석 사 학 위 논 문
Master's Thesis

양자 내성 서명을 이용한 탈중앙화 공개키 기반 구조의 설계 및 분석

Design and Analysis of Decentralized Public Key Infrastructure with Quantum-resistant Signatures

2018

안 형 철 (安 亨 哲 An, Hyeongcheol)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

석 사 학 위 논 문

양자 내성 서명을 이용한 탈중앙화 공개키 기반 구조의 설계 및 분석

2018

안 형 철

한 국 과 학 기 술 원

전산학부 (정보보호대학원)

양자 내성 서명을 이용한
탈중앙화 공개키 기반 구조의 설계 및 분석

안 형 철

위 논문은 한국과학기술원 석사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2018년 6월 12일

심사위원장 김 광 조 (인)

심 사 위 원 김 용 대 (인)

심 사 위 원 강 병 훈 (인)

Design and Analysis of Decentralized Public Key Infrastructure with Quantum-resistant Signatures

Hyeongcheol An

Advisor: Kwangjo Kim

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Master of Science in Computer Science (Information Security)

Daejeon, Korea
June 12, 2018

Approved by

Kwangjo Kim
Professor of Computer Science

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

MIS
20164355

안형철. 양자 내성 서명을 이용한 탈중앙화 공개키 기반 구조의 설계 및 분석. 전산학부 (정보보호대학원) . 2018년. 50+iv 쪽. 지도교수: 김광조. (영문 논문)
Hyeongcheol An. Design and Analysis of Decentralized Public Key Infrastructure with Quantum-resistant Signatures. School of Computing (Graduate School of Information Security) . 2018. 50+iv pages. Advisor: Kwangjo Kim. (Text in English)

초 록

블록체인은 2008년 비트코인(Bitcoin)에서 처음으로 제안되었으며, 분산형 데이터베이스 기술이다. 현재 키 관리 시스템 중 하나인 공개키 기반 구조(Public Key Infrastructure, PKI) 기술은 중앙집중형 구조로 single point failure의 가능성이 존재한다. 또한 대표적인 암호화폐인 비트코인과 이더리움에서 사용되는 전자서명 알고리즘인 ECDSA는 Shor 알고리즘에 따라 양자 컴퓨터 공격에 취약하다는 문제점이 존재한다. 이에, 본 석사 논문에서는 양자 내성 암호 기술을 적용한 블록체인 기반 키 관리 시스템에 대하여 제안한다. 분산형 구조인 블록체인을 기반으로 설계하여 단일 지점 오류에 안전하다. 또한, 대표적인 양자 내성 암호인 래티스 기반 전자서명인 GLP 전자서명을 사용하기 때문에 양자컴퓨터의 공격에도 안전하며, 장기간 안전성을 보장한다.

핵 심 낱 말 포스트 양자 암호, 블록체인, 공개키 기반 구조, 키 교환 프로토콜, 격자 문제

Abstract

The blockchain technique was first proposed by Satoshi Nakamoto in 2008. It is used for a distributed database technology. Public Key Infrastructure(PKI) system, which is one of the key management systems, is a centralized system. There is a possibility of a single point of failure in the currently used centralized PKI system. Classical digital signature algorithm; ECDSA has used the well-known cryptocurrencies, such as Bitcoin and Ethereum. Using the Shor's algorithm, ECDSA can be broken by the quantum computing attack. In this thesis, we propose a blockchain-based key management system using quantum-resistant cryptography, since we use a GLP digital signature scheme, which is a lattice-based mathematical problem. It is secure against the quantum adversary and ensures long-term safety. Besides, we design a decentralized blockchain structure, and it is secure for the single point of failure.

Keywords Post-quantum Cryptography, Blockchain, Public Key Infrastructure, Key Exchange Protocol, lattice problem

Contents

List of Tables

List of Figures

Chapter 1. Introduction

1.1 Overview of Post Quantum Cryptography

IBM developed a quantum computer with 5-qubit in 2016 and a new quantum computer with 50-qubit in Nov. 2017. The research team of IBM has developed a quantum computer that allows the public to simulate a quantum computer through an IBM Q Experience [1]. Therefore, the emergence of the quantum computer is not theoretical but becomes practical. Researcher and engineers predict that within the next twenty or so years, sufficiently large quantum computers will be built to break public key cryptosystems. If universal quantum computers can be feasible, public key cryptosystems whose difficulties are based on the number theoretic problem will be broken in a polynomial time.

Public key cryptosystems, such as Diffie-Hellman (DH) key exchange protocol and RSA, are based on the difficulty of Discrete Logarithm Problem (DLP), Elliptic Curve DLP (ECDLP), and Integer Factorization Problem (IFP). However, DLP and IFP can be solved within the polynomial time by Shor's algorithm [2] using the quantum computer. Symmetric key cryptosystems such as the Advanced Encryption Standards (AES) and Data Encryption Standard (DES) can be solved using Grover's algorithm [3]. Grover's algorithm can be used in the data search problem. In classical computers, the adversary can search in the database as $O(2^n)$ complexity. Using the quantum computer, the complexity of the data search problem reduces $O(\sqrt{2^n})$. Therefore, we need a secure public key cryptosystem against the quantum adversary. Post Quantum Cryptography (PQC) plays an important roles in building a secure cryptosystem against both classical and quantum adversaries. PQC primitives are based on mathematical hard problems which are lattice-based, code-based, hash-based, multivariate-based, and supersingular isogeny elliptic curves. Lattice-based cryptography is used for the encryption scheme, signature, and key exchange protocol. The National Institute of Standards and Technology (NIST) contested a public PQC cryptographic algorithm project until November 30, 2017, to select a secure cryptographic algorithm against the quantum adversary [4].

1.2 Motivation

A public-key cryptosystem needs Public Key Infrastructure (PKI), which guarantees the integrity of all user's public keys by binding them with its owner. The currently used PKI system is X.509 v3 [5] as recommended by the international standards. However, the X.509 PKI system has disadvantages such as centralization, single point failure, and fully trusted Certificate Authority (CA). CA is a trusted third party whose signature on the certificate guarantees the authenticity of the public key bound to each entity. If CA is not online called single point failure, the client cannot store or revoke their public keys. Therefore, the currently used centralized PKI system has problems with availability, due to the centralized CA. Due to the centralized CA, we fully trust CA server. Thus, we need decentralized PKI system to solve disadvantages of the centralized PKI system.

Currently, Web of Trust (WoT) [6] approach is achieved decentralized public key infrastructure called PGP [7] using the trustworthy users. However, PGP makes that it is difficult for new or remote users to join the network since existing member of WoT must meet with the new user in person to have his identity verified and public key signed for the first time. It is difficult to revoke the public key since member must push the revocation list. Therefore, there is a disadvantage that the public key cannot be canceled immediately.

The most famous cryptocurrency, Bitcoin [8] is the first decentralized virtual-currency. Bitcoin uses blockchain, which is a transaction database (or distributed ledger) shared by all peer nodes. With the transaction of the blockchain, anyone can find each block of information in the transaction history. A peer-to-peer (P2P), that is one of a decentralized networks, is a distributed system between peers. Each peer has equally the same privilege in their network. P2P does not have the concept of client or server. Therefore, each peer node operates both client and server on their network at the same time, since the blockchain technique is decentralized.

We focus on the lattice-based cryptography that is based on the mathematical hard problem such as Ring Learning with Errors (Ring-LWE) problem. Lattice-based cryptography can be used not only for encryption scheme but also for the key exchange protocol and signature. We use the GLP [9] signature scheme that is based on the ring-LWE problem. GLP signature scheme is a simple and efficient quantum-resistant signature algorithm.

In this thesis, we propose QChain, which is a quantum-resistant decentralized PKI system. To construct QChain, we combine blockchain and lattice-based cryptography which is one of PQC primitive. QChain is a practical method for managing public key encryption. To construct a quantum secure PKI, we use the lattice-based GLP signature scheme. We also compare the currently used X.509 v3 PKI system and our QChain from the point of connection, non-repudiation, revocation, scalability, trust model, and security level.

1.3 Organization

The rest of this thesis is organized as follows: Chapter 2 describes notations and definitions as preliminaries. The related work which consists of lattice-based cryptography, the blockchain, and public key infrastructure is described in Chapter 3. In Chapter 4, we describe our proposed scheme. The security requirement, generic attack, and feature analysis are presented in Chapter 5. Finally, the future work and concluding remarks are discussed in Chapter 6, respectively.

Chapter 2. Preliminaries

In this chapter, we state the notations, definitions, and lattice-based mathematical hard problems used in this thesis. After that, consensus algorithm such as proof-of-work and Byzantine fault tolerance is described briefly.

2.1 Definitions

Bellare and Rogaway defined the public key encryption scheme [10]. We briefly restate the definitions as below:

Definition 2.1.1. (Public Key Encryption Scheme). A public key encryption scheme is a tuple of Probabilistic Polynomial-Time (PPT) algorithms (Gen, Enc, Dec) satisfying the following:

- The key generation algorithm Gen() takes input a security parameter 1^λ and outputs a pair of keys $(pk, privK)$. These are called the public key and the private key, respectively. We assume that pk and $privK$ each have length at least λ, and that λ can be determined from pk and $privK$.
- The encryption algorithm Enc() takes as input a public key pk and a message m. It outputs a ciphertext c, and we write this as $c \leftarrow \mathsf{Enc}_{pk}(m)$.
- The decryption algorithm Dec() takes as input a private key $privK$ and a ciphertext c. It outputs a message m. We write this as $m = \mathsf{Dec}_{privK}(m)$.

We require that for every λ, every $(pk, privK)$ output by $\mathsf{Gen}(1^\lambda)$, and every message m, it holds that

$$\mathsf{Dec}_{privK}(\mathsf{Enc}_{pk}(m)) = m$$

Bellare and Rogaway also defined the digital signature scheme [10]. We briefly restate the definitions as below:

Definition 2.1.2. (Digital Signature Scheme). A digital signature scheme is a tuple of Probabilistic Polynomial-Time (PPT) algorithms (Gen, Sign, Ver) satisfying the following:

- The key generation algorithm Gen() takes input a security parameter 1^λ and outputs a pair of keys $(pk, privK)$. These are called the public key and the private key, respectively. We assume that pk and $privK$ each have length at least λ, and that λ can be determined from pk and $privK$.
- The signing algorithm Sign() takes as input a private key $privK$ and a message m. It outputs a signature σ, and we write this as $\sigma \leftarrow \mathsf{Sign}_{privK}(m)$.
- The deterministic verification algorithm Ver() takes as input a public key pk, a message m, and a signature σ. It outputs a bit b, with $b = 1$ meaning `VALID` and $b = 0$ meaning `INVALID`. We write this as $b := \mathsf{Ver}_{pk}(m, \sigma)$.

We require that for every λ, every $(pk, privK)$ output by $\mathsf{Gen}(1^\lambda)$, and every message m, it holds that

$$\mathsf{Ver}_{pk}(m, \mathsf{Sign}_{privK}(m)) = 1$$

2.2 Notations

Following notations are used in this thesis. Table 2.1 describes the notations.

Table 2.1: Notations and Variables

Variables	Description
$privK$	private key
sk	secret key
pk	public key
m	plaintext
c	ciphertext
e	Gaussian error
1^λ	security parameter
$\oplus$	bitwise exclusive-or operator
$\|\|$	concatenation operator
χ_σ	Gaussian distribution with standard deviation σ
σ	standard deviation
$H()$	cryptographic hash function
$Sign()$	cryptographic digital signature sign function
$Verify()$	cryptographic digital signature verify function
$\mathsf{NTT}()$	number theoretic transformation
$\mathsf{NTT}^{-1}()$	inverse number theoretic transformation

2.3 Lattice-based Mathematical Hard Problems

LWE problem is introduced by Regev [11] in 2009. LWE is a quantum-resistant mathematical hard problem against the quantum adversary.

Definition 2.3.1. (LWE Distribution). LWE distribution $A_{\mathbf{s},\chi} \in \mathbb{Z}_q^n \times \mathbb{Z}_q^n$, for a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ and choose uniformly random $\mathbf{a} \in \mathbb{Z}_q^n$, and choosing $e \leftarrow \chi$. and outputting;

$$(\mathbf{a}, b = \langle \mathbf{s}, \mathbf{a} \rangle + e \mod q)$$

Error distribution χ over $\mathbb{Z}$ is usually used for Gaussian distribution or binomial distribution. LWE problem has two kinds of version such as search and decision. In cryptography, we use decision version

LWE problem. Decision LWE problem is given m independent samples $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^n$. $A_{s,\chi}$ for a uniformly random $s \in \mathbb{Z}_q^n$ or uniform distribution, distinguish which chooses the sample.

Ring-LWE problem is introduced by Lyubashevsky *et al.* [12] in 2010. Ring-LWE is also a quantum-resistant mathematical hard problem against the quantum adversary.

Definition 2.3.2. (Ring-LWE Distribution). For a ring $\mathcal{R}$ of degree n over $\mathbb{Z}$, and defining quotient ring $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Ring-LWE distribution $A_{s,\chi} \in \mathcal{R}_q \times \mathcal{R}_q$, secret vector $s \in \mathcal{R}_q$ and choose uniformly random $a \in \mathcal{R}_q$, and choosing $e \leftarrow \chi$. and outputting;

$$(a, b = s \cdot a + e \mod q)$$

Error distribution χ over $\mathbb{Z}$ is usually used for Gaussian distribution or binomial distribution. The ring-LWE problem has two kinds of version such as search and decision. In cryptography, we use decision version ring-LWE problem. Decision ring-LWE problem is given m independent samples $(a_i, b_i) \in \mathcal{R}_q \times \mathcal{R}_q$. $s \in A_{s,\chi}$ for a uniformly random $\mathcal{R}_q$ or uniform distribution, distinguish which chooses the sample.

Module-LWE problem is introduced by Langlois *et al.* [13] in 2015. Module-LWE is also a quantum-resistant mathematical hard problem against the quantum adversary.

