Quantum secured blockchain framework for enhancing post quantum data security

Wasim Raja Mondal

MTSU, September 30th, 2025

Email: wmondal@mtsu.edu

Outline of the talk

- **Motivation**
- **The problem being solved**
- **Current solutions: system architecture, implementation and results**
- **Conclusion**

Motivations

Blockchain technology is vulnerable to quantum computers

How Quantum Computers can threat Blockchain?

- Breaking public-key Cryptography (Shor's algorithm)
- Disrupting Hash function (Grover's algorithm)

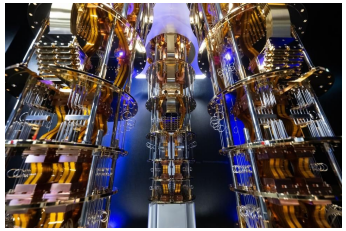


Image source: https://www.forbes.com/sites/johnkoetsier/2025/09/25/massive-quantum-computing-breakthrough-long-lived-qubits/

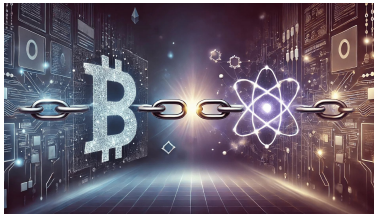


Image source:https://thequantuminsider.com/2024/09/08/blockchain-and-quantum-computing-are-on-a-collision-course-expert-warns/

What is Quantum Computer? A new type of computer that leverages principles of quantum mechanics (superposition,entanglement).

- Superposition: An array of n classical bits can represent only one of 2^n possible combinations at any given moment. An array of n qubits in superposition can represent all 2^n combinations simultaneously. This ability, known as quantum parallelism, allows a quantum computer to explore a vast solution space in a single step.
- Entanglement: Entanglement is a special and fragile correlation between two or more qubits that makes their states interdependent.
  - While superposition offers massive parallel processing, entanglement provides the ability to link the computational work of multiple qubits.
  - Entangled qubits are a core component of quantum error-correction techniques.

Background

Various research activities have been initiated for securing the Blockchain against Quantum computing

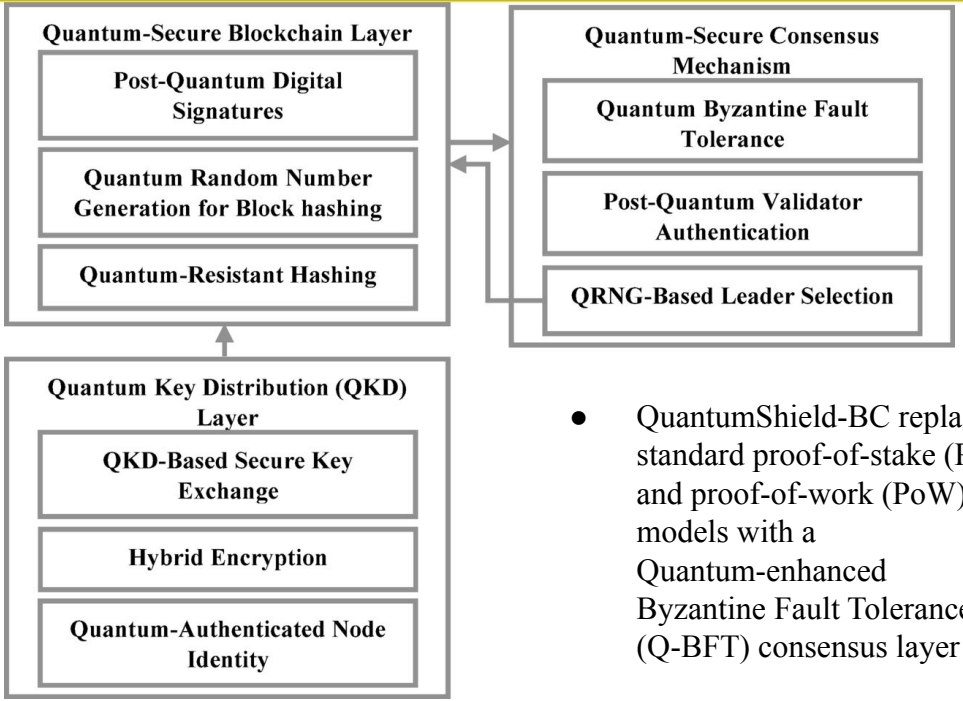
What research activities have been already performed?

