Thesis for Master's Degree

New Blockchain Consensus Algorithm for Energy Efficiency and Quantum Resistance

Hyeongju Kim

School of Electrical Engineering and Computer Science

Gwangju Institute of Science and Technology

2024

석사학위논문

에너지 절약과 양자내성을 지닌 새로운 블록체인 합의 알고리즘

김형주

전기전자컴퓨터공학부

광주과학기술원

2024

New Blockchain Consensus Algorithm for Energy Efficiency and Quantum Resistance

Advisor: Heung-No Lee

by

Hyeongju Kim

School of Electrical Engineering and Computer Science

Gwangju Institute of Science and Technology

A thesis submitted to the faculty of the Gwangju Institute of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the School of Electrical Engineering and Computer Science

Gwangju, Republic of Korea

Dec 13, 2023

Approved by [illegible]

Professor Heung-No Lee

Committee Chair

New Blockchain Consensus Algorithm for Energy Efficiency and Quantum Resistance

Hyeongju Kim

Accepted in partial fulfillment of the requirements for

the degree of Master of Science

Dec 13, 2023

Committee Chair [illegible]

Prof. Heung-No Lee

Committee Member [illegible]

Prof. Euiseok Hwang

Committee Member [illegible]

Prof. Sunbeom So

MS/EC 20211186 Hyeongju Kim. New Blockchain Consensus Algorithm for Energy Efficiency and Quantum Resistance. School of Electrical Engineering and Computer Science. 2024. 35p. Advisor: Prof. Heung-No Lee.

Abstract

In recent times, blockchain has captured the attention of diverse entities, including businesses, institutions, and organizations, all of which underscore the importance of decentralization. Blockchain, a distributed computing technology heavily reliant on cryptography, empowers users to generate and validate blocks on an equal footing without the intervention of a central server. Each user employs their cryptographic key as an identification method for this purpose. Due to these characteristics, cryptographic security assumes significant importance in the realm of blockchain. Regrettably, the current digital signature technology utilized for transaction generation in blockchain is vulnerable to the easy compromise of private keys through quantum computing algorithms. Consequently, there arises a pressing need for research on cryptographic systems that are secure against quantum computers. This thesis introduces the application of post-quantum digital signature algorithms in blockchain consensus protocols to effectively address this vulnerability.

MS/EC 김형주. 에너지 절약과 양자내성을 지닌 새로운 블록체인 합의 알고리즘.
20211186 전기전자컴퓨터공학부. 2024. 35p. 지도교수: 이 흥 노.

국 문 요 약

블록체인은 탈중앙성을 지향하는 여러 기업, 기관, 단체, 등의 주목을 받고 있다. 블록체인은 암호학에 크게 의존하는 분산 컴퓨팅 기술로, 중앙 서버의 개입 없이 모든 사용자는 각자의 암호키를 ID로 하여 동등한 자격으로 블록을 생성하고 검증할 수 있게 한다. 이러한 이유로 블록체인에서 암호 안전성은 상당히 중요하게 여겨진다. 하지만 현재 블록체인에서 트랜잭션을 발생할 때 사용되는 전자서명 기술은 양자컴퓨팅 알고리즘으로 비밀키를 쉽게 탈취하는 것이 가능하다. 따라서 양자컴퓨터에도 안전한 암호시스템에 대한 연구가 필요하며, 이 논문에서는 양자 후 전자서명 알고리즘과 함께 이를 블록체인 합의 프로토콜에서의 응용한 내용을 소개할 것이다.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Blockchain and consensus algorithms

Blockchain is a distributed ledger technology that stores and publicly shares data in blocks. Each block forms an immutable, tamper-proof chain structure by including information about the hash value of the previous block. Blockchains utilize predetermined consensus algorithms for block generation, ensuring an unforgeable connected structure. The most well-known consensus algorithm, introduced in the 2008 Satoshi Nakamoto paper[1], is Proof-of-Work (PoW). Also known as mining, this algorithm employs hash functions like SHA-256 and involves continuously changing the nonce until a hash value meeting specific conditions is found. PoW has several issues, and this paper will primarily address the following two:

- **Energy consumption.** Nodes that generate blocks using PoW receive agreed-upon tokens. Consequently, users worldwide compete using computing resources, resulting in a significant amount of energy being consumed for blockchain mining. According to CBECI statistics, the energy consumption for Bitcoin mining in 2022 was approximately 107TWh [2], comparable to the annual electricity consumption of New York City.

- **Centralization due to specialized equipment.** The computational structure of changing the nonce while performing the hash algorithm can be specialized through Application-Specific Integrated Circuits (ASICs). Nodes equipped with ASICs can find answers much faster than GPUs used by regular nodes. Consequently, block generation opportunities become concentrated around nodes that own large quantities of ASICs, undermining the decentralization, a crucial characteristic of blockchain.

The subsequent sections of this chapter will discuss technologies related to resolving these two issues.

1.2 Error Correction Code Proof-of-Work

In the year 2020, HN Lee et. al. introduced the Error Correction Code Proof-of-Work (ECCPoW) consensus algorithm in their paper[3] to address the issue of blockchain centralization caused by ASIC devices. Unlike traditional Proof-of-Work (PoW) puzzles that involve simple hash calculations, ECCPoW is an algorithm that solves puzzles based on coding theory, making it challenging to design ASICs for. In each block generation step, a new Low-Density Parity Check (LDPC) matrix is randomly generated, and mining nodes perform the task of finding a hash vector decodable against the same LDPC matrix. As a result, they output a codeword corresponding to the nonce of PoW as proof of work.

```
Algorithm 1: ECCPoW
  Data: a previous block header $pbh$
  Result: $(success, codeword)$
1 $h \leftarrow \mathcal{H}ash(pbh)$;
2 Construct a parity check matrix using $h$;
3 while True do
4     Generate a random nonce;
5     Create a hash vector using the nonce;
6     Check if the hash vector can be decoded using the parity check matrix;
7     if Decoding is successful then
8         return $(success, codeword)$;
9         break;
```

1.3 Verifiable Random Functions

Verifiable Random Functions (VRFs) are cryptographic concepts that generate random numbers while simultaneously providing verifiable evidence of the correct generation. This technology has been researched in the field of cryptography for a long time. Silvio Micali has made significant contributions to the advancement of VRFs and applied the cryptographic primitive of VRF to blockchain, developing a consensus node selection algorithm through a lottery [4]. The mechanism applied to the blockchain consensus lottery system is illustrated in Figure 1.1. The proof generation and verification of VRF involve the concept of digital signatures and can be summarized very simply as follows:

1. **Key Generation:** Generate a secret key and a public key, typically using elliptic curve-based cryptographic techniques like Elliptic Curve Digital Signature Algorithm (ECDSA).

2. **Proof Generation:** Use the secret key to encrypt a randomly chosen digest to

create the proof.

3. **Random Number Generation:** Input the proof into a hash function to generate a random number.

4. **Verification:** Decrypt the proof with the public key and compare the hashed value with the generated random number.

