Post-Quantum Coin（PQC）White Paper

Peter Q. Maxwell

April 16 2024

Content

1. Motivation and the Birth of Post-Quantum Coin (PQC)

Nowadays, most of blockchain and cryptographic currency systems face three major challenges or serious concerns:

(1) The EC-DSA signature or its variants like EC-Schnorr will not secure any longer in the post-quantum era.

(2) BTC is approaching its fourth halving of mint. The miners of cryptocurrencies face significant shrinking of minting.

(3) Few cryptocurrencies, if not at all, are truly decentralized.

With these three challenges in mind, we commence the design and construction of PQChain in 2016. With great efforts in the past eight years, we are happy to announce now the birth of PQChain and its associated token PQCoin. PQC is a blockchain network that supports resistance against quantum attacks, and utilizes multi-computational power, and is fully decentralized.

(1) It employs post-quantum lattice-based signature algorithms, a variant of ML-DAS but over large-Galois-group prime-degree prime-ideal number field, which eliminate the vulnerability of cyclotomic rings (commonly employed by most lattice-based cryptographic standards) in the long run. This design is reminisced of BTC choosing EC-DSA but refusing the underlying elliptic curves recommended by government agencies.

(2) We believe in the truth and power of PoW. PQC utilizes a multi-parent chain auxiliary proof-of-work (PoW) consensus algorithm. This mechanism supports the computational power of blockchain networks using double-SHA256 (such as BTC and its chidrens like BCH、BSV and XEC) and Scrypt (such as LTC, DOGE). That is, the PQC network does not need to run its own mining pools, instead any miner of BTC/BCH/BSV/XEC or LTC/DOGE can get the rewards of PQC for free! This is very friendly to most of PoW miners, with the income cutting down by the mint shrinking. By obtaining more computational power, the security of the PQC network will also be greatly enhanced. To address the disparity in computing power, a completely new difficulty

adjustment method is employed, which effectively improves the ability to adjust difficulty in the face of significant fluctuations in computing power while also ensuring fair reward distribution.

Finally, PQC is fully decentralized. It has no owners, every PQCoin has to be obtained via PoW mining.

2. Existing Technology Analysis

Bitcoin was introduced by Satoshi Nakamoto in 2008 through the paper titled "Bitcoin: A Peer-to-Peer Electronic Cash System." However, there are three areas that can be improved for blockchain systems based on the Proof-of-Work (PoW) consensus algorithm:

(1) The security of blockchain needs to be reconsidered. With the advancement of quantum computing technology, traditional digital signature algorithms face significant security threats that could compromise transactions on the blockchain. The current mainstream blockchains are vulnerable to potential quantum attacks.

Traditional digital signature algorithms, such as RSA and ECDSA, rely on the difficulty of computational problems (integer factorization and elliptic curve discrete logarithm problem, respectively) to ensure data integrity and authentication. However, quantum computers have the potential to solve these problems in an efficient way, undermining the security of traditional digital signatures and posing a serious threat to the security of blockchain systems. Therefore, it is crucial to promptly address this issue by using post-quantum signature algorithms in blockchain systems.

(2) Due to the high computational requirements during the consensus process, a drawback is the significant energy consumption. However, when the computational power reaches a large scale, it becomes difficult to execute a 51% attack, thereby providing higher security. Additionally, with the rapid development of blockchain technology, numerous emerging blockchains utilize proof-of-work (PoW) consensus algorithms, but they may lack sufficient computational power, making them vulnerable to 51% attacks. To address this issue and conserve resource consumption, an Auxiliary Proof of Work (AuxPoW) algorithm has been proposed.

The AuxPoW algorithm modifies the block structure and verification method based on the PoW algorithm. It incorporates the proof-of-work from other chains, achieving a simultaneous

consensus across multiple blockchains. The blockchain requiring auxiliary proof of work consensus is referred to as the "child chain," while the blockchain providing assistance for consensus is called the "parent chain." The AuxPoW consensus algorithm significantly enhances the security of blockchains that lack sufficient computational power while increasing the utilization of computational resources and reducing energy consumption and computational costs. A limitation is that current systems employing this mechanism mainly use a single parent chain or utilize a parent chain with the same hash algorithm to assist a child chain in proof-of-work consensus. For example, Litecoin acts as a parent chain to assist Dogecoin in AuxPoW consensus, and Bitcoin has previously served as a parent chain to assist Namecoin in AuxPoW consensus.

(3) The traditional PoW consensus mechanism adjusts the difficulty for the next interval based solely on a fixed number of blocks and the block generation time. It uses a difficulty adjustment algorithm to stabilize the average block generation time in the network by reflecting changes in computational power over time. However, during operation, if there is a period with a significantly shorter average block generation time before the difficulty adjustment, it can lead to a sharp increase in consensus difficulty. This results in a quick decrease in block generation speed, requiring a sensibly higher computational power and corresponding energy consumption for the generation of tokens in the next stage.

3. Post-Quantum Signature Algorithm

For quantum computers, Schnorr signatures based on discrete logarithm problems can be easily forged. Continuing to use Schnorr signatures in the post-quantum era is completely insecure for accounts and transactions. Therefore, in response to this challenge, the cryptography community has started researching post-quantum cryptography, aiming to construct cryptographic algorithms that can resist quantum attacks. In 2016, the National Institute of Standards and Technology (NIST) in the United States hosted a post-quantum cryptography solicitation project, and the selected cryptographic schemes include those based on lattice techniques.