- 2016: The Quantum Resistance Ledger (QRL) project on quantum-resistant cryptography (QRL)
- 2020: Concept of Quantum Blockchain was introduced based on Quantum entanglement and DPos(QB Chain)
- 2021: A hybrid model based on quantum blind signatures was introduced for security in healthcare(hbc)
- 2025: Dwave introduces Quantum Blockchain architecture (QbcDave)

What is the contribution of this work?

- This paper proposes a comprehensive blockchain framework, QuantumShield-BC
- It integrates PQC, QKD, and QRNG in an end-to-end manner in a single blockchain protocol
- It minimizes trust in random numbers via QRNG-based consensus and leader election that achieves strong security for random numbers
- It fosters the gradual introduction of PQC by modular, layered architecture that allows any underlying blockchains to be converted to post-quantum blockchain systems incrementally.

System architecture of QuantumShield-Bc



- QuantumShield-BC replaces standard proof-of-stake (PoS) and proof-of-work (PoW) models with a Quantum-enhanced Byzantine Fault Tolerance (Q-BFT) consensus layer

The framework combines encryption with lattice-based digital signatures, QKD-secured communication, QRNG-enhanced randomness, and a quantum-resistant consensus protocol to create a next-generation, tamper-proof blockchain ecosystem.

- Classical encryption introduces the risk of vulnerabilities in existing classical blockchain security models. QuantumShield-BC addresses this issue by utilizing lattice-based post-quantum digital signatures
- Quantum Key Distribution (QKD) is implemented by creating a secure peer-to-peer communication channel to prevent eavesdropping and man-in-the-middle attacks.
- Deterministic PRNGs pose a significant attack vector for classical blockchains, as they often lead to nonce prediction in digital signatures, which attackers can exploit. QuantumShield-BC further introduces quantum random number generation (QRNG), providing cryptographically secure random values for generating transaction hashes and executing smart contracts, as well as for the consensus mechanism, significantly augmenting entropy and security.

Algorithm for implementations

Post Quantum digital signature algorithm

Input: Transaction T, private key S_k

Output: Validated transaction $V(T)$

1. Compute hash of transaction: $H(T) = Hash_{PQ}(T)$
2. Generate post-quantum signature: $S = Sign_{PQ}(S_k, H(T))$
3. Attach signature to transaction: $T' = (T, S)$
4. Verify signature using public key P_k: $V(T) = Verify_{PQ}(P_k, S, H(T))$
5. If $V(T) = True$, *accept transaction*; else, reject.

- It begins by hashing the transaction data and generating a signature with a private key using a post-quantum algorithm.
- The signature is then attached to the transaction and verified using the corresponding public key.

Quantum Key distribution algorithm

Input: Quantum states Q_S (sent state), Q_R (received state)

Output: Secure cryptographic key K

1. Alice (Sender) generates quantum bits: $Q_S = GenerateQuantumBits()$
2. Alice randomly selects polarization bases (rectilinear or diagonal).
3. Alice sends Q_S over the quantum channel to Bob (Receiver).
4. Bob measures Q_R using randomly chosen bases.
5. Bob and Alice communicate classically to compare measurement bases.
6. Retain only matching basis results to form raw key K_{raw}
7. Perform error correction and privacy amplification: $K = SecureKeyExtraction(K_{raw})$
8. Output K as the final QKD-generated key for encryption.

- The sender transmits quantum states, which the receiver measures using random bases.
- Through classical communication, both parties compare bases and retain matching bits to form a raw key.
- After error correction and privacy amplification, a final secure key is established, ensuring tamper-proof communication against quantum adversaries.

Algorithm for implementations

Quantum resistant hashing algorithm

Input: Block data B_n, previous block hash B_{n-1}

Output: Secure hash $H(B_n)$

1. Concatenate block components: $D = B_{n-1} \parallel T_n \parallel S_n$
2. Apply post-quantum secure hash function: $H(B_n) = Hash_{PQ}(D)$
3. Return $H(B_n)$ as the quantum-resistant hash.

