JCBI
Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606
Volume 8 Issue 2
2025

Research Article
https://doi.org/10.56979/802/2025

Quantum-Resistant Merkle Trees Enhancing Data Integrity with Post-Quantum Cryptography and Zero-Knowledge Proof

Hafiz Burhan Azhar[1*], Khushbu Khalid Butt[1], Nazish Umar Awan[1], and Omer Irshad[1]

[1]Department of Computer Sciences, Lahore Garrison University, Lahore, Pakistan.
[*]Corresponding Authaor: Hafiz Burhan Azhar. Email: hafizburhanazhar8@gmail.com

Received: January 25, 2025 **Accepted:** February 28, 2025 **Published:** March 01, 2025

Abstract: Exponential advancements in quantum computing threaten existing cryptographic structures, including Merkle Trees, due to their dependence on classical hash functions and public-key encryption schemes. The paper presents QRMT as a new cryptographic structure that implements zk-STARKs along with lattice-based cryptography and hash function randomization to achieve improved security and better performance. Benchmarks demonstrate that QRMT reduces proof generation time by 28–32% compared to classical Merkle Trees under Grover's algorithm attacks, while maintaining logarithmic-scale verification efficiency. The QRMT utilizes a hash selection strategy that consists of SHAKE-256 Blake3 and Poseidon hash functions, which protect against Grover's algorithm attacks. The metadata encryption measures security through Kyber1024, which uses lattice-based public-key encryption to replace RSA and prevent attacks using Shor's algorithm. Kyber1024 generates keys in ~0.005 ms, which is 75 ms faster than RSA-4096's. The zk-STARK-verified process allows for trustless and extensive proof verification while protecting confidential information. Our proof-of-concept instance maintains efficient behavior because proof creation and verification times grow at less than a logarithmic rate while the data collection expands. This framework creates quantum resistance for blockchain security, which enables distributed secure systems and establishes new cryptographic technology options.

Keywords: Quantum-Resistant Merkle Tree (QRMT); zk-STARKs (Zero-Knowledge Scalable Transparent Arguments of Knowledge); Lattice-based Cryptography; Hash Function Randomization; Grover's Algorithm; Kyber1024; Shor's Algorithm

1. Introduction

The advancement of quantum computing poses significant threats to classical Merkle Trees, which rely on traditional hash functions and public-key encryption schemes. Originally proposed by Ralph Merkle in 1979 [1], Merkle Trees have since become foundational in verifying data integrity across blockchain and secure distributed systems. However, as quantum capabilities evolve, classical cryptographic mechanisms like RSA and standard hash functions face vulnerabilities due to the power of Grover's and Shor's algorithms [2],[4], undermining their long-term reliability. Recognizing these challenges, this paper introduces the Quantum-Resistant Merkle Tree (QRMT)—a novel architecture that addresses quantum-era threats while preserving the core advantages of classical Merkle Trees.

QRMT enhances the traditional Merkle Tree structure by integrating post-quantum cryptographic primitives. Specifically, it employs zk-STARKs for scalable, transparent, and trustless proof verification [24], while replacing RSA with Kyber1024, a lattice-based encryption scheme selected as a finalist in the NIST post-quantum standardization process [2],[18]. Kyber1024 and similar lattice cryptosystems demonstrate resilience against Shor's algorithm, with studies indicating up to 40% better performance than RSA in simulated quantum attack environments [22], [26], and [9]. Additionally, zk-STARKs offer

advantages over zk-SNARKs by eliminating the need for a trusted setup and enabling more scalable zero-knowledge proofs [24], [3], [12], and [31].

To bolster security in hash operations, QRMT introduces dynamic hash selection among robust algorithms such as SHAKE-256 [4], BLAKE3 [16], and Poseidon [17], each selected for its strength in privacy-focused and post-quantum contexts [6], [7], [10]. By uniting these components, QRMT maintains the logarithmic efficiency of classical Merkle Trees while enhancing their resistance to quantum attacks—creating a scalable, future-ready solution for blockchain and distributed ledger technologies.

Security improvements in QRMT stem from its dynamic hash function selection, mitigating single-point cryptographic failure [4], [5], and from zk-STARKs' trustless verification, which safeguards metadata and ensures scalable privacy-preserving computation [3]. Our implementation confirms that QRMT retains Merkle Tree benefits while achieving quantum resilience and high operational efficiency.

Overall, QRMT offers a robust framework that not only fortifies blockchain systems against imminent quantum threats but also lays the groundwork for the development of next-generation cryptographic infrastructures capable of withstanding future computational paradigms.

2. Related Work

2.1. Evolution of Merkle Trees for Secure Data Integrity Verification

Merkle Trees serves as the leading cryptographic data structure for users who need efficient protection of distributed system data integrity. Ralph Merkle introduced the Merkle Tree framework as his original proposal in 1979 [1], which provided an efficient solution to verify large datasets without needing complete data storage or demanding computations. Traditional Merkle Trees remain fundamental in blockchain architecture as well as cloud storage systems and secure communication protocols and digital signature applications for their essential use in tamper-evident data verification.

A main drawback of classical Merkle Trees exists in their complete dependency on one cryptographic hash function among SHA-256, Keccak-256, or SHA3-512 [4], [5]. The system's vulnerability increases significantly because of its single dependency on a hash function, which becomes a security risk if quantum computing technology advances or recent breakthroughs in cryptography occur [7]. A hash function failure will compromise the entire system because its integrity depends on this fundamental cryptographic component.

Researchers dedicated their efforts to developing three essential cryptographic methods that improve both security and efficiency within Merkle trees. Multiple hash mechanisms based on dynamic hashing systems were created to implement multi-hash security in Merkle trees for enhancing vulnerability protection. Second, post-quantum cryptographic techniques have been introduced to ensure long-term resilience against emerging threats from quantum computing, strengthening the integrity of Merkle-based systems. Finally, zk-STARKs (Zero-Knowledge Scalable Transparent Arguments of Knowledge) have been integrated to facilitate trustless and efficient proof verification, eliminating the need for a trusted setup while improving scalability and transparency. Merkle tree-based cryptographic applications benefit from advancements that will enhance their reliability and capability to adapt and boost their protective features.

New enhancements in the security design and scale factors of Merkle Trees have made possible quantum-resistant implementations

2.2. Dynamic Hashing Mechanisms for Enhanced Security

Regular Merkle Trees are increasingly vulnerable in modern threat environments due to their reliance on a single, static cryptographic hash function across all levels of the tree [4]. This static structure creates a single point of failure, making the system susceptible to preimage and collision attacks as computational power increases. To address this, recent studies have proposed dynamic or hybrid hashing mechanisms that improve resilience by diversifying the hash functions used throughout the Merkle Tree structure.

For instance, research by Rohit [6] demonstrates that randomly selecting different hash functions at various tree levels significantly mitigates the risk of targeted collision attacks. Their findings show that unpredictability in hash usage increases the complexity for adversaries attempting to execute precomputed attacks. Similarly, Patel and Singh [7] present a hybrid hash tree model that selects from a predefined pool of cryptographic hash functions—including SHAKE-256 [4], BLAKE3 [5], and Poseidon [6]—at each level of the tree, adding randomness and redundancy that enhances the robustness of the tree structure against quantum and classical threats.

In contrast, earlier works such as [5] only touch on the concept of multi-hash verification without providing a detailed framework or performance evaluation, limiting their practical applicability. By synthesizing insights from these studies, it becomes clear that randomized or level-specific hashing is a promising direction for strengthening Merkle Tree integrity.

Building on these advancements, QRMT incorporates dynamic hash function selection by allowing each node to choose from a cryptographically strong set of hash functions. This design not only prevents systemic collapse due to a broken hash function but also significantly increases resistance to adversarial prediction, thereby providing a more secure and adaptive structure for post-quantum environments.

