

T.C

GEBZE TECHNICAL UNIVERSITY

COMPUTER ENGINEERING

CSE 470 Fall 2024

Homework 2024

Due date: Dec 27th, 2024

|  |  |
| --- | --- |
| **Name/Surname** | Ziya Kadir TOKLUOĞLU |
| **Number** | 210104004228 |
| **Department** | Computer Engineering |
| **Advisor** | İbrahim SOĞUKPINAR |

**Research: Post quantum cryptography**

**Paper Outline**

1. **Introduction**
2. **Post-Quantum Cryptography**
3. **Description of Selected Algorithms**

* **CRYSTALS-Kyber**
* **CRYSTALS-Dilithium**
* **Falcon**

1. **Comparative Analysis**
2. **Conclusion**

**Introduction**

Quantum computers are a new type of computer that use the rules of quantum physics. They can solve certain problems much faster than regular computers. This is a problem for current cryptographic systems like RSA, Diffie-Hellman, and ECC (Elliptic Curve Cryptography). These systems rely on problems that are hard for regular computers but can be solved by quantum computers using Shor's algorithm.

To solve this problem, new cryptographic methods are being created that can resist attacks from both quantum and regular computers. This is called Post-Quantum Cryptography (**PQC**). The National Institute of Standards and Technology (**NIST**) started a project in 2016 to find and standardize these new methods. In 2022, NIST selected a few algorithms for this purpose.

Among these, two key encapsulation mechanisms (KEMs) or public key encryption algorithms are **CRYSTALS-Kyber**. For digital signatures, **CRYSTALS-Dilithium** and **Falcon** were chosen. CRYSTALS-Kyber, CRYSTALS-Dilithium, and Falcon were winners in 2022

This report explains these selected algorithms. It looks at how they work, their performance, and how secure they are against quantum attacks. The goal is to understand their strengths and weaknesses and how they can help build secure systems in the future. The information is based on NIST’s Post-Quantum Cryptography Standardization project, which is a key resource for this topic.

**Post-Quantum Cryptography**

The need for new cryptographic algorithms arises from the limitations of current systems in the face of quantum computing. Modern cryptographic methods like RSA and ECC are secure because they rely on problems that are difficult for classical computers, such as factoring large numbers or solving discrete logarithms. However, quantum computers, using algorithms like Shor’s, can solve these problems efficiently. This makes current cryptographic systems vulnerable to attacks once quantum computers become powerful enough.

Post-Quantum Cryptography (PQC) aims to develop algorithms that remain secure against both quantum and classical computers. These algorithms are designed based on mathematical problems that are believed to be hard even for quantum computers, such as lattice-based problems, multivariate polynomial equations, or hash-based methods.

The main features of these new algorithms include:

* **Quantum Resistance**: They rely on problems like lattice-based constructions that are resistant to attacks from quantum algorithms like Shor's and Grover's.
* **Efficiency**: Many of the proposed algorithms are designed to be efficient in terms of computation and communication, ensuring they can be integrated into existing systems without significant performance losses.
* **Flexibility**: These algorithms can be used in various cryptographic applications, including key exchange, encryption, and digital signatures.

The NIST Post-Quantum Cryptography Standardization project has identified several promising algorithms that meet these requirements. Among the most notable are CRYSTALS-Kyber and NTRU for key encapsulation and CRYSTALS-Dilithium and Falcon for digital signatures. These algorithms have been carefully evaluated for security, efficiency, and practicality, making them strong candidates for future cryptographic standards.

**Description of Selected Algorithms**

**CRYSTALS-Kyber**

● Kyber.CPAPKE, a public-key encryption scheme designed to encrypt messages of a fixed 32-byte length. This scheme is IND-CPA secure, meaning it's secure against chosen-plaintext attacks, but vulnerable to chosen-ciphertext attacks.