The lattice-based post-quantum cryptographic schemes rely on lattice problems such as SIS (Short Integer Solution) and LWE (Learning With Errors). These schemes have been suggested to possess quantum resistance. They also demonstrate high computational efficiency and good scalability, making them strong contenders in the NIST post-quantum cryptography

standardization project. Among the 7 post-quantum cryptographic schemes selected in the third round of NIST's solicitation, 5 are lattice-based schemes. In the first batch of 4 post-quantum cryptographic algorithms proposed for standardization, 3 are lattice-based schemes. The computational speed of lattice-based schemes is very fast, comparable to or even faster than that of the EC-DSA or EC-Schnorr signature algorithms.

Many known lattice-based post-quantum cryptographic schemes use cyclotomic rings, especially power-of-two cyclotomic rings, as their underlying algebraic structure. For example, digital signature schemes like CRYSTALS-Dilithium (aka, ML-DSA) and Falcon, two of the three post-quantum signature schemes to be standardized by NIST, both utilize the power-of-two cyclotomic ring $\mathbb{Z}_q[x]/(x^n+1)$, where $n = 2^k$. These cyclotomic rings allow for efficient multiplication operations using NTT (Number Theoretic Transform), enabling the construction of efficient cryptographic schemes. However, targeted attacks against cyclotomic rings have been proposed, exploiting subfields, small Galois groups and ring homomorphisms in these rings, and making cyclotomic ring-based cryptographic schemes subject to additional security threats compared to other lattices.

In response to this situation, the cryptographic scheme used in PQC uses $\mathbb{Z}_q[x]/(x^n-x-1)$, named the large-Galois-group prime-degree prime-ideal field, as new underlying algebraic structure, which has characteristics of "high security, prime order, large Galois group and inert modulus". Such characteristics making $\mathbb{Z}_q[x]/(x^n-x-1)$ them better resistant to targeted attacks such as subfield attacks and automorphism attacks. For the security concerns raised by power-of-two cyclotomic rings and the security advantages of large-Galois-group prime-degree prime-ideal field, the reader is referred to https://ntruprime.cr.yp.to/

The following is an introduction to the proposed post-quantum digital signature scheme based on large-Galois-group prime-degree prime-ideal field and CRYSTALS-Dilithium(aka. ML-DSA), referred to as Dilithium-Prime.

Dilithium-Prime consists of three parts: the key generation algorithm Dilithium-Prime.KeyGen, the signature algorithm Dilithium-Prime.Sign, and the verification algorithm Dilithium-Prime.Verify, as shown in Algorithm 1 to Algorithm 3. $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n-x-1)$ denotes the large-Galois-group prime-degree prime-ideal field, where n and q are both prime

numbers, and $x^n - x - 1$ is an irreducible polynomial over $\mathbb{Z}_q$.

Algorithm 2 presents both deterministic and random versions of the signature algorithm. The main difference lies in how the polynomial vector y is generated. In the deterministic algorithm, the random seed ρ' used to generate y is computed by a hash function. Conversely, in the random version, ρ' is generated by sampling 512 bits randomly. It's important to note that this does not imply that the deterministic algorithm cannot ensure the randomness of the seed ρ'. In the deterministic algorithm, the randomness of ρ' is derived from the hash value computed using the input private key sk and the signature message M. In this case, since the algorithm's randomness comes from the message, the rejection rate for signatures remains consistent for the same message across multiple signing attempts. We remark that, when verifying transactions, the Dilithium-Prime.Verify algorithm is unaware of whether the signature is generated randomly or deterministically..

Algorithm 1. Key generation algorithm Dilithium-Prime.KeyGen.

Input: Security parameters 1^λ

Output: Signature key pair (pk, sk)

$\zeta \leftarrow \{0,1\}^{256}$

$(\rho, \rho', K) \in \{0,1\}^{256} \times \{0,1\}^{512} \times \{0,1\}^{256} := \mathrm{H}(\zeta)$

$\mathbf{A} \in \mathcal{R}_q^{k \times l} := \mathrm{ExpandA}(\rho)$

$(\boldsymbol{s_1}, \boldsymbol{s_2}) \in \mathcal{S}_\eta^l \times \mathcal{S}_\eta^k := \mathrm{ExpandS}(\rho')$

$\boldsymbol{t} := \mathbf{A}\boldsymbol{s_1} + \boldsymbol{s_2}$

$(\boldsymbol{t_1}, \boldsymbol{t_0}) := \mathrm{Power2Round}_q(\boldsymbol{t}, d)$

$tr \in \{0,1\}^{256} := \mathrm{H}(\rho || \boldsymbol{t_1})$

RETURN $(pk := (\rho, \boldsymbol{t_1}), sk := (\rho, K, tr, \boldsymbol{s_1}, \boldsymbol{s_2}, \boldsymbol{t_0}))$

Algorithm 2. Signature algorithm Dilithium-Prime.Sign.

Input: Private key sk, message M

Output: Signature σ

$\mathbf{A} \in \mathcal{R}_q^{k \times l} := \mathrm{ExpandA}(\rho)$

$\mu \in \{0,1\}^{512} := \mathrm{H}(tr || M)$

$\kappa := 0,\ (\boldsymbol{z}, \boldsymbol{h}) := \perp$

$\rho' \in \{0,1\}^{512} := \mathrm{H}(K || \mu)$ // In the random version of the algorithm, $\rho' \leftarrow \{0,1\}^{512}$

05 **WHILE** $(\boldsymbol{z}, \boldsymbol{h}) := \perp$ **DO**

06 $\quad \boldsymbol{y} \in \tilde{S}_{\gamma_1}^{l} := \text{ExpandMask}(\rho', \kappa)$

