# Reinforcing Security and Usability of Crypto-Wallet with Post-Quantum Cryptography and Zero-Knowledge Proof

Yathin Kethepalli [*], Rony Joseph [*], Sai Raja Vajrala [*], Jashwanth Vemula [*], and Nenavath Srinivas Naik [†]
[*]*IIIT Naya Raipur,* [†]*IIITDM Kurnool*

*Abstract*—**Crypto-wallets or digital asset wallets are a crucial aspect of managing cryptocurrencies and other digital assets such as NFTs. However, these wallets are not immune to security threats, particularly from the growing risk of quantum computing. The use of traditional public-key cryptography systems in digital asset wallets makes them vulnerable to attacks from quantum computers, which may increase in the future. Moreover, current digital wallets require users to keep track of seed-phrases, which can be challenging and lead to additional security risks. To overcome these challenges, a new algorithm is proposed that uses post-quantum cryptography (PQC) and zero-knowledge proof (ZKP) to enhance the security of digital asset wallets. The research focuses on the use of the Lattice-based Threshold Secret Sharing Scheme (LTSSS), Kyber Algorithm for key generation and ZKP for wallet unlocking, providing a more secure and user-friendly alternative to seed-phrase, brain and multi-sig protocol wallets. This algorithm also includes several innovative security features such as recovery of wallets in case of downtime of the server, and the ability to rekey the private key associated with a specific username-password combination, offering improved security and usability. The incorporation of PQC and ZKP provides a robust and comprehensive framework for securing digital assets in the present and future. This research aims to address the security challenges faced by digital asset wallets and proposes practical solutions to ensure their safety in the era of quantum computing.**

## 1. Introduction

Digital asset wallets have emerged as an indispensable tool for the managing cryptocurrencies. These are designed to store and manage various digital assets, including but not limited to Bitcoin, Ethereum, and other cryptocurrencies. Digital asset wallets have revolutionized the traditional concept of wallets by providing a secure and convenient way to manage digital assets, thus enabling users to engage in various financial activities. The origin of digital asset wallets can be traced back to the introduction of Bitcoin, the first-ever cryptocurrency that was introduced in 2009 [Nakamoto(2009)]. The advent of Bitcoin led to the introduction of the blockchain technology, which allowed for the storage of transactions on a public ledger. These wallets are designed to store private keys required for accessing the blockchain and conducting transactions. In today's digital age, they serve a variety of purposes, including but not limited to, sending and receiving cryptocurrency, managing portfolios, investing in blockchain projects, and as a means of payment. They have gained immense popularity due to their security features, which are far more advanced than traditional wallets. Most of these wallets incorporate multi-signature authentication, and cold storage mechanisms to protect the user's funds from any malicious attacks.

Various types of digital asset wallets are available today, each with its own set of features and security protocols. These include software, hardware, paper, brain and smart contract wallets [Erinle et al.(2023)].

- **Software Wallets:** These wallets are installed on a user's computer or mobile device or accessed through a web browser. They are generally easy to use, but are more susceptible to online hacks and attacks.
- **Hardware Wallets:** These are physical devices that are used to store digital assets offline. They are generally more secure than software wallets, but can be more difficult to use.
- **Paper Wallets:** These are physical pieces of paper that contain a user's private key. They are often used as a backup for software or hardware wallets.
- **Brain Wallets:** Wallets created by deriving a private key from a passphrase chosen by the user. They can be accessed online from any device but are less secure due to vulnerability to brute-force and dictionary attacks.
- **Smart contract Wallets:** These wallets utilize smart contracts to securely manage assets and provide advanced customization features. These wallets offer enhanced security and enable functionalities like recoverable wallets, signless transactions, and batched transactions, surpassing the capabilities of traditional crypto wallets.

