Proof-of-randomness protocol for blockchain consensus:

the white paper version 1.0

Wen-Zhuo Zhang* and Victor Kai

QUAKEY Ltd.

(Dated: November 29, 2022)

Abstract

A proof-of-randomness (PoR) protocol could be a fair and low energy-cost consensus mechanism for blockchains. Each network node of a blockchain could use a true random number generator (TRNG) and hash algorism to fulfil the PoR protocol. In this whitepaper, we give the consensus mechanism of the PoR protocol, and show how it could make the random numbers unforgeable. The PoR protocol could generate a blockchain without any competition of computing power or stake of cryptocurrency. Besides, we give some advantages of integrating quantum random number generator (QRNG) chips in hardware wallets, and also discuss the route to cooperate with quantum key distribution (QKD) technology.

*Electronic address: sturmanz@hotmail.com
I. INTRODUCTION

Blockchain is a decentralized data recording technology of a network. All network nodes are distributed in space, while blocks are distributed in time-order with a chain connection. Usually, the current block records its generation timestamp, the private key and the nonce of the node who generate the current block, the hash value of the last block’s data, and all transactions in the network after last block. Consensus algorithm plays the central role of generating blockchains. The first blockchain, bitcoin[1], use proof-of-work (PoW) to select a network node to generate a new block. The PoW requires a hashing computer to calculate a hash value with very small probability. For example, a 256-bit hash value with all zeros in highest 70 bits. Such method consumes more and more computing power, which means it is wasting a lot of electricity energy. Recently, Ethereum [2] replace its PoW algorithm by proof-of-stake (PoS) method to save more than 99% power, however, the fairness of the PoS is questioned.

In this whitepaper, we present a physically fair consensus algorithm called proof-of-randomness (PoR), which is also a low-energy cost method. The PoR requires true random number generator (TRNG) hardware on each network node. The fairness comes form the physical true random number, while the main power consumption is from a TRNG chip which is usually less than 100mW on each node.

II. GENERATION AND TESTING OF TRUE RANDOM NUMBERS

Before running PoR protocol, every network node should acquire an array of binary true random numbers form a TRNG hardware. For stationary computers such as servers or desktop PCs, a PCI card with TRNG chips would be good. For much more mobile computers such as laptops, smart phones or tablet PC, a mobile USB device with quantum random number generator (QRNG) chips is a good choice. Since quantum randomness is the fundamental true randomness of the nature (Einstein called it ’God plays dice’), QRNG performs much better than classical TRNG in complex environment and also has much lower energy consumption as well as much smaller size than classical TRNG chips [3]. A mobile hardware wallet with QRNG chip would be a suitable solution for network nodes.

The quality of random numbers could be test by blockchain softwares by embedding testsuits, e.g., NIST SP 800-22 randomness testsuit[4]. This step is necessary to forbid any pseudo random
number generator (PRNG) since any pseudo random number is indeed determined and not fair to PoR. Typically, it requires 256 Mb binary random numbers to finish the test. When a node’s QRNG (or TRNG) chip has output 256 Mb random number data and passed the test, the blockchain software could split the 256 Mb data into 256 pieces, with each piece’s length at 1Mb, for example. Then the blockchain software could get a random byte from the QRNG (or TRNG) chip as the piece number to select out 1Mb random number date. We name it as \( R_m(n) \) with the network node number \( m \) and the blockchain time-sequence number \( n \).

### III. PROOF-OF-RANDOMNESS PROTOCOL

When a network node finishes its true random number test, the selected 1 Mb random number \( R(m, n) \) is calculated by SHA-256 to get the 256-bit hash value \( H_m(n) \) which is then sent to other nodes together with this node’s signature.

