ĐẠI HỌC BÁCH KHOA HÀ NỘI

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**BÁO CÁO PROJECT 3**

***Đề tài: Tìm hiểu về cơ chế đóng gói khóa trong mật mã hậu lượng tử***

**Sinh viên thực hiện**: *20200180 – Trần Bách Lưu Đức*

**GVHD:** ThS. Bùi Trọng Tùng

**Học kỳ: 20231**

**Hà Nội, tháng 01/2024**

**MỤC LỤC**

[LỜI NÓI ĐẦU 5](#_Toc156337249)

[Chương 1. Phân tích bài toán 6](#_Toc156337250)

[1.1. Mô tả bài toán 6](#_Toc156337251)

[1.2. Yêu cầu cần đạt được 6](#_Toc156337252)

[Chương 2. GIỚI THIỆU CÔNG NGHỆ/CƠ SỞ LÝ THUYẾT 7](#_Toc156337253)

[2.1. Lịch sử của mật mã học 7](#_Toc156337254)

[2.2. Hệ mật mã khóa công khai và những bài toán khó giải đối với máy tính cổ điển 7](#_Toc156337255)

[*2.2.1. Hệ mật mã khóa công khai* 7](#_Toc156337256)

[*2.2.2. Những bài toán khó giải đối với máy tính cổ điển* 7](#_Toc156337257)

[2.3. Điện toán lượng tử và mật mã hậu lượng tử 7](#_Toc156337258)

[*2.3.1. Điện toán lượng tử* 7](#_Toc156337259)

[*2.3.2. Mật mã hậu lượng tử* 7](#_Toc156337260)

[*2.3.3. Sự cần thiết của việc triển khai mật mã hậu lượng tử* 7](#_Toc156337261)

[2.4. Chuẩn hóa mật mã hậu lượng tử 7](#_Toc156337262)

[*2.4.1. Các họ thuật toán bị ảnh hưởng* 7](#_Toc156337263)

[*2.4.2. Chuẩn hóa mật mã hậu lượng tử* 7](#_Toc156337264)

[2.5. FrodoKEM 7](#_Toc156337265)

[*2.5.1. KEM* 7](#_Toc156337266)

[*2.5.2. FrodoKEM* 7](#_Toc156337267)

[2.6. Công nghệ sử dụng 7](#_Toc156337268)

[Chương 3. THIẾT KẾ GIẢI PHÁP 9](#_Toc156337269)

[3.1. Tìm hiểu về vật lý lượng tử, máy tính lượng tử 9](#_Toc156337270)

[3.2. Tìm hiểu về cơ chế đóng gói khóa 9](#_Toc156337271)

[3.3. Chạy thử FrodoKEM 9](#_Toc156337272)

[3.4. Xây dựng các module mã hóa, sinh HMAC 9](#_Toc156337273)

[3.5. Xây dựng một chương trình có logic hoàn chỉnh 9](#_Toc156337274)

[3.6. Xây dựng giao diện, tạo sản phẩm cuối cùng 10](#_Toc156337275)

[Chương 4. TRIỂN KHAI THỬ NGHIỆM VÀ ĐÁNH GIÁ 11](#_Toc156337276)

[4.1. Triển khai thử nghiệm 11](#_Toc156337277)

[4.2. Đánh giá 13](#_Toc156337278)

[KẾT LUẬN 14](#_Toc156337279)

[TÀI LIỆU THAM KHẢO 15](#_Toc156337280)

**DANH MỤC HÌNH VẼ**

[Hình 1. Mô hình mật mã khóa công khai 13](#_Toc156336265)

[Hình 2. Mô hình cơ chế đóng gói khóa 13](#_Toc156336266)

[Hình 3. Các thành phần của mô hình đóng gói khóa 14](#_Toc156336267)

[Hình 4. Chương trình đóng gói khóa sử dụng Java API 15](#_Toc156336268)

[Hình 5. Kết quả in ra của chương trình 15](#_Toc156336269)

[Hình 6. Giao diện XML của ứng dụng 20](#_Toc156336270)

[Hình 7. Giao diện chính thức 20](#_Toc156336271)

[Hình 8. Giao diện thành công 21](#_Toc156336272)

[Hình 9. Interal Storage của ứng dụng 21](#_Toc156336273)

[Hình 10. Nội dung file input.txt 22](#_Toc156336274)

[Hình 11. Nội dung file decrypted\_output.txt 22](#_Toc156336275)

[Hình 12. Nội dung file output\_encrypted.txt 22](#_Toc156336276)

[Hình 13. Các module nhỏ của chương trình 22](#_Toc156336277)

**DANH MỤC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **Từ viết tắt** | **Tiếng anh** | **Tiếng việt** |
| ETSI | European Telecommunications Standards Institute | Viện Tiêu chuẩn Viễn thông Châu âu |
| FIPS | Federal Information Processing Standard | Tiêu chuẩn xử lý thông tin liên bang |
| IND-CCA | Indistinguishablity under Chosen Ciphertext Attack | Tấn công chọn trước bản mã |
| IND-CPA | Indistinguishability under Chosen Plaintext Attack | Tấn công chọn trước bản rõ |
| KEM | Key Encapsulation Mechanism | Cơ chế đóng gói khóa |
| LWE | Learning With Errors | Học với lỗi |
| NIST | National Institute of Standards and Technology | Viện Tiêu chuẩn và Công nghệ Quốc gia |
| PKC | Public-Key Cryptosystem | Hệ mật mã khóa công khai |
| PQC | Post-Quantum Cryptography | Mật mã hậu lượng tử |
| SP | Special Publication | Ấn phẩm đặc biệt |

# LỜI NÓI ĐẦU

Trong những năm gần đây, đã có một số tiến bộ trong việc nghiên cứu và phát triển máy tính lượng tử. Thậm chí còn có những máy tính lượng tử quy mô nhỏ đã được công bố.

Các hệ mật mã khóa công khai đang được sử dụng ngày nay, sẽ bị phá vỡ bởi một thuật toán được phát triển bởi Shor (vào năm 1994) triển khai trên máy tính lượng tử.

Với khả năng ra đời của máy tính lượng tử sẽ xuất hiện hai xu hướng phát triển của khoa học mật mã, đó là: Mật mã lượng tử và Mật mã hậu lượng tử. [1]

Việc một máy tính lượng tử với quy mô đủ lớn sẽ ra đời khi nào, là một câu hỏi chưa ai có thể trả lời. Tuy nhiên, nếu điều đó thành sự thật, tất cả các thông tin được giữ bí mật hiện nay, đều sẽ trở nên công khai với kẻ tấn công. Do đó, bất kể chúng ta có thể ước tính chính xác thời điểm xuất hiện của kỷ nguyên điện toán lượng tử hay không, việc chuẩn bị trước cho một hệ thống bảo mật thông tin, kháng lại điện toán lượng tử là một điều vô cùng cần thiết.

Hiểu được điều đó, khi được tham gia vào lớp Project 3, do thầy Bùi Trọng Tùng – người có chuyên môn rất cao về đề tài an toàn thông tin, được thầy giới thiệu cho đề tài tìm hiểu về Mật mã hậu lượng tử, em đã rất vui mừng và hồi hộp muốn tìm hiểu về một chủ đề rất mới lạ với bản thân này.

