Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.077994

ARTICLE

Hash-Based Signature Authentication for Scalable and Security-Consistent QKD Post-Processing

Chaokun Wang[1], **Sijiang Xie**[1,*], **Yalong Yan**[2] **and Hong Zhao**[2]

[1]Department of Cyberspace Security, Beijing Electronic Science and Technology Institute, Beijing, China
[2]Institute of Information Security, Beijing Electronic Science and Technology Institute, Beijing, China

*Corresponding Author: Sijiang Xie. Email: xiesj@besti.edu.cn

Received: 21 December 2025; Accepted: 29 January 2026; Published: 08 May 2026

ABSTRACT: Quantum Key Distribution (QKD) ensures secure key establishment through the principles of quantum mechanics; however, its effectiveness in practice hinges on dependable identity verification via classical channels during the post-processing phase. Current QKD implementations typically depend on pre-existing symmetric-key authentication, which suffers from limited scalability and complicated key management in extensive networks. Authentication methods utilizing post-quantum cryptography (PQC) signatures, based on complex mathematical assumptions, introduce extra and uncertain security dependencies, potentially compromising the security model integrity that QKD aims to maintain. This paper explores the application of hash-based signatures (HBS) for identity verification in the post-processing of QKD. HBS methods derive their security from cryptographic hash functions, which are integral to QKD protocols, allowing for scalable public-key-style authentication without the need for new computational assumptions. A detailed authentication framework is proposed, incorporating HBS-based verification into all essential phases of QKD post-processing, such as mutual certificate validation, basis sifting, parameter estimation, error correction verification, and privacy amplification. Security assessments indicate that the suggested framework maintains the security model integrity of QKD by relying cryptographically solely on the collision resistance of hash functions—without introducing new computational assumptions. At the system deployment level, it adheres to standard PKI trust assumptions which are necessary for public-key-style authentication and consistent with practical QKD network operations. Additionally, system-level evaluations affirm the scalability and practical applicability of HBS-based authentication, while also addressing the operational trade-offs among various HBS approaches in realistic QKD deployment contexts.

KEYWORDS: Quantum key distribution; identity authentication; hash-based signatures; post-processing security; security-model consistency; post-quantum cryptography

1 Introduction

In light of the challenges posed by quantum computing, numerous traditional cryptographic methods have shown weaknesses, prompting the need for the creation of cryptographic strategies that can withstand quantum threats [1]. While traditional symmetric algorithms, such as hash functions, can enhance their quantum resilience by increasing security parameters—usually by ex-tending key lengths or output sizes—asymmetric algorithms like RSA and SM2 are inherently at risk and require replacement [2]. Presently, there are two viable quantum-resistant solutions: Post-Quantum Cryptography (PQC) and Quantum Key Distribution (QKD) [3]. PQC is based on mathematical challenges or hash functions that are believed to be secure against known quantum threats, although their long-term effectiveness is still in question [4]. On the other hand, QKD, which is rooted in the principles of quantum physics, provides theoretically

unconditional security [5]; however, due to practical engineering limitations, it must be combined with traditional cryptographic techniques to ensure effective quantum security. Overall, QKD is generally viewed as more expensive to implement compared to PQC [6].

QKD allows two entities to securely create a shared secret key. This process necessitates two types of channels: a quantum channel for the transmission of quantum states and a classical channel for subsequent processing [5]. The QKD procedure unfolds in two stages: the first involves the transmission of qubits, which are then measured, while the second entails the exchange of classical messages for tasks such as basis sifting, parameter estimation, error correction, and privacy amplification, culminating in the creation of a secure symmetric key [7]. To safeguard against man-in-the-middle attacks, it is essential to authenticate all classical communications [8]. Many current QKD systems rely on authentication methods that utilize pre-shared symmetric keys, resulting in considerable key management challenges and limited scalability in extensive networks [9]. Recent research has suggested the use of PQC for authentication to mitigate these problems, though it remains uncertain which PQC algorithm is the most effective [5].

This study posits that hash-based signature (HBS) methods are the sole appropriate choice for authenticating QKD. HBS is fundamentally dependent on the robustness of cryptographic hash functions, a concept that is both established and widely accepted [10]. Recent developments in post-quantum standardization strengthen the practical case for HBS. In August 2024, NIST finalized several PQC FIPS standards and published the Stateless Hash-Based Digital Signature Standard (FIPS-205) [10], which is based on SLH-DSA. This official recognition highlights HBS as a standardized option for quantum-resistant digital signatures and motivates its consideration for QKD post-processing authentication. Utilizing HBS allows QKD frameworks to sidestep the creation of additional mathematical security dependencies [10], while also addressing the key management issues linked to symmetric pre-shared keys.

The objectives of this study are as follows:

- This study aims to evaluate the shortcomings of current QKD authentication techniques, such as those relying on pre-shared symmetric keys and mathematical problem-oriented post-quantum cryptography methods, while also emphasizing the benefits of HBS during the post-processing phase of QKD.
- To create a practical framework for implementing HBS-based authentication in the post-processing of QKD data, which includes basis sifting, estimating parameters, verifying error correction, and enhancing privacy.
- To assess the overall effects and appropriateness of various HBS strategies when incorporated into QKD post-processing, taking into account factors such as scalability, communication costs, and practical limitations.

2 Comparative Analysis of Existing Authentication Schemes for QKD

2.1 Introduction to QKD Post-Processing and the Authentication Requirement

QKD enables two remote parties, conventionally named Alice and Bob, to establish a shared secret key [11]. Its implementation relies on two channels: a quantum channel for transmitting quantum states and a classical channel for post-processing [12]. The QKD process consists of two phases: the quantum transmission phase and the post-processing phase. The latter, conducted over the classical channel, is essential for distilling a secure, shared symmetric key from the raw quantum transmission data and involves the following sequential stages:

Basis Sifting: In the key selection phase of QKD, after receiving the qubits, Alice and Bob need to disclose the measurement bases they use. This process is carried out through a classical channel, where both parties compare their measurement results and select the bits measured under the same measurement base for

subsequent processing [13]. For example, if Alice uses the X basis for measurement and Bob uses the same X basis, they will retain these matching bits for use in subsequent steps.

Parameter Estimation: Alice and Bob sacrifice a random sample of their sifted keys to estimate the Quantum Bit Error Rate (QBER), which indicates potential eavesdropping.

Error Correction: Alice and Bob utilize an error correction protocol to rectify erroneous bits arising from noise or other disturbances [14]. Both parties will transmit error correction information via a classical channel, employing algorithms such as concatenated codes or Low-Density Parity-Check (LDPC) codes, which can rectify discrepancies in the remaining filtered keys to achieve an identical string.

Privacy amplification: They apply a universal hash function to the error-corrected key and shorten it to eliminate any partial information that a potential eavesdropper (Eve) may obtain, thereby ensuring the security of the final key [15]. By generating a shorter output as the final key, the design of the hash function needs to have good collision resistance to ensure that even if some information is eavesdropped, the attacker cannot deduce the final key.

All messages exchanged during these stages must be authenticated to prevent man-in-the-middle (MitM) attacks. If Eve can impersonate either party during post-processing, she can compromise the final key without detection. Therefore, the choice of authentication scheme is paramount to the overall security of the QKD system [16].

2.2 Authentication Based on Pre-Shared Symmetric Keys

As illustrated in Fig. 1, once the participants in QKD communication exchange quantum bit strings through the quantum channel, they must subsequently process this data via the classical channel. Taking the QKD receiver as the message sender in this context, the process begins with the sender generating a hash digest of the transmitted message [5]. This digest is then encrypted with a pre-shared symmetric key before being sent to the other participant. Upon receiving the ciphertext, the receiver (acting as the QKD sender) decrypts it using the same symmetric key, computes a hash digest for the original message with the identical hash function, and checks it against the decrypted data [9]. A match between the two digests confirms that both parties have successfully verified each other's identities, as the correct encryption and decryption rely on the shared symmetric key [5]. Conversely, if the digests do not match, it signifies a failure in authentication.

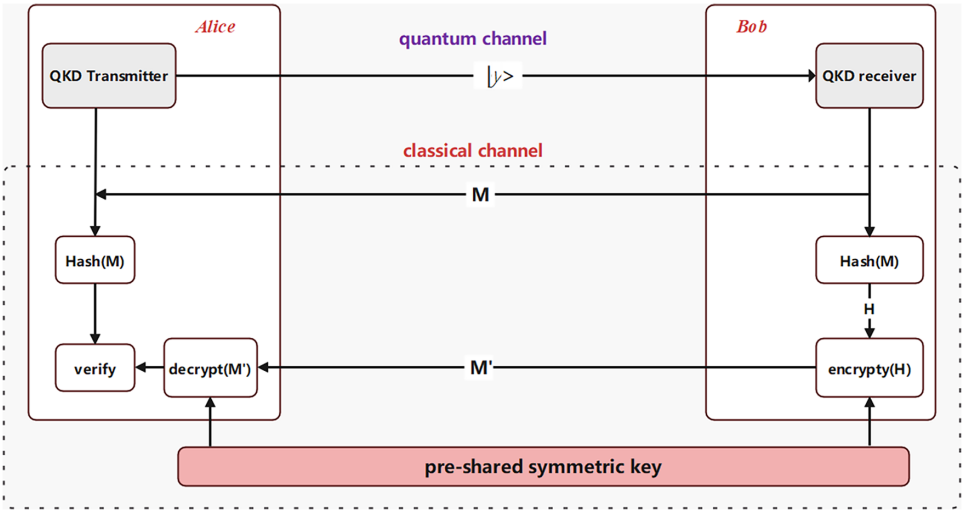

Figure 1: QKD authentication based on a pre-shared symmetric key.

Pre-shared symmetric keys are extensively utilized for identity verification because of their efficient computation and straightforward authentication process [9]. Generally, these keys are embedded in the QKD device by the manufacturer prior to shipping [16]. Once the devices are in the hands of the communicating parties, they utilize these embedded keys for authentication purposes. It is essential to maintain the physical security of the device throughout this process to avoid any potential key exposure.

However, when the number of QKD network users is large, this method is not easy to operate [5]. If the number of users is n, then the number of pre-shared key pairs is m:

$$m = C_n^2 = \frac{n(n-1)}{2} \tag{1}$$

This results in considerable logistical challenges. Additionally, onboarding a new user necessitates the secure creation of keys with all current users, and these keys need to be updated regularly, adding to the operational complexity.

Fig. 2 depicts the creation and utilization of QKD keys, which includes the quantum transmission stage, the post-processing stage, and the application stage [7]. In QKD systems that utilize pre-existing symmetric keys, the security during the quantum transmission stage is grounded in the fundamental characteristics of quantum mechanics; the integrity of the post-processing stage is determined by the attributes of hash functions; and the safety of the application stage is contingent upon the robustness of symmetric cryptography [9].

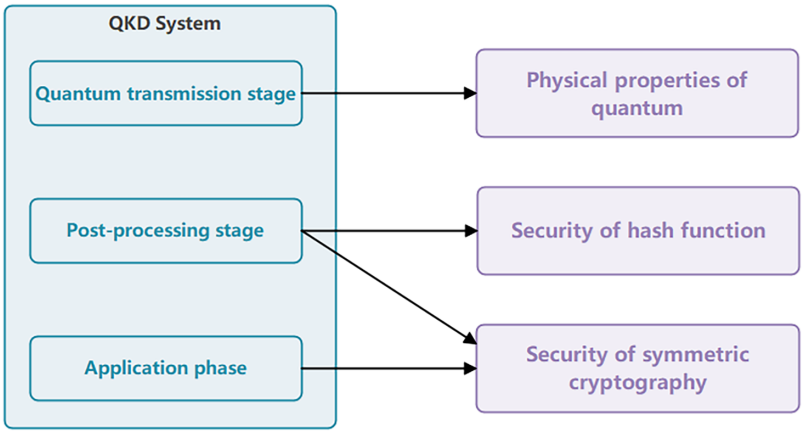

Figure 2: Cryptographic security dependencies of QKD with pre-shared symmetric key authentication.