● To enhance security, Kyber applies a modified Fujisaki–Okamoto (FO) transform to Kyber.CPAPKE. This transform results in Kyber.CCAKEM, an IND-CCA2 secure KEM. This means it is resistant to chosen-ciphertext attacks, making it a robust choice for various applications.

|  |  |  |
| --- | --- | --- |
| **Step** | **Notation** | **Explanation** |
| Sample secret vector **s** and error vector **e** | s, e ~ CBDn | Both are sampled from a centered binomial distribution |
| Generate random public matrix **A** | A ~ **Rkxk** | Public matrix is sampled uniformly from Rkxk |
| Compute public key t | T = A.s + e | Matrix-vector multiplication in the ring **Rq** adding noise vector **e**. |
| Output **(t, A)** as public key and  **s** as secret key |

**Key Generation**

**Encryption**

|  |  |  |
| --- | --- | --- |
| **Step** | **Notation** | Explanation |
| Input message **m**, public key  **(t, A),** and random **r.** | |  | | --- | | M ∈ Rq ​, r∼CBDη​, e1​,e2​∼CBDη​ |  |  | | --- | |  | | Message m, random vector rr, and error polynomials e1 and e2 are prepared. |
| Compute intermediate ciphertext components | **u=A⊤⋅r+e1** | Multiply **A** (transposed) with  r and add noise e1. |
| Compress components for transmission. | C = (Compressq ​(u), Compressq(v)) | Ciphertext is a compressed form of u and v. |

**Decryption**

|  |  |  |
| --- | --- | --- |
| Input ciphertext c= (c1 , c2) and secret key s | c1​, c2​∈Rq ​, s | Ciphertext components c1, c2 and secret key s are inputs |
| Decompress  u and v. | U = Decompressq ​(c1) ,  v = Decompressq ​(c2​) | Decompress the ciphertext components |
| Reconstruct the message m. | m=Compressq ​(v − sT ⋅u) | Subtract sT u from v then compress the result to recover m. |

|  |  |
| --- | --- |
| Symbol | Meaning |
| A | Public uniformly random matrix in Rk. |
| s | Secret vector in Rk, sampled from a binomial distribution. |
| e | Error vector Rk, sampled from a binomial distribution. |
| t | Public key component: t = A. s + e |
| r | Random vector in Rk, sampled from a binomial distribution. |
| e1 ,e2 | Error polynomials, sampled from a binomial distribution. |
| u | Intermediate ciphertext component: u = AT . r + e1. |
| v | Final ciphertext component: v = tT. r + e2 + Decompressq(m). |
| c | Ciphertext: Compressq  (u), Compressq (v)) |
| Compressq | Function to reduce the size of ciphertext components. |
| Decompressq | Function to expand the compressed ciphertext components. |

**Notation**

CRYSTALS-Kyber is a quantum-resistant cryptographic algorithm that relies on the Module Learning with Errors (MLWE) problem, which is considered difficult even for quantum computers. It operates as a Key Encapsulation Mechanism (KEM), meaning it facilitates secure key exchange between parties. Kyber has been designed to ensure robust security through a two-stage approach. Initially, it uses Kyber.CPAPKE, which provides IND-CPA security (resistant to chosen-plaintext attacks). The second stage employs the Fujisaki-Okamoto transform, enhancing the scheme to IND-CCA2 security, making it resistant to chosen-ciphertext attacks. This layered security design is essential for providing strong protection in cryptographic protocols where adaptive attackers might attempt to manipulate ciphertexts.

Kyber is highly efficient, thanks in part to its use of Number Theoretic Transforms (NTTs) for polynomial arithmetic. These transformations allow for fast computations, which are crucial for large-scale cryptographic applications. The algorithm operates on module lattices, which offer a balance between security and efficiency. Kyber has been parameterized in three versions—Kyber512, Kyber768, and Kyber1024—to accommodate varying security levels. The key length for Kyber512 is 1632 bytes, and it scales with the version, with Kyber1024 having a key length of 3168 bytes. Additionally, the ciphertext size for Kyber512 is 768 bytes, which is manageable for modern cryptographic systems. Kyber’s ability to scale efficiently makes it adaptable for different use cases, such as secure key exchange in post-quantum environments.

Performance is a critical aspect of Kyber’s design. The use of NTTs ensures that polynomial multiplications, which form the core of the encryption and decryption operations, are handled very efficiently. On modern processors, such as Intel’s AVX2 optimized implementation, Kyber can perform encryption and decryption at impressive speeds. For example, the AVX2 implementation of Kyber512 achieves 45200 cycles for encryption and 34572 cycles for decryption, making it suitable for use in environments with high throughput requirements. The algorithm’s key generation process is also efficient, reducing overhead without sacrificing security. Furthermore, Kyber’s use of symmetric primitives like SHA3-256 and SHAKE-128 ensures that the cryptographic operations remain fast while maintaining a high level of security. This combination of strong security features and optimized performance positions.