Em sẽ phát triển đề tài theo hướng: xây dựng một ứng dụng Android mã hóa cơ bản, có sử dụng thuật toán kháng lại điện toán lượng tử.

Công nghệ sử dụng:

* Ứng dụng Android Studio, sử dụng ngôn ngữ Kotlin để xây dựng ứng dụng
* Các thư viện Bouncy Castle, javax.crypto, java.security cho mã hóa, giải mã

Tóm tắt các chương

* Chương 1: Phân tích bài toán
* Chương 2: Giới thiệu công nghệ sử dụng
* Chương 3: Thiết kế giải pháp
* Chương 4: Triển khai sản phẩm, đánh giá và kết luận
* Tài liệu tham khảo

# Phân tích bài toán

## 1.1. Mô tả bài toán

Bài toán đặt ra là xây dựng được các module mã hóa bản rõ, và các module giải mã ngược lại, sử dụng mật mã hậu lượng tử.

## 1.2. Yêu cầu cần đạt được

* Mã hóa được dữ liệu đầu vào của người dùng
* Giải mã ngược lại được dữ liệu đã mã hóa
* Có thêm cơ chế xác thực, chống sửa đổi

Tác nhân: Hệ thống có 1 tác nhân chính là người dùng.

Đặc tả use case:

|  |  |  |  |
| --- | --- | --- | --- |
| **Mã use case** | **UC001** | **Tên use case** | **Mã hóa** |
| **Tác nhân** | User | | |
| **Mục đích sử dụng** | Mã hóa dữ liệu đầu vào của User | | |
| **Sự kiện kích hoạt** | User chọn chức năng mã hóa | | |
| **Điều kiện tiên quyết** | User phải có được Public Key của đối phương | | |
| **Luồng sự kiện chính (Thành công)** | 1. User nhập một đoạn văn bản cần mã hóa 2. User chọn tham số cho mã hóa 3. User nhấn nút mã hóa 4. Hệ thống thông báo thành công | | |
| **Luồng sự kiện thay thế** | 4a. Hệ thống thông báo thất bại | | |
| **Hậu điều kiện** |  | | |

# GIỚI THIỆU CÔNG NGHỆ/CƠ SỞ LÝ THUYẾT

## 2.1. Lịch sử của mật mã học

Nhu cầu che giấu thông tin của con người đã có từ thời xa xưa, kể từ khi chữ viết được phát triển. Đã có bằng chứng cho thấy con người đã có hệ thống mật mã riêng của mình từ rất sớm, ví dụ như những bản khắc trên đá và giấy cói của nhiều nền văn minh cổ đại như người Ai Cập, Do Thái và Syria. Thuật ngữ “mã hóa” có nguồn gốc từ Hy Lạp, nó kết hợp giữa mật mã (cryptography) và phân tích mã(cryptanalysis), tức là nghệ thuật tạo ra các hệ thống mật mã và nghệ thuật phân tích các hệ thống mật mã và cố gắng phá vỡ chúng.

Về cơ bản, mật mã được chia làm hai loại: mật mã khóa đối xứng và mật mã khóa không đối xứng (mật mã khóa công khai).

Ý tưởng của mật mã khóa đối xứng, đó là A và B phải thống nhất trước với nhau, về một khóa bí mật dùng chung giữa hai người. Khóa này sau đó sẽ được A và B bảo vệ, và được dùng để mã hóa và giải mã bản tin của hai người. Hệ mật mã khóa đối xứng có tốc độ mã hóa và giải mã cao, tuy nhiên nó gặp phải vấn đề về việc trao đổi khóa. Nếu A và B ở xa nhau, việc thống nhất với nhau về một khóa bí mật dùng chung rất khó, do lúc này khóa đó cũng chính là một thông tin cần mã hóa để truyền đi xa.

Về sau, một hệ thống mật mã mới ra đợi, khắc phục được nhược điểm trên, đó là hệ thống mật mã khóa công khai.

## 2.2. Hệ mật mã khóa công khai và những bài toán khó giải đối với máy tính cổ điển

### *2.2.1. Hệ mật mã khóa công khai*

Với hệ mật khóa công khai, tồn tại một cặp khóa công khai và bí mật của người nhận. Người gửi sẽ dùng khóa công khai để mã hóa bản tin truyền đi, và chỉ có thể được giải mã bởi khóa bí mật của người nhận. Trong hệ thống mật mã khóa công khai, cần có một hàm dễ tính toán theo một chiều (bất kỳ ai cũng có thể mã hóa thông điệp) và khó có thể đảo ngược trừ khi chúng ta có thêm một thông tin được gọi là cửa sập (trapdoor). Do đó, các hàm này được gọi là các hàm một chiều cửa sập (trapdoor one-way functions). Ý tưởng sử dụng các loại hàm này được Diffie và Hellman đề xuất vào năm 1976. Và đến năm 1977, hệ thống khóa công khai đầu tiên được đề xuất bởi Rivest, Shamir và Adleman ra đời, với tên gọi là RSA – lấy chữ cái đầu tên của ba ông.

Hệ mật mã khóa công khai còn được ứng dụng trong việc tạo chữ ký số, khi mà một người có thể tạo chữ ký số từ khóa bí mật của mình, và mọi người có thể sử dụng khóa công khai để xác thực chữ ký đó.

### *2.2.2. Những bài toán khó giải đối với máy tính cổ điển*

Hệ mật mã khóa công khai hiện nay yêu cầu những bài toán khó giải đối với máy tính (hard computational problems). Nghĩa là khi kẻ tấn công có được khóa công khai, chúng không thể tính toán ra được khóa bí mật, hay nói chính xác hơn, mất rất lâu để tính toán. Bởi vì những bài toán khó giải đó, yêu cầu máy tính cổ điển thực hiện rất nhiều phép toán, mà trong một thời gian cụ thể, chúng không thể đưa ra được kết quả.

Tuy nhiên, vấn đề là chúng ta không thực sự biết được thế nào được gọi là “những bài toán khó giải đó”, (có thể P = NP). Tất cả chúng ta có thể làm đó là phỏng đoán (conjecture) và thừa nhận phỏng đoán đó là đúng, từ đó xây dựng nên những mô hình tính toán. [3]

Không có hệ thống mật mã nào được chứng minh là an toàn trước tất cả các cuộc tấn công, vì vậy mức độ an toàn của một lược đồ mật mã chỉ có thể được ước tính. Việc ước tính độ an toàn của một thuật toán mật mã không hề đơn giản. Một cách phổ biến để trình bày mức độ an toàn ước tính của một lược đồ mật mã là sử dụng khái niệm số bit an toàn (bit security). Một cuộc tấn công chống lại một lược đồ mật mã với mức an toàn *b*-bit ước tính có thể yêu cầu *O(2b)*  phép toán [1].

Đối với hầu hết hệ thống mật mã khóa công khai phổ biến, tính an toàn của chúng dựa vào các vấn đề toán học được xác định rõ ràng với “phỏng đoán” là khó giải. Sự an toàn của chúng phụ thuộc vào thực tế là không có lời giải hiệu quả nào tồn tại cho những vấn đề đó.

