBuffer = await crypto.subtle.encrypt(
1253   { name: "AES-GCM", iv: iv },
1254   derivedAesKey,
1255   encodedContent
1256 );
1257
1258 return {
1259   encryptedKekBase64: parsedResult.abe_ciphertext_bundle_b64, // Đây là chuỗi base64 duy nhất
1260   ivBase64: arrayBufferToBase64(iv),
1261   encryptedDataAesBase64: arrayBufferToBase64(encryptedContentBuffer),
1262 };
1263 }
1264

```

Bước 6: Gửi dữ liệu đã mã hóa lên API upload

```

238
239 class UploadEHRTextView(APIView):
240     permission_classes = [IsAuthenticated]
241
242     def post(self, request):
243         user_id_from_token = request.user.id # Lấy từ JWT đã được xác thực
244
245         # Sử dụng serializer để validate và tạo đối tượng
246         # request.data sẽ là JSON payload từ client
247         serializer = ProtectedEHRTxtDataCreateSerializer(data=request.data)
248
249         if serializer.is_valid():
250             try:
251                 # Gán created_by_ac_user_id trước khi lưu
252                 ehr_entry = serializer.save(created_by_ac_user_id=user_id_from_token)
253
254                 # Trả về thông tin cơ bản của entry đã tạo (không bao gồm nội dung mã hóa)
255                 response_serializer = ProtectedEHRTxtDataResponseSerializer(ehr_entry)
256                 logger.info(f'User {user_id_from_token} đã upload thành công EHR text entry ID: {ehr_entry.id}')
257                 return Response({
258                     "message": "Dữ liệu văn bản đã được lưu trữ thành công.",
259                     "entry_id": ehr_entry.id,
260                     "data": response_serializer.data
261                 }, status=status.HTTP_201_CREATED)
262             except Exception as e:
263                 logger.error(f'Lỗi khi lưu ProtectedEHRTxtData cho user {user_id_from_token}: {e}')
264                 return Response({"error": "Lỗi phía server khi lưu trữ dữ liệu."}, status=status.HTTP_500_INTERNAL_SERVER_ERROR)
265         else:
266             logger.warning(f'Dữ liệu upload từ user {user_id_from_token} không hợp lệ: {serializer.errors}')
267             return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
268

```

4.5. Giải mã tại phía Data Users

4.5.1. Tìm các mảnh hồ sơ của bệnh nhân dựa vào patient_id

Nhập patient_id vào ô tìm kiếm để tìm tất cả các bản ghi thuộc về bệnh nhân này. Các thông tin hiển thị là tên bản ghi và mô tả cơ bản cho bản ghi đó.

Giải Mã Dữ Liệu Y Tế - Tìm Theo Bệnh Nhân

Tìm Kiếm Bản Ghi

Mã Bệnh Nhân:

Tìm Kiếm Bản Ghi

Xóa Kết Quả

Bản Ghi Của Bệnh Nhân: P1234 (11 bản ghi)

Hướng điều trị sau lần 4

14:17 24/06/2025

ID Bản Ghi:

44203250-cb27-4f29-97c9-8f5e32955901

Loại Dữ Liệu:

Ghi chú khám bệnh

Tạo Bởi User ID:

3

Cập Nhật Lần Cuối:

14:17 24/06/2025

 **Giải Mã Bản Ghi Này**

Tình trạng tái khám lần 4

13:38 24/06/2025

ID Bản Ghi:

e6ad09cc-6708-4589-89db-9d09d654d861

Loại Dữ Liệu:

Ghi chú khám bệnh

Tạo Bởi User ID:

Cập Nhật Lần Cuối:

[🔒 Giải Mã Bản Ghi Đây](#)**Tình trạng tái khám lần 4**

13:38 24/06/2025

ID Bản Ghi:

e6ad09cc-6708-4589-89db-9d09d654d861

Loại Dữ Liệu:

Ghi chú khám bệnh

Tạo Bởi User ID:

3

Cập Nhật Lần Cuối:

13:38 24/06/2025

[🔒 Giải Mã Bản Ghi Đây](#)**Xét nghiệm tái khám lần 3**

11:00 24/06/2025

ID Bản Ghi:

7ee6464d-ea41-4adc-8cf5-f9c16b92bf5c

Loại Dữ Liệu:

Tóm tắt KQ Xét nghiệm

Tạo Bởi User ID:

3

Cập Nhật Lần Cuối:

11:00 24/06/2025

[🔒 Giải Mã Bản Ghi Đây](#)**Ghi chú tái khám lần 3**

10:43 24/06/2025

ID Bản Ghi:

eb63f831-5893-40d4-bae0-8cc7ba0314fe

Loại Dữ Liệu:

Ghi chú khám bệnh

Tạo Bởi User ID:

3

Cập Nhật Lần Cuối:

10:43 24/06/2025

4.5.2. Truy cập vào một mảnh bản ghi sức khỏe

Khi truy cập vào một bản ghi nhất định, Data Users phải trải qua ba lớp kiểm tra truy cập:

- Kiểm tra xác thực người dùng với JWT.
- Kiểm tra quyền truy cập dựa trên thuộc tính với ABAC.
- Thử giải mã với Secret key gồm các thuộc tính của người đó.

4.5.2.1. Kiểm tra xác thực người dùng

Kiểm tra người dùng có token hợp lệ chưa với `permission_classes = [IsAuthenticated]` trước khi được thực hiện bất kỳ hành động nào