2.3 Authentication Based on Mathematical-Problem PQC Signatures

To address the key management issue, digital signatures based on PQC have been suggested [5]. These methods, including those based on lattices (like Dilithium) or codes, operate under a public-key framework. Illustrated in Fig. 3, every participant possesses a validated pair of public and private keys. The sender uses their private key to sign a message, while the recipient checks the signature using the sender's public key [9].

While this solves the scalability problem, it introduces a critical new concern: an additional security dependency. As illustrated in Fig. 4, the security of the entire system now depends on both the information-theoretic security of the quantum channel and the computational complexity of a particular mathematical problem [5]. This situation creates a dilemma: if the PQC algorithm is considered entirely resistant to quantum attacks, it could be utilized independently for key exchange, rendering QKD unnecessary. Conversely, if QKD is employed due to the uncertainty surrounding the long-term security of PQC, then

incorporating the same PQC for authentication introduces that uncertainty into the QKD framework, thus compromising its security by introducing a possible vulnerability. This method effectively diminishes the unconditional security aimed for by QKD, transforming it into a conditional security that hinges on the computational challenges posed by the PQC algorithm. Consequently, employing such PQC algorithms for identity verification in the post-processing of QKD data not only complicates security dependencies but also creates a potential weak point, ultimately jeopardizing the overall security integrity of QKD.

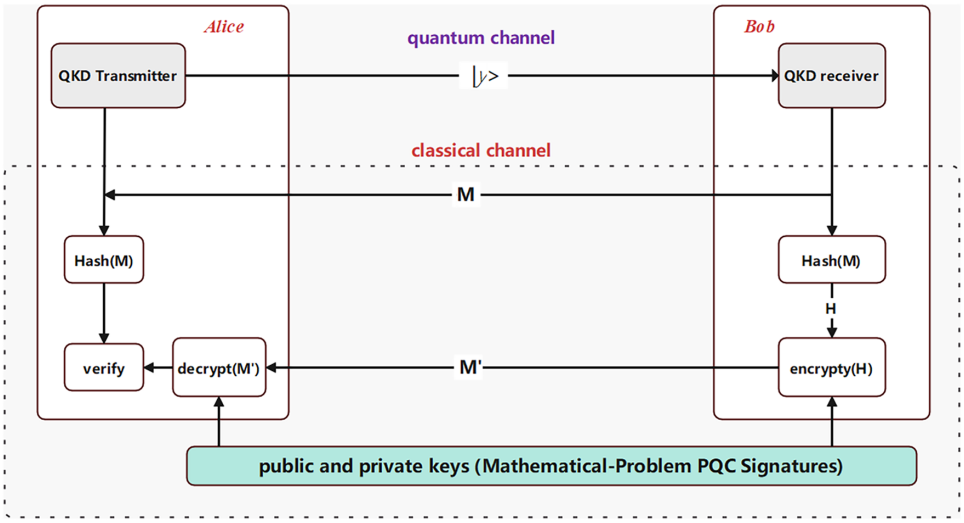

Figure 3: QKD authentication based on mathematical-problem PQC signatures.

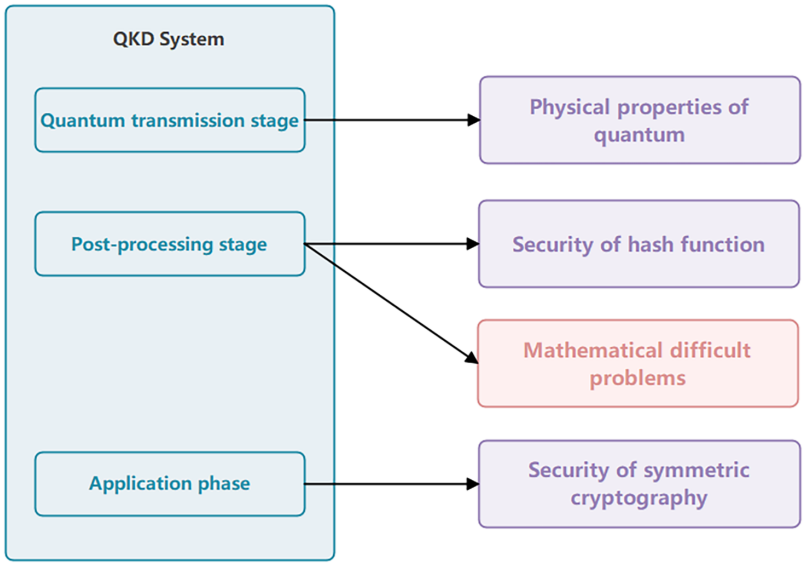

Figure 4: Cryptographic security dependencies of QKD with mathematical-problem PQC authentication.

Based on the above analysis, we can draw a clear conclusion: using the PQC algorithm based on mathematical problems (such as lattice, encoding, multivariable, etc.) for identity authentication in QKD systems, although it achieves scalable key management in operation, the hybridization of its security model constitutes a fundamental flaw. This scheme not only fails to strengthen QKD, but also weakens the

information theory security foundation pursued by QKD by introducing an independent and potentially fragile computational security assumption.

2.4 Authentication Based on Hash-Based Signatures

HBS, such as XMSS or SLH-DSA, derive their security exclusively from the cryptographic properties of the underlying hash function (e.g., collision resistance) [10]. The operational model is similar to other PQC signatures (Fig. 5), but the security foundation is fundamentally different and perfectly congruent with QKD.

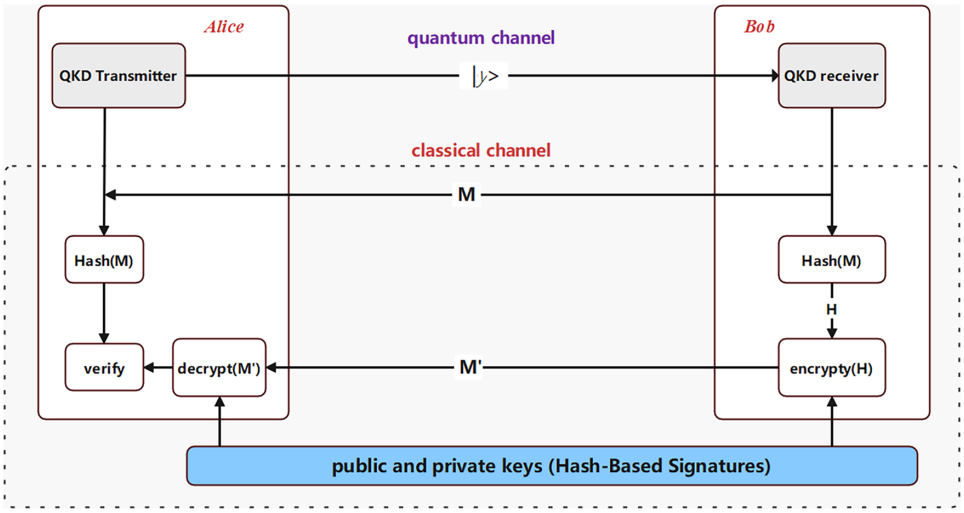

Figure 5: QKD authentication based on HBS.

The core advantage of HBS for QKD is that it introduces no new security assumptions [10]. The hash function is already a trusted component in the QKD protocol's privacy amplification stage [17]. Therefore, as shown in Fig. 6, using HBS for authentication does not create an additional security dependency layer. It simultaneously solves the key management complexity of symmetric schemes and avoids the security-model contamination of mathematical-problem-based PQC, achieving an optimal balance of scalability and aligned, long-term quantum resistance.

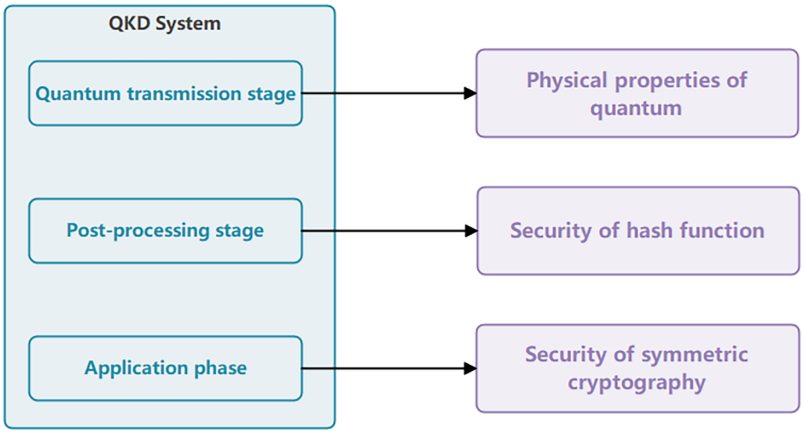

Figure 6: Cryptographic security dependencies of QKD with HBS authentication.

3 Design of an HBS-Based Authentication Framework for QKD

3.1 System Model, Assumptions, and Design Objectives

Following the comparative study outlined in Section 2, which identified HBS as the most suitable authentication method for QKD, this chapter introduces a comprehensive and cohesive implementation framework. The main goal is to create a thorough, sequential protocol derived from the BB84 protocol [18] that incorporates HBS-based authentication at each phase of the QKD post-processing, ensuring scalable and quantum-resistant authentication while avoiding the addition of new security vulnerabilities.

3.1.1 System Model and Entities

The framework involves the following entities operating within a standard QKD network model:

- **Participants in Communication:** Alice, who transmits the information, and Bob, the one receiving it, both seek to create a common secret key.
- **Trusted Certificate Authority, CA**: A trusted third party responsible for verifying the identities of Alice and Bob and issuing digital certificates. The CA itself uses an HBS scheme for signing certificates.
- **Quantum Channel:** This is utilized for sending quantum states, such as photons, in line with the BB84 protocol. While it is presumed to experience loss and interference, the principles of quantum mechanics safeguard it from undetected interception.
- **Classical Channel:** This is utilized for all communication related to post-processing. It is presumed that Eve has complete control over it [17], necessitating that any information shared through this channel be authenticated.

3.1.2 Core Security and Operational Assumptions

The design rests on the following foundational assumptions:

1. **Cryptographic Security Consistency:** The security dependencies of the authentication process are consistent with those of the native BB84 protocol [17]. No additional **Cryptographic** security assumptions (such as the hardness of mathematical problems) are introduced, ensuring that the theoretical unconditional security of QKD is not compromised.
2. **System-Level Trust Assumptions:** These are standard operational assumptions for PKI-based authentication in practical networks (not cryptographic dependencies), including that the Certificate Authority is honest and correctly verifies the real-world identities of participants (Alice/Bob) before issuing HBS-signed certificates, the channel for submitting CSR and receiving CA-issued certificates is physically secure or pre-authenticated to prevent CSR tampering or fake certificate injection, and the identity information provided by participants to the CA is genuine and uniquely identifies the communicating entity.
3. **End-to-End Authentication:** Every critical message exchanged during the classical post-processing phase must be authenticated to prevent MitM and replay attacks [9].
4. **Explicit Operability:** Provide a clear, sequential specification of actions for Alice and Bob at each stage, including what to sign, what to transmit, and what to verify.

3.1.3 Symbol and Term Definition

Key symbols and terms used in this chapter are defined in Table 1:

Table 1: Definition of key symbols and terms.

