Blockchain-based Decentralized Key Management System with Quantum Resistance*

Hyeongcheol An[1], Rakyong Choi[2], and Kwangjo Kim[1,2]

[1] Graduate of School of Information Security,
[2] School of Computing,
Korea Advanced Institute of Science of Technology (KAIST),
Daejeon, Republic of Korea
{anh1026, thepride, kkj}@kaist.ac.kr

Abstract. The blockchain technique was first proposed called Bitcoin in 2008 and is 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 single point failure in 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, it is vulnerable to an attack by the quantum adversary. In this paper, we propose a blockchain-based key management system using quantum-resistant cryptography. Since it uses a GLP digital signature scheme, which is a secure lattice-based digital signature scheme. Therefore, our construction is based on quantum-resistant cryptography, it is secure against the attack of a quantum adversary and ensures long-term safety. In addition, we design a decentralized blockchain structure with extended X.509 certificate, and it is secure for the single point of failure.

Keywords: blockchain · quantum-resistant · key management system.

1 Introduction

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 [9]. Therefore, the emergence of the quantum computer is not theoretical but becomes practical. 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 (EC-DLP), and Integer Factorization Problem (IFP). However, DLP and IFP can be solved

* This work was 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)

within the polynomial time by Shor's algorithm [16] using the quantum computer. 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. 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.

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 [19] 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 with each entity. Therefore, the currently used centralized PKI system has problems with availability, due to the centralized CA. The most famous cryptocurrency, Bitcoin [15] 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. Therefore, each peer node operates both client and server on their network at the same time, since the blockchain technique is decentralized.

In this paper, we propose QChain, a quantum-resistant decentralized PKI system with extended X.509 certificate. To construct QChain, we combine the blockchain and lattice-based cryptography which is one of PQC primitives. QChain is a practical method for managing public key cryptography in a decentralized manner.

2 Related Work

2.1 Blockchain-based PKI

Emercoin(EMC) [10] 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 [5].

Matsumoto *et al.* suggest the Ethereum-based PKI system called IKP [14]. 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. In addition, IKP uses the quantum-resistant hash function called Ethash [17]. Ethereum is based on ECDSA signature algorithm, which is not secure against the quantum adversaries.

Yakubov *et al.* propose the blockchain-based PKI management framework [18] 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 [3] is the public and decentralized PKI system using blockchain technique and based on Namecoin. In revocation phase, they did not use Certificate Revocation List (CRL). They consider that Certcoin uses RSA accumulators, which is insecure against the quantum adversaries.

2.2 Lattice-based Signature Scheme

Compared to the PQC primitives lattice-based primitives are faster and have smaller signatures size than others. In general since ring-LWE-based digital signatures can provide the smallest time-data complexity compared with others. Akleylek *et al.* proposed the ring-LWE based signature scheme called Ring-TESLA [1]. 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. 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 $y \xleftarrow{\$} \mathcal{R}_q$. 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.* [6,7] published the GLP signature scheme based on ring-LWE problem. 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 $r_2 r_3 r_4 r_5$. In Section 3.1, we will describe modified GLP signature scheme.

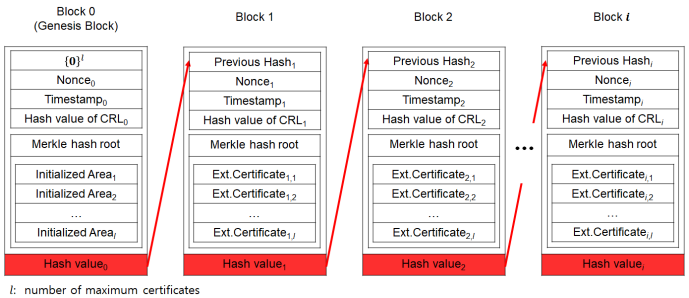

Fig. 1. Full Structure of QChain

3 Our Approach

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 the following sections, we describe the structure of scheme and modified GLP signature scheme. We integrate the modified GLP signature scheme, that is first approach in blockchain. 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.