07 $\quad \boldsymbol{w} := \mathbf{A}\boldsymbol{y}$

08 $\quad \boldsymbol{w_1} := \text{HighBits}_q(\boldsymbol{w}, 2\gamma_2)$

09 $\quad \tilde{c} \in \{0,1\}^{256} := \text{H}(\mu || \boldsymbol{w_1})$

10 $\quad c \in \mathcal{B}_\tau := \text{SampleInBall}(\tilde{c})$

11 $\quad \boldsymbol{z} := \boldsymbol{y} + c\boldsymbol{s_1}$

12 $\quad \boldsymbol{r_0} := \text{LowBits}_q(\boldsymbol{w} - c\boldsymbol{s_2}, 2\gamma_2)$

13 $\quad$ **IF** $\|\boldsymbol{z}\|_\infty \geq \gamma_1 - \beta$ **or** $\|\boldsymbol{r_0}\|_\infty \geq \gamma_2 - \beta$

14 $\quad\quad (\boldsymbol{z}, \boldsymbol{h}) := \perp$

15 $\quad$ **ELSE**

16 $\quad\quad \boldsymbol{h} := \text{MakeHint}_q(-c\boldsymbol{t_0}, \boldsymbol{w} - c\boldsymbol{s_2} + c\boldsymbol{t_0}, 2\gamma_2)$

17 $\quad\quad$ **IF** $\|c\boldsymbol{t_0}\|_\infty \geq \gamma_2$ **or** *the # of* $1's$ *in* $\boldsymbol{h}$ *is greater than* ω

18 $\quad\quad\quad (\boldsymbol{z}, \boldsymbol{h}) := \perp$

19 $\quad \kappa := \kappa + l$

20 **RETURN** $\sigma = (\tilde{c}, \boldsymbol{z}, \boldsymbol{h})$

Algorithm 3. Verification algorithm Dilithium-Prime.Verify.

Input: Public key pk, message M, signature $\sigma = (\tilde{c}, \boldsymbol{z}, \boldsymbol{h})$

Output: Verification passed (output 1) or verification failed (output 0)

01 $\mathbf{A} \in \mathcal{R}_q^{k \times l} := \text{ExpandA}(\rho)$

02 $\mu \in \{0,1\}^{512} := \text{H}(\text{H}(\rho || \boldsymbol{t_1}) || M)$

03 $c := \text{SampleInBall}(\tilde{c})$

04 $\boldsymbol{w_1'} := \text{UseHint}_q(\boldsymbol{h}, \mathbf{A}\boldsymbol{z} - c\boldsymbol{t_1} \cdot 2^d, 2\gamma_2)$

05 **IF** $\|\boldsymbol{z}\|_\infty < \gamma_1 - \beta$ **and** $\tilde{c} = \text{H}(\mu || \boldsymbol{w_1'})$ **and** *# of* $1's$ *in* $\boldsymbol{h}$ *is* $\leq \omega$

06 $\quad$ **RETURN** 1

07 **ELSE**

08 $\quad$ **RETURN** 0

To compress the size of the public key, Dilithium-Prime uses a seed ρ instead of a matrix A as the public key. This requires Dilithium-Prime.Sign and Dilithium-Prime.Verify algorithms to repeatedly generate A from ρ. Considering that matrix A is only used for polynomial multiplication in the algorithm, if storage is not a concern, the algorithm can precompute the corresponding form $\widehat{A}$ of matrix A in the NTT domain. Using $(\widehat{A}, t_1)$ as the public key saves time by avoiding repeated calls to ExpandA and NTT algorithms, thereby improving the efficiency

of the signature and verification algorithms. Since the efficiency of NTT algorithms on large-Galois-group prime-degree prime-ideal field is relatively low and it is not possible to directly generate $\widehat{\mathrm{A}}$ in the NTT domain, this strategy provides a more significant efficiency improvement for Dilithium-Prime compared to the CRYSTALS-Dilithium. Similarly, precomputing and storing the NTT forms of s_1, s_2, and t_0 in the private key can save time in subsequent algorithms.

In another scenario, the private key of Dilithium-Prime can be compressed to a 32-byte random seed $\zeta \leftarrow \{0,1\}^{256}$. In this case, the signature algorithm needs to regenerate ρ, K, tr, s_1, s_2, t_0 using ζ, which significantly reduces the efficiency of the algorithm while saving storage space.

Algorithms 1- through 3 employ four hash-based sampling algorithms. ExpandA maps a 256-bit random seed ρ to a matrix $\mathrm{A} \in \mathcal{R}_q^{k\times l}$ using a hash function. Note that the resulting matrix A is in the normal domain but not NTT domain. This is because the NTT computation in large-Galois-group prime-degree prime-ideal field is not bijective, and thus, $\widehat{\mathrm{A}}$ cannot be directly generated in the NTT domain. ExpandS generates $(\mathrm{s}_1, \mathrm{s}_2) \in \mathcal{S}_\eta^l \times \mathcal{S}_\eta^k$ using a 512-bit seed ρ'. ExpandMask generates $\mathrm{y} \in \tilde{\mathcal{S}}_{\gamma_1}^l$ either randomly or based on a seed ρ' computed from sk and M. Since the signature algorithm may repeat reject sampling multiple times, ExpandMask uses a counter κ to ensure that each sampled random polynomial vector is different from each other. B_τ is a subset of R, where the polynomials in $\mathcal{B}_\tau$ have exactly τ coefficients as 1 or -1, with the rest being 0, $|\mathcal{B}_\tau| = 2^\tau \cdot \binom{n}{\tau}$. SampleInBall generates an element c in $\mathcal{B}_\tau$ through a two-step hashing process, as detailed in Algorithm 4.