Digital asset wallets have emerged as a critical component in the rapidly evolving world of digital finance. Cryptography has become an essential component of these wallets, providing a secure means of storing and transferring digital assets. Mathematical algorithms form the backbone of modern cryptography, ensuring that these wallets remain secure and accessible only to those with the appropriate keys. Cryptography plays a crucial role in securing digital asset wallets by converting sensitive information into an unreadable format that can only be deciphered using the correct decryption keys. The importance of cryptography in digital asset wallets cannot be overstated, as it protects the user's funds from theft and malicious attacks. The use of cryptographic techniques, such as public-key cryptography and hash functions ensures that digital assets are stored and transferred securely, without the need for a central authority. The user's private keys are used to ensure that only authorized individuals have access to their digital assets.

The advent of quantum computing presents a formidable obstacle to modern cryptographic systems, which include digital asset wallets. While classical computers rely on binary digits or bits to represent information, quantum computers leverage the principles of quantum mechanics to use quantum bits or qubits. These qubits can exist simultaneously in multiple states, allowing quantum computers to perform calculations in parallel and potentially lead to exponential speedups for specific problems. This new technology has the potential to revolutionize multiple fields, such as cryptography and security. However, one of the primary implications of quantum computing for security lies in the breaking of cryptographic systems that rely on the complexity of specific mathematical problems, such as factoring large numbers or computing discrete logarithms. Most widely used cryptographic protocols, such as RSA and ECC, are based on these problems' difficulty and can be compromised by Shor's algorithm on a sufficiently powerful quantum computer [Shor(1997)], [Bernstein(2009b)]. Consequently, it is widely believed that numerous cryptographic systems currently in use will become vulnerable once large-scale quantum computers are available. Thus, the rise of quantum computing has introduced significant security concerns that require immediate attention. As researchers continue to explore new ways to counter the threat, it is essential to acknowledge the potential limitations of current cryptographic systems and plan for more secure alternatives.

Post-Quantum Cryptography (PQC) focuses on the design and analysis of cryptographic protocols that are resistant to attacks by classical and quantum computers. The primary objective is to devise cryptographic systems that can withstand attacks by quantum computers, which are expected to become increasingly powerful in the future. As the development of PQC is still in its nascent stages, several cryptographic schemes are presently being evaluated for their effectiveness against quantum attacks.

One of the cryptographic concepts that have gained considerable attention in recent times is zero-knowledge proof (ZKP). It is a technique that enables one party to prove to another that a given statement is true, without revealing any additional information beyond the veracity of the statement. In other words, ZKP enables authentication without the need to disclose any private information.

Our proposed methodology aims to address the security concerns associated with traditional public-key cryptography systems and seed-phrases in digital asset wallets by employing PQC and ZKP. It includes several innovative security features that enhance the security of digital asset wallets and makes them easier to use. One such feature is recovery of wallets in case of server downtime, which allows users to recover access to their wallets even if the server isn't accessible for any reason. This approach is more secure and less cumbersome than traditional recovery mechanisms that rely on email or phone verification. Another feature is the ability to rekey the private key associated with a specific username-password combination, which provides an additional layer of security in case a user's private key is compromised.

The rest of this paper is organized as follows: Section 2 presents related works on crypto wallets, Section 3 provides the necessary background information for understanding our proposed methodology. In Section 4, we provide a detailed explanation of our proposed methodology, which employs PQC and ZKP to enhance the security of digital asset wallets. We also discuss the innovative security features of our approach, including client-side private key derivation, social recovery, and rekeying. Section 5 presents the results of our experimental testing, which demonstrates the effectiveness of our approach in improving digital asset wallet security. We consider the potential impact of our approach on the adoption of digital wallets and the broader digital finance industry. We also discuss the limitations of our approach and the challenges that remain to be addressed. Section 6 discusses the broader implications of our work for the future of digital asset wallet security. Finally, in Section 7, we conclude the paper and discuss directions for future research.

## 2. Related Works

Over the years, various research studies have been conducted to investigate diverse aspects of digital asset wallets. These studies have examined both general and specific aspects of the subject, highlighting the multidimensional nature of research in this domain. The focus of these studies can be broadly classified into three categories: key management, wallet design, and wallet security.

### 2.1. Key Management

The security of digital asset wallets relies heavily on robust key management practices, which have been extensively studied in the literature [Mangipudi et al.(2022)], [He et al.(2018)], [Pal et al.(2021)], [He et al.(2019)], [Courtois and Mercer(2017)], [Belchior et al.(2022)], [Chen et al.(2020a)], [Suratkar et al.(2020)].

