

ĐẠI HỌC QUỐC GIA TP.HCM  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



MÔN HỌC: MẬT MÃ HỌC

MÃ MÔN HỌC: NT219.O11.ANTN

GIẢNG VIÊN: NGUYỄN NGỌC TỰ

---

**Authentication for Student Qualifications  
by Falcon Signatures Algorithm**

---

NGƯỜI THỰC HIỆN:

Nguyễn Thị Quỳnh Anh - 22520064

Nguyễn Thanh Bình - 22520136

# Contents

<b>1</b>	<b>Introduce</b>	<b>1</b>
1.1	Overview . . . . .	1
1.2	Problem statement . . . . .	1
1.2.1	Scenarios . . . . .	1
1.2.2	Related-Parites . . . . .	2
1.2.3	Security objective . . . . .	2
<b>2</b>	<b>Solution</b>	<b>2</b>
2.1	Post-quantum digital signature scheme . . . . .	2
2.1.1	FALCON: Fast-Fourier Lattice-based Compact Signatures over NTRU . . . . .	2
2.2	Reasons to choose Falcon Post-Quantum Signature Scheme . . . . .	4
2.3	Implementation of the Falcon Post-Quantum Signature Scheme . . . . .	5
<b>3</b>	<b>Implement</b>	<b>6</b>
3.1	Tools and resources . . . . .	6
3.2	Design and code . . . . .	6
3.2.1	System design . . . . .	6
3.2.2	Create Database . . . . .	7
3.2.3	Root CA . . . . .	9
3.2.4	Nhà phát hành . . . . .	12
3.2.5	Sinh viên . . . . .	17
3.2.6	Nhà tuyển dụng . . . . .	19
<b>4</b>	<b>Testing</b>	<b>23</b>
4.1	Nhà phát hành . . . . .	25
4.1.1	Tạo private key . . . . .	26

4.1.2	Kí và phát hành văn bằng . . . . .	28
4.2	Sinh viên . . . . .	31
4.3	Nhà tuyển dụng . . . . .	32
<b>5</b>	<b>Summary</b>	<b>34</b>
5.1	Results . . . . .	34
5.2	Tasks Chart . . . . .	35
5.3	Final Words . . . . .	35
<b>6</b>	<b>References</b>	<b>36</b>

# 1 Introduce

## 1.1 Overview

Văn bằng là các giấy tờ chứng nhận trình độ học vấn, năng lực chuyên môn hoặc kĩ năng của một cá nhân trong một lĩnh vực họ đã được đào tạo. Các văn bằng này có ý nghĩa quan trọng, nó là cơ sở để xét tuyển, thăng tiến, đào tạo và phát triển nghề nghiệp, được dùng để đi xin việc, nâng cao năng lực chuyên môn, hoặc tiếp tục học tập ở cấp độ cao hơn.

Do đó, việc xác thực độ tin cậy của văn bằng là vô cùng quan trọng. Điều đó đánh giá đúng đắn về năng lực chuyên môn nghiệp vụ mà một cá nhân đã được đào tạo.

Trong thời đại hiện nay, các văn bằng có thể ở dạng giấy hoặc thuận tiện hơn là dạng số. Vì vậy việc xác thực nội dung và nơi cấp của văn bằng cũng có thể được thực hiện từ xa và số hóa bằng chữ ký số.

Chữ ký số là một dạng chữ ký điện tử được tạo ra bằng sự biến đổi một thông điệp dữ liệu sử dụng hệ mã bất đối xứng, theo đó, người có được thông điệp dữ liệu ban đầu và khóa công khai của người ký có thể xác định được chính xác liệu việc ký có sử dụng đúng khóa bí mật (private key) tương ứng với khóa công khai (public key) trong cùng một cặp khóa hay không, từ đó xác thực được chủ nhân của chữ ký số (trong ngữ cảnh này là nơi cấp của văn bằng) và đảm bảo sự toàn vẹn nội dung dữ liệu.

## 1.2 Problem statement

Trong thời đại ngày nay, các văn bằng đa số được phát hành và sử dụng ở dạng giấy, muốn xác thực phải gửi đến nơi phát hành và xác thực tại nơi đó. Ngoài ra các văn bằng cũng có thể lưu trữ ở dạng số, do đó cần quản lý chứng chỉ số, một giải pháp thay thế để ký và xác thực các văn bằng nhanh hơn.

### 1.2.1 Scenarios

Trong hoàn cảnh đó, việc sử dụng một hệ thống ký và xác thực văn bằng số từ xa sẽ giúp ký và xác thực nhanh hơn. Từ đó giúp giảm thời gian di chuyển, vận chuyển mà thay vào đó là xác thực từ xa bằng hệ thống website. Các thể hệ của các hệ thống thuật toán chữ ký số hiện nay có thể được bảo mật trước các cuộc tấn công cổ điển, nhưng dễ bị tổn thương bởi các cuộc

tấn công sử dụng máy tính lượng tử. [1]

### 1.2.2 Related-Parites

- Nhà cung cấp dịch vụ: cung cấp hệ thống, chịu nhiệm vận hành, quản lý dữ liệu của các bên liên quan.
- Nơi phát hành văn bằng: có vai trò ký và phát hành văn bằng.
- Các đơn vị tuyển dụng: muốn xác thực nội dung và nơi cấp văn bằng.
- Sinh viên: có nhu cầu được cấp và sử dụng văn bằng vào mục đích cá nhân.

### 1.2.3 Security objective

- Tính xác thực: thông qua certificate có thể xác định được danh tính của nơi cấp.
- Tính bảo mật: sử dụng sơ đồ chữ ký số Falcon, nâng cao tính bảo mật trong thời đại hậu lượng tử.
- Tính toàn vẹn: đảm bảo nội dung văn bằng không bị giả mạo hoặc sửa đổi một cách trái phép.

## 2 Solution

### 2.1 Post-quantum digital signature scheme

Trong thời đại hậu lượng tử, việc chọn lựa phát triển sử dụng hệ thống chữ ký số mới như FALCON (Fast-Fourier Lattice-based Compact Signatures over NTRU) để tăng cường khả năng chống lại các cuộc tấn công bởi máy tính lượng tử trong cách thức ký và xác thực cũng như các hệ thống chữ ký số hiện nay.

#### 2.1.1 FALCON: Fast-Fourier Lattice-based Compact Signatures over NTRU

Thuật toán chữ ký Falcon là một thuật toán chữ ký số được tiêu chuẩn hóa bởi NIST, dựa trên lý thuyết mật mã học đại số, sử dụng các khái niệm như ma trận, đa thức và lưới. Nó có ưu điểm là tạo ra các chữ ký nhỏ gọn, có tính bảo mật cao và có thể chống lại các cuộc tấn công lượng tử. Chữ ký số này có thể được gửi qua mạng hoặc lưu trữ trên các thiết bị di động, và có thể được kiểm tra bởi các bên có quyền một cách nhanh chóng và chính xác.

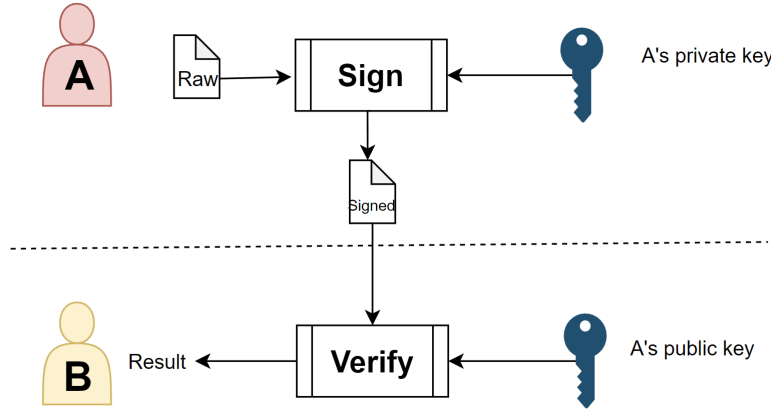


Figure 1: Sơ đồ ký và xác thực

Trong Falcon, GPV được sử dụng để xây dựng một hệ thống chữ ký dựa trên lattice, nó có thể được mô tả như sau:

- Public key là một ma trận có hạng đầy đủ  $A \in \mathbb{Z}_q^{n \times m}$ , với  $n < m$ , tạo thành một lattice  $A$ .
- Private key là một ma trận  $B \in \mathbb{Z}_q^{n \times m}$ , tạo thành một lưới trực giao  $A_q^\perp$ . Tại đó, lưới trực giao  $A_q^\perp$  là ký hiệu trực giao giữa module lattice  $A$  theo  $q$  và thỏa mãn điều kiện sau: với mọi  $x \in \mathbb{A}$  và  $y \in \mathbb{A}_\perp^\perp$  thì điều kiện  $\langle x, y \rangle \equiv 0 \pmod{q}$  được thỏa mãn. Tương đương các các hàng trực giao của  $A$  và  $B$  thỏa mãn:  $B \times A^t = 0$
- Đối với chữ ký số, thực hiện chữ ký số cho thông điệp  $m$ : dạng chữ ký là một giá trị số nguyên ngắn  $s \in \mathbb{Z}_q^m$ , sao cho  $sA^t = H(m)$ , trong đó  $H: \{0, 1\} \rightarrow \mathbb{Z}_q^m$  là một hàm băm.
- Đối với A, muốn xác thực chữ ký  $s$  sẽ được thực hiện chỉ bằng cách kiểm tra nếu  $s$  là một giá trị số nguyên ngắn thỏa mãn điều kiện  $sA^t = H(m)$ . [2]
- Việc xác thực chữ ký diễn ra phức tạp hơn: Ban đầu, người dùng phải tính giá trị tiền hình  $C_0 \in \mathbb{Z}_q^m$  thỏa mãn  $C_0A^t = H(m)$ . Sau đó,  $B$  được sử dụng để tính toán các vector đóng trực giao  $v \in A_q^\perp$  gần với

$C_0$ . Tính hợp lệ của chữ ký được xác định bởi  $s = c_0 - v$ . Khi giá trị  $c_0$  và  $v$  tiệm cận nhau thì:  $sA^t = c_0A^t - vA^t = c - 0 = H(m)$ . Kết quả nhận được là chữ ký  $s$  ngắn. Điều này cho thấy lợi thế của chữ ký số dựa trên thuật toán Falcon là tạo ra được chữ ký ngắn.