In the case that the network is relatively small, assuming \( M \) nodes, synchronizing of data among all \( M \) online nodes could be within several minutes. Then each node can receive every other online node’s 1Mb random number. So every node would acquire the same sum of 1 Mb random numbers, which is

\[
S_H(n) = \sum_{m=1}^{M} H(m, n).
\]

(1)

The sum is independent with the order of receiving other nodes’ \( R(m, n) \) on a node. Every node should calculate the hash value of \( S_H(n) \) by SHA-256 and broadcast the result \( HS(n) \) to all other nodes together with its digital signature. Here we call \( HS(n) \) the second hash value in order to distinguish the first hash value \( H_m(n) \) of every node. Since \( HS(n) \) depends on the 1 Mb random number of every node, it is unpredictable. Then the absolute difference value \( D(m, n) = |HS(n) - H_m(n)| \) is also unpredictable, which prevents the attacks of using controllable random numbers. A consensus of choosing the node with minimal or maximal \( D(m, n) \) as the \( n \)th block’s owner would be clear.

In the case that the network is relatively big, synchronizing of data among all nodes is not possible. It would probably happen in all public blockchains. Here we should not calculate the second hash value \( HS(n) \) anymore since different nodes may get different values. Instead, every node calculate the hash value of \( H_m(n) + H(n - 1) \) by SHA-256 and broadcast the result \( H'_m(n) \), where \( H(n - 1) \) is the prevhash value of the last block. Then every node calculate \( D(m, n) = |H'_m(n) - H_m(n)| \) and broadcast \( D(m, n) \) together with its digital signature again. A consensus of
choosing the node with minimal or maximal $D(m, n)$ as the $nth$ block’s owner is still work.

However, a kind of attack could be possible in such relatively big network, which is to try multiple times generation of random numbers and select a much closer $R(m, n)$ to the target value (minimal or maximal $D(m, n)$). A node with more QRNG chips or with expensive TRNG devices could do the attack. In order to prevent such kind of drawback to PoW (with a competition of random number generating power instead of CPU computing power), a node $m$ should broadcast its time-stamp of finishing the true random number test, $t_1(m, n)$, and its time-stamp of finishing $D(m, n)$, $t_2(m, n)$. Then a consensus of choosing the node with minimal $D(m, n)\Delta t(m, n)$ as the $nth$ block’s owner could be reasonable, where $\Delta t(m, n) = t_2(m, n) - t_1(m, n)$. If a node try to generate random numbers by multiple times, $\Delta t(m, n)S$ would increase and reduce its possibility of get minimal $D(m, n)\Delta t(m, n)$. Besides, $\Delta t(m, n)$ should set a minimal value in order to be not attacked by forging timestamps.

IV. OTHER ADVANTAGES OF USING QRNGS IN BLOCKCHAINS

Blockchains adopt elliptic curve cryptograph (ECC) as the main encryption and decryption method. The ellipse curve equation is

$$y^2 = ax^3 + bx + c,$$  \hspace{1cm} (2)

while most of current blockchains use a simplified elliptic curve equation, $y^2 = x^3 + 7$.

There are three kind of random numbers in ECC. The first one is a base-point on the elliptic curve, $G$, which starts a half-line together with the base point. The second one is the private key, $k$, which is the reflection times of the half-line on the elliptic curve. The public key $K$ is the final point
of the half-line on the elliptic curve determined by $G$ and $k$, and expressed by a "multiplication" defined on the curve, which is $K = G \ast k$. Since it is extremely difficult to calculate $k$ with open $G$ and $K$ values by a classical computer, the ECC is secure to all classical computers now.

The third random number is $R$, which encrypt the data $D$ together with the public key $K$. The encryption equations are

$$E_1 = K \ast R + D$$  \hspace{1cm} (3)
$$E_2 = G \ast R,$$  \hspace{1cm} (4)

while the decryption equation with private key $k$ is

$$E_1 - E_2 \ast k = K \ast R + D - E_2 = G \ast R.$$  \hspace{1cm} (5)

Here we can see when $R$ is not safe, e.g. a pseudo random number that would be attacked by stealing its short seed, an attacker could obtain $D$ with $D = E_1 - K \ast R$ since both $K$ and $E_1$ are open.