Symbol	Definition
U	Generic notation for communicating parties (Alice or Bob)
PK_U	HBS public key of party U
SK_U	HBS private key of party U
$Cert_U$	Digital certificate of party U, issued by CA with HBS signature
PK_{CA}	Public key of CA
SK_{CA}	Private key of CA
R_A/R_B	Session nonce generated by Alice/Bob
n_A/n_B	Capability list of Alice/Bob (including supported algorithms and parameters)
L_A/L_B	Measurement basis sequence of Bob (for quantum state measurement)
S_A/S_B	Sifted key of Alice/Bob
Q_A/Q_B	Sample index set of Alice/Bob
V_A/V_B	Sample bit string of Alice/Bob
X^N/Y^N	Raw key of Alice/Bob (length N)
K_{rec}	Reconciled key
H	Cryptographic hash function
H_{LDPC}	Parity-check matrix of LDPC code
$I_{\max}$	Maximum number of iterations for LDPC decoding
K_{final}	Final secure key

3.2 System Initialization and HBS-Based Certificate Issuance

Prior to initiating any QKD session, it is essential for all in-volved parties to create a reliable identity within the network using a Public Key Infrastructure (PKI) that relies solely on HBS. This initial setup phase is vital for establishing trust and involves three main steps: generating an HBS key pair, registering with a trusted Certificate Authority (CA), and obtaining a digital certificate signed with HBS.

3.2.1 Key Pair Generation

Every participant, such as Alice and Bob, is required to create their individual long-term cryptographic identities by employing a standardized HBS algorithm. In this framework, we designate a stateless HBS method (SLH-DSA-SHAKE256-256s, chosen by NIST for standardization) [10] that is appropriate for long-term key distribution and eliminates the management complexities associated with stateful methods (stateless indicates that there is no requirement to keep track of signature counters or the states of tree nodes).

The process for an entity U (where U represents Alice or Bob) is as follows:

1. **Algorithm and parameter selection:** Entity U selects the SLH-DSA-SHAKE256-256s variant, with security parameters aligned to NIST FIPS 205 standards (hash function: SHAKE256; signature length: 256 bytes; security level: 256 bits).
2. **Key generation:** U runs the key generation algorithm $HBS.KeyGen()$. This algorithm internally relies on a cryptographic hash function and a pseudorandom function:

$$HBS.KeyGen(U) = (PK_U, SK_U) \tag{2}$$

3.2.2 Registration and Identity Verification with the CA

With the HBS key pair generated, entity U must now bind its public key PK_U to a real-world identity. This is done by submitting a Certificate Signing Request (CSR) to the trusted CA.

1. Construct CSR: U creates a CSR data structure containing:

- Its HBS public key PK_U.
- Its claimed identity information ID_U, which may include: device serial number, MAC address, IP address (for network devices), organizational affiliation, and other unique identifiers as per the network's policy.
- A digital signature over the CSR content ($PK_U \parallel ID_U$) using its own private key SK_U. This signature proves possession of the corresponding private key and protects the CSR from tampering.

2. Submission and Verification: U transmits the CSR to the CA via a secure, authenticated channel (e.g., established during secure manufacturing provisioning or via an out-of-band physical process). Upon receipt, the CA performs a rigorous verification:

- **Format and integrity check:** Verifies the CSR structure and the attached signature using PK_U, confirming that the requester possesses SK_U.
- **Identity verification:** Validates the provided ID_U against the system's trusted records. This may involve checking against a pre-approved inventory list, using a separate secure authentication protocol, or relying on a trusted introducer. The goal is to ensure that ID_U genuinely and uniquely corresponds to the physical device or user represented by U.

3.2.3 HBS-Based Certificate Issuance

If the verification is successful, the CA issues a digital certificate for U. Crucially, the CA uses its own HBS private key to sign the certificate, ensuring the entire certification chain is quantum-resistant.

The CA computes the digest H of the To-Be-Signed (TBS) portion of $Cert_U$ using the same hash function underlying its HBS scheme. It then signs this digest using its private HBS key SK_{CA}:

$$\sigma_{CA} = HBS.Sign\left(SK_{CA}\right) \tag{3}$$

The CA sends the signed $Cert_U$ back to entity U. U stores this certificate locally. The CA's own public key PK_{CA} is assumed to be pre-distributed to all potential verifiers (like Alice and Bob) in the QKD network as a trust anchor.

Upon completion of this phase, each legitimate participant (Alice, Bob) possesses:

- A securely stored HBS private key (SK_A, SK_B).
- A corresponding, CA-issued digital certificate $(Cert_A, Cert_B)$ containing their identity, their HBS public key (PK_A, PK_B), and the CA's HBS signature.
- The trusted root public key PK_{CA} of the CA.

This establishes the foundational trust infrastructure. In any subsequent QKD session, presenting $Cert_A$ or $Cert_B$ allows the other party to verify the sender's identity using the known PK_{CA}, and then use the extracted PK_A or PK_B to authenticate all session-specific messages via HBS signatures.

3.3 Mutual Certificate Verification and Parameter Negotiation with HBS Authentication

The core objectives of this phase are twofold: 1) to perform mutual identity authentication, ensuring that each party is a legitimate entity holding a valid certificate; 2) to negotiate the parameters and algorithms

required for subsequent post-processing. This framework employs one-time session nonces and HBS signatures to achieve strong identity verification and prevent replay attacks, thereby establishing a secure and trusted foundation for the entire session [5].

3.3.1 Phase Objectives and Design Rationale

Following quantum transmission, Alice and Bob have not yet verified each other's identities. Serving as the starting point for all subsequent classical communication, this phase must accomplish:

- **Identity-Binding Authentication:** Confirm that the other party possesses the private key corresponding to the public key in their certificate, thereby proving their claimed identity [9].
- **Session Initialization:** Generate and exchange session-specific random numbers (R_A, R_B). These nonces will be used to bind all subsequent messages within this session, effectively preventing replay attacks.
- **Capability Negotiation:** Exchange and confirm the algorithms (e.g., error-correction code type, hash function) and parameters supported by both parties to ensure compatibility for post-processing.

3.3.2 Detailed Interaction Protocol

Assuming the quantum transmission is complete and Bob acts as the initiator of this phase (this role is interchangeable), the detailed steps are as follows:

Step 1: Bob Initiates Authentication and Negotiation:

1. **Generate Session Nonce**: Bob generates a high-entropy random number R_B. This nonce will serve as one of the "challenges" for this QKD session and will be used to bind all subsequent signatures sent by Bob.
2. **Prepare Capability List**: Based on his device's capabilities, Bob creates a structured list n_B enumerating the algorithms and parameters he supports. Containing:
   - HBS algorithm: SLH-DSA-SHAKE256-256s.
   - Error-correction code: H_{LDPC} (6400, 3200) (code length 6400, information length 3200).
   - Hash function: SHA-3-256 [19].
   - QBER threshold: 0.05 (5%) [17].
   - PA seed A.
   - Maximum LDPC decoding iterations: I_{max} = 50.
3. **Construct Message for Signing**: Bob constructs the message to be authenticated as M_B = (R_B, n_B, "Negotiation"). and signs it using SK_B:

$$\sigma_B = HBS.Sign\left(SK_B, H(M_B)\right) \tag{4}$$

4. **Send Initial Message**: Bob sends ($Cert_B$, R_B, n_B, σ_B) to Alice.

Step 2: Alice Verifies and Responds:

1. **Verify Bob's Certificate**: Alice uses PK_{CA} to verify the CA's signature on $Cert_B$. Successful verification confirms PK_B.
2. **Verify Bob's Signature**: Alice uses PK_B to verify σ_B against M_B. Success proves the message is untampered, and Bob is genuine.
3. **Generate Own Nonce and Parameters**: Alice generates a session nonce R_A and filters n_B to generate her capability list n_A.

4. **Construct Response Signature**: Alice constructs her response message $M_A = (R_B, R_A, n_A,$ "Negotiation") and signs it using SK_A:

$$\sigma_A = HBS.Sign\left(SK_A, H(M_A)\right) \tag{5}$$

5. **Send Response Message**: Alice sends $(Cert_A, R_A, n_A, \sigma_A)$ to Bob.

Step 3: Bob Completes Mutual Authentication:

1. **Verify Alice's Certificate and Signature**: Bob verifies $Cert_A$ using PK_{CA} and σ_A using PK_A against M_A.
2. **Confirm Parameters**: Bob confirms n_A is consistent with n_B, finalizing parameters for subsequent stages (LDPC code, hash function, QBER threshold, etc.).

The mutual certificate verification and parameter negotiation phase is successfully completed. The interaction flow is depicted in Fig. 7.

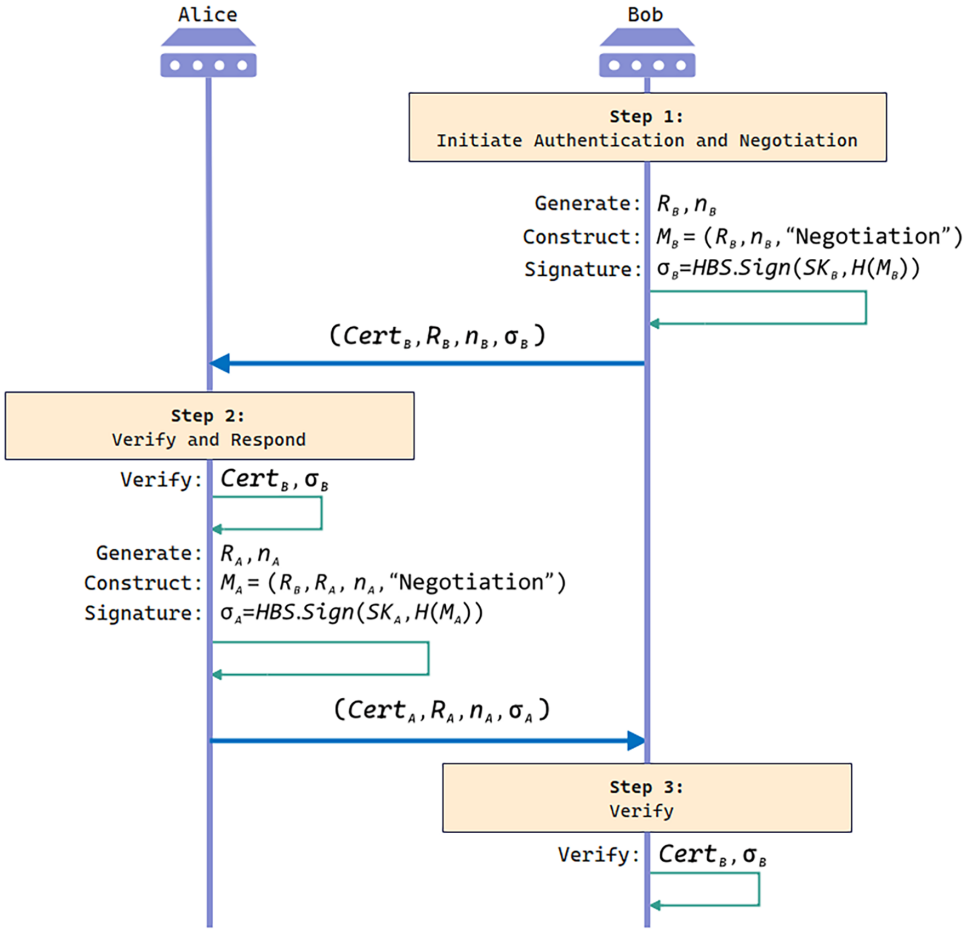

Figure 7: HBS authentication flow during the mutual certificate verification and parameter negotiation phase.

3.4 HBS Authentication in the Basis Sifting Phase

This part outlines how HBS-based authentication is incorporated into the basis sifting phase of QKD post-processing. Once a mutually authenticated session is set up, this stage focuses on the secure transmission of basis information necessary for sifting through raw quantum measurement data. The main goal of security

here is to maintain the authenticity and integrity of the basis vectors [17] being exchanged, thereby thwarting any attempts by MitM attackers to interfere with the sifting process.

3.4.1 Purpose of Basis Sifting and Associated Risks

The initial stage of basis sifting involves the classical communication that processes the out-comes of quantum transmissions. During this phase, Alice and Bob disclose their measurement bases (either X or Z) [9] and keep only the bits that correspond to matching bases to create sifted keys. A significant threat arises from Eve, who might manipulate the basis announcements, leading Alice and Bob to either discard valid bits or keep invalid ones, thereby compromising key information without any noticeable errors. Therefore, it is essential to authenticate the received basis sequence.

