
PQCWC: Post-Quantum Cryptography Winternitz-Chen Anonymous Scheme

Abel C. H. Chen

*Information & Communications Security Laboratory, Chunghwa Telecom
Laboratories, Taoyuan, Taiwan
E-mail: chchen.scholar@gmail.com*

Received 19 March 2026; Accepted 11 June 2026

Abstract

As quantum computing technology matures, it poses a threat to the security of mainstream asymmetric cryptographic methods. In response, the National Institute of Standards and Technology released the final version of post-quantum cryptographic (PQC) algorithm standards in August 2024. These post-quantum cryptographic algorithms are primarily based on lattice-based and hash-based cryptography. Therefore, this study proposes the Post-Quantum Cryptography Winternitz-Chen (PQCWC) anonymous scheme, aimed at exploring the design of anonymous schemes based on PQC for future applications in privacy protection. The anonymous scheme designed in this study is mainly built on the Winternitz signature scheme, which can prevent the original public key from being exposed in the certificate. Furthermore, the PQCWC anonymous scheme integrates the butterfly key expansion mechanism, introducing the world's first hash-based butterfly key expansion mechanism, achieving anonymity for both the registration authority and the certificate authority, thereby fully protecting privacy. In the experimental environment, this study compares various hash algorithms, including Secure

Hash Algorithm-1 (SHA-1), the SHA-2 series, the SHA-3 series, and the BLAKE series. The results demonstrate that the proposed anonymous scheme can achieve anonymity without increasing key length, signature length, key generation time, signature generation time, or signature verification time.

Keywords: Post-quantum cryptography, anonymous scheme, hash-based cryptography, Winternitz signature scheme.

1 Introduction

Quantum computing technology has been rapidly advancing, and with the suitable quantum algorithms, computational efficiency has the potential to achieve exponential acceleration [1]. For instance, Shor quantum algorithm [2] can quickly find hidden subgroups and their periods through quantum Fourier transform [3], enabling the rapid solution of factorization and discrete logarithm problems [4]. However, current mainstream asymmetric cryptography methods (e.g. RSA and elliptic curve cryptography (ECC)) rely on the security of these problems. With quantum computing, this security could be quickly compromised [5]. Therefore, asymmetric cryptography based on factorization and discrete logarithm problems is no longer considered secure [6]. In light of this, the National Institute of Standards and Technology (NIST) has, in recent years, been working towards establishing standards for post-quantum cryptography (PQC) [7]. NIST has solicited submissions for various algorithms based on lattice-based cryptography [8], hash-based cryptography [9], code-based cryptography [10], and multivariate-based cryptography [11] from around the world. In August 2024, NIST finalized the standards for PQC algorithms, including the Stateless Hash-based Digital Signature Algorithm (SLH-DSA) [12], the Module-Lattice-based Digital Signature Algorithm (ML-DSA) [13], and the Module-Lattice-based Key Encapsulation Mechanism (ML-KEM) [14]. Furthermore, several PQC applications have been developed, including in vehicular-to-everything (V2X) communications [15], the Internet of Things (IoT) [16], and blockchain [17]. Among these standard algorithms, the SLH-DSA is a type of hash-based cryptographic algorithm. Due to the irreversible nature of hashing, hash-based cryptography is mainly applied in digital signatures, making it unsuitable for key encapsulation.

The current certificate formats, such as X.509 certificates [18] and Security Credential Management System (SCMS) certificates [19] in

V2X communications, have established standards and specifications. When updating to post-quantum cryptographic algorithms, the primary modification involves changing the Object Identifier (OID) of the algorithm in the certificate format. The corresponding public key of the end entity to be signed is placed in the public key field, and the issuer generates the signature for the to-be-signed data and places it in the signature field. However, since the public key is included in the certificate, it could potentially allow an attacker to track the user associated with the same public key in the user's certificate in subsequent applications. The attacker could then analyze all messages signed with the corresponding private key, raising concerns about user privacy exposure. To address this, the IEEE 1609.2.1 standard [20] introduced the butterfly key expansion (BKE) mechanism [21]. This mechanism protects the user's original public key (referred to as the "caterpillar public key" in this context) by expanding it into "cocoon public keys" and "butterfly public keys." It ensures that the butterfly public key cannot be traced back to the caterpillar public key, thus safeguarding user privacy. However, the BKE mechanism defined in the IEEE 1609.2.1 standard is primarily built on ECC, which relies on the properties of elliptic curves for key expansion. This method is vulnerable to quantum computing attacks and, therefore, cannot provide quantum-safe security levels.

In light of the need for privacy protection and quantum-safe security, this study proposes a PQC anonymous scheme based on hash-based cryptography, named Post-Quantum Cryptography Winternitz-Chen (PQCWC). The proposed method primarily leverages the Winternitz one-time signature scheme [22] to design an anonymous certificate solution, incorporating the BKE mechanism to provide privacy protection, while ensuring quantum-safe security through the properties of hash-based cryptography. The contributions of this study are outlined as follows:

- This study proposes the PQCWC scheme, which provides an anonymous scheme based on hash-based cryptography that ensures quantum-safe security.
- This study proposes the world's first hash-based butterfly key expansion (HBKE) mechanism, which achieves anonymity for every end entity in the system, including the registration authority (RA) and certificate authority (CA), thus fully protecting privacy.
- This study compares various hash algorithms, including Secure Hash Algorithm-1 (SHA-1), the SHA-2 series, the SHA-3 series, and the BLAKE series, demonstrating that the proposed anonymous scheme

achieves anonymity without increasing key length, signature length, key generation time, signature generation time, or signature verification time.

This paper is divided into six sections. Section 2 reviews the existing methods, including a literature review, and introduces the Winternitz one-time signature method, the known certificate request and response process, and the IEEE 1609.2.1 BKE mechanism. Section 3 presents the PQC anonymous scheme (i.e. PQCWC scheme) designed in this study and explains how key expansion is achieved based on the characteristics of hash-based cryptography. Section 4 proposes the HBKE mechanism proposed in this study, detailing how to achieve BKE within the PQCWC scheme by expanding the hash-based caterpillar key into a hash-based cocoon public key, and further into a hash-based butterfly public key. Section 5 provides a performance comparison, constructing the proposed PQCWC and the HBKE mechanism with different hash algorithms and comparing computational efficiency. Finally, Section 6 summarizes the findings of this study and discusses possible future research directions.

2 Literature Reviews

This section first introduces the Winternitz one-time signature method, followed by an explanation of the well-known certificate request and response process, and a discussion of its anonymity and privacy issues. Lastly, the BKE mechanism designed in IEEE 1609.2.1 and its approach to privacy protection will be presented.

2.1 Winternitz One-Time Signature

The Winternitz one-time signature method is a classic hash-based cryptographic approach. It begins by assuming that the length of the data to be signed, denoted as D , is limited to L_D bits. The data is then divided into elements by splitting every w_1 bits into a single data element. This divides the data D into a sequence of m data elements, represented as $D = d_1 || d_2 || \dots || d_m$, where $m = \lceil \frac{L_D}{w_1} \rceil$. Subsequently, a signature value element is generated for each data element individually [22, 23]. The length of each signature value element, denoted as l_s , varies depending on the hash algorithm used, resulting in a total signature length of $L_S = m \times l_s$. The detailed steps are described as follows.

2.1.1 Key generation

During the key generation phase, a sequence of m random integer elements, denoted as $A = \{a_1, a_2, \dots, a_m\}$, is generated as the private key. For each integer element in the private key, $2^{w_1} - 1$ hash computations are performed to obtain a sequence of m hash value elements, denoted as $B = \{b_1, b_2, \dots, b_m\} = \{f(a_1)^{2^{w_1}-1}, f(a_2)^{2^{w_1}-1}, \dots, f(a_m)^{2^{w_1}-1}\}$, where $f(\cdot)$ is the hash function. The value of b_i is the result of performing $2^{w_1} - 1$ hash computations on a_i .

Therefore, as the value of w_1 increases, the number of hash computations required to generate the public key increases exponentially. Conversely, if the value of w_1 is smaller, although fewer hash computations are needed and the public key generation is faster, the number of elements in the sequence m will increase, leading to a growth in both private key length and public key length. Furthermore, while a smaller w_1 value speeds up computations, it may also increase the risk of the key being compromised more quickly. Therefore, selecting an appropriate value for w_1 is an important consideration.

2.1.2 Signature generation

During the signature generation phase, the data to be signed, D , is obtained and divided into a sequence of data elements. Since the data elements are segmented according to the w_1 bits length, each data element falls within the range $[0, 2^{w_1} - 1]$. The private key element a_i undergoes d_i hash computations to produce the signature value element s_i . Similarly, hash computations are performed on each private key element to obtain the signature value sequence $S = \{s_1, s_2, \dots, s_m\} = \{f(a_1)^{d_1}, f(a_2)^{d_2}, \dots, f(a_m)^{d_m}\}$.