When QRNGs are evolved in network nodes, both $k$ and $R$ can use quantum random numbers to prevent any crack by computers. It makes $E_1$ much safer to travel over the network. Therefore, a hardware wallet with a QRNG could be an upgrade of ECC’s security, which is also an upgrade of blockchain’s security.

Besides, QRNG chips in hardware wallets could provide quantum keys to encrypt their private keys in order to recover them. For current hardware wallets, when a wallet is lost, the private key is also lost. Then all the cryptocurrency bounding to this private key would be lost forever. When a hardware wallet use a quantum key $q$ from a QRNG chip to encrypt its private key $k$, the encrypted data $E$ could leave the wallet and be stored in a disk or on cloud. While the quantum key could be stored in an off-line disk which has no connections to $E$. If the hardware is lost, one can recover its private key by decrypting $E$ with $q$. This is another advantage of hardware wallets with QRNG chips.

V. IN COOPERATION WITH QUANTUM KEY DISTRIBUTION (QKD)

Since ECC is vulnerable to quantum computers, blockchains should upgrade their cryptograph to keep the future security when a general quantum computer with millions of qubits comes to
reality. Post-quantum cryptograph (PQC) is a natural solution which cost a general quantum computer much longer time to break than RSA-2048 and ECC. However, it does not guarantee that a future quantum algorithm which breaks the PQC in a very short time never happen.

Quantum key distribution (QKD) could be the terminate solution to prevent the attack form general quantum computers [5]. In QKD, all the keys are quantum random numbers, and any interception of key distribution would break it. Therefore, QKD is unconditional secure to any algorithm when use one-time encryption. However, QKD network rely on optical fibers (or even satellites) which makes its terminals briefly on servers. If every node of a network have a server together with a QKD terminal, one can use one-time keys form QKD network to do peer-to-peer authentication between any two nodes.

However, QKD terminals can not be used in mobile networks since there is no single-photon technology in radio-frequency or microwave. Here we could use QRNG chips in a mobile node to do an off-line local distribution of quantum keys to a server node with a QKD terminal. Then an on-line key update could work with current keys encrypting next keys (KEK). This method could work in mobile networks since it is indeed a classical communication encrypted by quantum keys. With this method applied on mobile networks, and QKD applied on fiber networks, a full quantum-safe network could be feasible.

VI. CONCLUSIONS

In this white paper, we present a proof-of-randomness (PoR) protocol. Such protocol could give a fair and low energy-cost consensus mechanism to blockchains with true random number generators (TRNG) and hash algorithm. Since low-cost quantum number generators (QRNG) make the true random number resource more accessible to common users, this PoR protocol could popularize the blockchain technology, especially public blockchains.

Besides, a hardware wallet with a QRNG chip could not only output quantum random numbers to do the PoR protocol, but also make elliptic curve cryptograph ECC by replacing all pseudo random numbers by quantum random numbers. Even more, a quantum random number could be a quantum key to encrypt the private key in the wallet. A backup of the encrypted private key and the quantum key in different devices could recover the private key in the case of the hardware wallet is lost.

Finally, QRNG chips could cooperate with quantum key distribution (QKD) in order to prevent
any attack form general quantum computers to blockchains in the future.

VII. ACKNOWLEDGEMENT

As a MiraclePlus alumni company (F22), we thank MP for its accelerator program and founders’ community.

[1] https://nakamotoinstitute.org/bitcoin/
[2] https://ethereum.org/en/whitepaper/
[3] https://www.idquantique.com/random-number-generation/overview/
[4] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf;
[5] V. Scarani, et. al., Rev. Mod. Phys. 81, 1301 (2009); F. Xu, X. Ma, Q. Zhang, H. Lo, and J. Pan, Rev. Mod. Phys. 92, 025002 (2020)