- Hệ chữ ký số Falcon sử dụng một lattice NTRU kết hợp với cấu trúc vòng, giúp giảm kích thước của Public key và độ phức tạp tính toán  $O(n)$  và tăng tốc độ xử lý của hệ thống bằng cách xử lý độ phức tạp giảm xuống  $O(n/\log n)$ . Về mặt lý thuyết, lattice NTRU đã được chứng minh là dạng lưới chuẩn nhỏ nhất, tức là tập hợp nhỏ nhưng có nhiều thuộc tính tốt, thể hiện ở các kết quả sau: Public key là phần dư của một đa thức một biến trên vòng đa thức  $h \in \mathbb{Z}_q[x]$ , với bậc lớn nhất là  $n-1$ . Với những ưu điểm của lưới NTRU khi áp dụng việc tạo khóa công khai như vậy, khung GPV được sử dụng với lưới NTRU đảm bảo an ninh cho hệ thống Falcon [3].
    - Lưới NTRU được biểu diễn như sau:  $\varphi = x^n + 1$  với  $(n = 2^k)$
    - Private key của NTRU là một bộ gồm 4 đa thức  $f, g, F, G \in \mathbb{Z}[x]/(\varphi)$ :  $fG - Gf = q \text{ mod } \varphi$ , trong đó đa thức  $f$  phải khả nghịch modulo  $q$ .
    - Đối với khóa công khai, đa thức  $h$  có thể được tính bằng:  $h \leftarrow g.f^{-1} \text{ mod } q$ , đa thức  $h$  được gọi là khóa công khai.
- Do đó, đối với hệ thống Falcon, khóa công khai là đa thức  $h$  và khóa bí mật là một tập gồm bốn đa thức  $f, g, F, G$ .

## 2.2 Reasons to choose Falcon Post-Quantum Signature Scheme

Trong khung GPV,  $v$  được tính toán dựa trên các thuật toán ngẫu nhiên được tạo ra từ biến thể thuật toán để tìm mặt phẳng gần nhất tương ứng với  $v$  [5]. Vì thuật toán để tìm mặt phẳng gần nhất nguyên thủy dễ bị tấn công vào tập cơ sở tương ứng với  $B$  (private key), nên scheme này không an toàn. Tuy nhiên, điều này đã được giải quyết khi thuật toán được sử dụng với một  $m$  và lấy mẫu  $s$  theo phân phối yêu cầu [4]. Phân phối Gaussian trên translated lattice là  $C_0 + A_q^\perp$ . Phương pháp này đã được chứng minh tăng tính bảo mật hơn cho  $B$ , và đây là thuật toán đầu tiên sử dụng tập hợp lấy mẫu trapdoor.

Hai ma trận  $\begin{bmatrix} 1 & h \\ 0 & q \end{bmatrix}$  và  $\begin{bmatrix} f & g \\ F & G \end{bmatrix}$  nằm trên cùng một lattice, nếu đa thức  $f$  và  $g$  đủ lớn thì public key  $h$  được tạo ra đảm bảo tính ngẫu nhiên tốt [3].

Tuy nhiên, khi tạo  $f$  và  $g$  nhỏ hơn thì vẫn khó tìm được các đa thức trùng khớp với  $f'$  và  $g'$  và thỏa mãn điều kiện  $h = g' \cdot (f')^{-1} \bmod q$ . Làm cho lattice NTRU khó giải khi lattice đủ lớn, tăng độ phức tạp và đảm bảo tính có khả năng chống lại máy tính lượng tử.

## 2.3 Implementation of the Falcon Post-Quantum Signature Scheme

Quá trình tạo các cặp khóa, chữ ký và xác thực chữ ký số của thuật toán Falcon dựa trên cơ sở GVP được thực hiện như sau:

- Đối với tạo cặp khóa, public key là  $A = [1|h^*]$ , đa thức  $h$  đã biết trước.
- Private key là:  $B = \begin{bmatrix} g & -f \\ G & -F \end{bmatrix}$
- Xác thực khóa 2 vector trực giao  $A$  và  $B$  thông qua biểu thức:  $B \times A^* = 0 \bmod q$
- Đối với chữ ký số, chữ ký cho một thông điệp  $m$  có dạng giá trị "salt"  $r$  đi cùng với một cặp đa thức  $(s_1, s_2)$  thỏa mãn:  $s_1 + s_2 h = H(r||m)$ . Vì  $(s_1$  được xác định hoàn toàn bởi  $m$ ,  $r$  và  $s_2)$ ; chữ ký là một dạng cặp của  $(r, s_2)$  [2]



## 3 Implement

### 3.1 Tools and resources

- Ngôn ngữ lập trình:
  - Python: sử dụng các thư viện như *fastAPI* web framework để build API hỗ trợ code backend. Các thư viện khác như *sqlalchemy* để tương tác với cơ sở dữ liệu MySQL; *PyPDF2* để làm việc với file PDF; *qrcode* tạo mã QR và nhiều thư viện hỗ trợ khác.
  - Javascript kết hợp HTML: xây dựng frontend.
- Cơ sở dữ liệu: MySQL - có giao diện và cách sử dụng dễ dàng và tiện lợi, có thể lưu trữ các dữ liệu của người dùng ở các dạng khác nhau.
- Opensource: code implement bằng python của Falcon được công bố bởi NIST.

### 3.2 Design and code

#### 3.2.1 System design

Sơ đồ hệ thống được thiết kế gồm 4 bên liên quan: Nhà phát hành, các đơn vị tuyển dụng, sinh viên và Root CA

- Nhà phát hành: Nơi ký và phát hành các văn bằng.
- Các đơn vị tuyển dụng (Tổ chức, doanh nghiệp, trường học,...): muốn xác thực nội dung và nơi cấp các văn bằng.
- Sinh viên: được cung cấp và sử dụng văn bằng theo mục đích cá nhân của họ.
- Root CA: đóng vai trò tin cậy, có khả năng tạo, ký, xác thực, đảm bảo tính toàn vẹn và tin cậy của các certificate cho Nhà phát hành.

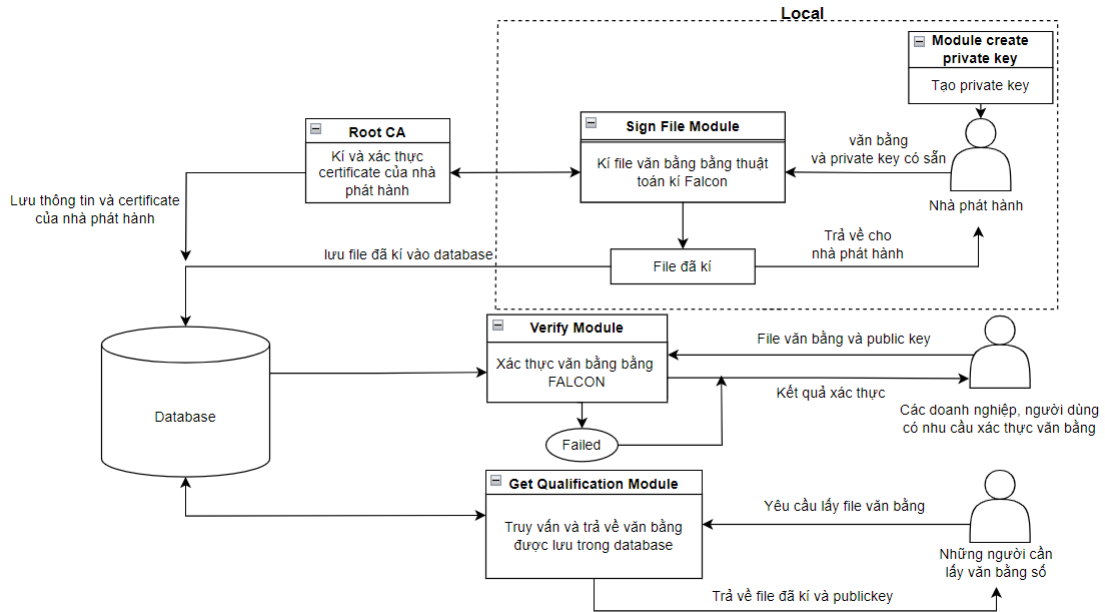


Figure 2: Sơ đồ hệ thống

### 3.2.2 Create Database

Tạo database với các trường, thuộc tính được lưu ba bảng trong cơ sở dữ liệu "DATA\_FALCON" và xác định các quan hệ giữa chúng thông qua các khóa ngoại. Các bảng sẽ được sử dụng để lưu trữ thông tin về cơ sở giáo dục, thông tin sinh viên và các bằng cấp của sinh viên.