Algorithm 4. $\mathcal{B}_\tau$ sampling algorirhm SampleInBall.

Input: seed $\tilde{c} \in \{0,1\}^{256}$

Output: $c \in \mathcal{B}_\tau$

01 *Initialize* $c \in \{0,1\}^n := \{0\}^n$

02 **FOR** $i = n - \tau$ *to* $n - 1$

03 $j \leftarrow \{0,1,\cdots,i\}$

04 $s \leftarrow \{0,1\}$

05 $c_i := c_j$

06 $c_j := (-1)^s$

07 **RETURN** c

Dilithium-Prime employs a series of algorithms to extract the high and low bits of elements in $\mathbb{Z}_q$, further compressing the size of the public key. Algorithms 5 through 10 provide detailed steps for this process. Apart from separating the high and low bits, the $\mathrm{MakeHint}_q$ algorithm (Algorithm 9) computes a bit-sized "hint" h for any given $r \in \mathbb{Z}_q$ and a norm-smaller $z \in \mathbb{Z}_q$. This hint allows the verifier to calculate the high bits of $r + z$ using only r and h. The process of calculating the high bits of $r + z$ is described by the $\mathrm{UseHint}_q$ algorithm (Algorithm 10).

Algorithm 5. High-low bit decomposition algorithm $\mathrm{Power2Round}_q$.

Input: $r \in \mathbb{Z}_q, 0 < d \leq \lfloor \log_2 r \rfloor, q$

Output: The high bits and low bits of r, denoted as $(r_1,\ r_0)$.

$r := r \bmod^+ q$

$r_0 := r \bmod^\pm 2^d$

$r_1 := (r - r_0)/2^d$

RETURN (r_1, r_0)

Algorithm 6. High-low bit decomposition algorithm $\mathrm{Decompose}_q$.

Input: $r \in \mathbb{Z}_q, 0 < \alpha < \frac{q}{2}, q$

Output: The high bits and low bits of r, denoted as $(r_1,\ r_0)$.

$r := r \bmod^+ q$

$r_0 := r \bmod^\pm \alpha$

IF $r - r_0 = q - 1$

    $r_1 := 0$

    $r_0 := r_0 - 1$

ELSE

    $r_1 := (r - r_0)/\alpha$

RETURN (r_1, r_0)

Algorithm 5 and Algorithm 6 describe two different methods for decomposing high and low bits. The $\mathrm{Power2Round}_q$ algorithm directly decomposes r at its low d bits based on its binary orm, yielding high bits r_1 and low bits r_0 which can be respectively viewed as quotient and remainder. Note that in this case, the high bits r_1 satisfy $0 \leq \mathrm{r}_1 \leq \lfloor \mathrm{q}/2^{\mathrm{d}} \rfloor$, then the distance between $\mathrm{r}_1 \cdot 2^{\mathrm{d}}$ and $\mathrm{r}_1' \cdot 2^{\mathrm{d}}$ is usually $\geq 2^{\mathrm{d}}$, However, considering the boundary conditions $\mathrm{r}_1 = \lfloor \mathrm{q}/2^{\mathrm{d}} \rfloor$ and $\mathrm{r}_1' = 0$, the difference between $\lfloor \mathrm{q}/2^{\mathrm{d}} \rfloor \cdot 2^{\mathrm{d}}$ and 0 in mod q terms

may be small. In this case, adding a number to r can lead to a change in the high bits by more than 1.

To limit the size of the "hint" h to within 1 bit, the $\mathrm{Decompose_q}$ algorithm selects a factor α of $\mathrm{q}-1$, and calculates the high and low bits of r as $\mathrm{r}=\mathrm{r}_1\cdot\alpha+\mathrm{r}_0$, $\mathrm{r}_1\in\{0,1,\cdots,(\mathrm{q}-1)/\alpha\}$. Since $\mathrm{q}-1$ and 0 differ by only 1 in mod q terms, setting r_1 to 0 and subtracting 1 from the corresponding remainder when $\mathrm{r}_1=(\mathrm{q}-1)/\alpha$, $\mathrm{r}=\mathrm{q}+\mathrm{r}_0-1$, then the change in the high bits can be avoided from being too large.

Algorithm 7. High bits extraction algorithm $\mathrm{HighBits}_q$.

Input: $r\in\mathbb{Z}_q, 0<\alpha<\frac{q}{2}, q$

Output: The high bits of r, denoted as r_1

```
(r_1, r_0) := Decompose_q(r, α)
RETURN r_1
```

Algorithm 8. Low bits extraction algorithm $\mathrm{LowBits}_q$.

Input: $r\in\mathbb{Z}_q, 0<\alpha<\frac{q}{2}, q$

Output: The low bits of r, denoted as r_0

```
(r_1, r_0) := Decompose_q(r, α)
RETURN r_0
```

Algorithm 9. Hint generation algorithm $\mathrm{MakeHint}_q$.

Input: $r\in\mathbb{Z}_q, 0<\alpha<\frac{q}{2}, |z|\le\frac{\alpha}{2}, q$

Output: $h\in\{0,1\}$

```
r_1 := HighBits_q(r, α)
v_1 := HighBits_q(r + z, α)
IF r_1 ≠ v_1
    RETURN 1
ELSE
    RETURN 0
```

Algorithm 10. Hint restoration algorithm $\mathrm{UseHint}_q$.

Input: $r\in\mathbb{Z}_q, 0<\alpha<\frac{q}{2}, h\in\{0,1\}, q$

Output: The high bits of $r + z$, denoted as r_1.