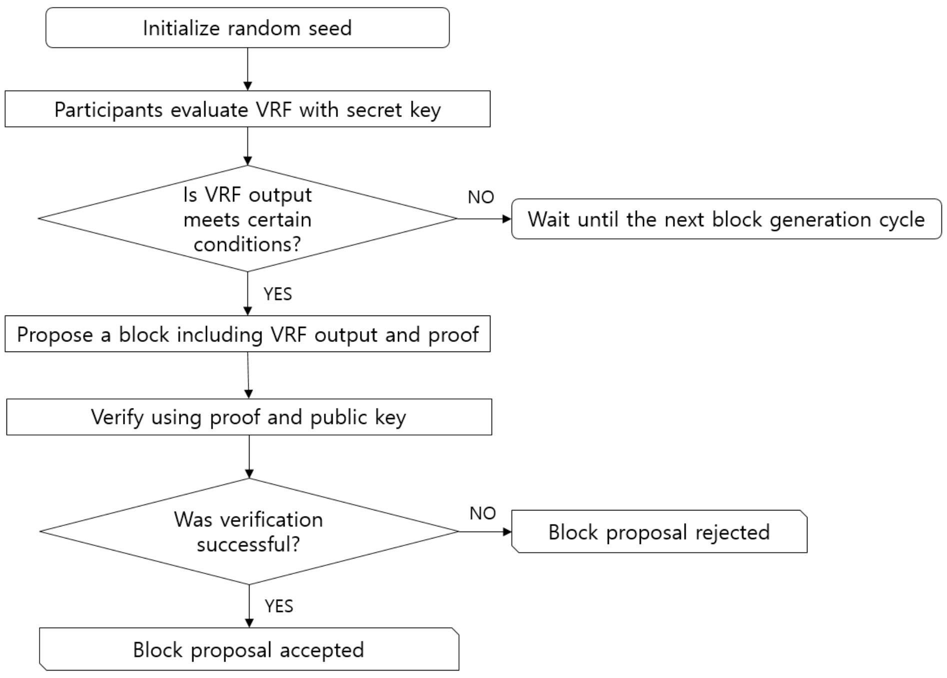

Initialize random seed
Participants evaluate VRF with secret key
Is VRF output meets certain conditions?
NO
Wait until the next block generation cycle
YES
Propose a block including VRF output and proof
Verify using proof and public key
Was verification successful?
NO
Block proposal rejected
YES
Block proposal accepted

Figure 1.1: VRF-based sortition mechanism.

Chapter 2

Background

2.1 Digital Signature algorithms

Digital signatures involve signing a message using a private key and verifying it with a public key. This concept evolved from the public-key distribution scheme introduced in the 1976 paper by Diffie and Hellman[5]. At that time, the Diffie-Hellman protocol was based on the Discrete Logarithm Problem (DLP). In 1977, Rivest, Shamir, and Adleman (RSA) introduced digital signature technology based on the difficulty of factoring large integers[6]. This protocol is still widely used on the internet today. Summarizing the process of signing with RSA, note that the practical RSA is much more complex for security reasons:

Key Generation:

1. Choose two large random primes p and q to create public and private keys, set $N = pq$, and $\phi = (p-1)(q-1)$.

2. Choose an arbitrary e that is smaller than ϕ and coprime to ϕ.

3. Calculate the value of d such that $(e \cdot d) \mod \phi = 1$.

4. Return public_key $= (N, e)$ and secret_key $= (N, d)$.

Signing (a message M, secret_key):

1. Hash the message using an algorithm like SHA-256: $M \leftarrow \text{Hash}(M)$.

2. Encrypt the hashed message using the secret key to create the signature: $\text{sig} = M^d \mod N$.

Verification (a message M, a signature sig, public_key):

1. Hash the message using an algorithm like SHA-256: $M \leftarrow \text{Hash}(M)$.

2. Decrypt the signature using the public key to calculate M': $M' = \text{sig}^e \mod N$.

3. Compare M and M', return True if they are equal, and False otherwise.

In 1985, Neal Koblitz and Victor S. Miller introduced Elliptic Curve Cryptography (ECC) in their paper [7]. ECC provides better efficiency and security than RSA and is widely used in modern security protocols such as blockchain. Security in ECC is based on the difficulty of the DLP when given two points P and Q on an elliptic curve over a finite field, where $Q = nP$. Generally, for a publicly known Generator point G, the private key is an arbitrary integer n, and the public key is nG. Signatures are derived from coordinates associated with a randomly generated point from G.

2.2 Quantum computer and Shor's algorithm

Classical computers represent information using bits, which are states with voltage and ground representing 0 and 1. However, quantum computers, which are gaining attention as the next generation of computers, utilize quantum mechanical properties such as superposition and entanglement to have multiple states known as Qubits.

Quantum computers can perform parallel processing of operations using the superposition of input values. According to the algorithm proposed by Peter Shor in 1994 [8], cryptographic algorithms based on the current Diffie-Hellman problem (DLP) and the difficulty of integer factorization become vulnerable.

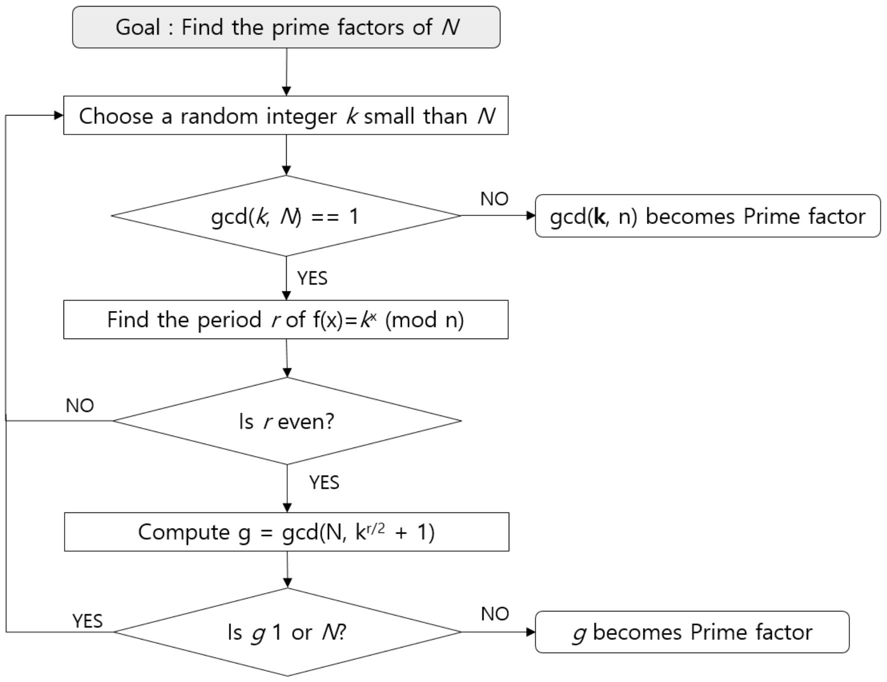

Goal : Find the prime factors of N
Choose a random integer k small than N
gcd(k, N) == 1
NO
gcd(k, n) becomes Prime factor
YES
Find the period r of f(x)=k^x (mod n)
NO
Is r even?
YES
Compute g = gcd(N, k^{r/2} + 1)
YES
Is g 1 or N?
NO
g becomes Prime factor

Figure 2.1: Shor's algorithm

For example, Let us consider the factorization of 77 composed of two prime numbers, 7 and 11, using Shor's algorithm. First, choose an arbitrary number, let's say 5, smaller than 77. If, by good fortune, the greatest common divisor with 77 is not 1, we have immediately found a prime factor. Next step is to find the period of the function $f(x) = 5^x \mod 77$ (See figure 2.3).

The essence of Shor's algorithm lies in finding the period of the function, which can be efficiently achieved using the Quantum Fourier Transform (QFT). Other calculations, such as finding greatest common divisors, can be efficiently performed by classical

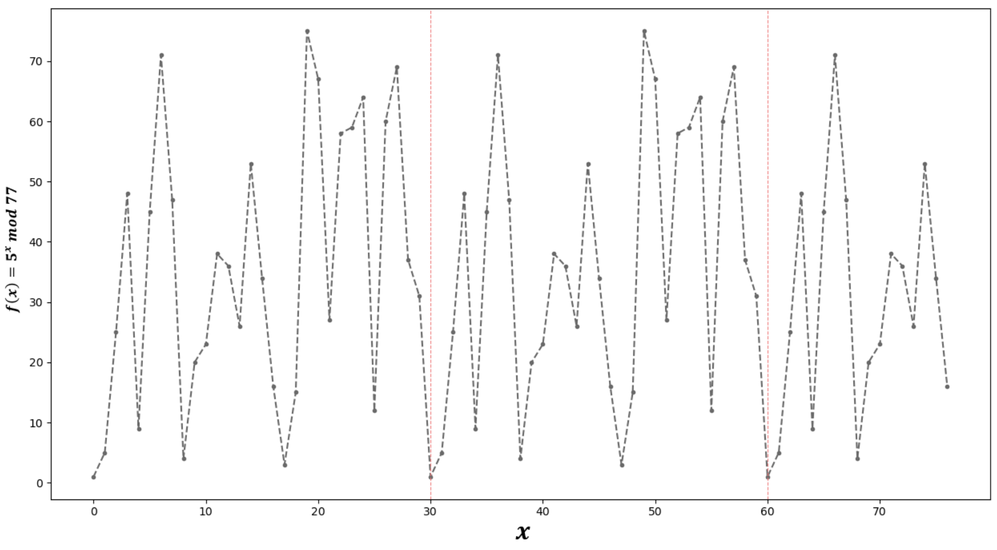


Figure 2.2: Period of function $f(x) = 5^x \mod 77$.

computers. In the example, the period is 30, and by calculating the greatest common divisor of 77 and $5^{15}+1$, we can confirm the desired prime factor 7. In this way, current cryptographic systems with periodicity can have their secret keys efficiently discovered within polynomial time by quantum computer algorithms.