Trong đó có các thuộc tính đặc biệt như *institution\_name* và *name\_email\_address* trong bảng *Institution* là unique đảm bảo rằng mỗi tổ chức (institution), mỗi địa chỉ email của các tổ chức trong bảng là duy nhất, không bị trùng lặp.

```

1 CREATE DATABASE IF NOT EXISTS DATA_FALCON;
2 USE DATA_FALCON;
3 CREATE TABLE Institution (
4     institution_id INT AUTO_INCREMENT PRIMARY KEY,
5     institution_name VARCHAR(255) UNIQUE,
6     authority_person VARCHAR(255),
7     email_address VARCHAR(255) UNIQUE,
8     public_file LONGBLOB ,
9     certificate_file LONGBLOB
10 );
11 CREATE TABLE StudentInfor (
12     id_sv INT AUTO_INCREMENT PRIMARY KEY,
13     school VARCHAR(255),
14     student_name VARCHAR(255)
15 );
16 CREATE TABLE Qualifications (
17     degree_code INT AUTO_INCREMENT PRIMARY KEY,
18     id_sv INT,
19     issue_date DATE,
20     expiration_date DATE,
21     institution_id INT,
22     pdf_file LONGBLOB,
23     FOREIGN KEY (id_sv) REFERENCES StudentInfor (
24         id_sv),
25     FOREIGN KEY (institution_id) REFERENCES
26     Institution (institution_id)
27 );

```

Mỗi bản ghi trong bảng "Qualifications" liên kết với một cơ sở giáo dục trong bảng "Institution" thông qua trường "institution \_id".

Mỗi bản ghi trong bảng "Qualifications" liên kết với một sinh viên trong bảng "StudentInfor" thông qua trường "id \_sv".

Quan hệ này cho phép lưu trữ thông tin về bằng cấp của sinh viên và liên kết chúng với cả thông tin về cơ sở giáo dục và sinh viên tương ứng.

### 3.2.3 Root CA

Root CA đóng vai trò một bên tin cậy, cung cấp dịch vụ chứng thực chữ ký số, giới hạn các quyền hạn cho Nhà phát hành, đảm bảo tính toàn vẹn thông tin và chỉ có những bên được phép mới có thể truy cập vào những thông tin nhạy cảm.

Root CA trong đề án này đóng vai trò là tạo, ký và xác thực certificate cho Nhà phát hành dựa trên một dạng nhất định, giúp cho người dùng biết được nguồn gốc của các văn bằng chứng chỉ được ký.

Root CA bao gồm 3 phần sau giúp hỗ trợ cho việc ký của Nhà phát hành cũng như việc xác thực của các đơn vị tuyển dụng.

- Định dạng của certificate:

```
1 certificate_format = {
2   "Version": None,
3   "Issuer": None,
4   "Subject": None,
5   "Author": None,
6   "Public Key Algorithm": None,
7   "Public Key": None,
8   "Validity" : {
9     "Not Before" : None,
10    "Not After" : None
11  },
12  "Signature Algorithm" : None,
13  "Signature" : None
14  }
15
```

Đây là một biến dictionary định nghĩa định dạng của một chứng chỉ. Certificate cho phép lưu trữ các thông tin về Nhà phát hành như: version, tác giả, publickey, thời gian hiệu lực, chữ ký số,...

- Tạo chứng chỉ:

```
1 def create_cert(info: dict, cert_file: str):
2
3     cert_content = "\n".join([f"{key}: {value}"
4     for key, value in info.items()])
5
6     cert_base64 = base64.b64encode(cert_content.
7     encode("utf-8"))
8
9     begin = b"---BEGIN CERTIFICATE---\n"
10    end = b"\n---END CERTIFICATE---\n"
11
12    with open(cert_file, "wb") as cert_file:
13        cert_file.write(begin)
14        cert_file.write(cert_base64)
15        cert_file.write(end)
```

Hàm *create\_cert* nhận thông tin input theo dạng dictionary và tên tệp; tạo nội dung dựa trên format đã định nghĩa. Sau đó, hàm mã hóa nội dung chứng chỉ thành base64 và lưu nó vào tệp tin chứng chỉ dưới dạng file PEM.

- Phân tích chứng chỉ:

```
1 def parse_cert(cert_file: str):
2
3     cert_base64 = cert_file.strip().replace("---
4     BEGIN CERTIFICATE---", "").replace("---END
5     CERTIFICATE---", "")
6
7     cert_content = base64.b64decode(cert_base64)
8     .decode("utf-8")
9
10    cert_info = certificate_format
11    lines = cert_content.split("\n")
```

```

11     for line in lines:
12         key, value = line.split(": ", 1)
13         cert_info[key] = value
14         if key == "Validity":
15             value = eval(value)
16             val = {}
17             val["Not Before"] = value["Not
Before"]
18             val["Not After"] = value["Not After
"]
19             cert_info["Validity"] = val
20
21     decoded_public_key = base64.b64decode(str(
cert_info['Public Key']).encode())
22     restored_key = pickle.loads(
decoded_public_key)
23     cert_info["Public Key"] = restored_key
24     return cert_info
25

```

Hàm *parse\_cert* sẽ phân tích chứng chỉ để trích xuất thông tin. Đầu tiên, hàm sẽ decode base64 để nhận lại nội dung chứng chỉ. Tiếp theo, hàm phân tích thông tin từ nội dung chứng chỉ thành một dictionary, sử dụng format đã định nghĩa. Trong quá trình phân tích, public key được khôi phục từ dạng base64 và chuyển đổi trở lại đối tượng ban đầu bằng pickle.

### 3.2.4 Nhà phát hành

Nhà phát hành đóng vai trò ký và phát hành các chứng chỉ đã được ký số dựa trên thuật toán Falcon. Nhà phát hành sẽ được xây dựng dựa trên 2 module hoạt động tại local nhằm bảo vệ private key khỏi bị tấn công từ bên ngoài vì private key chỉ mỗi bên Nhà phát hành được biết: *Sign File Module* - có vai trò ký các văn bản và *Create Private key Module* - có vai trò tạo private key ban đầu cho Nhà phát hành.

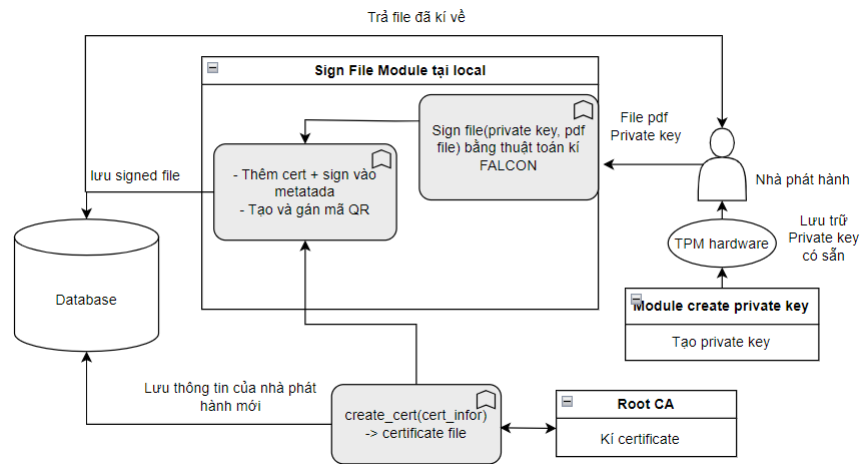


Figure 3: Sơ đồ Nhà phát hành

- Create Private key Module

```
1 class digital_signature():
2     def __init__(self):
3         self.signature = None
4         self.sk = None
5         self.pk = None
6     def create_key(self) -> (falcon.SecretKey,
7                             falcon.PublicKey):
8         self.sk = falcon.SecretKey(KEY_LENGTH)
9         self.pk = falcon.PublicKey(self.sk)
```

Khởi tạo cặp khóa Public và Private key dựa trên thuật toán Falcon.

```

1      def SaveSecret2Pem(self, filepath: str):
2          falcon_Private_key_begin = "----- Begin
Falcon Private Key -----\\n"
3          falcon_Private_key_end = "\\n----- End
Falcon Private Key -----\\n"
4          serialized_key = pickle.dumps(self.sk)
5          encoded_key = base64.b64encode(
serialized_key).decode('utf-8')
6          with open(filepath, 'w') as file:
7              file.write(falcon_Private_key_begin)
8              file.write(encoded_key)
9              file.write(falcon_Private_key_end)
10
11     def SavePublic2Pem(self, filepath: str):
12         falcon_public_key_begin = "----- Begin
Falcon Public Key -----\\n"
13         falcon_public_key_end = "\\n----- End
Falcon Public Key -----\\n"
14         serialized_key = pickle.dumps(self.pk)
15         encoded_key = base64.b64encode(
serialized_key).decode('utf-8')
16         with open(filepath, 'w') as file:
17             file.write(falcon_public_key_begin)
18             file.write(encoded_key)
19             file.write(falcon_public_key_end)
20

```

Các hàm giúp trả về người dùng file Private key và Public key dạng Pem. Để bảo mật, người dùng có thể sử dụng TPM Hardware hoặc BitLocker để có thể seal dữ liệu tại local một cách an toàn.