3.4.2 Integrated Authentication Protocol for Basis Sifting

Leveraging the authenticated session context, the authenticated sifting protocol is designed as follows:

Step 1: Bob Sends and Authenticates His Measurement Bases (L_B)

1. **Prepare Message**: Bob retrieves his measurement basis sequence L_B.
2. **Generate Signature**: Bob constructs $M_B^{Sift} = (R_A, L_B, \text{“Sifting”})$ signs it using SK_B:

$$\sigma_B^{Sift} = HBS.Sign\left(SK_B, H\left(M_B^{Sift}\right)\right) \tag{6}$$

3. **Transmit**: Bob sends the tuple (L_B, σ_B^{Sift}) to Alice.

Step 2: Alice Verifies and Sends Her Generation Bases (L_A)

1. **Verify Bob's Message:** Alice uses $PK_B$to verify σ_B^{Sift} against M_B^{Sift}. Success confirms L_B is authentic, authentic and untampered.
2. **Prepare & Sign Her Message**: Alice retrieves her generation basis sequence L_A. She constructs her authentication message as $M_A^{Sift} = (R_B, L_A, \text{“Sifting”})$, signing it using SK_A:

$$\sigma_A^{Sift} = HBS.Sign\left(SK_A, H(M_A^{Sift})\right) \tag{7}$$

3. **Transmit:** Alice sends the tuple $\left(L_A\sigma_A^{Sift}\right)$ to Bob.

Step 3: Bob Verifies Alice's Message

Bob uses $PK_A$to verify σ_A^{Sift} against M_A^{Sift}. Successful verification confirms the authenticity and integrity of L_A.

Following these steps, the authenticated basis sifting process is illustrated in Fig. 8.

After mutual verification, both parties locally perform the sifting operation: they compare $L_A[i]$ and $L_B[i]$ for each qubit index i. They discard the bit positions where the bases differ and keep the bits where $L_A[i] = L_B[i]$. These kept bits form their respective sifted keys, S_A and S_B.

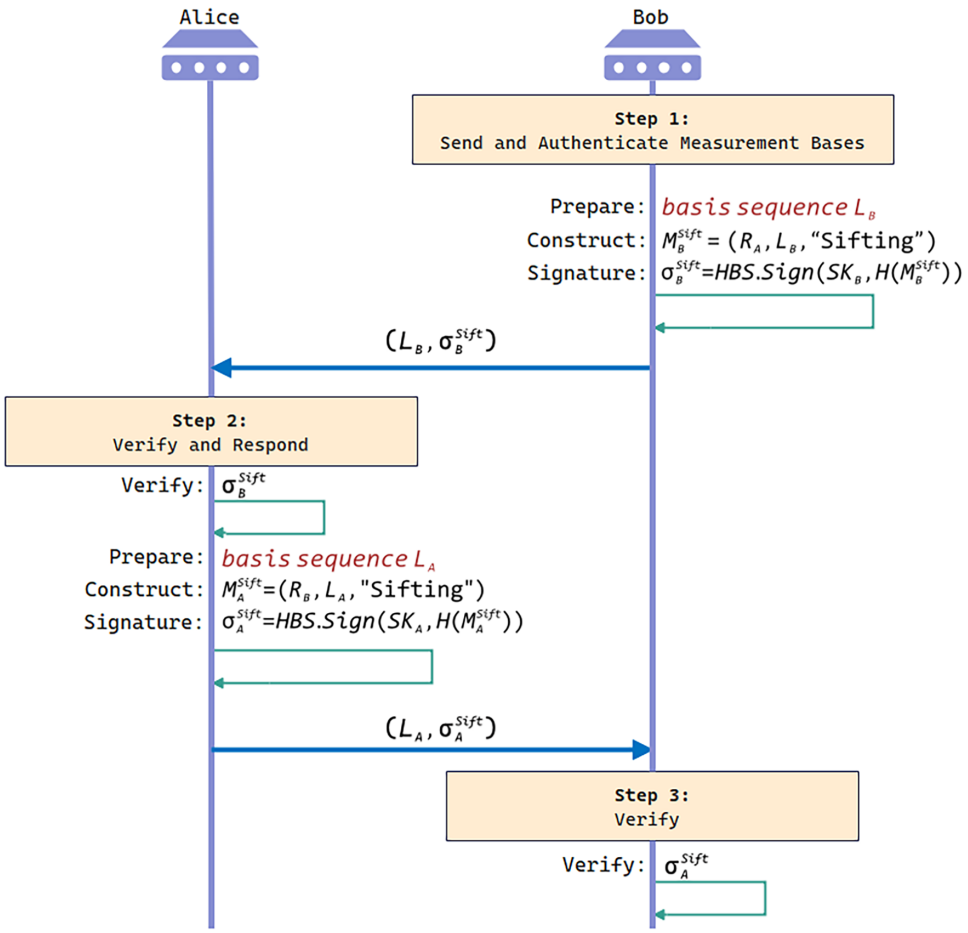

Figure 8: HBS authentication flow during the basis sifting phase.

3.5 HBS Authentication in the Parameter Estimation Phase

This section details the integration of HBS-based authentication into the Parameter Estimation stage. Following the generation of sifted keys (S_A and S_B), this phase aims to assess quantum channel security by estimating the Quantum Bit Error Rate (QBER). The core security objective is to guarantee the integrity and authenticity of publicly compared sample data [17], preventing Eve from concealing eavesdropping by manipulating the QBER.

3.5.1 Purpose of Parameter Estimation and Associated Threats

The stage of estimating parameters is essential for identifying eavesdropping activities [17]. Alice and Bob share a random selection of sifted key bits openly, with the Quantum Bit Error Rate (QBER) indicating possible information exposure to Eve. If the QBER exceeds a certain limit, the protocol must be terminated [9]. The main risk involves tampering with data: Eve might change the indices or bits of the samples to reduce the QBER deceptively, concealing her involvement. Therefore, it is crucial to verify both the positions of the samples and the values of the bits.

3.5.2 Integrated Authentication Protocol for Parameter Estimation

Leveraging the authenticated session state, the protocol involves two rounds of signed communication:

Step 1: Bob Prepares and Authenticates His Sample Data

1. **Generate Random Sample Set**: Bob generates a random sequence $Q_B = [l_1, l_2, ..., l_k]$, where each l_i is an index within the length of his sifted key S_B. To avoid sample overlap, Q_B is restricted to the first 20% of the sifted key.
2. **Extract Sample Bit Values**: Bob extracts the bit values from his sifted key S_B at the positions specified in Q_B. Let this bit string be $V_B = S_B[Q_B]$.
3. **Generate Signature**: Bob constructs the authentication message as $M_B^{PE} = (R_A, Q_B, V_B, \text{“Estimation”})$ and signs it using SK_B:

$$\sigma_B^{PE} = HBS.Sign\left(SK_B, H(M_B^{PE})\right) \tag{8}$$

4. **Transmit**: Bob sends the tuple $(Q_B, V_B, \sigma_B^{PE})$ to Alice.

Step 2: Alice Verifies, Computes QBER and Responds

1. **Verify Bob's Data**: Alice uses PK_B to verify σ_B^{PE} against M_B^{PE}. Success confirms the authenticity of Q_B and V_B.
2. **Compute QBER**: Alice extracts $V_A = S_A[Q_B]$ and calculates the QBER:

$$QBER_{AB} = \frac{Number\ of\ mismatches\ between\ V_A\ and\ V_B}{k} \tag{9}$$

She checks if $QBER_{AB}$ is below the agreed security threshold.

3. **Prepare and Generate Signature**: Alice generates Q_A (sample index set, restricted to the last 20% of S_A avoid overlap with Q_B), extracts her values $V_A = S_A[Q_A]$, and constructs $M_A^{PE} = (R_B, Q_A, V_A, \text{“Estimation”})$, signing it using SK_A:

$$\sigma_A^{PE} = HBS.Sign\left(SK_A, HM_A^{PE}\right) \tag{10}$$

4. **Transmit**: Alice sends the tuple $(Q_A, V_A, \sigma_A^{PE})$ to Bob.

Step 3: Bob Verifies and Independently Computes QBER

1. **Verify Alice's Data**: Bob use PK_A to verify σ_A^{PE} against M_A^{PE}.
2. **Compute QBER and Finalize**: Bob computes $QBER_{BA}$ and confirms it is below the threshold. The protocol proceeds only if both parties agree to continue.

Step 4: Sample Deletion

Both parties securely delete bits at indices Q_A and Q_B from their sifted keys to form raw keys X^N(Alice) and Y^N(Bob). The complete authenticated parameter estimation protocol is illustrated in Fig. 9.

3.6 HBS Authentication in the Error Correction and Verification Phase

This section details the integration of HBS-based authentication into the error correction (reconciliation) and verification stage. Following parameter estimation, Alice and Bob hold raw keys X^N and Y^N with discrepancies [20]—this phase reconciles the keys to ensure $Y'^N = X^N$. while authenticating all correction information. Tampering during this phase could lead to key inconsistency, compromising the protocol.

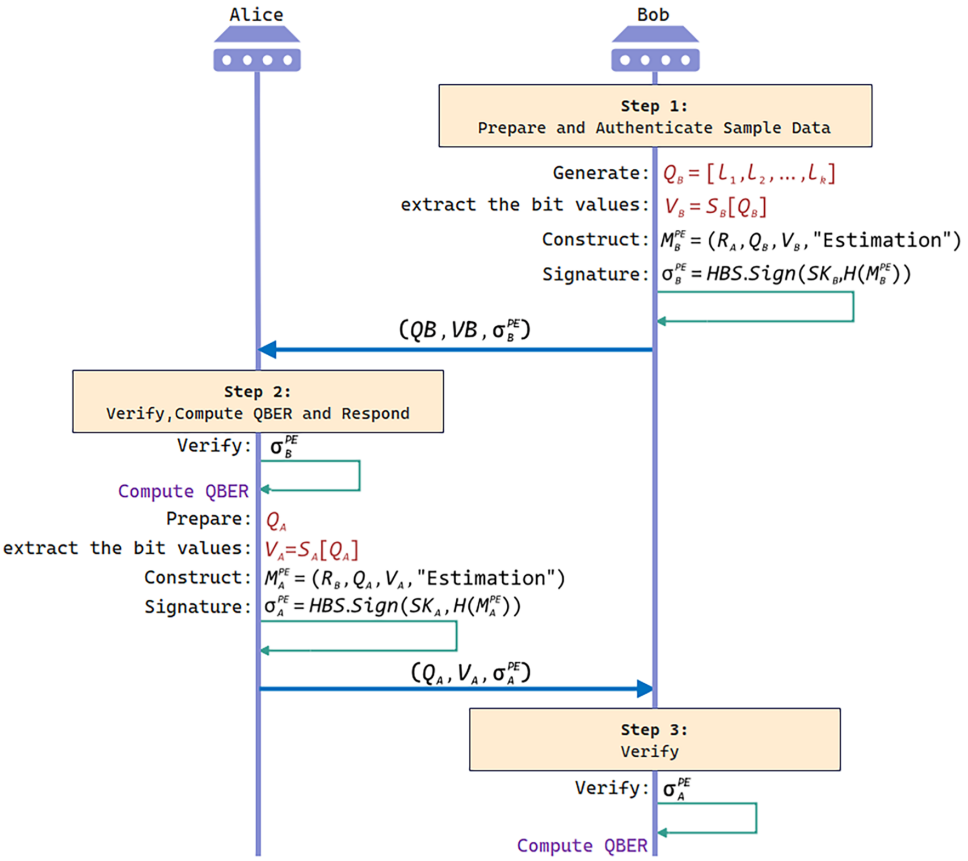

Figure 9: HBS authentication flow during the parameter estimation phase.