The subsequent content will introduce Post-Quantum Cryptography, which is resilient to quantum computers.

2.3 Lattice-based Cryptography

2.3.1 Definition of lattice

A lattice is a set of points defined in an n-dimensional Euclidean space, forming a periodic structure. Since problems defined using lattices cannot be efficiently solved in polynomial time by quantum computers as well as classical computers, they are considered a promising scheme for next-generation cryptography. Mathematically, it is defined as follows.

Lattice. Let $\mathbf{b}_1, \cdots, \mathbf{b}_m \in \mathbb{R}^n$ be a n-dimensional m linearly independent basis vectors, then the set of all integer combination of $\mathbf{b}_i$ is called lattice.[9]

$$\Lambda(\mathbf{b}_1, \cdots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{m} x_i \mathbf{b}_i : x_i \in \mathbb{Z} \right\}$$

Alternatively, if we represent the basis in matrix form $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_n] \in \mathbb{R}^{n \times m}$ then the lattice can be defined as,

$$\Lambda(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}$$

Most lattice-based cryptography has a structure using a q-ary lattice.

q-ary lattice. Given integers q, m, n, and let a basis matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$. Then $\Lambda_q(\mathbf{B})$ denotes q-ary lattice if lattice Λ satisfying $q\mathbb{Z}^n \subseteq \Lambda \subseteq \mathbb{Z}^n$.

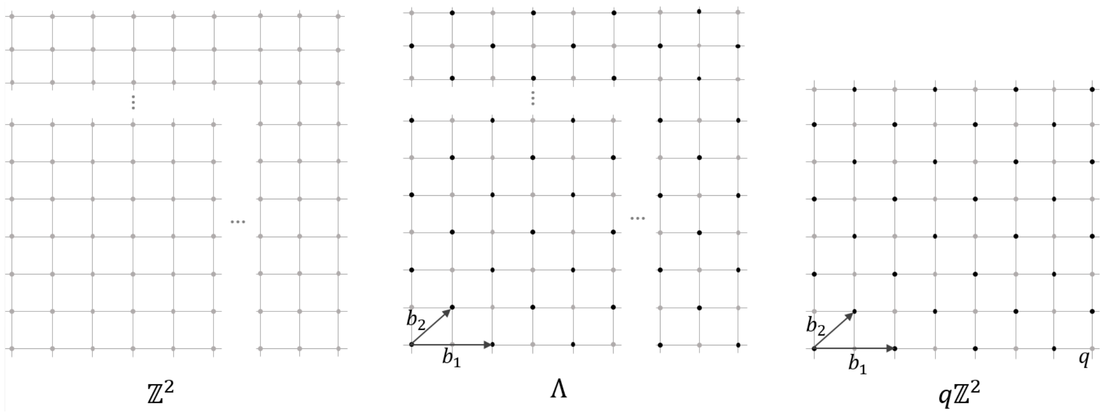


Figure 2.3: Lattice points in 2-dimension.

There exist two definition of n-dimensional q-ary lattices,

$$\Lambda_q(\mathbf{B}) = \left\{\mathbf{y} \in \mathbb{Z}^m : \mathbf{y} = \mathbf{B}^T\mathbf{s} \mod q \text{ for some } \mathbf{s} \in \mathbb{Z}^n\right\}$$

$$\Lambda_q^{\perp}(\mathbf{B}) = \{\mathbf{y} \in \mathbb{Z}^m : \mathbf{B}\mathbf{y} = 0 \mod q\} .$$

To aid understanding, let's consider the example of a 2-ary lattice below. Assume we have a 4 by 7 matrix B in a 4-dimensional space. Let s be $(1, 0, 0, 1)$.

$$B = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Then, $B^T s$ yields the following lattice point. And several points are generated in the same way for other s.

$$B^T s = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

In the context of code theory, interpreting q-ary lattice points, they are codewords obtained by a linear combination of the generating matrix B and the data vector s.

The second definition of q-ary lattice is the set of vectors y such that their multiplication by a matrix B results in zero, i.e., it is the kernel of B. In the case of the matrix B from the above example, y_i that satisfies the equation $By_i = 0 \mod 2$ are,

$$y_1 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad y_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad y_3 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

From the perspective of code theory, y_i corresponds to the codewords formed by B as the parity check matrix, so satisfies $By_i = 0$.

Dual lattice. Given an arbitrary lattice Λ, the dual lattice, denoted Λ^* is defined as,

$$\Lambda^* = \{\mathbf{x} \in \mathbb{R}^n : \forall \mathbf{v} \in \Lambda, \langle \mathbf{x}\ , \mathbf{v}\ \rangle \in \mathbb{Z}\}$$

It is easy to see that the above two q-ary lattices are dual to each other, namely $\Lambda_q(\mathbf{B}) = q \cdot \Lambda_q^{\perp}(\mathbf{B})^*$ and $\Lambda_q^{\perp}(\mathbf{B}) = q \cdot \Lambda_q(\mathbf{B})^*$. Now Let's see hardness problems based on lattice.

2.3.2 Hard lattice problems

The beginning of lattice-based cryptography can be traced to the seminal work of Ajtai in 1996 [10]. Ajtai demonstrated that the Shortest Vector Problem problem was NP-hard and he introduced a one-way hash function based on Short Integer Solutions problem. Subsequently, in 2005, Oded Regev introduced a new cryptographic system by presenting the lattice-based Learning With Errors(LWE) problem in his paper[11]. The LWE problem has been shown to be as hard as the worst-case lattice problem.

Shortest Vector Problem(SVP). Let $\lambda_1(\mathcal{L})$ be a minimum distance of a lattice $\mathcal{L}$ i.e., $\lambda_1(\mathcal{L}) := \min_{v \in \mathcal{L}} \|v\|$. Then SVP is to find a nonzero lattice vector $v \in \mathcal{L}$ such that $\|v\| = \lambda_1(\mathcal{L})$

Closest Vector Problem(CVP). Given a lattice $\mathcal{L}$ generated by basis B and given a target vector $t \in \mathbf{span}(B)$, t is not necessarily in $\mathcal{L}$. Then the CVP is to find the vector $v \in \mathcal{L}$ closest to t, minimizing the $\|v - t\|$.

Short Integer Solution(SIS). Given n-dimensional vectors $A = a_1, a_2, \cdots, a_m \in \mathbb{Z}_q^n$ in modulo q. SIS problem is to find nontrivial small vectors $z = z_1, z_2, \cdots, z_m \in \mathbb{Z}$ such that $Az = 0 \mod q$, i.e.,

$$z_1 a_1 + z_2 a_2 + \cdots + z_m a_m = 0.$$

This can be represented in lattice context as follows,

$$\mathcal{L}^{\perp}(A) = \{z \in \mathbb{Z}^m : Az = 0 \mod q\}.$$

Learning With Errors(LWE). Given a (secret) fixed vector $s \in \mathbb{Z}_q^n$ and randomly chosen vectors a in $\mathbb{Z}_q^n$. Let $e \in \mathbb{Z}_q$ be a random error vectors selected from distribution χ, then we set $b = \langle s, a \rangle + e$. There are two versions of LWE problems.