*“Cryptographers seldom sleep well” – Silvio Micali*

Phép đo an toàn bit thu được bằng cách xem xét tất cả các cuộc tấn công chung đã biết vào vấn đề toán học và lược đồ mật mã nói chung. Điều này đồng nghĩa với việc tồn tại các cuộc tấn công chưa xác định, làm cho mức độ an toàn đang được ước tính hiện tại không còn an toàn nữa. [1] Có một câu câu nói nổi tiếng của Silvio Micali – nhà khoa học máy tính người Ý: Nhà mật mã học hiếm khi ngủ ngon, bởi vì sự nghiệp của họ phụ thuộc vào những ước tính an toàn – điều mà có thể bị phá vỡ ngay khi họ thức dậy vào sáng ngày hôm sau. Tưởng tượng rằng, khi bạn thức dậy vào ngày hôm sau, cả thế giới đã thay đổi, và những phỏng đoán về độ an toàn mà bạn đã tin vào cả sự nghiệp mình, bỗng nhiên tan thành mây khói, chỉ bởi vì có một ai đó xuất hiện, với một thuật toán “thông minh hơn”, giải được vấn đề mà bạn coi là khó giải chỉ trong thời gian ngắn hơn nhiều.

Và sự thật đúng là có những thuật toán như vậy.

## 2.3. Điện toán lượng tử và mật mã hậu lượng tử

### *2.3.1. Điện toán lượng tử*

Máy tính chúng ta đang sử dụng rộng rãi ngày nay đôi khi được gọi là máy tính cổ điển (truyền thống). Hành vi của các phần tử trong các máy tính này có thể được mô tả bằng các định luật vật lý cổ điển, tức là những định luật được cho là chính xác vào khoảng đầu thế kỷ XX. Tuy nhiên, vào đầu thế kỷ XX, các nhà khoa học nhận ra rằng những định luật đó không mô tả chính xác hành vi của tất cả các hệ thống. Ví dụ, các vật thể ở quy mô nguyên tử hoạt động khác nhau trong các thí nghiệm so với dự đoán của vật lý cổ điển. Để mô tả chính xác hơn các hệ thống này, các nhà khoa học đã phát triển một lý thuyết vật lý mới gọi là vật lý lượng tử (quantum physic). Lý thuyết này bao gồm các yếu tố của thuyết phi tất định (non-determinism), và nó mô hình hóa chính xác hơn hành vi của tất cả các hệ thống. [1]

Với mô hình vật lý mới này, một loại máy tính mới đã xuất hiện: Máy tính lượng tử. Máy tính lượng tử là một thiết bị sử dụng các định luật vật lý lượng tử để giải các bài toán.

Trong một máy tính cổ điển, thông tin được lưu trữ và hoạt động dưới dạng “bit”. Mỗi bit được biểu diễn trong máy tính bởi một đối tượng tồn tại ở một trong hai trạng thái, thường được gọi là 0 và 1.

Một cách tổng quát, chúng ta có thể biểu diễn bit cổ điển thành vectơ như sau

Classical bits:

0 → **|**0⟩ =

1 → **|**1⟩ =

Trong một máy tính lượng tử, thông tin được lưu trữ và hoạt động dưới dạng các bit lượng tử, hay “qubit” (quantum bit). Bit trong máy tính cổ điển là một qubit “bị hạn chế”. Một qubit có thể tồn tại ở một trong nhiều trạng thái khác nhau. Có thể xem trạng thái của một qubit như một vectơ đơn vị trong một không gian vectơ phức hai chiều (two-dimensional complex vector space). Chúng ta chọn một cơ sở trực chuẩn thuận tiện (convenient orthonormal basis), các trạng thái cơ sở tính toán được ký hiệu là **|**0⟩, **|**1⟩, và biểu diễn qubit theo cơ sở đó.

Qubit: **|**Ψ⟩ = = α**|**0⟩ + β**|**1⟩ ∈ ℂ2

Trong đó α và β là biên độ (amplitude) của **|**0⟩ và **|**1⟩, và để qubit là một vectơ đơn vị (unit vector), thì:

|α|2 + |β|2 = 1

Và sự kết hợp tuyến tính của các trạng thái cơ bản như vậy thường được gọi là chồng chập (superposition). Do đó, trạng thái của một qubit là sự chồng chập trạng thái của hai qubit cơ bản là qubit **|**0⟩ (với β = 0) và qubit **|**1⟩ (với α = 0)

Chính nhờ sự chồng chập trạng thái này, máy tính lượng tử có thể giải hiệu quả các vấn đề toán học khó giải trên máy tính cổ điển.

### *2.3.2. Mật mã hậu lượng tử*

Như đã nói ở trên, tính an toàn của hầu hết các hệ mật mã khóa công khai (PKC) được sử dụng trong thực tế phụ thuộc vào cơ sở là việc tính thừa số nguyên tố (prime factorization) và tìm logarit rời rạc (discrete logarithm), là những bài toán khó giải đối với máy tính cổ điển. Năm 1994, Peter Shor đã tìm được một thuật toán thời gian đa thức (polynomial-time algorithm) mà giải hai bài toàn này sử dụng máy tính lượng tử. Các hệ mật mã khóa công khai có thể chống lại các cuộc tấn công này được gọi là các hệ mật mã kháng lượng tử (quantum resistant) hoặc hậu lượng tử (post-quantum). Hiện tại, chủ yếu có năm loại mật mã khóa công khai được cho là chống lại các cuộc tấn công cổ điển và lượng tử: [1]

* Mật mã dựa trên mã (code-based cryptography)
* Mật mã dựa trên hàm băm (hash-based cryptography)
* Mật mã dựa trên lưới (lattice-based cryptography)
* Mật mã khóa công khai đa biến (multivariate public-key cryptography)
* Mật mã dựa trên đường cong elliptic siêu kỳ dị (supersingular elliptic curve isogeny cryptography)

Mật mã hậu lượng tử, còn được gọi là mật mã an toàn lượng tử (quantum-safe) hoặc kháng lượng tử, là hệ thống mật mã được cho là an toàn trước tấn công của các thuật toán chạy trên máy tính lượng tử. Mật mã hậu lượng tử là mật mã cổ điển, nó không dùng bất kỳ tính chất lượng tử nào mà dựa trên các vấn đề toán học khó (hard mathematical problems), tương tự như hệ mật mã công khai mà chúng ta đang sử dụng hiện nay. Tuy nhiên, mật mã hậu lượng tử tránh sử dụng các bài toán phân tích thừa số nguyên tố và bài toán logarit rời rạc (integer factorizatoin and discrete log problems) để mã hóa dữ liệu. Bởi vì như đã nói ở trên, hai bài toán này tuy giải rất lâu ở máy tính cổ điển, nhưng được giải nhanh hơn rất nhiều trên máy tính lượng tử.

Tóm lại, nhiệm vụ của mật mã hậu lượng tử, đó là thiết kế ra một hệ mật mã, chạy trên máy tính cổ điển hiện thời, chống lại được tấn công lượng tử.