- It concatenates the current block data with the previous block's hash to form the input.
- A secure hash function, such as SPHINCS + or Keccak, is then applied to produce a tamper-proof hash.

Transaction validation algorithm

Input: Transaction T, digital signature S_k, public key P_k

Output: Boolean value $V(T)$ indicating transaction validity

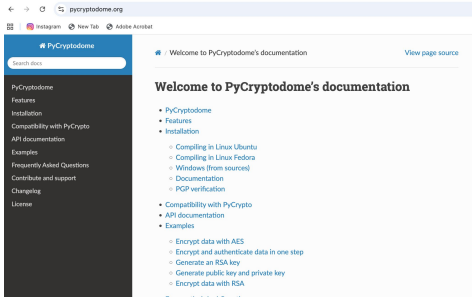
1. Extract transaction data and signature: $(T, S) \leftarrow Extract(T)$
2. Compute hash of the transaction: $H(T) = Hash_{PQ}(T)$
3. Verify the digital signature: $V(T) = Verify_{PQ}(P_k, S, H(T))$
4. Return $V(T)$:

   If $V(T) = True, accept\ transaction.$

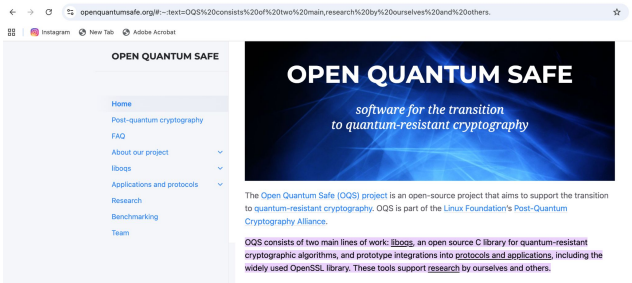
   Otherwise, reject transaction.

- It extracts the transaction and its digital signature, computes the hash of the transaction, and verifies the signature using the sender's public key.
- If the verification is successful, the transaction is accepted; otherwise, it is rejected.

Set-up details



[PyCryptodome]

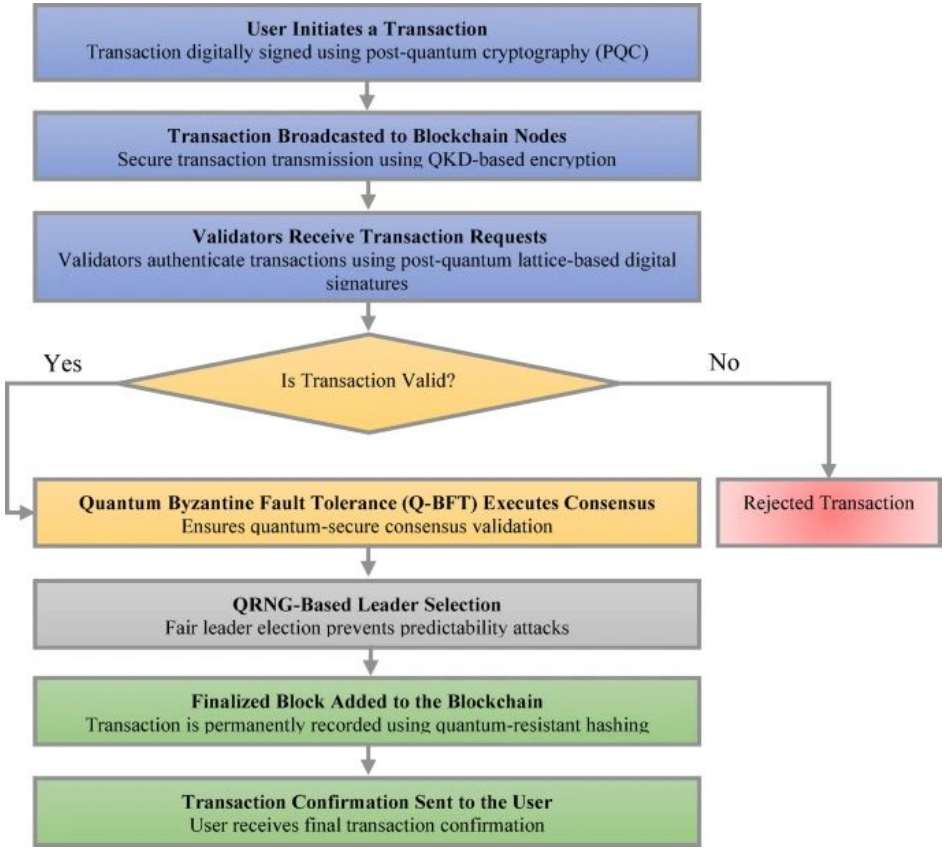


[OpenQuantumSafe]