**CRYSTALS-Dilithium**

**Key Generation**

|  |  |  |
| --- | --- | --- |
| **Step** | **Notation** | **Explanation** |
| Generate matrix A and secret vectors s1 , s2 | A∈ Rkxl , s1, s2 ~ Sηl ,Sηk​ | A is a uniformly random matrix; s1 , s2 are small polynomials. |
| Compute t = A. s1+s2 | T ∈ Rk | Public key polynomial vector **t** |
| Extract higher-order bits of **t** | (t1​,t0​) = Power2Round(t,d) | Reduces size of the public key. |

**Signing**

|  |  |  |
| --- | --- | --- |
| **Step** | **Notation** | **Explanation** |
| Expand randomness p′ from seed | ρ′=CRH(K∥μ) | Ensures deterministic signing or allows randomized mode. |
| Generate random polynomial **y** | y ∈ ~Sγl  **​** | Masking vector with bounded coefficients. |
| Compute intermediate vector w | w = A . y | Multiply A by y |
| Extract higher-order bits of w | w1​=HighBits(w,2γ2​) | Captures significant components of w |
| Compute challenge polynomial c | c=SampleInBall(H(μ∥w1​)) | Challenge derived from hash of message digest and w1 |
| Compute potential signature z. | z=y+c⋅s1​ | Combines randomness y and secret s1​. |
| Verify signature conditions. | ∥z∥∞​<γ1 −β,∥LowBits(w−c⋅s2​,2γ2​)∥∞​<γ2​−β | Ensures signature correctness and security |
| Generate hint h for verification | h=MakeHint(−c⋅t0​,w−c⋅s2​+c⋅t0​,2γ2​) | Encodes carries caused by missing low-order bits of t. |

**Verification**

|  |  |  |
| --- | --- | --- |
| **Step** | **Notation** | **Explanation** |
| Recompute challenge c. | C = SampleInBall(H(μ∥w1′​)) | Verifies hash consistency with message and signature. |
| Use hint to recover w1 | W1′​= UseHint(h,A⋅z−c⋅t1​⋅2d,2γ2​) | Reconstructs w1’ for verification |
| Verify signature validity. | ∥z∥∞​<γ1​−β,c=H(μ∥w1′​) | Ensures z bounds and hash consistency. |

**Notation Summary**

|  |  |
| --- | --- |
| **Symbol** | **Meaning** |
| Rq | Polynomial ring Zq​[X]/(Xn+1), where q is a prime. |
| A | Public uniformly random matric in Rkxl |
| s1 , s2 | Secret vectors with small coefficients. |
| t , t1 , t0 | Public key components: t = A . s1 + s2 |
| z | Signature component |
| γ1​,γ2​,β | Parameters controlling signature size and security. |

CRYSTALS-Dilithium is a post-quantum digital signature algorithm that relies on lattice-based cryptography, specifically using the hardness of finding short vectors in lattices. This algorithm is designed to provide secure and efficient signatures resistant to quantum attacks, which could potentially break traditional public-key cryptosystems like RSA and ECC. Dilithium is part of the CRYSTALS (Cryptographic Suite for Algebraic Lattices) project and is designed to be simple to implement securely. It avoids using complex distributions for generating randomness, instead opting for uniform sampling to prevent implementation vulnerabilities. The algorithm uses polynomial rings for algebraic operations, with the signing and verification processes involving multiplication of polynomials in these rings, making it efficient and scalable for post-quantum cryptography.

Dilithium operates with a key generation, signing, and verification process that balances security with efficiency. The key generation step involves the creation of a matrix and secret vectors, followed by the computation of a public key. The size of the public key and signature is smaller than many other lattice-based schemes, which is essential for practical usage where transmission of these elements is necessary. The signing procedure involves several rounds of polynomial manipulation and rejection sampling to ensure that the signature is both valid and secure. The rejection sampling process ensures that the signature does not leak any information about the secret key, while the use of efficient hash functions, such as SHAKE-256, ensures that the process remains fast. Dilithium supports both deterministic and randomized signatures, allowing flexibility depending on the security requirements of the application. The choice between deterministic and randomized signatures primarily depends on the potential vulnerability to side-channel attacks.