3.6.1 The LDPC Reconciliation Process

We employ LDPC code-based reconciliation scheme [21] (consistent with negotiated parameters H_{LDPC} and $I_{\max}$). The core idea is for Alice to send a syndrome (compressed error correction information) and hash digest of her key to Bob, who uses his raw key and the syndrome to decode errors via iterative belief propagation. The hash digest verifies key consistency.

As illustrated in Fig. 10. The core idea is for Alice to send Bob a syndrome (or compressed error correction information) of her key, which Bob uses along with his own key to decode and correct errors through an iterative belief propagation algorithm. A hash commitment of Alice's key is also sent for final verification [21].

3.6.2 Integrated Authentication Protocol for LDPC Reconciliation

Step 1: Initial Syndrome and Commitment Exchange

1. **Alice Prepares Correction Data:**
   - Computes the syndrome of her raw key: $S = H_{LDPC} \cdot \left(X^N\right)^T \pmod 2$.
   - Computes the hash digest of her key: $h = H\left(X^N\right)$ (hash digest, ensuring binding to the raw key).
2. **Alice Signs and Sends:**
   - Constructs $M_A^{EC} = (R_B, S, h, \text{“Correction”})$.
   - Generates the signature: $\sigma_A^{EC} = HBS.Sign(SK_A, H(M_A^{EC}))$.
   - Sends the tuple (S, h, σ_A^{EC}) to Bob.

Step 2: Bob Verifies, Decodes Locally, and Reports Result

Bob uses Alice's public key PK_A to verify σ_A^{EC} against the reconstructed M_A^{EC}. If verification fails, the protocol is aborted. If successful, Bob stores S and h, and begins the LDPC decoding process using his raw key Y^N and the syndrome S.

1. **Verify Alice's Message**: Bob uses PK_A to verify σ_A^{EC} against M_A^{EC}. If verification fails, the protocol is aborted.
2. **LDPC Decoding**: Bob uses his raw key Y^N and syndrome S to perform iterative belief propagation decoding. The decoding terminates when either the syndrome of the decoded key is 0 (error correction complete) or $I_{max} = 50$ is reached.
3. **Report Result**:
   - **Success:** If $H(Y'^N) = h$. (decoded key matches Alice's digest), Bob constructs:
   - $M_B^{final} = (R_A,''\text{SUCCESS}'', h,''\text{Correction}'')$ and signs it:
   - **Failure:** If $I_{\max}$ is reached without a match. Bob sends $M_B^{final} = (R_A,''\text{FAIL}'',''\text{Correction}'')$ with signature.
4. **Transmit Result**: Bob sends the tuple $(M_B^{final}, \sigma_B^{result})$ to Alice.

Step 3: Alice Acknowledges

Alice verifies using PK_B. If the result is "SUCCESS", she confirms Bob holds the correct reconciled key $K_{rec} = X^N = Y'^N$; if "FAIL", the protocol aborts. The HBS authentication flow during error correction is illustrated in Fig. 11.

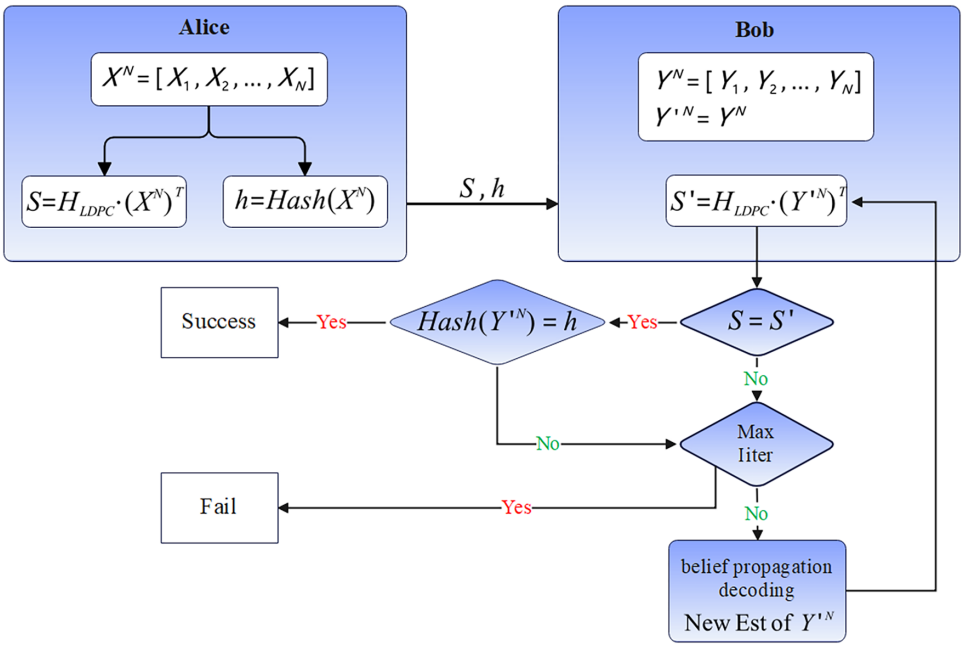

Figure 10: LDPC-based error correction process in QKD post-processing.

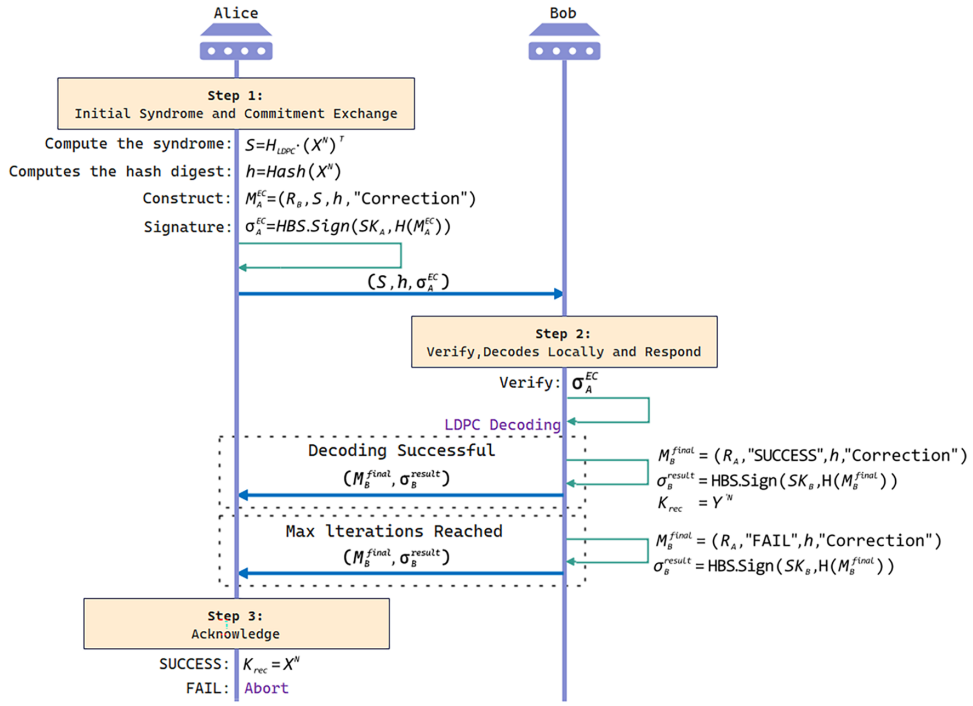

Figure 11: HBS authentication flow during the LDPC-based error correction and verification phase.

3.7 Considerations for the Privacy Amplification Phase

This section addresses the final stage of the QKD post-processing chain: Privacy Amplification (PA) [22]. The primary objective of PA is to distill a shorter, information-theoretically secure final key [23] from K_{rec}, eliminating residual information Eve may have acquired. Within the proposed framework, PA does not require additional explicit authentication [23].

3.7.1 Purpose and Input Security Guarantees

Following successful error correction, K_{rec} is identical for Alice and Bob, but partial information may have been leaked during parameter estimation and error correction. PA applies a Strongly Universal hash function family to compress [22]. K_{rec} into K_{final} making Eve's knowledge negligible. The input K_{rec} and PA parameters are already secured by prior authentication, so PA reduces to local deterministic computation.

Since the parameters determining the PA compression [23] (the total estimated leakage $\lambda = \lambda_{QBER} + \lambda_{EC}$) and the input key K_{rec} are already secured, the PA stage reduces to a local, deterministic computation using a publicly agreed method.

3.7.2 Implementation Based on Toeplitz Matrix Hashing

The PA is executed locally and identically by Alice and Bob using a Toeplitz matrix-based universal hash family [23], which is efficient and provably secure. The procedure [23] is as follows:

1. **Determine Final Key Length:** Based on total estimated leakage $\lambda = \lambda_{QBER} + \lambda_{EC}$ (leakage from parameter estimation and error correction), the final key length is set to $r = n - \lambda - s$.

2. **Generate Random Seed:** Alice generates a random seed A and sends it to Bob via the classical channel. No additional authentication is needed—seed leakage does not compromise K_{final} due to the randomness of the hash function.
3. **Construct Hash Matrix:** Both parties construct a binary Toeplitz matrix $T(A)$ of size $r \times (n-r)$ from A. The finalcompression matrix is $G(A) = [I_r \mid T(A)]$, where I_r is the $r \times r$ identity matrix.
4. **Compute Final Key:** The reconciled key K_{rec} is split into the first r bits, K_1, and the remaining $n-r$ bits, K_2. The final secret key is computed as:

$$K_{final} = G(A) \cdot K_{rec} = K_1 \oplus T(A) \cdot K_2 \tag{11}$$

This process ensures that both parties derive the same K_{final} from their shared K_{rec} and seed A, completing the key distillation without further authenticated communication.

4 Performance Evaluation and System-Level Analysis

4.1 Experimental Objectives and Design Principles

This chapter aims to assess how various authentication methods influence the functioning of QKD systems, rather than merely comparing cryptographic primitives on their own. It specifically emphasizes aspects such as scalability, overall system overhead, and alignment with the security framework of QKD.

Drawing from the insights presented in Section 2 and 3, the subsequent design principles will direct the assessment of the experiment:

1. System-Oriented Evaluation.

Authentication methods are assessed as part of the overall QKD post-processing system, in-stead of being considered independent cryptographic techniques.

2. Model-Consistency Awareness.

The studies differentiate between the performance of operations and the framework of security dependencies. Specifically, the assessment does not aim to experimentally validate information-theoretic security [18]; rather, it emphasizes whether the authentication system adds extra security assumptions.

3. Scenario-Driven Comparison.

Tests are performed in practical QKD operational conditions, encompassing the scaling of multi-user networks and the authentication of brief messages multiple times, reflecting real-world QKD implementations.

Accordingly, three complementary experiments are designed:

- **Experiment I:** Scalability analysis of authentication schemes in QKD networks
- **Experiment II:** Impact of authentication overhead on QKD post-processing performance
- **Experiment III:** Comparative evaluation of different HBS schemes in QKD environments

4.2 Experimental Setup and Common Parameters

4.2.1 QKD Protocol and Post-Processing Configuration

All experiments utilize a QKD protocol grounded in BB84, incorporating classical post-processing steps as outlined in Section 3. The parameters listed in Table 2 remain constant throughout all experiments unless stated otherwise.

Authentication is applied to all classical messages exchanged during:

- Mutual certificate verification and parameter negotiation
- Basis sifting
- Parameter estimation
- Error correction verification

Table 2: Common QKD system parameters.

Parameter	Value
Raw key length	10^6 bit
Basis sifting ratio	50%
QBER	2%
Error correction scheme	LDPC
LDPC code rate	1/2
Maximum LDPC iterations	50
Privacy amplification	Toeplitz universal hashing

4.2.2 Authentication Schemes under Comparison

Three categories of authentication schemes evaluated in this work are summarized in Table 3:

Table 3: Authentication schemes considered.