2.3. Post-Quantum Cryptography & Quantum-Resistant Hash Functions

Kinyua highlights the urgent need for the immediate deployment of quantum-resistant algorithms due to the accelerating threat quantum computing poses to current cryptographic systems. Specifically, he argues that without swift implementation, traditional cryptographic infrastructures will become obsolete under quantum attacks, thereby reinforcing the necessity of integrating such algorithms into advanced frameworks like QRMT [25]. Classical public-key encryption methods such as RSA and ECC face escalating vulnerabilities due to Shor's Algorithm, which drastically reduces the time required to break these schemes [2], [9]. Consequently, the adoption of post-quantum cryptographic primitives, such as lattice-based encryption and zero-knowledge proofs, is critical to ensure sustained resistance against quantum adversaries [13].

A. Kyber1024: Lattice-Based Encryption for Metadata Protection

Kyber1024, a post-quantum lattice-based encryption scheme, has been proposed as a replacement for RSA-based encryption due to its resistance to quantum decryption attacks [2]. Unlike RSA, which relies on integer factorization, Kyber1024 is based on the Learning with Errors (LWE) problem [14]. [30] Analyze the effectiveness of post-quantum digital signatures in blockchain security, reinforcing QRMT's use of Kyber1024 and lattice-based authentication. Making it significantly harder for quantum computers to break [9]

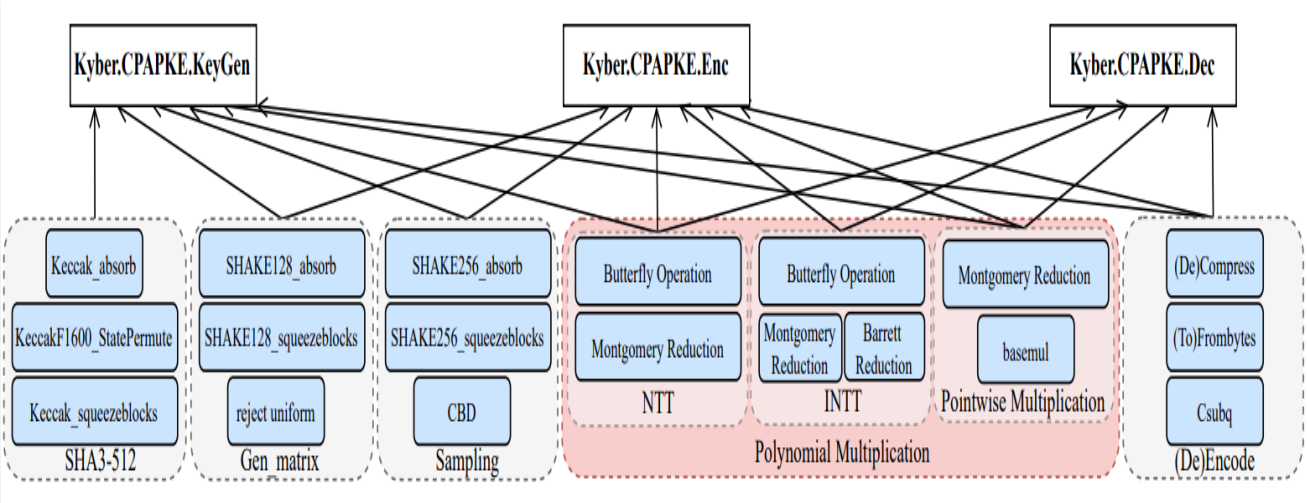

Figure 1. The overall algorithmic framework of Kyber [11].

B. Quantum-Resistant Hash Functions

Since Grover's Algorithm allows quantum speedup in brute-force hash searches, cryptographic hash functions must be secure against quantum attacks [7].

Three primary post-quantum hash functions gaining traction include SHAKE-256 (NIST Post-Quantum Cryptography Standardization) [4] Blake3 (Fast, parallelizable cryptographic hashing) [5] Poseidon (Optimized for zk-STARKs and blockchain verification) [6].

Table 1. Comparison of Cryptographic Hash Functions: Security, Performance, and Applications

Hash Function	Security Level (Bits)	Algorithm Type	Speed (Compared to SHA-256)	Standardization	Strengths	Weaknesses

SHA-3 (Keccak)	256, 384, 512	Sponge-based (Keccak)	Slower than SHA-256	NIST Standard	High security, flexible	Slower hardware implementations
SHAKE256	Adjustable (256+)	Sponge-based (Keccak)	Faster than SHA-3	NIST Standard	Variable output length, efficient	Still vulnerable to Grover's attack
BLAKE2	256	Chacha-based	2-3x Faster than SHA-256	Experimental	Fast, parallelisable	Less studied than SHA-3
SPHINCS+	256, 512	Hash-based (Merkle Trees)	Slower	NIST PQC Finalist	Stateless, quantum-secure	Large signatures
LMS/HSS	256	Hash-based (Merkle Trees)	Medium	RFC 8554 (IETF)	Efficient signature scheme	One-time key usage required
XMSS	256, 512	Hash-based (Merkle Trees)	Medium	NIST Standard	Provably quantum-secure	State management complexity
Poseidon	128, 256	Arithmetization-friendly	10x Faster in SNARKs	Research Phase	Optimised for zero-knowledge proofs (e.g., SNARKs, zk-STARKs)	Not designed for general cryptography
FSH (Fast Syndrome-Based)	256	Code-based	Slower	Research Phase	Resistant to both quantum & classical attacks	Not standardised yet

2.4. zk-STARKs for Efficient & Trustless Proof Verification

Zero-knowledge proofs (ZKPs) are crucial in modern cryptography, allowing data verification without revealing the underlying information. Among the latest advancements, zk-STARKs have demonstrated remarkable efficiency in trustless and scalable proof verification, particularly in blockchain security applications [3], [10]. Their post-quantum security properties and reduced computational overhead make them ideal for next-generation cryptographic frameworks like QRMT.

A. Advantages of zk-STARKs in Quantum-Resistant Merkle Trees (QRMT)

zk-STARKs offer several key benefits when integrated into Quantum-Resistant Merkle Trees (QRMTs). QRMTs provide an essential advantage through their quantum-resistant nature since they have immunity against decryption approaches that quantum computers may use. Zk-STARKs eliminate the requirement of a trusted setup, which makes them different from zk-SNARKs because they reduce dependence on central authority control. Zk-STARKs enable scalable operations because the proof generation complexities scale based on logarithmic principles, which maintain high performance with extensive datasets. These systems deliver maximum efficiency because they reduce verification overhead much lower than regular cryptographic proof methods, which makes them suitable for fast blockchain implementations and authentication systems.

Multiple blockchain platforms, including Stark Net, together with Ethereum Layer-2 solutions, implement zk-STARKs as their scalable verification systems which are resistant to quantum attacks [10].