```

268 class RetrieveEHRTextView(APIView):
269
270     permission_classes = [IsAuthenticated, SatisfiesCPABEPolicyPermission]
271
272     def get_object(self, entry_id_uuid):
273
274         try:
275             return ProtectedEHRTTextData.objects.get(id=entry_id_uuid)
276         except ProtectedEHRTTextData.DoesNotExist:
277             raise Http404
278
279     def get(self, request, entry_id_uuid):
280         ehr_entry = self.get_object(entry_id_uuid)
281         self.check_object_permissions(request, ehr_entry)
282         data_to_return = {
283             'id': str(ehr_entry.id), # Chuyển UUID thành string
284             'patient_id_on_rs': ehr_entry.patient_id_on_rs,
285             'description': ehr_entry.description,
286             'data_type': ehr_entry.data_type,
287             'cpabe_policy_applied': ehr_entry.cpabe_policy_applied,
288             'encrypted_kek_b64': ehr_entry.encrypted_kek_b64,
289             'aes_iv_b64': ehr_entry.aes_iv_b64,
290             'encrypted_main_content_b64': ehr_entry.encrypted_main_content_b64,
291             'created_at': ehr_entry.created_at.isoformat() # Định dạng ISO cho datetime
292         }
293
294         logger.info(f"User {request.user.id} được phép truy cập ciphertext của EHR entry ID: {ehr_entry.id} "
295                   f"(Policy CP-ABE '{ehr_entry.cpabe_policy_applied}' đã được kiểm tra phía server)")
296
297         return Response(data_to_return)
298
299

```

4.5.2.2. Kiểm tra ABAC

Nếu người dùng có các thuộc tính phù hợp với chính sách áp dụng cho bản ghi đó, người dùng sẽ được phép lấy dữ liệu đã mã hóa về. Ngược lại, nếu thuộc tính trong JWT token của người dùng không thỏa mãn chính sách áp dụng với bản ghi đó, người dùng sẽ bị từ chối trước khi được lấy dữ liệu (dù đã mã hóa rồi).

Giải Mã Bản Ghi Y Tế

[← Quay Lại Danh Sách](#)
[🏠 Trang Chủ](#)

1. Thông Tin Bản Ghi

Thông Tin Bản Ghi

ID: 44203250-cb27-4f29-97c9-8f5e32955901

Mã Bệnh Nhân: P1234

Mô Tả: Hướng điều trị sau lần 4

Loại Dữ Liệu: Ghi chú khám bệnh

Ngày Tạo: 14:17:05 24/6/2025

2. Khóa Giải Mã

Tải lên file Public Key (.bin hoặc .json):

No file selected.

Tải lên file Private Key (.bin hoặc .json):

No file selected.



Không Có Quyền Truy Cập

Thuộc tính CP-ABE của bạn không thỏa mãn chính sách truy cập của dữ liệu này.

[← Quay Lại](#)[🏠 Trang Chủ](#)

Nếu bạn tin rằng bạn nên có quyền truy cập, vui lòng liên hệ quản trị viên hệ thống.

4.5.3. Tiến hành giải mã

4.5.3.1. Upload các public-key và secret-key

Data Users thực hiện việc upload public-key và private key cho phần giải mã. Scheme Waters 11 triển khai với charm cần cả public key cho phần giải mã.

2. Khóa Giải Mã

Tải lên file Public Key (.bin hoặc .json):

public_key.bin

Đã tải Public Key: public_key.bin

Tải lên file Private Key (.bin hoặc .json):

secret_key.bin

Đã tải Private Key: secret_key.bin

4.5.3.1. Tiến hành giải mã

Sau khi tải lên các file public_key và secret_key, người dùng tiến hành giải mã file đã mã hóa ngay trên máy của Data Users và tải file này về.

Browse... secret_key.bin

Đã tải Private Key: secret_key.bin

3. Giải Mã

Giải Mã Nội Dung

Nội dung đã giải mã:

🎉 File đã giải mã thành công! (Base64 decoded)

📁 **Loại file:** PDF DOCUMENT
📏 **Kích thước:** 79.17 KB
📄 **Mô tả gốc:** Đơn thuốc tái khám lần 4
📄 **Data Type:** TREATMENT_PLAN
🔗 **MIME Type:** application/pdf
🔄 **Encoding:** Base64 → Binary
🕒 **Tạo lúc:** 16:00:23 24/6/2025

📄 Tải về file đã giải mã

👁 Xem trước (nếu có thể)

Giải mã thành công! (Base64 decoded)

File PDF được mã hóa và tải về:

Jun 24 16:01

decrypted_document_1750755644628.pdf — Mozilla Firefox

file:///home/arie/Downloads/decrypted_document_1750755644628.pdf

Import bookmarks...

Automatic Zoom

decrypted_document_1750755644628.pdf Completed — 79.2 KB

decrypted_document_1750755237387.pdf Completed — 187 KB

public_key(6).bin Completed — 14.6 KB

decrypted_document_1750749444338.pdf Completed — 187 KB

decrypted_document_1750748776317.pdf Completed — 314 KB

Show all downloads

ĐƠN THUỐC TÁI KHÁM (Lần 4)

Chẩn Đoán Hiện Tại:

- Tăng huyết áp (I10)
- Rối loạn lipid máu (E78.5)
- Tình trạng sau đặt stent mạch vành (Z95.5) - ổn định

Diễn Biến Lâm Sàng Từ Lần Tái Khám Trước:

- Bệnh nhân tuân thủ điều trị tốt.
- Không có triệu chứng đau ngực hay khó thở bất thường.
- Huyết áp tại nhà dao động quanh mức 130/80 mmHg.
- Sinh hoạt bình thường, có đi bộ nhẹ nhàng hằng ngày.

Kết Quả Cận Lâm Sàng Mới (nếu có):
(Phần này có thể để trống hoặc ghi "Không có chỉ định mới trong lần khám này" hoặc "Chờ kết quả xét nghiệm máu định kỳ")

ĐƠN THUỐC:

1. Amlodipin 5mg
 - Uống: 1 viên/ngày (sau bữa ăn sáng)
 - Số lượng: 30 viên
2. Atorvastatin 20mg
 - Uống: 1 viên/ngày (buổi tối)
 - Số lượng: 30 viên
3. Clopidogrel 75mg
 - Uống: 1 viên/ngày (sau bữa ăn sáng)
 - Số lượng: 30 viên
4. Perindopril 5mg

4.5.4. Quy trình kiểm tra ABAC

(*main_server_project/resource_api_app/permissions.py*)

Bước 1: Kiểm tra các điều kiện cơ bản

Trước khi thực hiện các kiểm tra ABAC trên Data Users, cần thực hiện các kiểm tra cơ bản như người dùng đã xác thực chưa, token có hợp lệ không, và cần kiểm tra các thành phần CP-ABE đã có chưa để thực hiện việc xác thực với `MSP.prune()` ở sau.