Category	Scheme	Notes
Symmetric-key	MAC (HMAC-SHA-256)	Requires pre-shared keys
Mathematical-problem PQC	Dilithium-3	Lattice-based
Hash-based signature	SLH-DSA-SHAKE256-256s	Stateless HBS

For symmetric-key authentication, pairwise pre-shared keys are assumed.

A certificate-oriented PKI framework is presumed for the PQC and HBS schemes, as out-lined in Section 3.2.

4.3 Experiment I: Scalability Analysis in Multi-User QKD Networks

4.3.1 Objective

This study examines the impact of user count on the overhead associated with authentication in a QKD network, emphasizing the intricacies of key management and the expenses of communication.

The objective is to show that although symmetric-key authentication works well for smaller systems, it becomes unfeasible as the network expands [5]. In contrast, HBS-based authentication offers scalability without the need for additional security assumptions.

4.3.2 Network Model

We consider a QKD network consisting of N users, where any pair of users can establish a QKD session.

The total number of potential QKD links in the network is:

$$S(N) = \frac{N(N-1)}{2} \tag{12}$$

Three authentication paradigms are considered:

1. **Pre-shared symmetric-key authentication**, where each pair of users shares a unique long-term secret key.
2. **Mathematical-problem-based PQC signature authentication**, using a PKI model.
3. **HBS authentication**, also using a PKI model.

4.3.3 Key Management Complexity

In authentication systems that utilize symmetric keys, every user pair is required to keep a unique pre-shared secret key. Consequently, the overall quantity of long-term authentication keys needed within the network is:

$$K_{sym}(N) = \frac{N(N-1)}{2} \tag{13}$$

Conversely, authentication methods based on PQC and HBS utilize a public-key framework, where every user possesses a single pair of public and private keys along with an associated certificate:

$$K_{PK} = N \tag{14}$$

This disparity results in a quadratic increase in the complexity of key management for symmetric-key authentication, whereas public-key authentication systems experience a linear growth in this aspect.

4.3.4 User Join and Maintenance Operations

Scalability is further affected by the operational cost of adding new users and maintaining authentication material.

When a new user joins a QKD network with N existing users:

- Symmetric-key authentication: requires secure key establishment with all existing users:

$$C_{join}^{sym}(N) = N \tag{15}$$

- PQC- and HBS-based authentication require only a single registration operation with the CA:

$$C_{join}^{pk} = 1 \tag{16}$$

Similarly, key updates or revocation events in symmetric-key systems require re-distribution of keys across multiple user pairs, whereas public-key-based schemes confine such operations to certificate updates.

4.3.5 Quantitative Results

To illustrate the scalability trends, we evaluate the above metrics for different network sizes, as detailed in Table 4.

Table 4: Number of long-term authentication keys or credentials.

Number of Users N	Symmetric Keys K_{sym}	PQC/HBS K_{PK}
2	1	2
10	45	10
50	1225	50
100	4950	100
500	124,750	500

(1) Number of Long-Term Authentication Materials

The Fig. 12 illustrates the quantity of long-term authentication resources needed in a multiuser QKD network, contrasting symmetric-key authentication with public-key methods (PQC and HBS). While symmetric-key authentication shows a quadratic increase as the user count rises, public-key authentication demonstrates a linear scaling.

(2) User Join Operations

Table 5 lists the number of secure operations required when a new user joins a QKD network of varying sizes.

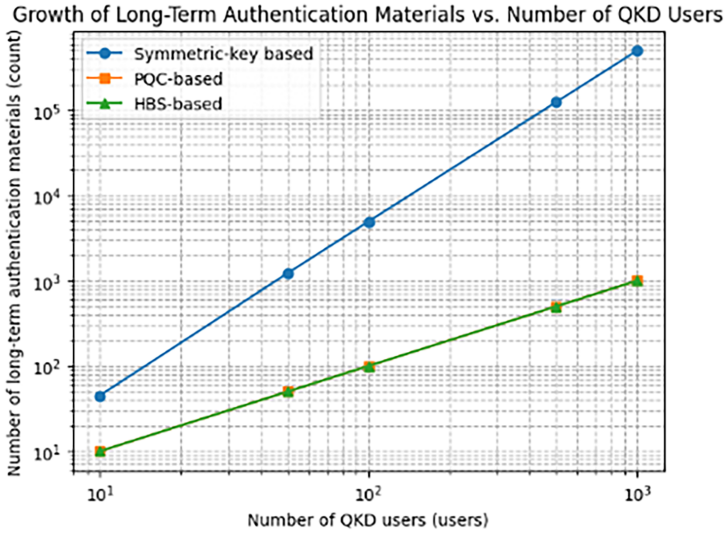

Figure 12: Growth of long-term authentication materials vs. number of QKD users.

Table 5: Number of secure operations required when a new user joins.

Network Size N	Symmetric-Key Scheme	PQC/HBS Scheme
10	10	1
50	50	1
100	100	1
500	500	1

The Fig. 13 depicts the quantity of authentication processes needed when a new participant enters an established QKD network. In the case of symmetric-key authentication, secure key establishment with all current users is necessary, resulting in a linear increase relative to the size of the network. In contrast, public-key authentication necessitates merely one registration action.

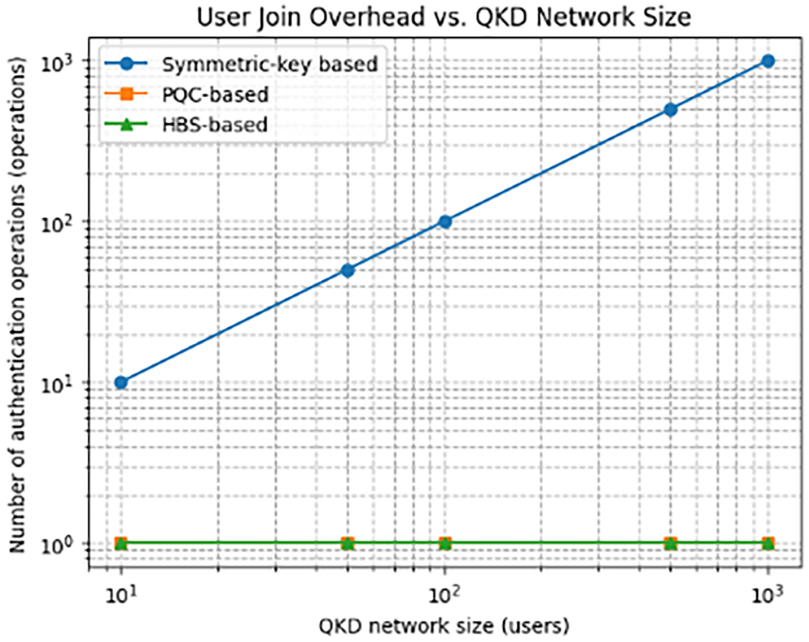

Figure 13: User join overhead vs. QKD network size.

The findings distinctly show that the constraints on scalability for symmetric-key authentication stem from the exponential increase in the complexity of key management and operations [5], not from the costs associated with cryptographic computations. With a rising number of QKD users, the practicality of symmetric-key authentication diminishes significantly due to the overwhelming demands for key provisioning, updates, and ongoing maintenance [9].

Authentication methods utilizing PQC and HBS effectively resolve scalability issues by implementing a public-key framework, resulting in proportional increases in authentication resources and fixed-time user integration processes. Therefore, scalability by itself does not adequately differentiate PQC from HBS approaches, prompting a deeper analysis focused on the consistency of security models and their operational effectiveness [17], which will be explored in later tests.

4.4 Experiment II: Impact of Authentication on QKD Post-Processing Performance

4.4.1 Objective

This experiment aims to assess how authentication affects the performance of QKD post-processing by examining the protocol workflow [24]. Since dedicated QKD hardware is unavailable, the focus is not on determining the exact execution time. Rather, it investigates if the inclusion of authentication creates a structural performance limitation [17] in the QKD post-processing phase.

4.4.2 Simulation Environment

The QuTiP library [25] is utilized to model quantum communication and measurement techniques. This simulation generates the classical data necessary for the post-processing of QKD, which encompasses basis details, error metrics, and confirmation messages. The evaluation of authentication overhead is conducted through an analytical timing framework [8] based on the patterns and frequency of these classical message interactions.

A prepare-and-measure QKD protocol is utilized as a key simulation framework [26]. This selection is driven by the observation that the classical post-processing steps involved in this study—such as basis sifting, parameter estimation, verification of error correction, and privacy amplification—are applicable to both prepare-and-measure and entanglement-based QKD protocols [5]. Since authentication is solely implemented for classical control messages during the post-processing phase, the assessed feasibility and overhead aspects in this study are not influenced by the particular method of quantum-state generation [17]. Consequently, the findings derived from the prepare-and-measure approach can be readily extended to other families of QKD protocols at the post-processing stage.

4.4.3 QKD Post-Processing Model

Authentication is applied to classical messages exchanged at each stage. Let M_i denote the number of authenticated messages in stage i. The total number of authentication operations is:

$$M = \sum_i M_i \tag{17}$$

4.4.4 Authentication Overhead Model

The authentication overhead is modeled as:

$$T_{auth} = M \cdot (T_{sign} + T_{verify}) \tag{18}$$

where T_{sign} and T_{verify} represent the signing and verification costs of the authentication scheme [9]. These values are obtained from representative software implementations reported in the literature and are used solely to estimate relative overhead trends rather than exact execution time. In analytical timing model we used representative sign/verify latencies; these can be considered conservative for software implementations. Hardware-accelerated SLH-DSA designs reported in the literature achieve markedly lower latencies and area footprints, and could therefore reduce the per-message T_{auth} assumed in our experiments [27].

The total post-processing time for one QKD session is denoted by T_{QKD}, and the authentication overhead ratio is defined as:

$$R_{auth} = \frac{T_{auth}}{T_{\mathrm{QKD}}} \tag{19}$$

4.4.5 Experimental Parameters

Table 6 summarizes the representative authentication overhead ratios for different raw key block sizes.

4.4.6 Results and Overhead Analysis

Table 7 summarizes the representative authentication overhead ratios for different raw key block sizes under three authentication paradigms: symmetric-key authentication, PQC-based authentication, and HBS-based authentication. Rather than targeting precise execution times, the values are intended to capture comparative overhead trends at the system level.

Table 6: Simulation and overhead modeling parameters.

Parameter	Value
Quantum simulation tool	QuTiP
QKD protocol	Prepare-and-measure
Raw key block size	10^5–10^7 bits
500	500
Authenticated messages per session M	8–12
Authentication schemes	Symmetric-key, PQC-based, HBS-based
Evaluation method	Analytical, workflow-level

Table 7: Representative authentication overhead ratios (%).

Raw Key Block Size (bits)	Symmetric-Key	PQC-Based	HBS-Based
10^5	0.02	0.20	0.80
10^6	0.008	0.08	0.30
10^7	0.003	0.03	0.12

Specifically, the overhead ratios reported in Table 7 are derived from an analytical performance model that combines the classical message flow counts obtained from our QuTiP-based QKD post-processing simulation with representative signing and verification latencies for each authentication primitive, as reported in prior benchmarking studies.

As shown in Table 7 and Fig. 14, the authentication overhead ratio decreases rapidly with increasing raw key block size [28]. This trend occurs because the authentication process is linked to a constant number of classical control messages, while the computational demands of error correction and privacy amplification grow with the volume of key material being processed [17]. The overhead for symmetric-key authentication is minimal across all block sizes. In contrast, authentication based on PQC does present a higher overhead, yet it stays under a few tenths of a percent for practical block sizes. Although HBS-based authentication has a greater cost per operation, its relative overhead remains under one percent for feasible QKD setups and continues to decline as the block size increases.