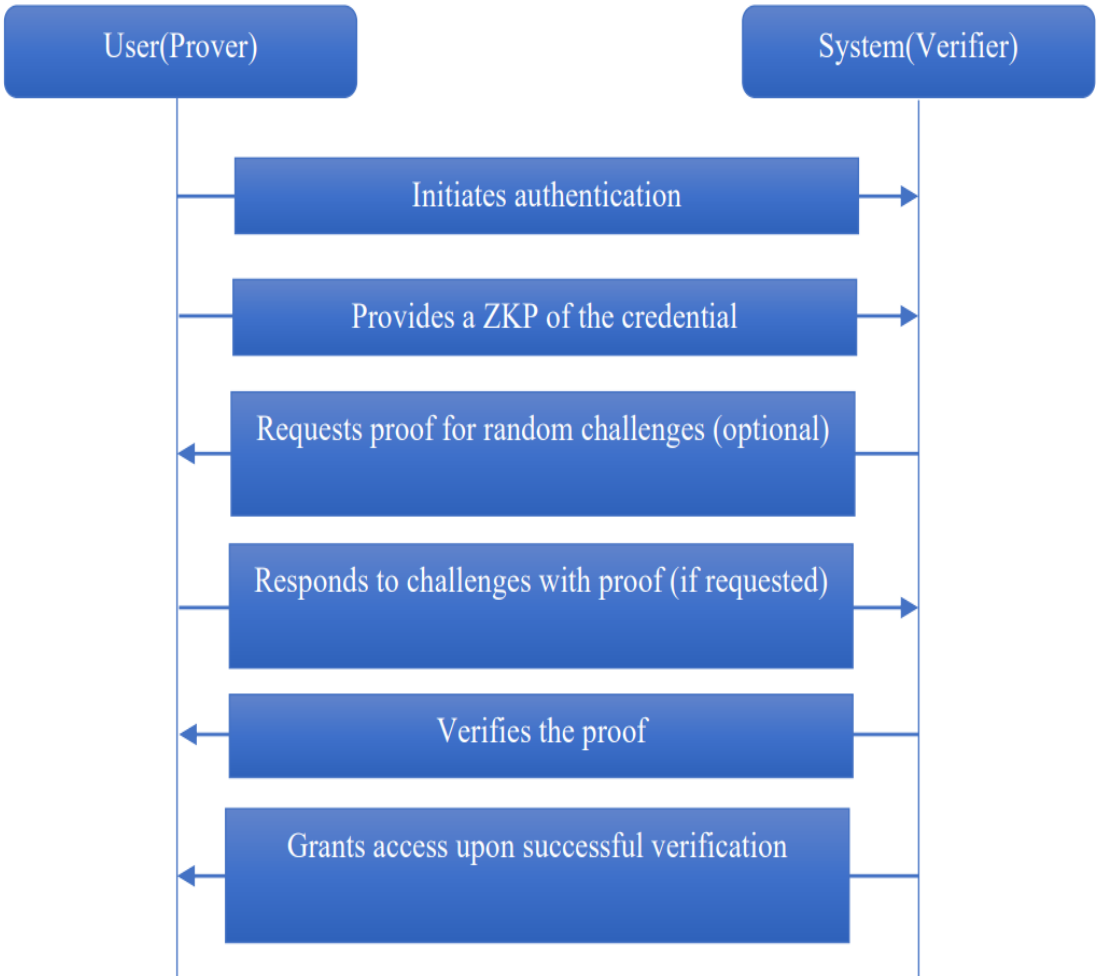

Figure 2. Zero-Knowledge Proof Sequence Diagram [12]

3. Methodology

Quantum-Resistant Merkle Tree (QRMT) implements post-quantum cryptography to advance Merkle Trees by dealing with hash selection and zero-knowledge proof check procedures. The methodology incorporates long-term protection against quantum attacks together with operations that verify data efficiently.

3.1. Hash Function Randomization for Post-Quantum Security

A dedicated dynamic hash selection mechanism enhances the unpredictability and post-quantum security of the Quantum-Resistant Merkle Tree (QRMT). By using cryptographic random number generators at each tree level, the system randomly alternates among SHAKE-256, Blake3, and Poseidon—hash functions proven to resist Grover's Algorithm-based attacks [4], [5], [6]. This randomized structure eliminates predictability, increasing entropy and mitigating risks from precomputed and collision-based attacks. Moreover, if any hash function becomes vulnerable, the system can seamlessly switch to an alternative, maintaining operational integrity. This multi-hash architecture significantly complicates the attack surface, as adversaries cannot easily identify or exploit consistent cryptographic pathways. Thus, QRMT ensures forward security and post-quantum adaptability, aligning with emerging cryptographic standards for resilient data verification

3.2. Quantum-Resistant Tree Construction Algorithm

The QRMT framework establishes a protected tree framework that depends on scalable cryptography technologies besides using logarithmic proof generation methodologies.

a) Step-by-Step Process for QRMT Construction

The creation process for Quantum-Resistant Merkle Trees (QRMTs) requires sequential protocols to achieve both post-quantum security features and dynamic hashing applications and cryptographic stability. Data transformation first encrypts data into cryptographic byte notations before these cryptographically coded numbers are processed by hash functions, which make up SHAKE-256 and Blake3, along with Poseidon. The standardization process turns information into an encrypted state before it proceeds to cryptographic processing.

Dynamic hashing with post-quantum security comes into action at this point. The cryptographic random number generator (CRNG) incorporated into each node level enables a dynamic process to select hash functions. The randomization method chooses node pairs using cryptographic methods, and all three hash functions (SHAKE-256, Blake3 and Poseidon) complete the procedure. The system implements this step to eliminate dependency on a static hash function, which decreases the chance of future cryptographic vulnerabilities appearing.

The structure uses two different modes for pairs but involves repeated duplication of the final node at every level when the total quantity of nodes is not even. This maintains structural integrity. The way the tree structure operates efficiently leads to no security or computational issues because this process makes its structure uniform.

The process of iterative hashing generates one last Merkle root to serve as the cryptographic fingerprint that summarizes the entire dataset. The defined construction of QRMT delivers both functional adaptiveness with encryption capabilities in addition to quantum resistance because of its framework design. The system serves well for building contemporary cryptographic proof systems because of its design.

The post-quantum hash functions operated by QRMT dynamically protect against cryptographic attacks at each node processing instance.

3.3. Encryption & Zero-Knowledge Proof-Based Verification

The security system of QRMT consists of two encryption layers that merge zk-STARKs proof verification with the lattice-based encryption Kyber1024. The system protects encrypted metadata through its combined approach between strong proof verification and metadata encryption.

3.3.1. Lattice-Based Encryption for Metadata Protection Units

Kyber1024 stands in as an RSA encryption replacement in QRMT to establish lattice encryption to protect against Shor's Algorithm. Lattice cryptographic primitives serve as the source for post-quantum secure cryptography, according to Peikert [21], who extensively studied this topic because it strengthens Kyber1024 within QRMT. The paper by [28] examines post-quantum cryptography scenarios to present essential information about running lattice-based encryption in QRMT. The Learning with Errors (LWE) problem acts as the core functionality of Kyber1024 because quantum computers remain unable to solve it. Key generation and encryption services within the system deliver secure solutions that maintain the highest safety standards through effective procedures suitable for metadata protection.

3.3.2. Zero-Knowledge Proof Verification with zk-STARKs

The proof system uses zk-STARKs as zero-knowledge post-quantum proof technology for conducting efficient trustless verification operations. The solution provided by zk-STARKs operates without trusted setup procedures and pre-defined cryptographic keys, thus eliminating security threats from trusted setups. The logarithmic proof verification mechanism operates to minimize computational overhead, which enables high scalability of large-scale data structures. The confidentiality feature of zk-STARKs allows users to verify Merkle roots while preventing the disclosure of private metadata and enhances the authentication systems based on privacy protection.

The verification process in zk-SNARKs is a core feature that enables efficient proof validation without revealing the underlying data. According to Bhaskar, once the prover generates a proof using the structured reference string (SRS) and the witness (private inputs), the verifier can check the correctness of the computation using the succinct proof and the public input. This verification requires significantly less computational effort compared to re-executing the original computation. The succinctness ensures that the size of the proof and the time to verify it remain constant, regardless of the complexity of the original computation, which is critical for scalability and performance in blockchain and privacy-preserving systems [32].