```
34
35 class SatisfiesCPABEPolicyPermission(BasePermission):
36     """
37     Permission class để kiểm tra xem thuộc tính CP-ABE của người dùng
38     có thỏa mãn chính sách được lưu trữ cùng dữ liệu hay không
39     """
40     message = "Thuộc tính của bạn không thỏa mãn chính sách truy cập của dữ liệu này."
41
42     def has_object_permission(self, request, view, obj):
43         """
44         Kiểm tra quyền trên một đối tượng 'obj' cụ thể (là instance của ProtectedEHRTxtData).
45         'obj' phải có thuộc tính 'cpabe_policy_applied'.
46         """
47         # Bước 1: Kiểm tra các điều kiện cơ bản
48         if not request.user or not request.user.is_authenticated:
49             logger.warning(f"User không được authenticated trong SatisfiesCPABEPolicyPermission: user={request.user}")
50             return False
51
52         if not hasattr(request, 'auth') or not request.auth:
53             logger.warning(f"request.auth không tồn tại trong SatisfiesCPABEPolicyPermission: hasattr={hasattr(request, 'auth')}, auth={getattr(request, 'auth', 'None')}")
54             return False
55
56         if not CHARM_GROUP_FOR_POLICY_CHECK or not MSP_UTIL_FOR_POLICY_CHECK:
57             logger.error(f"Thành phần Charm-Crypto (Group/MSP) chưa được khởi tạo. Group={CHARM_GROUP_FOR_POLICY_CHECK}, MSP={MSP_UTIL_FOR_POLICY_CHECK}")
58             # Quyết định hành vi: an toàn nhất là từ chối
59             self.message = "Lỗi hệ thống: Không thể xác minh chính sách truy cập."
60             return False
61
```

Bước 2: Lấy thuộc tính của người dùng từ JWT token

Bước này thực hiện việc lấy attributes được lưu trong payload của token và thực hiện việc biến đổi thành list các string thuộc tính để `MSP.prune()` có thể xử lý.

```
61
62 # Bước 2: Lấy thuộc tính CP-ABE của người dùng từ JWT
63 token_payload = request.auth.payload
64
65 # Đọc từ user_attributes thay vì cpabe_ids (theo JWT payload thực tế)
66 user_cpabe_ids_str = token_payload.get('user_attributes', "") # Đọc từ user_attributes
67
68 user_attributes_list_for_charm = [] # Charm MSP thường làm việc với list các string thuộc tính
69 if user_cpabe_ids_str:
70     user_attributes_list_for_charm = [
71         attr.strip() for attr in user_cpabe_ids_str.split(',') if attr.strip()
72     ]
73
74 if not user_attributes_list_for_charm:
75     logger.warning(f"User {request.user.id} không có thuộc tính CP-ABE nào trong token.")
76     self.message = "Bạn không có thuộc tính CP-ABE nào để đối chiếu với chính sách."
77     return False
78
79 logger.debug(f"User {request.user.id} CP-ABE attributes for policy check: {user_attributes_list_for_charm}")
80
```

Bước 3: Lấy chuỗi policy được lưu trong dữ liệu của bản ghi

```
80
81 # Bước 3: Lấy chuỗi policy CP-ABE của đối tượng dữ liệu 'obj'
82 if not hasattr(obj, 'cpabe_policy_applied') or not obj.cpabe_policy_applied:
83     logger.warning(f"Resource (ID: {getattr(obj, 'id', 'N/A')}) không có 'cpabe_policy_applied'. Mặc định từ chối.")
84     self.message = "Dữ liệu không có chính sách truy cập CP-ABE được định nghĩa."
85     return False
86
87 resource_cpabe_policy_string = obj.cpabe_policy_applied
88 logger.debug(f"Checking policy for resource (ID: {getattr(obj, 'id', 'N/A')}): '{resource_cpabe_policy_string}' "
89             f"against user attributes {user_attributes_list_for_charm}")
90
```

Bước 4: Sử dụng `MSP.prune()` trong charm crypto để thực hiện việc đánh giá ABAC

Thực hiện việc parse chính sách áp dụng cho bản mã đó thành các cây thuộc tính, sau đó thực hiện việc so sánh với user_attributes được lấy từ bước 2 ở JWT. Trả về giá trị True nếu thuộc tính của người dùng đó thỏa mãn cây thuộc tính, hoặc ngược lại để đưa ra quyết định có cho phép truy cập hay không.