The findings suggest that, although HBS-based authentication involves greater computational demands, it does not hinder the efficiency of QKD post-processing [17]. The observed overhead traits affirm the practicality of incorporating HBS methods into QKD systems regarding performance. Alongside the scalability benefits shown in Experiment I, these results endorse HBS-based authentication as a legitimate and secure substitute for current QKD authentication methods.

4.5 Experiment III: Comparative Evaluation of Different HBS Schemes in QKD Environments

4.5.1 Objective

Experiments I and II have demonstrated the scalability advantages and protocol-level feasibility of integrating HBS into QKD post-processing. However, HBS encompasses multiple constructions with significantly different performance characteristics [10]. For practical deployment in QKD systems, it is therefore necessary to identify which HBS schemes are more suitable under realistic post-processing constraints.

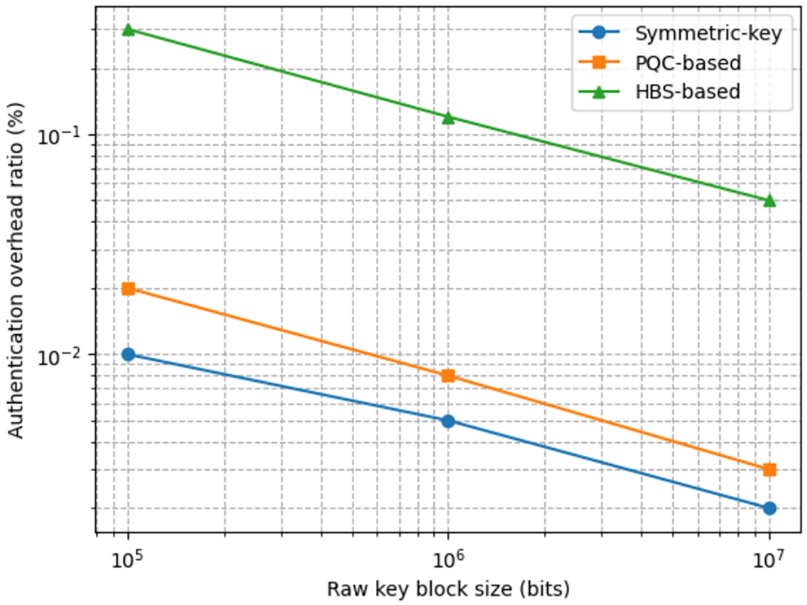

Figure 14: Authentication overhead ratio vs. raw key block size (logarithmic scale).

This experiment aims to assess various HBS schemes within QKD settings, emphasizing their authentication burdens and practical applicability, while offering clear recommendations for choosing appropriate HBS schemes for QKD authentication. Although SLH-DSA is used as a representative stateless HBS construction in the protocol design for its security and robustness, the proposed authentication framework is agnostic to specific HBS instantiations. In Experiment III, we further evaluate multiple HBS schemes to identify the most suitable choices for practical QKD deployments.

4.5.2 Evaluated HBS Schemes

Three representative HBS schemes are selected, covering both stateful and stateless designs:

- LMS (Leighton–Micali Signature) [29]

A stateful Merkle-tree-based signature scheme standardized in RFC 8554 and NIST SP 800–208. LMS offers compact signatures and efficient verification, making it a strong candidate for high-frequency authentication scenarios.

- HSS (Hierarchical Signature System) [30]

A hierarchical extension of LMS that supports a large number of signatures by stacking multiple LMS trees. HSS trades increased signature size and verification cost for improved scalability in long-lived systems.

- SLH-DSA (FIPS-205) [10]

A stateless hash-based signature scheme standardized through the NIST PQC process. SLH-DSA eliminates state management requirements at the cost of significantly larger signatures and higher computational overhead.

These frameworks embody unique design principles within the HBS lineage and are commonly viewed as standard models in the realm of post-quantum cryptography.

4.5.3 Evaluation Metrics

To evaluate the suitability of different HBS schemes in QKD environments, the following metrics are considered.

1. Signature size (S_{sig})

The size of a digital signature directly determines the classical communication overhead incurred during authentication. Larger signatures increase the transmission time of authenticated messages and contribute to the overall authentication delay.

2. Verification cost (C_{ver})

C_{ver} reflects the computational complexity of signature verification and directly affects the authentication processing time at the classical post-processing stage.

3. State management requirement

Indicates whether the signature scheme requires strict state synchronization, which may introduce operational constraints in long-running QKD systems.

4. Authentication overhead ratio (R_{auth})

To capture the combined impact of communication and computation overhead, the authentication overhead ratio is defined as:

$$R_{auth} = \frac{T_{auth}}{T_{\text{QKD}}} \tag{20}$$

where the total authentication time T_{auth} is modeled as

$$T_{auth} = M \cdot \left(T_{ver} \left(C_{ver} \right) + T_{comm} \left(S_{\text{sig}} \right) \right) \tag{21}$$

here, M denotes the number of authenticated classical messages per QKD session, T_{ver} denotes the verification time as a function of the verification cost, and T_{comm} denotes the communication time as a function of the signature size.

All metrics are evaluated at the protocol level using analytical modeling informed by standardized parameter sets and published performance evaluations. The analysis focuses on relative overhead trends rather than implementation-specific absolute timing.

4.5.4 Experimental Setup and Scheme Parameters

The QKD post-processing workflow follows the prepare-and-measure model described in Experiment II. Authentication is applied to all classical control messages exchanged during basis sifting, parameter estimation, error correction verification, and privacy amplification [17]. For each QKD session, the number of authenticated classical messages is fixed and independent of the raw key block size. The raw key block size is set to 10^6 bits, which represents a typical operating regime in practical high-throughput QKD systems and allows the relative impact of authentication overhead to be meaningfully evaluated. Authentication operations are assumed to be performed once per classical message exchange, reflecting a conservative and protocol-consistent deployment scenario.

Three representative HBS schemes—LMS, HSS, and SLH-DSA—are evaluated at approximately NIST security level 1 [10]. This security level is sufficient for long-term post-quantum security in QKD applications while avoiding unnecessary overhead introduced by higher parameter sets. Their representative characteristics, derived from standardized parameter selections and published performance evaluations, are summarized in Table 8.

Table 8: Representative characteristics of HBS schemes (NIST Level 1).

Scheme	Stateful	Signature Size (KB)	Verification Cost (Relative)
LMS	Yes	2.5–3	1×
HSS (2-layer)	Yes	4–6	~1.5×
SLH-DSA-SHAKE256-256s	No	16–30	~5–10×

The reported values are representative and intended to illustrate relative performance trends rather than implementation-specific execution time.

4.5.5 Authentication Overhead in QKD Post-Processing

Using the characteristics summarized in Table 8, the authentication overhead ratio R_{auth} is estimated for each scheme. The resulting representative overhead ratios are reported in percentage form in Table 9, and the comparative overhead trends are visually summarized in Fig. 15.

Table 9: Authentication overhead ratio R_{auth} (%) for different HBS schemes.

Scheme	R_{auth}
LMS	~0.10%
HSS	~0.18%
SLH-DSA-SHAKE256-256s	~0.45%

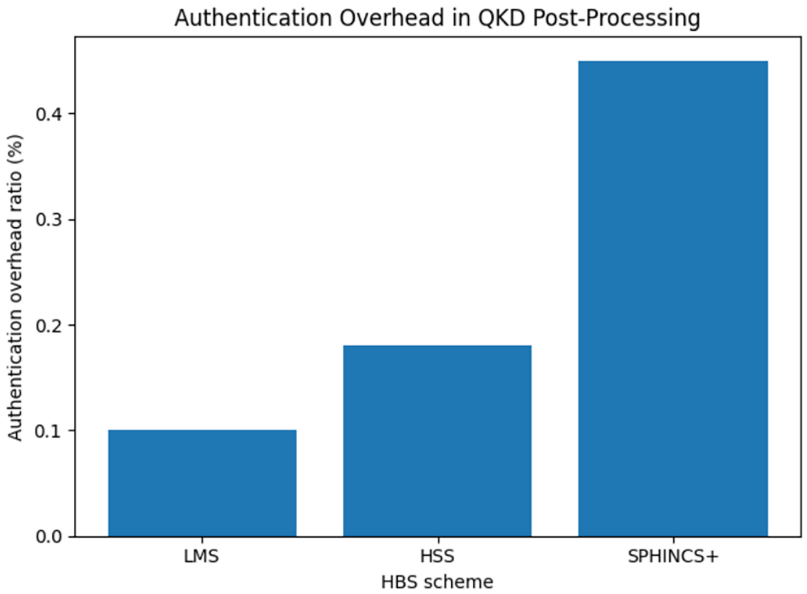

Figure 15: Authentication overhead in QKD post-processing for different HBS schemes.

While SLH-DSA presents the greatest overhead, its cost for authentication is still under one percent of the total time required for QKD post-processing. In contrast, stateful methods like LMS have significantly reduced overhead, making them better suited for regular authentication tasks within QKD systems.

The findings indicate distinct trade-offs among the assessed HBS methods. LMS stands out for its minimal authentication burden, thanks to its small signatures and streamlined verification, making it ideal for QKD post-processing that requires frequent authentication. HSS enhances scalability by accommodating more signatures, albeit with a slight increase in overhead. On the other hand, SLH-DSA, despite its appealing stateless architecture, results in considerably higher communication and computational expenses. While its overhead is manageable from a practical standpoint, it is not as efficient as stateful options for frequent authentication in QKD settings.

4.5.6 Summary

This study conducts a comparative analysis of various hash-based signature methods utilized in the post-processing phase of QKD. The findings reveal that, although all the methods assessed provide security against quantum attacks and are viable in terms of performance, there are notable differences in their authentication overhead [30]. Among these, stateful methods, especially LMS, present the least overhead, making them ideal for regular authentication in QKD frameworks. HSS enhances scalability but comes with a moderate increase in overhead, while SLH-DSA removes the need for state management, resulting in significantly higher communication and computational demands [10]. These results underscore the importance of choosing the right scheme for effective HBS-based QKD authentication, with stateful designs offering a more advantageous balance between performance and complexity in standard QKD settings.

4.6 Parameter Sensitivity Analysis and Rationality of Experimental Settings

To justify the rationality of experimental parameters used in this work, a sensitivity analysis was conducted with respect to three key factors: QBER, raw key block length, and H_{LDPC} reconciliation parameters. The evaluation focuses on their impact on final secret key yield, authentication overhead, and post-processing efficiency. Unless otherwise specified, the baseline configuration uses $H_{LDPC}(6400, 3200)$, raw key length of 10^6 bits, and a BB84 protocol with one-way error correction.

Table 10 illustrates the effect of raw key block length on authentication efficiency under a fixed QBER of 5%. While the absolute authentication time remains nearly constant, increasing the raw key length significantly reduces the authentication cost per final secret bit. When the raw key length increases from 1×10^5 to 1×10^6 bits, the authentication time per bit decreases by nearly one order of magnitude.

Table 10: Effect of raw key block length on authentication efficiency (QBER = 5%).

Raw Key Length (bits)	Final Secret Key (bits)	Authentication Time (ms)	Authentication Time per Bit (μs/bit)
1×10^5	31,600	11.5	0.364
5×10^5	159,000	11.7	0.074
1×10^6	318,000	11.8	0.037

This result indicates that larger raw key blocks effectively amortize the fixed authentication overhead, making 10^6 bits a practical choice for achieving both high efficiency and stable security in QKD post-processing.

Table 11 summarizes the influence of QBER on the final secret key yield and authentication overhead. As expected, the final key length decreases monotonically as QBER increases, due to higher error correction

leakage and stronger privacy amplification requirements. When QBER increases from 2% to 10%, the final key yield ratio drops from 0.452 to 0.207.

Table 11: Impact of QBER on final key yield and authentication overhead.