Definition 2.3.3. (Module-LWE Distribution). For a ring $\mathcal{R}$ of degree n over $\mathbb{Z}$, and defining quotient ring $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Error distribution χ over $\mathbb{Z}$ is usually used for Gaussian distribution or binomial distribution. Module-LWE distribution $A_{m,k,\eta} \in \mathcal{R}_q^{m \times k} \times \mathcal{R}_q^m$, secret vector $s \in \beta_\eta^k$ and choose uniformly random $a_i \in \mathcal{R}_q^k$, and choosing $e_i \leftarrow \beta_\eta$. and outputting;

$$(a, b_i = a_i^T \cdot s + e_i \mod q)$$

The module-LWE problem has two kinds of version such as search and decision. In cryptography, we use decision version module-LWE problem. Decision module-LWE problem is given m independent samples $(a_i, b_i) \in \mathcal{R}_q^k \times \mathcal{R}_q$. $s \in \beta_\eta^k$ for a uniformly random $\mathcal{R}_q$ or uniform distribution, distinguish which chooses the sample.

2.4 Consensus Algorithm

In this section, we describe the consensus algorithm, which uses blockchain, such as Proof-of-Work (PoW) and Byzantine Fault Tolerance (BFT). Consensus algorithm decides whose block to add into blockchain.

2.4.1 Hashcash

PoW and PoS algorithm is based on hash chain. Lamport suggested a method of user authentication method called hash chain [14] using a hash function. Then, Back suggested the hashcash [15] to prevent denial of service (DoS) attack in 2002. However, consensus algorithms such as PoW and PoS are based on hashcash method.

Definition 2.4.1. (Hashcash). To demonstrate work on x, find y such that

$$H(x, y) < z$$

where, $H()$: hash function, y: nonce, and z: target hash value.

If target hash value z is small, the prover needs more computing power to find nonce y. Therefore, we can modify the difficulty level by changing target hash value z.

2.4.2 Proof-of-Work

A proof of work is a piece of data which is difficult time or power-consuming to produce but easy for others to verify and which satisfies specific requirements. For well-known cryptocurrency Bitcoin, they use the PoW method based on the hashcash problem.

Definition 2.4.2. (Bitcoin Proof-of-Work). Find nonce n such that

$$H(\ n\ ||\ H_{prev}()\ ||\ Blockdata\ ) < z$$

where, $H()$: SHA-256 hash function, n: nonce, and z: target hash value.

In PoW of Bitcoin, the difficulty level is adjusted once every two weeks. The meaning of the adjusting difficulty level is to change the value of the target hash value mentioned in Definition 2.4.2. The Bitcoin difficulty adjustment equation is as follows:

$$Diff_{new} = Diff_{old} \times \frac{20160}{t}$$

where, $Diff_{new}$: new difficulty level of Bitcoin, $Diff_{old}$: previous difficulty level of Bitcoin, and t: total mining time of 2,016 blocks (min).

2.4.3 Byzantine Fault Tolerance

Lamport *et al.* introduced the Byzantine Generals Problem(BGP) [16] in 1982. BGP assumes a situation where the generals of each unit communicate with each other through a messenger and plan

an attack together while the various units of the Byzantine army are trying to attack the enemy city. In this situation, some of the generals may have mixed traitors. At that time, despite the existence of the traitor, how many generals must be for the commanders to plan the same attack.

Byzantine Fault Tolerance (BFT) algorithm is based on the BGP and used for the fault-tolerant computing system. The general BFT algorithm has five-phase, which consists of the request, pre-prepare, prepare, commit, and reply. The BFT consensus algorithm is a method by which a leader is elected, that leader creates a block, propagates it to the verifier, and the verifiers' vote. The most important feature of BFT is that it requires 2/3 or more consent among all voters to generate the block.

Table 2.2 compare PoW and BFT consensus algorithm. PoW can be applied in the public blockchain, and BFT can be applied in the private or consortium blockchain. BFT consensus algorithm has the advantage that there is no waste of energy and it is possible to agree immediately by voting through the stake. Therefore, power consumption is low, and a leader must exist. However, as compared with PoW, it is limited in scalability and has a high latency because it has to propagate block status immediately to all blocks.

Table 2.2: Comparison with PoW and BFT consensus algorithm

Consensus Algorithm	**PoW**	**BFT**
Operating member	Anyone	Specific operator
Scalability	Unlimited	limited
Performance(transaction)	low	high
Performance(latency)	high	low
Power Consumption	high	low

Chapter 3. Related Works

In this chapter, we introduce the related works of lattice-based cryptography which is one of the most popular PQC primitives. Then, we briefly describe the overview of the blockchain. Finally, public key infrastructure standards and previous approach of blockchain-based PKI are presented.

3.1 Lattice-based Cryptography

In this section, the well-known lattice-based mathematical hard problem such as Learning with Errors (LWE), Ring Learning with Errors (Ring-LWE), and Module Learning with Errors (Module-LWE) problems will be described in brief.

Lattice-based cryptography is one of the most popular PQC primitives. Therefore, lattice-based cryptography is secure against the quantum adversary. There are many kinds of lattice-based cryptographic primitives such as LWE, ring-LWE, module-LWE, Learning with Rounding (LWR), and so on. Lattice-based cryptography can be used not only for encryption scheme but also for key exchange protocol and digital signature scheme. We will describe LWE, ring-LWE, and module-LWE problems in brief.

3.1.1 Key Exchange Protocols

We focus on lattice-based key exchange protocols. OQS project [17] is an open source and a consist of 9 PQC cryptography. The OQS project is based on three kinds of PQC primitives such as lattice-based, code-based, and supersingular isogeny elliptic curve. Key exchange protocols such as Frodo, BCNS, NewHope, MSrln, Kyber, and NTRU are based on lattice-based scheme. IQC and MSR SIDH are based on supersingular isogeny elliptic curve scheme. Besides, McBits is based on code-based scheme. Table 3.1 describes algorithms of liboqs. To merge with OpenSSL, they implement same header file form in OpenSSL.

We will describe lattice-based key exchange protocol in detail.

Table 3.1: Algorithms of liboqs

Primitive		Protocol
Lattice-based	LWE	Frodo
	Ring-LWE	BCNS
		NewHope
		MSrln
	Module-LWE	Kyber
	NTRU	
Supersingular Elliptic Curve	SIDH	IQC Reference
		MSR SIDH
Code-based	Error-correcting codes	McBits

NewHope

Alkim *et al.* [18] proposed ring-LWE key exchange protocol called NewHope in 2016. Protocol 1 describes key exchange protocol of NewHope. To compute NewHope, we define $\mathsf{HelpRec}()$ and $\mathsf{Rec}()$ functions.

Protocol 1: NewHope

Alice		Bob
$\mathsf{seed} \xleftarrow{\$} \{0,1\}^{256}$		
$a \leftarrow \mathsf{Parse}(\mathsf{SHAKE\text{-}128}(\mathsf{seed}))$		
$s, e, \xleftarrow{\$} \Psi_{16}^n$		$s', e', e'' \xleftarrow{\$} \Psi_{16}^n$
	$\xrightarrow{(b,\mathsf{seed})}$	$a \leftarrow \mathsf{Parse}(\mathsf{SHAKE\text{-}128}(\mathsf{seed}))$
		$u \leftarrow as' + e'$
		$v \leftarrow bs' + e''$
$v' \leftarrow us$	$\xleftarrow{(u,r)}$	$r \xleftarrow{\$} \mathsf{HelpRec}(v)$
$\nu \leftarrow \mathsf{Rec}(v', r)$		$\nu \leftarrow \mathsf{Rec}(v, r)$
$\mu \leftarrow \mathsf{SHA3\text{-}256}(\nu)$		$\mu \leftarrow \mathsf{SHA3\text{-}256}(\nu)$

Let $\mathsf{CVP}_{\hat{D}_4}(\mathbf{x} \in \mathbb{R}^4)$ is that an integer vector $\mathbf{z}$ such that is a closest vector to $\mathbf{x} : \mathbf{x} - \mathbf{Bz} \in \mathcal{V}$. The $\mathsf{HelpRec}(\mathbf{x}; b)$ is defined as follows:

$$\mathsf{HelpRec}(\mathbf{x}; b) = \mathsf{CVP}_{\hat{D}_4}\Big(\frac{2^r}{q}(\mathbf{x} + b\mathbf{g})\Big) \mod 2^r$$

where $b \in \{0, 1\}$ is uniformly chosen random bit.

The $\mathsf{Decode}(\mathbf{x} \in \mathbb{R}^4/\mathbb{Z}^4)$ is that a bit k such that $k\mathbf{g}$ is a closest vector to $\mathbf{x} + \mathbb{Z}^4 : \mathbf{x} - k\mathbf{g} \in \mathcal{V} + \mathbb{Z}^4$.

The $\mathsf{Rec}(\mathbf{x}, \mathbf{r})$ is defined as follows:

$$\mathsf{Rec}(\mathbf{x}, \mathbf{r}) := \mathsf{Decode}\Big(\frac{1}{q}\mathbf{x} - \frac{q}{2^r}\mathbf{Br}\Big)$$

Parameters of NewHope are $n = 1024$ and $q = 12289$. They use binomial distribution in error sampling Ψ_{16}^n.

Frodo

Bos *et al.* [19] proposed LWE key exchange protocol called Frodo in 2016. Protocol 2 describes key exchange protocol of Frodo. To compute Frodo, we define $\mathsf{rec}()$, rounding, and cross-rounding functions.

Let the number B of bits that from one coefficient in $\mathcal{Z}_q$ be such that $B < (\log_2 q) - 1$. Let $\overline{B} = (\log 2q) - B$. The rounding function $\lfloor \cdot \rceil_{2^B}$ is defined as follows:

$$\lfloor \cdot \rceil_{2^B} : v \mapsto \Big\lfloor 2^{-\overline{B}} v \Big\rceil \mod 2^B$$

The cross-rounding function $\langle \cdot \rangle_{2^B}$ is defined as follows:

$$\langle \cdot \rangle_{2^B} : v \mapsto \Big\lfloor 2^{-\overline{B}+1} v \Big\rfloor \mod 2$$

Then, we can define $\mathsf{rec}()$ function as follows:

$$\mathsf{rec}(w, \langle v \rangle_{2^B}) := \lfloor v \rceil_{2^B} \quad if \quad |v - w| < 2^{\overline{B}-2}$$

Protocol 2: Frodo

Alice		Bob
$\mathsf{seed}_{\mathbf{A}} \xleftarrow{\$} U(\{0,1\}^s)$		
$\mathbf{A} \leftarrow \mathsf{Gen}(\mathsf{seed}_{\mathbf{A}})$		
$\mathbf{S}, \mathbf{E} \xleftarrow{\$} \chi(\mathbb{Z}_q^{n \times \overline{n}})$		
$\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$	$\xrightarrow[\in \{0,1\}^s \times \mathbb{Z}_q^{n \times \overline{n}}]{\mathsf{seed}_{\mathbf{A}}, \mathbf{B}}$	$\mathbf{A} \leftarrow \mathsf{Gen}(\mathsf{seed}_{\mathbf{A}})$
		$\mathbf{S}', \mathbf{E}' \xleftarrow{\$} \chi(\mathbb{Z}_q^{n \times \overline{n}})$
		$\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$
		$\mathbf{C} \leftarrow \langle \mathbf{V} \rangle_{2^B}$
	$\xleftarrow[\in \mathbb{Z}_q^{\overline{m} \times n} \times \mathbb{Z}_2^{\overline{m} \times \overline{n}}]{\mathbf{B}'\mathbf{C}}$	
$K \leftarrow \mathsf{rec}(\mathbf{B}'\mathbf{S}, \mathbf{C})$		$K \leftarrow \lfloor \mathbf{V} \rceil_{2^B}$

There are four kinds of parameter sets in Frodo such as Challenge, Classical, Recommended, and Paranoid. In OQS library (liboqs) and this paper, we test recommended parameter set. Parameters of Frodo are $n = 752, q = 2^{15}, B = 4$. They use rounded Gaussian distribution in error sampling χ.

BCNS

Bos *et al.* [20] proposed ring-LWE key exchange protocol called BCNS in 2015. Protocol 3 describes key exchange protocol of BCNS.

Protocol 3: BCNS

Alice		Bob
$s, e \xleftarrow{\$} \chi$		$s', e' \xleftarrow{\$} \chi$
$b \leftarrow as + e \in \mathcal{R}_q$	$\xrightarrow{b}$	$b' \leftarrow as' + e' \in \mathcal{R}_q$
		$e'' \xleftarrow{\$} \chi$
		$v \leftarrow bs' + e'' \in \mathcal{R}_q$
		$\overline{v} \xleftarrow{\$} \mathsf{dbl}(v) \in \mathcal{R}_{2q}$
	$\xleftarrow{b',c}$	$c \leftarrow \langle \overline{v} \rangle_{2q,2} \in \{0,1\}^n$
$k_A \leftarrow \mathsf{rec}(2b's, c) \in \{0,1\}^n$		$k_B \leftarrow \lfloor \overline{v} \rceil_{2q,2} \in \{0,1\}^n$

To compute BCNS, we define $\mathsf{dbl}()$, $\mathsf{rec}()$, modular rounding, and cross-rounding functions. Let $\lfloor \cdot \rceil : \mathbf{R} \leftarrow \mathbb{Z}$ be the $\lfloor x \rceil = z$ for $z \in \mathbb{Z}$ and $x \in [z - 1/2, z + 1/2)$. The modular rounding function $\lfloor \cdot \rceil_{q,2}$ is defined as follows:

$$\lfloor \cdot \rceil_{q,2} : \mathbb{Z} \leftarrow \mathbb{Z}, \quad x \mapsto \lfloor x \rceil_{q,2} = \left\lfloor \frac{2}{q} x \right\rceil \mod 2$$

The cross-rounding function $\langle \cdot \rangle_{q,2}$ is defined as follows:

$$\langle \cdot \rangle_{q,2} : \mathbb{Z} \leftarrow \mathbb{Z}, \quad x \mapsto \langle \cdot \rangle_{q,2} = \left\lfloor \frac{4}{q} x \right\rceil \mod 2$$

Let $\mathsf{dbl}()$: $\mathbb{Z}_q \leftarrow \mathbb{Z}_{2q}$, $x \longmapsto \mathsf{dbl}(x) = 2x - e$, where e is sampled from $\{-1, 0, 1\}$ with probabilities $p_{-1} = p_1 = \frac{1}{4}$ and $p_0 = \frac{1}{2}$.