```
91 # Bước 4: Kiểm tra CP-ABE policy toàn diện sử dụng MSP evaluation
92 try:
93     logger.info(f"Starting comprehensive CP-ABE policy evaluation:")
94     logger.info(f" Resource policy: '{resource_cpabe_policy_string}'")
95     logger.info(f" User attributes: {user_attributes_list_for_charm}")
96
97     # 4.1. Parse policy string thành policy tree
98     policy_object = MSP_UTIL_FOR_POLICY_CHECK.createPolicy(resource_cpabe_policy_string)
99     logger.info(f" Policy object created: {type(policy_object)}")
100
101     # 4.2. Sử dụng MSP.prune() để kiểm tra toàn diện
102     # prune() trả về False nếu không satisfy, hoặc subset attributes nếu satisfy
103     satisfied_attributes = MSP_UTIL_FOR_POLICY_CHECK.prune(policy_object, user_attributes_list_for_charm)
104
105     logger.info(f" MSP.prune() result: {satisfied_attributes}")
106     logger.info(f" MSP.prune() type: {type(satisfied_attributes)}")
107
108     if satisfied_attributes is not False and satisfied_attributes is not None:
109         # Policy được thỏa mãn
110         logger.info(f"✅ Policy '{resource_cpabe_policy_string}' SATISFIED by user attributes "
111                    f"{user_attributes_list_for_charm}")
112         logger.info(f" Satisfying subset: {satisfied_attributes}")
113         return True
114     else:
115         # Policy không được thỏa mãn - tìm hiểu nguyên nhân
116         all_policy_attributes = MSP_UTIL_FOR_POLICY_CHECK.getAttributeList(policy_object)
117         user_attr_set = set(user_attributes_list_for_charm)
118         policy_attr_set = set(all_policy_attributes)
119
120         logger.warning(f"❌ Policy '{resource_cpabe_policy_string}' NOT SATISFIED")
121         logger.info(f" All policy attributes: {all_policy_attributes}")
122         logger.info(f" User has: {user_attr_set}")
123         logger.info(f" Policy requires: {policy_attr_set}")
124
125         missing_attr = policy_attr_set - user_attr_set
126         if missing_attr:
127             self.message = f"Thiếu thuộc tính: {' '.join(missing_attr)}. Policy cần: {resource_cpabe_policy_string}"
128         else:
129             self.message = f"Thuộc tính không thỏa mãn cấu trúc policy: {resource_cpabe_policy_string}"
130
131         return False
132 except Exception as e:
133     logger.error(f"Lỗi khi đánh giá policy CP-ABE '{resource_cpabe_policy_string}' "
134                f"cho resource (ID: {getattr(obj, 'id', 'N/A')}) với thuộc tính {user_attributes_list_for_charm}: {e}")
135     # Trong trường hợp lỗi phân tích policy hoặc lỗi không mong muốn, an toàn nhất là từ chối
136     self.message = "Lỗi hệ thống khi xác minh chính sách truy cập dữ liệu."
137     return False
138
```

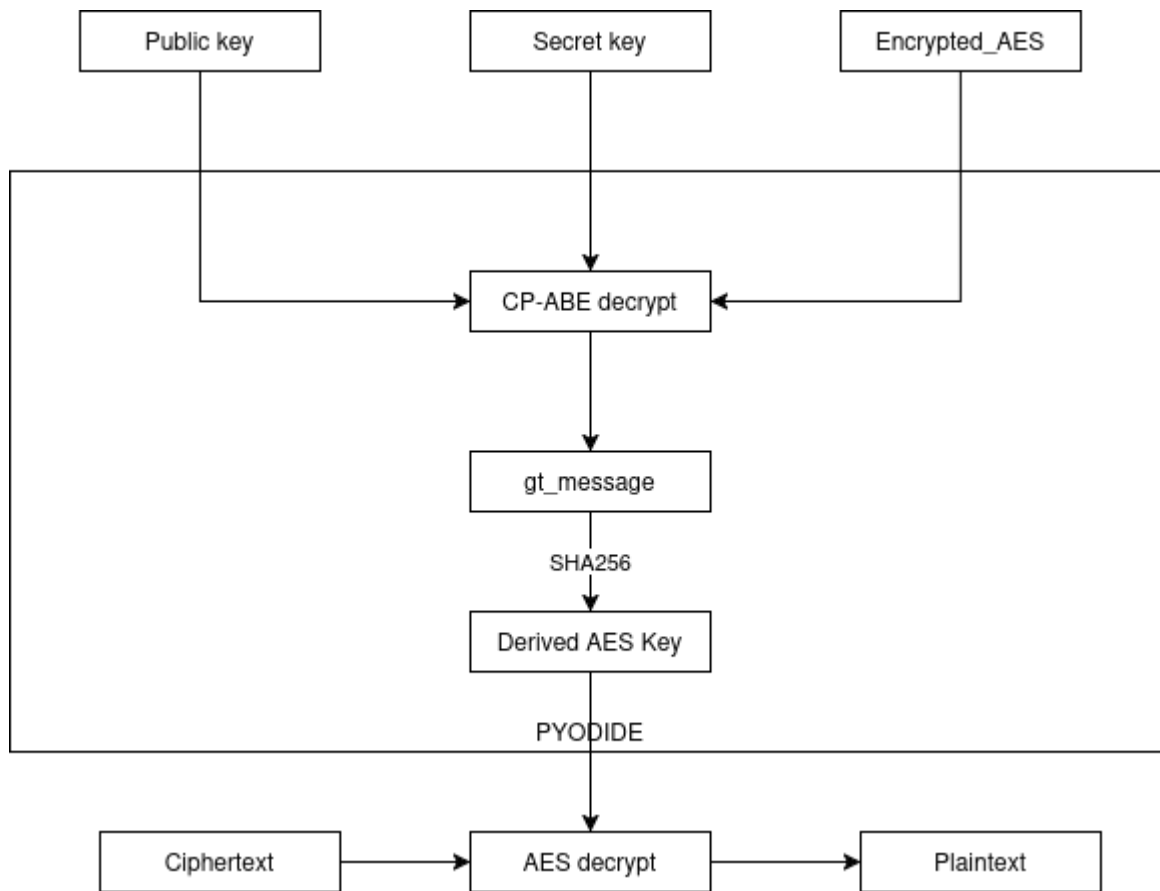
Bước 5: Quyết định cho phép truy cập

Dựa vào kết quả so sánh thuộc tính người dùng với cây chính sách, sẽ ra quyết định có cho phép người dùng có quyền truy cập API đó hay không.