In terms of efficiency, Dilithium is optimized for high performance. It uses Number Theoretic Transforms (NTT) for fast polynomial multiplication, which significantly reduces the computational overhead associated with these operations. This optimization is crucial for handling large key sizes and ensuring the algorithm's practicality in real-world applications. The algorithm's performance can be further enhanced through hardware acceleration techniques like AVX2, with implementations showing notable speed improvements. The key sizes for different security levels in Dilithium range from approximately 1312 bytes for NIST Level 2 to 2592 bytes for NIST Level 5, while signature sizes also scale accordingly. Despite the larger key and signature sizes at higher security levels, Dilithium remains one of the most efficient lattice-based signature schemes, making it a strong contender in post-quantum cryptography competitions. Additionally, the modular nature of Dilithium allows for straightforward adaptation to different security levels, making it a flexible solution for future cryptographic needs.

**Falcon**

**Key Generation**

|  |  |  |
| --- | --- | --- |
| **Step** | **Notation** | **Explanation** |
| Generate private key polynomials f, g, F, G satisfying the NTRU equation. | fG – gF = q mod φ, f, g, F, G ∈  Z[x]/(φ) | Private key includes small polynomials f, g, F, G that satisfy the NTRU equation. |
| Compute the public key h. | h = g/f mod q | Public key is derived as h, using modular arithmetic. |

**Signature Generation**

|  |  |  |
| --- | --- | --- |
| **Step** | **Notation** | **Explanation** |
| Hash the message m with a random salt r to create a hash value c. | c = H(r∥m) modq | Combines the message m and salt **r** to generate **c**. |
| Find a short vector (s1 , s2) such that s1 + s2h = c | S1 ​,s2 ​∈Z[x]/(φ), s1+s2h=c mod q | Use lattice techniques to find (s1 , s2) as short polynomials. |
| Output the signature as s2 | Signature = s2 | Only s2 is required for the signature |

**Signature Verification**

|  |  |
| --- | --- |
| **Symbol** | **Meaning** |
| φ | Cyclotomic polynomial xn + 1, where n= 2k |
| q | Prime modulus, typically 12289 |
| f, g, F, G | Private key polynomials satisfying fG – gF = q mod φ |
| h | Public key polynomial h = g /f mod q |
| c | Hash values derived from the message m and random salt r. |
| **s1 , s2** | Signature components where s1 + s2h = c mod q |
| **||(s1 , s2)||** | Norm of the signature vector, must be less than a defined bound β |

Falcon is a post-quantum digital signature scheme designed to offer both strong security and efficiency. It is based on lattice cryptography, specifically utilizing **NTRU lattices** to ensure compactness and performance. Falcon’s key generation, signature generation, and verification are all optimized to be efficient while maintaining security against quantum computing threats. The signature generation uses a **trapdoor sampler** known as **Fast Fourier Sampling**, which enables efficient computation. This fast sampling, combined with **NTRU lattices**, allows Falcon to generate signatures that are both small in size and secure. The use of NTRU lattices also ensures that the public key size is minimized, which is important for post-quantum cryptography where key sizes often grow significantly compared to classical systems. This compactness makes Falcon particularly appealing for environments with constrained resources, such as embedded devices.

The security of Falcon is based on the hardness of the **Shortest Vector Problem (SVP)** in lattice-based cryptography. By using **NTRU lattices** and the **GPV framework**, Falcon benefits from strong security proofs both in the **classical random oracle model (ROM)** and the **quantum random oracle model (QROM)**. This makes Falcon resilient not only to classical attacks but also to quantum attacks, offering a future-proof solution for digital signatures. The signing process in Falcon relies on polynomials in a cyclotomic ring, and it uses Gaussian sampling to ensure the generated signatures are short and secure. Falcon’s structure is highly modular, meaning that it can be adapted to different cryptographic settings by adjusting its underlying lattice structure or sampling techniques, ensuring both flexibility and security.