The computation time for this process is similar to that of key generation and depends on the value of w_1 . As w_1 increases, the number of hash computations required for generating the signature increases exponentially. Conversely, if w_1 is smaller, the value of m increases, resulting in a larger number of signature value elements and a longer signature sequence.

2.1.3 Signature verification

During the signature verification phase, the data to be signed, D , the signature values, S , and the public key, B , are obtained. For each signature value element s_i , $2^{w_1} - 1 - d_i$ hash computations are performed to produce the verification value element v_i . This process is repeated for each signature value element to obtain the verification value sequence $V = \{v_1, v_2, \dots, v_m\} = \{f(s_1)^{2^{w_1}-1-d_1}, f(s_2)^{2^{w_1}-1-d_2}, \dots, f(s_m)^{2^{w_1}-1-d_m}\}$. If each verification

value element matches its corresponding public key element, then the verification is successful, i.e., $V = \{v_1, v_2, \dots, v_m\} = \{b_1, b_2, \dots, b_m\} = B$.

2.2 Well-known Certificate Request and Response Process

In the basic public key infrastructure (PKI), a CA issues certificates to end entities. The end entity generates a certificate signing request (CSR) for the CA, which includes information about the end entity, the end entity's public key, and a list of requested permissions. After reviewing the end entity's credentials, the CA issues the end entity's certificate, which contains the end entity's information, public key, and permission list [24].

Furthermore, to manage the PKI, a RA is established to review the end entity's credentials and generate the corresponding CSR for the CA. This setup clarifies the division of roles among the entities. The RA handles communication with end entities and verifies their credentials, playing a crucial auditing role in various applications. The CA is responsible for producing end entity certificates based on the CSR [25].

However, in the current PKI system, the issued certificates for end entities include the end entity's original public key. Consequently, if an attacker collects all messages verified with the same public key, it could lead to privacy exposure issues for the user.

2.3 Butterfly Key Expansion in the IEEE 1609.2.1 Standard

Since privacy is especially important for end entities in the V2X communications, effective privacy protection measures are required. The IEEE 1609.2.1 standard proposes the BKE mechanism [20, 21], which uses key expansion to avoid storing the original public key (i.e. the caterpillar key) in the certificate, thereby achieving the goal of privacy protection. This study focuses on explicit certificates, and implicit certificates are not within the scope of this study. Further details are provided below.

2.3.1 Generating caterpillar key pairs

The end entity generates two sets of original key pairs: the caterpillar key pair for signing and the caterpillar key pair for encryption. The signing caterpillar key pair consists of the signing caterpillar private key p_1 and the signing caterpillar public key P_1 ; the encryption caterpillar key pair consists of the encryption caterpillar private key p_2 and the encryption caterpillar public key P_2 . In the IEEE 1609.2.1 standard, ECC is primarily used for asymmetric cryptography. Assuming the elliptic curve base point is G , and the order of

the elliptic curve is n , the signing caterpillar public key P_1 is p_1G , and the encryption caterpillar public key P_2 is p_2G . Furthermore, to establish the same key expansion function and parameter values with the RA, two Advanced Encryption Standard (AES) keys, q_1 and q_2 , are generated. The end entity can then encapsulate the signing caterpillar public key P_1 , the encryption caterpillar public key P_2 , the AES keys q_1 and q_2 , along with the end entity information and the requested permission list into an EeRaCertRequest packet to be sent to the RA [20].

2.3.2 Generating cocoon public keys

When the RA receives the EeRaCertRequest packet, it can use the AES key q_1 and key expansion function $g_1(\iota, q_1)$ to generate a pseudorandom number r_1 , and use the AES key q_2 and key expansion function $g_2(\iota, q_2)$ to generate a pseudorandom number r_2 . The value ι represents a time period known to both the RA and the end entity, with the detailed calculation defined in the IEEE 1609.2.1 standard [20]. The RA can then expand the signing caterpillar public key P_1 based on the pseudorandom number r_1 into the signing cocoon public key $T_1 = P_1 + r_1G$, and expand the encryption caterpillar public key P_2 based on the pseudorandom number r_2 into the encryption cocoon public key $T_2 = P_2 + r_2G$. Finally, the RA sends the signing cocoon public key T_1 , the encryption cocoon public key T_2 , and the requested permission list to the CA.

2.3.3 Generating a butterfly public key

When the CA receives the request packet, it generates a random number r_3 , and uses this random number r_3 to expand the signing cocoon public key T_1 into the signing butterfly public key $U = T_1 + r_3G$. The CA then issues the end entity certificate, placing the butterfly public key in the public key field and the corresponding permission list in the permission field. The CA encrypts the end entity certificate and the random number r_3 using the encryption cocoon public key T_2 , resulting in the ciphertext Z , which is then returned to the RA.

2.3.4 Generating cocoon private keys and a butterfly private key

When the RA receives the message, it forwards the message to the end entity. The end entity uses the shared known time period and the expansion functions to generate the pseudorandom numbers $r_1 = g_1(\iota, q_1)$ and $r_2 = g_2(\iota, q_2)$, and based on these pseudorandom numbers, produces the signing cocoon private key $t_1 = p_1 + r_1$ and the encryption cocoon private key $t_2 = p_2 + r_2$. The

encryption cocoon private key t_2 is then used to decrypt the ciphertext Z , revealing the plaintext contents, which include the end entity certificate and the random number r_3 . Finally, the signing cocoon private key t_1 is expanded using the random number r_3 to generate the butterfly private key $u = t_1 + r_3$, which pairs with the butterfly public key U in the end entity. In practical applications, the end entity can use the butterfly private key u to sign a secure protocol data unit (SPDU), and other end entities can verify the signature using the butterfly public key U .

3 The Proposed Post-Quantum Cryptography Winternitz-Chen Anonymous Schemes

In light of the privacy concerns caused by traditional certificate schemes [24, 25] storing the original public keys of end entities, this study proposes the PQCWC anonymous schemes. These schemes achieve privacy protection by storing the expanded public key in the certificate instead of the original public key through a key expansion mechanism.

To accommodate different application needs, this study presents two PQCWC anonymous schemes that can be chosen based on the application field. Sections 3.1 and 3.2 describe the processes of PQCWC anonymous schemes 1 and 2, respectively, and Section 3.3 mathematically proves the proposed PQCWC anonymous schemes.

The proposed PQCWC anonymous schemes are based on the following assumptions:

- Communication between the end entity and the CA is established through a secure connection.
- The end entity and the CA share known parameters w_2 and a pseudorandom number generator.
- The CA has an encryption key pair (including private key k_{CA} and public key K_{CA}). The CA's encryption key pair can be implemented using a ML-KEM [14] to achieve quantum-level security. Furthermore, the end entity knows the CA's encryption public key K_{CA} .

3.1 PQCWC Anonymous Scheme 1

The process of the proposed PQCWC Anonymous Scheme 1 is illustrated in Figure 1. It includes the generation of the signing key pair by the end entity, the generation of the expanded public key and anonymous certificate by the CA, and the generation of the expanded private key by the end entity.

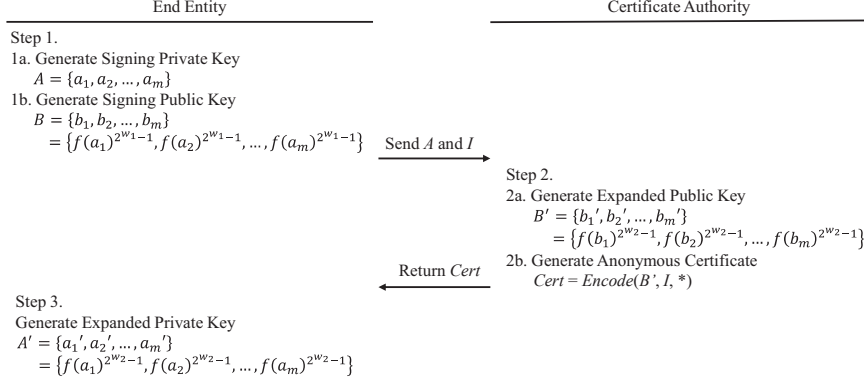


Figure 1 The proposed PQCWC anonymous scheme 1.

3.1.1 Generation of the signing key pair by the end entity

The end entity generates a hash-based signing key pair, consisting of a signing private key $A = \{a_1, a_2, \dots, a_m\}$ and a signing public key $B = \{b_1, b_2, \dots, b_m\} = \{f(a_1)^{2^{w_1}-1}, f(a_2)^{2^{w_1}-1}, \dots, f(a_m)^{2^{w_1}-1}\}$, as defined in Section 2.1. The end entity then packages the signing public key B , along with the relevant end entity information I to be signed, into a certificate request packet and sends it to the CA.