Search LWE : For m independent LWE samples $(a_1, b_i), (a_2, b_2), \cdots, (a_m, b_m)$, the problem is to find s with given LWE samples.

Decision LWE : Let $A_{s,\chi}$ denote the LWE distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. The problem is to distinguish LWE distribution $A_{s,\chi}$ from uniform distribution.

2.3.3 NTRU lattice

NTRU was first studied as a public key cryptosystem by Jeffrey Hoffstein et al., in 1996 [12]. Later, a digital signature algorithm using NTRU, was studied in 2003 [13].

Let the polynomial ring $R = \mathbb{Z}[X]/(X^N - 1)$ of degree less than n. Define R-module set such that $M_{h,q} = \{(u, v) \in R^2 | v \equiv u * h \mod q\}$ for $q \in \mathbb{Z}, h \in R$ (the notation $*$ denotes polynomial multiplication). The set of R-module $M_{h,q}$ becomes a full-rank NTRU lattice of $\mathbb{Z}^{2N}$.

For $f, g \in R$, if we set $h = g * f^{-1} \mod q$ so that $(f, g) \in M_{h,q}$, then one of the R-basis for generating this NTRU lattice is the bad basis $A_{h,q} = \left[\begin{array}{c|c} -h & I_n \\ \hline qI_n & O_n \end{array}\right]$, which typically have large coefficients. It will be used as the public key.

And if we find F and G that satisfy the NTRU equation $f * G - g * F = q$, then the other R-basis become good basis(short basis) $B_{f,g} = \left[\begin{array}{c|c} g & -f \\ \hline G & -F \end{array}\right]$, and it will be used as a secret key. These two R-basis $A_{h,q}$ and $B_{f,g}$ generate the same NTRU lattice $M_{h,q}$.

Notice that, each polynomial can be defined in anticirculant matrix form as follows.

$$M_f = \begin{pmatrix} f_0 & f_1 & f_2 & \cdots & f_{N-1} \\ -f_{N-1} & f_0 & f_1 & \cdots & f_{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -f_1 & -f_2 & -f_3 & \cdots & f_0 \end{pmatrix}$$

Therefore, each NTRU basis becomes a 2N by 2N matrix, and the matrix for the multiplication of two polynomials(i.e., $M_{(f*g \mod q)}$) is given by $M_f M_g$.

R-module. Let R be a ring and $\mathbf{M}$ be an abelian group under addition. R-module is a set equipped with two operations: addition and scalar multiplication operation associates each element of the R with an operation on the elements of the $\mathbf{M}$.

$$R \times \mathbf{M} \mapsto \mathbf{M}$$

Chapter 3

Post-Quantum Digital signatures

3.1 DURANDAL

In 2009, V. Lyubashevsky introduced a scheme that combines Schnorr's signature[14] with lattice context in his paper [15]. This concept is based on the Proof of Knowledge cryptography, prooving the knowledge of a vector with small weight associated to a given syndrome. It involves using a random public matrix H and the secret matrix S of small weight vectors. The signature is a proof of knowledge of the small weight matrix S based on a sparse challenge c. The signature itself takes the form of $z = y + cS$, where y is a random vector with a moderate weight, typically much higher than the weights of cS. The proof of knowledge relies on the fact that the verifier can be convinced of the prover's knowledge of the secret matrix S through the use of cS in the signature.

Nicolas Aragon et al. presented a variation of V. Lyubashevsky's approach, introducing an digital signature scheme based on rank metric in their paper Durandal[16]. In DURANDAL, they proposed an efficient way to randomize a signature in a rank metric context by extending the number of small weight secret vectors and adding another secret matrix S'. The signature is represented as $z = y + cS + pS'$, where p contributes to the extra randomization. This approach relaxes the conditions for the prover and enables the derivation of a randomized signature. DURANDAL is proofed in the EUF-CMA security model, reducing security to Rank Support Learning(RSL) problem. The following are definition of RSL problem and an algorithm that summarizes DURANDAL's signature scheme.

Rank Syndrome Decoding(RSD) problem. Given a full-rank parity-check matrix $H \in \mathbb{F}_{q^m}^{(n-k)\times n}$ of $[n,k]$ linear code, vector $s \in \mathbb{F}_{q^m}^{(n-k)}$ and integer r. It is hard to find a syndrome vector $e \in \mathbb{F}_{q^m}^{n}$ such that $\|e\| = r$ and $He^T = s^T$.

Rank Support Learning (RSL) problem.[17] Let H be an $(n-k)\times n$ random full-rank matrix over $\mathbb{F}_{q^m}$. Let $\mathcal{O}$ be an oracle which gives exactly N samples $(Hs_1^T, Hs_2^T, \ldots, Hs_N^T)$ where vector s_i is randomly chosen from random subspace E of $\mathbb{F}_{q^m}$. The RSL problem is to recover E given only access to the oracle. The corresponding decisional problem is to distinguish the pair (H, Hs_i) from the pair (H, Y) where Y is randomly chosen from $\mathbb{F}_{q^m}^{(n-k)\times N}$.

Algorithm 2: DURANDAL_Initialization

```
1 choose a random subspace $E$ of dimension $r$ of $\mathbb{F}_q^m$;
2 choose a random subspace $W$ of dimension $w$ of $\mathbb{F}_q^m$;
3 choose a random subspace $F$ of dimension $d$ of $\mathbb{F}_q^m$;
4 choose a filtered subspace $U$ of dimension $rd - \lambda$ of $EF$;
```

For filtered subspace U, it is to prevent recovering the E space through LDPC decoding[18], and it is recommended to see [16] for more details.

Algorithm 3: DURANDAL_Key generation

```
  Data: -
  Result: a public key $pk$, a secret key $sk$
1 choose an $(n-k)\times n$ ideal double circulant matrix $H$;
2 sample $l, l'$ vectors $s_i, s'_i$ respectively from the same support $E$;
3 $t_i \leftarrow H{s_i}^T$ and $t'_i \leftarrow H{s'_i}^T$;
4 $pk \leftarrow (H, T = \{t_1, t_2, \cdots, t_l\}, T' = \{t'_1, t'_2, \cdots, t'_{l'}\})$
    $sk \leftarrow (S = \{s_1, s_2, \cdots, s_l\}, S' = \{s'_1, s'_2, \cdots, s'_{l'}\})$;
5 return $(pk, sk)$
```

The definition of *ideal double circulant codes* that appears in the key generation step is as follows.

Ideal double circulant codes. Let $P(X) \in \mathbb{F}_q[X]$ be a irreducible polynomial of degree n and the vector $g_1, g_2 \in \mathbb{F}_{q^m}^n$. In the generator matrix $G = [A|B]$, we define $[2n, n]_{q^m}$ linear code C as ideal double circulant codes, If circulant matrices A and B are,

$$A = \begin{bmatrix} g_{10} + Xg_{11} + \cdots + X^{n-1}g_{1(n-1)} \bmod P \\ Xg_{10} + X^2g_{11} + \cdots + X^n g_{1(n-1)} \bmod P \\ \vdots \\ X^{n-1}g_{10} + X^n g_{11} + \cdots + X^{2n-2}g_{1(n-1)} \bmod P \end{bmatrix}$$

$$B = \begin{bmatrix} g_{20} + Xg_{21} + \cdots + X^{n-1}g_{2(n-1)} \bmod P \\ Xg_{20} + X^2g_{21} + \cdots + X^n g_{2(n-1)} \bmod P \\ \vdots \\ X^{n-1}g_{20} + X^n g_{21} + \cdots + X^{2n-2}g_{2(n-1)} \bmod P \end{bmatrix}$$

and we have $C = \left\{(xg_1, xg_2 \bmod P), x \in \mathbb{F}_{q^m}^n\right\}$. If g_1 is invertible so that $g = g_1^{-1}g_2 \bmod P$, it can be expressed as $C = \left\{(x, xg), x \in \mathbb{F}_{q^m}^n\right\}$.