QBER (%)	Final Secret Key (bits)	Key Yield Ratio	Post-Processing Time (ms)	Authentication Time (ms)	Authentication Overhead (%)
2	452,000	0.452	182	11.6	6.0%
5	318,000	0.318	201	11.8	5.9%
8	238,000	0.238	227	12.1	5.3%
10	207,000	0.207	249	12.4	5.0%

Notably, the absolute authentication time remains nearly constant across different QBER levels, since the number of authenticated control messages in the post-processing stage is largely independent of channel noise. As a result, the relative authentication overhead slightly decreases at higher QBER values, demonstrating that the proposed HBS-based authentication framework maintains stable performance even under noisy channel conditions.

Table 12 compares different H_{LDPC} configurations under identical channel conditions. Shorter H_{LDPC} blocks exhibit higher frame error rates (FER), leading to more discarded blocks and lower final key yield. Conversely, longer LDPC blocks reduce FER and slightly improve key yield, but at the cost of increased decoding iterations and longer post-processing latency.

Table 12: Comparison of H_{LDPC} parameters under QBER = 5%.

H_{LDPC} Parameters	Code Rate	FER (%)	Avg. Iterations	Post-Processing Time (ms)	Final Key (bits)
(5120, 2560)	0.50	6.8	21	173	297,000
(6400, 3200)	0.50	2.1	38	201	318,000
(10,000, 5000)	0.50	0.9	61	278	325,000

The H_{LDPC} (6400, 3200) configuration achieves a balanced trade-off between error correction reliability and computational efficiency. It significantly reduces FER compared to shorter blocks while avoiding the excessive decoding delay of larger configurations, making it a suitable and practical choice for QKD post-processing systems.

5 Conclusion

This study examined the issue of identity verification within QKD systems, focusing on the consistency of security models, scalability, and overall system practicality. Through a thorough evaluation of current authentication methods, we demonstrated that pre-shared symmetric-key authentication faces inherent scalability challenges [5] in multi-user QKD environments. Additionally, authentication methods that rely on mathematical problems in PQC introduce extra and uncertain security assumptions [9], which can undermine the information-theoretic security goals of QKD. To overcome these challenges, we introduced

a new authentication framework for QKD post-processing utilizing HBS. This framework incorporates HBS authentication at all essential phases of QKD post-processing, such as mutual certificate validation, basis sifting, parameter estimation, error correction verification, and privacy amplification. The proposed solution enables scalable public-key-style authentication while depending solely on cryptographic hash functions, which are integral to QKD protocols. Consequently, this framework maintains the security model consistency of QKD without adding new computational assumptions [17].

We conducted a series of experiments focused on system performance to assess the scalability, efficiency, and real-world applicability of HBS-based authentication within QKD frame-works. The findings indicate that HBS-based authentication successfully resolves the key management challenges [5] associated with symmetric-key methods and does not introduce any performance limitations in the post-processing phase of QKD. Additionally, a comparative analysis of various HBS models reveals that stateful designs, especially LMS, provide the best balance be-tween performance and complexity for regular authentication in QKD systems, whereas stateless options like SLH-DSA are still practical when ease of use is a priority.

Overall, this study establishes hash-based signatures as a natural and well-aligned authentication primitive for QKD systems. By unifying scalability, post-quantum security, and security-model consistency, the proposed framework provides a practical and principled foundation for authentication in future large-scale QKD deployments. Future work should explicitly evaluate HBS hardware accelerators integrated with QKD devices and characterize real-world metrics such as power consumption, FPGA/ASIC area and throughput [4]. Such hardware-level studies will complement our system-level model and are needed to fully quantify implementation trade-offs in deployed QKD networks.

Acknowledgement: The authors would like to express their sincere gratitude to Prof. Sijiang Xie for his valuable guidance on the research framework and technical routes. We also appreciate the constructive suggestions from Yalong Yan and Hong Zhao during the experiment design and manuscript revision. Additionally, thanks are extended to the Department of Cyberspace Security and Institute of Information Security at Beijing Electronic Science and Technology Institute for providing the necessary research support.

Funding Statement: This work was supported by the Quantum Science and Technology-National Science and Technology Major Project (QNMP) under Grant Nos. 2021ZD0301301, 2021ZD0300705 and the Yunnan Provincial Key Area Science and Technology Program Project under Grant No. 202502AD080015.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Chaokun Wang and Sijiang Xie; methodology, Chaokun Wang; software, Chaokun Wang; validation, Chaokun Wang, Yalong Yan and Hong Zhao; formal analysis, Chaokun Wang; investigation, Chaokun Wang; resources, Sijiang Xie; data curation, Chaokun Wang; writing—original draft preparation, Chaokun Wang; writing—review and editing, Sijiang Xie, Yalong Yan and Hong Zhao; visualization, Chaokun Wang; supervision, Sijiang Xie; project administration, Sijiang Xie. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hossain Faruk MJ, Tahora S, Tasnim M, Shahriar H, Sakib N. A review of quantum cybersecurity: threats, risks and opportunities. In: Proceedings of the 2022 1st International Conference on AI in Cybersecurity (ICAIC); 2022 May 24–26; Victoria, TX, USA. doi:10.1109/ICAIC53980.2022.9896970.

2. Hafiz MW, Hwang SO. A probabilistic model of quantum states for classical data security. Front Phys. 2023;18(5):51304. doi:10.1007/s11467-023-1293-3.
3. Ahn J, Kwon HY, Ahn B, Park K, Kim T, Lee MK, et al. Toward quantum secured distributed energy resources: adoption of post-quantum cryptography (PQC) and quantum key distribution (QKD). Energies. 2022;15(3):714. doi:10.3390/en15030714.
4. Basu K, Soni D, Nabeel M, Karri R. NIST post-quantum cryptography—a hardware evaluation study. Cryptology EPrint Archive. 2019.
5. Wang LJ, Zhang KY, Wang JY, Cheng J, Yang YH, Tang SB, et al. Experimental authentication of quantum key distribution with post-quantum cryptography. npj Quantum Inf. 2021;7(1):67. doi:10.1038/s41534-021-00400-7.
6. Xu G, Mao J, Sakk E, Wang SP. An overview of quantum-safe approaches: quantum key distribution and post-quantum cryptography. In: Proceedings of the 2023 57th Annual Conference on Information Sciences and Systems (CISS); 2023 Mar 22–24; Baltimore, MD, USA. doi:10.1109/ciss56502.2023.10089619.
7. Yang YH, Li PY, Ma SZ, Qian XC, Zhang KY, Wang LJ, et al. All optical metropolitan quantum key distribution network with post-quantum cryptography authentication. Opt Express. 2021;29(16):25859–67. doi:10.1364/OE.432944.
8. Prakasan A, Jain K, Krishnan P. Authenticated-encryption in the quantum key distribution classical channel using post-quantum cryptography. In: Proceedings of the 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS); 2022 May 25–27; Madurai, India. doi:10.1109/ICICCS53718.2022.9788239.
9. Mosca M, Stebila D, Ustaoğlu B. Quantum key distribution in the classical authenticated key exchange framework. In: Proceedings of the 6th International Workshop on Post-Quantum Cryptography (PQCrypto 2013); 2013 Jun 4–7; Berlin, Germany. p. 136–54.
10. National Institute of Standards and Technology. Stateless hash-based digital signature standard. Gaithersburg, MD, USA: National Institute of Standards and Technology; 2024. Report No.: FIPS 205.
11. Zeng G, Wang X. Quantum key distribution with authentication. arXiv:quant-ph/9812022. 1998.
12. Zheng X, Zhao Z. Quantum key distribution with two-way authentication. Opt Quantum Electron. 2021;53(6):304. doi:10.1007/s11082-021-02845-8.
13. Ranganathan S, Karthikeyan S, Chavan CP, Shanthala PT. Integrating quantum key distribution (QKD) with post-quantum cryptography (PQC): combining BB84 protocol with lattice-based cryptographic techniques. In: Proceedings of the 2024 IEEE 4th International Conference on ICT in Business Industry & Government (ICTBIG); 2024 Dec 13–14; Indore, India. doi:10.1109/ICTBIG64922.2024.10911719.
14. Tarable A, Paganelli RP, Ferrari M. Rateless protograph LDPC codes for quantum key distribution. IEEE Trans Quantum Eng. 2024;5:4100311. doi:10.1109/TQE.2024.3361810.
15. Bennett CH, Brassard G, Crepeau C, Maurer UM. Generalized privacy amplification. IEEE Trans Inf Theory. 1995;41(6):1915–23. doi:10.1109/18.476316.
16. Nath A, Maity S, Banerjee S, Roy R. Quantum key distribution (QKD) for symmetric key transfer. Int J Sci Res Comput Sci Eng Inf Technol. 2024;10(3):270–80. doi:10.32628/cseit24103105.
17. Fung CF, Ma X, Chau HF. Practical issues in quantum-key-distribution postprocessing. Phys Rev A. 2010;81(1):012318. doi:10.1103/physreva.81.012318.
18. Prestridge SA, Dunham JG, Rajan D. Information-theoretic security in BB84 QKD. In: Proceedings of the 2023 33rd International Telecommunication Networks and Applications Conference; 2023 Nov 29–Dec 1; Melbourne, Australia. doi:10.1109/ITNAC59571.2023.10368530.
19. Luo P, Fei Y, Zhang L, Ding AA. Differential fault analysis of SHA3-224 and SHA3-256. Cryptology EPrint Archive. 2016. doi:10.1109/fdtc.2016.17.
20. Nakassis A, Mink A. LDPC error correction in the context of quantum key distribution. In: Proceedings of the SPIE Defense, Security, and Sensing; 2012 Apr 23–27; Baltimore, MD, USA. doi:10.1117/12.919117.
21. Mink A, Nakassis A. LDPC for QKD reconciliation. arXiv:1205.4977. 2012.
22. Bennett CH, Brassard G, Robert JM. Privacy amplification by public discussion. SIAM J Comput. 1988;17(2):210–29. doi:10.1137/0217014.

23. Tang BY, Liu B, Zhai YP, Wu CQ, Yu WR. High-speed and large-scale privacy amplification scheme for quantum key distribution. Sci Rep. 2019;9(1):15733. doi:10.1038/s41598-019-50290-1.
24. Engle RD, Mailloux LO, Grimaila MR, Hodson DD, McLaughlin CV, Baumgartner G. Implementing the decoy state protocol in a practically oriented Quantum Key Distribution system-level model. J Def Model Simul Appl Methodol Technol. 2019;16(1):27–44. doi:10.1177/1548512917698053.
25. Lambert N, Giguère E, Menczel P, Li B, Hopf P, Suárez G, et al. QuTiP 5: the quantum toolbox in Python. Phys Rep. 2026;1153:1–62. doi:10.1016/j.physrep.2025.10.001.
26. Mailloux LO, Morris JD, Grimaila MR, Hodson DD, Jacques DR, Colombi JM, et al. A modeling framework for studying quantum key distribution system implementation nonidealities. IEEE Access. 2015;3:110–30. doi:10.1109/ACCESS.2015.2399101.
27. Amiet D, Leuenberger L, Curiger A, Zbinden P. FPGA-based SLH-DSA implementations: mind the glitch. In: Proceedings of the 2020 23rd Euromicro Conference on Digital System Design (DSD); 2020 Aug 26–28; Kranj, Slovenia. doi:10.1109/DSD51259.2020.00046.
28. Lim CCW, Xu FH, Pan JW, Ekert A. Security analysis of quantum key distribution with small block length and its application to quantum space communications. Phys Rev Lett. 2021;126(10):100501. doi:10.1103/PhysRevLett.126.100501.
29. McGrew D, Curcio M, Fluhrer S. RFC 8554: Leighton-Micali hash-based signatures. New York, NY, USA: ACM Digital Library; 2019. doi:10.17487/RFC8554.
30. Sun S, Liu T, Guan Z, He Y, Jing J, Hu L, et al. LMS-SM3 and HSS-SM3: instantiating hash-based post-quantum signature schemes with SM3. Cryptology EPrint Archive. 2022.