[QRNGAPI]

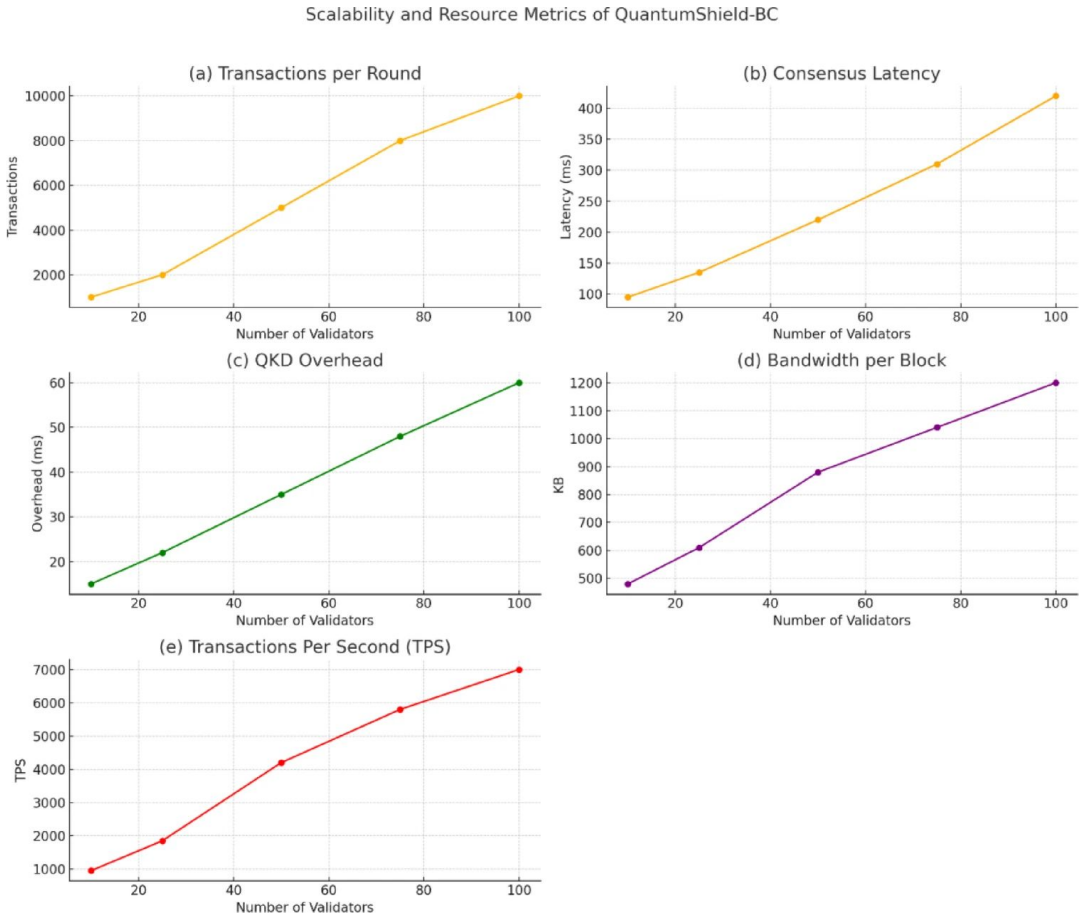
- In the simulation environment, 25 validator nodes, each operated by a Docker container within a local distributed environment, were utilized.
- Each container adopted a blockchain node style, featuring independent signing, verification, and consensus modules, connected via a virtual network with QKD emulation channels.
- The post-quantum digital signatures implemented in the prototype are based on CRYSTALS-Dilithium (only level 2), Falcon (512-bit), and SPHINCS + signatures to benefit from NIST standardization and implementation support.
- They experimented with the prototype, which was run on an Ubuntu 22.04 system with a 16-core CPU, 64GB RAM, and Python 3.10.