In terms of efficiency, Falcon is optimized for fast signing and verification. Signature verification, in particular, is very fast, requiring only polynomial-time operations in the **Fast Fourier Transform (FFT)** domain. This efficiency is crucial for practical deployment, as it allows Falcon to handle large volumes of signatures without significant performance degradation. For instance, Falcon can generate more than 1000 signatures per second on moderate hardware. Key sizes for Falcon are also relatively small, with the public key for **Falcon-512** being 704 bytes and signatures being 384 bytes, which is significantly smaller than many other post-quantum signature schemes. However, Falcon does require careful implementation, particularly in the key generation and sampling phases, and may pose challenges in environments with limited floating-point precision. Despite these challenges, Falcon remains one of the most efficient and compact signature schemes for post-quantum cryptography, making it a strong candidate for future cryptographic standards.

**Comparative Analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | CRYSTALS-Kyber | CRYSTALS-Dilithium | Falcon |
| Cryptographic Purpose | Key Encapsulation Mechanism (KEM) | Digital Signature | Digital Signature |
| Underlying Hard Problem | Module Learning with Errors (MLWE) | Lattice-based (short integer solution problem) | NTRU lattice-based (Shortest Vector Problem - SVP) |
| Key Sizes | Kyber512: Public key = 800 bytes, Ciphertext = 768 bytes | Level 2: Public key = 1312 bytes, Signature = 2420 bytes | Falcon-512: Public key = 704 bytes, Signature = 384 bytes |
| Efficiency | High: Optimized with Number Theoretic Transform (NTT) for fast polynomial arithmetic. | Moderate: Efficient polynomial multiplication using NTT and flexible rejection sampling. | Very High: Fast Fourier Sampling (FFS) and compact signature/key sizes enable high throughput. |
| Security | IND-CPA and IND-CCA2 secure through Fujisaki–Okamoto transform. | Strong against quantum attacks with efficient rejection sampling. | Strong against quantum and classical attacks with security proofs in ROM and QROM. |
| Performance | Suitable for high-throughput environments, with fast encryption/decryption cycles. | Moderate signing and verification speed; deterministic or randomized signatures possible. | Extremely fast signing and verification, suitable for resource-constrained environments. |
| Compactness | Moderate: Ciphertext and key sizes are relatively small for a lattice-based KEM. | Moderate: Signature size is larger than Falcon but balances security and efficiency. | High: Compact key and signature sizes make it ideal for embedded systems and constrained devices. |
| Implementation Complexity | Low: Simple and efficient to implement. | Moderate: Avoids complex randomness distributions for ease of secure implementation. | High: Requires careful handling of floating-point precision during key generation and sampling. |
| Applications | Secure key exchange, TLS, VPNs, and post-quantum cryptographic systems. | Digital signatures for authentication in post-quantum systems. | Digital signatures in constrained environments like IoT and embedded systems. |

CRYSTALS-Kyber is highly efficient for secure key exchange and can be seamlessly integrated into existing cryptographic protocols. Its scalability across different security levels makes it versatile for various applications.

CRYSTALS-Dilithium provides robust digital signatures with efficient operations and a balance between security and performance. However, its larger signature size may present a drawback in bandwidth-sensitive applications.

Falcon stands out for its compactness and speed, making it particularly well-suited for environments with limited computational resources. Its reliance on floating-point arithmetic, however, introduces challenges in secure implementation

**Conclusion**

The analysis of the CRYSTALS-Kyber, CRYSTALS-Dilithium, and Falcon algorithms highlights their suitability for post-quantum cryptography, each addressing different cryptographic needs with unique strengths and trade-offs. CRYSTALS-Kyber excels as a Key Encapsulation Mechanism (KEM) due to its efficient polynomial arithmetic and robust security features, making it a strong candidate for secure key exchange. CRYSTALS-Dilithium, as a digital signature algorithm, offers a balance between simplicity, efficiency, and resistance to quantum attacks, while also minimizing the risk of side-channel vulnerabilities. Falcon, on the other hand, prioritizes compactness and high performance, standing out for its small signature and public key sizes, making it ideal for resource-constrained environments.

Together, these algorithms demonstrate the progress in creating cryptographic solutions resistant to both classical and quantum threats. Their adoption will likely depend on specific application requirements, such as efficiency, key size, or system constraints. Moving forward, further research and implementation studies will play a critical role in refining these algorithms for widespread deployment and ensuring secure communication in the quantum era