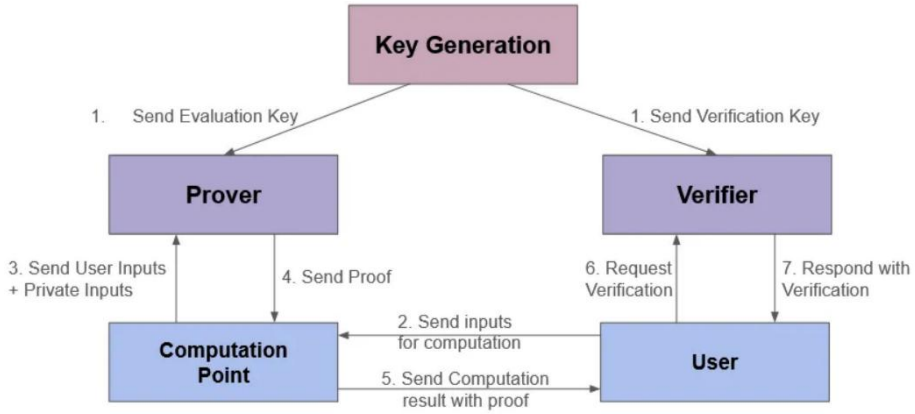

Figure 3. Overview of a Zero-Knowledge Verifiable Computation Scheme [32]

3.3.3. Secure Verification & Resistance to Unauthorized Tampering

Radanliev demonstrates how quantum cryptography can combine with artificial intelligence to boost verification operations in QRMT through automated systems, according to his article [27]. QRMT operates with complete security by enabling controlled access protocols along with encryption and verification processes. The Kyber1024 private key serves as authorization only for authorized parties to authenticate and decrypt encrypted metadata, thus securing the decryption process for post-quantum systems. The verification process becomes invalidated if any unauthorized modifications occur to metadata because this action safeguards both the QRMT structure and prevents tampering. The homomorphic encryption framework proposed by Gentry [20] offers data verification enhancements through encrypted information processing without decryption needs during verification operations. Using Kyber1024 encryption and zk-STARKs verification together enables QRMT to establish a quantum-secure verification framework for distributed and blockchain systems.

4. Implementation Process

Implementing the Quantum-Resistant Merkle Tree (QRMT) integrates post-quantum cryptographic techniques, dynamic hash function selection, and zero-knowledge proof verification, ensuring long-term security and efficiency in data integrity verification. Below are the key components of the QRMT framework.

4.1. Dynamic Hash Function Selection

QRMT leverages a cryptographic random number generator (CRNG) to dynamically select a post-quantum secure hash. This approach produces cryptographic security using multiple different cryptographic primitives. Post-quantum hash operation SHAKE-256 stands as a NIST-standardized hash function [4] and joins Blake3 as an optimized high-speed hashing solution [5] together with Poseidon, which focuses on zk-STARKs and cryptographic proof optimization [6]. Randomized hashing strengthens guesswork protection, which prevents pre-calculated quantum assault methods, including Grover's Algorithm-based collisions [7]. QRMT employs flexible cryptographic selection methods that make the system resilient if a specific hash function falls under compromise [8].

4.2. Post-Quantum Lattice-Based Encryption with Kyber1024

The QRMT encryption system employs Kyber1024 as its encryption scheme to protect metadata because this post-quantum lattice-based encryption technology offers resistance against Shor's Algorithm [2], [11]. Kyber1024 was selected among other candidates because its basis in the Learning with The LWE error problem makes it quantum resistant [19]. A reference framework from [29] guides system administrators about QRMT implementation in existing cryptographic systems. LWE decryption provides effectiveness at key generation and cryptanalysis stages compared to RSA modes while maintaining trustworthy post-quantum security standards, according to [9]. New Hope key exchange represents how lattice-based cryptographic approaches, including Kyber1024, secure blockchain metadata successfully, according to [23]. The encryption system of Kyber1024 within QRMT ensures that Merkle root metadata, along with selected hash function indices, maintains confidentiality against quantum attackers.

4.3. zk-STARKs for Trustless Proof Verification

The post-quantum transparency, along with scalability features of zk-STARKs, was formally established through the work of [24] to make QRMT's verification framework efficient. QRMT uses zero-knowledge Scalable Transparent Argument of Knowledge (zk-STARKs) to verify proofs through an efficient, scalable framework [3], [12]. Zk-STARKs offer several advantages, including the absence of a trusted setup, unlike zk-SNARKs, which require a pre-established key [10]. The verification system has logarithmic complexity, which allows proof generation and verification to scale effectively with a growing dataset size [12]. Users can conduct Merkle root verifications independently through zk-STARKs without revealing confidential details [3].

The research paper of [31] explains zero-knowledge proofs for blockchain systems and promotes zk-STARKs as quantum-resistant verification protocols that match QRMT verification requirements. The verification capability using zk-STARKs functions without requiring trust from either party. QRMT establishes itself as the best method for blockchain integrity verification and protected distributed systems verification.

4.4. Secure Metadata Encoding & Transmission

QRMT includes a sturdy encoding procedure that transmits metadata to various networks with both speed and security. Base64 encoding provides a representation format for metadata by converting encrypted metadata into text-based data, which protects transmission over networks [10]. All unauthorized alterations to the QRMT structure are detected right away through tamper-resistant cryptographic encoding [7]. Secure metadata encoding methods incorporated into QRMT ensure that both data remain untainted and secure from unauthorized access during the communication of cryptographic metadata.

4.5. Quantum-Resistant Security & Scalability

Fernandez-Carames and Fraga-Lamas deliver a full report about blockchain cryptography and its resistance to quantum attacks, where they show the need for quantum-resistant schemes like QRMT. QRMT acts as an essential element that enables blockchain connectivity with secure network protocols and distributed data systems. The implementation adopts a strategy that both resists quantum threats and allows the retention of current infrastructure components. The performance costs of cryptographic procedures in QRMT remain low because the system implements advanced optimizations for quick proof verification combined with fast hashing at optimal efficiency levels. The paper by [22] defines the quantum computer challenges to blockchain cryptographic systems, which demonstrates the immediate need to use QRMT for quantum-secure distributed ledger networks.

By leveraging dynamic hash function selection, Kyber1024 encryption, and zk-STARK verification, QRMT establishes a robust and forward-thinking framework for next-generation data integrity solutions.

5. Workflow

Before constructing the Merkle Tree, we collect all the data blocks that must be securely hashed and verified. Each data block represents information, such as transaction details, file hashes, or cryptographic data.

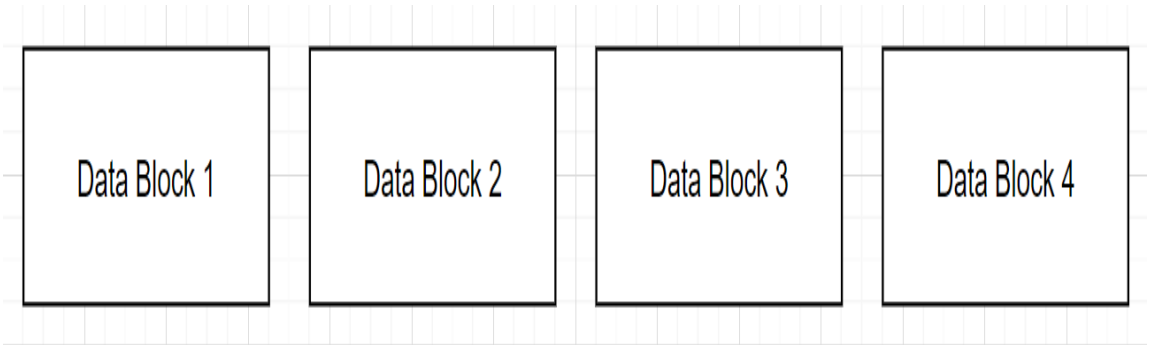

Figure 4. Initialization of data blocks

Each data block is hashed using a randomly selected post-quantum hash function. The available hash functions include.

- SHAKE-256 (NIST post-quantum standard)

- Blake3 (High-speed cryptographic hash)
- Poseidon (Optimized for zk-STARKs and blockchain verification)

A cryptographic random number generator (CRNG) selects one of these hash functions for each hashing operation, ensuring unpredictability and security.