Algorithm 4: DURANDAL_Sign

```
  Data: a message $m$, a secret key $sk$
  Result: a signature $\sigma = (z, F, c, p)$
1 set $x = Hy^T$ where $y \in (W + EF)^n$;
2 set $c = \mathcal{H}(x, F, m) \in F^{l'k}$;
3 through linear algebra, compute $p \in F$ ensuring the rank of sum $y + cS' + pS$
    is at most $w + rd - \lambda$;
4 $z \leftarrow y + cS' + pS$;
5 return $\sigma = (z, F, c, p)$
```

```
Algorithm 5: DURANDAL_Verification
  Data: a signature $(z, F, c, p)$, a message $m$, a public key $pk$
  Result: $True$ or $False$
1 if $\|zv\| > rd + w - \lambda$ then
2     return $False$;
3 else
4     if $\mathcal{H}(Hz^T - T'c^T + Tp^T, F, m) \neq c$ then
5         return $False$;
6     return $True$;
```

3.2 DILITHIUM

Dilithium [19] is currently the most promising candidate among NIST post-quantum secure digital signature algorithms. Dilithium is based on the problem of finding short vectors in lattices and employs the Fiat-Shamir with aborts Framework [15].

In the key generation phase, start by generating 256-bit random seeds ρ and ρ'. Then, use ρ to create matrix A with the ExpandA expansion function, and use ρ' to generate the secret key s_1, s_2 with coefficients uniformly distributed between $-\eta$ and η using the ExpandA function. The ExpandA function is a bit-string function that can extend the output to the desired length. For instance, $y \in R := \text{ExpandA}(x)$ represent a value with a uniform distribution in the set R for input x.

$$\text{ExpandA} : \text{seed } \rho \mapsto \text{matrix } A$$

Next, calculate $t = As_1 + s_2$ and use it to compute t_1 with the Power2Round function. Power2Round is a function that partitions elements of the set.

$$(r \mod {}^{+}q) - (r \mod {}^{\pm}2^d) \leftarrow \text{Power2Round}_q(r, d)$$

Finally, the public key is $pk = (\rho, t_1)$, and the secret key is $sk = (\rho, s_1, s_2, t)$.

Algorithm 6: Dilithium_Key Generation

```
Data: -
Result: a public key $pk$, a secret key $sk$
1 $\rho, \rho' \leftarrow 0, 1^{256}$;
2 $A \in R_q^{k\times l} :=$ ExpandA($\rho$);
3 $(s_1, s_2) \in S_\eta^l \times S_\eta^k :=$ ExpandA($\rho'$);
4 $t := As_1 + s_2$;
5 $t_1 :=$ Power2Round$_q(t, d)$;
6 $pk \leftarrow (\rho, t_1)$;
7 $sk \leftarrow (\rho, s_1, s_2, t)$;
8 return $(pk, sk)$
```

In the signing phase, the secret key t is split into t_1 and t_0 using the Power2Round function. Then, a 256-bit random seed r is generated to sample y, a set of coefficients with set $S_{\gamma_1 - 1}$, using the ExpandA function. Subsequently, calculate $w = Ay$ and compute w_1 using the HighBits function.

The HighBits(w, α) function, for $w := w \mod {}^+q$ and $w_0 := w \mod {}^\pm\alpha$, returns 0 if $w - w_0 = q - 1$ otherwise, it returns $(w - w_0)/\alpha$. In a similar context, the Decompose(w, α) function returns $(0, w_0 - 1)$ if $w - w_0 = q - 1$ and $(w - w_0/\alpha, w_0)$ otherwise.

$$\text{HighBits}(w, \alpha) = \begin{cases} 0, & \text{if } w - w_0 = q - 1 \\ (w - w_0)/\alpha, & \text{otherwise} \end{cases}$$

$$\text{Decompose}(w, \alpha) = \begin{cases} (0, w_0 - 1), & \text{if } w - w_0 = q - 1 \\ (w - w_0/\alpha, w_0), & \text{otherwise} \end{cases}$$

Next, use a hash function to compute $c = \mathcal{H}(\rho, t_1, w_1, \mu)$ and then use it to calculate $z := y + cs_1$. Compute (r_1, r_0) using the Decompose function. Repeat this process by changing the seed r until r_0, r_1 and z satisfy specific conditions.

Finally, compute h using the MakeHint function, and if ct_0 and h satisfy certain conditions, return the signature $\sigma := (z, h, c)$. The MakeHint(z, r, α) function generates a 1-bit hint to derive HighBits$(r + z, \alpha)$ from given parameters. It returns 0 if HighBits(r, α) equals HighBits$(r + z, \alpha)$ and 1 otherwise.

$$\text{MakeHint}(z, r, \alpha) = \begin{cases} 0, & \text{if HighBits}(r, \alpha) == \text{HighBits}(r + z, \alpha) \\ 1, & \text{otherwise} \end{cases}$$

Algorithm 7: Dilithium_Sign

```
Data: a message $m$, a secret key $sk$
Result: a signature $sig$
1  $A \in R_q^{k \times l} := \text{ExpandA}(\rho)$;
2  $t_1 := \text{Power2Round}_q(t, d)$;
3  $t_0 := t - t_1 \cdot 2^d$;
4  while $\|z\|_\infty \geq \gamma_1 - \beta$ or $\|r_0\|_\infty \geq \gamma_2 - \beta$ or $r_1 \neq w_1$ do
5      $r \leftarrow 0, 1^{256}$;
6      $y \in S^l_{\gamma_1 - 1} := \text{ExpandA}(r)$;
7      $w := Ay$;
8      $w_1 := \text{HighBits}_q(w, 2\gamma_2)$;
9      $c := \mathcal{H}(\rho, t_1, w_1, m)$;
10     $z := y + cs_1$;
11     $(r_1, r_0) := \text{Decompose}_q(w - cs_2, 2\gamma_2)$;
12 $h := \text{MakeHint}_q(-ct_0, w - cs_2 + ct_0, 2\gamma_2)$;
13 if $\|ct_0\|_\infty \geq \gamma_2$ or the number of 1's in $h$ is greater than $\omega$ then
14     go to 4
15 $\sigma \leftarrow (z, h, c)$;
16 return $\sigma$
```

In the verification phase, use the UseHint function, which restores HighBits$(r+z, \alpha)$ from the h, to verify whether the specific conditions from the signing phase are satisfied.

Algorithm 8: UseHint

```
Data: $y, r, \alpha$
Result: $(r_1 + 1) \mod {}^+m$ or $(r_1 - 1) \mod {}^+m$ or $r_1$
1 $m \leftarrow (q - 1)/\alpha$;
2 $(r_1, r_0) \leftarrow \text{Decompose}(r, \alpha)$;
3 if $y = 1$ and $r_0 > 0$ then
4     return $(r_1 + 1) \mod {}^+m$
5 else
6     if $y = 1$ and $r_0 \leq 0$ then
7         return $(r_1 - 1) \mod {}^+m$
8     return $r_1$
```

Algorithm 9: Dilithium_Verification

```
Data: a message $m$, a signature $\sigma$, a public key $pk$
Result: $True$ or $False$
1 $A := \text{ExpandA}(\rho)$;
2 $w_1 := \text{UseHint}(h, Az - ct_1 \cdot 2^d, 2\gamma_2)$;
3 $t_0 := t - t_1 \cdot 2^d$;
4 if $c = \mathcal{H}(\rho, t_1, w_1, m)$ and $\|z\|_\infty < \gamma_1 - \beta$ and the number of 1's in h is $\leq \omega$
   then
5     return $True$
6 else
7     return $False$
```

Recommended parameters :

$q = 8380417, d = 14, \gamma_1 = 523776, \gamma_2 = 261888, k = 5, l = 4, \eta = 5, \beta = 235, w = 96.$

3.3 FALCON

FALCON(**FA**st Fourier **L**attice-based **CO**mpact signatures over **N**TRU) is a post-quantum scheme designed to provide secure digital signatures for which there is currently no algorithm that can be efficiently attacked by quantum computers. FALCON is based on lattice-based cryptography. For compactness, it relies on the NTRU lattices[12] which involves finding a short vector in a certain lattice defined over polynomial rings, reducing the key size to $O(n)$ and speeding up computations. Moreover, FFT is used for efficient polynomial multiplication. FALCON is known for its competitive performance in terms of speed and signature size, making it suitable for use in blockchains.