### *2.3.3. Sự cần thiết của việc triển khai mật mã hậu lượng tử*

Chưa thực sự có một hệ thống máy tính lượng tử đủ lớn được xây dựng để tấn công vào các hệ mật mã khóa công khai. Thậm chí là trong một vài năm nữa điều này vẫn chưa xảy ra. Trong khi đó, việc nghiên cứu và phát triển máy tính lượng tử vẫn có những tiến bộ nhất định. Thậm chí, đã có những máy tính lượng tử quy mô nhỏ (small-scale) được công bố. IBM đã cung cấp cho người dùng cơ hội trải nghiệm máy tính lượng tử quy mô nhỏ của họ với bộ xử lý 5 bit lượng tử trong dự án IBM Quantum Experience. Google thậm chí còn phát biểu rằng họ đang đi đúng hướng để đạt được mục tiêu về ưu thế lượng tử (quantum supremacy). [1]

Câu hỏi về việc máy tính lượng tử sẽ xuất hiện khi nào, hay thậm chí, liệu máy tính lượng tử có xuất hiện, là một câu hỏi chưa ai có thể trả lời thích đáng. Tuy nhiên, việc triển khai một hệ thống mật mã kháng lượng tử, triển khai trên máy tính truyền thống ngày nay là vô cùng cần thiết. Có những lý do xác đáng để chứng minh cho ý kiến này như sau: [3]

* Harvesting attacks: kẻ tấn công sẽ thu thập các bản mã ở hiện tại, kèm theo khóa công khai của chúng để có thể phá vỡ sau này.
* Thay đổi quá khứ: giả mạo chữ ký cho những khóa cũ. Kẻ tấn công lấy khóa công khai của người dùng để ký các giao dịch họ không thực hiện, để đến tương lai khi giải mã được khóa bí mật, sẽ giả mạo lại giao dịch đó với chữ ký từ khóa bí mật. Vấn đề này đặc biệt nghiêm trọng trong lĩnh vực Blockchain.

*“Who controls history controls the future” – George Orwell, 1984*

* Việc triển khai một hệ thống mật mã với quy mô lớn thường mất rất nhiều thời gian, thông thường phải lên đến hơn 10 năm. Trong lịch sử, đã mất gần hai thập kỷ để triển khai cơ sở hạ tầng mật mã khóa công khai hiện đại (modern public key cryptography infrastructure)

Trên thế giới, Chính phủ Mỹ lo lắng về khả năng về sự ra đời của máy tính lượng tử với quy mô lớn trong một thời gian. Cơ quan an ninh quốc gia Hoa Kỳ đã tuyên bố rằng:

*“IAD will initiate a transition to quantum resistant algorithms in the not too distant future… Our ultimate goal is to provide cost effective security against a potential quantum computer.” – NSA, 2015*

Viện Tiêu chuẩn và Công nghệ Quốc gia của Mỹ (NIST) coi việc này là mối đe dọa nghiêm trong cho hệ thống mật mã hiện thời. Viện Tiêu chuẩn Viễn thông Châu âu (ETSI) thậm chí còn thận trọng hơn khi khuyến nghị bất kỳ tổ chức nào có nhu cầu lưu trữ dữ liệu mã hóa lâu hơn năm 2025 nên lo lắng về sự ra đời của máy tính lượng tử. [1]

Để chống lại mối đe dọa này, các viện tiêu chuẩn hóa đã bắt đầu xem xét việc tiêu chuẩn hóa các thuật toán hậu lượng tử, tức là các thuật toán được cho là an toàn trước các cuộc tấn công từ máy tính lượng tử. Vào cuối năm 2016, NIST đã tổ chức một cuộc thi về Chuẩn hóa mật mã hậu lượng tử (Post-Quantum Cryptography Standardization – PQC NIST) để tìm ra các thuật toán mã hóa công khai an toàn trước sự tấn công của máy tính lượng tử. [1]

## 2.4. Chuẩn hóa mật mã hậu lượng tử

### *2.4.1. Các họ thuật toán bị ảnh hưởng*

[1] Có hai loại mật mã chính. Mật mã khóa đối xứng, như AES sẽ gặp rủi ro bởi thuật toán Grover, một thuật toán tìm kiếm lượng tử với tốc độ tăng tốc bậc hai so với các thuật toán tìm kiếm cổ điển. Tuy nhiên mối đe dọa này không quá nghiêm trọng bởi nó có thể dễ dàng bị đối phó bằng cách tăng gấp đôi chiều dài khóa. Ví dụ thay thế AES-128 thành AES-256 sẽ đảm bảo độ bảo mật 128-bit trước các tấn công lượng tử. Thuật toán tìm kiếm tương tự có thể được áp dụng cho hàm băm, vì vậy việc sử dụng các hàm băm với kích thước đầu ra lớn hơn là cần thiết để bảo vệ sự tấn công của điện toán lượng tử. Do đó mật mã khóa đối xứng có thể chống lại cuộc tấn công lượng tử, nhưng loại mật mã này có những hạn chế. Nó yêu cầu cả hai bên truyền thông chia sẻ một khóa bí mật dùng chung trước hoặc thực hiện một giao thức trao đổi khóa. Và đây không phải là ngữ cảnh người dùng cần kết nối an toàn từ trình duyệt của họ đến sàn thương mại điện tử.

Mật mã khóa công khai, chẳng hạn như RSA và ECC, được xây dựng dựa trên các bài toán “khó giải”. Đối với ngữ cảnh , máy tính thông thường cần hàng trăm hoặc hàng nghìn năm để giải những bài toán này, khiến chúng trở nên “không thể bị phá vỡ” một cách hiệu quả. Nhưng những điều này sẽ không đủ để bảo vệ dữ liệu và thiết bị trong thời đại lượng tử. Khi máy tính lượng tử được phát triển hoàn chỉnh, một máy tính sử dụng thuật toán Shor sẽ có khả năng bẻ khóa thuật toán RSA 2048-bit có lẽ chỉ trong vài ngày.

Các giao thức trao đổi khóa như DH (Diffie-Hellman) và ECDH (Elliptic Curve Diffie-Hellman) cũng bị phá vỡ bởi điện toán lượng tử, bởi vì chúng dựa trên cùng các bài toán logarit rời rạc.

### *2.4.2. Chuẩn hóa mật mã hậu lượng tử*

[1] Khi tất cả các đề xuất của cuộc thi mật mã hậu lượng tử NIST đều đã nộp vào ngày 30 tháng 11 năm 2017, quá trình tìm kiếm các ứng cử viên hàng đầu đã bắt đầu. Trong khi nhiều hệ mật mã hậu lượng tử đã được đề xuất, các hệ mật mã này cần phải trải qua nghiên cứu, thử nghiệm và đánh giá kỹ lưỡng để đảm bảo kết quả an toàn và tin cậy nhất.

Trước khi nhận được đệ trình, NIST đã chỉ rõ rằng họ muốn thay thế các thuật toán kháng lượng tử cho nhiều tiêu chuẩn đã được thiết lập trước đó của họ, chẳng hạn như các thuật toán thiết lập khóa (key establishment algorithms) và các lược đồ chữ ký số (digital signature schemes). Bất kỳ tiêu chuẩn mới tiềm năng nào được đồng ý sẽ được xuất bản dưới dạng các Ấn phẩm đặc biệt (Special Publications – SP) hoặc các Tiêu chuẩn xử lý thông tin liên bang (Federal Information Processing Standards – FIPS), tương ứng.