Key management encompasses a range of methods for securely handling and storing private keys, which can be implemented across various digital and physical applications and distributed among multiple devices. Mangipudi et al. [Mangipudi et al.(2022)] categorized wallets into single-device and multi-device wallets based on their ability to distribute the risk associated with private key compromise. Expanding on this notion, Pal et al. [Pal et al.(2021)] explored additional avenues for key management, such as cloud-based key storage.

Furthermore, researchers have delved into key management approaches specific to individual assets [Eskandari et al.(2018)], specialized hardware devices like Hardware Security Modules (HSMs) [Götte and Scheuermann(2021)], [Shbair et al.(2021)], and innovative key recovery methods [He et al.(2019)]. These studies underscore the significance of effective key management in crypto-wallets and highlight diverse strategies for safeguarding private keys.

## 2.2. Wallet Design

The design of cryptocurrency wallets has garnered significant attention in research, particularly in terms of wallet architecture. Khadzhi et al. [Khadzhi et al.(2020)] conducted an analysis of transaction activity on hardware wallets, shedding light on their usage patterns. Khan et al. [Khan et al.(2019)] proposed a novel architectural design that leverages QR codes to authenticate transactions between hardware and software wallets. Eyal [Eyal(2022)] offered valuable insights into wallet design by assessing the failure probability of wallets from a security standpoint. Furthermore, specific studies have explored the design characteristics of Ethereum wallets [Di Angelo and Salzer(2020)], [Homoliak et al.(2018)].

A notable architectural concept highlighted in the literature is the main and sub-wallet setup, akin to a savings and current account structure for blockchain wallets. Rezaeighaleh and Zou [Rezaeighaleh and Zou(2019)] discussed the concept of sub-wallets and main-wallets within deterministic wallets and developed a prototype implementation in a hardware wallet. This mechanism was found to be resistant to classical wallet attacks, including Man-In-The-Middle attacks, where malicious actors attempt to substitute a user's address with a malicious one.

These studies underscore the importance of thoughtful architectural considerations in cryptocurrency wallet design, paving the way for innovative solutions that enhance security and user experience.

## 2.3. Wallet Security

The security and vulnerability of blockchain systems have been extensively examined in the literature [Chen et al.(2020b)], [Li et al.(2020)], [Guo and Yu(2022)], [Zamani et al.(2020)], [Urien(2021)]. Notably, Homoliak et al. [Homoliak et al.(2018)] conducted a comprehensive analysis of attacks on crypto wallet applications and proposed defense mechanisms to mitigate vulnerabilities. Their study emphasizes the adoption of air-gapped hardware wallet solutions as an effective defense measure.

Addressing server-side vulnerabilities, Bui et al. [Bui et al.(2019)] specifically focus on wallet attack scenarios and propose defense mechanisms, particularly against man-in-the-middle attacks. Another study explores prevention mechanisms for phishing attacks, a prevalent type of wallet attack [Andryukhin(2019)]. Meanwhile, Chen et al. [Chen et al.(2020b)] delve into the vulnerabilities, attacks, and security mechanisms specific to the Ethereum blockchain. Their research contributes to the existing body of knowledge by shedding light on vulnerabilities and attacks across different layers of the blockchain, with a particular focus on wallet-related threats in the application layer.

These studies collectively contribute to enhancing wallet security in blockchain systems by identifying vulnerabilities, proposing defense mechanisms, and highlighting the importance of robust security practices throughout the ecosystem.

## 3. Technical Background

### 3.1. Post-Quantum Cryptography

PQC pertains to cryptographic techniques, primarily public-key algorithms, which are conjectured to resist cryptanalytic attacks by quantum computers. The imminent threat posed by quantum computing towards traditional cryptographic protocols, which depend on the computational complexity of certain mathematical problems, most notably, integer factorization, the discrete logarithm, and the elliptic-curve discrete logarithm problems, have engendered the necessity for the development of PQC.