```
107
108     if satisfied_attributes is not False and satisfied_attributes is not None:
109         # Policy được thỏa mãn
110         logger.info(f"✅ Policy '{resource_cpabe_policy_string}' SATISFIED by user attributes "
111                    f"{user_attributes_list_for_charm}")
112         logger.info(f" Satisfying subset: {satisfied_attributes}")
113         return True
114     else:
115         # Policy không được thỏa mãn - tìm hiểu nguyên nhân
116         all_policy_attributes = MSP_UTIL_FOR_POLICY_CHECK.getAttributeList(policy_object)
117         user_attr_set = set(us (variable) all_policy_attributes: list
118                             policy_attr_set = set(all_policy_attributes)
119
120         logger.warning(f"❌ Policy '{resource_cpabe_policy_string}' NOT SATISFIED")
121         logger.info(f" All policy attributes: {all_policy_attributes}")
122         logger.info(f" User has: {user_attr_set}")
123         logger.info(f" Policy requires: {policy_attr_set}")
124
125         missing_attr = policy_attr_set - user_attr_set
126         if missing_attr:
127             self.message = f"Thiếu thuộc tính: {' '.join(missing_attr)}. Policy cần: {resource_cpabe_policy_string}"
128         else:
129             self.message = f"Thuộc tính không thỏa mãn cấu trúc policy: {resource_cpabe_policy_string}"
130
131         return False
```


4.6. Quy trình giải mã dữ liệu tại phía Data Users

(*main_server_project/resource_api_app/templates/decrypt_record.html*)



Bước 1: Kiểm tra quyền truy cập với ABAC trước khi thực hiện hành động tiếp theo.

Bước 2: Khởi tạo Pyodide và charm

Khi truy cập vào trang giải mã, pyodide và charm-crypto được khởi tạo, sau đó là charm wheel package để có thể thực thi code python trên giao diện người dùng.

```
359 const micropip = pyodide.pyimport( micropip );
360
361 loadingStatus.textContent = "Đang cài đặt Charm-Crypto...";
362 const charmWheelURL = "https://quackusarle.github.io/charm_crypto_wheel_for_pyodide/charm_crypto-0.50-cp312-cp312-pyodide 2024.0.wasm32.whl";
363 await micropip.install(charmWheelURL);
364
365 loadingStatus.textContent = "Đang khởi tạo hệ thống CP-ABE...";
366 // Initialize CP-ABE system same as upload template
367 await pyodide.runPythonAsync(`
368 from charm.toolbox.pairinggroup import PairingGroup
369 from charm.schemes.abenc.waters11 import Waters11
370
371 group = PairingGroup('SS512')
372 waters_abe = Waters11(group, uni_size=100)
373
374 globals()['waters11_abe_scheme'] = waters_abe
375 globals()['waters11_group'] = group
376 `);
377
```

Bước 3: Load Public key và private key

Thực hiện việc load và xử lý các key thành các charm object để chuẩn bị cho giải mã.

```
481     }
482
483     // --- Public Key Handling (same pattern as upload) ---
484     publicKeyFileInput.addEventListener("change", async (event) => {
485         const file = event.target.files[0];
486         if (!file) {
487             publicKeyStatus.textContent = "Chưa chọn file.";
488             publicKeyObject = null;
489             updateDecryptButtonState();
490             return;
491         }
492     });
```

```
520
521     const result = await pyodide.runPythonAsync(`
522 import base64
523 import json
524 from charm.core.engine.util import bytesToObject
525
526 result_data = None
527 error_info = None
528
529 try:
530     pk_bytes_list = ${JSON.stringify(Array.from(pkBytes))}
531     pk_bytes = bytes(pk_bytes_list)
532
533     print(f"Received {len(pk_bytes)} bytes for public key deserialization")
534
535     if 'watersl1_group' not in globals():
536         raise Exception("Watersl1 group not initialized")
537
538     group = _watersl1_group
539     pk_obj = bytesToObject(pk_bytes, group)
540
541     globals()['_global_public_key_object'] = pk_obj
542
543     pk_info = {
544         "keys": list(pk_obj.keys()) if hasattr(pk_obj, 'keys') else [],
545         "status": "loaded",
546         "type": "charm_pk_object"
547     }
548
549     result_data = pk_info
550     print(f"Successfully loaded public key")
551 except Exception as e:
552     error_info = str(e)
553     print(f"Error deserializing public key: {e}")
554     import traceback
555     print(f"Traceback: {traceback.format_exc()}")
556
557 if result_data:
558     result = json.dumps(result_data)
559 else:
560     result = json.dumps({"error": error_info or "Unknown error"})
561
562 result
563 `);
564
565
```