Tất cả các thuật toán được đệ trình đều được đánh giá bởi một hội đồng tuyển chọn gồm các nhân viên của NIST, cũng như các thành viên khác của cộng đồng khoa học. Bản thân NIST tuyên bố rằng: “Mặc dù NIST sẽ thực hiện các phân tích của riêng mình về các thuật toán đã gửi, nhưng NIST thực sự khuyến khích công chúng đánh giá và công bố kết quả. Cần lưu ý rằng mức độ phức tạp của tiêu chuẩn Mã hóa hậu lượng tử (PQC) thường được cho là phức tạp hơn đáng kể so với tiêu chuẩn hóa cho máy tính cổ điển, vì không chỉ là các yêu cầu phức tạp hơn nhiều, mà sự hiểu biết về máy tính lượng tử và khả năng tiềm ẩn của chúng vẫn rất khó nắm bắt.

Các tiêu chí đánh giá của cuộc thi:

+ *An toàn*

An toàn được phân loại theo các cấp độ khác nhau như sau:

* An toàn có thể so sánh với hoặc cao hơn so với một thuật toán mật mã khối có khóa 128 bit đối với tìm kiếm khóa theo phương án vét cạn (ví dụ: AES-128)
* An toàn có thể so sánh với hoặc cao hơn so với một hàm băm 256 bit đối với tìm kiếm xung đột (ví dụ SHA-256)
* An toàn có thể so sánh với hoặc cao hơn so với một thuật toán mật mã khối có khóa 192 bit đối với tìm kiếm khóa theo phương án vét cạn (ví dụ AES-192)
* An toàn có thể so sánh với hoặc cao hơn so với một hàm băm 384 bit đối với tìm kiếm xung đột (ví dụ: SHA-384)
* An toàn có thể so sánh với hoặc cao hơn so với một thuật toán mật mã khối có khóa 256 bit đối với tìm kiếm khóa theo phương án vét cạn (ví dụ: AES-256)

*+ Chi phí*

* Kích thước khóa công khai, bản mã và chữ ký
* Hiệu quả tính toán của thuật toán khóa công khai và khóa riêng
* Hiệu quả tính toán khi tạo khóa
* Giải mã không thành công

*+ Thuật toán và đặc điểm triển khai*

* Sự mềm dẻo
* Sự đơn giản
* Sự chấp thuận

Sau khi trải qua 3 vòng, cuộc thi đã chọn ra được 7 ứng cử viên lọt vào chung kết (“bộ thứ nhất”) và 8 ứng cử viên thay thế (“bộ thứ hai”). Ở project này, em sẽ tìm hiểu và sử dụng FrodoKEM, là ứng cử viên thay thế, trong “bộ thứ hai” của cuộc thi.

## 2.5. FrodoKEM

### *2.5.1. KEM*

[5] Như ta đã biết, đối với hệ mật mã khóa công khai, cần có một cặp khóa công khai-bí mật của người nhận, người gửi sẽ mã hóa thông điệp sử dụng khóa công khai, và chỉ có người nhận có thể giải mã được nó nhờ sử dụng khóa bí mật của mình. Dưới đây là mô hình của hệ mật mã khóa công khai.

A diagram of a key to an email

Description automatically generated

Hình 1. Mô hình mật mã khóa công khai

Tuy nhiên, hệ mật mã khóa công khai thông thường sẽ không hiệu quả bằng hệ mật mã khóa đối xứng. Do đó, người ta thường muốn sử dụng hệ mật mã khóa đối xứng hơn. Nhưng như đã đề cập ở trên, hệ mật mã khóa đối xứng gặp vấn đề về việc trao đổi khóa, khi hai bên truyền thông cần thống nhất với nhau về một khóa bí mật dùng chung (shared secret key) trước. Trong bối cảnh hai bên truyền thông ở xa nhau, để thống nhất được khóa bí mật, họ cần thực hiện một giao thức trao đổi khóa an toàn.

Cơ chế đóng gói khóa (Key Encapsulation Mechanism – KEM) cho phép hai bên truyền thông làm điều đó. KEM sẽ giúp hai bên truyền thông thống nhất được một khóa bí mật dùng chung với mô hình tương tự hệ mật mã khóa công khai, và dùng khóa đó để mã hóa đối xứng. Dưới đây là mô hình của KEM.

A diagram of a mail and key

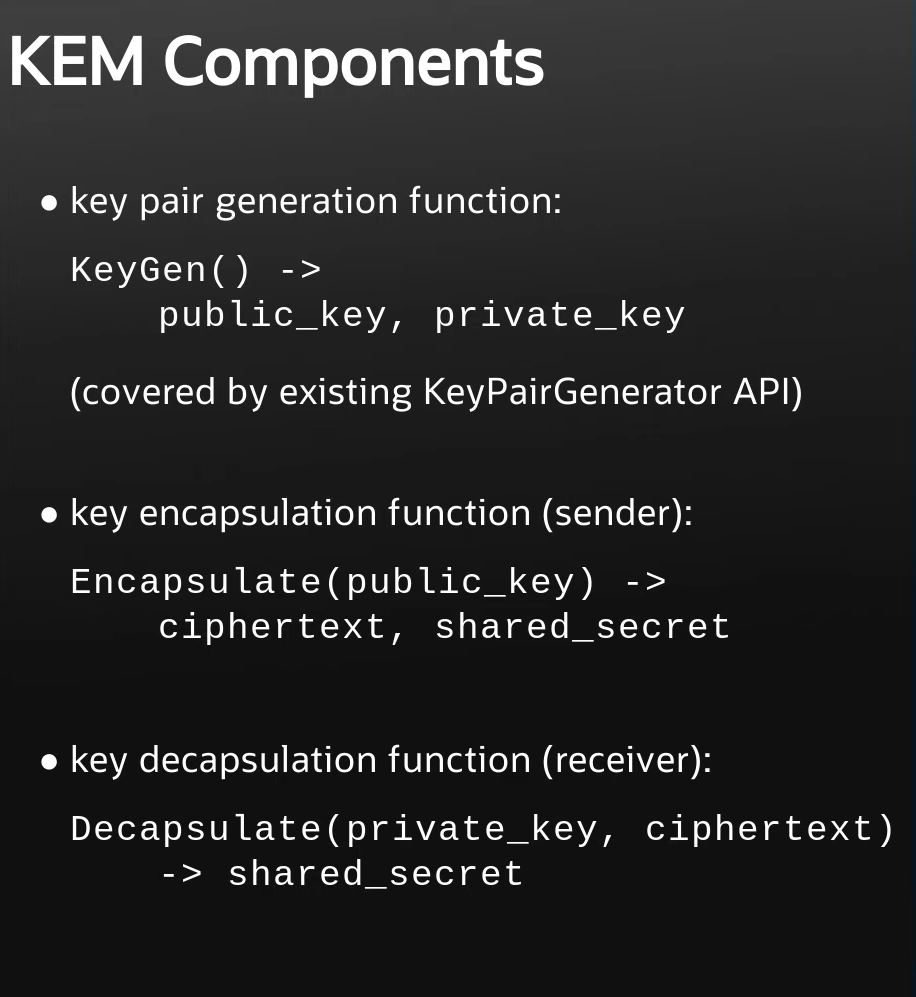
Description automatically generated

Hình 2. Mô hình cơ chế đóng gói khóa

KEM là một cơ chế sử dụng cặp khóa công khai-bí mật tương tự hệ mật mã khóa công khai. Người gửi – Sender sẽ dùng khóa công khai, để tạo ra một gói bọc khóa (encapsulation). Encapsulation là một bản mã, nó chứa một khóa đối xứng được chọn ngẫu nhiên. Người nhận Receiver, sẽ giải mã Encapsulation, quá trình này thường được gọi là giải đóng gói (Decapsulate), để lấy ra khóa bí mật dùng chung. Hai bên sau đó sẽ dùng khóa này để mã hóa thông điệp gửi đi với hệ mật mã đối xứng.