3.1.2 Generation of the expanded public key and anonymous certificate by the CA

Upon receiving the certificate request packet and verifying the end entity eligibility, the CA uses the shared parameter w_2 , known to both the CA and the end entity, to expand the signing public key B . The expanded public key $B' = \{b'_1, b'_2, \dots, b'_m\} = \{f(b_1)^{2^{w_2}-1}, f(b_2)^{2^{w_2}-1}, \dots, f(b_m)^{2^{w_2}-1}\}$ is generated, where $f(\cdot)$ is the hash function, and b'_i is the result of applying $2^{w_2} - 1$ hash computations on each b_i . The CA then produces the anonymous certificate of the end entity based on the expanded public key B' and the relevant end entity information I , and returns it to the end entity.

3.1.3 Generation of the expanded private key by the end entity

After receiving the anonymous certificate, the end entity generates the expanded private key $A' = \{a'_1, a'_2, \dots, a'_m\} = \{f(a_1)^{2^{w_2}-1}, f(a_2)^{2^{w_2}-1}, \dots, f(a_m)^{2^{w_2}-1}\}$. The end entity can then use the expanded private key A' to generate signatures, and other end entities can verify the signatures using the expanded public key B' from the anonymous certificate.

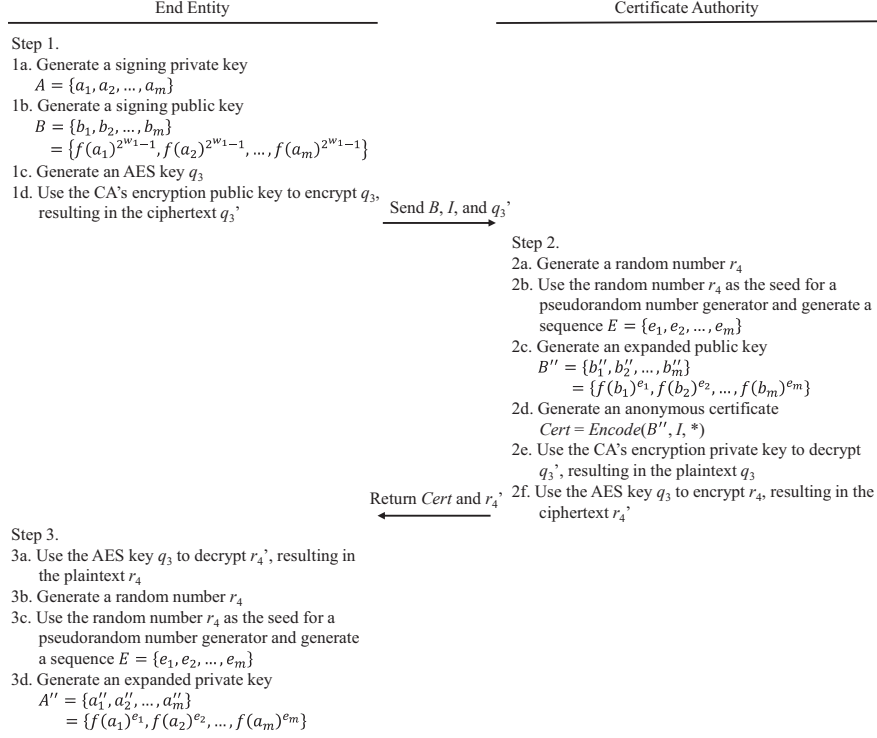


Figure 2 The proposed PQCWC anonymous scheme 2.

3.2 PQCWC Anonymous Scheme 2

The process of the proposed PQCWC Anonymous Scheme 2, as shown in Figure 2, includes the generation of a signing key pair and AES key by the end entity, the generation of the expanded public key and anonymous certificate by the CA, and the generation of the expanded private key by the end entity.

3.2.1 Generation of the signing key pair and AES key by the end entity

The end entity generates a signing key pair based on a hash function, consisting of the signing private key $A = \{a_1, a_2, \dots, a_m\}$ and the signing public key $B = \{b_1, b_2, \dots, b_m\} = \{f(a_1)^{2^{w_1}-1}, f(a_2)^{2^{w_1}-1}, \dots, f(a_m)^{2^{w_1}-1}\}$, as defined in Section 2.1. Furthermore, the end entity generates an AES key q_3 , which is encrypted using the CA's encryption public key K_{CA} , resulting in the ciphertext q_3' . The end entity then packages the signing public key B ,

the ciphertext q'_3 , and the relevant information I to be signed, into a certificate request packet and sends it to the CA.

3.2.2 Generation of the expanded public key and anonymous certificate by the CA

Upon receiving the certificate request packet and confirming the end entity eligibility, the CA decrypts the ciphertext q'_3 using its private key k_{CA} , obtaining the plaintext AES key q_3 . The CA then generates a random number r_4 and uses it as the seed for a pseudorandom number generator, producing a sequence of m pseudorandom elements $E = \{e_1, e_2, \dots, e_m\}$, where each element lies within the range $[0, 2^{w_2} - 1]$. The signing public key B is expanded to generate the expanded public key $B'' = \{b''_1, b''_2, \dots, b''_m\} = \{f(b_1)^{e_1}, f(b_2)^{e_2}, \dots, f(b_m)^{e_m}\}$, where $f(\cdot)$ is a hash function and b_i is the result of applying e_i hash computations to b_i . The CA then generates the anonymous certificate of the end entity based on the expanded public key B'' and the relevant end entity information I , and encrypts the random number r_4 with the AES key q_3 , returning the ciphertext to the end entity.

3.2.3 Generation of the expanded private key by the end entity

Upon receiving the ciphertext containing r_4 , the end entity decrypts it using the AES key q_3 , sets r_4 as the seed for the pseudorandom number generator, and produces a sequence of m pseudorandom elements $E = \{e_1, e_2, \dots, e_m\}$, with each element lying within the range $[0, 2^{w_2} - 1]$. The signing private key A is expanded to generate the expanded private key $A'' = \{a''_1, a''_2, \dots, a''_m\} = \{f(a_1)^{e_1}, f(a_2)^{e_2}, \dots, f(a_m)^{e_m}\}$. The end entity can then use the expanded private key A'' to generate signatures, and other end entities can verify the signatures using the expanded public key B'' .

3.3 Proofs

This section provides the theoretical proofs for the two proposed PQCWC anonymous schemes.

3.3.1 The proofs of the proposed PQCWC anonymous scheme 1

The relationship between the expanded public key and the expanded private key in the proposed PQCWC Anonymous Scheme 1 can be proven using Equation (1). It can be observed that each expanded private key element a'_i , after $2^{w_1} - 1$ hash calculations, becomes the expanded public key element b'_i . Each expanded private key element maintains exactly $2^{w_1} - 1$ hash

calculations with its corresponding expanded public key element.

$$\begin{aligned}
B' &= \{b'_1, \dots, b'_m\} \\
&= \{f(b_1)^{2^{w_2}-1}, \dots, f(b_m)^{2^{w_2}-1}\} \\
&= \{f(a_1)^{2^{w_1}-1+2^{w_2}-1}, \dots, f(a_m)^{2^{w_1}-1+2^{w_2}-1}\} \\
&= \{f(a'_1)^{2^{w_1}-1}, \dots, f(a'_m)^{2^{w_1}-1}\}. \tag{1}
\end{aligned}$$

When generating a signature, the expanded private key can be used to sign the data to be signed $D = \{d_1, d_2, \dots, d_m\}$, as shown in Equation (2). During signature verification, the expanded public key can be used to verify the signature $S = \{s_1, \dots, s_m\}$. As shown in Equation (3), when the signature is correct, each verification value element will match its corresponding public key element.

$$\begin{aligned}
S &= \{s_1, \dots, s_m\} \\
&= \{f(a'_1)^{d_1}, \dots, f(a'_m)^{d_m}\}. \tag{2}
\end{aligned}$$

$$\begin{aligned}
V &= \{v_1, \dots, v_m\} \\
&= \{f(s_1)^{2^{w_1}-1-d_1}, \dots, f(s_m)^{2^{w_1}-1-d_m}\} \\
&= \{f(a'_1)^{2^{w_1}-1-d_1+d_1}, \dots, f(a'_m)^{2^{w_1}-1-d_m+d_m}\} \\
&= \{f(a'_1)^{2^{w_1}-1}, \dots, f(a'_m)^{2^{w_1}-1}\} = B'. \tag{3}
\end{aligned}$$