In the key generation phase, two crucial calculations are performed. The first entails solving the NTRU equations, resulting in the creation of the public key pk and private key sk used for verification and signing. The second calculation involves generating a tree structure from the derived private key information. This tree is a fundamental element for the randomness of signature values, ensuring the security of the FALCON algorithm.

Algorithm 10: FALCON_Key Generation

```
Data: -
Result: a public key $pk$, a secret key $sk$
1 compute polynomial $f, g, F, G \in \mathbb{Z}[x]/(\phi)$ that satisfies the NTRU equation
   $fG - gF = q \mod \phi$;
2 $h \leftarrow gf^{-1} \mod q$;
3 $pk \leftarrow \left[\begin{array}{c|c} -h & I_n \\ \hline qI_n & O_n \end{array}\right]$;
4 $B \leftarrow \left[\begin{array}{c|c} g & -f \\ \hline G & -F \end{array}\right]$;
5 $\hat{B} \leftarrow \mathrm{FFT}(B)$;
6 compute the FALCON Tree $T$ from $\hat{B}$;
7 $sk \leftarrow (\hat{B}, T)$;
8 return $(pk, sk)$
```

In the signing phase, a hash-and-sign signature scheme based on the GPV framework is utilized. It computes the point $c \in \mathbb{Z}_q[x]/\phi$ corresponding to the hashed value of the message with a random 320-bit salt. Subsequently, a preimage $t = (t_1, t_2)$ satisfying $t_1 + t_2 h = c \mod q$ is calculated, where t is not necessarily short. Next, a short signature vector $s = (s_1, s_2)$ is computed through a sampling process from the obtained t. To ensure the secrecy of the private key during sampling, the tree generated in the key generation phase is used. Finally, since s_1 can be easily calculated from s_2 and public information, salt and s_2 become the signature values.

Algorithm 11: FALCON_Sign

```
    Data: a message $m$, a secret key $sk$, a bound $\beta$
    Result: a signature $sig$
1   Generate a 320bit random salt $r \in \{0,1\}^{320}$;
2   compute hash point $c$ corresponding to concatenated string $(r||m)$.
      $c = \mathcal{H}(r||m, q, n)$;
3   $t \leftarrow \left(-\frac{1}{q}\mathrm{FFT}(c) \cdot \mathrm{FFT}(F), \frac{1}{q}\mathrm{FFT}(c) \cdot \mathrm{FFT}(f)\right)$;
4   while $\|s\| > \beta$ do
5       sample lattice point $z$ around $t$ using FALCON_Tree;
6       $s = (t - z)\hat{B}$;
7   compress $s_2$ to a bitstring;
8   $sig \leftarrow (r, \text{encoded } s_2)$;
9   return $sig$
```

The process of compressing s_2 and more detailed information about the FALCON Tree can be found in the official FALCON documentation [20]. Verification involves calculating s_1 from the signature, message, and public key, and then checking whether the norm of $s = (s_1, s_2)$ is below a specific bound.

Algorithm 12: FALCON_Verification

```
    Data: a signature $sig$, a message $m$, a public key $(h, q)$, a bound $\beta$
    Result: $True$ or $False$
1   $(r, \text{encoded } s_2) \leftarrow sig$;
2   compute hash point $c$ corresponding to concatenated string $(r||m)$;
3   $s_1 \leftarrow c - s_2 h \mod q$;
4   if $\|(s_1, s_2)\| \leq \beta$ then
5       return $True$
6   else
7       return $False$
```

Chapter 4

Application to Blockchain

4.1 Proposed method

We explored VRF and several quantum-resistant digital signature technologies. The VRF currently employed in blockchains such as Algorand utilizes elliptic curve cryptography for digital signatures. This reliance on elliptic curve cryptography becomes a security concern with the advancement of quantum computing, potentially leading to serious incidents of compromise of private keys.

We designed a Verifiable Coin Toss(VCT) function akin to VRF by utilizing FALCON, a quantum-resistant digital signature technology validated by National Institute of Standards and Technology(NIST). And we will integrate this function with the ASIC-resistant consensus algorithm ECCPoW to introduce an energy-efficient and fair consensus selection algorithm. The reason for choosing FALCON is the necessity for an algorithm with a small signature size to compact the block size(see Table 4.1). The summary of our Green algorithm is depicted in Figure 4.1.

	PUBLIC KEY SIZE	SIGNATURE SIZE
DILITHIUM5	2,592 bytes	4,595 bytes
FALCON-1024	1,793 bytes	1,280 bytes
SPHINCS+	48 bytes	30,696 bytes

Table 4.1: Digital signature passed through NIST round 3

Our contribution involves designing a blockchain consensus selection algorithm that

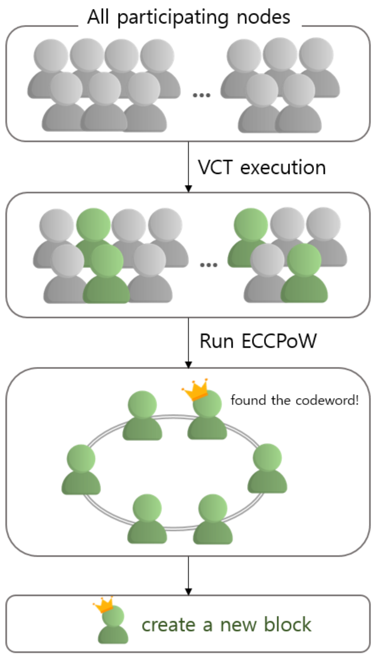


Figure 4.1: Green consensus process.

is secure against quantum computing algorithms. When this algorithm is applied, a subset of nodes is fairly and randomly chosen among all participating nodes to execute the PoW algorithm, thereby reducing energy consumption. Additionally, all nodes can verify the evidence provided by the selected nodes to ensure they were not chosen in an abnormal manner.

The output of FALCON is a vector with a small norm value. Leveraging the features of FALCON, we can determine the Head and Tail of a coin toss function as follows.

Algorithm 13: Verifiable_Coin_Toss

```
    Data: a previous block header $pbh$, a FALCON secret key $sk$, a bound $\beta$, a
          threshold $\tau$
    Result: a proof $\pi$, a result $res$
 1  Generate a 320bit random salt $r \in \{0,1\}^{320}$;
 2  compute hash point $c$ corresponding to concatenated string $(r||m)$.
      $c = \mathcal{H}(r||m, q, n)$;
 3  $t \leftarrow \left(-\frac{1}{q}\mathrm{FFT}(c) \cdot \mathrm{FFT}(F), \frac{1}{q}\mathrm{FFT}(c) \cdot \mathrm{FFT}(f)\right)$;
 4  while $\|s\| > \beta$ do
 5      sample lattice point $z$ around $t$ using FALCON_Tree;
 6      $s = (t - z)\hat{B}$;
 7  if $\|s\| \leq \tau$ then
 8      $res \leftarrow Head$
 9  else
10      $res \leftarrow Tail$
11  compress $s_2$ to a bitstring;
12  $\pi \leftarrow (r, \text{encoded } s_2)$;
13  return $(\pi, res)$
```

This function is designed to randomly select a subset of nodes from the entire set of nodes in the consensus algorithm, accompanied by outputting proof π to demonstrate fairness. All nodes can use the proof and the verification algorithm below to verify the correctness of the result of the VCT function.

