SCIS 2018 2018 Symposium on Cryptography and Information Security
Niigata, Japan, Jan. 23 - 26, 2018
The Institute of Electronics, Information and Communication Engineers

QChain: Quantum-resistant and Decentralized PKI using Blockchain

Hyeongcheol An * Kwangjo Kim *

Abstract: Blockchain was developed under public domain and distributed database using the P2P connection. Therefore, blockchain does not have any central administrator or Certificate Authority(CA). However, Public Key Infrastructure(PKI) must have CA which issues and signs the digital certificates. PKI CA must be fully trusted by all parties in a domain. Also, current public key cryptosystem can be broken using quantum computing attacks. The post-quantum cryptography (PQC) must be secure against the quantum adversary. We combine blockchain technique with one of post-quantum cryptography lattice-based cryptosystems. In this paper, we suggest QChain which is quantum-resistant decentralized PKI system using blockchain. We propose modified lattice-based GLP signature scheme. QChain uses modified GLP signature which uses Number Theoretic Transformation (NTT). We compare currently used X.509 v3 PKI and QChain certificate.

Keywords: blockchain, post-quantum cryptography, lattice, decentralized PKI

1 Introduction

1.1 Motivation

Public-key cryptosystem needs Public Key Infrastructure (PKI) which is to guarantee the integrity of for all user's public key. Currently used PKI system is X.509 v3[1] as the international standards. However, X.509 PKI system has some disadvantage such as centralization, single point failure, and fully trusted Certificate Authority (CA). CA is a trusted third party whose signature on the certificate guarantees for the authenticity of the public key bound to each entity. CA must need to securely store and revocation. If CA is not online called single point failure, the client cannot store or revocation their public keys. Therefore, currently used centralized PKI system has some problem with availability. Due to centralized CA, we fully trust CA server. Thus, we need to decentralized PKI system to solve disadvantage of centralized PKI system.

The most famous cryptocurrency Bitcoin [2] is the first decentralized crypto and virtual currency. Bitcoin uses blockchain which is a transaction database shared by all peer nodes. With transaction of the blockchain, anyone can find each information of transaction history. A peer-to-peer (P2P) which is decentralized network is distributed system between peers. Each peer has equally same privilege on their network. P2P does not have the concept of client or server. However, each peer nodes operate both client and server on their network at the same time. Since blockchain is decentralized, we combine blockchain and PKI system.

Public key cryptosystem such as Diffie-Hellman (DH) key exchange protocol and RSA are based on the difficulty of Discrete Logarithm Problem (DLP) and Integer Factorization Problem (IFP). However, DLP and IFP can be solved within the polynomial time by Shor's algorithm[3] using the quantum computer. Therefore, we need secure public key cryptosystem against the quantum adversary. Post Quantum Cryptography (PQC) is secure cryptosystem against both classical and quantum adversaries. We focus on lattice-based cryptography which 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 key exchange protocol and signature. We use GLP[4] signature scheme which based on Ring-LWE problem. Because GLP scheme is simple and efficient signature algorithm.

In this paper, we suggest QChain which is 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 quantum secure PKI, we use lattice-based GLP signature scheme. We also compare currently used X.509 v3 PKI system and our QChain from the point of connection, non-repudiation, revocation, scalability, trust model, and security level.

1.2 Outline of the Paper

In Section 2, we introduce blockchain and lattice-based cryptography such as well-known mathematical hard problem Learning with Errors (LWE) and Ring-LWE. Section 3 describes related work using blockchain and PKI systems. Then, we construct QChain which is quantum-resistant decentralized PKI using blockchain technique in Section 4. We compare currently used PKI standard X.509 v3 and QChain, then we suggest future work and conclusion in Sections 5 and 6, respectively.

* Graduate School of Information Security, KAIST. 291, Daehak-ro, Yuseong-gu, Daejeon, South Korea 34141. $\{anh1026, kkj\}$@kaist.ac.kr

2 Background

In this section, the blockchain and the lattice-based cryptography such as LWE and Ring-LWE problem will be described in brief.