The significance of PQC in ensuring secure communication is paramount in light of recent advancements in quantum computing technology. The development and deployment of PQC systems and protocols, which are both robust and efficient, is an active area of research in the field of cryptography. It is imperative to ensure that the cryptographic protocols of the future are resilient against the potent attacks of quantum computers, and PQC stands at the forefront of this endeavor.

TABLE 1. Comparison of Key Sizes, Ciphertext, and Signature Sizes for Various Post-Quantum Cryptography Algorithms at a 128-bit Security Level

| Algorithm | Type | Public Key | Private Key | signature |
|---|---|---|---|---|
| NTRU Encrypt | Lattice | 766.25 B | 842.875 B | - |
| Streamlined NTRU Prime | Lattice | 154 B | - | - |
| BLISS-II | Lattice | 7 KB | 2 KB | 5 KB |
| Rainbow | Multivariate | 124 KB | 95 KB | - |
| SPHINCS | Hash Signature | 1 KB | 1 KB | 41 KB |
| SPHINCS+ | Hash Signature | 32 B | 64 B | 8 KB |
| GLP-Variant GLYPH Signature | Ring-LWE | 2 KB | 0.4 KB | 1.8 KB |
| NewHope | Ring-LWE | 2 KB | 2 KB | - |
| Goppa-based McEliece | Code-based | 1 MB | 11.5 KB | - |
| Quasi-cyclic MDPC-based McEliece | Code-based | 1,232 B | 2,464 B | - |
| Random Linear Code based encryption | RLCE | 115 KB | 3 KB | - |
| SIDH | Isogeny | 564 B | 48 B | - |
| SIDH (compressed keys) | Isogeny | 330 B | 48 B | - |
| 3072-bit Discrete Log | not PQC | 384 B | 32 B | 96 B |
| 256-bit Elliptic Curve | not PQC | 32 B | 32 B | 65 B |

A ubiquitous trait of numerous PQC algorithms is their require larger key sizes than the conventional pre-quantum public key algorithms. Such algorithms present a conundrum in key size, computational efficiency, and ciphertext or signature size, warranting careful consideration of the trade-offs between these factors. The Table 1 presents key sizes, ciphertext and signature sizes for various schemes operating at a post-quantum security level of 128 bits, thereby enabling an assessment of these trade-offs. Currently, PQC research is mostly focused on six different approaches using lattices, multivariate equations, hashes, codes, supersingular elliptic curve isogeny and symmetric keys [Gershon(2013)], [Spectrum(2008)].

**3.1.1. Lattice-based cryptography.** This approach involves the use of systems such as learning with errors, ring learning with errors (ring-LWE) [Bernstein(2009a)], [Bernstein(2010)], [Peikert(2014)], ring learning with errors key exchange, and ring learning with errors signature. It also includes the older NTRU or GGH encryption schemes and the newer NTRU signature and BLISS signatures [Güneysu et al.(2012)]. Notably, some schemes like NTRU encryption have been subjected to extensive studies without yielding any feasible attack. In contrast, others like the ring-LWE algorithms have security proofs that reduce to a worst-case problem [Zhang et al.(2015)]. The PQC Study Group sponsored by the European Commission has recommended the Stehle–Steinfeld variant of NTRU for standardization instead of the NTRU algorithm because it has a security reduction, whereas the latter is still patented [Ducas et al.(2013)], [Lyubashevsky et al.(2013)]. Studies have indicated NTRU may have more secure properties than other lattice-based algorithms [Augot et al.(2015)]. Ring-LWE, a specific version of Ring-LWE signatures, has security reduction to the shortest-vector problem (SVP) in a lattice known to be NP-hard. Furthermore, the security of the NTRU encryption scheme and the BLISS signature is believed to be related to the Closest Vector Problem (CVP) in a Lattice, also known as NP-hard. However, it cannot be provably reducible.