Không giống các cơ chế mã hóa truyền thống, cơ chế đóng gói khóa KEM không yêu cầu việc thêm phần đệm (padding) bởi vì nó sử dụng các đặc tính của khóa công khai để tạo ra một khóa đối xứng tương ứng. Chính vì vậy KEM là bước đầu được sử dụng để trao đổi khóa đối xứng, và sau đó sử dụng khóa này để mã hóa sử dụng các hàm mã hóa đối xứng thông thuộc.

KEM có ba thành phần chính như sau:



Hình 3. Các thành phần của mô hình đóng gói khóa

+ Hàm sinh khóa KeyGen sinh ra cặp khóa công khai-bí mật

+ Hàm bọc khóa Encapsulate nhận khóa công khai làm đầu vào, sinh ra một bản mã và khóa bí mật dùng chung

+ Hàm giải bọc khóa Decapsulate nhận khóa bí mật, bản mã của khóa làm đầu vào, thu được khóa bí mật dùng chung

Dưới đây là đoạn code mô tả cơ chế đóng gói khóa này, sử dụng các KEM API của Java.



Hình 4. Chương trình đóng gói khóa sử dụng Java API

Đoạn chương trình mô tả khái quát các bước của cơ chế đóng gói khóa, bắt đầu từ việc sinh ra cặp khóa, bên phía Sender sẽ bọc lại khóa, và phía Receiver sẽ giải bọc khóa. Hàm sendBytes được comment lại bởi nó minh họa việc Sender truyền gói bọc khóa đi cho Receiver

Và đây là kết quả in ra của chương trình.

A screen shot of a computer

Description automatically generated

Hình 5. Kết quả in ra của chương trình

Ngoài việc dùng hàm equals để kiểm tra, chương trình còn in trực tiếp ra khóa sinh từ phía Sender và khóa giải được từ phía Receiver để tăng tính trực quan, thuận tiện cho việc so sánh bằng mắt thường.

### *2.5.2. FrodoKEM*

#### 2.5.2.1. Giới thiệu qua về FrodoKEM

[6] FrodoKEM là hệ mật mã hậu lượng tử đã qua vòng tuyển chọn cuộc thi Chuẩn hóa mật mã hậu lượng tử (PQC NIST), và được chọn làm hệ mật mã thay thế, “bộ thứ hai” ở vòng thứ ba của cuộc thi.

FrodoKEM là họ cơ chế đóng gói khóa KEM, an toàn trước IND-CCA2 (tấn công chọn trước bản mật). Cần lưu ý rằng an toàn trước IND-CCA thì sẽ an toàn trước IND-CPA (tấn công chọn trước bản rõ), và điều ngược lại thì sẽ không đúng. IND-CCA yêu cầu kẻ tấn công phải có nhiều quyền hơn so với IND-CPA.

FrodoKEM là hệ mật mã khóa công khai, được thiết kế để trở nên kháng lượng tử. Sự bảo mật của FrodoKEM dựa trên các tham số của vấn đề học với lỗi (Learning with errors – LWE) đã được nghiên cứu một cách kỹ càng, có mối liên quan chặt chẽ với các vấn đề khó được giả định của bài toán về lưới không có cấu trúc về mặt đại số nói chung (“algebraically unstructured” lattices).

Khối xây dựng cơ bản (core building block) của FrodoKEM là một lược đồ mật mã khóa công khai, được gọi là FrodoPKE, an toàn trước IND-CPA, cũng liên quan chặt chẽ với độ khó của vấn đề học với lỗi (LWE) tương ứng.

Các tham số của FrodoKEM được sinh ra bởi bộ sinh số giả ngẫu nhiên PRG. Tùy vào mục đích, người dùng có thể chọn một trong hai biến thể (variant) AES hay SHAKE cho hàm PRG.

*So sánh biến thể AES-128 và SHAKE-128:*

+ Các biến thể AES (Advanced Encryption Standard) đặc biệt thích hợp cho các thiết bị có phần cứng hỗ trợ AES (như AES-NI trên các nền tảng của Intel). Nếu thiết bị có phần cứng này, việc triển khai sẽ có hiệu suất cao hơn vì AES có thể được thực hiện nhanh chóng bằng phần cứng

+ Các biến thể SHAKE thường cung cấp hiệu suất cạnh tranh hoặc tốt hơn so đối với các thiết bị mà phần cứng không hỗ trợ AES

* Chính vì vậy, việc lựa chọn AES hay SHAKE sẽ phụ thuộc vào ngữ cảnh triển khai cụ thể và khả năng hỗ trợ phần cứng của thiết bị.

Ngoài ra, FrodoKEM còn có hai biến thể chính về việc sử dụng lại khóa (key reuse), đó là biến thể tiêu chuẩn (standard variant) và (ephemeral variant) biến thể ngắn hạn.

*So sánh biến thể Standard và Ephemeral:*

+ Standard variant: không áp đặt bất kỳ một tiêu chuẩn nào cho việc sử dụng lại các cặp khóa

+ Ephemeral variant: sinh ra một cặp khóa mới với mỗi một lần khởi tạo giao thức. Biến thể này còn được gọi là eFrodoKEM

=> Về tổng quan, biến thể tiêu chuẩn sẽ thích hợp cho những ứng dụng mà một lượng lớn bản mã có thể được mã hóa thành một khóa công khai duy nhất. Trong khi đó, biến thể ngắn hạn sẽ phù hợp cho các ứng dụng mà số bản mã được tạo ra so với bất kỳ khóa công khai nào là khá nhỏ.

#### 2.5.2.2. Độ an toàn của FrodoKEM

* FrodoKEM-640 và eFrodoKEM-640: có độ an toàn lớn hơn hoặc bằng so với tấn công vét cạn vào hệ mật AES128
* FrodoKEM-976 và eFrodoKEM-976: có độ an toàn lớn hơn hoặc bằng so với tấn công vét cạn vào hệ mật AES192
* FrodoKEM-1344 và eFrodoKEM-1344: có độ an toàn lớn hơn hoặc bằng so với tấn công vét cạn vào hệ mật AES256

Cần lưu ý rằng, FrodoKEM có hai biến thể cho việc chọn hàm sinh số giả ngẫu nhiên (PRG) đó là AES và SHAKE, nên khi viết FrodoKEM-640, chúng ta có thể hiểu là nó bao gồm cả FrodoKEM-640-AES và FrodoKEM-640-SHAKE. Điều này tương tự cho FrodoKEM-976 và FrodoKEM-1344.

Như đã đề cập ở trên, độ an toàn của mô hình gói khóa FrodoKEM an toàn trước IND-CCA2.

INC-CCA2: cuộc tấn công chọn trước bản mã. Hệ mật mã an toàn trước cuộc tấn công này khi kẻ tấn công không thể có thêm tri thức gì từ việc phân biệt kết quả giải mã của một bản mã chọn trước, với một bản mã mục tiêu.