Define the sets $I_0 = \{-, 1, \cdots, \lfloor \frac{2}{q} \rceil - 1\}$ and $I_0 = \{-\lfloor \frac{q}{2} \rfloor, \cdots, -1\}$. Let $E = [-\frac{q}{4}, \frac{q}{4})$ the reconciliation function $\mathsf{rec}()$ function as follows:

$$\mathsf{rec}(w,b) = \begin{cases} 0 & \text{if } w \in I_b + E \mod 2q \\ 1 & \text{otherwise} \end{cases}$$

Parameters of BCNS are $n = 1024, q = 2^{32} - 1, \sigma = 8/\sqrt{2\pi} \approx 3.192$. They use discrete Gaussian distribution in error sampling χ.

MSrln

Longa *et al.* [21] proposed ring-LWE key exchange protocol called MSrln in 2016. They suggest modular reduction technique using Montgomery reduction. Number Theoretic Transform (NTT) is used in polynomial multiplication and addition operations. Key exchange protocol scheme is same as NewHope protocol. Also, they use same parameters from NewHope key exchange protocol.

Kyber

Bos *et al.* [22] proposed module-LWE key exchange protocol called Kyber in 2017. Protocol 4 describes key exchange protocol of Kyber. To compute Kyber, we define $\mathsf{Compress}()_q$ and $\mathsf{Decompress}()_q$ functions. Let $x \in \mathbb{Z}_q$ and $d < \lceil \log 2(q) \rceil$. The $\mathsf{Compress}()_q$ function is defined as follows:

Protocol 4: Kyber

Alice		Bob
$\rho, \sigma \leftarrow \{0,1\}^{256}$		
$\mathbf{A} \leftarrow \mathsf{Sam}(\rho) \in \mathcal{R}_q^{k \times k}$		$m \leftarrow \{0,1\}^{256}$
$(\mathbf{s}, \mathbf{e}) \leftarrow \mathsf{Sam}(\sigma) \in \beta_\eta^k \times \beta_\eta^k$		$(\hat{K}, r, d) \leftarrow G((\mathbf{t}, \rho), m)$
$\mathbf{t} \leftarrow \mathsf{Compress}_q(\mathbf{As} + \mathbf{e}, d_t)$	$\xrightarrow{(\mathbf{t},\rho)}$	$(\mathbf{u}, v) \leftarrow \mathsf{Enc}((\rho, \mathbf{t}), m; r))$
		$c \leftarrow (\mathbf{u}, v, d)$
	$\xleftarrow{c}$	
$m' \leftarrow \mathsf{Dec}(\mathbf{s}, (\mathbf{u}, v))$		
$(\hat{K}', r', d') \leftarrow G(pk, m')$		
$(\mathbf{u}', v') \leftarrow \mathsf{Enc}((\rho, \mathbf{t}), m'; r')$		$K \leftarrow H(c, K)$
$(\mathbf{u}', v', d') = (\mathbf{u}, v, d);$		
$K \leftarrow H(\hat{K}', c)$		
$(\mathbf{u}', v', d') \neq (\mathbf{u}, v, d);$		
$K \leftarrow H(z, c)$		

$$\mathsf{Compress}()_q(x,d) = \lceil (2^d/q) \cdot x \rfloor \mod^+ 2^d$$

The $\mathsf{Decompress}()_q$ is defined as follows:

$$\mathsf{Decompress}()_q(x,d) = \lceil (q/2^d) \cdot x \rfloor$$

The $\mathsf{Enc}(pk, m)$ function is defined as follows:

$$\mathsf{Enc}(pk,m) = (\mathbf{u}, v)$$

$$\mathbf{u} = \mathsf{Compress}_q(\mathbf{A}^T r + \mathbf{e}_1, d_u)$$

$$v = \mathsf{Compress}_q(\mathbf{t}^T r + e_2 + \left\lceil \frac{q}{2} \right\rceil \cdot m, d_v)$$

$$where, \mathbf{t} = \mathsf{Decompress}_q(\mathbf{t}, d_t), (r, \mathbf{e}_1, e_2) \in \beta_\eta^k \times \beta_\eta^k \times \beta_\eta$$

The $\mathsf{Dec}(privK, (u, v))$ function is defined as follows:

$$\mathsf{Dec}(privK,(u,v)) = \mathsf{Compress}_q(v - \mathbf{s}^T \cdot \mathbf{u}, 1)$$

$$where, \mathbf{u} = \mathsf{Decompress}_q(v, d_v), v = \mathsf{Decompress}_q(u, d_u)$$

Parameters of Kyber are $n = 256, q = 7681, k = 3, \eta = 4, d_u = 11, d_v = 3, d_t = 11$. They use binomial distribution in error sampling β_η^k. $H()$ and $G()$ are cryptographic hash functions.

There is three version of key exchange protocol such as unauthenticated, one-sided authenticated, and authenticated. Protocol 4 describes unauthenticated key exchange protocol using Kyber.

3.1.2 Public-key Encryption Schemes

Lybashevsky *et al.* first proposed the ring-LWE public key encryption scheme in 2010. Ring-LWE encryption scheme describes in this subsection briefly. A public key is sampled by Gaussian distribution, and the cyclotomic ring R is defined as $R = \mathbb{Z}[X]/(X^n + 1)$.

A public key encryption scheme is a tuple of Probabilistic Polynomial-Time (PPT) algorithms (Gen, Enc, Dec) satisfying the following:

- The key generation algorithm Gen() takes input a security parameter 1^λ and outputs a pair of keys $(pk, privK)$. These are called the public key and the private key, respectively. We assume that $pk = (a, b \approx s \cdot a) \in R_q \times R_q$ and $privK \in R$ each have length at least λ, and that λ can be determined from pk and $privK$.

- The encryption algorithm Enc() takes as input a public key pk and a message $m \in R_2$. It outputs a encryption message $c = (u \approx a \cdot r, v \approx b \cdot r + m \cdot \lfloor \frac{q}{2} \rceil) \in R_q \times R_q$, and we write this as $c \leftarrow \mathsf{Enc}_{pk}(m)$.

- The decryption algorithm Dec() takes as input a private key $privK$ and a ciphertext c. It outputs a message $m = v - s \cdot u$. where, $m \cdot \lfloor \frac{q}{2} \rceil \approx m \cdot \lfloor \frac{q}{2} \rceil + b \cdot r - s \cdot a \cdot r$

  We write this as $m = \mathsf{Dec}_{privK}(m)$.

We require that for every λ, every $(pk, privK)$ output by $\mathsf{Gen}(1^\lambda)$, and every message m, it holds that

$$\mathsf{Dec}_{privK}(\mathsf{Enc}_{pk}(m)) = m$$

3.1.3 Digital Signature Schemes

Akleylek *et al.* proposed the ring-LWE based signature scheme called Ring-TESLA [23]. Private key consist of a tuple of three polynomials $(s, e_1, e_2) \xleftarrow{\$} \mathcal{R}_q$, e_1 and e_2 with small coefficients. Centered discrete Gaussian distribution $\mathcal{D}_\sigma$ is used for sampling errors. Public key is a tuple of (b_1, b_2). Polynomial $a_1, a_2 \xleftarrow{\$} \mathcal{R}_q$, and computes $b_1 = a_1 s + e_1 \mod q$ and $b_2 = a_2 s + e_2 \mod q$. To sign the message m, signing algorithm samples $y \xleftarrow{\$} \mathcal{R}_q$ with coefficient in $[-B, B]$. Then, computes $c' = H(\lfloor v_1 \rceil_{d,q}, \lfloor v_2 \rceil_{d,q}, m)$ and polynomial $z = y + sc$. Signature value is a tuple of (z, c'). To verify signature (z, c') with message m, verification algorithm computes $H(\lfloor a_1 z - b_1 c \rceil_{d,q}, \lfloor a_2 z - b_2 c \rceil_{d,q}, m)$.

Güneysu *et al.* [9, 24] published the GLP signature scheme based on ring-LWE problem and implements embedded hardware systems. Polynomial ring defines $\mathcal{R}^{p^n} = \mathbb{Z}_q[\mathbf{X}]/(\mathbf{X}^n + 1)$ and $\mathcal{R}_k^{p^n}$ defines subset of the ring $\mathcal{R}^{p^n}$. $\mathcal{R}_k^{p^n}$ consists of all polynomials with coefficients in the range $[-k, k]$. To sign message μ, it needs cryptographic hash function H with range D_{32}^n. For $n \geq 512$ consists of all polynomials of degree $n-1$ that have all zero coefficients except for at most 32 coefficient that is ± 1. First, we need to read 5-bit $(r_1 r_2 r_3 r_4 r_5)$ at a time. If r_1 is 0, put -1 in position $r_2 r_3 r_4 r_5$. Otherwise, put 1 in position $r_2 r_3 r_4 r_5$. Then, we convert the 512-bit string into a polynomial of degree at least 512 as follows: i^{th} coefficient of the polynomial the i^{th}-bit of the bit-string. If the polynomial is of degree ≥ 512, then all of its higher-order terms will be 0. Algorithm 1 describe GLP signature scheme.

Ducas *et al.* [25] proposed BLISS signature scheme, which is the lattice-based signature with bimodal Gaussian distribution in 2013.

Algorithm 1: GLP Signature

Signing Key : $s_1, s_2 \xleftarrow{\$} \mathcal{R}_1^{p^n}$
Verification Key: $a \xleftarrow{\$} \mathcal{R}^{p^n}$, $\mathbf{t} \leftarrow \mathbf{a}\mathbf{s}_1 + \mathbf{s}_2$
Hash Function : $H : \{0,1\}^* \rightarrow D_{32}^n$

1 Sign$(\mu, \mathbf{a}, \mathbf{s}_1, \mathbf{s}_2)$
2 **begin**
3 $\quad \mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_k^{p^n}$;
4 $\quad \mathbf{c} \leftarrow H(\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2, \mu)$;
5 $\quad \mathbf{z}_1 \leftarrow \mathbf{s}_1\mathbf{c} + \mathbf{y}_1$;
6 $\quad \mathbf{z}_2 \leftarrow \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$;
7 $\quad$ **if** $z_1 \notin \mathcal{R}_{k-32}^{p^n}$ *or* $z_2 \notin \mathcal{R}_{k-32}^{p^n}$ **then**
8 $\quad\quad$ go to line 3;
9 $\quad$ **else**
10 $\quad\quad$ return $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$;
11 $\quad$ **end**
12 **end**
13 Verify$(\mu, \mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{a}, \mathbf{t})$
14 **begin**
15 $\quad$ **if** $z_1, z_2 \in \mathcal{R}_{k-32}^{p^n}$ **then**
16 $\quad\quad$ $c \neq H(\mathbf{a}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{t}\mathbf{c}, \mu)$;
17 $\quad\quad$ return **reject**;
18 $\quad$ **else**
19 $\quad\quad$ return **success**;
20 $\quad$ **end**
21 **end**

3.2 Overview of Blockchain

Blockchain was introduced to the Bitcoin cryptocurrency system. Bitcoin is first decentralized crypto and virtual currency and designed as a P2P network by Nakamoto in 2008. It operates in a P2P environment and adopts Proof of Work (PoW) agreement algorithm. All users in the blockchain network can create a transfer transaction with public key cryptography. A user called miner can take advantage of Proof-of-Work (PoW) operations by generating blocks with multiple valid transactions. The generated blocks are broadcast to the entire network and registered in the chain. After proposed the Bitcoin, many other cryptocurrencies such as Ethereum [26], Ripple [27], and IOTA [28] has proposed by cryptocurrency research groups. IBM is also proposed for the Hyperledger Fabric [29, 30], which is based on permissioned blockchain platform. Figure 3.1 shows the simplified version of Bitcoin blockchain.

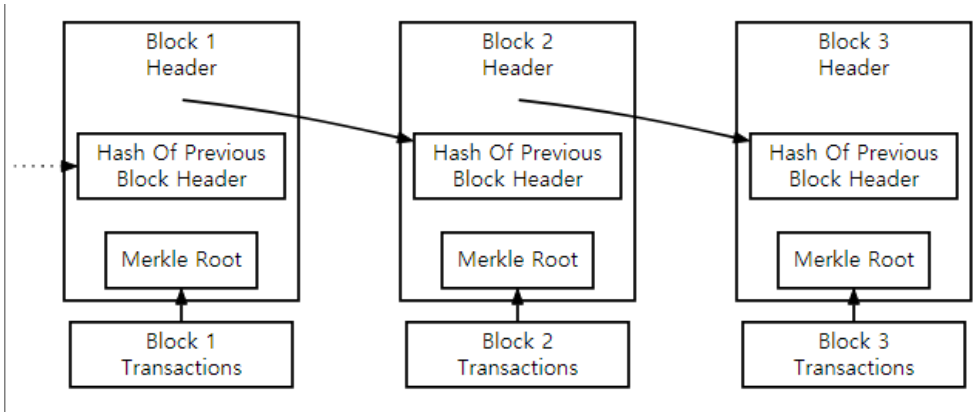

Figure 3.1: Simplified Blockchain of Bitcoin

Every block's header has hash value of previous block header. Using the transaction of each block, we can make Merkle hash tree. The first block called *genesis block* is defined as hardcoded into the application to utilize blockchain. *Genesis block* consists of a timestamp, nonce, version information, and Merkle tree hash value. After generate *genesis block*, block 1 generates using previous *genesis block* hash value.

Therefore, blockchain is designed as a decentralized managing technique of Bitcoin for issuing and transferring cryptocurrency. This technique can support the public ledger of all Bitcoin or other cryptocurrency transactions that have ever been executed, without any control of a Trusted Third Party (TTP). The advantage of Blockchain is that the public ledger cannot be modified or deleted after all user nodes have approved the data. Thus, blockchain is fully distributed and decentralized technique

system. The blockchain is well-known for data integrity and security. Blockchain technology can also be applied to other types of usage. It can, for example, create an environment for digital contracts and P2P data sharing in a cloud service. Blockchain technique can be used for other services and applications such as smart contract, medical industry, and also PKI system.