- Sign File Module:

```
1 def signing_pdf(self, input_file: bytes):
2     tmp = "./test/_tmp.pdf"
3     with open(tmp, "wb") as file:
4         file.write(input_file)
5     doc = fitz.open(tmp)
6     text = ""
7     for page_num in range(doc.page_count):
8         page = doc[page_num]
9         text += page.get_text()
10    sig = self.sk.sign(text.encode("utf-8"))
11    sign_b64 = base64.b64encode(sig).decode()
12    self.signature = sign_b64
13    doc.close()
14    os.remove(tmp)
15
```

Tạo chữ ký số Falcon cho văn bản và trả về chữ ký số được dưới dạng base64.

```

1 def add_data_to_metadata(self, pdf_bytes: bytes,
2   cert_file: bytes, output_file: str):
3     try:
4         writer = PyPDF2.PdfWriter()
5         pdf_stream = io.BytesIO(pdf_bytes)
6         reader = PyPDF2.PdfReader(pdf_stream)
7         metadata = reader.metadata
8         extracted_metadata = {}
9         extracted_metadata['/Signature'] =
10        self.signature
11        extracted_metadata['/Certificate'] =
12        cert_file.decode()
13        writer.append_pages_from_reader(
14        reader)
15        for key in metadata:
16            writer.add_metadata({PyPDF2.
17            generic.create_string_object(key): PyPDF2.
18            generic.create_string_object(str(metadata[key
19            ]))})
20        for key in extracted_metadata:
21            writer.add_metadata({PyPDF2.
22            generic.create_string_object(key): PyPDF2.
23            generic.create_string_object(str(
24            extracted_metadata[key]))})
25        with open(output_file, 'wb') as fout
26        :
27            writer.write(fout)
28        except Exception as e:
29            print(f"Error adding signature to
30            PDF metadata: {e}")

```

Sau khi tạo chữ ký số và có được certificate từ Root CA, ta sẽ thêm vào metadata của file văn bản.

Ngoài ra, trong *Sign File Module* còn hỗ trợ các tính năng:

- Kiểm tra tính hợp lệ của địa chỉ email bằng cách gọi hàm *is\_valid\_email*, nếu địa chỉ email không hợp lệ, API sẽ trả về thông báo lỗi tương ứng.
- Khởi tạo một đối tượng *digital\_signature* từ dữ liệu đầu vào, bao gồm authority, ins, studentName, school, đối tượng này đại diện cho thông tin về chữ ký số.
- Tải private key từ dữ liệu đầu vào và giải mã từ base64.
- Kiểm tra xem *ins* (tổ chức phát hành) đã tồn tại trong cơ sở dữ liệu chưa, nếu chưa, một chứng chỉ mới được tạo và lưu vào cơ sở dữ liệu.
- Kiểm tra xem *studentName* (tên sinh viên) đã tồn tại trong cơ sở dữ liệu chưa, nếu chưa, sinh viên mới sẽ được thêm vào cơ sở dữ liệu.
- Giải mã tệp đầu vào từ base64 và lưu dưới dạng một tệp tạm thời.
- Sử dụng thư viện *PyPDF2* để mở tệp PDF đầu vào và thực hiện quá trình ký văn bản.
- Chứng chỉ và chữ ký được thêm vào metadata của tệp PDF đầu vào.
- Sinh mã QR từ thông tin sinh viên và chữ ký để tiện lợi cho việc truy xuất nhanh thông tin, mã QR được thêm vào tệp PDF.
- Tệp PDF sau khi đã ký được ghi lại và trả về dưới dạng đối tượng JSON kèm theo thông tin về văn bản.
- Lưu các bản ghi đã ký lên database giúp người dùng dễ dàng truy xuất.

### 3.2.5 Sinh viên

Sinh viên là người dùng muốn tải các văn bằng online để ứng tuyển vào các Nhà tuyển dụng.

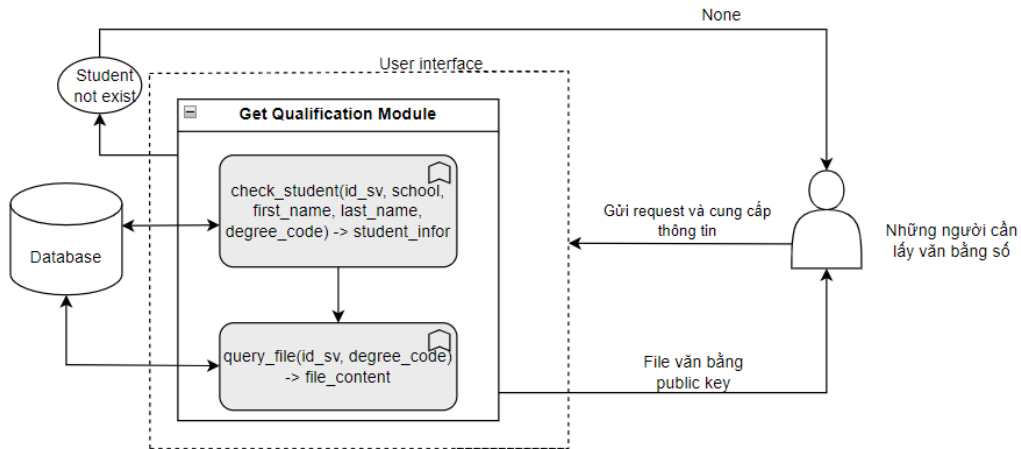


Figure 4: Hệ thống lấy văn bằng

Sau khi kết nối với cơ sở dữ liệu, có thể xử lý yêu cầu và truy vấn:

```
1 student_id = data.get("student_id")
2 qual_code = data.get("qual_code")
3
4 student_info = db.query(Students).filter(
5     Students.id_sv == student_id
6 ).first()
7
8 if not student_info:
9     raise HTTPException(status_code=404, detail="
10         Student not found")
11
12 qual_info = db.query(Quals).filter(
13     Quals.id_sv == student_info.id_sv,
14     Quals.degree_code == qual_code
15 ).first()
```

```

16 if not qual_info:
17     raise HTTPException(status_code=404, detail="
    Qualification not found")
18
19 ins = db.query(Ins).filter(Ins.institution_id ==
    qual_info.institution_id).first()
20
21 if not ins:
22     raise HTTPException(status_code=404, detail="
    Institution not found")
23
24 public_base64 = base64.b64encode(ins.public_file).
    decode("utf-8")
25 pdf_file_content = qual_info.pdf_file
26 pdf_base64 = base64.b64encode(pdf_file_content).
    decode("utf-8")
27
28 response_data = {
29     "pdf_content_base64": pdf_base64,
30     "public": public_base64
31 }
32

```

- Dữ liệu đầu vào được trích xuất từ data, sau đó sử dụng để truy vấn thông tin sinh viên và bằng cấp từ cơ sở dữ liệu.
- Nếu sinh viên hoặc bằng cấp không tồn tại, một HTTPException sẽ thông báo lỗi.
- Sau đó, thông tin về cơ sở giáo dục (Ins), nội dung tệp PDF (pdf\_file\_content), và nội dung tệp public (public\_base64) được truy xuất và mã hóa.
- Cuối cùng, kết quả được trả về dưới dạng JSONResponse với nội dung là 'response\_data'.
- Sinh viên sẽ nhận kết quả trả về là 1 file pdf văn bằng đã ký và 1 file public key, từ đó có thể chuyển tiếp cho Nhà tuyển dụng để xác thực.

### 3.2.6 Nhà tuyển dụng

Sau khi Nhà tuyển dụng nhận được file văn bằng và public key từ sinh viên, có thể xác thực trên hệ thống:

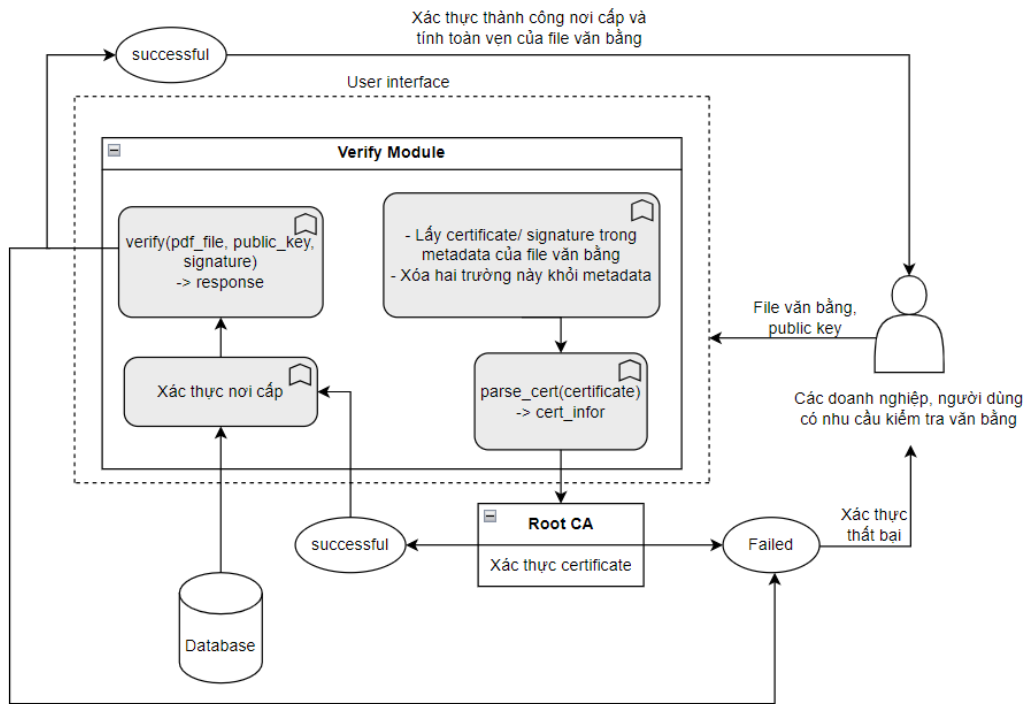


Figure 5: Hệ thống xác thực văn bằng

Sau khi tạo phiên và định nghĩa mô hình:

```
1 def verify(self, input_file: bytes) -> bool:
2     if not input_file:
3         return False
4     tmp = "./test/_tmp.pdf"
5     with open(tmp, "wb") as file:
6         file.write(input_file)
7     doc = fitz.open(tmp)
8     text = ""
9     for page_num in range(doc.page_count):
10         page = doc[page_num]
11         text += page.get_text()
```

```

12         sign = base64.b64decode(self.signature)
13         result = self.pk.verify(text.encode("utf-8")
, sign)
14         doc.close()
15         os.remove(tmp)
16         return result
17

```

Hàm xác thực chữ ký số dựa trên 3 input: chữ ký số encoded base64, file văn bản cần xác thực và public key. Ngoài ra, *Verify Module* sẽ xác thực nhiều bên liên quan:

```

1 if not form_data.file or not form_data.pubKey:
2     raise HTTPException(status_code=400, detail="
Empty file")
3 content = base64.b64decode(form_data.file)
4 pubkey = base64.b64decode(form_data.pubKey)
5 pair = dsa.digital_signature()
6 pair.load_public_key(pubkey)
7 cert = pair.detach_signature_and_cert(content)
8 response_data = {}
9

```

- Kiểm tra nội dung truyền vào: nếu không có nội dung hoặc public key, nếu không sẽ raise một lỗi HTTPException.
- Giải mã nội dung và public key từ base64.
- Sử dụng thư viện *digital\_signature* để tách chữ ký và chứng chỉ từ nội dung.

```

1 ca.load_CA_public_key("Root_CA/public.pem")
2 signature_ca = cert_info["Signature"]
3 raw_signature_ca = bytes.fromhex(signature_ca.
replace(":", ""))
4 message = cert_info.copy()
5 del message["Signature"]
6 message["Public Key"] = base64.b64encode(pickle.
dumps(cert_info["Public Key"])).decode('utf-8')
7 message = json.dumps(message, default=str)
8 hashed_message = hashlib.sha512(message.encode()).
digest()
9 res = ca.pk.verify(hashed_message, raw_signature_ca)

```

```

10 if res:
11     response_data["root_ca"] = True
12     response_data["root_ca_name"] = cert_info["
Issuer"]
13 else:
14     response_data["root_ca"] = False
15     return JsonResponse(content=response_data)
16

```

- Xác thực nội dung từ Root CA bằng cách tải public key của Root CA và xác thực chữ ký của Root CA.

```

1 ins = db.query(Ins).filter(Ins.institution_name ==
    cert_info.get("Subject"), Ins.authority_person ==
    cert_info["Author"]).first()
2 if not ins:
3     response_data["result"] = False
4     response_data["reason"] = "Khong co nha phat
hanh " + cert_info.get("Subject")
5     return JsonResponse(content=response_data)
6
7 ins_pubkey = dsa.digital_signature()
8 ins_pubkey.load_public_key(ins.public_file)
9

```

- Xác thực nơi cấp bằng cách truy vấn cơ sở dữ liệu về thông tin của Nhà phát hành và xác thực public key của Nhà phát hành.

```

1 def remove_data_from_metadata(self, input: bytes,
    output_file: str):
2     try:
3         input_file = io.BytesIO(input)
4         reader = PyPDF2.PdfReader(input_file)
5         writer = PyPDF2.PdfWriter()
6         metadata= reader.metadata
7         writer.append_pages_from_reader(reader)
8         for key in metadata:
9             if key == "/Signature" or key == "/"
Certificate":
10                 continue

```



```

11         writer.add_metadata({PyPDF2.generic.
    create_string_object(key): PyPDF2.generic.
    create_string_object(str(metadata[key]))})
12         with open(output_file, 'wb') as fout:
13             writer.write(fout)
14     except Exception as e:
15         print(f"Error removing metadata: {e}")
16 if str(pair.pk) == str(cert_info["Public Key"]) ==
    str(ins_pubkey.pk):
17     response_data["institution_name"] = ins.
    institution_name
18     response_data["authority_person"] = ins.
    authority_person
19 else:
20     response_data["result"] = False
21     response_data["reason"] = "Khong dung Public key
    "
22     return JsonResponse(content=response_data)
23 tmp = "test/tmp.pdf"
24 pair.remove_data_from_metadata(content, tmp)
25 content = qr.remove_qr_code_from_pdf(tmp)
26

```

- Kiểm tra thông tin của Public key và xóa những thông tin không cần thiết để kiểm tra tính toàn vẹn.

```

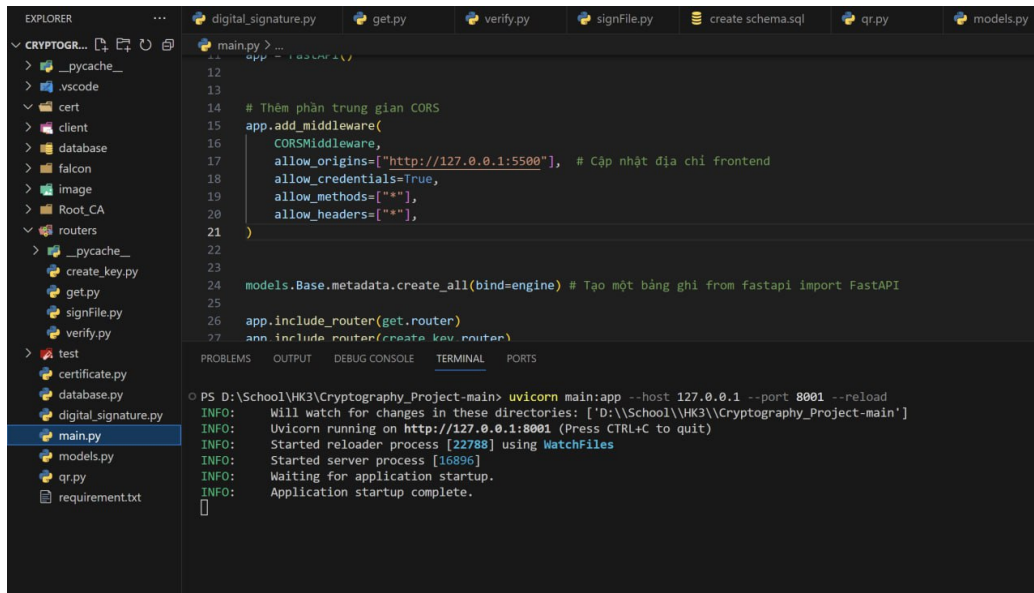
1 out = "test/pdf_tmp.pdf"
2 with open(out, "rb") as file:
3     _content = file.read()
4     verification_result = pair.verify(_content)
5     response_data["result"] = verification_result
6     if verification_result == False:
7         response_data["reason"]
8

```

- Xác thực tính toàn vẹn của chữ ký sau khi gỡ bỏ những thông tin không cần thiết.

## 4 Testing

Trước khi test, cần chạy các câu lệnh hệ thống:



The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like `__pycache__`, `.vscode`, `cert`, `client`, `database`, `falcon`, `image`, `Root_CA`, `routers`, and files like `create_key.py`, `signFile.py`, `verify.py`, `test`, `certificate.py`, `database.py`, `digital_signature.py`, `main.py`, `models.py`, `qr.py`, and `requirement.txt`. The code editor shows the `main.py` file with the following code:

```
11 app = FastAPI()
12
13
14 # Thêm phân trung gian CORS
15 app.add_middleware(
16     CORSMiddleware,
17     allow_origins=["http://127.0.0.1:5500"], # Cập nhật địa chỉ frontend
18     allow_credentials=True,
19     allow_methods=["*"],
20     allow_headers=["*"],
21 )
22
23
24 models.Base.metadata.create_all(bind=engine) # Tạo một bảng ghi from fastapi import FastAPI
25
26 app.include_router(get.router)
27 app.include_router(create_key.router)
```

The terminal at the bottom shows the output of the `uvicorn` command:

```
PS D:\School\HK3\Cryptography_Project-main> uvicorn main:app --host 127.0.0.1 --port 8001 --reload
INFO: Will watch for changes in these directories: ['D:\School\HK3\Cryptography_Project-main']
INFO: Uvicorn running on http://127.0.0.1:8001 (Press CTRL+C to quit)
INFO: Started reloader process [22788] using WatchFiles
INFO: Started server process [16896]
INFO: Waiting for application startup.
INFO: Application startup complete.
```