## 2.6. Công nghệ sử dụng

* Phần mềm Android Studio
* Ngôn ngữ Kotlin
* Các hàm mã hóa, giải mã, hàm sinh số ngẫu nhiên, hàm sinh mã MAC của thư viện Bouncy Castle, Java

# THIẾT KẾ GIẢI PHÁP

Mục tiêu của project là tìm hiểu về mật mã hậu lượng tử và ứng dụng nó. Vậy để xây dựng nên ứng dụng cuối cùng có sử dụng mật mã hậu lượng tử, em sẽ chia dự án thành từng mục tiêu nhỏ, hoàn thành từng mục tiêu nhỏ trước rồi dần dần xây dựng nên một sản phẩm hoàn chỉnh.

## 3.1. Tìm hiểu về vật lý lượng tử, máy tính lượng tử

Hai tuần đầu em học qua về vật lý lượng tử trên khóa học “Quantum Cryptography” trên trang Edx để hiểu qua thế nào là máy tính lượng tử, máy tính lượng tử khác gì so với máy tính cổ điển hiện thời và có những ưu điểm gì mà có thể phá vỡ các hệ mật mã khóa công khai hiện tại.

## 3.2. Tìm hiểu về cơ chế đóng gói khóa

Như hình 2.1, 2.2, 2.3, em đã xem video của Java để phần nào hiểu được thế nào là cơ chế đóng gói khóa, và chạy thử một chương trình có sử dụng các API của Java. [5]

## 3.3. Chạy thử FrodoKEM

Sau khi đã hiểu thế nào là cơ chế đóng gói khóa, em đã chạy thử chương trình sử dụng cơ chế đóng gói khóa của FrodoKEM như trên trang Docprimekey hướng dẫn. [6]

## 3.4. Xây dựng các module mã hóa, sinh HMAC

Việc chạy thử FrodoKEM đã thành công, điều này đồng nghĩa với việc em đã tạo ra được một khóa để sử dụng trong hệ mật mã khóa đối xứng. Việc tiếp theo là xây dựng các hàm mã hóa, giải mã, hàm sinh mã HMAC, giải mã HMAC.

## 3.5. Xây dựng một chương trình có logic hoàn chỉnh

Sau khi đã có từng module thành phần, bước tiếp theo đó là ghép chúng lại để tạo thành một logic thống nhất. Đó là lấy khóa sinh ra từ module FrodoKEM dùng cho các module mã hóa đối xứng. Đây là mô hình tổng quan cho sản phẩm cuối cùng của em:

Giả sử Sender muốn gửi một đoạn tin mã hóa cho Receiver. Các bước cần thực hiện:

*1. Mã hóa:*

Receiver sinh ra cặp khóa Public-Private\_key, chia sẻ Public\_key

* Sender lấy Public\_key làm đầu vào cho hàm đóng gói khóa, sinh ra một bản mật Encapsulation, chứa khóa bí mật dùng chung Shared\_secret
* Sender sau đó sẽ mã hóa bản rõ với hệ mật mã khóa đối xứng (SKC), sử dụng Shared\_secret sinh ra bản mật c
* Sender sinh ra mã xác thực thông điệp HMAC cho bản mật này
* Sender gửi Encapsulation, bản mật c, mã HMAC cho Receiver

*2. Giải mã:*

* Receiver giải mã Encapsulation, sử dụng Private\_key của mình để lấy Shared\_secret
* Receiver kiểm tra mã HMAC có hợp lệ hay không
* Receiver giải mã bản mật c, sử dụng Shared\_secret, thu được bản rõ m

## 3.6. Xây dựng giao diện, tạo sản phẩm cuối cùng

Cuối cùng, từ một chương trình hoàn chỉnh đó, ban đầu được viết bằng ngôn ngữ Java, em sẽ chuyển sang ngôn ngữ Kotlin tương ứng để tiện cho việc xây dựng ứng dụng Android. Ý tưởng của sản phẩm demo sẽ là mã hóa từ dữ liệu người dùng nhập vào, vậy giao diện của ứng dụng đơn giản sẽ chỉ cần một EditText để người dùng nhập nội dung, sau đó là một Button để khi người dùng ấn sẽ kích hoạt chương trình chính. Ngoài ra, còn cho phép người dùng chọn các tham số của FrodoKEM dưới dạng một dropdownlist, nên cần thêm một Spinner.

# TRIỂN KHAI THỬ NGHIỆM VÀ ĐÁNH GIÁ

## 4.1. Triển khai thử nghiệm

Dưới đây là giao diện thô của ứng dụng.

A screenshot of a computer

Description automatically generated

Hình 6. Giao diện XML của ứng dụng

Giao diện XML của ứng dụng như đã đề cập ở chương ba, chỉ đơn giản có ba phần EditText, Spinner và Button.

EditText sẽ nhận lấy dữ liệu đầu vào của người dùng. Spinner sẽ hiện ra các tùy chọn tham số của FrodoKEM. Và cuối cùng sau khi đã nhập vào EditText, chọn tham số trong Button, người dùng sẽ nhấn vào Button để có thể chạy chương trình mã hóa.

Đây là giao diện hoàn chỉnh của ứng dụng, và màn hình chạy thử thành công.

A screenshot of a cell phone

Description automatically generated

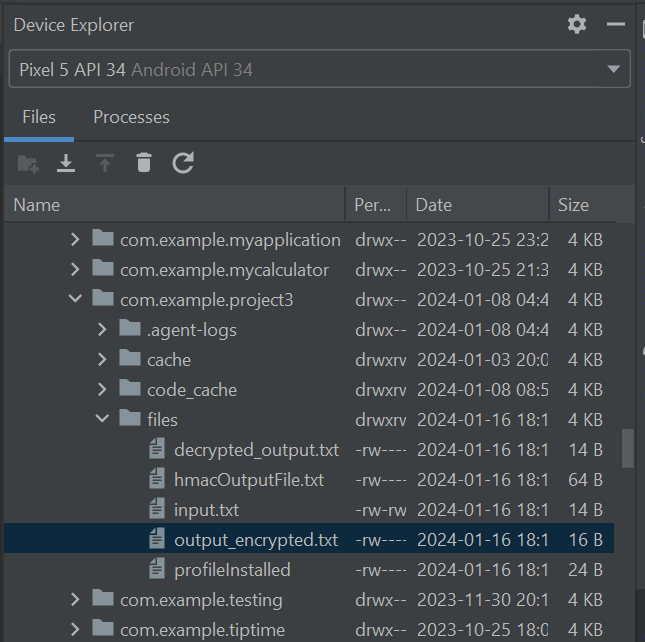
Hình 7. Giao diện chính thức

A screenshot of a phone

Description automatically generated

Hình 8. Giao diện thành công

Ứng dụng làm việc với Internal Storage. Chương trình sẽ lưu văn bản từ EditText vào một file input.txt, sau đó sẽ dùng file này làm đầu vào cho hàm mã hóa.



Hình 9. Interal Storage của ứng dụng

Do nguyên tắc của lập trình Android về quyền truy cập vào Internal Storage, đó là những file chỉ được truy cập bởi ứng dụng tạo ra nó. Nên ta sẽ không thể dùng phần mềm Files của điện thoại để đọc được các file xuất ra của ứng dụng. Vì vậy, chỉ có thể kiểm tra bằng cách xem trực tiếp file trong Device Explorer của Android Studio sử dụng quyền của nhà phát triển. Ở đây, các file được tạo ra bởi ứng dụng sẽ được lưu vào thư mục *“data/data/com.example.project3/files”*, project3 là tên dự án Android Studio mà em đặt ra từ ban đầu.