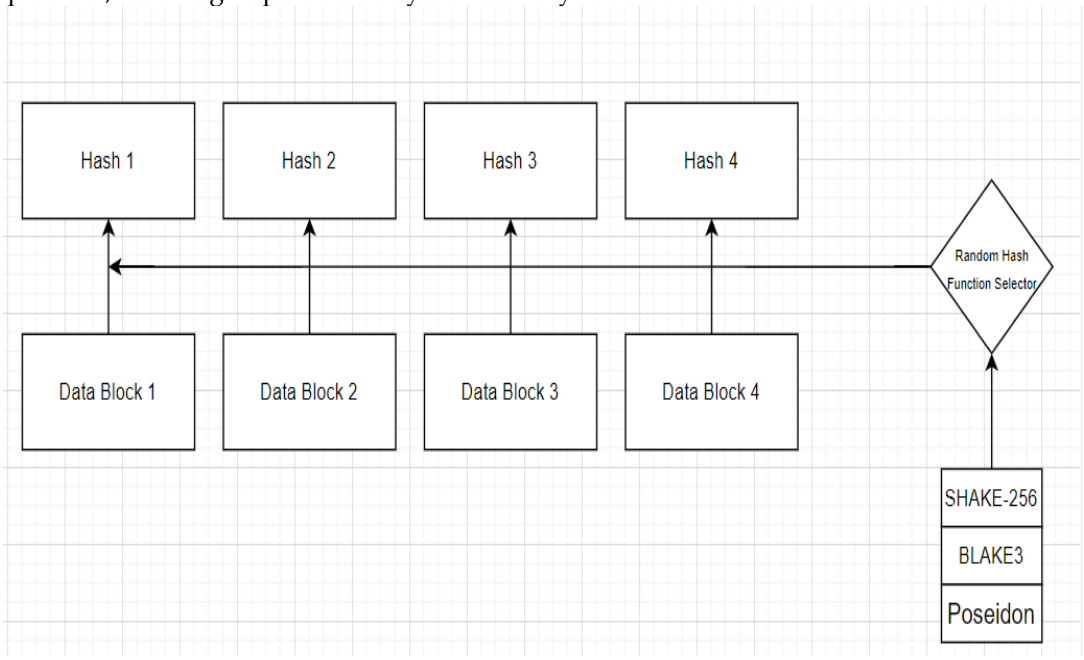

Figure 5. Cryptographic Hashing Using a Randomized Hash Function

Once each data block is hashed, the algorithm assigns an index to the hashed output and stores it in a new index array. This index keeps track of which hash function was used at each level, ensuring traceability and verification.

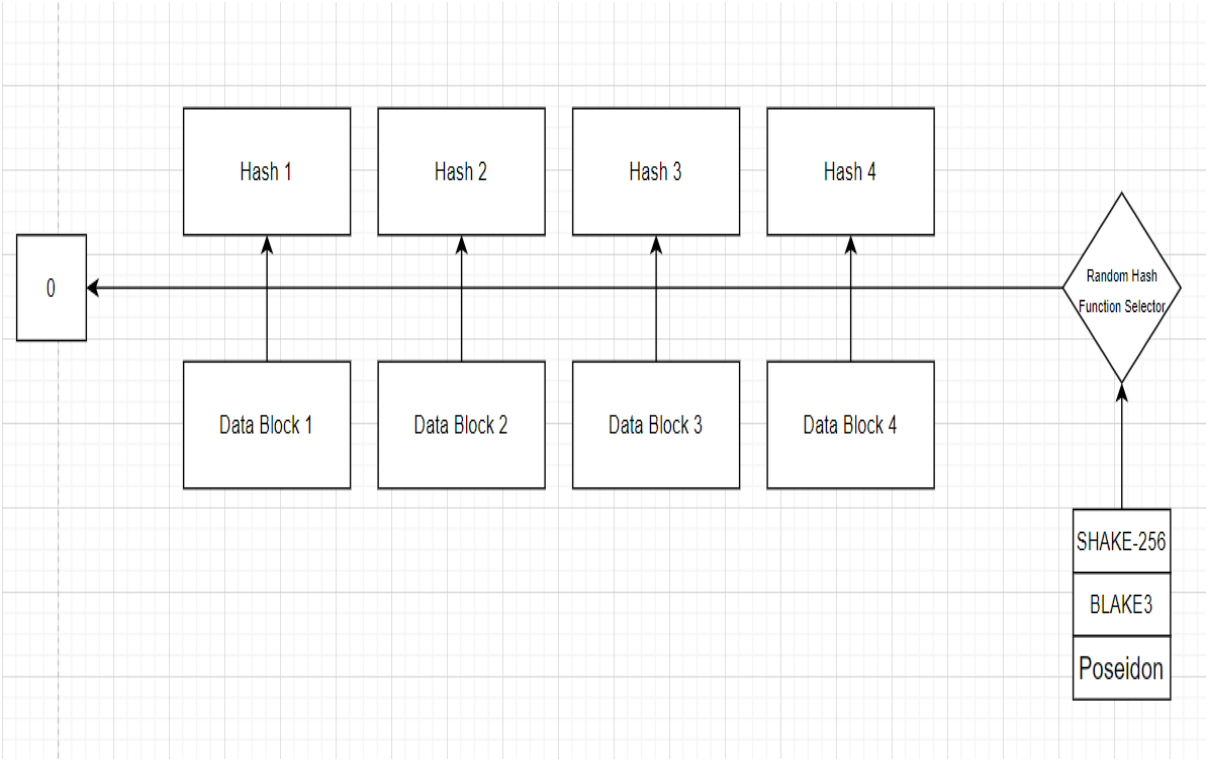

Figure 6. Storing Hash Values in an Array-Based Index Structure

The algorithm proceeds to the next level by pairing adjacent hashed values, hashing them together, and storing the new results at the next level of the tree.

Again, a random hash function is selected for each pairing. The process repeats at every level until only one final root value remains.

6. PSEUDO CODE

```
# Quantum-Resistant Merkle Tree (QRMT) Implementation
# Post-quantum cryptographic primitives:
# - SHAKE-256, Blake3, Poseidon for hashing
# - Kyber1024 for encryption
# - zk-STARKs for verification

# Global constants
HASH_FUNCTIONS = [SHAKE256, Blake3, Poseidon]  # Post-quantum hash
functions
SECURITY_PARAM = 256  # Security parameter in bits

def BuildQRMT(data_blocks):
    """
    Constructs a Quantum-Resistant Merkle Tree from input data blocks
    Args:
        data_blocks: List of raw data blocks to be included in the tree
    Returns:
        Tuple: (encrypted_root, proof) where:
            - encrypted_root: Kyber1024-encrypted Merkle root + indices
            - proof: zk-STARK proof for verification
    """

    # Phase 1: Leaf Node Construction
    leaf_nodes = []
    hash_indices = []  # Tracks hash functions used at each level

    for block in data_blocks:
        # Randomly select post-quantum hash function
        hash_idx, hash_fn = RandomSelectHashFunction(HASH_FUNCTIONS)

        # Store hash function index for verification
        hash_indices.append(hash_idx)

        # Hash data block with selected function
        hashed_data = hash_fn(block, output_size=SECURITY_PARAM)
        leaf_nodes.append(hashed_data)

    # Phase 2: Tree Construction
    current_level = leaf_nodes
    level_hashes = [hash_indices.copy()]  # Track hash indices per level

    while len(current_level) > 1:
        next_level = []
        current_level_indices = []

        for i in range(0, len(current_level), 2):
            left_node = current_level[i]
            right_node = current_level[i+1] if i+1 < len(current_level)
else left_node

            # Random hash selection for this parent node
            hash_idx, hash_fn = RandomSelectHashFunction(HASH_FUNCTIONS)
            current_level_indices.append(hash_idx)

            # Concatenate and hash child nodes
            combined = left_node + right_node  # Byte concatenation
            parent_hash = hash_fn(combined, output_size=SECURITY_PARAM)
            next_level.append(parent_hash)

        current_level = next_level
        level_hashes.append(current_level_indices)
```

```
    # Final Merkle root
        merkle_root = current_level[0]

        # Phase 3: Post-Quantum Protection
        # Encrypt root and hash indices with Kyber1024
        metadata = {
            'root': merkle_root,
            'hash_indices': level_hashes,
            'timestamp': GetCurrentTime()
        }
        encrypted_data = Kyber1024_Encrypt(
            plaintext=Serialize(metadata),
            public_key=QRMT_PUBLIC_KEY
        )

        # Generate zk-STARK proof
        proof = GenerateZkStarkProof(
            merkle_root=merkle_root,
            hash_indices=level_hashes,
            security_param=SECURITY_PARAM
        )

        return (encrypted_data, proof)

    def VerifyQRMT(encrypted_data, proof, data_block=None, position=None):
        """
        Verifies a QRMT proof with optional membership check
        Args:
            encrypted_data: Kyber1024-encrypted root + indices
            proof: zk-STARK proof
            data_block: (Optional) Specific data block to verify
            position: (Optional) Position of data block in tree
        Returns:
            bool: True if verification succeeds, False otherwise
        """

        # Decrypt metadata with Kyber1024
        try:
            metadata = Deserialize(Kyber1024_Decrypt(
                ciphertext=encrypted_data,
                private_key=QRMT_PRIVATE_KEY
            ))
        except DecryptionError:
            return False

        merkle_root = metadata['root']
        level_hashes = metadata['hash_indices']

        # Verify zk-STARK proof
        if not VerifyZkStarkProof(proof, merkle_root):
            return False

        # Optional: Verify specific data block inclusion
        if data_block is not None and position is not None:
            if not VerifyMembership(
                data_block,
                position,
                merkle_root,
                level_hashes,
                HASH_FUNCTIONS
```