## 3.2. Zero-Knowledge Proofs

ZKPs are cryptographic protocols that enable one party (the prover) to convince another party (the verifier) that a statement is true, without revealing any additional information about the statement or any secrets used to prove it. The fascinating aspect of ZKPs is that it is trivial to demonstrate possession of certain knowledge by merely revealing it [Lixie(2021)]. However, the challenge lies in proving possession of such knowledge without disclosing the knowledge itself or any other related information. In this sense, they aim to achieve high privacy and confidentiality by not exposing sensitive data beyond the necessary proof of knowledge. These proofs can be interactive, where both parties engage in a back-and-forth conversation, or non-interactive, where a single message generates the proof, commonly relying on the Fiat-Shamir heuristic [Blum et al.(1988)]. Nonetheless, the validity of the proof still relies on computational assumptions and cryptographic primitives [Wu and Wang(2014)], [Quisquater et al.(1990)]. ZKPs have gained tremendous attention due to their potential applications in many fields, including cryptocurrency, blockchain, and cybersecurity.

The three main properties of a ZKP are as follows:

- **Completeness** requires that if the statement is true, then a verifier will be convinced of its truth by an honest prover.
- **Soundness** requires that if the statement is false, no cheating prover can convince an honest verifier that it is true, except with some small probability.
- **Zero-knowledge** requires that if the statement is true, no verifier can learn anything other than the fact that the statement is true. This is formalized by showing that every verifier has some simulator that can produce a transcript that “looks like” an interaction between an honest prover and the verifier in question.

The initial two properties are applicable to a broader class of interactive proof systems. However, it is the third property that uniquely characterizes a ZKP. This distinctive quality ensures that the verifier gains no additional knowledge beyond the fact that the statement is true, even though the prover provides convincing evidence of their knowledge of the secret. This feature sets ZKPs apart from other interactive proof systems [Chaum et al.(1988)].

Different variants of ZKPs can be defined based on how closely the distributions produced by the simulator and the real proof protocol match. Perfect ZKPs require that the distributions are the same, while statistical ZKPs allow for some statistical difference between the two distributions. Computational ZKPs go even further by requiring no efficient algorithm to distinguish between the two distributions. These variants are important because they allow for different levels of security and efficiency depending on the specific use case.

## 3.3. Kyber Algorithm

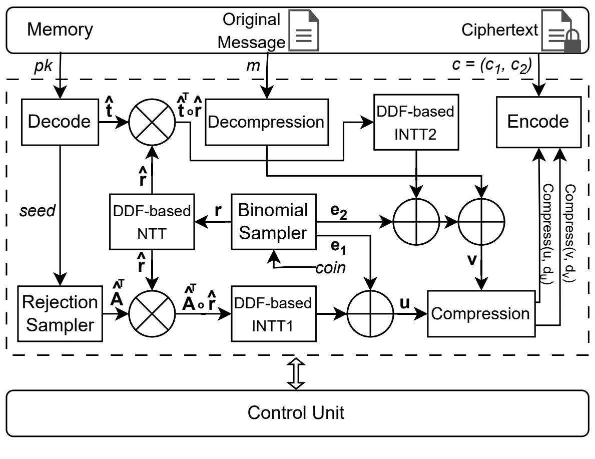

Figure 1. Working mechanism of Kyber algorithm

The Kyber algorithm is a sophisticated lattice-based cryptographic scheme that leverages the learning with errors (LWE) problem to establish a secure communication channel between multiple parties. The LWE problem entails detecting a concealed vector within a noisy vector, which is a challenging mathematical problem, and uses module LWE with cyclotomic rings as its trapdoor function. The Kyber algorithm generates both a public key and a private key, which are instrumental in facilitating the encryption and decryption processes, respectively, as shown in Figure 1. The construction of these keys is grounded on matrices and polynomials derived from lattices and geometric structures utilized to create complex mathematical problems.

The Kyber algorithm crafts the public key by generating a secret vector with coefficients from a finite field. The next step is to develop a matrix with random coefficients and multiply it by the secret vector. The resulting matrix undergoes rounding to the closest multiple of a certain number, ultimately constituting the public key. The public key is essentially the fundamental matrix constructed from the lattice basis. To form the private key, the Kyber algorithm crafts a matrix that comprises random coefficients and multiplies it by the public key. The resulting matrix is rounded to the nearest multiple of a certain number, which produces the private key. Like the public key, the private key is a fundamental matrix constructed from the same lattice basis as the public key matrix.

