

VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
FALCUTY OF COMPUTER NETWORKS AND COMMUNICATIONS



MIDTERM REPORT

DESIGN AND IMPLEMENTATION OF ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM FOR MESSAGE INTEGRITY VERIFICATION.

Instructor: Ph.D.Nguyễn Ngọc Tự

Course: NT219.N22.ATCL

Group members: Huỳnh Đình Khải Minh - 21521123
Trần Thành Lợi - 21522296
Nguyễn Nguyễn Duy An - 21520536



Table of contents

1	Scenario	2
2	Research motivations	2
3	Related Work	2
4	Proposed scheme	2
4.1	Digital Signature	2
4.2	CRYSTALS-Dilithium	4
4.2.1	Overview	4
4.2.2	Scientific Background	5
5	Demo	6
5.1	Overview	6
5.2	Application concept	6
5.3	Deployment	7
6	References	8

1 Scenario

Bên phát hành A cần pushlish một tài liệu trên diện rộng đến các recipient, tài liệu được gửi đi sẽ được giữ ở bản gốc để recipient có thể tiếp cận và đọc được dễ dàng. Tuy nhiên điều này sẽ đặt ra vấn đề liên quan đến việc xác thực tính toàn vẹn từ dữ liệu với bên recipient. Để bảo toàn tính toàn vẹn của tài liệu, bên A trước khi pushlish sẽ thực hiện tạo một chữ ký số cho tài liệu đó và pushlish chữ ký kèm với tài liệu. Bên B sau khi tải hoặc nhận tài liệu sẽ tiến hành verify tài liệu dựa trên chữ ký đã được cung cấp từ bên pushlish. Mỗi tài liệu sẽ có một chữ ký số riêng biệt vậy nên việc thay đổi thông tin tài liệu là không thể, quá trình ký và xác thực này sẽ thực hiện bằng cách sử dụng các thuật toán tạo và xác thực chữ ký số.

2 Research motivations

Một số thuật toán tạo chữ ký số được biết đến và sử dụng hiện nay như RSA, DSA, các thuật toán dựa trên ECC như ECDSA. Các thuật toán chữ ký số hiện nay được xây dựng trên nền tảng các bài toán tính toán rất khó, như phân tích các số nguyên tố lớn và giải các phương trình đường cong elip. Tuy nhiên, các máy tính lượng tử có thể giải quyết những bài toán này một cách nhanh chóng và hiệu quả hơn, làm cho các thuật toán chữ ký số hiện nay trở nên dễ dàng bị tấn công. Tuy nhiên một số thuật toán DSA dựa trên lattice-based cryptography, thuật toán này được thiết kế để giải các bài toán tính toán khó hơn, ngay cả đối với các máy tính lượng tử. Một trong số đó đạt đủ tiêu chuẩn bởi NIST và đang được chuẩn hóa là CRYSTALS-DILITHIUM Algorithm.

3 Related Work

Một số công việc liên quan đến triển khai thuật toán trong đề án môn học này như:

- Đề án sẽ mô tả và tìm hiểu về triển khai thuật toán đồng thời đánh giá hiệu năng của thuật toán khi ký và xác thực trên các dòng máy tính phổ thông, đồng thời sẽ so sánh với các thuật toán PQC khác.
- Giới thiệu một ứng dụng có thể đồng thời ký và upload tài liệu, đồng thời cho phép người dùng tải về và xác thực.

4 Proposed scheme

4.1 Digital Signature

Đầu tiên ta sẽ nói về cách hoạt động của thuật toán tạo chữ ký số. Quá trình này bao gồm 3 phần là tạo khóa, ký và xác thực. Trong đó hai phần chính của thuật toán này là sign và verify dữ liệu.

- Key generation: A key generation algorithm that selects a private key uniformly at random from a set of possible private keys. The algorithm outputs the private key and a corresponding public key.
- Signing:

- A message that would be sent is first converted into a compact form called a message digest. Message digest (MD) is obtained by transforming message M using one-way hash function.

$$MD = H(M)$$

- Furthermore, the message digest (MD) is encrypted with the public key algorithm using a sender secret key (SK) into signature S

$$S = \text{ESK}(MD)$$

- Messages M is connected with signature S, then they are sent over through communication channel. In this case, we say that the message M has been signed by the sender with a digital signature S.



Figure 1: The signing process by the sender.

- Verifying:

- After the message M and its digital signature S are transmitted through a communication channel and received by the receiver, the authenticity of the message is verified by decrypting the digital signature S with the sender's public key (PK). This decryption generates the original message digest, MD, according to the formula.

$$MD = \text{DPK}(S)$$

- The sender then converts message M into a message digest MD' using the same one-way hash function. If MD' = MD, means that the received message is authentic and comes from the correct sender.