```
589
590     // --- Private Key Handling (same pattern as upload) ---
591     privateKeyFileInput.addEventListener("change", async (event) => {
592         const file = event.target.files[0];
593         if (!file) {
594             privateKeyStatus.textContent = "Chưa chọn file.";
595             privateKeyObject = null;
596             updateDecryptButtonState();
597             return;
598         }
599     });
```

```

619         throw new Error('File JSON không hợp lệ: ${parseError.message}');
620     }
621 } else {
622     // Binary format - use same approach as upload template
623     const fileBuffer = await file.arrayBuffer();
624     const skBytes = new Uint8Array(fileBuffer);
625
626
627
628     const result = await pyodide.runPythonAsync`
629 import base64
630 import json
631 from charm.core.engine.util import bytesToObject
632
633 result_data = None
634 error_info = None
635
636 try:
637     sk_bytes_list = ${JSON.stringify(Array.from(skBytes))}
638     sk_bytes = bytes(sk_bytes_list)
639
640     print(f'Received {len(sk_bytes)} bytes for private key deserialization")
641
642     if '_watersll_group' not in globals():
643         raise Exception("Watersll group not initialized")
644
645     group = _watersll_group
646     sk_obj = bytesToObject(sk_bytes, group)
647
648     globals()['_global_private_key_object'] = sk_obj
649
650     sk_info = {
651         "keys": list(sk_obj.keys()) if hasattr(sk_obj, 'keys') else [],
652         "status": "loaded",
653         "type": "charm_sk_object"
654     }
655
656     result_data = sk_info
657     print(f'Successfully loaded private key")
658
659 except Exception as e:
660     error_info = str(e)
661     print(f'Error deserializing private key: {e}")
662     import traceback
663     print(f'Traceback: {traceback.format_exc}")
664
665 if result_data:
666     result = json.dumps(result_data)
667 else:
668     result = json.dumps({"error": error_info or "Unknown error"})
669

```

Bước 4: Giải mã khóa AES với CP-ABE

Thực hiện việc lấy khóa AES đã mã hóa truyền vào Pyodide, giải mã khóa này với các public-key và private-key charm object đã thực hiện ở bước trước. Sau bước giải mã ta được một gt_message. Sử dụng SHA256 để băm gt_message này thành khóa AES lại.

Cả mã hóa và giải mã đều là băm SHA256 gt_message trở thành khóa AES có 256 bit vì sau khi giải mã, Data User nhận được gt_message ban đầu dùng để băm ra khóa AES để mã hóa, băm gt_message này để nhận được cùng một giá trị với khóa dùng để mã hóa. Và gt_message không phải là 256 bit, việc này giải thích cho việc tại sao và tại sao lại cần thêm một bước băm này.

```

838 sk = _global_private_key_object
839
840 # Deserialize the ciphertext
841 encrypted_kek_b64 = "${currentRecord.encrypted_kek_b64}"
842 ciphertext_bytes = base64.b64decode(encrypted_kek_b64)
843 ciphertext = bytesToObject(ciphertext_bytes, group)
844
845 print(f"Ciphertext keys: {list(ciphertext.keys())}")
846
847 # Fix policy structure if needed
848 # During encryption, policy was converted to string to avoid BinNode serialization issues
849 # Now we need to convert it back to policy tree for decryption
850 if 'policy' in ciphertext and isinstance(ciphertext['policy'], str):
851     print("Converting string policy back to tree structure...")
852     policy_tree = waters_abe.util.createPolicy(ciphertext['policy'])
853     ciphertext['policy'] = policy_tree
854     print("✓ Policy tree reconstructed")
855 elif 'policy' not in ciphertext:
856     raise Exception("Ciphertext missing required 'policy' key")
857
858 # Validate private key structure
859 if not isinstance(sk, dict) or 'attr_list' not in sk:
860     raise Exception("Private key invalid or missing 'attr_list' field")
861
862 # Get public key and perform CP-ABE decryption
863 if '_global_public_key_object' not in globals():
864     raise Exception("Public key not loaded")
865
866 pk = _global_public_key_object
867 print("Performing CP-ABE decryption...")
868 gt_message = waters_abe.decrypt(pk, ciphertext, sk)
869 print("✓ CP-ABE decryption successful")
870
871 # Derive the AES key from GT message
872 web_crypto_key_bytes = hashlib.sha256(group.serialize(gt_message)).digest()
873 web_crypto_key_base64 = base64.b64encode(web_crypto_key_bytes).decode('utf-8')
874
875 final_result = {
876     "aes_key_base64": web_crypto_key_base64,
877     "status": "success"
878 }
879
880 print("✓ AES key derived successfully")
881
882 except Exception as e:
883     print(f"Decryption failed: {e}")
884     import traceback
885     print(f"Traceback: {traceback.format_exc()}")
886     final_result = {"error": str(e)}
887
888 json.dumps(final_result)
889 );
890

```

Bước 5: Sử dụng khóa AES nhận được để giải mã dữ liệu với WebCrypto API