The Kyber family of cryptographic algorithms delivers varying degrees of security, with Kyber-512 presenting a level comparable to AES-128, Kyber-768 having a security level equivalent to AES-192, and Kyber-1024 providing a security level that is comparable to AES-256. The security of the Kyber algorithm rests on the hardness of the LWE problem and the utilization of lattices in constructing the keys and ciphertext.

## 3.4. SHA256

Secure Hash Algorithms (SHA) are a family of cryptographic hash functions created by the United States National Security Agency (NSA). While there are other variants such as SHA 224 and SHA 384, SHA 256 has been at the forefront of real-world applications. SHA256 function produces a 256-bit value, typically a 64-digit hexadecimal number using 64 rounds of mixing and compression as shown in Algorithm 1. It takes an input message of any length and produces a fixed-length output hash.

SHA 256 is widely utilized in various applications where data integrity and security are paramount. Its significance extends to digital signature algorithms, message authentication codes, and notably, blockchain networks such as Bitcoin, Ethereum. It plays a fundamental role in ensuring immutability and tamper resistance of data within blockchain systems. It accomplishes this by generating unique hash values for each block in the blockchain, securing transactions and maintaining the integrity of the entire chain.

The working of SHA256 involves the following steps:

- **Padding:** The input message is padded to make its length a multiple of 512 bits. Padding is added so the resulting padded message length is congruent to 448 modulo 512. The last 64 bits are used to store the original message length.

**Algorithm 1** SHA256 function

```
function SHA256($message$)
    $h_0 \leftarrow \mathit{0x6a09e667}$, $h_1 \leftarrow \mathit{0xbb67ae85}$, $h_2 \leftarrow \mathit{0x3c6ef372}$,
    $h_3 \leftarrow \mathit{0xa54ff53a}$, $h_4 \leftarrow \mathit{0x510e527f}$, $h_5 \leftarrow \mathit{0x9b05688c}$,
    $h_6 \leftarrow \mathit{0x1f83d9ab}$, $h_7 \leftarrow \mathit{0x5be0cd19}$
    $k_i \leftarrow \lfloor 2^{32} \times |\sin i| \rfloor$ for $i$ in 0 to 63
    $a \leftarrow h_0$, $b \leftarrow h_1$, $c \leftarrow h_2$, $d \leftarrow h_3$,
    $e \leftarrow h_4$, $f \leftarrow h_5$, $g \leftarrow h_6$, $h \leftarrow h_7$
    for $i$ in 1 to $N$ do
        if $0 \leq j < 16$ then
            $M_{i,j}$
        else
            $\sigma_1(w_{j-2}) + w_{j-7} + \sigma_0(w_{j-15}) + w_{j-16}$
        end if
        for $j$ in 0 to 63 do
            $T_1 \leftarrow h + \Sigma_1(e) + \mathcal{C}\langle(e, f, g) + k_j + w_j$
            $T_2 \leftarrow \Sigma_0(a) + \mathcal{M}\dashv|(a, b, c)$
            $h \leftarrow g$, $g \leftarrow f$, $f \leftarrow e$,
            $e \leftarrow d + T_1$,
            $d \leftarrow c$, $c \leftarrow b$, $b \leftarrow a$,
            $a \leftarrow T_1 + T_2$
        end for
        $h_0 \leftarrow h_0 + a$, $h_1 \leftarrow h_1 + b$, $h_2 \leftarrow h_2 + c$, $h_3 \leftarrow h_3 + d$,
        $h_4 \leftarrow h_4 + e$, $h_5 \leftarrow h_5 + f$, $h_6 \leftarrow h_6 + g$, $h_7 \leftarrow h_7 + h$
    end for
    $hash \leftarrow$ concatenate $h_0$, $h_1$, $h_2$, $h_3$, $h_4$, $h_5$, $h_6$, $h_7$
    return $hash$
end function
```

- **Message schedule:** The padded message is divided into 512-bit blocks, and a message schedule is created for each block. The message schedule consists of 64 32-bit words.
- **Initial hash values:** The initial hash values are predefined for each round of the SHA256 algorithm. There are eight hash values, each represented by a 32-bit word.
- **Compression function:** The compression function is applied to each message schedule and the initial hash values to produce a new set of hash values. The compression function involves several rounds of logical operations and also uses a set of pre-defined constant values.
- **Final hash:** After all the blocks have been processed, the final hash value is produced by concatenating the eight 32-bit hash values produced in the compression function.