Figure 3.2 shows the simplified single block of Bitcoin. Bitcoin header contains the following:

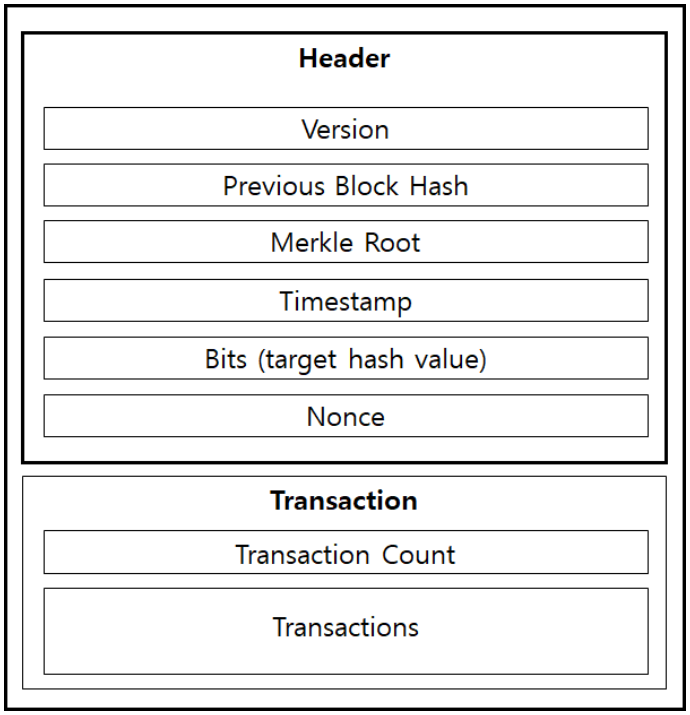

Figure 3.2: Simplified Single Block of Bitcoin

- **Version**: The block version number indicates which set of block validation rules to follow.
- **Previos Block Hash**: A SHA256() hash in internal byte order of the previous block's header. This ensures no previous block can be changed without also changing this block's header.
- **Merkle Root**: The Merkle root is derived from the hashes of all transactions included in this block, ensuring that none of those transactions can be modified without modifying the header.
- **Bits**: An encoded version of the target threshold this block's header hash must be less than or equal to the previous target value.
- **Nonce**: An arbitrary number of miners change to modify the header hash in order to produce a hash less than or equal to the target threshold.

3.3 Public Key Infrastructure

In this section, we present the X.509 PKI standards, briefly. Then, previous work of blockchain-based PKI system is described.

3.3.1 PKI Standards

A certificate is a digital document that contains public key and metadata. Legal CA can issue the valid certificates. X.509 is defined by the International Telecommunications Union's Standardization sector [5]. Figure 3.3 shows the structure of X.509 v3 certificate. Certificates contain the following fields:

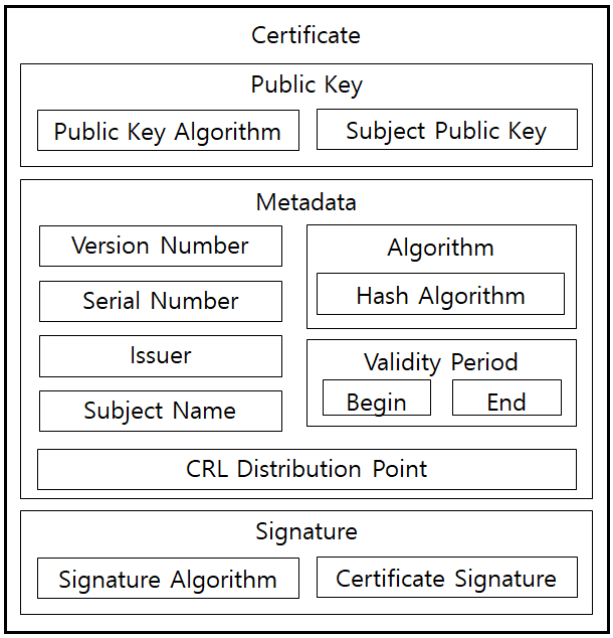

Figure 3.3: Structure of X.509 v3 Certificate

- **Public Key**: This field consists of the public key algorithm and subject public key. It contains the specific public key algorithm and public key value for each user.

- **Version Number**: X.509 standards has three kinds of version. Version 1 is default format, and if the Initiator Unique Identifier or Subject Unique Identifier is present, that must be used version 2. If more extension of certificates, the version must be used 3.

- **Subject Name**: The name of the user to whom certificate refers.

- **Issuer**: The name of CA that issued and signed the certificate.

- **Validity Period**: Valid date of certificate consist of begin and end date.

- **Signature**: This field includes signature algorithm and certificate signature. It covers all other field value and signs the certificate.

3.3.2 Blockchain-based PKI

Current PKI system is based on the centralized database. However, there is the vulnerability of single point failure. Since blockchain aims to provide a decentralized and unmodifiable ledger of information, it has qualities considered highly suitable for the storage and management of public keys.

Emercoin(EMC) [31] is cryptocurrency, which is used for blockchain-based PKI system. EMCSSH integrates between the OpenSSH and EMC blockchain, providing decentralized PKI. EMC blockchain is based on both Proof-of-Work and Proof-of-Stake consensus protocol and forked from Peercoin. EMC uses the SHA-256 hash function, and it is not secure against the quantum adversaries by Grover's algorithm [3].

Lewison *et al.* propose the blockchain-based PKI system [33] in 2016. This research describes the concept of a blockchain-based PKI and shows the advantage of their system. However, they did not consider the quantum adversary and consensus protocol.

Matsumoto *et al.* suggest the Ethereum-based PKI system called IKP [34]. IKP's decentralized nature and smart contract system allow open participation offer incentives for vigilance over CAs, and enable financial resourse against misbehavior. However, there are some security issues for Ethereum platform. Compared to the Lewison work, IKP uses the quantum-resistant hash function called Ethash [26] based on Keccak [35]. In addition, IKP uses the quantum-resistant hash function called Ethash [26]. Ethereum is based on ECDSA signature algorithm, which is not secure against the quantum adversaries. Therefore, IKP does not guarantee the long-term security.

Yakubov *et al.* propose the blockchain-based PKI management framework [36] in 2018. They design a blockchain-based PKI, which modifies the X.509 certificates. X.509 v3 certificate standard consists of extension fields, which are reserved for extra information. They modify X.509 v3 certificate and design hybrid X.509 certificate, which consists of blockchain name, CA key and subject key identifier, and hashing algorithm in the extension field. This work is based on smart contract in Ethereum.

Certcoin [37] is the public and decentralized PKI system using blockchain technique and based on

Namecoin [38]. In revocation phase, they did not use Certificate Revocation List (CRL). The weak point of this approach is that Certcoin uses only timestamp (lifetime) in each public key. They consider that Certcoin uses RSA accumulators, which is insecure against the quantum adversaries. However, Certcoin benefits a fault tolerance and redundancy.

Chapter 4. QChain: Our Proposed Scheme

We have described the lattice-based cryptography, blockchain, and public key infrastructure. In this chapter, we design the quantum-resistant PKI scheme called QChain. Specifically, we propose our construction and structure of the QChain, which consist of key exchange protocol, Merkle hash tree, and modified GLP signature scheme.

4.1 Overview of Scheme

Our proposed quantum-resistant PKI scheme is based on the ring-LWE problem. In this section, we describe the full structure of QChain in detail. We construct QChain, which is quantum-resistant PKI using blockchain. In following sections, we describe the structure of scheme, which contains Merkle hash tree, modified GLP signature scheme, and key exchange protocol. We also propose a QChain with key exchange protocol that can increase the efficiency in a communication process. QChain uses the extension field of X.509 v3 certificate. Therefore, there is an advantage that it can be compatible with existing X.509 certificate standards. QChain assumes a permissioned blockchain. Therefore, consensus protocol uses BFT instead of PoW or PoS.

4.2 Structure of QChain

Our scheme is designed to prevent quantum computing attacks. Figure 4.1 shows the full structure of QChain. We use ring-LWE encryption scheme, which is quantum-resistant primitive in QChain. More precisely, the public key encryption scheme is based on ring-LWE by Lyubashevsky *et al.* [12] which is secure against the quantum computing attacks.

Figure 4.2 shows the extended certificate for QChain. QChain certificate contains the following fields:

- **Public Key**: This field consists of the public key algorithm and subject public key. It contains the specific public key algorithm and public key value for each user.

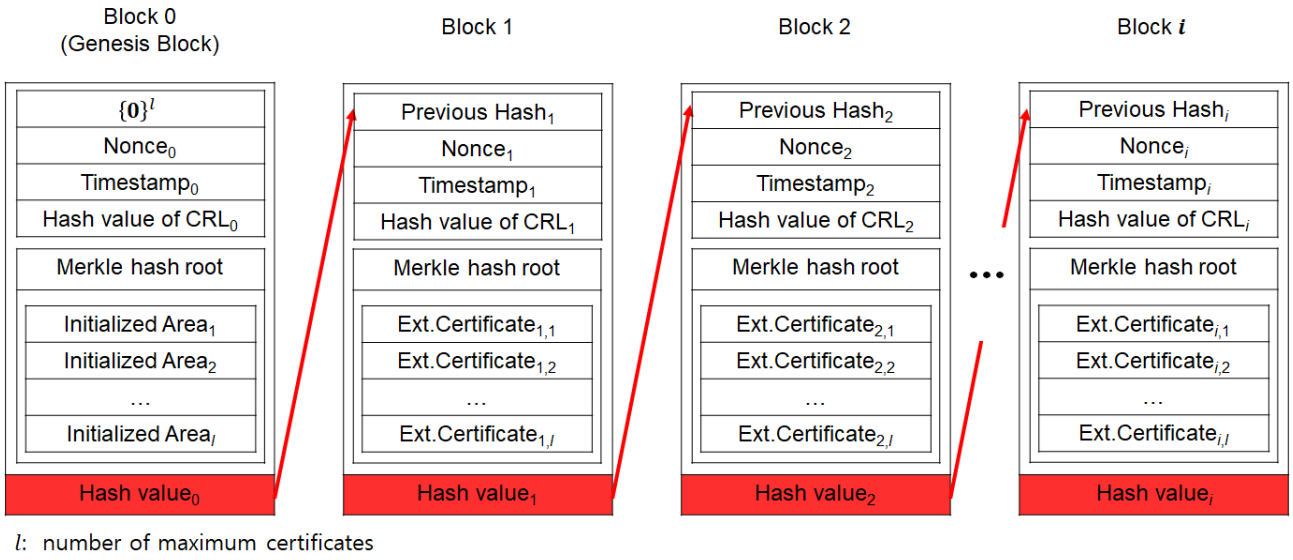

Figure 4.1: Full Structure of QChain

- **Version Number**: X.509 standards has three kinds of version. Version 1 is default format, and if the Initiator Unique Identifier or Subject Unique Identifier is present, that must use version 2. For more extension of certificates, the version must be used 3.

- **Subject Name**: The name of the user to whom certificate refers.

- **Issuer**: The name of CA that issued and signed the certificate.

- **Validity Period**: Valid date of certificate consist of begin and end date.

- **Signature**: This field includes signature algorithm and certificate signature. It covers all other field values and signs the certificate.

- **CRL Distribution Point**: This field includes a list of which establishes a CRL distribution points. Each distribution point contains a name and optionally reasons for revocation and the CRL issuer name, specifically, block leader.

- **Asserted Data**: This field consists of the previous hash value and Merkle root. Previous hash value is based on the previous block.

If the leader is a malicious node, the certificate is abolished and a new leader is elected. Thus, it prevents malicious node of the leader. The leader has a CRL, and the user confirms revocation of the

public key in the leader's CRL. The previous leader transfers the CRL and its hash value to the next leader when the leader changes.

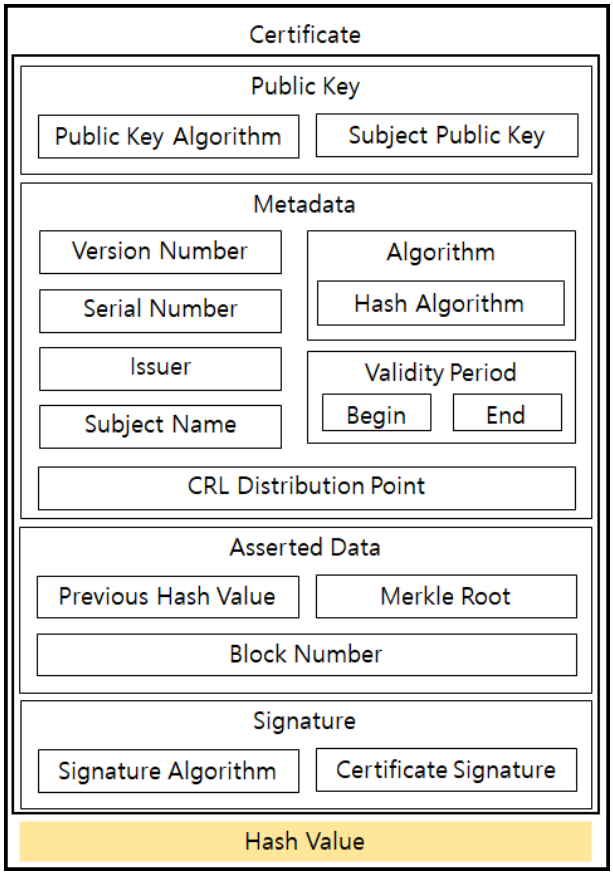

Figure 4.2: Extended Certificate for QChain

4.2.1 QChain Scheme

The polynomial ring defines $\mathcal{R}_q = \mathbb{Z}_q[\mathbf{X}]/(\mathbf{X}^n + 1)$. The error distribution χ_σ uses a discrete Gaussian distribution with standard deviation σ. For efficient encryption time, we use Number Theoretic Transformation (NTT) [39] operations. The NTT is commonly used in the implementation of lattice-based cryptography. NTT operation denotes $\hat{z} = \mathsf{NTT}(z)$. Cryptographic nonce and random number are randomly selected $nonce \xleftarrow{\$} \{0,1\}^n$ and $rand \xleftarrow{\$} \{0,1\}^n$. We denote the hash function and signature algorithm $H()$ and $Sign()$, respectively. The public and private key denote pk and $privK$, respectively. Equation 4.2.1 defines the error-reconciliation function. In Section 4.2.5, we will introduce the modified GLP digital signature scheme.