3.1 Modified GLP Signature

GLP signature scheme is known to be faster than GPV [4] and LYU [12] scheme which belong to lattice-based signature scheme, and is believed to be secure against side-channel attacks till now. We has briefly described the GLP signature scheme in Section 2.2. In order to increase its performance, we modify the GLP signature scheme by integrating Number Theoretic Transformation (NTT) [8] like **Algorithm 1**. Let $\mathcal{R}_q^k$ be a subset of the ring $\mathcal{R}_q$, and that consists of all polynomials with coefficients in the range $[-k, k]$.

Algorithm 1: Modified GLP Signature

Signing Key : $\mathbf{r_1}, \mathbf{r_2} \xleftarrow{\$} \chi_\sigma$
Verification Key: $\mathbf{a} \xleftarrow{\$} \mathcal{R}_q$, $\hat{a} \leftarrow \mathsf{NTT}(\mathbf{a})$, $\hat{r_1} \leftarrow \mathsf{NTT}(\mathbf{r_1})$, $\hat{r_2} \leftarrow \mathsf{NTT}(\mathbf{r_2})$, $\hat{\mathbf{t}} \leftarrow \hat{a}\hat{r_1} + \hat{r_2}$
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 $\quad \mathbf{c} \leftarrow H(\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2, \mu)$; $\hat{\mathbf{c}} = \mathsf{NTT}(\mathbf{c})$;
5 $\quad \hat{\mathbf{z_1}} \leftarrow \hat{\mathbf{r}}_1 * \hat{\mathbf{c}} + \hat{\mathbf{y}}_1$; $\hat{\mathbf{z_2}} \leftarrow \hat{\mathbf{r}}_2 * \hat{\mathbf{c}} + \hat{\mathbf{y}}_2$;
6 $\quad \mathbf{z}_1 \leftarrow \mathsf{NTT}^{-1}(\hat{\mathbf{z_1}})$; $\mathbf{z}_2 \leftarrow \mathsf{NTT}^{-1}(\hat{\mathbf{z_2}})$;
7 $\quad$ **if** $z_1 \notin \mathcal{R}_q^{k-32}$ *or* $z_2 \notin \mathcal{R}_q^{k-32}$ **then**
8 $\quad\quad$ go to line 3;
9 $\quad$ **else**
10 $\quad\quad$ return $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$;
11 $\mathsf{Verify}(\mu, \mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{a}, \mathbf{t})$
12 **begin**
13 $\quad$ **if** $z_1, z_2 \in \mathcal{R}_q^{k-32}$ **then**
14 $\quad\quad$ $c \neq H(\mathbf{a}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{t}\mathbf{c}, \mu)$;
15 $\quad\quad$ return **reject**;
16 $\quad$ **else**
17 $\quad\quad$ return **success**;

3.2 Structure of QChain

Figure 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.* [13] which is secure against the quantum adversaries.

Figure 2 shows extended certificate for QChain. In the structure of QChain, each block consists of the previous hash, nonce, timestamp, a centralized public key of the user, hash value of the block, and Merkle hash tree. Users can communicate with the application data using the public key cryptosystem based on ring-LWE scheme. QChain certificate contain the following fields:

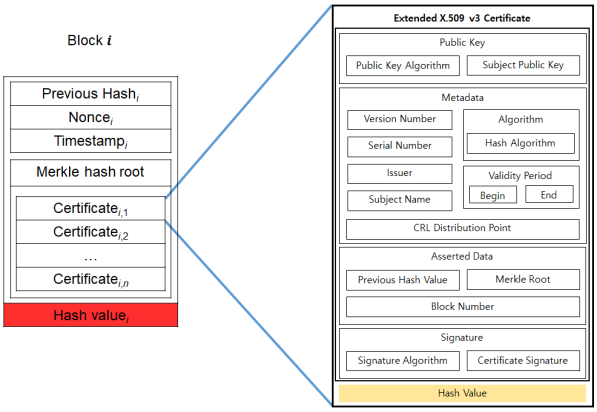

Fig. 2. Extended Certificate for QChain

- **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.
- **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, Merkle root, block number. 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.