## 3.5. BCrypt

BCrypt is a widely used password-hashing function based on the Blowfish cipher. It provides a strong level of security and is known for its adaptability and resistance to brute-force attacks. It is designed to be computationally expensive, making it time-consuming and resource-intensive for attackers attempting to crack hashed passwords. One of the key features of BCrypt is its adaptive nature. It adjusts the computation complexity based on a configurable iteration count, known as the cost parameter. By increasing the cost, the hash function requires more time and computational resources to compute the hash value, effectively slowing down potential attackers. This adaptive nature allows BCrypt to remain resilient against evolving hardware capabilities and increases the security of hashed passwords over time. To further enhance security, it incorporates a salt value into the hashing process. The salt is a randomly generated string that is combined with the password before hashing. This ensures that even if two users have the same password, their hash values will be different due to the unique salt values. The inclusion of salt protects against rainbow table attacks, where an attacker precomputes the hash values of commonly used passwords to quickly identify matches.

**Algorithm 2** BCrypt function

```
function BCRYPT($cost, salt, message$)
    $P, S \leftarrow$ EKSBLOWFISHSETUP($cost, salt, message$)
    $ctext \leftarrow$ "Any Input Message"
    for $i \leftarrow 0$ to $63$ do
        $ctext \leftarrow$ ENCRYPTECB($P, S, ctext$)
    end for
    $hash \leftarrow$ CONCATENATE($cost, salt, ctext$)
    return $hash$
end function
```

It has gained significant popularity and has been implemented in various programming languages and frameworks, including C, C++, Java, JavaScript, Python, and Ruby. It is the default password hash algorithm for OpenBSD, a highly secure operating system. The input to the bcrypt function includes the message string (up to 72 bytes), a numeric cost parameter, and a 16-byte salt value as shown in Algorithm 2. The cost parameter determines the number of iterations used in the hash value computation. It can be adjusted based on the desired level of security and the available computational resources. The function uses these inputs to compute a 24-byte (192-bit) hash. The final output will be a string as shown in Figure 2.

```
$2<a/b/x/y>$[cost]$[22 character salt][31 character hash]
```

Figure 2. Final output of BCrypt function

### 3.6. Lattice-based Threshold Secret Sharing

A secret sharing scheme is a technique used to distribute a secret among multiple participants by providing them with shares. A dealer distributes the shares through a secure channel to ensure that only authorized subsets of participants can recover the secret. A public key cryptosystem can represent the secure channel. Lattice-based cryptography is a promising area for designing new public key cryptosystems that can resist quantum computers [Goldreich et al.(1997)], [Hoffstein et al.(1998)], [Regev(2009)]. Lattice-based Threshold Secret Sharing Scheme (LTSSS) is a variation of method used by Steinfeld et al. [Steinfeld et al.(2004)].

The LTSSS consists of three phases: public parameter generation, share generation, and secret reconstruction.

In the share generation phase, the dealer selects $n$ distinct $\mathbf{l}^{(i)}$ vectors of dimension $m$ uniformly at random, and an $m$-dimensional vector $\mathbf{a}$ is chosen such that its first component is assigned to the secret and the remaining $(m-1)$ components are assigned random values. Then, the shares are computed by adding some noise $\epsilon_i$ to the inner product of $\mathbf{l}^{(i)}$ and $\mathbf{a}$, for each $i \in 1, 2, \ldots, n$.

In the secret reconstruction phase, a combiner (server) generates a $(t+m)$-dimensional lattice basis $\mathbf{M}$ by exploiting $t$ out of $n$ vectors $\mathbf{l}^{(i)}$, and a $(t+m)$-dimensional vector $\mathbf{t}'$ is generated using $t$ out of $n$ shares, which is close to a certain lattice point whose $(t+1)^{\text{th}}$ component is a known fraction of the secret. By running an approximation algorithm to find the closest vector of the lattice generated by the basis $\mathbf{M}$ to $\mathbf{t}'$, the secret can be recovered.