For a polynomial $\mathbf{g} = \Sigma_{i=0}^{1023} g_i X^i \in \mathcal{R}_q$, we define

$$\mathsf{NTT}(\mathbf{g}) = \hat{\mathbf{g}} = \sum_{i=0}^{1023} \hat{g}_i X^i, with$$

$$\hat{g}_i = \sum_{j=0}^{1023} \gamma^j g_j \omega^{ij}$$

where, $\omega = 49, \gamma = \sqrt{\omega} = 7$. The function NTT^{-1} defines the inverse of NTT function.

$$\mathsf{NTT}^{-1}(\hat{\mathbf{g}}) = \mathbf{g} = \sum_{i=0}^{1023} \hat{g}_i X^i, with$$

$$g_i = n^{-1}\gamma^{-i} \sum_{j=0}^{1023} \hat{g}_j \omega^{ij}$$

where, $n^{-1} \mod q = 12277, \quad \gamma^{-1} \mod q = 8778, \quad \omega^{-1} \mod q = 1254.$

The QChain scheme is described as follows:

• $\mathsf{QChain.Setup}(1^\lambda)$: Choose security parameter λ and output a parameter n, q, and $\sigma = \sqrt{16/2} \approx 2.828$ [40].

• $\mathsf{QChain.KeyGen}(n, \sigma)$: Polynomial r_1 and r_2 sampled from the Gaussian distribution use NTT operation in polynomial multiplication and addition.

$$r_{1,i}, r_{2,i} \leftarrow \chi_\sigma;$$

$$y_{1,i}, y_{2,i} \xleftarrow{\$} \mathcal{R}_q^k;$$

$$a_i \xleftarrow{\$} \mathcal{R}_q; \quad \hat{a}_i \leftarrow \mathsf{NTT}(a_i);$$

$$\hat{r}_{1,i} \leftarrow \mathsf{NTT}(r_{1,i}); \quad \hat{r}_{2,i} \leftarrow \mathsf{NTT}(r_{2,i});$$

$$\hat{y}_{1,i} \leftarrow \mathsf{NTT}(y_{1,i}); \quad \hat{y}_{2,i} \leftarrow \mathsf{NTT}(y_{2,i});$$

$$\hat{p}_i \leftarrow \hat{r}_{1,i} - \hat{a}_i * \hat{r}_{2,i};$$

$$\hat{t}_i \leftarrow \hat{a}_i * \hat{r}_{1,i} + \hat{r}_{2,i};$$

The public key is $(\hat{a}_i, \hat{p}_i, \hat{t}_i) \in pk_i$ and the private key is $(\hat{r}_{1,i}, \hat{r}_{2,i}, \hat{y}_{1,i}, \hat{y}_{2,i}) \in privK_i$ for user i.

• $\mathsf{QChain.GenesisBlock.Setup}()$: The genesis block is the first block of QChain. We also call it block 0, which is hardcoded into the software of our system. The genesis block does not have previous

hash value. Therefore, we use $\{0\}^n$ for previous hash value in genesis block. We fix $i = 2^{10}$ in genesis block.

$$nonce \xleftarrow{\$} \{0,1\}^n; rand_i \xleftarrow{\$} \{0,1\}^n;$$

$$where, 0 \leq i \leq 2^{10}$$

$$timestamp \leftarrow \textbf{current time};$$

• QChain.GenesisBlock.Merkle(): We construct Merkle hash tree after QChain.GenesisBlock.Setup() using random number $rand_i$, $timestamp$, hash function $H()$, and the signature algorithm $Sign()$. In genesis block, we fix $pk_i = rand_i$, $ID_i = i$, and $Username_i = i$. Each pk_i defines as follows:

$\mathbf{pk}_i\mathbf{Info.} =$

$$rand_i||H(i)||timestamp_i||Sign(rand_i)$$

Using $\mathbf{pk}_i$ **Info.**, we construct Merkle hash tree as follows:

$$H_{\frac{i-1}{2},\cdots,j} = \begin{cases} H_{\frac{i-1}{2},\cdots,0} = H(\mathbf{pk}_i\mathbf{Info.}) & \text{if } i = odd \\ H_{\frac{i-1}{2},\cdots,1} = H(\mathbf{pk}_i\mathbf{Info.}) & \text{if } i = even \end{cases}$$

Then, we compute the top hash value H_{root} using each hash value of leaf nodes.

• QChain.GenesisBlock.Final(): We finally construct the genesis block in this final algorithm. To make a previous hash of block 1, QChain needs a hash value. Previous hash value computes as follows:

$$H_{Block0} = H((\{0\})^n||nonce||timestamp||H_{root})$$

• QChain.User.Setup(pk_i, H_{root}): In the user setup algorithm, it is similar to QChain.GenesisBlock.Setup() algorithm. The user setup algorithm runs as follows:

$$\text{Previous hash} \leftarrow H_{Block0};$$

$$nonce \xleftarrow{\$} \{0,1\}^n;$$

$$pk_i \leftarrow \textbf{User public key} \in \{0,1\}^n;$$

$$where, 0 \leq i \leq l \leq 2^{10}$$

$$timestamp_i \leftarrow \textbf{current time};$$

• QChain.User.Add($ID_i, Username_i, privK_i, pk_i$): After the genesis block has been made by the QChain.User.Setup() algorithm, we add information about the user's public keys as follows:

$$H(ID_i), \quad ID_i \leftarrow \text{User ID};$$

$$H(Username_i), \quad Username_i \leftarrow \text{Username};$$

$$(\hat{r}_{1,i}, \hat{r}_{2,i}, \hat{y}_{1,i}, \hat{y}_{2,i}) \leftarrow privK_i;$$

$$y_{1,i} \leftarrow \mathsf{NTT}^{-1}(\hat{y}_{1,i}); \quad y_{2,i} \leftarrow \mathsf{NTT}^{-1}(\hat{y}_{2,i});$$

$$(\hat{a}_i, \hat{p}_i) \leftarrow pk_i; \quad a_i \leftarrow \mathsf{NTT}^{-1}(\hat{a}_i);$$

$$c_i \leftarrow H(a_i y_{1,i} + y_{2,i}, ID_i); \quad \hat{c}_i \leftarrow \mathsf{NTT}(c)$$

$$r_{1,i} \leftarrow \mathsf{NTT}^{-1}(\hat{r}_{1,i}); \quad r_{2,i} \leftarrow \mathsf{NTT}^{-1}(\hat{r}_{2_i});$$

$$Sign(ID_i, a_i, r_{1,i}, r_{2,i});$$

Using ID_i and $Username_i$, we compute each hash and signature value. The output signature value is $(z_{1,i}, z_{2,i}, \hat{c}_i)$. Then, we construct Merkle hash tree same as genesis block process. The maximum users of each block are 2^{10}. Because we restrict the maximum depth of Merkle hash tree due to complexity. We will explain the Merkle hash tree in Section 4.2.4. The $Sign()$ algorithm is a modified GLP signature scheme.

• QChain.User.Verify($ID_i, pk_i, Sign(ID_i)$): To verify the public key pk_i and $Sign(ID_i)$ of the user,

using the verify algorithm $Verify()$. The user verify algorithm runs as follows:

$$\hat{a_i}, \hat{t_i} \leftarrow pk_i;$$

$$a_i \leftarrow \mathsf{NTT}^{-1}(\hat{a_i}); \quad t_i \leftarrow \mathsf{NTT}^{-1}(\hat{t_i});$$

$$z_{1,i}, z_{2,i}, \hat{c_i} \leftarrow Sign(ID_i);$$

$$c_i \leftarrow \mathsf{NTT}^{-1}(\hat{c_i});$$

$$Verify(ID_i, z_{1,i}, z_{2,i}, c_i, a_i, t_i);$$

Using public parameters pk_i and $Sign(ID_i)$, we can easily verify the user.

• $\mathsf{QChain.User.Enc}(pk_i, m)$: To encrypt a message $m \in \mathcal{R}_2$, the encryption algorithm runs as follows:

$$(\hat{a}_i, \hat{p}_i, \hat{t}_i) \leftarrow pk_i;$$

$$(a_i, p_i, t_i) \leftarrow (\mathsf{NTT}^{-1}(\hat{a}_i), \mathsf{NTT}^{-1}(\hat{p}_i), \mathsf{NTT}^{-1}(\hat{t}_i));$$

$$e_1, e_2, e_3 \leftarrow \chi_\sigma;$$

$$\hat{e}_1 \leftarrow \mathsf{NTT}(e_1); \quad \hat{e}_2 \leftarrow \mathsf{NTT}(e_2);$$

$$\hat{m} \leftarrow m \cdot \left\lfloor \frac{q}{2} \right\rfloor;$$

$$(\hat{c}_1, \hat{c}_2) \leftarrow (\hat{a}_i * \hat{e}_1 + \hat{e}_2, \quad \hat{p}_i * \hat{e}_1 + \mathsf{NTT}(e_3 + \hat{m}));$$

Then, we can generate $(\hat{c}_1, \hat{c}_2)$ and the ciphertext is $c = (\hat{c}_1, \hat{c}_2)$ using a user public key pk_i and message m.

• $\mathsf{QChain.User.Dec}(privK_i, c)$: To decrypt message $c = (\hat{c}_1, \hat{c}_2)$, decryption algorithm as follows:

$$\hat{r}_{2,i} \leftarrow privK_i;$$

$$(\hat{c}_1, \hat{c}_2) \leftarrow c;$$

$$m' \leftarrow \mathsf{NTT}^{-1}(\hat{c}_1 * \hat{r}_2 + \hat{c}_2);$$

$$m \leftarrow \mathsf{Decode}(m');$$

$\mathsf{Decode}()$ is an error reconciliation function. In $\mathsf{QChain.Enc}()$ function, we encode the message m. To decode the message m', we use $\mathsf{Decode}()$ function. The $\mathsf{Decode}()$ function defines as follows:

$$\mathsf{Decode}(m) := \left\lfloor \frac{2}{q} \cdot m \cdot \lfloor q/2 \rfloor \right\rceil \cdot \left\lfloor \frac{q}{2} \right\rfloor \tag{4.1}$$

We design QChain scheme to contain ten algorithms. Figure 4.3 shows detail structure of each block of QChain. In the structure of QChain, each block consists of the previous hash, nonce, timestamp, a public key of the user, hash value of the block, and Merkle hash tree. The public key and private key of users are based on the ring-LWE key generation scheme. Users can communicate with the application data using the public key cryptosystem of the based on ring-LWE scheme.

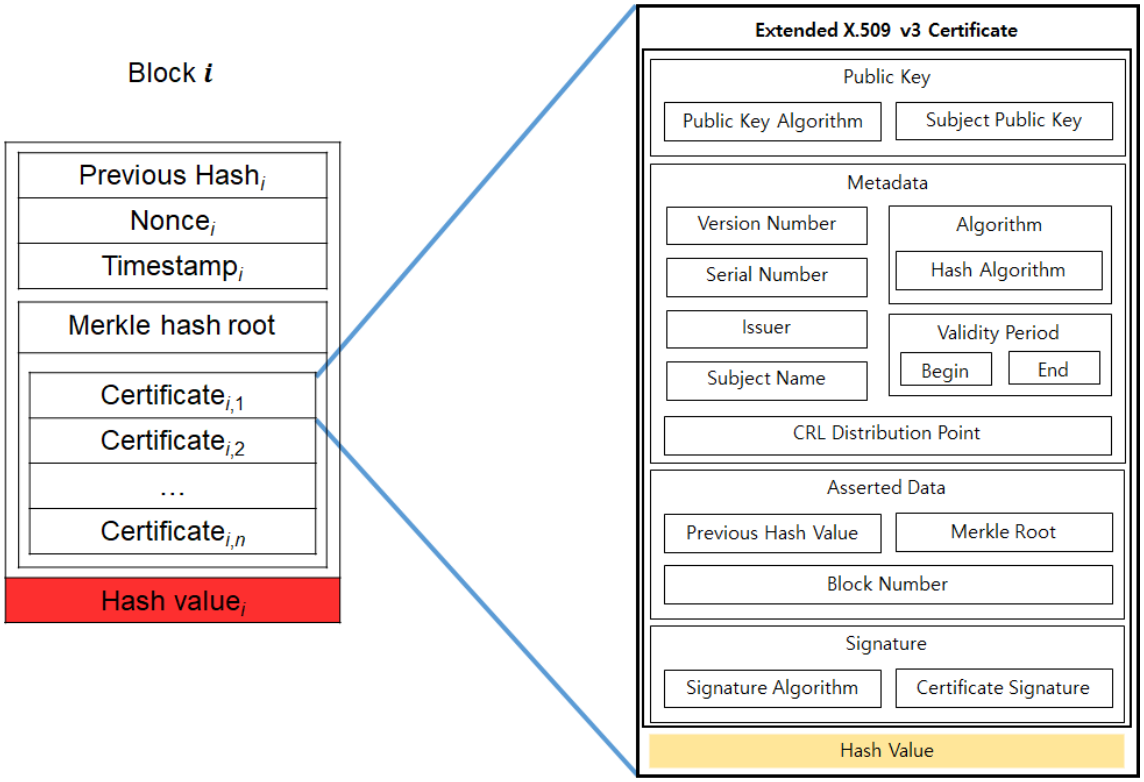

Figure 4.3: Detailed Structure of QChain

Figure 4.4 shows the simplified protocol of QChain between two users. The first QChain operator initiates genesis block (block 0). The operator has five-steps algorithms. QChain.Setup() sets the parameter of QChain. QChain.KeyGen() makes a public and a private key of users. Then, QChain.GenesisBlock.Setup(), QChain.GenesisBlock.Merkle(), and QChain.GenesisBlock.Final() algorithms to operate the genesis block. After generating genesis block, QChain makes next block called Block 1. To register the public key, users set QChain.User.Setup() algorithm and they can register the public key with algorithm QChain.User.Add(). They can also verify the public key with algorithm QChain.User.Verify(). Using this algorithm, users can challenge to QChain for verifying the anonymous user. QChain will answer if it

is an authenticated user or not. Finally, through algorithms QChain.Enc() and QChain.Dec(), users can communicate application data securely with each other.