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

$$\mathsf{NTT}(\mathbf{g}) = \hat{\mathbf{g}} = \sum_{i=0}^{1023} \hat{g}_i X^i \quad where, \hat{g}_i = \sum_{j=0}^{1023} \gamma^j g_j \omega^{ij}$$

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

$$\mathsf{NTT}^{-1}(\hat{\mathbf{g}}) = \mathbf{g} = \sum_{i=0}^{1023} \hat{g}_i X^i \quad where, g_i = n^{-1}\gamma^{-i} \sum_{j=0}^{1023} \hat{g}_j \omega^{ij}$$

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 $\sigma = \sqrt{16/2} \approx 2.828$ [2].
• $\mathsf{QChain.KeyGen}(n, \sigma)$: 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.

$$nonce \xleftarrow{\$} \{0,1\}^n; rand_i \xleftarrow{\$} \{0,1\}^n; where, 0 \le i \le 2^{10}$$

• $\mathsf{QChain.GenesisBlock.Merkle}()$: We construct Merkle hash tree 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 $Username_i = i$. 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 operates as follows:

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

$$nonce \xleftarrow{\$} \{0,1\}^n; pk_i \leftarrow \textbf{User public key} \in \{0,1\}^n; where, 0 \leq i \leq l \leq 2^{10}$$

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

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

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 the memory complexity. The $Sign()$ algorithm is a modified GLP signature scheme.

• QChain.User.Verify($ID_i, pk_i, Sign(Cert_i)$): To verify the public key pk_i and $Sign(Cert_i)$ of the user, using the verify algorithm $Verify()$. The user verify algorithm runs 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 public parameters pk_i and $Sign(ID_i)$, we can easily verify the user.

• QChain.User.Enc(pk_i, m): To encrypt a message $m \in \mathcal{R}_2$, the encryption algorithm runs 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 a user public key pk_i and message m.

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

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

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

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

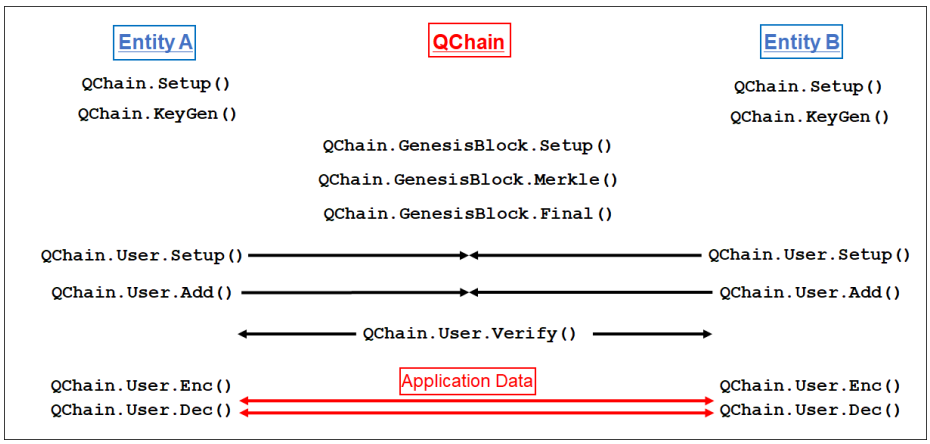

Fig. 3. A typical use case of QChain to setup secure communication

Figure 3 illustrates a typical use case of QChain to setup secure communication. The first QChain operator initiates genesis block (block 0). The operator has five-step algorithms. QChain.Setup() sets the parameter of QChain.KeyGen() makes a public and a private key of users. Then, QChain.Genesis Block.Setup(), QChain.GenesisBlock.Merkle(), and QChain.GenesisBlock.Final() algorithms operate to generate the genesis block. 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.User.Enc() and QChain.User.Dec(), users can communicate application data securely with each other.

4 Security Analysis

4.1 Generic Attack

Grover *et al.* suggest the database search algorithm called Grover's algorithm [5]. 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 [11]. 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 $2^{0.298n_2+O(n_2)}$ [11]. 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 [16].

4.2 Feature Analysis

PKI system is required as register key or domain, update and look up, and revocation of the public key.

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: 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 CRL. Similarly, QChain can manage the revocation list by 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.