2.1 Blockchain

Blockchain was introduced to the Bitcoin cryptocurrency system. Bitcoin is a first decentralized virtual currency and designed as a P2P network by Satoshi 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 Bitcoin, many other cryptocurrencies such as Ethereum[5], Ripple, and IOTA has proposed by cryptocurrency research groups. Figure 1 shows the simplified version of Bitcoin blockchain.

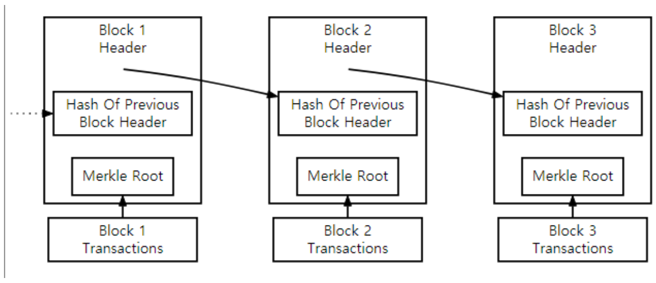

Figure 1: Simplified Bitcoin Blockchain

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 the data has been approved by all user nodes. 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.

2.2 Lattice-based Cryptography

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 Learning-with-Errors (LWE), Ring Learning-with-Errors (Ring-LWE), Module Learning-with-Errors (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. We will describe LWE and Ring-LWE problems in brief. In this paper, we use Ring-LWE scheme using key generation and signature.

2.2.1 Learning with Errors

LWE problem is introduced by Regev[6] in 2009. LWE is a quantum-resistant mathematical hard problem against the quantum adversary.

Error distribution χ over $\mathbb{Z}$ is usually used Gaussian distribution or binomial 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;

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

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. Then, distinguish $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^n$ or uniformly random $A_{s,\chi}$.

2.2.2 Ring Learning with Errors

Ring-LWE problem is introduced by Lyubashevsky *et al.*[7] in 2010. Ring-LWE is also a quantum-resistant mathematical hard problem against the quantum adversary.

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 Gaussian distribution or binomial distribution. Ring-LWE distribution $A_{s,\chi} \in \mathcal{R}_q \times \mathcal{R}_q$ and choose secret vector $s \in \mathcal{R}_q$ and choose uniformly random $a \in \mathcal{R}_q$, and choosing $e \leftarrow \chi$. and outputting;

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

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. Then, distinguish $(a_i, b_i) \in \mathcal{R}_q \times \mathcal{R}_q$ or uniformly random $\mathcal{R}_q$.

3 Related Work

Even though lots of researches on lattice-based cryptography have been studied for various cryptographic applications, small research has been researched in PKI system and blockchain areas. In this section, we introduce a PKI system with blockchain and lattice-based signature such as GLP scheme.

3.1 Application using Blockchain Technique

Axon *et al.*[8] proposed blockchain-based PKI in 2006. Since this scheme based on Shamir's secret sharing scheme, it is not safe against the quantum adversary. Meanwhile, Bansarkhani *et al.* [9] proposed first lattice-based multi-signature scheme to Bitcoins based on ring-LWE problem. This scheme can be used for the Bitcoin-Blockchain transaction.

Kosba *et al.*[10] proposed smart contract system called Hawk using blockchain technique in 2016. The Hawk does not store financial transactions in the blockchain. In a public view, it can preserve private contracting information. Etherum[5] is a blockchain-based platform which is developed by using smart contract. Not only human-to-human contracts but also machine-to-machine such as Internet-of-things (IoT) devices can be used for smart contract system. Azaria *et al.*[11] suggested medical data access and permission management system called MedRec in 2016. MedRec is decentralized record management system to use Electronic Medical Records (EMRs) using blockchain technique.

3.2 Lattice-based Signature

Akleylek *et al.* proposed Ring-TESLA[12] Ring-LWE based signature scheme. Secret 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.*[4, 13] published 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, there need 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 are ± 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.