4.2 CRYSTALS-Dilithium

4.2.1 Overview

Dilithium is a digital signature scheme that is strongly secure under chosen message attacks based on the hardness of lattice problems over module lattices. The security notion means that an adversary having access to a signing oracle cannot produce a signature of a message whose signature he hasn't yet seen, nor produce a different signature of a message that he already saw signed. Dilithium is one of the candidate algorithms submitted to the NIST post-quantum cryptography project.

Algorithm 1 Key Generator

```
A  $\leftarrow R_q^{k \times l}$ 
 $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_n^l * S_n^k$ 
 $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ 
return  $(pk = (\mathbf{A}, \mathbf{t}), sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1), \mathbf{s}_2))$ 
```

Algorithm 2 Sign

```
 $\mathbf{z} := \perp$ 
while  $\mathbf{z} = \perp$  do
   $\mathbf{y} \leftarrow S_{\gamma_1-1}^l$ 
   $\mathbf{w}_1 := HighBits(\mathbf{A}\mathbf{y}, 2\gamma_2)$ 
   $c \in B_{60} := \mathbf{H}(M \parallel \mathbf{w}_1)$ 
   $\mathbf{z} := \mathbf{y} + c\mathbf{s}_1$ 
  IF  $\|\mathbf{z}\|_\infty$ 
    end
```

Algorithm 3 Key Generator

```
A  $\leftarrow R_q^{k \times l}$ 
 $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_n^l * S_n^k$ 
 $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ 
return  $(pk = (\mathbf{A}, \mathbf{t}), sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1), \mathbf{s}_2))$ 
```

4.2.2 Scientific Background

The design of Dilithium is based on the "Fiat-Shamir with Aborts" technique of Lyubashevsky which uses rejection sampling to make lattice-based Fiat-Shamir schemes compact and secure. The scheme with the smallest signature sizes using this approach is the one of Ducas, Durmus, Lepoint, and Lyubashevsky which is based on the NTRU assumption and crucially uses Gaussian sampling for creating signatures. Because Gaussian sampling is hard to implement securely and efficiently, we opted to only use the uniform distribution. Dilithium improves on the most efficient scheme that only uses the uniform distribution, due to Bai and Galbraith, by using a new technique that shrinks the public key by more than a factor of 2. To the best of our knowledge, Dilithium has the smallest public key + signature size of any lattice-based signature scheme that only uses uniform sampling.

- **Key Generator:**

1. Signer selects a random integer $dA \in [1, n - 1]$.
2. Calculate $QA = dA * G = (x1, y1)$.
3. Secret key = dA , and public key = QA .

- **Signing:**

1. Choose a random integer k , whose value between $[1, n - 1]$.
2. Calculate $(x1, y1) = k * G$ and $r = x1 \bmod n$, if $r = 0$, then go back to step 1.
3. Calculate $e = Hash(m)$.
4. Calculate $s = k^{-1}(e + dA * r) \pmod n$. If $s = 0$ then go to step 1.

The signature for the message m is (r, s) .

- **Verifying:**

1. Verify that r and s is an integer between $[1, n - 1]$.
2. Calculate $e = Hash(m)$
3. Calculate $w = s^{-1} \pmod n$
4. Calculate $u1 = ew \pmod n$ and $u2 = rw \pmod n$
5. Calculate $(x1, y1) = u1 * G + u2 * QA$
6. Calculate $v = x1 \pmod n$
7. If $v = r$, the signature is valid, otherwise the signature is invalid.



Figure 2: ECDSA.

5 Demo

5.1 Overview

Programming language: C#

Platform: .NET Framework 4.7.2

These resources will be used to develop a secure and efficient chat application that utilizes ECDSA digital signatures for verifying the authenticity and integrity of messages.

5.2 Application concept

The TCP messaging application involves 3 parties: the sender, receiver, and server. When the sender sends a message, it will be signed with ECDSA, and then sent to the server. After receiving the message, the server will send it back to the receiver along with the public key. The receiver will then verify the signature of the message. If the message data has been changed during transmission and does not match the original data, the message will not be verified, and the receiver will be notified that the message has been tampered with.

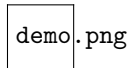



Figure 3: Demo the operation process of the program.

5.3 Deployment

The program development plan includes the following parts:

1. Create a messaging program using the TCP protocol.
2. Set up the ECDSA to create a signature for each outgoing message.
3. Setting up signature verification for messages on the receiver side.
4. Notify the receiver if the message is changed.



deployment.png

Figure 4: Demo the operation process of the program.



6 References