01 $m := (q-1)/\alpha$

02 $(r_1, r_0) := \text{Decompose}_q(r, \alpha)$

03 **IF** $h = 1$ **and** $r_0 > 0$

04 **RETURN** $(r_1 + 1) \bmod^+ m$

05 **ELSE IF** $h = 1$ **and** $r_0 \leq 0$

06 **RETURN** $(r_1 - 1) \bmod^+ m$

07 **ELSE**

08 **RETURN** r_1

Table 1: the parameter sets for Dilithium-Prime

Security Level	II	III	V
q [modulus]	7681537	7681537	7681537
d [dropped bits from $\boldsymbol{t}$]	13	13	13
τ [the number of nonzero coefficient in c]	39	49	60
Challenge entropy [$\log\binom{251}{\tau} + \tau$]	192	225	257
γ_1 [coefficient range of $\boldsymbol{y}$]	2^{18}	2^{19}	2^{19}
γ_2[low-order rounding range]	$(q-1)/32$	$(q-1)/16$	$(q-1)/16$
(k, l)[dimensions of matrix **A**]	(4,4)	(6,5)	(8,7)
η[secret key range]	2	2	2
β [$2\tau \cdot \eta$]	156	196	240
ω [the maximal number of nonzero coefficient in h]	80	55	75
Repetitions [rejection sample]	3.49	2.96	6.10
pk size	1288	1916	2544
sk size	2504	3605	4801
σ size	2504	3233	4511
LWE hardness	(121,110)	(162,146)	(247,224)
SIS hardness	(110,100)	(169,153)	(243,220)

In Dilithium-Prime, the input to the algorithms is a polynomial vector in $\mathcal{R}_q$ rather than an element in $\mathbb{Z}_q$. In this context, the operation is equivalent to inputting each coefficient of each polynomial in the polynomial vector into Algorithm 5 through Algorithm 10 for calculation.

We present the parameter sets for Dilithium-Prime in Table 1. We recommend to use Lever III parameter set, which provide reasonable redundance over 128-bit quantum security.

4. Multi-Parent Chain Auxiliary Proof-of-Work Consensus Algorithm

This consensus algorithm initially adopts the 'longest chain protocol,' where each node always selects and attempts to extend the blockchain that represents the highest amount of proof-of-work. The system establishes a fixed block generation time and limits the block size to minimize occurrences of forks. Attackers are disincentivized from wasting computational power on creating additional forks; instead, their goal is to create a single, non-forked longer branch that incorporates illicit transactions to become the main chain.

Considering the characteristics of proof-of-work, it becomes apparent that launching a double-spending attack requires the attacker to acquire a substantial amount of ASIC (Application-Specific Integrated Circuit) mining power, surpassing the combined computational power of other honest miners. This entails purchasing, redirecting, or controlling over 51% of the existing mining power and mining a longer forked chain. The cost associated with such an attack is significant. Even if they manage to control approximately 30% or more of the mining power through strategies like selfish mining, there remains a considerable risk of the entire forked attack chain not receiving any rewards.

To maintain decentralization, no central authority or entity detects attacks and retroactively investigates the pre-attack stage. However, during the early stages of a new chain, when the total computational power is relatively low, some intervention may occur. Once the computational power reaches a certain threshold, the system will transition to complete decentralization.

In addition, the consensus algorithm used to achieve consensus among nodes adopts the Multi-Parent Chain Auxiliary Proof-of-Work (MP-AuxPoW) consensus. It consists of four main stages: AskForWork, Work, SubmitWork, VerifyWork, and RecordWork. The specific process is as follows:

(1) AskForWork Stage:

During this stage, nodes that aim to participate in the MP-AuxPoW consensus select a specific parent chain to contribute their computational power and request to participate in the MP-AuxPoW consensus. The process is as follows:

a. Nodes that have selected a parent chain submit their node addresses and the ID of the parent chain to the mining pool.

b. The mining pool calls the getBlockTemplate API to request an MP-AuxPoW consensus template from the subchain. The template is a pre-packaged block with filled fields but not yet finalized through consensus.

c. The subchain sends the block header hash and the corresponding parent chain difficulty to the mining pool. The difficulty value is calculated based on the parent chain ID and the difficulty adjustment algorithm.

d. The mining pool writes the received hash value and the corresponding parent chain difficulty into the coinbase transaction of the parent chain. It then packages the transactions in the transaction pool, creating a working template for the parent chain in the proof-of-work process. Finally, the mining pool returns the block header of the parent chain template.

(2) Work Stage:

During this stage, after filling the parent chain block template with the received subchain information, the consensus process based on the difficulty adjustment. The specific process is as follows:

a. Nodes perform the proof-of-work consensus based on the parent chain template by generating random numbers and conducting hash calculations. Determine whether the hash calculation results meet the difficulty requirements of the parent chain itself and the difficulty of receiving child chains.

b. If the hash calculation meets the difficulty requirements of the subchain, the work is submitted to the subchain.

c. If the hash calculation meets the difficulty requirements of the parent chain, a block is created on the parent chain.

(3) SubmitWork Stage:

During this stage, the parent chain that successfully mined a block for the auxiliary subchain submits the block information to the subchain. The specific process is as follows:

a. The mining pool submits the parent chain block header, the Merkle branch corresponding to the parent chain block's coinbase transaction, and the specific fields of the coinbase transaction to the auxiliary subchain.

b. The subchain fills these three pieces of information into the CAuxPoW field of the subchain block header. The hashAuxPoW field of the CPureBlockHeader is updated to point to the newly written CAuxPoW field.