3.3.2 The proofs of the proposed PQCWC anonymous scheme 2

The relationship between the expanded public key and the expanded private key in the proposed PQCWC Anonymous Scheme 2 can be proven using Equation (4). It can be observed that each expanded private key element a''_i , after $2^{w_1} - 1$ hash calculations, becomes the expanded public key element b''_i . Each expanded private key element maintains exactly $2^{w_1} - 1$ hash calculations with its corresponding expanded public key element. It is worth mentioning that PQCWC Anonymous Scheme 1 can be considered a special case of PQCWC Anonymous Scheme 2, where any element e_i is equal to $2^{w_2} - 1$.

$$\begin{aligned}
B'' &= \{b''_1, \dots, b''_m\} \\
&= \{f(b_1)^{e_1}, \dots, f(b_m)^{e_m}\} \\
&= \{f(a_1)^{2^{w_1}-1+e_1}, \dots, f(a_m)^{2^{w_1}-1+e_m}\} \\
&= \{f(a''_1)^{2^{w_1}-1}, \dots, f(a''_m)^{2^{w_1}-1}\}. \tag{4}
\end{aligned}$$

When generating a signature, the expanded private key can be used to sign the data to be signed $D = \{d_1, d_2, \dots, d_m\}$, as shown in Equation (5). During signature verification, the expanded public key can be used to verify the signature $S = \{s_1, \dots, s_m\}$. As shown in Equation (6), when the signature is correct, each verification value element will match its corresponding public key element.

$$\begin{aligned} S &= \{s_1, \dots, s_m\} \\ &= \{f(a_1'')^{d_1}, \dots, f(a_m'')^{d_m}\}. \end{aligned} \quad (5)$$

$$\begin{aligned} V &= \{v_1, \dots, v_m\} \\ &= \{f(s_1)^{2^{w_1}-1-d_1}, \dots, f(s_m)^{2^{w_1}-1-d_m}\} \\ &= \{f(a_1'')^{2^{w_1}-1-d_1+d_1}, \dots, f(a_m'')^{2^{w_1}-1-d_m+d_m}\} \\ &= \{f(a_1'')^{2^{w_1}-1}, \dots, f(a_m'')^{2^{w_1}-1}\} = B''. \end{aligned} \quad (6)$$

4 The Proposed Hash-based Butterfly Key Expansion

The PQCWC anonymous schemes proposed in Section 3 provides anonymity to other end entities but not to the CA. The CA can still identify the relationship between the expanded public key and the original public key. To achieve anonymity even from the CA, this study proposes a hash-based BKE mechanism (shown in Figure 3) built on the proposed PQCWC anonymous scheme. By performing one key expansion at the RA and another at the CA, this mechanism ensures anonymity from the RA, the CA, and other end entities. Below is a detailed explanation of the hash-based BKE mechanism and its proof of concept.

4.1 The Process of the Proposed Hash-based Butterfly Key Expansion

The hash-based BKE mechanism is primarily built on the proposed PQCWC anonymous scheme. Since PQCWC Anonymous Scheme 1 is a special case of PQCWC Anonymous Scheme 2, the following description uses PQCWC Anonymous Scheme 2 as an example.

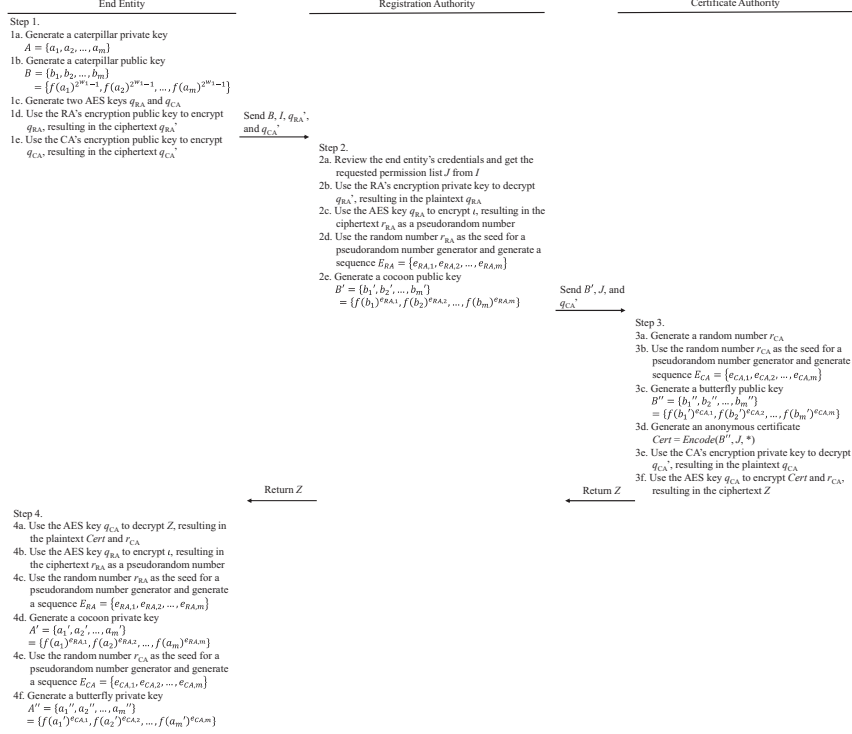


Figure 3 The proposed hash-based butterfly key expansion mechanism.

The proposed hash-based BKE mechanism is based on the following assumptions:

- Communication between the end entity, RA, and CA is established over a secure communication channel.
- The end entity and the RA share a commonly known time period ι .
- The end entity, RA, and CA share a commonly known pseudorandom number generator.
- The RA has an encryption key pair, and the end entity is aware of the RA's public encryption key.
- The CA has an encryption key pair, and the end entity is aware of the CA's public encryption key.

The process of the proposed hash-based BKE mechanism is illustrated in Figure 3. It involves the end entity generating the caterpillar key pair and AES keys, the RA generating the cocoon public key, the CA generating the

butterfly public key and anonymous certificate, and the end entity generating the butterfly private key.

4.1.1 End entity generating the caterpillar key pair and AES keys

The end entity generates a hash-based key pair for signing, which includes the caterpillar private key $A = \{a_1, a_2, \dots, a_m\}$ and the caterpillar public key $B = \{b_1, b_2, \dots, b_m\} = \{f(a_1)^{2^{w_1}-1}, f(a_2)^{2^{w_1}-1}, \dots, f(a_m)^{2^{w_1}-1}\}$, as defined in Section 2.1. Additionally, two AES keys, q_{RA} and q_{CA} , are generated. The AES key q_{RA} is encrypted using the registration authority's public encryption key, resulting in the ciphertext q'_{RA} , and the AES key q_{CA} is encrypted using the certificate authority's public encryption key, resulting in the ciphertext q'_{CA} . The end entity then packages the caterpillar public key B , the ciphertexts q'_{RA} and q'_{CA} , along with the relevant signing information I , into a certificate request packet and sends it to the registration authority.

4.1.2 RA generating the cocoon public key

After receiving the certificate request packet and verifying the end entity's eligibility, the RA retrieves the relevant information J from the signing request data I , which includes the permission list for the request. The RA then uses its private encryption key to decrypt the ciphertext q'_{RA} , obtaining the plaintext (i.e., the AES key q_{CA}). The RA encrypts the shared known time period ι , producing the ciphertext r_{RA} , and sets r_{RA} as the seed for a pseudorandom number generator. The RA generates a sequence of m pseudorandom elements $E_{RA} = \{e_{RA,1}, e_{RA,2}, \dots, e_{RA,m}\}$, where each pseudorandom element falls within the range $[0, 2^{w_2} - 1]$. The RA then expands the caterpillar public key B , producing the cocoon public key $B' = \{b'_1, b'_2, \dots, b'_m\} = \{f(b_1)^{e_{RA,1}}, f(b_2)^{e_{RA,2}}, \dots, f(b_m)^{e_{RA,m}}\}$, where $f(\cdot)$ is a hash function, and the value of b'_i is the result of performing $e_{RA,i}$ hash computations on b_i . The RA then sends the cocoon public key B' , the permission list J , and the ciphertext q'_{CA} to the CA.

4.1.3 CA generating the butterfly public key and anonymous certificate

The CA receives the packet and uses its private encryption key to decrypt the ciphertext q'_{CA} , obtaining the plaintext (i.e., the AES key q_{CA}). The CA generates a random number r_{CA} , which is set as the seed for a pseudorandom number generator. The CA then produces a sequence of m pseudorandom elements $E_{CA} = \{e_{CA,1}, e_{CA,2}, \dots, e_{CA,m}\}$, with each pseudorandom element falling within the range $[0, 2^{w_2} - 1]$. The CA expands the cocoon

public key B' , producing the butterfly public key $B'' = \{b''_1, b''_2, \dots, b''_m\} = \{f(b'_1)^{e_{CA,1}}, f(b'_2)^{e_{CA,2}}, \dots, f(b'_m)^{e_{CA,m}}\}$, where $f(\cdot)$ is a hash function, and the value of b''_i is the result of performing $e_{CA,i}$ hash computations on b'_i . The CA then generates an anonymous certificate $Cert$ for the end entity based on the butterfly public key B'' and the permission list J . Finally, the CA encrypts the anonymous certificate $Cert$ and the random number r_{CA} using the AES key q_{CA} to produce the ciphertext Z , which is sent back to the registration authority.