Algorithm 14: VCT_verification

```
    Data: a previous block header $pbh$, a proof $\pi$, a FALCON public key $(h, q)$, a
          bound $\beta$
    Result: $True$ or $False$
 1  $(r, \text{encoded } s_2) \leftarrow \pi$;
 2  compute hash point $c$ corresponding to concatenated string $(r||pbh)$;
 3  $s_1 \leftarrow c - s_2 h \mod q$;
 4  if $\|(s_1, s_2)\| \leq \beta$ then
 5      return $True$
 6  else
 7      return $False$
```

Additionally, the proportion of nodes selected by VCT among all nodes can be adjusted as desired using the parameter τ.

Let us calculate the parameter τ that adjusts the size of the consensus set. First we need to calculate mean and variance of norm of signature. s_1 and s_2 are n-degree polynomials, and a_i and b_i are independent and identically distributed(i.i.d.) random variables.

$$s_1 = a_0 + a_1X + \cdots + a_{1023}X^{1023}$$
$$s_2 = b_0 + b_1X + \cdots + b_{1023}X^{1023}$$

Let us denote the variance of each a_i(and b_i) as σ^2, and let $Y = (a_0^2 + a_1^2 + \cdots + a_{1023}^2) + (b_0^2 + b_1^2 + \cdots + b_{1023}^2)$. Then expected value and variance of Y can be calculated as follows:

$$\begin{aligned} E[Y] &= E[a_0^2 + a_1^2 + \cdots + a_{1023}^2 + b_0^2 + b_1^2 + \cdots + b_{1023}^2] \\ &= E[a_0^2] + E[a_1^2] + \cdots + E[a_{1023}^2] + E[b_0^2] + E[b_1^2] + \cdots + E[b_{1023}^2] \\ &= Var[a_0] + Var[a_1] + \cdots + Var[a_{1023}] + Var[b_0] + Var[b_1] + \cdots + Var[b_{1023}] \\ &= 2 \cdot 1024 \cdot \sigma^2 \end{aligned}$$

$$
\begin{aligned}
Var[Y] &= Var[a_0^2 + a_1^2 + \cdots + a_{1023}^2 + b_0^2 + b_1^2 + \cdots + b_{1023}^2] \\
&= Var[a_0^2] + Var[a_1^2] + \cdots + Var[a_{1023}^2] + Var[b_0^2] + Var[b_1^2] + \cdots + Var[b_{1023}^2] \\
&= 2 \cdot 1024 \cdot Var[a_0^2] \\
&= 2 \cdot 1024 \cdot (E[a_0^4] - E[a_0^2]^2) \\
&\because E[X^4] = \int_{-\infty}^{\infty} x^4 f(x) dx \\
&\qquad = \int_{-\infty}^{\infty} \left(x^4 \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \right) dx \\
&\qquad = 3\sigma^2 \int_{-\infty}^{\infty} \left(x^2 \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \right) dx \\
&\qquad = 3\sigma^2 \cdot \sigma^2 = 3\sigma^4 \\
&= 2 \cdot 1024 \cdot (3\sigma^4 - \sigma^4) \\
&= 4096 \cdot \sigma^4
\end{aligned}
$$

By substituting 168.3886, which is specified in the FALCON document, as the standard deviation σ for each coefficient, the mean and standard deviation for Y are,

$$E[Y] \approx 58,070,468$$

$$std[Y] \approx 1,814,702$$

Now the threshold τ can be calculated by the Inverse Cumulative Distribution Function(Inverse CDF).

$$x = Q(p)$$

x is the value of the random variable, $Q(p)$ is the Inverse CDF(or Quantile function) for the probability value p. The Inverse CDF for a specific probability value p is calculated using the mean μ and standard deviation σ as follows,(erf^{-1} means Inverse Error Function)

$$Q(p) = \mu + \sigma \cdot \text{erf}^{-1}(2p - 1)$$

We can compute the value of $x = Q(p)$ by using mathematical tool(such as SciPy library in Python).

p	threshold
0.05	55,085,531
0.10	55,745,222
0.15	56,189,632
0.20	56,543,158
0.25	56,846,452
0.30	57,118,819

Table 4.2: Threshold that P($X \leq$ *threshold*) $= p$

The following section is about the experiment, we will set the VCT parameter τ to 55,745,222. In other words, only 10% of the total nodes will execute ECCPoW.

4.2 Experiment and Result

We conducted experiments measuring the electricity consumption for generating 100 blocks using only the ECCPoW consensus algorithm and a combination of VCT and ECCPoW. The parameters required for VCT, such as n, q, β, etc., were set to the recommended values for FALCON-1024 in the official documentation[20], and the probability of finding the correct answer in ECCPoW was set to 4.830240e-6. In practice, the probability of finding the correct answer in a blockchain is inversely proportional

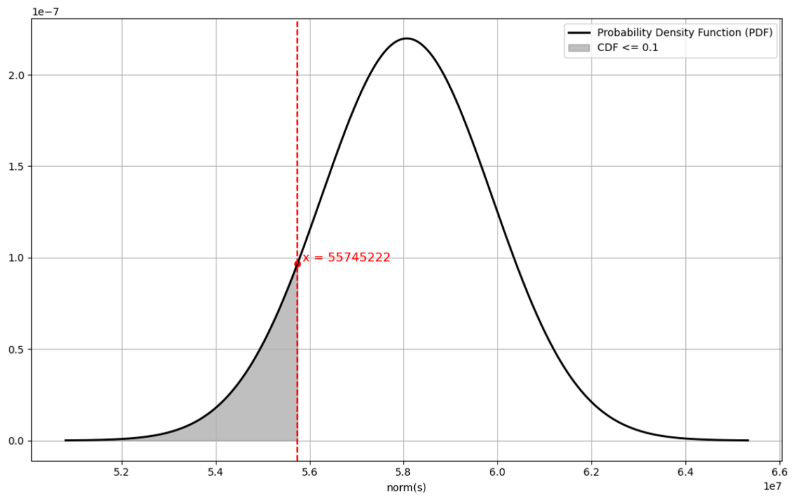
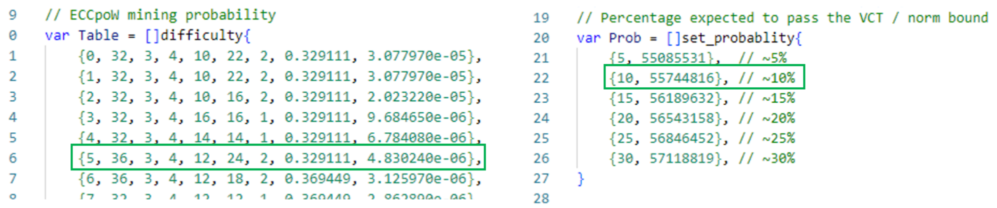


Figure 4.2: Threshold that selects 10% on average.

to the number of participating nodes to maintain a consistent block generation period. Assuming 100 nodes, if the number of participating nodes increases tenfold, the speed of finding the correct answer also increases in a similar proportion, resulting in faster block generation times. Conversely, a decrease in the number of nodes leads to slower block generation times. Therefore, we assumed 100 nodes and set the probability of getting heads in VCT to 10%, adjusting the probability of finding the correct answer in VCT+ECCPoW to 3.07797e-5 (Figure 4.3).