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    : $H : \{0,1\}^* \to D_{32}^n$
1  Sign$(\mu, \mathbf{a}, \mathbf{s}_1, \mathbf{s}_2)$
2  begin
3      $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_k^{p^n}$;
4      $\mathbf{c} \leftarrow H(\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2, \mu)$;
5      $\mathbf{z}_1 \leftarrow \mathbf{s}_1\mathbf{c} + \mathbf{y}_1$;
6      $\mathbf{z}_2 \leftarrow \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$;
7      if $z_1 \notin \mathcal{R}_{k-32}^{p^n}$ or $z_2 \notin \mathcal{R}_{k-32}^{p^n}$ then
8          go to line 3;
9      else
10         return $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$;
11     end
12 end
13 Verify$(\mu, \mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{a}, \mathbf{t})$
14 begin
15     if $z_1, z_2 \in \mathcal{R}_{k-32}^{p^n}$ then
16         $c \neq H(\mathbf{a}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{t}\mathbf{c}, \mu)$;
17         return reject;
18     else
19         return success;
20     end
21 end
```

Ducas *et al.*[14] proposed BLISS signature scheme which is lattice-based signature with bimodal Gaussian distribution in 2013.

4 QChain

We construct QChain which is quantum-resistant PKI using blockchain. In this section, we describe full structure of QChain and detail of construction.

4.1 Structure of QChain

Figure 2 shows full structure of QChain. We use ring-LWE encryption scheme which is quantum-resistant primitive in QChain. More precisely, Public key encryption scheme is based on ring-LWE by Lyubashevsky *et al.*[7] which secure against quantum computing attacks. The polynomial ring defines $\mathcal{R}_q = \mathbb{Z}_q[\mathbf{X}]/(\mathbf{X}^n + 1)$ where polynomial f is an irreducible of degree $n-1$. The error distribution χ_σ uses discrete Gaussian distribution with standard deviation σ. For efficient encryption time, we use Number Theoretic Transformation (NTT)[15] operations. NTT is a commonly used in 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 hash function and signature algorithm $H()$ and $Sign()$, respectively.

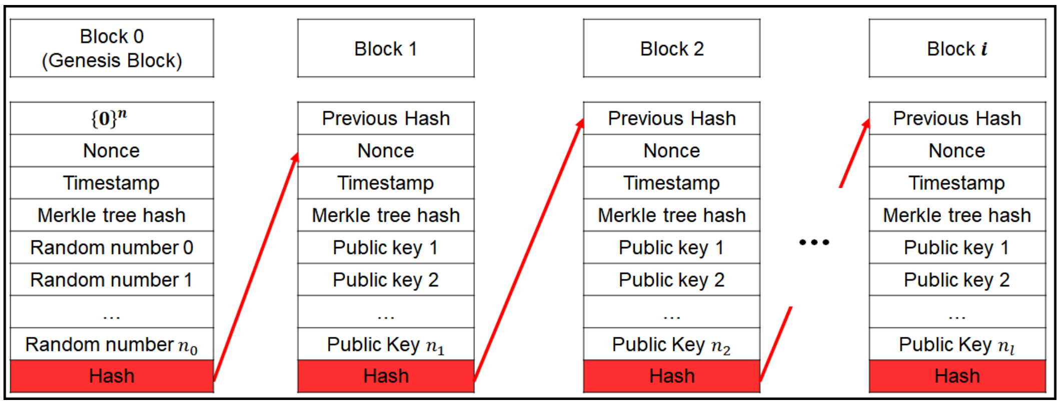

Figure 2: Full Structure of QChain

Equation 4.1 defines error-reconciliation function. In section 4.3, we will introduce detail 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 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, $n^{-1} \mod q = 12277$, $\gamma^{-1} \mod q = 8778$, $\omega^{-1} \mod q = 1254$.

The QChain scheme is described as follows:

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

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