4.3 Comparision with Related Work

We compare the features between our construction and related work, such as Certcoin, IKP, and Emercoin. Table 1 shows the comparison of QChain and related work. In dependence on existing cryptocurrency system, Certcoin is based on Namecoin, which is forked from Bitcoin. Emercoin [10] is also based on Peercoin. Lastly, IKP [14] is based on Ethereum smart contract platform.

Table 1. Comparison of QChain and Related Work

System	**QChain**	Certcoin [3]	IKP [14]	Emercoin [10]
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
Complexity on Signature using Quantum Computer	$2^{0.268n+O(n)}$	polynomial-time[1]	polynomial-time[1]	polynomial-time[1]
Hash Function	Not Specific	SHA256	Ethash	SHA256
Revocation Method	CRL, Timestamp	Timestamp	Smart Contract	Update by Administrator

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

We extend the X.509 v3 certificate with extension fields. Certcoin also extends the same approach. However, IKP and Emercoin do 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 primitives. Other construction uses ECDSA and RSA digital signature. In other words, Certcoin, IKP, and Emercoin are not secure against quantum adversaries. 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 timestamps without CRL, which makes the disadvantage. Therefore, user needs 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.

5 Concluding Remarks

This paper proposes QChain, which is a decentralized PKI system that uses the blockchain technique based on the ring-LWE scheme. QChain provides a quantum resistant PKI system secure against the quantum adversaries who will appear in the near future by combining the blockchain technique and one of PQC primitives, lattice-based cryptography. Our construction uses extended X.509 certificate. Therefore, we can easily integrate current X.509 standards. 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. 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 sphere-sieve 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 as an open source project. Our implementation needs the consensus algorithm in the validating blocks. QChain is designed to release single-point of failure in the current X.509 v3-based key management system, that works like a kind of decentralized PKI. Thus, QChain can fit the consortium or private blockchain applications. In the practical implementation, SHA3 or other secure and efficient hash functions can be considered.

References

1. Akleylek, S., Bindel, N., Buchmann, J., Krämer, J., Marson, G.A.: An efficient lattice-based signature scheme with provably secure instantiation. In: International Conference on Cryptology in Africa–AFRICACRYPT'16. pp. 44–60. Springer (2016)
2. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: 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)

3. Fromknecht, C., Velicanu, D., Yakoubov, S.: A decentralized public key infrastructure with identity retention. In: Cryptology ePrint Archive, Report 2014/803 (2014), http://eprint.iacr.org/2014/803
4. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth annual ACM symposium on Theory of computing. pp. 197–206. ACM (2008)
5. Grover, L.K.: 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). https://doi.org/10.1145/237814.237866, http://doi.acm.org/10.1145/237814.237866
6. Güneysu, T., Lyubashevsky, V., Pöppelmann, T.: 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)
7. Güneysu, T., Lyubashevsky, V., Pöppelmann, T.: 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)
8. Güneysu, T., Oder, T., Pöppelmann, T., Schwabe, P.: Software speed records for lattice-based signatures. In: International Workshop on Post-Quantum Cryptography–PQCrypto'13. pp. 67–82. Springer (2013)
9. IBM Research: IBM Q experience. https://www.research.ibm.com/ibm-q/ (2018), [Online; accessed 20-Mar.-2018]
10. Khovayko, O.: Emercoin. https://emercoin.com (2018), [Online; accessed 15-May.-2018]
11. Laarhoven, T., Mosca, M., Van De Pol, J.: Finding shortest lattice vectors faster using quantum search. Designs, Codes and Cryptography **77**(2-3), 375–400 (2015)
12. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 738–755. Springer (2012)
13. Lyubashevsky, V., Peikert, C., Regev, O.: 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)
14. Matsumoto, S., Reischuk, R.M.: IKP: Turning a PKI around with blockchains. In: Cryptology ePrint Archive, Report 2016/1018 (2016), http://eprint.iacr.org/2016/1018
15. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
16. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings, Annual Symposium on Foundations of Computer Science–FOCS'94. pp. 124–134. IEEE (1994)
17. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper **151** (2014)
18. Yakubov, A., Shbair, W., Wallbom, A., Sanda, D., 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)
19. Yee, P.: Updates to the internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile (2013)