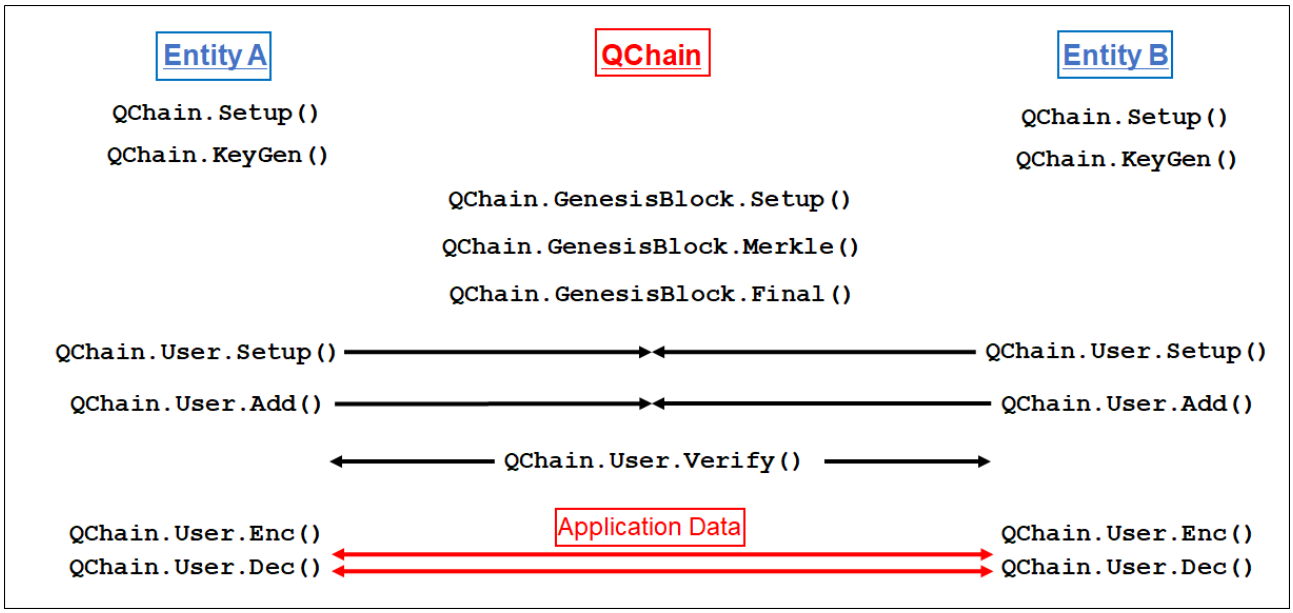

Figure 4.4: Protocol of QChain and Users

4.2.2 QChain with Key Exchange Protocol

Figure 4.5 shows the simplified protocol of QChain between two users with key exchange protocol. Compared to Figure 4.4, server and client communicate in QChain.Enc() and QChain.Dec() algorithm can be replaced by a blockcipher. Therefore, it is possible to increase the efficiency over the previous protocol in the communication process of the application data.

- QChain.KE($ID_i, KE()$): To share the common secret key $sk_{i,j}$, using the key exchange protocol $KE()$, which are the public parameter and key exchange protocol. Therefore, we can easily compute the common secret key $sk_{i,j}$. In Section 3.1.1 describe the detailed lattice-based key exchange protocols.

- QChain.Enc($sk_{i,j}, m$): To encrypt a plaintext m, using the blockcipher, which is symmetric key encryption. Then, we can generate the ciphertext that is $c = Enc_{sk_{i,j}}(m)$ using common secret key $sk_{i,j}$ and plaintext m.

- QChain.Dec($sk_{i,j}, c$): To decrypt a ciphertext c, using the blockcipher, which is symmetric key encryption. Then, we can generate the plaintext that is $c = Dec_{sk_{i,j}}(c)$ using common secret key

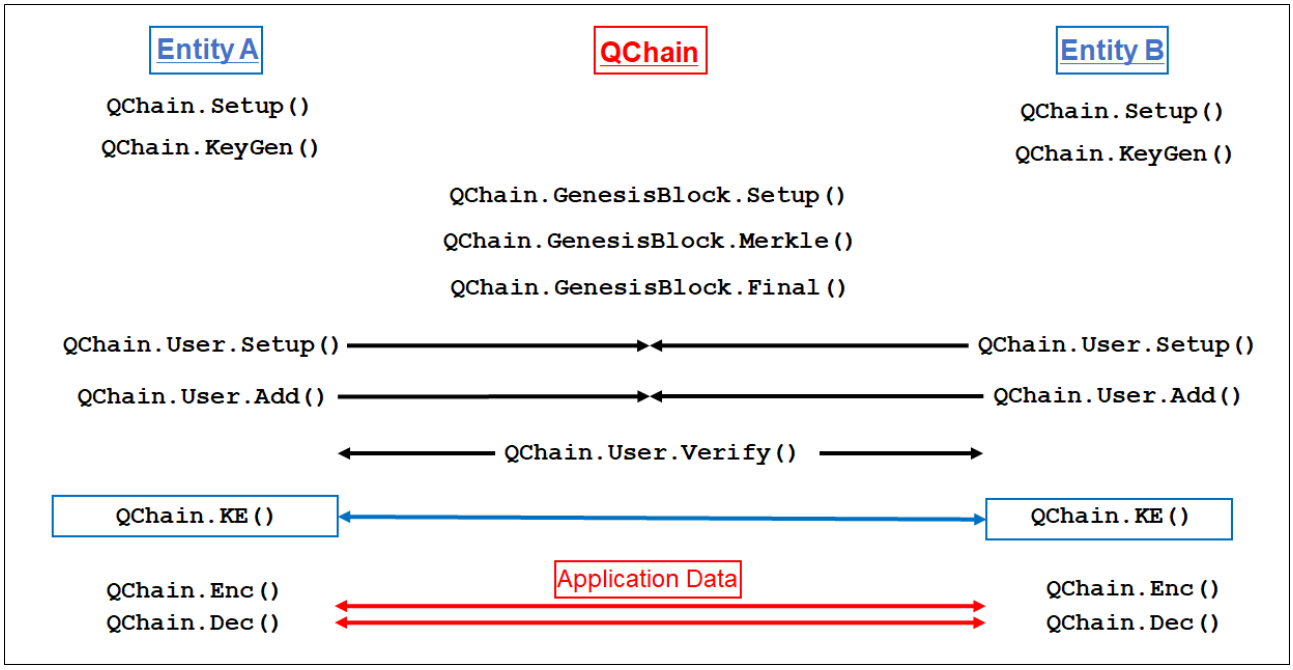

Figure 4.5: QChain Protocol with Key Exchange Protocol

$sk_{i,j}$ and ciphertext c.

4.2.3 Performance of Key Exchange Protocols

In the previous section, we proposed a design that can increase the efficiency by adding the key exchange protocol to QChain. In this section, we show detail results of the quantum-resistant library called liboqs, in case of payload and runtime. We compare the experimental setup and performance of liboqs with tables and graphs.

Experimental Setup

The experimental environment is as follows: Intel(R) CPU i7-5500, RAM 16GB, and test on Ubuntu v16.04. The compiler also uses gcc v5.4.0. We download the reference liboqs source code in GitHub[1].

Performance of liboqs

Table 4.6 describes payload of OQS project. NTRU has smallest total payload as 2049-byte. Ring-LWE and SIDH key exchange protocols have a smaller payload than code-based protocol. The largest payload in the table is McBits, which is 311,877-byte. We also check payload of LWE scheme is larger than ring-LWE scheme. Because ring-LWE computes ring structure. Therefore, ring-LWE is efficient

[1] https://github.com/open-quantum-safe/liboqs

than LWE scheme. Notably, in case of McBits, since the payload of Alice → Bob is about 0.3MB. Therefore, McBits can be utilized in the IoT device when the server has high computational power. The size of the shared key between the server and the client is 32-bit, 128-bit or 194-bit. Session key of supersingular isogeny elliptic curve is 194-bit. In Alice to Bob's payload has the largest McBit as 311736-byte and the smallest NTRU as 1,027-byte. In Bob to Alice's payload has the largest Frodo as 11288-byte and the smallest McBits as 141-byte. As a result of combining both payloads, the largest payload is McBits as 311,877-byte, and the smallest payload is NTRU as 2,049-byte.

Table 4.1: Payload on Open Quantum Safe Protocol

Mathematical Problem	Protocol	Payload (byte)			
		Alice → Bob	Bob → Alice	Total Payload	Session Key Size
Lattice-based	Frodo	11,280	11,288	22,568	32
	BCNS	4,096	4,224	8,320	128
	NewHope	1,824	2,048	3,872	32
	MSrln	1,824	2,048	3,872	32
	Kyber	1,088	1,184	2,272	32
	NTRU	1,027	1,022	2,049	32
Code-based	McBits	311,736	141	311,877	32
Supersingular Isogeny	IQC	1,164	1,164	2,328	194
Elliptic Curve	MSR SIDH	1,164	1,164	2,328	194

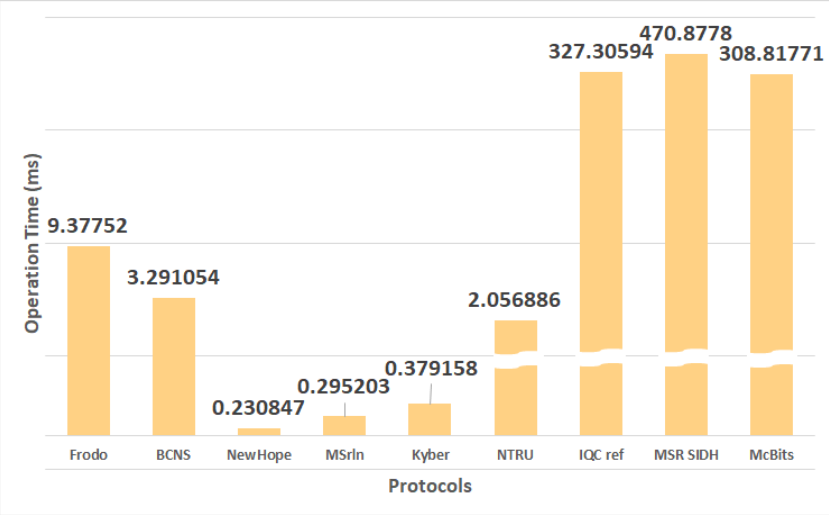

Figure 4.6: Comparing Runtime of OQS Protocols

Figure 4.6 shows runtime of OQS protocols. NewHope, MSrln, and Kyber based on the ring-LWE scheme are faster than other protocols. Runtime of NewHope is about 0.23ms and Kyber is 0.38ms. However, total runtime of supersingular isogeny elliptic curves such as IQC and MSR SIDH are at least 300ms. McBits has almost same result in SIDH schemes. These three kinds of schemes are about ten times slower than ring-LWE schemes. The fastest key exchange protocol is NewHope, which takes 0.23ms. However, The slowest key exchange protocol is MSR SIDH, which takes 470.88ms.

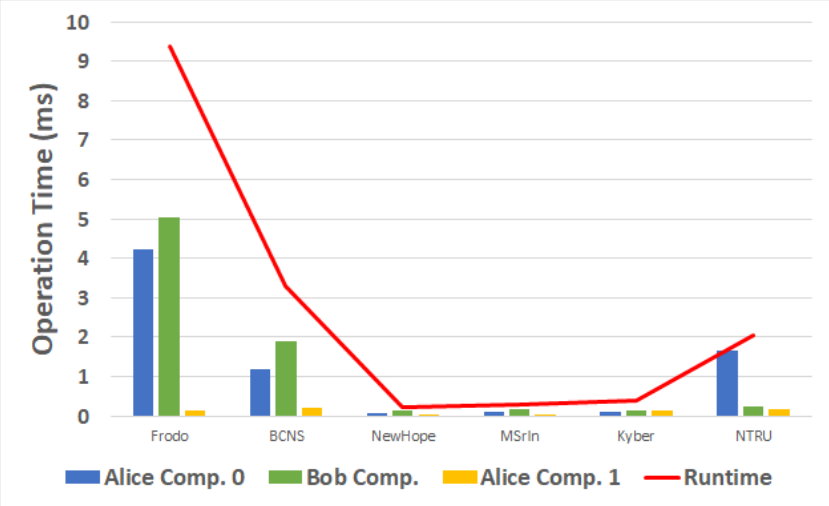

Figure 4.7: Runtime of Lattice-based Protocol

Figure 4.7 shows detail runtime of lattice-based OQS protocols. Redline means total runtime of the protocol. First Alice pre-computation (Alice Comp. 0) phase initiates key exchange protocol. Bob receives Alice's payload, and Bob computes shared key (Bob Comp.). Then, Alice computes shared key(Alice Comp. 1). Frodo is slowest key exchange protocol more than 3ms in Alice Comp. 0 and Bob Comp.. Because Frodo uses LWE scheme for security reason. However, LWE is slower than ring-LWE schemes. NewHope, MSrln, and Kyber consume less than 1ms in all phase.

Figure 4.8 shows runtime of code-based and SIDH protocols. SIDH schemes take more than 100ms in Bob Comp. phase. In the case of McBits takes more than 300ms in Alice Comp. 0 phase. The protocol with the longest computation time is 45 ms in MSR SIDH. As a result of comparing IQC and MSR SIDH, SIDH schemes are practical to use IQC with short (Alice Comp. 0) operation time. Compared with a result of Figure 4.7, the total operation time is about 30 times longer.