$$\begin{aligned} r_{1,i}, r_{2,i} &\leftarrow \chi_\sigma; \\ y_{1,i}, y_{2,i} &\overset{\$}{\leftarrow} \mathcal{R}_q^k; \\ a_i \overset{\$}{\leftarrow} \mathcal{R}_q; \quad \hat{a}_i &\leftarrow \mathsf{NTT}(a_i); \\ \hat{r}_{1,i} \leftarrow \mathsf{NTT}(r_{1,i}); \quad \hat{r}_{2,i} &\leftarrow \mathsf{NTT}(r_{2,i}); \\ \hat{y}_{1,i} \leftarrow \mathsf{NTT}(y_{1,i}); \quad \hat{y}_{2,i} &\leftarrow \mathsf{NTT}(y_{2,i}); \\ \hat{p}_i &\leftarrow \hat{r}_{1,i} - \hat{a}_i * \hat{r}_{2,i}; \\ \hat{t}_i &\leftarrow \hat{a}_i * \hat{r}_{1,i} + \hat{r}_{2,i}; \end{aligned}$$

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

- $\mathsf{QChain.GenesisBlock.Setup}()$: Genesis block is the first block of QChain. We also call 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.

$$\begin{aligned} nonce \overset{\$}{\leftarrow} \{0,1\}^n; rand_i \overset{\$}{\leftarrow} \{0,1\}^n; \\ where, 0 \le i \le 2^{10} \\ timestamp \leftarrow \textbf{current time}; \end{aligned}$$

- $\mathsf{QChain.GenesisBlock.Merkle}()$: We construct Merkle hash tree after $\mathsf{QChain.GenesisBlock.Setup}()$ using random number $rand_i$, $timestamp$, hash function $H()$, and 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:

$$\begin{aligned} &\mathbf{pk}_i\mathbf{Info.} = \\ &rand_i || H(i) || H(i) || timestamp_i || Sign(rand_i) \end{aligned}$$

Using $\mathbf{pk}_i$ **Info.**, 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 top hash value H_{root} using each hash value of leaf nodes.

- $\mathsf{QChain.GenesisBlock.Final}()$: We finally construct genesis block in this final algorithm. To make a previous hash of block 1, QChain needs a hash value of genesis block. Previous hash value computes as follows:

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

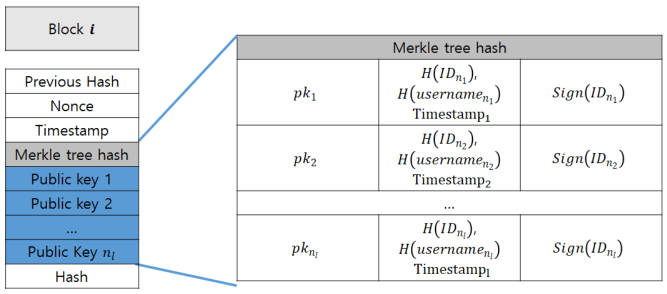

Figure 3: Detailed Structure of QChain

• $\mathsf{QChain.User.Setup}(pk_i, H_{root})$: In user setup algorithm, it is similar $\mathsf{QChain.GenesisBlock.Setup}()$ algorithm. User setup algorithm as follows:

$$\begin{aligned} \text{Previous hash} &\leftarrow H_{Block0}; \\ nonce &\xleftarrow{\$} \{0,1\}^n; \\ pk_i \leftarrow \textbf{User public key} &\in \{0,1\}^n; \\ where, 0 \leq i &\leq l \leq 2^{10} \\ timestamp_i &\leftarrow \textbf{current time}; \end{aligned}$$

• $\mathsf{QChain.User.Add}(ID_i, Username_i, sk_i, pk_i)$: After the genesis block has made by the previous algorithm, we add information about user's public keys as follows:

$$\begin{aligned} H(ID_i), \quad ID_i &\leftarrow \text{User ID}; \\ H(Username_i), \quad Username_i &\leftarrow \text{Username}; \\ (\hat{r}_{1,i}, \hat{r}_{2,i}, \hat{y}_{1,i}, \hat{y}_{2,i}) &\leftarrow sk_i; \\ y_{1,i} \leftarrow \mathsf{NTT}^{-1}(\hat{y}_{1,i}); \quad y_{2,i} &\leftarrow \mathsf{NTT}^{-1}(\hat{y}_{2,i}); \\ (\hat{a}_i, \hat{p}_i) \leftarrow pk_i; \quad a_i &\leftarrow \mathsf{NTT}^{-1}(\hat{a}_i); \\ c_i \leftarrow H(a_i y_{1,i} + y_{2,i}, ID_i); \quad \hat{c}_i &\leftarrow \mathsf{NTT}(c) \\ r_{1,i} \leftarrow \mathsf{NTT}^{-1}(\hat{r}_{1,i}); \quad r_{2,i} &\leftarrow \mathsf{NTT}^{-1}(\hat{r}_{2_i}); \\ Sign(ID_i, a_i, r_{1,i}, r_{2,i}); \end{aligned}$$

using ID_i and $Username_i$, 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 user of each block is 2^{10}. Because we restrict maximum depth of Merkle hash tree due to complexity reason. We will explain in detail in section 4.2. The $Sign()$ algorithm is modified GLP signature scheme.

• $\mathsf{QChain.User.Verify}(ID_i, pk_i, Sign(ID_i))$: To verify the public key pk_i and $Sign(ID_i)$ of user, using verify algorithm $Verify()$. User verify algorithm as follows:

$$\begin{aligned} \hat{a}_i, \hat{t}_i &\leftarrow pk_i; \\ a_i \leftarrow \mathsf{NTT}^{-1}(\hat{a}_i); \quad t_i &\leftarrow \mathsf{NTT}^{-1}(\hat{t}_i); \\ z_{1,i}, z_{2,i}, \hat{c}_i &\leftarrow Sign(ID_i); \\ c_i &\leftarrow \mathsf{NTT}^{-1}(\hat{c}_i); \\ Verify(ID_i, z_{1,i}, &z_{2,i}, c_i, a_i, t_i); \end{aligned}$$

using pk_i and $Sign(ID_i)$ which are the public parameter, we can easily verify the user.

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

$$\begin{aligned} (\hat{a}_i, \hat{p}_i, \hat{t}_i) &\leftarrow pk_i; \\ (a_i, p_i, t_i) \leftarrow (\mathsf{NTT}^{-1}(\hat{a}_i), &\mathsf{NTT}^{-1}(\hat{p}_i), \mathsf{NTT}^{-1}(\hat{t}_i)); \\ e_1, e_2, e_3 &\leftarrow \chi_\sigma; \\ \hat{e}_1 \leftarrow \mathsf{NTT}(e_1); \quad \hat{e}_2 &\leftarrow \mathsf{NTT}(e_2); \\ \hat{m} &\leftarrow m \cdot \left\lfloor \frac{q}{2} \right\rfloor; \\ (\hat{c}_1, \hat{c}_2) \leftarrow (\hat{a}_i * \hat{e}_1 + \hat{e}_2, \quad &\hat{p}_i * \hat{e}_1 + \mathsf{NTT}(e_3 + \hat{m})); \end{aligned}$$

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

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

$$\begin{aligned} \hat{r}_{2,i} &\leftarrow sk_i; \\ (\hat{c}_1, \hat{c}_2) &\leftarrow c; \\ m' &\leftarrow \mathsf{NTT}^{-1}(\hat{c}_1 * \hat{r}_2 + \hat{c}_2); \\ m &\leftarrow \mathsf{Decode}(m'); \end{aligned}$$

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

$$\mathsf{Decode}(m) := \left\lceil \frac{2}{q} \cdot m \cdot \lfloor q/2 \rfloor \right\rfloor \cdot \left\lfloor \frac{q}{2} \right\rfloor \qquad (1)$$

We design QChain scheme contained 10 algorithms. In Figure 3 describes detail structure of each block of QChain. In a structure of QChain, each block consists of previous hash, nonce, timestamp, a public key of the user, hash value of the block, and Merkle hash tree. The public key and secret key of users are based on Ring-LWE key generation scheme. Users can communicate application data using public key cryptosystem based on Ring-LWE scheme.