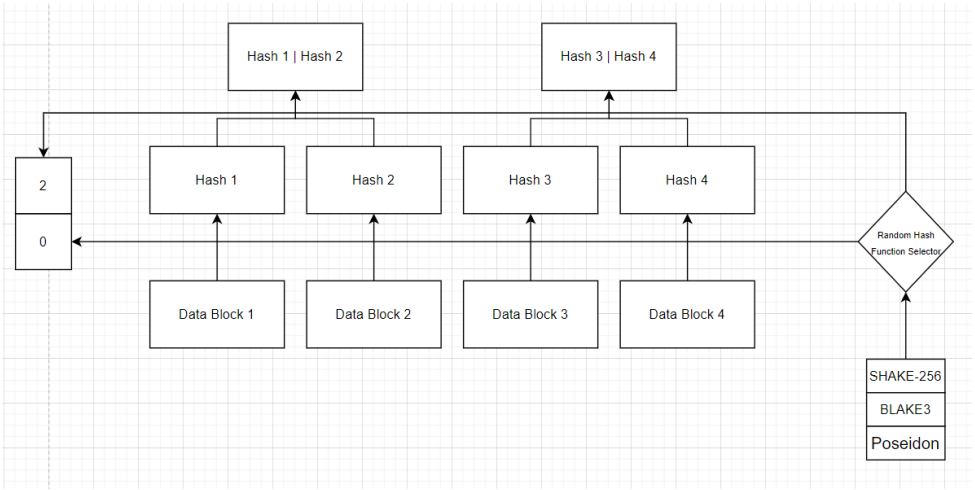

Figure 7. Advancing to a Higher Level and Iteratively Repeating the Hashing Process

This iterative process continues until a single final hash remains, known as the Merkle Root.

- The Merkle Root represents the original data blocks' cryptographic fingerprint.
- The index array (containing selected hash functions) is finalized at this stage.

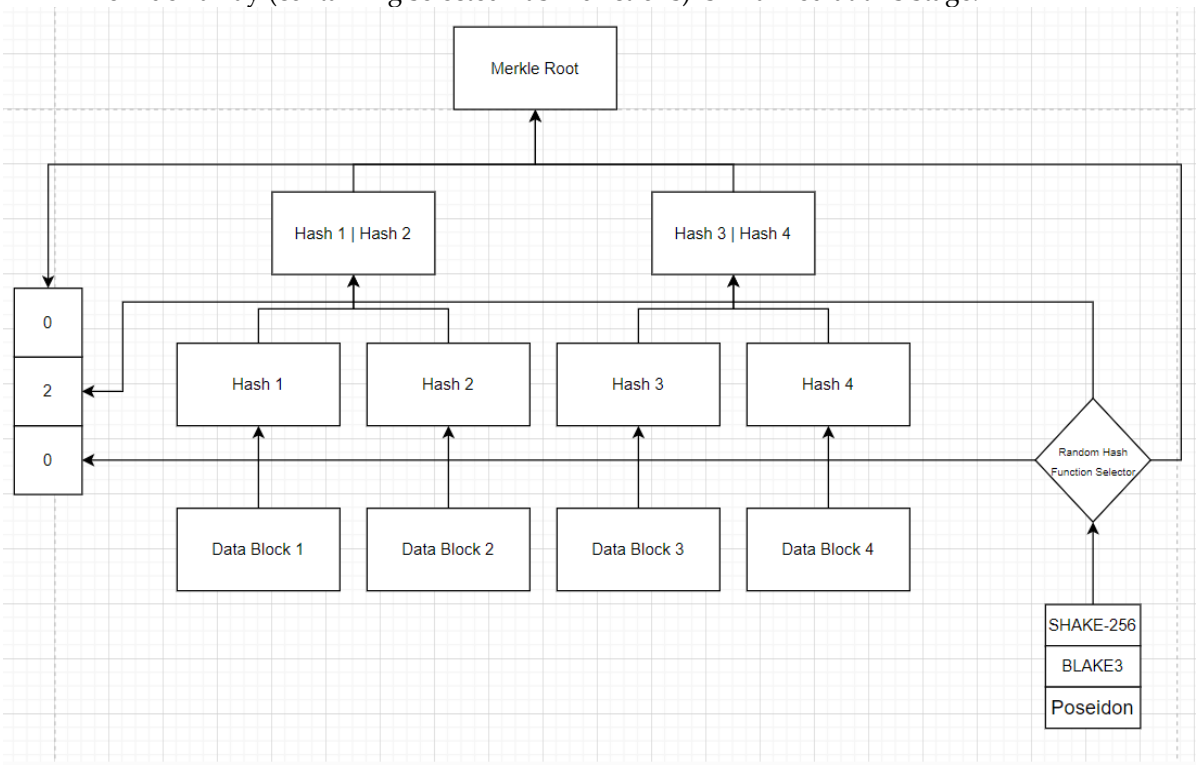

Figure 8. Iteratively Processing Until the Merkle Root is Computed

Once the Merkle Root is computed, we combine the following elements:

1. Merkle Root – The final cryptographic fingerprint of all data blocks.
2. Index Array – The recorded sequence of hash function selections at each level.
3. Proofs – The Merkle proof data is required for verification.

7. Formulae Derivation

A. Hash Function Selection (Dynamic Post-Quantum Hashing)

- $H = \{H_1, H_2, H_3\} = \{SHAKE-256, BLAKE3, Poseidon\}$ be the set of available post-quantum secure hash functions
- A cryptographic random number generator (CRNG) selects a hash function index $i \in \{1, 2, 3\}$
- The selected hash function is $H_i \in H$.

B. Hashing of Leaf Nodes

For each data block d_j, a randomly chosen hash function H_i is applied:

$$L_j = H_i(d_j)$$

Where:

- L_j is the hashed leaf node.
- d_j is the original data block.
- H_i is selected using CRNG.

C. Parent Node Hashing

For each pair of child nodes A and B, the parent node P is computed as:

$$P = H_i(A \,||\, B)$$

Where:

- $A \,||\, B$ denotes the concatenation of two child node hashes.
- $H_i \in H$ is the selected hash function for this level.

D. Merkle Root Computation

This recursive process continues until the final Merkle root R is obtained:

$$R = H_k(P_1 \,||\, P_2)$$

Where:

- $P_1, P_2,$ are the last remaining parent nodes
- H_k is the final randomly selected hash function.ds

E. Lattice-Based Encryption of Metadata (Kyber1024)

The Merkle root R and the hash function index array $\vec{I}$ are encrypted as follows:

$$C = Kyber1024\_Encrypt(R, \vec{I})$$

Where:

- R is the Merkle root.
- $\vec{I} = [i_1, i_2, \ldots, i_n]$ represents the sequence of selected hash function indices.
- C is the ciphertext output.