```

889 );
890
891 const parsedResult = JSON.parse(result);
892
893 if (parsedResult.error) {
894     throw new Error(parsedResult.error);
895 }
896
897 // Now decrypt the content with AES
898 const aesKeyBytes = base64ToArrayBuffer(parsedResult.aes_key_base64);
899 const aesKey = await crypto.subtle.importKey(
900     "raw",
901     aesKeyBytes,
902     { name: "AES-GCM" },
903     false,
904     ["decrypt"]
905 );
906
907 const iv = base64ToArrayBuffer(currentRecord.aes_iv_b64);
908 const encryptedContent = base64ToArrayBuffer(currentRecord.encrypted_main_content_b64);
909
910 const decryptedBuffer = await crypto.subtle.decrypt(
911     { name: "AES-GCM", iv: iv },
912     aesKey,
913     encryptedContent
914 );
915

```

Chương 5: TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết quả Đạt được

Đồ án "Ứng dụng Ciphertext-Policy Attribute-Based Encryption thực thi kiểm soát truy cập cho Hồ sơ Sức khỏe Điện tử" đã thành công trong việc nghiên cứu, thiết kế và triển khai một mô hình hệ thống cơ bản nhằm giải quyết các thách thức về bảo mật và kiểm soát truy cập chi tiết cho dữ liệu y tế nhạy cảm. Các kết quả chính đã đạt được bao gồm:

Xây dựng Kiến trúc Hệ thống Phân tán:

- Đã thiết kế và hiện thực hóa một kiến trúc gồm hai thành phần server chính tách biệt: Auth Center và Resource Server, cùng với một Client Application cho phép mã hóa và giải mã trực tiếp trên thiết bị người dùng
- Auth Center: Chịu trách nhiệm quản lý định danh người dùng (đăng ký, đăng nhập), quản lý thuộc tính CP-ABE, khởi tạo hệ thống CP-ABE để tạo Khóa Công Khai và Khóa Bí Mật Chủ, cấp phát Khóa Bí Mật CP-ABE cá nhân cho người dùng, và cấp phát JSON Web Tokens (JWT) ký bằng thuật toán bất đối xứng ES256.
- Resource Server: Chịu trách nhiệm cung cấp API để người dùng (Data Creators) upload dữ liệu EHR (bao gồm metadata không mã hóa và nội dung chi tiết đã được mã hóa phía client), lưu trữ an toàn các bản ghi này (bao gồm ciphertext của KEK, ciphertext của nội dung, IV, và chính sách CP-ABE đã áp dụng). RS cũng cung cấp API để người dùng (Data Users) truy xuất ciphertext và thực thi kiểm soát truy cập API dựa trên thuộc tính (ABAC) thông qua việc xác minh JWT và các claim nhúng trong đó.
- Client Application: Triển khai logic mã hóa lai (AES-GCM cho nội dung, CP-ABE Waters11 cho khóa AES - KEK) và giải mã hoàn toàn ở phía client sử dụng JavaScript, thư viện Web Crypto API, và Charm-Crypto (thông qua Pyodide). Giao diện người dùng cho phép định nghĩa chính sách CP-ABE một cách linh hoạt dựa trên việc chọn lựa thuộc tính.

Triển khai Cơ chế Xác thực và Ủy quyền:

- Sử dụng JSON Web Tokens (JWT) ký bằng thuật toán bất đối xứng ES256 cho việc xác thực các yêu cầu API, đảm bảo tính toàn vẹn và xác thực nguồn gốc của token.
- Nhúng các thuộc tính CP-ABE và các thông tin định danh cơ bản vào payload của JWT, cho phép Resource Server thực hiện Attribute-Based Access Control (ABAC) ở tầng API để kiểm soát quyền truy cập vào ciphertext trước khi trả về cho client.

Áp dụng Thành công Mã hóa Lai CP-ABE và AES ở phía người dùng:

- Hệ thống CP-ABE Waters11 đã được khởi tạo, cho phép tạo PK, MSK và SK.
- Quy trình mã hóa lai đã được hiện thực hóa ở client: một khóa AES (KEK) ngẫu nhiên được tạo để mã hóa nội dung dữ liệu EHR, sau đó KEK này được mã hóa bằng CP-ABE với chính sách do người dùng định nghĩa.

- Quy trình giải mã tương ứng cũng được thực hiện ở client: người dùng sử dụng SK cá nhân để giải mã KEK, sau đó dùng KEK để giải mã nội dung dữ liệu.

Kiểm soát Truy cập Chi tiết:

- Hệ thống cho phép định nghĩa chính sách truy cập CP-ABE một cách linh hoạt cho từng "mảnh thông tin EHR", đảm bảo chỉ những người dùng sở hữu đúng tập thuộc tính mới có thể giải mã dữ liệu.
- Lớp ABAC trên Resource Server cung cấp một tầng kiểm soát bổ sung ở phía API trước khi người dùng được phép lấy dữ liệu dù đã mã hóa.

5.2. Hạn chế của Đề tài

Bên cạnh những kết quả đạt được, do giới hạn về thời gian và nguồn lực của một đề án môn học, đề tài vẫn còn một số hạn chế và các khía cạnh chưa được giải quyết triệt để:

1. **Thu hồi Thuộc tính và Khóa:** Hệ thống hiện tại chưa triển khai cơ chế thu hồi thuộc tính hoặc thu hồi Khóa Bí Mật CP-ABE một cách hiệu quả. Đây là một thách thức lớn trong các hệ thống ABE.