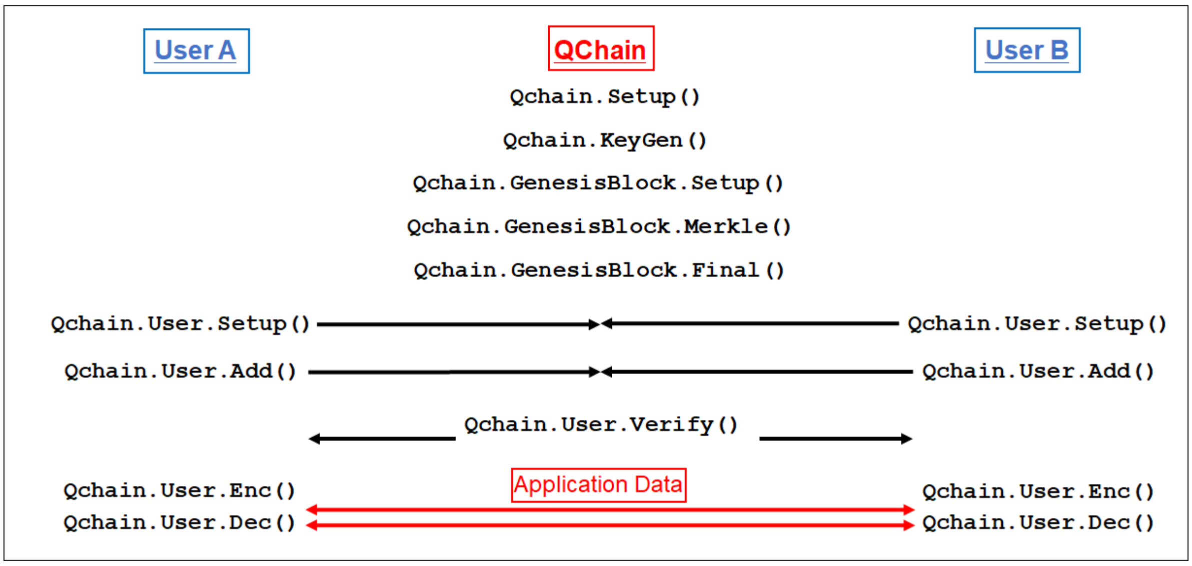

Figure 4: Protocol of QChain and User

4.2 Merkle Hash Tree

Figure 5 shows example of QChain Merkle hash tree with depth 2. We restrict maximum 2^{10} user 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. QChain Merkle hash tree consist of $2^{10}-1$ nodes. Therefore, $2^{10}-1$ hash operations are needed to generate H_{root} hash value. $2^{10}-1$ hash operation is reasonable computational power and efficient for users. The complexity of searching pk_i is $O(\log_2(n))$ in average case of each block. QChain.User.Add() algorithm has also same complexity $O(\log_2(n))$.

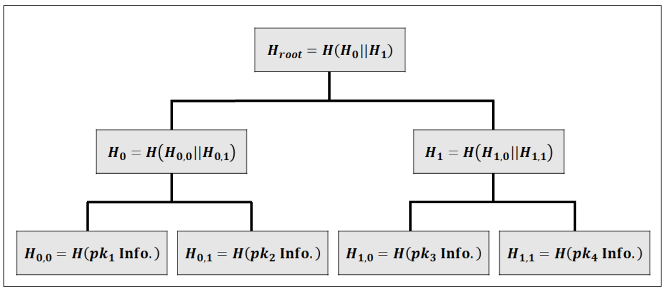
$H_{root} = H(H_0 || H_1)$
$H_0 = H(H_{0,0} || H_{0,1})$
$H_1 = H(H_{1,0} || H_{1,1})$
$H_{0,0} = H(pk_1\ \text{Info.})$
$H_{0,1} = H(pk_2\ \text{Info.})$
$H_{1,0} = H(pk_3\ \text{Info.})$
$H_{1,1} = H(pk_4\ \text{Info.})$

Figure 5: Example of QChain Merkle Hash Tree

4.3 Signature

In section 2, we describe GLP signature scheme. To efficient use, we modify GLP signature scheme. Algorithm 2 describes modified lattice-based GLP signature scheme. We integrate NTT for polynomial multiplication and addition. Modified GLP signature scheme is used for QChain.GenesisBlock.Merkle(), QChain.User.Add $(ID_i, Username_i, sk_i)$, 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     : $H : \{0,1\}^* \to D_{32}^n$
1  Sign$(\mu, \mathbf{a}, \mathbf{r}_1, \mathbf{r}_2)$
2  begin
3      $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_q^k$;
4      $\mathbf{c} \leftarrow H(\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2, \mu)$;
5      $\hat{\mathbf{c}} = \mathsf{NTT}(\mathbf{c})$;
6      $\hat{\mathbf{z}_1} \leftarrow \hat{\mathbf{r}}_1 * \hat{\mathbf{c}} + \hat{\mathbf{y}}_1$;
7      $\hat{\mathbf{z}_2} \leftarrow \hat{\mathbf{r}}_2 * \hat{\mathbf{c}} + \hat{\mathbf{y}}_2$;
8      $\mathbf{z}_1 \leftarrow \mathsf{NTT}^{-1}(\hat{\mathbf{z}_1})$;
9      $\mathbf{z}_2 \leftarrow \mathsf{NTT}^{-1}(\hat{\mathbf{z}_2})$;
10     if $z_1 \notin \mathcal{R}_q^{k-32}$ or $z_2 \notin \mathcal{R}_q^{k-32}$ then
11         go to line 3;
12     else
13         return $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$;
14     end
15 end
16 Verify$(\mu, \mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{a}, \mathbf{t})$
17 begin
18     if $z_1, z_2 \in \mathcal{R}_q^{k-32}$ then
19         $c \neq H(\mathbf{a}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{t}\mathbf{c}, \mu)$;
20         return reject;
21     else
22         return success;
23     end
24 end
```