(4) VerifyWork Stage:

During this stage, the subchain receives the block information successfully mined by the parent chain and proceeds with storing and verifying the information. The specific process is as follows:

a. Verify whether the subchain block header hash in the coinBaseTx of the CAuxPoW field matches the block header hash and difficulty values submitted by the subchain to the mining pool. This step is to verify if the information submitted by the subchain is consistent with the coinbase transaction in the parent chain.

b. Verify that the coinBaseTx and corresponding merkleBranch of the CAuxPoW field calculate whether the root hash of the block transaction Merkle tree of the block parent chain is the same as the root hash value in the hashBlockHeader. This step is to verify if the coinbase transaction storing the correct information is present in the mined parent chain blocks.

c. Determine the type of parent chain from which the work originated based on the nVersion field in the hashBlockHeader of the CAuxPoW field. Select the corresponding difficulty value submitted and perform the corresponding parent chain's hash verification on the hashBlockHeader to determine if it meets the difficulty requirements.

(5) Block Header Data Structure:

Based on the Bitcoin (BTC) block as a reference, the block header data structure in the quantum-resistant blockchain system utilizing multi-parent chain auxiliary proof-of-work consensus is designed. This design builds upon the BTC block header and adds the CAuxPoW field for auxiliary proof-of-work consensus in multiple parent chains. Additionally, a hashAuxPoW field with a size of 32 bytes is included in the block header to facilitate the multi-parent chain auxiliary proof-of-work consensus. The specific design is illustrated in Figure 1.

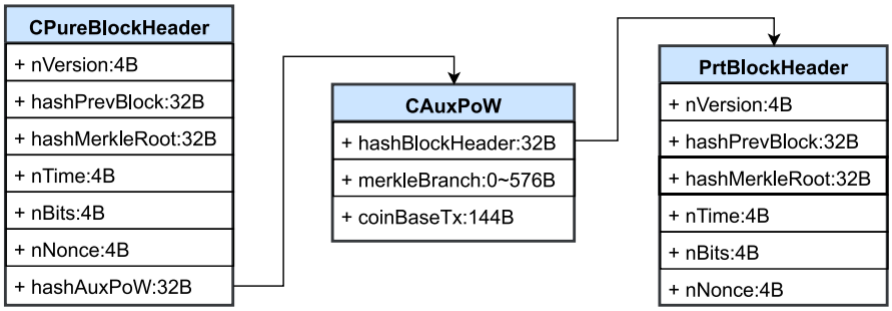


Figure 1: Block Header Based on Multi-Parent Chain Auxiliary Proof-of-Work

5. Difficulty Adjustment Algorithm for Multi-Parent Chains

(1) Principles of Difficulty Adjustment Algorithm Design:

a. The subchain blockchain system aims to maintain a consistent block production rate of 2016 blocks within a 168-hour period, resulting in an average block time of approximately 5 minutes. Difficulty adjustment takes place once a week to ensure timely adaptation to the current state of the subchain system.

b. The block production ratio between chains with high computational power and chains with lower computational power is maintained at approximately 10:1. This approach serves to protect low-computational-power parent chains, allowing miners on these chains to maintain a reasonable probability of block production and incentivizing their participation in consensus. Furthermore, it prevents the subchain system from becoming overly reliant on parent chains with high computational power as the sole source of mining power. Each type of parent chain has its own block difficulty. If the ratio is exceeded, the difficulty of the chain with high computational power is increased; if it falls below the ratio, adjustments are made accordingly.

c. The blockchain system is designed to withstand malicious attacks such as sudden increases or decreases in computational power, 51% attacks, selfish mining attacks, and block hopping attacks.

(2) Difficulty Adjustment Formula

The symbols and their meanings in the difficulty adjustment formula are shown in Table 2:

Table 2: Difficulty adjustment formula symbol meaning table

symbol	meaning
A	Using a parent chain with the $Double-SHA256$ hash algorithm, which refers to a parent chain with higher computational power
B	Using a parent chain with the $Scrypt$ hash algorithm, which refers to a parent chain with lower computational power
n	Difficulty adjustment round number, represented by $n \subseteq \{1,2,3,\dots\}$, The closer the round is to block production, the larger the value of n. $n = n$ represents the most recent round
m	Height of the subchain block
L_n	$L \in \{A, B\}$, where L represents the class of the chain for the difficulty adjustment round
$Block$	Represents a pointer to the previous block and can obtain block information
T	Get the pointer to the block with height m
H_{L_n}	L-class chain's difficulty value for the nth difficulty adjustment cycle
$actual\ time\ /AT$	Actual block time of 2016 blocks in the subchain
$expected\ time\ /ET$	Expected block time of 2016 blocks in the subchain
ϕ_n	Function to obtain the number of blocks mined by the A and B-class chains in the nth cycle, output is the number of A and B-class blocks
x_n	Number of blocks mined by the A-class chain in the nth cycle, $n \subseteq \{1,2,3,\dots\}$
y_n	Number of blocks mined by the B-class chain in the nth cycle, $n \subseteq \{1,2,3,\dots\}$
x_A	Weighted average of chain-like block production changes in the last 4 cycles$A$
y_B	Weighted average of chain-like block production changes in the last 4 cycles$B$
T_{L_n}	Exponential ratio calculated based on the weighted average of block variations between the $n-1th$ and nth cycles of the L-class chain and the variations in previous cycles, with a base of 1.2
c_{less6}	Number of times the B-class chain's blocks were less than 6 in the last 4 cycles
$c_{more500}$	Number of times the B-class chain's blocks were greater than 500 in the last 4 cycles
μ	Upper limit of some multiplication parts $(e.g., \frac{\alpha \cdot y_n}{x_n}, \frac{2016 \cdot \alpha}{x_n\,(\alpha+1)}, \frac{2016}{y_n(\alpha+1)})$, its value is 3
ν	Upper limit of some multiplication parts $\left(e.g., \frac{AT}{ET}, \frac{x_n}{\alpha \cdot y_n}, \frac{\alpha \cdot y_n}{x_n}\right)$, its value is 4
σ	Lower limit of some multiplication parts $(e.g., \frac{AT}{ET}, \frac{x_n}{\alpha \cdot y_n}, \frac{\alpha \cdot y_n}{x_n})$, its value is $\frac{1}{4}$
ξ	Lower limit of some multiplication parts $(e.g., \frac{\alpha \cdot y_n}{x_n}, \frac{2016}{y_n\,(\alpha+1)})$, its value is $\frac{1}{3}$
δ	Base of the logarithm.
$log_{10}()$	Logarithm with δ as the base, used to limit the difficulty adjustment value to prevent specific attacks
α	Expected proportion of block production between the two chain classes, its value is 10
β	Ratio parameter for the parent chain, its value is 0.5