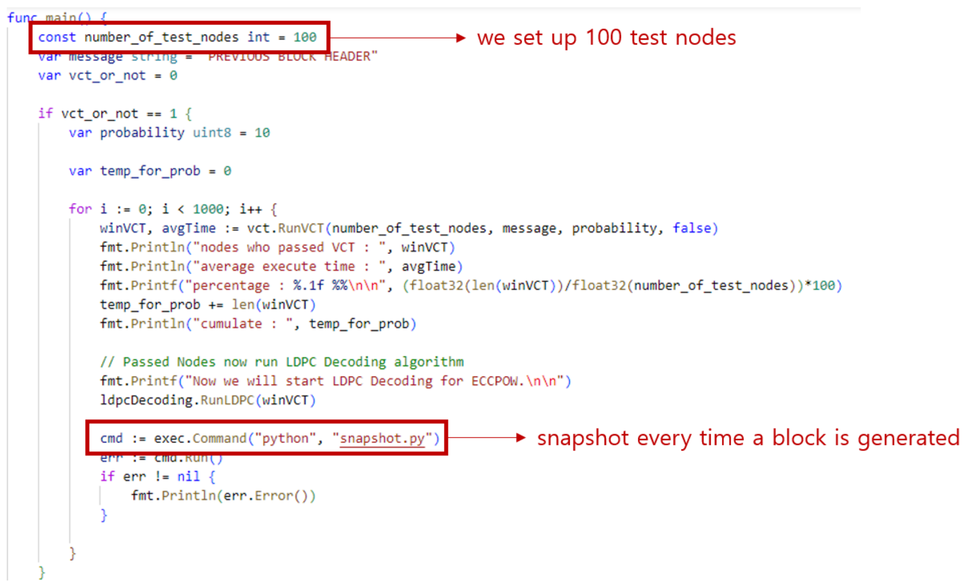
```
// ECCpoW mining probability
var Table = []difficulty{
    {0, 32, 3, 4, 10, 22, 2, 0.329111, 3.077970e-05},
    {1, 32, 3, 4, 10, 22, 2, 0.329111, 3.077970e-05},
    {2, 32, 3, 4, 10, 16, 2, 0.329111, 2.023220e-05},
    {3, 32, 3, 4, 16, 16, 1, 0.329111, 9.684650e-06},
    {4, 32, 3, 4, 14, 14, 1, 0.329111, 6.784080e-06},
    {5, 36, 3, 4, 12, 24, 2, 0.329111, 4.830240e-06},
    {6, 36, 3, 4, 12, 18, 2, 0.369449, 3.125970e-06},
    {7, 32, 3, 4, 12, 12, 1, 0.369449, 2.862800e-06}
```

```
// Percentage expected to pass the VCT / norm bound
var Prob = []set_probablity{
    {5, 55085531},  // ~5%
    {10, 55744816}, // ~10%
    {15, 56189632}, // ~15%
    {20, 56543158}, // ~20%
    {25, 56846452}, // ~25%
    {30, 57118819}, // ~30%
}
```

Figure 4.3: Defining probabilities of ECCPoW(left) and VCT(right).

We utilized an electricity consumption meter(Figure 4.4), capturing snapshots using a camera at each block's generation to measure the electricity usage during the generation of 100 blocks.

Figure 4.4: Measure the amount of elctricity used when generating block.



```go
func main() {
    const number_of_test_nodes int = 100
    var message string = "PREVIOUS BLOCK HEADER"
    var vct_or_not = 0

    if vct_or_not == 1 {
        var probability uint8 = 10

        var temp_for_prob = 0

        for i := 0; i < 1000; i++ {
            winVCT, avgTime := vct.RunVCT(number_of_test_nodes, message, probability, false)
            fmt.Println("nodes who passed VCT : ", winVCT)
            fmt.Println("average execute time : ", avgTime)
            fmt.Printf("percentage : %.1f %%\n\n", (float32(len(winVCT))/float32(number_of_test_nodes))*100)
            temp_for_prob += len(winVCT)
            fmt.Println("cumulate : ", temp_for_prob)

            // Passed Nodes now run LDPC Decoding algorithm
            fmt.Printf("Now we will start LDPC Decoding for ECCPOW.\n\n")
            ldpcDecoding.RunLDPC(winVCT)

            cmd := exec.Command("python", "snapshot.py")
            err := cmd.Run()
            if err != nil {
                fmt.Println(err.Error())
            }

        }
    }
```

Figure 4.5: Test code for experiment.

As a result, it was confirmed that when 100 blocks were generated with ECCPoW consensus, the total amount of electricity consumed was 5466kwh, and when Green consensus combining ECCPoW and VCT was used, a total of 687kwh of electricity was used(It has decreased by approximately 87.43%.).

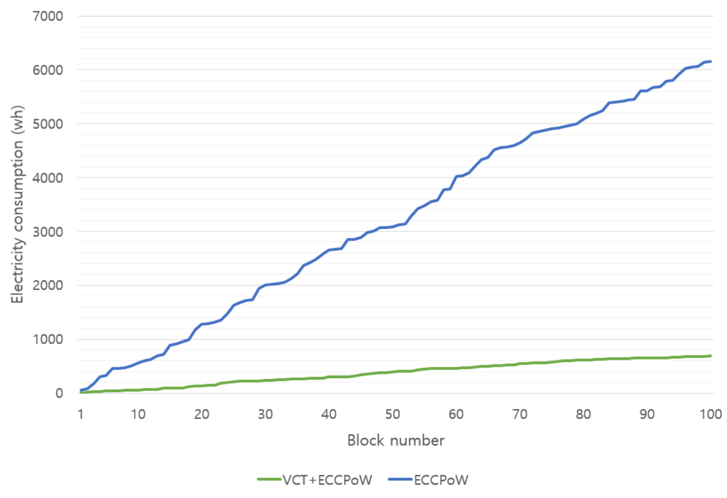


Figure 4.6: Cumulative electricity consumption.

4.3 Conclusion

In summary, this paper begins by exploring VRF and the associated technology of digital signatures. Additionally, the paper discusses the risk of private key exposure in current digital signature algorithms due to quantum computers. Consequently, the paper examines post-quantum digital signature algorithms currently under consideration by NIST, specifically designing the VCT function using FALCON, one of the candidates. Our contribution is demonstrated through experiments, where we designed a quantum-resistant coin tossing protocol using the FALCON and observed a significant reduction in electricity consumption when combined with ECCPoW. Therefore, utilizing the VCT algorithm has the potential to save significant amounts of electricity globally, currently wasted in blockchain mining.

References

1. S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, p. 21260, 2008.

2. C. D. A. P. Team, "Cambridge bitcoin electricity consumption index," 2021. November 10, 2023.

3. H. Jung and H.-N. Lee, "Eccpow: Error-correction code based proof-of-work for asic resistance," *Symmetry*, vol. 12, no. 6, p. 988, 2020.

4. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, pp. 51–68, 2017.

5. W. Diffie, "New direction in cryptography," *IEEE Trans. Inform. Theory*, vol. 22, pp. 644–654, 1976.

6. R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

7. V. S. Miller, "Use of elliptic curves in cryptography," in *Conference on the theory and application of cryptographic techniques*, pp. 417–426, Springer, 1985.

8. P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

9. D. Micciancio and O. Regev, "Lattice-based cryptography," *Post-quantum cryptography*, pp. 147–191, 2009.

10. M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 99–108, 1996.

11. O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.

12. J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *International algorithmic number theory symposium*, pp. 267–288, Springer, 1998.

13. J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte, "Ntrusign: Digital signatures using the ntru lattice," in *Topics in Cryptology—CT-RSA 2003: The Cryptographers' Track at the RSA Conference 2003 San Francisco, CA, USA, April 13–17, 2003 Proceedings*, pp. 122–140, Springer, 2003.

14. C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of cryptology*, vol. 4, pp. 161–174, 1991.

15. V. Lyubashevsky, "Fiat-shamir with aborts: Applications to lattice and factoring-based signatures," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 598–616, Springer, 2009.

16. N. Aragon, O. Blazy, P. Gaborit, A. Hauteville, and G. Zémor, "Durandal: a rank metric based signature scheme," in *Advances in Cryptology–EUROCRYPT*

2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38, pp. 728–758, Springer, 2019.

17. P. Gaborit, A. Hauteville, D. H. Phan, and J.-P. Tillich, "Identity-based encryption from codes with rank metric," in *Annual International Cryptology Conference*, pp. 194–224, Springer, 2017.

18. P. Gaborit, G. Murat, O. Ruatta, and G. Zémor, "Low rank parity check codes and their application to cryptography," in *Proceedings of the Workshop on Coding and Cryptography WCC*, vol. 2013, 2013.

19. L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-dilithium: A lattice-based digital signature scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 238–268, 2018.

20. T. Prest, P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon," *Post-Quantum Cryptography Project of NIST*, 2020.