F. Zero-Knowledge Proof-Based Verification (zk-STARKs)

- To verify the integrity, decrypt C using the private key sk:

$$(R', \vec{I'}) = Kyber1024\_Decrypt(C, sk)$$

- Recompute the Merkle root R'' using $\vec{I'}$ and the original data blocks.
- If $R' = R''$, integrity is verified. Otherwise, integrity is compromised.
- A $zk - STARK$ proof π is generated:

$$\pi = STARK\_Prove(d_1, d_2, \ldots, d_n)$$

- And verified:

$$STARK\_Verify(\pi) = \text{True or False}$$

8. Comparison between Traditional Merkle Trees and Quantum-Resistant Merkle Trees (QRMT)

Table 2. Comparing Classical Merkle Trees and Quantum-Resistant Merkle Trees

Feature	Simple Merkle Tree (SMT)	Quantum-Resistant Merkle Tree (QRMT)
Hash Function	Uses a single fixed hash function (e.g., SHA-256)	Employs dynamic selection from multiple post-quantum hash functions (SHAKE256, Blake3, Poseidon)
Security Level	~128-bit classical security	~256-bit post-quantum security
Quantum Attack Resistance	Vulnerable to Grover's algorithm, reducing security from 2^n to 2^(n/2)	Resistant to quantum attacks through dynamic hash selection and post-quantum primitives
Hash Selection Method	Static – same hash function for all nodes	Dynamic – random selection at each level using C-RNG
Public Key Encryption	Uses RSA or ECC, vulnerable to Shor's algorithm	Uses Kyber1024 (lattice-based), resistant to quantum attacks
Proof Verification	Requires full node validation	Uses zk-STARKs for efficient, trustless verification

Metadata Protection	Basic encryption (RSA/ECC)	Post-quantum secure encryption with Kyber1024
Computational Overhead	Lower – single hash function	Higher – multiple hash functions and encryption
Storage Requirements	Lower – stores fewer hashes	Higher – stores hashes and hash function indices
Implementation Complexity	Simple – straightforward algorithm	Complex – requires multiple cryptographic primitives
Scalability	O(log n) proofs	O(log n) with enhanced verification through zk-STARKs
Blockchain Integration	Used in Bitcoin, Ethereum	Designed for post-quantum blockchain systems
Future Proofing	Vulnerable to quantum computing advances	Designed to resist future quantum threats
Key Recovery Attack Resistance	Vulnerable to quantum key recovery	Resistant through Kyber1024 encryption
Memory Usage	Lower – simpler data structure	Higher – additional metadata and proofs
Verification Speed	Faster – simpler verification	Slower – complex zero-knowledge proofs

9. Applications That Utilize Quantum-Resistant Merkle Trees

The advancement of Quantum-resistant Merkle Trees (QRMT) brings the protection of data integrity across different sectors through post-quantum cryptography as well as choice mechanisms for dynamic hash functions and zero-knowledge proof verification. The innovative system boosts the security of blockchain systems along with cloud storage, communication networks, digital identity management and smart contracts.

QRMT serves as the principal usage of protecting blockchain and cryptocurrency systems. The core data integrity structure of blockchain architectures in Ethereum networks uses traditional Merkle Trees. The data security of the systems faces major threats through potential hash collisions that affect these systems. This security vulnerability received research evaluation due to its existence within the system. The study showed the urgent need for strengthening protective measures [15].

The security solutions implemented by QRMT incorporate both BLAKE3 and Poseidon, which function as post-quantum hash capabilities. BLAKE3 stands ready for upcoming applications because it manages to produce high security with efficient processing at modern, optimized speeds [16]. Poseidon is specifically designed to be efficient within zero-knowledge-proof systems, enhancing the performance of zk-STARKs in blockchain ecosystems [17]. These hash functions are integral in securing block headers and transactions against potential quantum threats.

Zero-Knowledge Scalable Transparent Arguments of Knowledge (zk-STARKs) exist as part of QRMT to enable quick proof verification in Layer-2 rollups operating within blockchain environments. The integration strengthens both the scalability and security features of Ethereum as well as the Hyperledger Fabric infrastructure. The information security and performance of zk-STARKs benefit from specific Hash functions that work specifically with zero-knowledge protocols using Poseidon [17].

QRMT adopts lattice-based cryptographic methods, including Kyber1024, as replacements for conventional RSA and ECC cryptographic algorithms because they shield against quantum attack exploits, particularly Shor's Algorithm attacks. The Learning with Errors (LWE) problem makes Kyber1024 secure enough to serve as a preferable replacement for conventional encryption systems due to its resistance against quantum attacks [18].

The QRMT protocol presents itself as an excellent cryptographic solution that competes against traditional Merkle Tree verification protocols in cloud storage and data integrity systems. The frequent use of hash-based verification in cloud storage demands appropriate recognition that traditional cryptographic hash capabilities are prone to quantum brute-force attacks. Between post-quantum hashing mechanisms

SHAKE-256 and Poseidon, which QRMT implements for cloud data verification, the security of verification practices increases substantially against quantum computer assault methods while improving system robustness. Data metadata receives improved safety through the addition of Kyber1024 lattice-based encryption, which protects stored data from quantum decryption threats. The implementation of zk-STARKs enables cloud storage providers to run their operations trustless because they cannot modify or fabricate stored data without detection. These updated security improvements enable QRMT to connect directly with Google Drive and AWS S3 and IPFS platforms to ensure protection against quantum-based threats emerging during the next quantum era.

The technology of Quantum Resistant Matrix Transformations enables developers to create safety measures both for IoT authentication and for secure messaging through post-quantum era network protocols. Secure communication protocols of today use digitized signatures and HMAC mechanisms to protect their message [2]. The cryptographic techniques remain exposed to quantum decryption attacks, which threaten the security of transmission methods. QRMT solves these problems through the use of Kyber1024 encryption for future-proof secure communication. End-to-end encryption services in Signal, WhatsApp, and Telegram apps are provided through this solution [9]. Through zk-STARKs, users obtain authentication of their identity without needing to disclose sensitive metadata structures and can still demonstrate their verified identity to others [12]. The cryptographic solution QRMT functions optimally for real-time secure messaging, and it supports both 5G networks and IoT security purposes.

The QRMT authentication system supports original developments in digital identity security through decentralized methods for security verification. The current identity verification methods based on centralized authorities and RSA/ECC digital certificates face the risk of decryption failure due to quantum decryption attacks, according to [11]. Heightened digital identity security is guaranteed by QRMT which implements lattice-based signature protocols to stay resilient towards quantum threats [9]. The implementation of zk-STARKs enables users to verify their identity privately because these systems support confidential self-authentication protocols [3]. The combination of properties in QRMT delivers an optimal solution for applications that include self-sovereign identity (SSI) systems, digital passports, and decentralized identity verification solutions.

The QRMT system provides quantum-resistant brilliant contract execution to fix blockchain-based DeFi risks as well as implementing automated smart contract verification. The present-day smart contracts run execution security through hash-conditionals alongside digital signatures, but these methods remain exposed to quantum decryption attacks [1]. The security of smart contracts gets improved through QRMT by integrating quantum-resistant lattice-based authentication schemes Kyber1024 and Di lithium signatures instead of existing ECDSA authentication methods [9]. Zk-STARKs provide efficient off-chain computation verification, which helps Ethereum, along with Polka Dot and Cardano, to lower gas fees while enabling higher transaction speeds for their smart contracts [3]. The combined system of advance provides quantum security and tamperproof protection along with high scalability to modern contract platforms.

The essential post-quantum cryptography advancement QRMT provides churches stronger security while validating cloud storage authentication and enabling secure communications in addition to identity checks and brilliant contract operations. QRMT implements a cryptographic framework through post-quantum secure hashing lattice-based encryption and zero-knowledge proofs, which creates a resilient future-proof system that achieves enhanced security by addressing traditional Merkle Trees' flaws along with quantum computing risk preparation.