File input.txt sẽ chứa nội dung nhập vào của người dùng, sau đó bản mã in ra sẽ được ghi vào file output\_encrypted. File này sau đó sẽ được dùng để giải mã ngược để tìm ra bản rõ ban đầu, được ghi vào file decrypted\_output.txt.

A screen shot of a computer

Description automatically generated

Hình 10. Nội dung file input.txt

A screen shot of a computer

Description automatically generated

Hình 11. Nội dung file decrypted\_output.txt

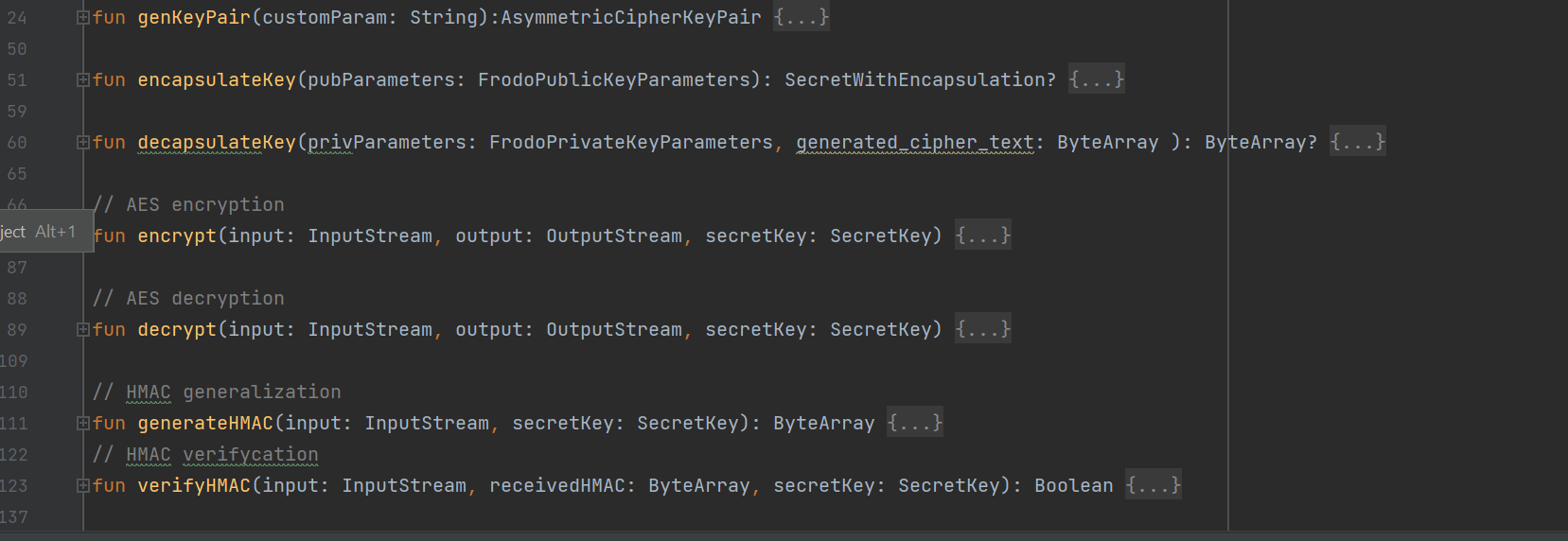
A screen shot of a computer

Description automatically generated

Hình 12. Nội dung file output\_encrypted.txt

Ta thấy nội dung của file input.txt và file decrypted\_output.txt có nội dung như nhau. Chứng tỏ chương trình đã mã hóa và giải mã ngược thành công.

Đây là các module nhỏ của chương trình.



Hình 13. Các module nhỏ của chương trình

Nội dung từng module:

* genKeyPair: sinh cặp khóa công khai-bí mật
* encapsulateKey: sinh ra khóa đối xứng, bọc khóa
* decapsulateKey: giải bọc khóa, lấy ra khóa đối xứng
* encrypt: mã hóa đối xứng AES
* decrypt: giải mã AES
* generateHMAC: tạo mã xác thực
* verifyHMAC: xác thực mã HMAC

## 4.2. Đánh giá

* Ứng dụng đã mã hóa được dữ liệu đầu vào của người dùng
* Đã giải mã ngược lại thành công, có thể kiểm tra được mã HMAC đúng hay sai

# KẾT LUẬN

Kết quả đã đạt được:

* Ứng dụng đã mô tả được phần nào hoạt động của Sender, các bước mà Sender cần làm. Tuy nhiên, do không có một Receiver thực sự, nên vẫn còn một số bước chưa đúng. Đó là việc khi ấn nút Encrypt, thay vì lấy Public\_key của Receiver, Sender phải tự sinh ra cặp khóa Private\_Public key để minh họa.
* Ứng dụng lưu kết quả mã hóa, mã HMAC, Encapsulation vào file txt ở trong Internal Storage. Và theo nguyên tắc của lập trình Android, các file trong Internal Storage chỉ được truy cập bởi ứng dụng tạo ra nó. Chính vì vậy các file này không thể được xem bởi các ứng dụng khác.

Định hướng phát triển đề tài:

* Xây dựng ứng dụng, có cả chức năng của Sender và Receiver, tức là người dùng vừa có thể đóng vai trò Receiver, giải mã dữ liệu nhận về, vừa đóng vai trò Sender, gửi dữ liệu mã hóa đi.
* Xây dựng module truy nhập file, chia sẻ file cho các ứng dụng khác.

# TÀI LIỆU THAM KHẢO

1. Sách Mật mã hậu lượng tử - Nhà xuất bản khoa học và kỹ thuật .
2. Khóa học về mật mã lượng tử trên trang edX.

Link: [CaltechX: Quantum Cryptography | edX](https://www.edx.org/learn/cryptography/caltech-quantum-cryptography)

1. Bài giảng về mật mã hậu lượng tử của Chris Peikert.

Link: [Post-Quantum Cryptography - Chris Peikert - 3/6/2022 (youtube.com)](https://www.youtube.com/watch?v=dbP2cgTsrRo&t=1677s)

1. Hướng dẫn dùng các thư viện mật mã hậu lượng tử của Bouncy Castle.

Link: [Introduction to Key Encapsulation Mechanisms (KEMs) (primekey.com)](https://doc.primekey.com/bouncycastle/post-quantum-hybrid-cryptography-in-bouncy-castle/introduction-to-key-encapsulation-mechanisms-kems)

1. Video giới thiệu về mô hình đóng gói khóa.

Link: [Strengthen your Java App's Defenses with Key Encapsulation Mechanism API - Inside Java Newscast #54 (youtube.com)](https://www.youtube.com/watch?v=zNnS-WdHN1E&t=106s)

1. Đề xuất tiêu chuẩn hóa sơ bộ của FrodoKEM

Link: [FrodoKEM-standard\_proposal-20230314.pdf](https://frodokem.org/files/FrodoKEM-standard_proposal-20230314.pdf)

1. Slide Nhập môn An toàn thông tin – ThS. Bùi Trọng Tùng.