Figure 6: Run backend

Chạy câu lệnh `uvicorn main:app --host 127.0.0.1 --port 8001 --reload`  
Sau đó tạo cơ sở dữ liệu:

```
C:\Users\admin>mysql -u root -p < "D:\School\HK3\Cryptography_Project-main\database\create schema.sql"
Enter password: *****
C:\Users\admin>
```

Figure 7: Tạo cơ sở dữ liệu

Giao diện chung của hệ thống:

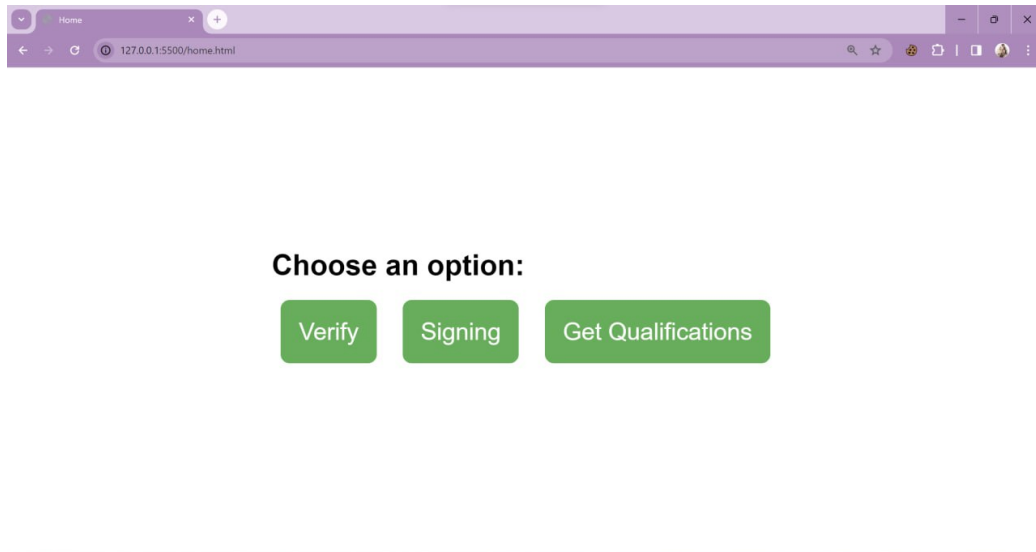


Figure 8: Enter Caption

Có các sự lựa chọn nhằm vào từng đối tượng khác nhau: *Verify* - Nhà tuyển dụng, *Signing* - Nhà phát hành, *Get Qualifications* - Sinh viên cần lấy văn bằng.

## 4.1 Nhà phát hành

Giao diện:

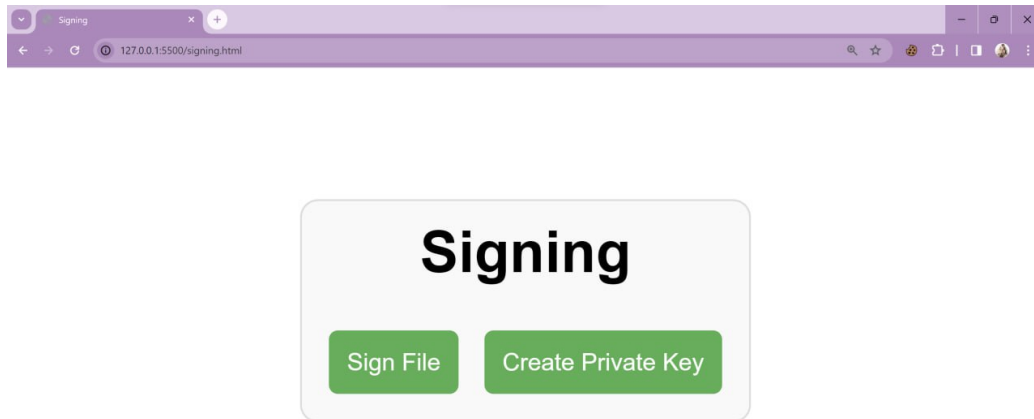


Figure 9: Giao diện của Nhà phát hành

Giao diện của Nhà phát hành với 2 chọn lựa: *Sign File* - ký văn bản và *Create Private Key* - tạo private key cho Nhà phát hành.

#### 4.1.1 Tạo private key

Trước tiên, cần tạo một thư mục/usb token hoạt động dựa trên TPM Hardware và Bitlocker để có thể bảo vệ Private key:

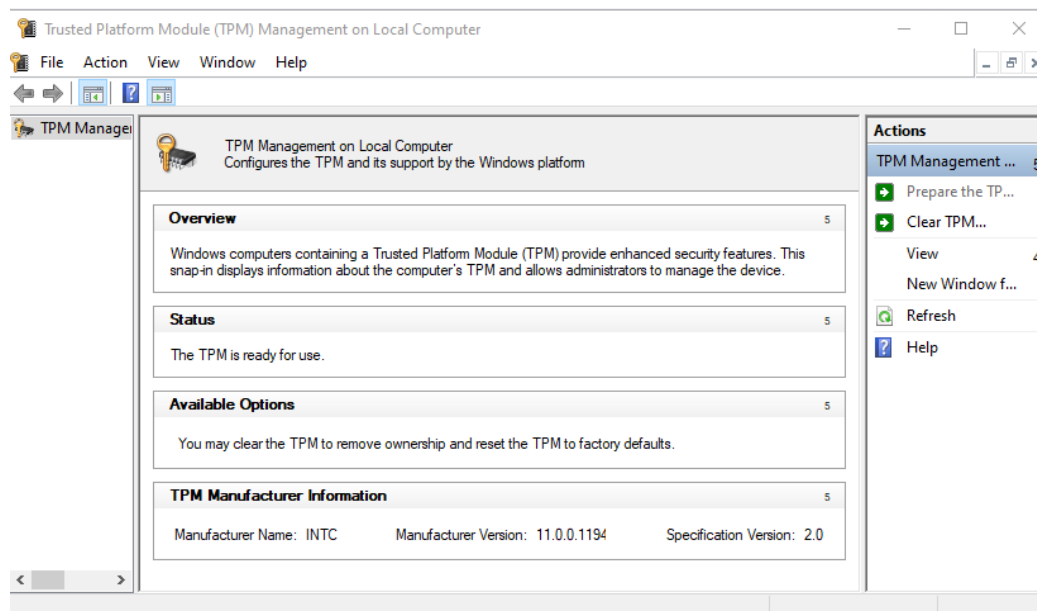


Figure 10: Kiểm tra TPM Hardware trên máy

Sau khi khởi động TPM Hardware, sử dụng thêm BitLocker của Windows để có thể seal dữ liệu, lưu recovery-key vào TPM Haradware:

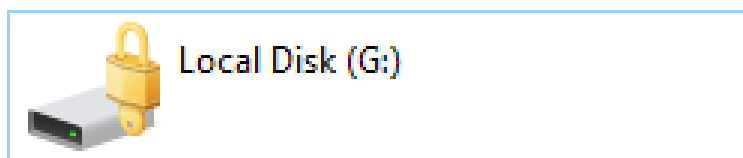


Figure 11: Đã cài đặt BitLocker

Bắt đầu test tạo private key:

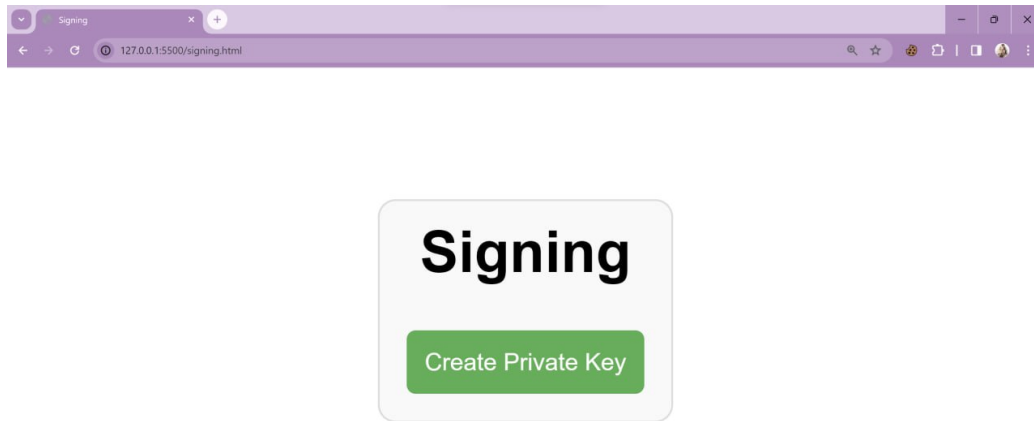


Figure 12: Tạo Private key

Chọn vào button *Create Private Key* sẽ hiển thị folder, cần mở khóa folder đã seal trước khi lưu file:

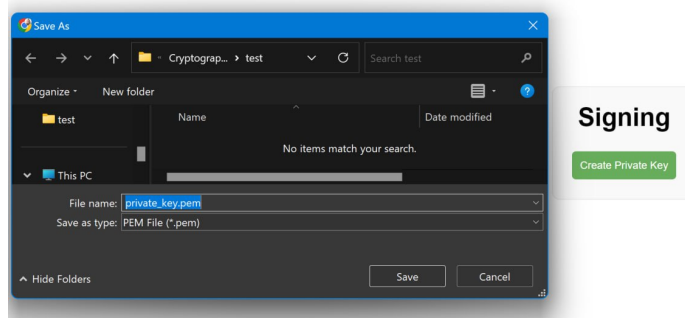


Figure 13: Tạo và lưu Private key

### 4.1.2 Kí và phát hành văn bằng

Sau khi có và lưu trữ thành công private key, ta quay trở lại giao diện ban đầu và ấn vào button *Sign File* sẽ hiện ra giao diện:

The form is titled "Signing" and is divided into two main sections: "Enter information of institution" and "Enter information of student".