γ	Ratio parameter for the difference between expected and actual block production on the same class parent chain, its value is $\gamma = 1 - \beta = 0.5$

The difficulty adjustment formula is classified according to the ratio of $\frac{\mathrm{AT}}{\mathrm{ET}}$, as shown below, which $\mathrm{H_{A_{n+1}}}$represents the difficulty of calculating the $n+1th$ cycle of Class A chain, and $\mathrm{H_{B_{n+1}}}$represents the difficulty of calculating the $n+1th$ cycle of Class B chain for sub-chains.

$If\ \frac{AT}{ET} > 1.05$, the formula is as follows:

$$H_{A_{n+1}} = H_{A_n} \cdot \frac{AT}{ET} \cdot max\left\{\left\{max\left[min\left(\alpha \cdot \frac{y_n}{x_n},\ \mu\right), \xi\right] * \beta + \min\left(\frac{2016 \cdot \alpha}{x_n(\alpha+1)}, \mu\right) * \gamma\right\} * T_{A_n},\ 1\right\}$$

$$H_{B_{n+1}} = H_{B_n} \cdot \frac{AT}{ET} \cdot max\left\{\left\{max\left[min\left(\frac{x_n}{\alpha \cdot y_n},\ \nu\right), \sigma\right] * \beta + max\left[min\left(\frac{2016}{y_n(\alpha+1)}, \mu\right), \xi\right] * \gamma\right\} * T_{B_n},\ 1\right\}$$

$If\ 0.95 < \frac{AT}{ET} < 1.05$, the formula is as follows:

$$H_{A_{n+1}} = H_{A_n} \cdot \frac{AT}{ET} \cdot \left\{max\left[min\left(\alpha \cdot \frac{y_n}{x_n},\ \mu\right), \xi\right] * \beta + \min\left(\frac{2016 \cdot \alpha}{x_n(\alpha+1)}, \mu\right) * \gamma\right\} * T_{A_n}$$

$$H_{B_{n+1}} = H_{B_n} \cdot \frac{AT}{ET} \cdot \left\{max\left[min\left(\frac{x_n}{\alpha \cdot y_n},\ \nu\right), \sigma\right] * \beta + max\left[min\left(\frac{2016}{y_n(\alpha+1)}, \mu\right), \xi\right] * \gamma\right\} * T_{B_n}$$

$If\ \frac{AT}{ET} < 0.95$ ， , the formula is as follows:

$$H_{A_{n+1}} = H_{A_n} \cdot \frac{AT}{ET} \cdot min\left\{\left\{max\left[min\left(\alpha \cdot \frac{y_n}{x_n},\ \mu\right), \xi\right] * \beta + \min\left(\frac{2016 \cdot \alpha}{x_n(\alpha+1)}, \mu\right) * \gamma\right\} * T_{A_n},\ 1\right\}$$

$$H_{B_{n+1}} = H_{B_n} \cdot \frac{AT}{ET} \cdot min\left\{\left\{max\left[min\left(\frac{x_n}{\alpha \cdot y_n},\ \nu\right), \sigma\right] * \beta + max\left[min\left(\frac{2016}{y_n(\alpha+1)}, \mu\right), \xi\right] * \gamma\right\} * T_{B_n},\ 1\right\}$$

Based on the recent 4 cycles' block production, T_{A_n} and T_{B_n} are determined as follows: $T_{A_n} = 1.2^{\frac{x_{n-1}-x_n}{x_A}}$, $T_{B_n} = 1.2^{\frac{y_{n-1}-y_n}{y_B}}$, Here, x_A is calculated as: $x_A = \frac{1}{2}|x_n - x_{n-1}| + \frac{3}{10}|x_{n-1} - x_{n-2}| + \frac{1}{5}|x_{n-2} - x_{n-3}|$. If x_A equals 0, then x_A is set as $x_n - x_{n-1}$. This means that the more recent block production difference reflects the change in computational power of the corresponding chain more accurately, and it has a higher weight in the calculation. The calculation for y_B is similar to x_A .When $0 < x_A \leq 15$ and $|x_n - x_{n-1}| \leq 20$, x_A is set as 56. This is because when $0 < x_A \leq 15$, it indicates that the chain with higher computational power has had

relatively stable block production in the recent cycles. If $|x_n - x_{n-1}|$ is also small in this case, it suggests that the block production and network is becoming stable, and x_A is increased to further reduce the impact of these factors. The goal is to make $\frac{x_{n-1}-x_n}{x_A}$ as close to 1 as possible, aiming for a smaller adjustment magnitude in difficulty.

Similarly, when $0 < y_B \leq 8$ and $|y_n - y_{n-1}| \leq 10$, y_B is set as 28. The reason for this setting is the same as for x_A.