4.1.4 End entity generating the butterfly private key

After the end entity receives the ciphertext Z , it uses the AES key q_{CA} to decrypt and obtain the anonymous certificate $Cert$ and the random number r_{CA} . The end entity then uses the AES key q_{RA} to encrypt the commonly known time period ι , producing the ciphertext r_{RA} . This r_{RA} is set as the seed for a pseudorandom number generator, which generates a sequence of m pseudorandom elements $E_{RA} = \{e_{RA,1}, e_{RA,2}, \dots, e_{RA,m}\}$, with each element restricted to the range $[0, 2^{w_2} - 1]$. Next, the end entity expands the caterpillar private key A , producing the cocoon private key $A' = \{a'_1, a'_2, \dots, a'_m\} = \{f(a_1)^{e_{RA,1}}, f(a_2)^{e_{RA,2}}, \dots, f(a_m)^{e_{RA,m}}\}$, where $f(\cdot)$ is a hash function, and the value of a' is the result of performing $e_{RA,i}$ hash computations on a_i . Then, the end entity sets r_{CA} as the seed for a pseudorandom number generator, which produces a sequence of m pseudorandom elements $E_{CA} = \{e_{CA,1}, e_{CA,2}, \dots, e_{CA,m}\}$, again within the range $[0, 2^{w_2} - 1]$. The end entity then expands the cocoon private key A' to generate the butterfly private key $A'' = \{a''_1, a''_2, \dots, a''_m\} = \{f(a'_1)^{e_{CA,1}}, f(a'_2)^{e_{CA,2}}, \dots, f(a'_m)^{e_{CA,m}}\}$, where $f(\cdot)$ is a hash function, and the value of a''_i is the result of performing $e_{CA,i}$ hash computations on a'_i . Subsequently, the end entity can use the butterfly private key A'' to produce a signature, and other end entities can verify the signature using the butterfly public key B'' .

4.2 Proofs

The relationship between the butterfly public key and butterfly private key based on the hash butterfly key expansion can be proven using Equation (7). It can be observed that each butterfly private key element a''_i , after $2^{w_1} - 1$ hash computations, becomes the corresponding butterfly public key element b''_i . The relationship between each butterfly private key element and its corresponding butterfly public key element maintains a consistent $2^{w_1} - 1$ hash

computations. When generating a signature, the butterfly private key can be used to sign the data to be signed $D = \{d_1, d_2, \dots, d_m\}$, as shown in Equation (8). When verifying the signature, the butterfly public key can be used to verify the signature $S = \{s_1, \dots, s_m\}$. As shown in Equation (9), when the signature is correct, each verification element will match its corresponding public key element.

$$\begin{aligned}
B'' &= \{b_1'', \dots, b_m''\} \\
&= \{f(b_1')^{e_{CA,1}}, \dots, f(b_m')^{e_{CA,m}}\} \\
&= \{f(b_1)^{e_{CA,1}+e_{RA,1}}, \dots, f(b_m)^{e_{CA,m}+e_{RA,m}}\} \\
&= \{f(a_1)^{2^{w_1}-1+e_{CA,1}+e_{RA,1}}, \dots, f(a_m)^{2^{w_1}-1+e_{CA,m}+e_{RA,m}}\} \\
&= \{f(a_1')^{2^{w_1}-1+e_{CA,1}}, \dots, f(a_m')^{2^{w_1}-1+e_{CA,m}}\} \\
&= \{f(a_1'')^{2^{w_1}-1}, \dots, f(a_m'')^{2^{w_1}-1}\}. \tag{7}
\end{aligned}$$

$$\begin{aligned}
S &= \{s_1, \dots, s_m\} \\
&= \{f(a_1'')^{d_1}, \dots, f(a_m'')^{d_m}\}. \tag{8}
\end{aligned}$$

$$\begin{aligned}
V &= \{v_1, \dots, v_m\} \\
&= \{f(s_1)^{2^{w_1}-1-d_1}, \dots, f(s_m)^{2^{w_1}-1-d_m}\} \\
&= \{f(a_1'')^{2^{w_1}-1-d_1+d_1}, \dots, f(a_m'')^{2^{w_1}-1-d_m+d_m}\} \\
&= \{f(a_1'')^{2^{w_1}-1}, \dots, f(a_m'')^{2^{w_1}-1}\} = B''. \tag{9}
\end{aligned}$$

5 Experimental Results and Discussions

This section will compare the proposed method in terms of security and computational efficiency to verify its feasibility. First, the experimental environment will be described, followed by a security comparison with state-of-the-art (SOTA) methods and an evaluation of the computational efficiency of the proposed method using various hashing algorithms.

5.1 Experimental Environment

To practically verify the efficiency of the PQC anonymous schemes proposed in this study, a Windows 10 Enterprise Edition computer was used to simulate

the end entity, RA, and CA. The key generation time, key expansion time, signature generation time, and signature verification time were evaluated. The detailed specifications of the software and hardware used in the experiment include a CPU Intel(R) Core(TM) i7-10510U, 8 GB of memory, OpenJDK 18.0.2.1, and the Bouncy Castle Release 1.72 library. This study compares various hashing algorithms, including SHA-1, the SHA-2 family, the SHA-3 family, and the BLAKE family, demonstrating that the proposed anonymous schemes can achieve anonymity without increasing key length, signature length, key generation time, signature generation time, or signature verification time.

5.2 The Comparison of Security

In terms of security, the BKE mechanism in the IEEE 1609.2.1 standard provides anonymity, including anonymity from other end entities and CA [20, 21]. However, since the BKE mechanism in IEEE 1609.2.1 is primarily based on ECC, it may be vulnerable to quantum computing attacks and cannot achieve quantum-safe security levels. The schemes in [22, 23] are mainly built on hash-based cryptography and can achieve quantum-safe security level. However, these schemes include the original public key in the certificate, which can pose privacy risks, as shown in Table 1.

The proposed PQCWC anonymous schemes in this study are primarily based on hash-based cryptography, enabling it to achieve quantum-safe security levels while allowing for key expansion. This means that even without using the BKE mechanism, it can provide anonymity from other end entities. Furthermore, based on the PQCWC anonymous schemes, this study proposes a hash-based BKE mechanism, enabling anonymity from the RA and the CA and providing enhanced privacy protection.

Table 1 The comparison of security

Scheme	Quantum-Safe	Privacy Protection	
		Anonymous to Other End Entities	Anonymous to CA
[20]		x	x
[21]		x	x
[22]	x		
[23]	x		
The Proposed PQCWC Scheme 1	x	x	
The Proposed PQCWC Scheme 2	x	x	
The Proposed Hash-based BKE	x	x	x

5.3 The Comparison of Signature Lengths

In the experimental environment, this study assumes that the data to be signed, D , is of a fixed length of 256 bits (which could be the digest value after computing the original data using SHA-256). Each private key element a_i is also 256 bits long. A comparison is made between cases where the lengths of w_1 are 8 bits or 16 bits, with the signature lengths shown in Tables 2 and 3. The number of elements, m , in the private key sequence, public key sequence, and signature sequence primarily depends on the length of the data D and the length of w_1 . The hash algorithm used will determine the length of each element in the signature sequence, which, in turn, determines the total length of the signature sequence. For example, when w_1 is 8 bits, m is 32, and the hash value length for SHA-1 is 160 bits. Therefore, the signature sequence contains 32 hash values, each of 160 bits, resulting in a total length of 5120 bits (i.e., 640 bytes). As observed in Tables 2 and 3, the proposed method does not increase the signature length, achieving anonymity within the same length. Additionally, although increasing the w_1 length can reduce the signature length, the computation time will increase exponentially, which will be discussed in detail in subsequent sections.