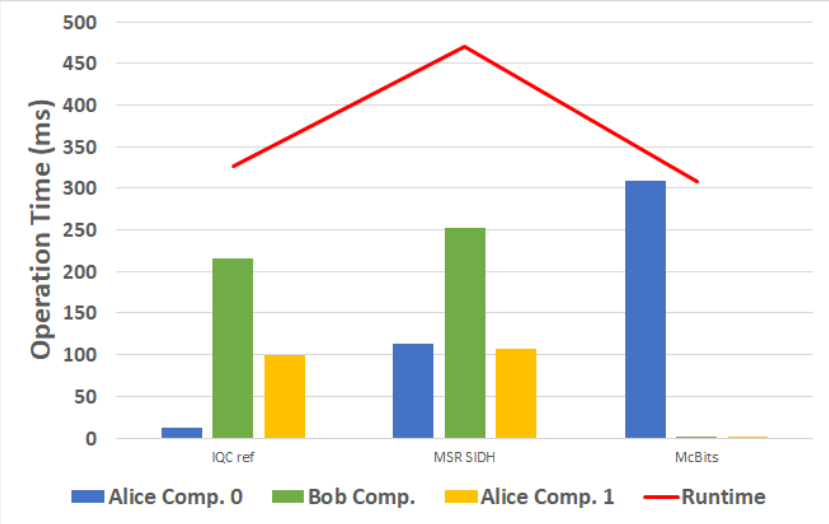

Figure 4.8: Runtime of Code-based and SIDH Protocols

4.2.4 Merkle Hash Tree

In QChain, Merkle hash tree is used for computing each user's hash values. Figure 4.9 shows an example of QChain Merkle hash tree with depth 2. We restrict the maximum 2^{10} user of each block in QChain.GenesisBlock.Setup() and QChain.User.Add() algorithms. The depth of QChain Merkle hash tree is 10 Because QChain considers efficient computation. The QChain Merkle hash tree consists of $2^{10} - 1$ nodes. Therefore, $2^{10} - 1$ hash operations are needed to generate H_{root} hash value. The $2^{10} - 1$ hash operation is a reasonable computational power and efficiency for users. The complexity of searching for pk_i is $O(\log_2(n))$ in the average case of each block. The QChain.User.Add() algorithm has also the same complexity $O(\log_2(n))$.

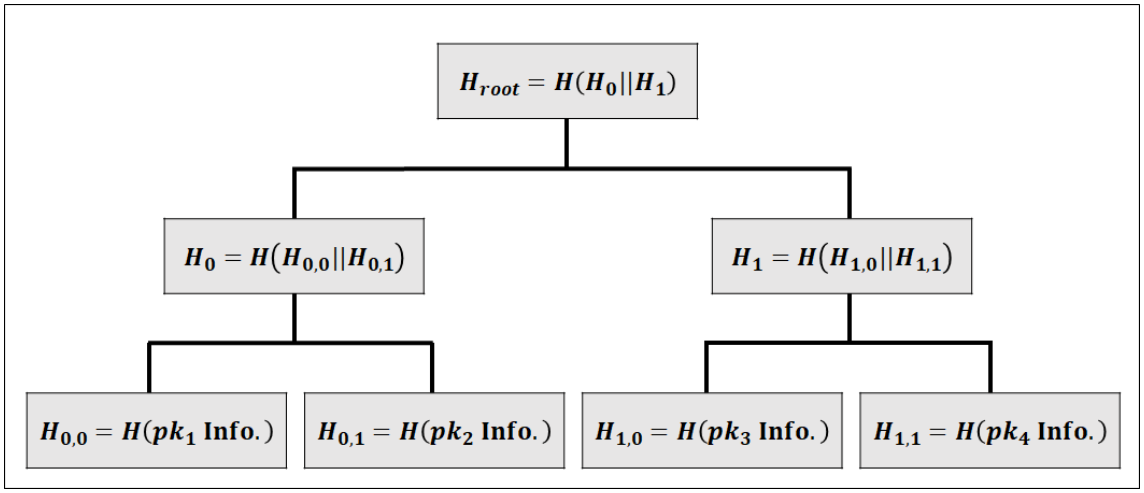

Figure 4.9: Example of QChain Merkle Hash Tree

4.2.5 Modified GLP Signature

In Section 3.1.3, we describe the GLP signature scheme. For efficient use, we modify the GLP signature scheme. Algorithm 2 describes the modified lattice-based GLP signature scheme. We integrate NTT for polynomial multiplication and addition. The modified GLP signature scheme is used for QChain.GenesisBlock.Merkle(), QChain.User.Add $(ID_i, Username_i, privK_i)$, and QChain.User.Verify$(ID_i, \mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{a}, \mathbf{t})$ algorithms. $\mathcal{R}_q^k$ to be a subset of the ring $\mathcal{R}_q$. $\mathcal{R}_q^k$ consists of all polynomials with coefficients in the range $[-k, k]$.

Algorithm 2: Modified GLP Signature

Signing Key : $r_1, r_2 \xleftarrow{\$} \chi_\sigma$

Verification Key: $a \xleftarrow{\$} \mathcal{R}_q$, $\hat{a} \leftarrow \mathsf{NTT}(a)$, $\hat{r_1} \leftarrow \mathsf{NTT}(r_1)$, $\hat{r_2} \leftarrow \mathsf{NTT}(r_2)$, $\hat{\mathbf{t}} \leftarrow \hat{a}\hat{r_1} + \hat{r_2}$

Hash Function : $H : \{0,1\}^* \rightarrow D_{32}^n$

1 $\mathsf{Sign}(\mu, \mathbf{a}, \mathbf{r}_1, \mathbf{r}_2)$
2 **begin**
3 $\quad \mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_q^k$;
4 $\quad \mathbf{c} \leftarrow H(\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2, \mu)$;
5 $\quad \hat{\mathbf{c}} = \mathsf{NTT}(\mathbf{c})$;
6 $\quad \hat{\mathbf{z_1}} \leftarrow \hat{\mathbf{r}}_1 * \hat{\mathbf{c}} + \hat{\mathbf{y}}_1$;
7 $\quad \hat{\mathbf{z_2}} \leftarrow \hat{\mathbf{r}}_2 * \hat{\mathbf{c}} + \hat{\mathbf{y}}_2$;
8 $\quad \mathbf{z}_1 \leftarrow \mathsf{NTT}^{-1}(\hat{\mathbf{z_1}})$;
9 $\quad \mathbf{z}_2 \leftarrow \mathsf{NTT}^{-1}(\hat{\mathbf{z_2}})$;
10 $\quad$ **if** $\boldsymbol{z}_1 \notin \mathcal{R}_q^{k-32}$ *or* $\boldsymbol{z}_2 \notin \mathcal{R}_q^{k-32}$ **then**
11 $\quad\quad$ go to line 3;
12 $\quad$ **else**
13 $\quad\quad$ return $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$;
14 $\quad$ **end**
15 **end**
16 $\mathsf{Verify}(\mu, \mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{a}, \mathbf{t})$
17 **begin**
18 $\quad$ **if** $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathcal{R}_q^{k-32}$ **then**
19 $\quad\quad$ $c \neq H(\mathbf{a}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{t}\mathbf{c}, \mu)$;
20 $\quad\quad$ return **reject**;
21 $\quad$ **else**
22 $\quad\quad$ return **success**;
23 $\quad$ **end**
24 **end**

Chapter 5. Security Analysis

We have described the design of QChain based on blockchain-based construction. In this chapter, we analyze the security requirement, generic attack, and compare with X.509 standards with QChain in feature analysis.

5.1 Security Requirements

Garay and Kiayias analyzed the core of Bitcoin backbone protocol [41]. They formalize and prove the security requirements for PoW consensus protocol such as common-prefix and chain-quality.

- **Common-prefix**: The blockchains maintained by the honest players will possess a large common prefix. More specifically, if an honest party "prunes" (*i.e.,* cuts off) k blocks from the end of its local chain, the probability that the resulting pruned chain will not be a prefix of another honest party's chain drop exponentially in the security parameter.
- **Chain-quality**: The ratio of blocks in the chain of any honest player. They are proved that malicious players are bounded by $\frac{t}{n-t}$, where t: number of malicious nodes, n: number of all nodes.

Kiayias *et al.* proves the PoS protocol called Ouroboros [42]. They define the security properties; safety and liveness.

- **Safety(Persistence)**: Once a node of the system proclaims a certain transaction tx as stable, the remaining nodes if queried, will either report tx in the same position in the ledger or will not report as stable any transaction in conflict to tx. Here the notion of stability is a predicate that is parameterized by a security parameter k; specifically, a transaction is declared stable if and only if it is in a block that is more than k blocks deep in the ledger.
- **Liveness**: If all honest nodes in the system attempt to include a certain transaction, then after the passing of time corresponding to u slots (called the transaction confirmation time), all nodes, if queried and responding honestly, will report the transaction as stable.

In the security requirements of the above research results, the safety of the consensus protocol of blockchain is presented. If the QChain uses a consensus protocol of PoW and PoS in the public blockchain platform, the above security requirements can be satisfied.

On the other hand, in terms of long-term security, which is not satisfactory in the majority of the blockchain, we assume an attack model for a quantum adversary. The digital signature of QChain is designed to be a lattice-based one of post-quantum cryptography, so it is secure for attack by a quantum adversary. The following Section 5.2 discusses the generic attack on quantum and classical adversary.

5.2 Generic Attack

Grover *et al.* suggest the database search algorithm called Grover's algorithm [3]. Our construction uses n_1-bit hash function. To break hash function, the complexity of brute-force attack is $O(\sqrt{2^{n_1}})$. Attacking a lattice-based cryptosystem, which has n_2-bit security key dimension with a finding shorest lattice vector using sphere-sieve also requires $2^{0.268n_2+O(n_2)}$-bit complexity [43]. Due to the ring-LWE problem is as hard as the worst case, so there is a decrease in attack amount as square root complexity despite the attack using the quantum computer. However, Shor's algorithm cannot attack our QChain construction. The encryption algorithm and digital signature of QChain are not based on IFP or DLP problems. Therefore, our construction is secure against Shor's algorithm. The attack complexity in a generic attack using a quantum computer is $min(O(2^{\frac{n_1}{2}}), 2^{0.268n_2+O(n_2)})$.

Using the classical computing attack, the hash function is secure if QChain uses the SHA3 hash function. Therefore, we can assume that the complexity of the hash function is $O(2^{n_1})$. The attack complexity of signature is $2^{0.298n_2+O(n_2)}$ [43]. Thus, total attack complexity in a generic attack using a classical computer is $min(O(2^{n_1}), 2^{0.298n_2+O(n_2)})$. However, the attack complexity of RSA and ECDSA is $O((\log n)^2(\log\log n)(\log\log\log n))$ using Shor's algorithm [2].

5.3 Feature Analysis

Table 5.1 describes the comparison of QChain and X.509 v3 PKI system. PKI system is required as register key or domain, update and look up the public key, revoke the lost key. The viewpoint of comparison is connection, non-repudiation, revocation, scalability, and model.

Table 5.1: Comparison of QChain and X.509 v3

System	**QChain**	X.509 v3
Connection	Offline	Online
Non-repudiation	O	O
Revocation	O	O
Scalability	$O(n)$	$O(n)$
Trust Model	Decentralized	Centralized

i) Connection: QChain can keep offline states except initiating genesis block. On the other hand, X.509 v3 PKI system which is used for current international standard must keep online states in TTP-server side. If TTP of X.509 v3 PKI system is offline, the user cannot verify that the public key is authenticated or not.

ii) Non-repudiation: QChain has the block which consists of user's public keys with their signatures. The user cannot deny their public information such as public key and user ID. X.509 v3 PKI system has a certificate which consists of a public key, username, and signature. Therefore, the user cannot deny their certificate.

iii) Revocation: We have already described revocation complexity of QChain in Section 4.2.4. The complexity of revocation is $O(\log_2(n))$. QChain also uses a timestamp for each block and user's information. By using a timestamp for each block, the QChain operator can specify the time to expire on each block. Since the timestamp is used for each user, the QChain operator can determine the expiration time according to the characteristics of the user. Compared with QChain, X.509 v3 PKI system stores revocation in the user's certificate. The X.509 v3 PKI system also creates and uses Certificate Revocation List (CRL). Similarly, QChain can manage the revocation list by CRL. Therefore, we can manage the revoked public key using CRL.

iv) Scalability: QChain increases linearly with scalability. Therefore, the complexity is $O(n)$. The advantage with QChain is that it does not need to increase the number of TTP servers even if the number of users and public information increases. However, the X.509 v3 PKI system must increase the computing power of the server in order to add the user's public information, because TTP of X.509 v3 PKI system stores and authenticates the user's public information.

v) Trust Model: The main point of QChain is decentralized service for PKI system. Therefore, QChain does not need TTP where X.509 v3 PKI system must have TTP. Due to the existence of TTP, X.509 v3 PKI system has a problem of single point failure.

We compare QChain and X.509 v3 PKI system. Non-repudiation, revocation, and scalability have the same results as the currently used X.509 v3 PKI system. However, the advantage of QChain is that it can be maintained offline because QChain has no central server such as TTP. In addition, our solution QChain can fundamentally solve single point failure, which is a most serious problem of X.509 v3 PKI system. Finally, since the QChain using the blockchain technique is a decentralized system, a malicious behavior of the TTP can be prevented.

5.3.1 Comparision with Related Work

In this section, we compare the features between our construction and related work, such as Certcoin, IKP, and Emercoin. Table 5.2 shows the comparison of QChain and related work. In dependence on existing cryptocurrency system, Certcoin is based on Namecoin [38], which is forked from Bitcoin. Emercoin [31] is also based on Peercoin. Lastly, IKP [34] is based on Ethereum smart contract platform.

We extend the X.509 v3 certificate with extension fields. Certcoin also extends the same approach. However, IKP and Emercoin does not use X.509 certificate. Instead, they use smart contract and makes new blocks, respectively. Therefore, it cannot be applied currently used PKI standard. QChain only uses GLP digital signature scheme, which is one of the post-quantum primitive. Other construction uses ECDSA and RSA digital signature. In other words, Certcoin, IKP, and Emercoin are not secure against quantum computing attack.

In revocation method, our construction is based on CRL and timestamp. Utilizing CRL is the most efficient method of public key revocation. In addition, we use the timestamp to assist CRL. Unlike our construction, Certcoin uses only timestamp without CRL, which makes the disadvantage. Using

Table 5.2: Comparison of QChain and Related Work

System	**QChain**	Certcoin [37]	IKP [34]	Emercoin [31]
Dependence on Existing Cryptocurrency System	N	Namecoin (fork of Bitcoin)	Ethereum	Peercoin (fork of Bitcoin)
Extending X.509 Certificates	Y	Y	N	N
Signature Scheme	GLP	ECDSA	ECDSA	RSA
Hash Function	Any kind of hash function	SHA256	Ethash	SHA256
Complexity on Signature using Quantum Computer	$2^{0.268n+O(n)}$	polynomial-time[1]	polynomial-time[1]	polynomial-time[1]
Quantum-resistance	O	X	X	X
Revocation Method	CRL Timestamp	Timestamp	Smart Contract	Update by Administrator

1: $O((\log n)^2(\log\log n)(\log\log\log n))$

timestamp has the disadvantage that user need to manually update a new certificate when the user needs to revoke the public key. IKP and Emercoin revoke by CA in the same way of current PKI standard as a smart contract.