**Enter information of institution**

- Institution Name:** A text input field with the placeholder "Institution Name".
- Authority Name:** A text input field with the placeholder "Authority Name".
- Email:** A text input field with the placeholder "Email".

**Enter information of student**

- School:** A text input field with the placeholder "School".
- Student Name:** A text input field with the placeholder "Student Name".

**Qualification File:** A file upload section with a "Choose File" button and the text "No file chosen".

**Private Key:** A file upload section with a "Choose File" button and the text "No file chosen".

**Sign File:** A green button at the bottom of the form.

Figure 14: Giao diện thô

Ta cần điền các thông tin cần thiết và chọn file private key đã tạo và lưu trữ trước đó để kí văn bằng, lúc này sẽ hiện ra giao diện để chọn folder lưu file đã kí.

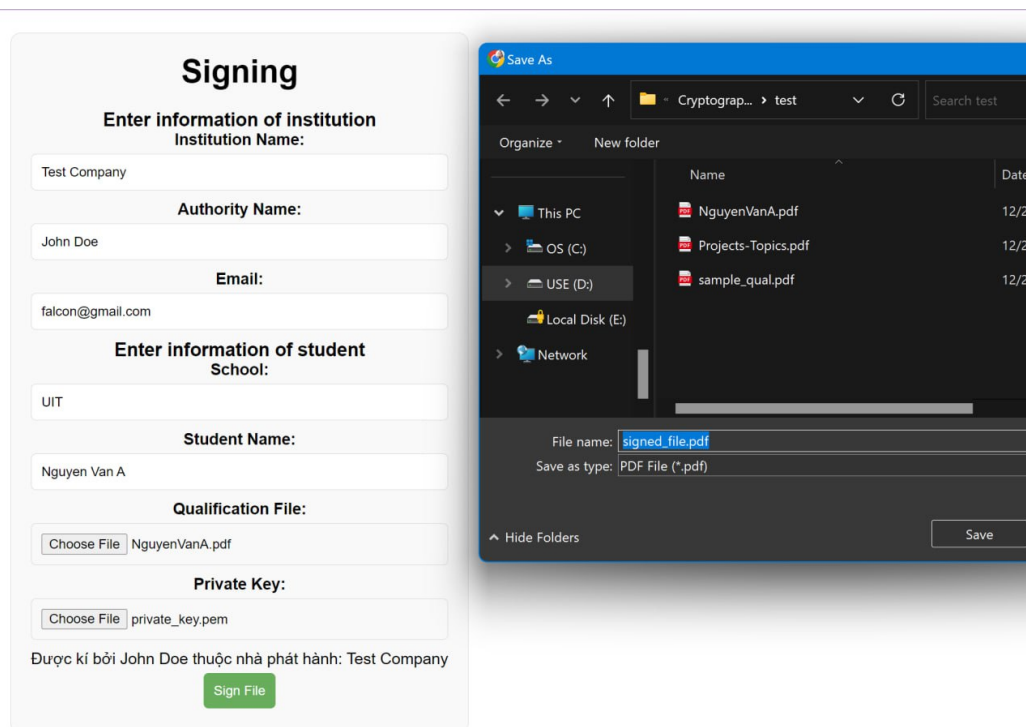


Figure 15: Đã kí và Nhà phát hành chọn nơi lưu trữ



Lúc này, tại database đã được cập nhật thông tin từ local

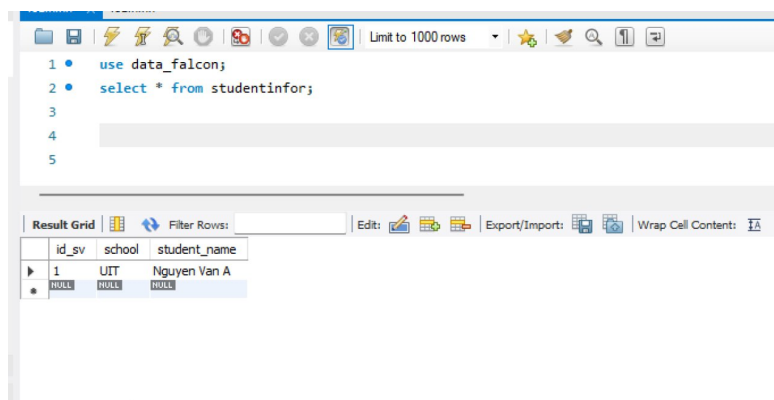


Figure 16: Minh họa database 1

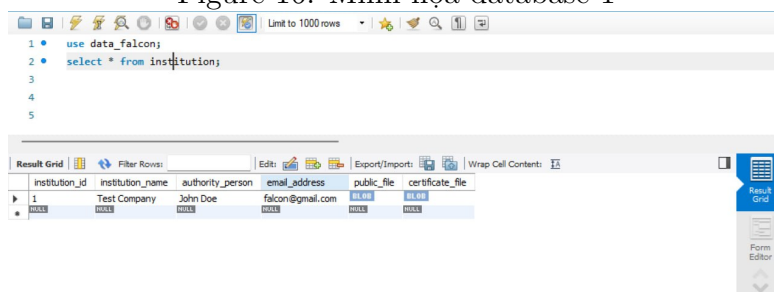


Figure 17: Minh họa database 2

Các dữ liệu về các file đã kí đã được đưa vào database từ local, sẵn sàng cho Sinh viên truy xuất cũng như cho Nhà tuyển dụng xác thực.

## 4.2 Sinh viên

Sau khi chọn mục dành cho sinh viên - Get Qualification sẽ hiện ra giao diện:

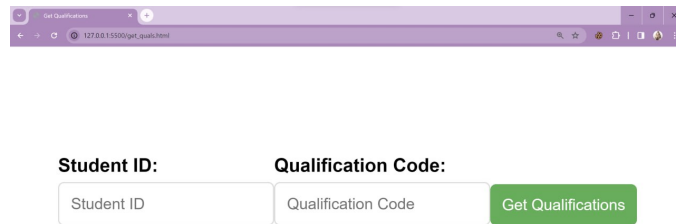


Figure 18: Giao diện lấy văn bằng

Sinh viên cần điền đúng số Student ID và mã Code (đã được cấp trên bản giấy) của văn bằng sẽ trả về 2 kết quả. Như vậy, sinh viên đã có thể dùng 2 file được trả về gửi đến Nhà tuyển dụng để xác thực:

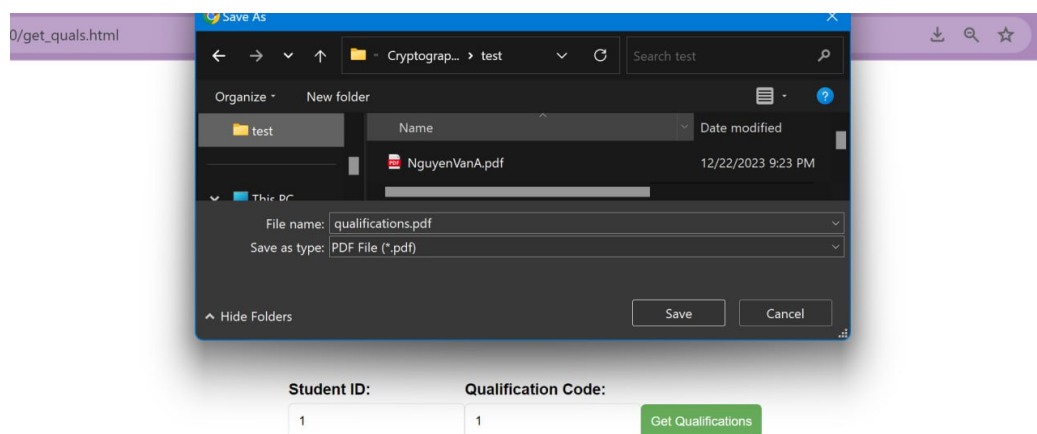


Figure 19: Trả về file pdf đã kí

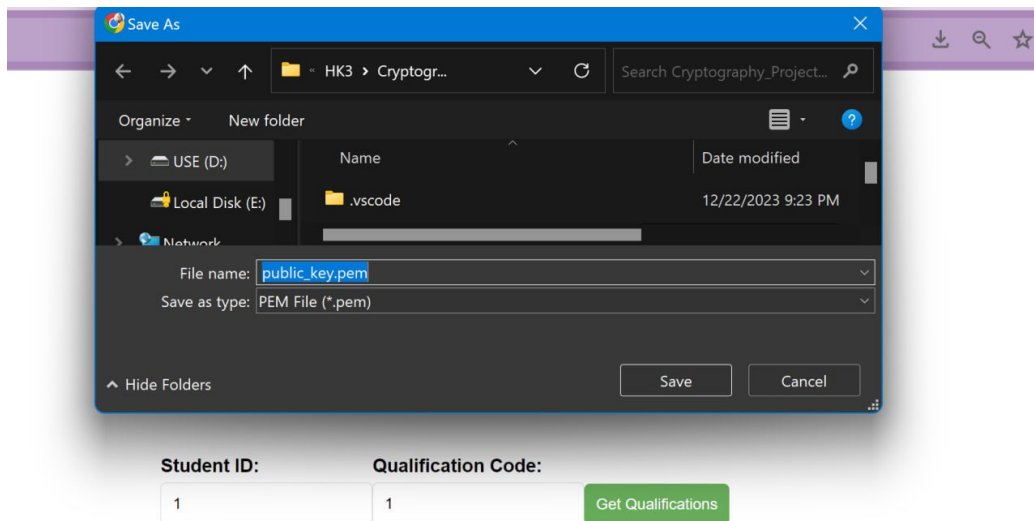


Figure 20: Trả về file public key

### 4.3 Nhà tuyển dụng

Sau khi Nhà tuyển dụng nhận được 2 file public key và file pdf đã kí tương ứng, có thể xác thực bằng cách chuyển đến giao diện *Verify*:

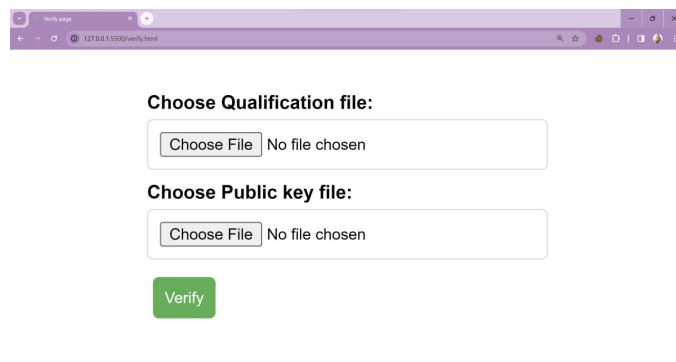
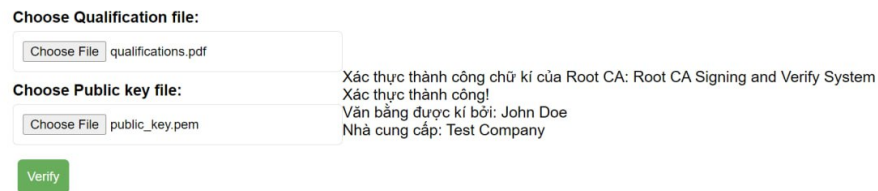


Figure 21: Giao diện xác thực văn bằng

Nhà tuyển dụng tải lên cách file được sinh viên gửi, giao diện sẽ trả về kết quả xác thực:



The screenshot shows a web interface for file verification. It has two sections: 'Choose Qualification file:' and 'Choose Public key file:'. The first section has a 'Choose File' button and the filename 'qualifications.pdf'. The second section has a 'Choose File' button and the filename 'public\_key.pem'. To the right of these sections, the verification result is displayed in Vietnamese: 'Xác thực thành công chữ kí của Root CA: Root CA Signing and Verify System', 'Xác thực thành công!', 'Văn bằng được kí bởi: John Doe', and 'Nhà cung cấp: Test Company'. At the bottom left, there is a green 'Verify' button.

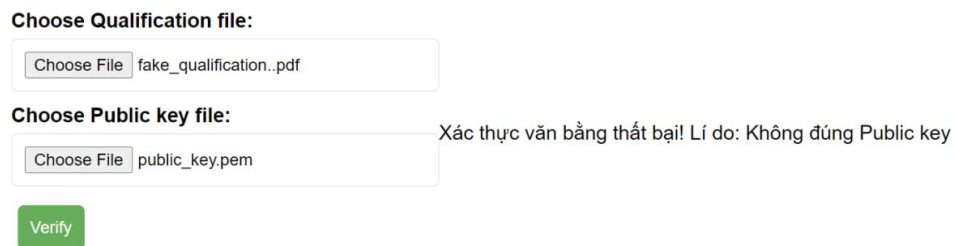
**Choose Qualification file:**  
Choose File qualifications.pdf

**Choose Public key file:**  
Choose File public\_key.pem

Xác thực thành công chữ kí của Root CA: Root CA Signing and Verify System  
Xác thực thành công!  
Văn bằng được kí bởi: John Doe  
Nhà cung cấp: Test Company

Verify

Figure 22: Xác thực thành công



The screenshot shows a web interface similar to Figure 22, but with a failed verification result. The 'Choose Qualification file:' section has a 'Choose File' button and the filename 'fake\_qualification..pdf'. The 'Choose Public key file:' section has a 'Choose File' button and the filename 'public\_key.pem'. To the right, the error message is displayed in Vietnamese: 'Xác thực văn bằng thất bại! Lý do: Không đúng Public key'. At the bottom left, there is a green 'Verify' button.

**Choose Qualification file:**  
Choose File fake\_qualification..pdf

**Choose Public key file:**  
Choose File public\_key.pem

Xác thực văn bằng thất bại! Lý do: Không đúng Public key

Verify

Figure 23: Xác thực thất bại

## 5 Sumary

### 5.1 Results

Thông qua quá trình xây dựng hệ thống chữ ký số áp dụng thuật toán chữ ký Falcon, chúng em phần nào đã xây dựng được hệ thống đáp ứng được những mục tiêu đề ra như:

- Tính xác thực: thông qua certificate có thể xác định được danh tính của nơi cấp.
- Tính bảo mật: sử dụng sơ đồ chữ ký số Falcon, nâng cao tính bảo mật trong thời đại hậu lượng tử.
- Tính toàn vẹn: đảm bảo nội dung văn bản không bị giả mạo hoặc sửa đổi một cách trái phép.

Đầu tiên, chúng em đã xây dựng được hệ thống Root CA sử dụng kí bằng thuật toán Falcon hỗ trợ trong việc xác thực của Nhà phát hành.

Chúng em đã xây dựng được hệ thống kí - nhận - xác thực các văn bản giữa các bên liên quan với hệ thống chữ ký hậu lượng tử Falcon có thể nâng cao khả năng chống lại sức mạnh tấn công của máy tính lượng tử. Tại hệ thống này, người dùng như Nhà phát hành dễ dàng kí và có thể tải các dữ liệu lên database giúp cho sinh viên dễ dàng truy xuất; Sinh viên cũng như Nhà tuyển dụng thuận tiện trong việc lấy cũng như xác thực văn bản một cách tiện lợi, tiết kiệm thời gian.

Nhìn chung, xây dựng hệ thống chữ ký số bằng thuật toán hậu lượng Falcon đã góp phần giúp việc kí và xác thực trở nên tiện lợi, nhanh chóng cho các bên cũng như có tính bảo mật hiệu quả hơn trong thời đại hiện nay.

## 5.2 Tasks Chart

Task	Quỳnh Anh	Thanh Bình
Front-end coding	X	
Back-end coding	X	X
System design	X	X
Testing	X	X
Report	X	X

Figure 24: Tasks chart

## 5.3 Final Words

Kính gửi thầy Nguyễn Ngọc Tự,

Chúng em nhân dịp này xin gửi lời cảm ơn đến thầy vì những bài giảng, những lời góp ý, giải thích và gợi ý cho chúng em hoàn thành đồ án cũng như môn học này. Chúng em rất biết ơn vì thầy đã mang đến cho chúng em một góc nhìn mới về nghề, về cách làm nghề và đó như một sự truyền cảm hứng, truyền lửa đến cho chúng em.

Chúng em rất vui khi thời gian qua được học tập và làm việc cùng thầy. Chúng em tin rằng những kiến thức và kĩ năng không chỉ từ đồ án, từ những buổi thực hành mà tất cả những bài giảng của thầy sẽ là hành trang vào nghề vô cùng to quý với chúng em.

Một lần nữa, chúng em xin cảm ơn về những sự hỗ trợ và hướng dẫn từ thầy, chúc thầy luôn thành công trong chặng đường sắp tới và đào tạo ra thật nhiều thế hệ giỏi tiếp theo.

Trân trọng,

Nguyễn Thị Quỳnh Anh

Nguyễn Thanh Bình.

## 6 References

- [1] Beckwith, L., Nguyen, D. T., & Gaj, K. (2023). Hardware Accelerators for Digital Signature Algorithms Dilithium and FALCON. IEEE Design & Test.
- [2] Luc, N. Q., Nguyen, T. T., Quach, D. H., Dao, T. T., & Pham, N. T. (2023). Building Applications and Developing Digital Signature Devices based on the Falcon Post-Quantum Digital Signature Scheme. Engineering, Technology & Applied Science Research, 13(2), 10401-10406.
- [3] Stehlé, D., & Steinfeld, R. (2011). Making NTRU as secure as worst-case problems over ideal lattices. In Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings 30 (pp. 27-47). Springer Berlin Heidelberg.
- [4] Klein, P. (2000, February). Finding the closest lattice vector when it's unusually close. In Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms (pp. 937-941).
- [5] McGuire, G., & Robinson, O. (2020). Lattice sieving in three dimensions for discrete log in medium characteristic. Journal of Mathematical Cryptology, 15(1), 223-236.