Workflow of QuantumShield-Bc



- A user-initiated transaction digitally signed using post-quantum cryptography.
- The transaction is then securely transmitted across blockchain nodes using quantum key distribution (QKD).
- Validators authenticate the transaction using lattice-based signatures, followed by QRNG-based leader selection to ensure unbiased consensus initiation.
- The Quantum Byzantine Fault Tolerance (Q-BFT) protocol is executed for multi-party consensus, and upon reaching agreement, the block is finalized and appended to the blockchain.
- The user receives a confirmation, completing the secure and quantum-resilient transaction flow.

QuantumShield-BC scales well and consumes fewer resources



- Figure(a), we observe a linear increase in the number of transactions per round from 10 to 100 validators, showcasing one of the system's throughput limits.
- Figure (b): Consensus latency as a function of the number of validators grows slowly as we increase the validator size. One more signature must be cryptographically verified for each additional validator, but remains less than an order of magnitude arising from PoW-based delays.
- Figure (c), QKD overhead continues to grow gradually, exhibiting the expected extra key exchanges, but remaining tolerable as optimal key reuse strategies mitigate it.
- Figure (d), Bandwidth consumption (per block) is also growing primarily due to post-quantum signature size, but it is still scalable

Classical and QuantumShield-Bc

Feature/metric	Classical blockchain (ECC + PoW)	QuantumShield-BC (PQC + QKD + Q-BFT)
Digital signature scheme	ECDSA (256-bit)	CRYSTALS-dilithium (level 2)
Consensus mechanism	Proof-of-work (PoW)	Quantum BFT (Q-BFT)
Key exchange	Classical public key exchange	Quantum key distribution (QKD)
Nonce generation	Pseudo-random generator	Quantum random number generator (QRNG)
Block finalization time	High (due to mining)	Low (QRNG-based leader selection)
Signature verification time	Fast	Moderate
Energy consumption (consensus)	High	Low
Resistance to quantum attacks	Weak (Shor's applicable)	Strong (post-quantum secure)
Sybil attack mitigation	No (PoW susceptible)	Yes (validator authentication via PQC)
Replay attack mitigation	Limited	Strong (QRNG-based nonce uniqueness)

Note that Energy consumption for the Quantum-BC is low compared to classical in addition to strong resistance to quantum attack.

Limitation of the current study

Theoretically unbreakable but several limitations:

1. The imno QKD implementation was simulated, rather than deployed with real quantum hardware, which may not capture all operational complexities.

2. Practical implementations of Quantum Key Distribution (QKD) raise several issues. The significant challenges include the high hardware costs of quantum photon sources, detectors, and synchronization systems

3. The system cannot be generalized to non-public networks, as it was tested on a controlled testbed of a maximum of 100 validators.

4. Post-quantum algorithms, such as Dilithium and SPHINCS+, utilize larger keys and signatures with higher computational complexity, which can impact performance, especially in resource-constrained environments.

All of these would require further investigation in a real-world setting to ensure scalability, efficiency, and the possibility of integrating with legacy blockchain and quantum infrastructure ecosystems.

Conclusions

- Quantum computing will inevitably threaten security landscapes by breaking well-established cryptographic primitives, including the public keys used for blockchain security
- The framework, which is designed, simulated, and evaluated systematically, achieves enhanced transaction authenticity, consensus fairness, and quantum-resistant security, compared to classical blockchain designs.
- When quantum hardware matures, real-world QKD channels could be implemented, optimizing the PQC algorithms for low-resource usage and scaling the system to thousands of validators.
- Exploring the multi-chain potential of QuantumShield-BC and the benefits of brilliant contract execution in a post-quantum world would also be interesting avenues to pursue.
- Finally, QuantumShield-BC provides a building block for quantum-safe distributed systems.
- Its myriad cryptographic, communication, and consensus-level protections make it a stepping stone toward quantum-proof blockchain applications in finance, healthcare, and critical infrastructure.
- Its scalability, hardware adaptability, and operational readiness will be enhanced through further research, enabling deployment in blockchain environments.