Building upon the algorithm mentioned above, to prevent sudden increases or decreases in the computational power of A and B-class chains and combination attacks, the following adjustments will be made if $y_n \leq 6$:

if $\frac{AT}{ET} \geq 1$ and $c_{less6} \geq 3$,then $H_{B_{n+1}} = H_{B_n} \cdot \log_\delta(\delta + \frac{AT}{ET})$

Otherwise:

$$H_{B_{n+1}} = H_{B_n} \cdot \log_\delta(\delta + \frac{AT}{ET} \cdot \left\{max\left[min\left(\frac{x_n}{\alpha \cdot y_n}, \nu\right), \sigma\right] * \beta + max\left[min\left(\frac{2016}{y_n(\alpha+1)}, \mu\right), \xi\right] * \gamma\right\} * T_{B_n})$$

If $y_n \geq 500$, adjustments will be made to $H_{A_{n+1}}$ based on the following formula:

if $\frac{AT}{ET} \geq 1$ and $c_{more500} \geq 3$, then $H_{A_{n+1}} = H_{A_n} \cdot \log_\delta(\delta + \frac{AT}{ET})$

Otherwise:

$$H_{A_{n+1}} = H_{A_n} \cdot \log_\delta(\delta + \frac{AT}{ET} \cdot \left\{max\left[min\left(\alpha \cdot \frac{y_n}{x_n}, \mu\right), \xi\right] * \beta + \min\left(\frac{2016 \cdot \alpha}{x_n(\alpha+1)}, \mu\right) * \gamma\right\} * T_{A_n})$$

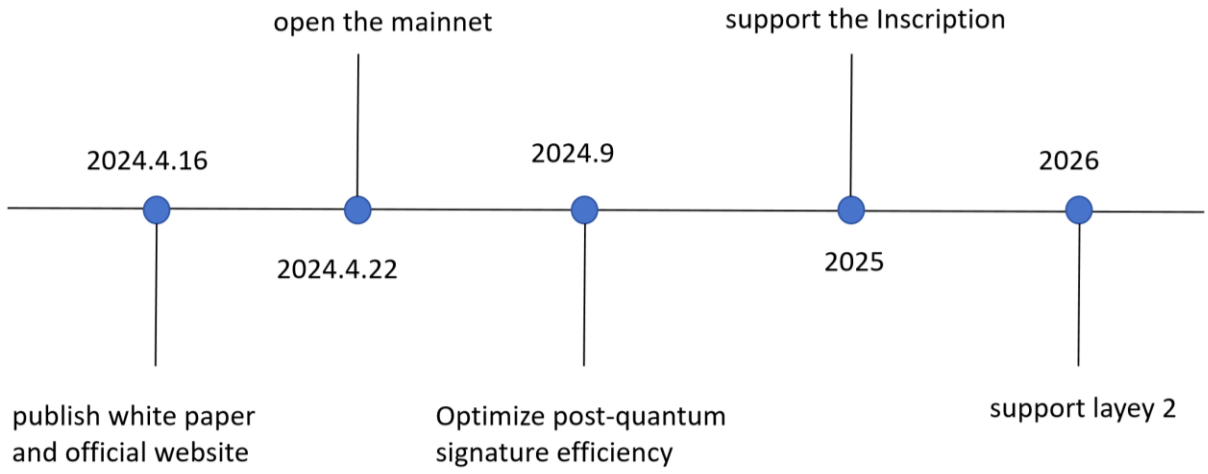
Where $\frac{actual\ time}{expected\ time}$ represents the ratio of the actual block generation time of 2016 blocks to the expected block generation time, which is limited in each formula $\min\left\{max\left(\frac{actual\ time}{expected\ time}, \frac{1}{4}\right), 4\right\}$. This part reflects the network block production speed, changes in computing power in the network, etc. If blocks are produced too frequently recently, then the mining difficulty will increase, otherwise, the mining difficulty will decrease.

When there are less than four adjustment periods, the difficulty of the two parent chains will be adjusted separately according to the logic similar to the Bitcoin difficulty adjustment algorithm. The formula is as follows:

$$Diff_{n+1} = Diff_n \frac{nActualTime}{nExceptTime} = Diff_n \frac{nActualTime}{2016 * 5minutes}$$

Here, $Diff_{n+1}, Diff_n$, $nActualTime$ and $nExceptTime$ respectively represent the new difficulty target for the next stage, the difficulty target for the current stage, the total time taken to generate the most recent 2016 valid blocks in the network, and the expected time for generating 2016 blocks (2016 * 5 minutes). Additionally, each difficulty adjustment is also limited within an upper and lower boundary of five times the previous difficulty.

6. RodaMap



7. Token Allocation

Taking inspiration from Bitcoin's token economic model, the token allocation will follow a halving cycle.

Total Supply of PQC: 168 million PQC (pqcoin);

Each PQC is composed of 100 million MAXWELL;

Total Supply of MAXWELL: 18 trillion MAXWELL;

Miner Block Rewards: A total of 168 million tokens will be allocated as block rewards for miners. The target block time is set to 5 minutes. To incentivize early-bird participation, the initial

26,250 blocks will be designated as an early-bird period, with a reward of 1,024 PQC per block. Subsequently, each block will have a reward of 168 PQC. Every 2,016 blocks (approximately one week), a difficulty adjustment will occur. Every 420,000 blocks, the reward will be halved. This halving process will repeat 33 times.

8. Contact and Community:

Web: www.pqcoin.net
Contact: Postquantumcoin@outlook.com
X:https://twitter.com/PQ_coin
Discord:https://discord.gg/gGW2p5BuVJ