It is worth mentioning that although the private key sequence consists of m elements, it is possible to store only a 256-bit random number as the seed for the pseudorandom number generator, which can then generate the m elements of the private key. Similarly, while the public key sequence also consists of m elements, the entire public key sequence can be hashed using SHA-256, reducing the public key length to 256 bits. Therefore, since both

Table 2 The comparison of signature lengths ($w_1 = 8$) (unit: bits)

Hash Algorithm	Hash Value Length	w_1	m	[22, 23]	The Proposed Method
SHA-1	160	8	32	5120	5120
SHA-224	224	8	32	7168	7168
SHA-256	256	8	32	8192	8192
SHA-384	384	8	32	12288	12288
SHA-512	512	8	32	16384	16384
SHA3-224	224	8	32	7168	7168
SHA3-256	256	8	32	8192	8192
SHA3-384	384	8	32	12288	12288
SHA3-512	512	8	32	16384	16384
BLAKE2-256	256	8	32	8192	8192
BLAKE2-384	384	8	32	12288	12288
BLAKE2-512	512	8	32	16384	16384

Table 3 The comparison of signature lengths ($w_1 = 16$) (unit: bits)

Hash Algorithm	Hash Value Length	w_1	m	[22, 23]	The Proposed Method
SHA-1	160	16	16	2560	2560
SHA-224	224	16	16	3584	3584
SHA-256	256	16	16	4096	4096
SHA-384	384	16	16	6144	6144
SHA-512	512	16	16	8192	8192
SHA3-224	224	16	16	3584	3584
SHA3-256	256	16	16	4096	4096
SHA3-384	384	16	16	6144	6144
SHA3-512	512	16	16	8192	8192
BLAKE2-256	256	16	16	4096	4096
BLAKE2-384	384	16	16	6144	6144
BLAKE2-512	512	16	16	8192	8192

the private and public keys in hash-based cryptography can be relatively short, this study does not delve deeply into the length of private and public keys.

5.4 The Comparison of Key Generation and Key Expansion

In this section, to compare the computation time for key generation and the computation time for the proposed PQCWC anonymous schemes (i.e., key expansion), different hash algorithms (i.e., the hash function $f(\cdot)$ mentioned in previous sections) are used to generate 1000 keys and 1000 expanded keys, respectively. For a fair comparison, the same values of w_1 and w_2 are used, and the results for individual elements are shown in Tables 4 and 5. From the experimental results, it can be observed that the key generation time mainly depends on $2^{w_1} - 1$ hash computations, while the key expansion time mainly depends on $2^{w_2} - 1$ hash computations. Therefore, when the values of w_1 and w_2 are the same, there is little difference between the key generation time and the key expansion time for the PQCWC Anonymous Scheme 1. Additionally, since the PQCWC Anonymous Scheme 2 uses random numbers in the range $[0, 2^{w_2} - 1]$ for expansion, the computation time is lower than that of key generation. To better visualize the differences between the various schemes, Figures 4 and 5 present box plots of the experimental results.

From Figures 4 and 5, it can be observed that the data distribution of key generation time and key expansion time for PQCWC Anonymous Scheme 1 is similar. However, for PQCWC Anonymous Scheme 2, both the average and median key expansion times are lower than the key generation time, though

Table 4 The comparison of key generation and key expansion ($w_1 = 8$) (unit: milliseconds)

Hash Algorithm	Key Generation	Key Expansion Based on PQCWC Scheme 1	Key Expansion Based on PQCWC Scheme 2
SHA-1	0.085	0.079	0.041
SHA-224	0.110	0.104	0.053
SHA-256	0.129	0.120	0.059
SHA-384	0.166	0.155	0.077
SHA-512	0.146	0.142	0.072
SHA3-224	0.250	0.241	0.122
SHA3-256	0.302	0.286	0.141
SHA3-384	0.220	0.212	0.111
SHA3-512	0.324	0.313	0.160
BLAKE2-256	0.130	0.121	0.060
BLAKE2-384	0.139	0.132	0.067
BLAKE2-512	0.161	0.154	0.073

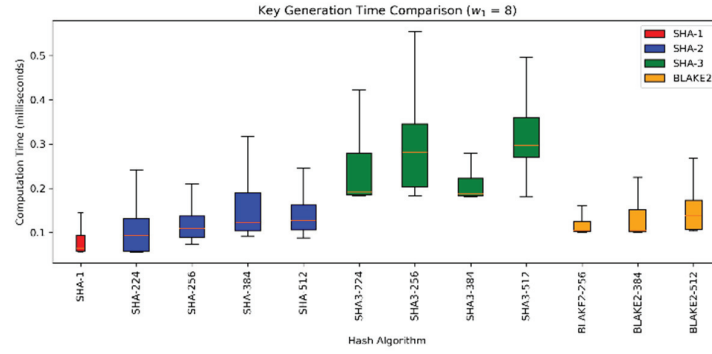
Table 5 The comparison of key generation and key expansion ($w_1 = 16$) (unit: milliseconds)

Hash Algorithm	Key Generation	Key Expansion Based on PQCWC Scheme 1	Key Expansion Based on PQCWC Scheme 2
SHA-1	15.049	14.902	7.365
SHA-224	15.764	15.608	7.727
SHA-256	15.540	15.379	7.756
SHA-384	18.579	18.470	8.902
SHA-512	19.138	19.368	9.600
SHA3-224	55.740	55.634	28.029
SHA3-256	51.622	51.787	26.001
SHA3-384	51.501	51.803	26.327
SHA3-512	51.961	52.158	26.054
BLAKE2-256	28.725	28.741	14.370
BLAKE2-384	28.464	28.368	14.290
BLAKE2-512	28.536	28.328	14.289

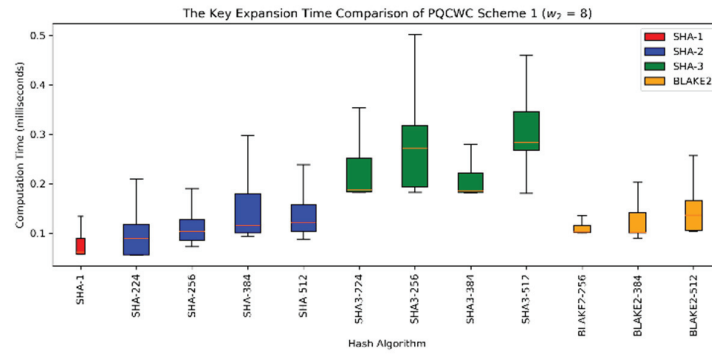
the range between the maximum and minimum values is wider. This is due to the random number being in the range of $[0, 2^{w_2} - 1]$, which causes a greater variation in computation time.

Furthermore, to objectively verify the significance of these differences, this study employs a statistical t -test to calculate pairwise comparisons between different schemes. Since there are 12 different algorithms tested, the

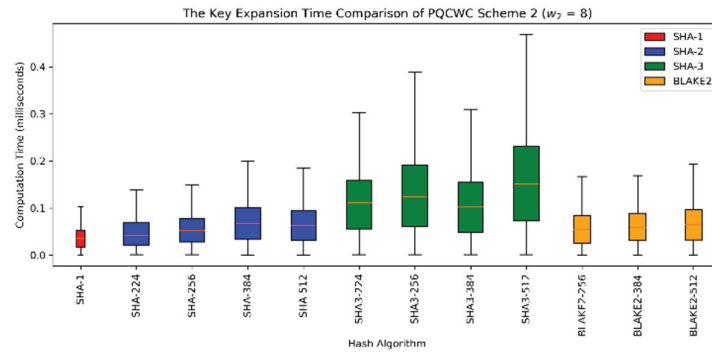
(a). Key generation time



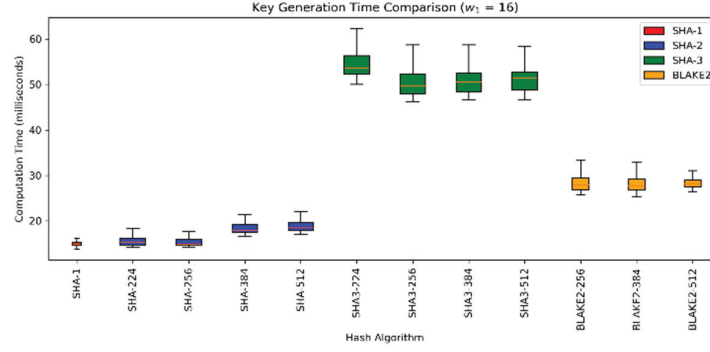
(b). Key expansion time by the proposed PQCWC Anonymous Scheme 1



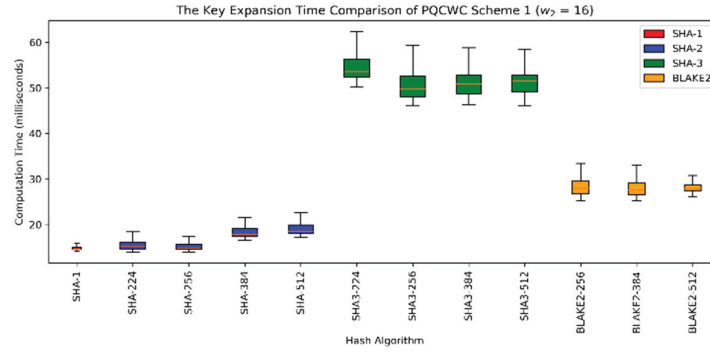
(c). Key expansion time by the proposed PQCWC Anonymous Scheme 2

**Figure 4** The comparison of key generation and key expansion ($w_1 = 8$) (Unit: milliseconds).

(a). Key generation time



(b). Key expansion time by the proposed PQCWC Anonymous Scheme 1



(c). Key expansion time by the proposed PQCWC Anonymous Scheme 2

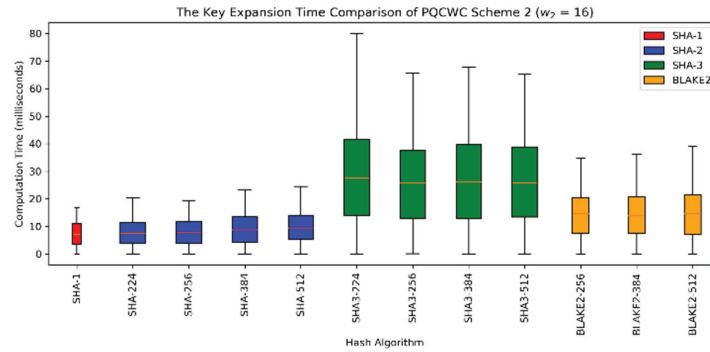


Figure 5 The comparison of key generation and key expansion ($w_1 = 16$) (Unit: milliseconds).