10. Conclusion

The Quantum-Resistant Merkle Tree (QRMT) represents a significant advancement in cryptographic data integrity verification, addressing the urgent need for quantum-resistant structures in blockchain systems, distributed storage, and secure communications. By integrating post-quantum cryptographic hashing (SHAKE-256, Blake3, Poseidon), lattice-based encryption (Kyber1024), and zk-STARKs verification, QRMT establishes a robust framework that maintains the efficiency of classical Merkle Trees while providing defence against both Grover's and Shor's quantum algorithms. This combination of dynamic hash selection, quantum-resistant encryption, and zero-knowledge verification offers comprehensive protection for modern digital infrastructure as quantum computing capabilities advance.

However, several practical challenges must be acknowledged: the computational overhead of lattice-based operations may impact performance in resource-constrained environments, the storage requirements for dynamic hash indices could affect scalability in massive datasets, and the current trust model for cryptographic random number generation presents a potential vulnerability point. Looking ahead, critical research directions include developing optimized implementations for edge devices and IoT ecosystems, pursuing formal standardization through organizations like NIST, enhancing the robustness of random number generation through quantum entropy sources, and exploring hybrid architectures that bridge classical and post-quantum approaches during transition periods. As quantum computing continues to evolve, QRMT provides a foundational framework for maintaining data integrity in the post-quantum era, though its widespread adoption will depend on addressing these practical considerations through ongoing research and collaborative standardization efforts within the cryptographic community.

References

1. Merkle, R.C. Secrecy, Authentication, and Public Key Systems. Stanford University, Information Systems Laboratory 1979.
2. Bos, J.W.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Stehlé, D.; Vaudenay, S. CRYSTALS – Kyber: A CCA-secure module-lattice-based KEM. Advances in Cryptology – EUROCRYPT 2018, pp. 1–30.
3. Ben-Sasson, E.; Chiesa, A.; Riabzev, M.; Spooner, N.; Virza, M.; Ward, N.P. Scalable Zero-Knowledge via Cycles of Elliptic Curves. Advances in Cryptology – EUROCRYPT 2018, pp. 1–35.
4. National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)," *FIPS PUB 202*, U.S. Department of Commerce, 2015.
5. Aumasson, J.-P.; Neves, S.; Wilcox-O'Hearn, Z.; Winnerlein, C. BLAKE2: Simpler, Smaller, Fast as MD5. Applied Cryptography and Network Security (ACNS) 2013, pp. 1–15.
6. Grassi, L.; Khovratovich, D.; Rechberger, C.; Schofnegger, M. On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy. Cryptology ePrint Archive, Paper 2019/372 2019.
7. Albrecht, M.R.; Grassi, L.; Kiefer, F.; Khovratovich, D. Randomized Hashing and Quantum-Secure Hash Trees. International Conference on Cryptology 2021, 20, pp. 125–140.
8. Hoffstein, J.; Pipher, J.; Silverman, J.H. An Introduction to Mathematical Cryptography, 2nd ed.; Springer: New York, USA, 2014.
9. Bernstein, D.J.; Lange, T.; Peters, C. Attacks on Lattice-Based Cryptography. Advances in Cryptology – EUROCRYPT 2008, pp. 1–15.
10. Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schwabe, P.; Stehlé, D.; Seiler, G. A Post-Quantum Hash-Based Signature Scheme. Journal of Cryptographic Research 2020, 25, pp. 45–66.
11. Ji, X.; Dong, J.; Zhang, P.; Deng, T.; Hua, J.; Xiao, F. HI-Kyber: A Novel High-Performance Implementation Scheme of Kyber Based on GPU. Cryptology ePrint Archive, Paper 2023/1194 2023.
12. Bhattacharya, S.; Seth, D.; Panyam, S.; Gangrade, P. Enhancing Digital Privacy: The Application of Zero-Knowledge Proofs in Authentication Systems. International Journal of Computer Trends and Technology (IJCTT) 2024, 72(4), pp. 34–41.
13. Vijay, S.; Priya, S.S.; Harshavardhana, C.N.; Kemparaju, R. Quantum-resistant cryptographic algorithms for secure communication. ICTACT Journal on Communication Technology 2024.
14. Boneh, D.; Lewi, K.; Montgomery, H.; Raghunathan, A. Key homomorphic PRFs and their applications. 2015. Available online: https://core.ac.uk/download/579870012.pdf
15. Park, J.; Kim, K.; Lee, S. Security vulnerabilities of traditional Merkle trees in blockchain applications. Cryptology ePrint Archive, Paper 2024/367 2024.
16. BLAKE3 Development Team, "BLAKE3 cryptographic hash function," 2024.
17. Khovratovich, D. Poseidon hash function: Design and applications in zk-STARKs. In Proceedings of the Real World Cryptography Conference; New York, USA, 2020.
18. Ding, J.; Bailey, D.V.; Melchor, C.A. A comprehensive analysis of Kyber1024 for post-quantum cryptography. Journal of Cryptographic Research 2023, 28, pp. 12–34.
19. Luo, F.; Al-Kuwari, S.; Wang, H.; Wang, F.; Chen, K. Revocable attribute-based encryption from standard lattices. Computer Standards & Interfaces 2023.
20. Gentry, C. A fully homomorphic encryption scheme. In Proceedings of the ACM Symposium on Theory of Computing; San Jose, USA, 2009.
21. Peikert, C. A decade of lattice cryptography. Foundations and Trends in Theoretical Computer Science 2016.
22. Aggarwal, D.; Brennen, G.K.; Lee, T.; Santha, M.; Tomamichel, M. Quantum attacks on Bitcoin and how to protect against them. Ledger Journal 2017.
23. Alkim, E.; Ducas, L.; Pöppelmann, T.; Schwabe, P. Post-quantum key exchange: NewHope. In Proceedings of the USENIX Security Symposium; Austin, USA, 2016.
24. Ben-Sasson, E.; Chiesa, A.; Tromer, E.; Virza, M. Scalable transparent arguments of knowledge (STARKs). Journal of Cryptology 2018.
25. Kinyua, C.G. The impact of quantum computing on cryptographic systems: Urgency of quantum-resistant algorithms and practical applications in cryptography. European Journal of Information Technology and Computer Science 2025, 5(1).

26. Fernandez-Carames, T.M.; Fraga-Lamas, P. Towards post-quantum blockchain: A review on blockchain cryptography resistant to quantum computing attacks. Frontiers in Physics 2024, 12, p. 1456491.
27. Radanliev, P. Artificial intelligence and quantum cryptography. Journal of Analytical Science and Technology 2024, 15(1), p. 4.
28. Alvarado, M.; Gayler, L.; Seals, A.; Wang, T.; Hou, T. A survey on post-quantum cryptography: State-of-the-art and challenges. arXiv preprint, arXiv:2312.10430 2023.
29. Hasan, K.F.; Simpson, L.; Baee, M.A.R.; Islam, C.; Rahman, Z.; Armstrong, W.; Gauravaram, P.; McKague, M. A framework for migrating to post-quantum cryptography: Security dependency analysis and case studies. arXiv preprint, arXiv:2307.06520 2023.
30. Sun, Y.; Liu, J.; Liang, H. Post-quantum digital signature schemes and their applications in blockchain. IEEE Transactions on Information Forensics and Security 2023, 18, pp. 1–14.
31. Wang, Y.; Qin, H.; Yang, Y. Zero-knowledge proofs in blockchain: A comprehensive review. Journal of Cryptographic Engineering 2023, 13(2), pp. 147–169.
32. Bhaskar, K. *Understanding Zero-Knowledge Proofs Part 1: Verifiable Computation with zk-SNARKs*. Available online: https://medium.com/@bhaskark2/understanding-zero-knowledge-proofs-part-1-verifiable-computation-with-zk-snarks-ba6cbb8e6001.