Chapter 6. Concluding Remarks

This thesis proposes QChain, which is a decentralized PKI system that uses the blockchain technique based on the ring-LWE scheme. QChain is a quantum-resistant PKI system against the quantum adversary. We combine the blockchain technique and PQC primitive, which is lattice-based cryptography. For an efficient design of QChain, we use the NTT operations in polynomial multiplication and addition. We also modify the GLP signature scheme, which is based on the ring-LWE problem for the NTT operations. There are two versions of QChain. First, QChain protocol is based on the modified the GLP signature scheme. Second, the protocol is based on the GLP signature scheme and practical lattice-based key exchange protocol such as BCNS [20], Frodo [19], and NewHope [18]. There are security requirements for the blockchain system, that is safety, liveness, and fault tolerance. Security requirements are analyzed according to the consensus protocol. The generic attack on QChain is described for both quantum and classical adversaries. We consider the best known generic attack algorithm, such as Grover's algorithm and BKZ-2.0 algorithm. Finally, we compare the currently used X.509 v3 PKI system with our QChain in feature analysis.

As future work, several directions should be explored from here. First, we will implement QChain using C–language as an open source project. Our implementation needs the consensus algorithm in the validating blocks. QChain is focused on the key management system, that is the public key infrastructure. Therefore, we consider that QChain uses the consortium or private blockchain. Second, we will consider the quantum-resistant hash function. In the implementation, We will also consider the SHA3 [35] or other secure hash function against the quantum adversary. Third, we need a formal security proof of the QChain scheme using the game theory technique. Finally, the blockchain technique has many advantages in authentication and secure communication. QChain can be used in the access control and user authentication areas.

Bibliography

[1] IBM Research, "IBM Q experience." https://www.research.ibm.com/ibm-q/, 2018. [Online; accessed 20-Mar.-2018].

[2] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings, Annual Symposium on Foundations of Computer Science–FOCS'94*, pp. 124–134, IEEE, 1994.

[3] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing–STOC'96*, pp. 212–219, ACM, 1996.

[4] NIST, "NIST Post-quantum cryptography." https://csrc.nist.gov/Projects/Post-Quantum-Cryptography, 2018. [Online; accessed 20-Jan.-2018].

[5] P. Yee, "Updates to the internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile," 2013.

[6] R. Haenni and J. Jonczy, "A new approach to PGP's web of trust," in *European e-Identity Conference–EEMA'07*, 2007.

[7] P. R. Zimmerman, *How PGP works/Why do you need PGP?* The MIT Press Cambridge, 1996.

[8] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[9] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann, "Practical lattice-based cryptography: A signature scheme for embedded systems," in *Conference on Cryptographic Hardware and Embedded Systems–CHES'12*, pp. 530–547, Springer, 2012.

[10] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the 1st ACM conference on Computer and communications security–ACM CCS'93*, pp. 62–73, ACM, 1993.

[11] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, p. 34, 2009.

[12] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques–EUROCRYPT'10*, pp. 1–23, Springer, 2010.

[13] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Designs, Codes and Cryptography*, vol. 75, no. 3, pp. 565–599, 2015.

[14] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[15] A. Back, "Hashcash–a denial of service counter-measure," 2002.

[16] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems–TOPLAS'82*, vol. 4, no. 3, pp. 382–401, 1982.

[17] D. Stebila and M. Mosca, "Post-quantum key exchange for the internet and the open quantum safe project," in *Cryptology ePrint Archive, Report 2016/1017*, 2016. http://eprint.iacr.org/2016/1017.

[18] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange—a new hope," in *USENIX Security Symposium–USENIX Security'16*, pp. 327–343, 2016.

[19] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila, "Frodo: Take off the ring! practical, quantum-secure key exchange from LWE," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security–ACM CCS'16*, pp. 1006–1018, ACM, 2016.

[20] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, "Post-quantum key exchange for the TLS protocol from the ring learning with errors problem," in *IEEE Symposium on Security and Privacy–IEEE S&P'16*, pp. 553–570, IEEE, 2015.

[21] P. Longa and M. Naehrig, "Speeding up the number theoretic transform for faster ideal lattice-based cryptography," in *in International Conference on Cryptology And Network Security–CANS'16*, pp. 124–139, Springer, 2016.

[22] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, and D. Stehlé, "CRYSTALS–Kyber: a CCA-secure module-lattice-based KEM," in *Cryptology ePrint Archive, Report 2017/634*, 2017. http://eprint.iacr.org/2017/634.

[23] S. Akleylek, N. Bindel, J. Buchmann, J. Krämer, and G. A. Marson, "An efficient lattice-based signature scheme with provably secure instantiation," in *International Conference on Cryptology in Africa–AFRICACRYPT'16*, pp. 44–60, Springer, 2016.

[24] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann, "Practical lattice-based cryptography: A signature scheme for embedded systems.," in *Conference on Cryptographic Hardware and Embedded Systems–CHES'12*, vol. 7428, pp. 530–547, Springer, 2012.

[25] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky, "Lattice signatures and bimodal gaussians," in *Annual International Cryptology Conference on Advances in Cryptology–CRYPTO'13*, pp. 40–56, Springer, 2013.

[26] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, 2014.

[27] D. Schwartz, N. Youngs, and A. Britto, "The Ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, 2014.

[28] S. Popov, "The tangle," *IOTA White Paper*, 2016.

[29] E. Androulaki, C. Cachin, A. De Caro, A. Sorniotti, and M. Vukolic, "Permissioned blockchains and hyperledger fabric," *ERCIM NEWS*, no. 110, pp. 9–10, 2017.

[30] C. Cachin, "Architecture of the Hyperledger blockchain fabric," in *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016.

[31] O. Khovayko, "Emercoin." https://emercoin.com, 2018. [Online; accessed 15-May.-2018].

[32] S. King and S. Nadal, "Peercoin." https://peercoin.net/whitepaper, 2018. [Online; accessed 10-Jun.-2018].

[33] K. Lewison and F. Corella, "Backing rich credentials with a blockchain PKI," tech. rep., Tech. Rep, 2016.

[34] S. Matsumoto and R. M. Reischuk, "IKP: Turning a PKI around with blockchains," in *Cryptology ePrint Archive, Report 2016/1018*, 2016. http://eprint.iacr.org/2016/1018.

[35] M. J. Dworkin, "SHA-3 standard: Permutation-based hash and extendable-output functions," *Federal Inf. Process. Standards.(NIST FIPS-202)*, 2015.

[36] A. Yakubov, W. Shbair, A. Wallbom, D. Sanda, *et al.*, "A blockchain-based PKI management framework," in *The First IEEE/IFIP International Workshop on Managing and Managed by Blockchain (Man2Block) colocated with IEEE/IFIP NOMS 2018, Tapei, Tawain 23-27 April 2018*, 2018.

[37] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A decentralized public key infrastructure with identity retention," in *Cryptology ePrint Archive, Report 2014/803*, 2014. http://eprint.iacr.org/2014/803.

[38] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan, "An empirical study of Namecoin and lessons for decentralized namespace design.," in *WEIS*, 2015.

[39] T. Güneysu, T. Oder, T. Pöppelmann, and P. Schwabe, "Software speed records for lattice-based signatures," in *International Workshop on Post-Quantum Cryptography–PQCrypto'13*, pp. 67–82, Springer, 2013.

[40] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé, "Classical hardness of learning with errors," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing–STOC'13*, pp. 575–584, ACM, 2013.

[41] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques–EUROCRYPT'15*, pp. 281–310, Springer, 2015.

[42] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference–CRYPTO'17*, pp. 357–388, Springer, 2017.

[43] T. Laarhoven, M. Mosca, and J. Van De Pol, "Finding shortest lattice vectors faster using quantum search," *Designs, Codes and Cryptography*, vol. 77, no. 2-3, pp. 375–400, 2015.

[44] Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better lattice security estimates," in *International Conference on the Theory and Application of Cryptology and Information Security–Asiacrypt'11*, pp. 1–20, Springer, 2011.

Acknowledgments in Korean

이 논문을 작성하기까지 많은 분들께 다양한 도움을 받았습니다. 연구의 방향과 연구자의 태도뿐만 아니라 인생에 대한 조언과 지도를 해주신 지도교수님이신 김광조 교수님께 깊은 감사의 말씀을 드립니다. 또한 바쁘신 와중에서 학위논문심사에 참여하셔서 진심어린 조언을 주신 김용대 교수님과 강병훈 교수님께도 감사의 말씀을 드립니다.

그리고 연구실의 선배이자 랩장인 최락용 형님과 이지은에게 연구에 대한 도움뿐만 아니라 연구실 생활과 같은 부분에서 이야기하며 격려를 받았습니다. 한학기 먼저 졸업하신 김성숙 누님과 정보보호대학원 1년 후배인 한성호와 최낙준은 같이 늦은 밤과 새벽 함께 공부를 하고 야식을 먹으며 많은 도움이 되었습니다. 한학기 밖에 함께 하지 못한 이나비 누님과 홍동연에게도 졸업에 대한 많은 격려를 받았습니다. 같이 졸업하는 Aminanto Erza Muhamad와 Harry Chandra Tanuwidjaja와 행정적인 도움을 주신 홍지연씨 모두 감사드립니다.

또한 정보보호대학원 동기인 김문범 형님, 김건우 형님과 함께 학교 수업과 프로젝트, 석사논문을 작성하며 많은 도움과 격려를 받았습니다. 이외에도 정보보호대학원 교수님들과 선후배님들 모두에게 감사드립니다. 서울에 올라가도 반갑게 맞이해주는 중고등학교부터 이어진 친구인 정재우, 최도림, 강민주, 나길주, 이재근에게도 감사합니다. 지금까지 언급하지 않은 모든 분들께 모두 감사를 드립니다.

끝으로 지금까지 믿음을 가지고 지켜봐 주시고 응원해준 할머니, 부모님, 이모, 동생 혜경이에게 깊은 감사를 드립니다. 앞으로도 지금의 저보다 한층 더 나아진 모습으로 보답하겠습니다. 감사합니다.

Curriculum Vitae in Korean

이 름: 안 형 철

생 년 월 일: 1990년 10월 26일

전 자 주 소: anh1026@kaist.ac.kr

학 력

2006. 3. – 2009. 2. 서울 대신고등학교

2010. 3. – 2016. 8. 세종대학교 수학과 (B.S.)

2016. 9. – 2018. 8. 한국과학기술원 정보보호대학원 (M.S.)

경 력

2017. 3. – 2017. 6. 한국과학기술원 정보보호론 일반조교

2017. 9. – 2017. 12. 한국과학기술원 고급정보보호 일반조교

2018. 3. – 2018. 8. 대전과학고등학교 심화자율연구 조교

연 구 과 제

2016. 9. – 2018. 4. 양자컴퓨터 공격에 안전한 새로운 래티스 기반 완전 준동형 서명 방식 설계 및 안전성 분석

2017. 3. – 2018. 2. 생체모방 알고리즘(Bio-Inspired Algorithm)을 활용한 통신기술 연구

2017. 8. – 2018. 8. 양자 컴퓨터 환경에서 래티스 문제를 이용한 다자간 인증키 교환 프로토콜 연구

2018. 5. – 2018. 8. 암호화폐와 스마트 컨트랙트 응용 시스템 설계 및 보안 취약성 분석 연구

연 구 업 적

1. **안형철**, Muhamad Erza Aminanto, 최락용, 김광조, "**ELK: 래티스 기반 암호 라이브러리의 확장**," 2016 정보보호학술발표회논문집 충청지부, 대전대학교, 대전. 2016.10.07.
2. **Hyeongcheol An**, Sungsook Kim, Jeeun Lee, Rakyong Choi, and Kwangjo Kim, "**Timing and Fault Attacks on Lattice-based Cryptographic Libraries**," 2017 Symposium on Cryptography and Information Security, Session 2B3-6 (SCIS 2017), Jan., 24-27, 2017, Naha, Japan.
3. **안형철**, 최락용, 이지은, 김광조, "**래티스 기반 키 교환 프로토콜의 비교**," 한국정보보호학회 하계학술대회(CISC-S'17), 순천향대학교, 아산, 2017.06.23.
4. 최락용, **안형철**, 이지은, 김성숙, 김광조, "**양자 컴퓨터 공격에 안전한 격자 기반 키 교환 방식 비교**," 한국통신학회논문지, 제42권, 제11호, pp.2200-2207. 2017.11.30.
5. **안형철**, 한성호, 최낙준, 김광조, "**OQS 프로젝트 중 격자 기반 키 교환 방식의 타이밍 등 공격 분석**," 한국정보보호학회 동계학술대회(CISC-W'17), 고려대학교, 서울, 2017.12.09.
6. **Hyeongcheol An** and Kwangjo Kim, "**QChain: Quantum-resistant and Decentralized PKI using Blockchain**," 2018 Symposium on Cryptography and Information Security, Session 3C1-5 (SCIS 2018), Jan., 23-26, 2018, Niigata, Japan.
7. **Hyeongcheol An**, Rakyong Choi, Jeeun Lee, and Kwangjo Kim, "**Performance Evaluation of liboqs in Open Quantum Safe Project (Part I)**," 2018 Symposium on Cryptography and Information Security, Session 3A4-1 (SCIS 2018), Jan., 23-26, 2018, Niigata, Japan.
8. Rakyong Choi, **Hyeongcheol An**, and Kwangjo Kim, "**AtLast: Another Three-party Lattice-based PAKE Scheme**," 2018 Symposium on Cryptography and Information Security, Session 2B1-4 (SCIS 2018), Jan., 23-26, 2018, Niigata, Japan.
9. Seongho Han, Nakjun Choi, **Hyeongcheol An**, Rakyong Choi, and Kwangjo Kim, "**Prey on Lizard: Mining Secret Key on Lattice-based Cryptosystem**," 2018 Symposium on Cryptography and Information Security, Session 3A4-2 (SCIS 2018), Jan., 23-26, 2018, Niigata, Japan.

10. **안형철**, 김광조, "**블록체인 기반 분산형 키 관리 시스템의 설계**," 한국정보보호학회 하계학술대회 (CISC-S'18), 동신대학교, 나주, 2018.06.21. – [**우수논문**]