Table 6 The t -test results for the comparison of key generation and key expansion ($w_1 = 8$)

t -value	Key Generation	Key Expansion Based on PQCWC Scheme 1
Key Expansion Based on PQCWC Scheme 1	0.279	
Key Expansion Based on PQCWC Scheme 2	<u>3.807</u>	<u>3.545</u>

Table 7 The t -test results for the comparison of key generation and key expansion ($w_1 = 16$)

t -value	Key Generation	Key Expansion Based on PQCWC Scheme 1
Key Expansion Based on PQCWC Scheme 1	0.001	
Key Expansion Based on PQCWC Scheme 2	<u>2.992</u>	<u>2.977</u>

degree of freedom is 22, and when the t -value exceeds 2.074, it indicates a significant difference [26]. The results of the t -tests are shown in Tables 6 and 7. From the experimental results, it can be observed that there is no significant difference between the key generation time and the key expansion time for PQCWC Anonymous Scheme 1. However, there is a significant difference between the key expansion time for PQCWC Anonymous Scheme 2 and both the key generation time and the key expansion time for PQCWC Anonymous Scheme 1. Furthermore, as indicated by Tables 4 and 5, the key expansion time for PQCWC Anonymous Scheme 2 is significantly lower. Therefore, the proposed PQCWC anonymous credential scheme offers higher computational efficiency.

5.5 The Comparison of Signature Generation

In this section, to compare the computation time of generating signatures using the original private key with that of generating signatures using the proposed PQCWC anonymous schemes (i.e., expanded private key), different hash algorithms were used to generate 1,000 signatures. The experimental results are shown in Tables 8 and 9. The results indicate that regardless of whether the original private key or the expanded private key is used, the number of hash computations for each private key element depends on the value of the element d_i in the data D to be signed. Therefore, the computation times for generating signatures using the original private key, the expanded private key in PQCWC Anonymous Scheme 1, and the expanded private key in PQCWC Anonymous Scheme 2 are all similar. The t -test results, shown in Tables 10 and 11, indicate no significant differences across the computation times.

Table 8 The comparison of signature generation ($w_1 = 8$) (unit: milliseconds)

Hash Algorithm	[22, 23]	Signature Generation Based on PQCWC Scheme 1	Signature Generation Based on PQCWC Scheme 2
SHA-1	0.041	0.038	0.041
SHA-224	0.051	0.051	0.047
SHA-256	0.061	0.062	0.059
SHA-384	0.074	0.073	0.075
SHA-512	0.070	0.067	0.065
SHA3-224	0.128	0.119	0.119
SHA3-256	0.149	0.142	0.140
SHA3-384	0.108	0.112	0.109
SHA3-512	0.155	0.155	0.156
BLAKE2-256	0.060	0.058	0.058
BLAKE2-384	0.068	0.068	0.067
BLAKE2-512	0.077	0.075	0.074

Table 9 The comparison of signature generation ($w_1 = 16$) (unit: milliseconds)

Hash Algorithm	[22, 23]	Based on PQCWC Scheme 1	Based on PQCWC Scheme 2
SHA-1	7.580	7.576	7.520
SHA-224	7.832	7.872	7.851
SHA-256	7.499	7.502	7.484
SHA-384	9.234	9.217	9.182
SHA-512	9.504	9.459	9.482
SHA3-224	29.137	29.099	29.138
SHA3-256	25.430	25.474	25.420
SHA3-384	25.828	25.787	25.721
SHA3-512	26.247	26.149	26.110
BLAKE2-256	14.407	14.451	14.393
BLAKE2-384	14.393	14.408	14.381
BLAKE2-512	14.622	14.633	14.600

Table 10 The t -test results for the comparison of signature generation ($w_1 = 8$)

t-value	[22, 23]	Based on PQCWC Scheme 1
Based on PQCWC Scheme 1	0.119	
Based on PQCWC Scheme 2	0.182	0.064

Table 11 The t -test results for the comparison of signature generation ($w_1 = 16$)

t-value	[22, 23]	Based on PQCWC Scheme 1
Based on PQCWC Scheme 1	0.002	
Based on PQCWC Scheme 2	0.011	0.008

5.6 The Comparison of Signature Verification

In this section, to compare the computation time for signature verification using the original public key with that using the expanded public key in the proposed PQCWC anonymous credential scheme, different hash algorithms were used to verify the 1,000 signatures generated in the previous section. The experimental results are shown in Tables 12 and 13. The results indicate that, whether using the original public key or the expanded public key, the number of hash computations for each signature element is determined by $2^{w_1} - 1 - d_i$, where d_i is the value of the element in the data D to be signed.

Table 12 The comparison of signature verification ($w_1 = 8$) (unit: milliseconds)

Hash Algorithm	[22, 23]	Signature Generation Based on PQCWC Scheme 1	Signature Generation Based on PQCWC Scheme 2
SHA-1	0.040	0.042	0.039
SHA-224	0.051	0.051	0.050
SHA-256	0.060	0.057	0.057
SHA-384	0.077	0.076	0.073
SHA-512	0.070	0.069	0.068
SHA3-224	0.118	0.120	0.118
SHA3-256	0.145	0.137	0.141
SHA3-384	0.105	0.108	0.104
SHA3-512	0.160	0.157	0.155
BLAKE2-256	0.061	0.060	0.060
BLAKE2-384	0.067	0.068	0.066
BLAKE2-512	0.080	0.077	0.077

Table 13 The comparison of signature verification ($w_1 = 16$) (unit: milliseconds)

Hash Algorithm	[22, 23]	Based on PQCWC Scheme 1	Based on PQCWC Scheme 2
SHA-1	7.489	7.469	7.443
SHA-224	7.910	7.940	7.939
SHA-256	8.007	8.053	7.998
SHA-384	9.419	9.408	9.453
SHA-512	9.650	9.623	9.643
SHA3-224	26.767	26.578	26.517
SHA3-256	26.364	26.362	26.215
SHA3-384	25.914	25.881	25.771
SHA3-512	26.093	25.942	25.995
BLAKE2-256	14.563	14.593	14.453
BLAKE2-384	14.230	14.254	14.102
BLAKE2-512	14.120	13.954	13.913

Table 14 The t -test results for the comparison of signature verification ($w_1 = 8$)

t -value	[22, 23]	Based on PQCWC Scheme 1
Based on PQCWC Scheme 1	0.067	
Based on PQCWC Scheme 2	0.146	0.081

Table 15 The t -test results for the comparison of signature verification ($w_1 = 16$)

t -value	[22, 23]	Based on PQCWC Scheme 1
Based on PQCWC Scheme 1	0.012	
Based on PQCWC Scheme 2	0.027	0.016

Therefore, the computation times for verifying signatures using the original public key, the expanded public key in PQCWC Anonymous Scheme 1, and the expanded public key in PQCWC Anonymous Scheme 2 are similar. The t -test results, shown in Tables 14 and 15, indicate no significant differences in computation time.

6 Conclusions and Future Work

This study proposes the PQCWC anonymous schemes, which achieves quantum-safe security level while also providing anonymity to protect privacy. Furthermore, based on the PQCWC scheme, the study further proposes a hash-based butterfly key expansion mechanism to enable anonymity from the CA, fully safeguarding the privacy of end entities. In the experimental section, the study demonstrates that the proposed method shows no significant differences from the state-of-the-art in terms of signature length, key generation time, key expansion time, signature generation time, and signature verification time. It provides anonymity with no additional computational overhead.

Future research could explore extending the PQCWC anonymous schemes by integrating Merkle trees and combining it with the SLH-DSA algorithm [12] to enhance practicality.