5 Comparison with X.509 v3

In Figure 4 shows simplified protocol of QChain between 2 users. The first QChain operator initiates genesis block (block 0). The operator has 5 steps algorithms. QChain.Setup() sets the parameter of QChain. QCh ain.KeyGen() makes a public key and secret key of users. Then, QChain.Genesis Block.Setup(), QChain. GenesisBlock.Merkle(), and QChain. GenesisBlock.Final() algorithms operate genesis block. After generating genesis block, QChain makes next block called Block 1. To register public key, users set QChain. User.Setup() algorithm. Users can register the public key with algorithm QChain.User.Add(). Users 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.

Table 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 public key, revoke the lost key. The viewpoint of comparison is connection, non-repudiation, revocation, scalability, and model.

Table 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 used for current international standard must keep online states in server side (TTP). 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. In 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. 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 creating blockchain for 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 the 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 to TTP. 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. Also, 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.

6 Conclusion and Future Work

In this paper, we construct QChain which is decentralized PKI system using blockchain technique. QChain is a quantum-resistant PKI system against the quantum adversary. We combine blockchain technique and PQC primitive which is lattice-based cryptography. To efficient design QChain, we use NTT operations in polynomial multiplication and addition. QChain uses GLP signature scheme based on Ring-LWE problem. We also modify GLP signature scheme using NTT operations. Finally, we compare currently used X.509 v3 PKI system with our QChain.

As future work, several directions should be explored from here. First, we will implement QChain using C–language as an open source project. The goal of this project is efficient and secure decentralized PKI system. Second, we will merge hash function and lattice-based key exchange protocol such as BCNS[17], Frodo[18], and NewHope[19]. To make secure communication against the quantum adversary, we have to combine quantum secure key exchange protocols. we will also add SHA3[20] as secure hash function. Third, we need to precise security proof of QChain scheme.

Acknowledgements

This work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00555, Towards Provable-secure Multi-party Authenticated Key Exchange Protocol based on Lattices in a Quantum World) and National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2015R1A2A2A01006812, Design and Security Analysis of Novel Lattice-based Fully Homomorphic Signatures Robust to Quantum Computing Attack).

References

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

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

[3] 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.

[4] 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.

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

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

[7] 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.

[8] L. Axon and M. Goldsmith, "Pb-pki: a privacy-aware blockchain-based pki," in *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications*, SCITEPRESS, 2016.

[9] R. El Bansarkhani and J. Sturm, "An efficient lattice-based multisignature scheme with applications to bitcoins," in *in International Conference on Cryptology And Network Security–CANS'16*, pp. 140–155, Springer, 2016.

[10] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *IEEE Symposium on Security and Privacy–IEEE S&P'16*, pp. 839–858, IEEE, 2016.

[11] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *International Conference on IEEE Open and Big Data–IEEE OBD'16*, pp. 25–30, IEEE, 2016.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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