2. **Bảo vệ Khóa Bí Mật Chủ và Khóa Ký JWT:** Trong khuôn khổ đề án, MSK và private key ký JWT được lưu trữ dưới dạng file hoặc được giả định đọc từ biến môi trường. Trong thực tế, cần các giải pháp bảo mật cao hơn như HSM hoặc dịch vụ quản lý khóa chuyên dụng.
3. **Giao diện Người dùng (UI/UX):** Giao diện người dùng phía client được xây dựng ở mức độ cơ bản để minh họa luồng hoạt động.
4. **Hiệu suất với Pyodide/Charm-Crypto ở Client:** Việc khởi tạo và chạy các phép toán CP-ABE phức tạp trong trình duyệt bằng Pyodide có thể ảnh hưởng đến hiệu suất và trải nghiệm người dùng, đặc biệt trên các thiết bị có cấu hình thấp.
5. **Môi trường Triển khai và Khả năng Mở rộng:** Hệ thống mới chỉ được kiểm thử trên môi trường cục bộ, chưa đánh giá được hiệu năng và khả năng mở rộng khi triển khai trên quy mô lớn với nhiều người dùng đồng thời.
6. **Các Tính năng EHR Toàn diện:** Đề án tập trung vào khía cạnh bảo mật và kiểm soát truy cập, chưa bao gồm đầy đủ các tính năng của một hệ thống EHR thực thụ.

5.3. Hướng Phát triển trong Tương lai

Để hoàn thiện và nâng cao hệ thống, một số hướng phát triển tiềm năng trong tương lai bao gồm:

1. **Nghiên cứu và Triển khai Cơ chế Thu hồi:** Tích hợp các kỹ thuật thu hồi thuộc tính/ khóa hiệu quả cho CP-ABE, ví dụ như sử dụng các scheme ABE có hỗ trợ thu hồi trực tiếp, proxy re-encryption, hoặc các phương pháp dựa trên thời gian hiệu lực của khóa/ thuộc tính.
2. **Tăng cường Bảo mật cho Khóa Chủ:** Tích hợp với Hardware Security Modules (HSM) hoặc các dịch vụ quản lý khóa trên cloud (Azure Key Vault, AWS KMS) để bảo vệ MSK và private key ký JWT.
3. **Cải thiện Giao diện Người dùng và Trải nghiệm Người dùng (UI/UX):** Phát triển một giao diện trực quan và thân thiện hơn cho việc xây dựng chính sách CP-ABE, có thể bao gồm các công cụ kéo thả hoặc các mẫu policy nâng cao.

4. **Tối ưu hóa Hiệu suất Client-Side Encryption/Decryption:** Nghiên cứu các cách tối ưu hóa việc sử dụng Pyodide/Charm-Crypto, hoặc xem xét các thư viện WebAssembly khác cho CP-ABE.
5. **Xây dựng Bộ Đánh giá Policy CP-ABE Mạnh mẽ trên Resource Server:** Nếu vẫn muốn RS kiểm tra sự phù hợp của thuộc tính người dùng với policy dữ liệu trước khi trả ciphertext, cần phát triển hoặc tích hợp một bộ phân tích và đánh giá policy CP-ABE đầy đủ trên RS.
6. **Triển khai trên Nền tảng Cloud và Đánh giá Khả năng Mở rộng:** Di chuyển hệ thống lên một nền tảng cloud (AWS, Azure, GCP), sử dụng các dịch vụ được quản lý (CSDL, object storage) và đánh giá hiệu năng, khả năng chịu tải, cũng như chi phí vận hành.
7. **Tích hợp Multi-Factor Authentication (MFA):** Thêm MFA cho Auth Center để tăng cường bảo mật cho quá trình đăng nhập.
8. **Mở rộng Tính năng EHR:** Bổ sung các chức năng nghiệp vụ y tế khác để hệ thống trở nên hoàn chỉnh hơn.
9. **Audit Log Chi tiết và Giám sát An ninh:** Xây dựng hệ thống ghi log và giám sát toàn diện để phát hiện và ứng phó với các sự cố bảo mật.
10. **Nghiên cứu các Scheme ABE Nâng cao:** Khám phá các biến thể ABE khác như Key-Policy ABE (KP-ABE) hoặc các scheme có khả năng ẩn thuộc tính (attribute hiding) nếu phù hợp với các yêu cầu bảo mật cụ thể.

5.4. Kết luận

Đồ án đã thành công trong việc đặt nền móng cho một hệ thống EHR bảo mật sử dụng CP-ABE với kiến trúc phân tán và các cơ chế xác thực hiện đại. Những kết quả đạt được cho thấy tiềm năng của việc áp dụng các kỹ thuật mật mã tiên tiến để giải quyết bài toán kiểm soát truy cập dữ liệu y tế nhạy cảm. Mặc dù còn một số hạn chế, các hướng phát triển được đề xuất sẽ giúp hoàn thiện và nâng cao hệ thống, hướng tới một giải pháp toàn diện và thực tiễn hơn trong tương lai.

Chương 6: THAM KHẢO

1. [Charm: A Framework for Rapidly Prototyping Cryptosystems](#)
2. [Pyodide: A Cpython port to WebAssembly/Emscripten](#)
3. [Towards a cryptography encyclopedia: a survey on attribute-based encryption](#)
4. [Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization Waters 2011](#)
5. [Securing Healthcare Records in the Cloud Using Attribute-Based Encryption](#)
6. [A practical application of CP-ABE for mobile PHR system: a study on the user accountability](#)