Acknowledgments

Some contributions of the manuscript have been published as a preprint on the web pages of arXiv (i.e. <https://arxiv.org/abs/2410.03678>). This work acknowledges and appreciates that IEEE 1609.2, IEEE 1609.2.1, ETSI TS 103 097, ETSI TS 102 941, ETSI TR 104 239-1, and ETSI TR 104 171 have incorporated the author's relevant contributions and recommendations.

References

- [1] R. Ibarrondo, G. Gatti and M. Sanz, “Quantum Genetic Algorithm With Individuals in Multiple Registers,” in *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 3, pp. 788–797, 2024, doi:10.1109/TEVC.2023.3296780.
- [2] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” in *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997. DOI:10.1137/S0097539795293172.
- [3] K. Oonishi and N. Kunihiro, “Shor’s Algorithm Using Efficient Approximate Quantum Fourier Transform,” in *IEEE Transactions on Quantum Engineering*, vol. 4, pp. 1–16, 2023, Art no. 3102016, doi:10.1109/TQE.2023.3319044.
- [4] Y. Aono et al., “The Present and Future of Discrete Logarithm Problems on Noisy Quantum Computers,” in *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–21, 2022, Art no. 3102021, doi:10.1109/TQE.2022.3183385.
- [5] Y. Wang, H. Zhang and H. Wang, “Quantum polynomial-time fixed-point attack for RSA,” in *China Communications*, vol. 15, no. 2, pp. 25–32, 2018, doi:10.1109/CC.2018.8300269.
- [6] S. P. Jordan and Y.-K. Liu, “Quantum Cryptanalysis: Shor, Grover, and Beyond,” in *IEEE Security & Privacy*, vol. 16, no. 5, pp. 14–21, 2018, doi:10.1109/MSP.2018.3761719.
- [7] G. Alagic et al., “Status Report on the Third Round of the NIST Post-quantum Cryptography Standardization Process,” *NIST Special Publication*, NISTIR 8413, 2022. DOI:10.6028/NIST.IR.8413.
- [8] T. Tosun and E. Savas, “Zero-Value Filtering for Accelerating Non-Profiled Side-Channel Attack on Incomplete NTT-Based Implementations of Lattice-Based Cryptography,” in *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 3353–3365, 2024, doi:10.1109/TIFS.2024.3359890.
- [9] S. Suhail, R. Hussain, A. Khan and C. S. Hong, “On the Role of Hash-Based Signatures in Quantum-Safe Internet of Things: Current Solutions and Future Directions,” in *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 1–17, 2021, doi:10.1109/JIOT.2020.3013019.
- [10] Cohen, R. G. L. D’Oliveira, S. Salamatian and M. Médard, “Network Coding-Based Post-Quantum Cryptography,” in *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 49–64, 2021, doi:10.1109/JSAIT.2021.3054598.

- [11] J. Dey and R. Dutta, “Progress in Multivariate Cryptography: Systematic Review, Challenges, and Research Directions,” in *ACM Computing Surveys*, vol. 55, no. 12, Article No.: 246, 2023, doi:10.1145/3571071.
- [12] National Institute of Standards and Technology, “Stateless Hash-Based Digital Signature Standard,” *Federal Information Processing Standards Publication*, FIPS 205, 2024. DOI:10.6028/NIST.FIPS.205.
- [13] National Institute of Standards and Technology, “Module-Lattice-Based Digital Signature Standard,” *Federal Information Processing Standards Publication*, FIPS 204, 2024. DOI:10.6028/NIST.FIPS.204.
- [14] National Institute of Standards and Technology, “Module-Lattice-Based Key-Encapsulation Mechanism Standard,” *Federal Information Processing Standards Publication*, FIPS 203, 2024. DOI:10.6028/NIST.FIPS.203.
- [15] K.-A. Shim, “A Survey on Post-Quantum Public-Key Signature Schemes for Secure Vehicular Communications,” in *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14025–14042, 2022, doi:10.1109/TITS.2021.3131668.
- [16] H. Gharavi, J. Granjal and E. Monteiro, “Post-Quantum Blockchain Security for the Internet of Things: Survey and Research Directions,” in *IEEE Communications Surveys & Tutorials*, vol. 26, no. 3, pp. 1748–1774, 2024, doi:10.1109/COMST.2024.3355222.
- [17] Z. Yang, H. Alfauri, B. Farkiani, R. Jain, R. D. Pietro and A. Erbad, “A Survey and Comparison of Post-Quantum and Quantum Blockchains,” in *IEEE Communications Surveys & Tutorials*, vol. 26, no. 2, pp. 967–1002, 2024, doi:10.1109/COMST.2023.3325761.
- [18] C. H. Chen, “Post-Quantum Cryptography X.509 Certificate,” *2024 International Conference on Smart Systems for applications in Electrical Sciences (ICSSSES)*, Tumakuru, India, 2024, pp. 1–6, doi:10.1109/ICSSSES62373.2024.10561274.
- [19] C. H. Chen, “Comments on “Exploring Secure V2X Communication Networks for Human-Centric Security and Privacy in Smart Cities” – Privacy-Preserving Certificate in V2X Communications,” in *IEEE Access*, vol. 13, pp. 76923–76933, 2025, doi:10.1109/ACCESS.2025.3563461.
- [20] “IEEE Draft Standard for Wireless Access in Vehicular Environments (WAVE) – Certificate Management Interfaces for End Entities,” in *IEEE P1609.2.1/D2*, September 2025, pp. 1–259, 9 Oct. 2025. URL: <https://ieeexplore.ieee.org/servlet/opac?punumber=11201012>.
- [21] Hammi, J. P. Monteuuis, H. Labiod, R. Khatoun and A. Serhrouchni, “Using Butterfly Keys: A Performance Study of Pseudonym Certificates

- Requests in C-ITS,” *2017 1st Cyber Security in Networking Conference (CSNet)*, Rio de Janeiro, Brazil, 2017, pp. 1–6, doi:10.1109/CSNET.2017.8242002.
- [22] M. Honda and Y. Kaji, “Optimum Fingerprinting Function for Winternitz One-Time Signature,” *2024 IEEE International Symposium on Information Theory (ISIT)*, Athens, Greece, 2024, pp. 2898–2902, doi:10.1109/ISIT57864.2024.10619351.
- [23] M. Honda and Y. Kaji, “Extended Version—to Be, or Not to Be Stateful: Post-Quantum Secure Boot Using Hash-based Signatures,” *Journal of Cryptographic Engineering*, vol. 14, pp. 631–648, 2024, doi:10.1007/s13389-024-00362-4.
- [24] S. Khan et al., “A Survey on X.509 Public-Key Infrastructure, Certificate Revocation, and Their Modern Implementation on Blockchain and Ledger Technologies,” in *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2529–2568, 2023, doi:10.1109/COMST.2023.3323640.
- [25] P. Kurariya, A. Bhargava, S. Sailada, N. Subramanian, J. Bodhankar and A. Kumar, “Experimentation on Usage of PQC Algorithms for eSign,” *2022 IEEE International Conference on Public Key Infrastructure and its Applications (PKIA)*, Bangalore, India, 2022, pp. 1–6, doi:10.1109/PKIA56009.2022.9952354.
- [26] C. H. Chen, G. Liu, Y. C. Wei, Z. Li and B. Y. Lin, “A Pre-signed Response Method Based on Online Certificate Status Protocol Request Prediction,” in *Enterprise Information Systems*, vol. 16, no. 8–9, 2022, doi:10.1080/17517575.2021.1986861.

Biography



Abel C. H. Chen (Senior Member, IEEE) has published over 400 journal articles, conference papers, and patents. His contributions were published in *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Transactions on Emerging Topics in Computational Intelligence*, *IEEE Internet of Things Journal*, *ACM Transactions on Sensor Networks*, *Information Science*, *IEEE Communications Letters*, and *Physica A: Statistical Mechanics and its Applications*. He has also submitted numerous contributions to more than 100 standards, including IEEE 1609.2, IEEE 1609.2.1, ETSI TS 102 941, ETSI TR 104 171, and ETSI GR QKD 007. He served as the Chair for several conferences, such as AAAI-22 Workshop, WWW 2021 Workshop, IEEE BIBM 2021 Workshop, IEEE TrustCom 2021 Workshop, IEEE APNOMS 2020, and IEEE ICC 2020. He serves as an Editor for several journals, such as *Scientific Data*, *IEEE Open Journal of Intelligent Transportation Systems*, and *Network: Computation in Neural Systems*. He served as an Associate Editor or the Guest Editor for several journals, such as *IEEE Access*, *IEICE Transactions on Information and Systems*, *Journal of Applied Statistics*, and *ISPRS International Journal of Geo-Information*. He was listed among the Top 2% Scientists Worldwide, in 2022, 2023, 2024, and 2025 by Stanford University. Some of his publications have been recognized as highly cited papers on Web of Science using data from Essential Science Indicators (ESI).