The LTSSS surpasses Shamir's approach as it embraces quantum security, avoids finite fields, supports diverse access structures, and offers efficient operations along with error-tolerance [Khorasgani et al.(2014)]. Information-theoretic security and flexibility make it an appealing alternative, despite being less mature in the field.

## 4. Proposed Methodology

The proposed methodology endeavors to heighten the security of digital asset wallets by amalgamating PQC with ZKP. As shown in Figure 3, the proposed architecture comprises a client and a server, wherein the client engenders a secret key predicated on the user's username and passwords leveraging Lattice-based cryptography. The server, in turn, generates a random point on the lattice, which acts as a secret key and disseminates it with the client utilizing a LTSSS. In a secure manner, the client utilizes the generated secret key in tandem with ZKP to derive the private and public keys. Notably, this is achieved without any sensitive information being exposed to the server, thus bolstering the system's overall security.

### 4.1. Algorithmic Insights

The procedure described in Algorithm 3 is a function used to compute a hash value for a given message. It utilizes two cryptographic algorithms, bcrypt and SHA-256, to ensure the security and integrity of the resulting hash. The algorithm takes a message as input and begins by setting a cost parameter to 12. This cost parameter determines the computational effort required to compute the bcrypt hash.

**Algorithm 3** Function to compute Hash value

```
procedure HASH($message$)
    $cost \leftarrow 12$
    $hashBcrypt \leftarrow$ BCRYPT($cost, salt, message$)
    $hashValue \leftarrow \text{SHA256}(hashBcrypt)$
    return $hashValue$
end procedure
```

Higher values of cost increase the time and resources needed to compute the hash, thereby enhancing security against brute-force attacks. The bcrypt function applies a one-way hashing algorithm that incorporates the cost and salt to generate a bcrypt hash. The salt is a random value used to further strengthen the hash and protect against rainbow table attacks. The resulting bcrypt hash is then passed as input to the SHA-256 algorithm. It is designed to be computationally secure and resistant to collision attacks, where different inputs produce the same hash value.The computed SHA-256 hash value is assigned to the variable "hashValue" and returned as the output of the function.

**Algorithm 4** Client-side Lattice Computation

```
procedure LATTICECOMPUTATION($\eta$)
    $d \leftarrow 256$
    $q \leftarrow 2^{14}$
    $s \leftarrow 2d$
    $\sigma \leftarrow \sqrt{\frac{d}{q}}$
    $binary\_string \leftarrow$ ConvertToBinary($\eta, 16$)
    $\rho \leftarrow []$
    for $i \leftarrow 0$ to $d-1$ step 16 do
        $chunk \leftarrow$ binary_string[$i : i+16$]
        $decimal\_value \leftarrow$ int($chunk, 2$)
        $\rho$.append($decimal\_value \mod q$)
    end for
    $\Delta_1 \leftarrow []$
    for $i \leftarrow 0$ to $d-1$ do
        $\Delta_1$.append(random.gaussian($0, \sigma$))
    end for
    $\omega_1 \leftarrow []$
    for $i \leftarrow 0$ to $s-1$ do
        $\omega_1$.append(random.int($0, 1$))
    end for
    $\tau_1 \leftarrow (\sum_{i=0}^{d-1} \rho[i] \cdot \omega_1[i]) + \left(\frac{q}{2} \cdot \sum_{i=0}^{d-1} \Delta_1[i]\right) \mod q$
    return $\tau_1, \Delta_1, \omega_1$
end procedure
```

The procedure presented in Algorithm 4 performs client-side computations to transform a hexadecimal value into a lattice point within a lattice-based cryptography scheme. It accomplishes through the following steps:

- Initialize the parameters: Set the dimension $d$ to 256, the modulus $q$ to $2^{14}$, and the lattice size $s$ to twice the dimension.
- Calculate the standard deviation $\sigma$: It is computed as the square root of the ratio between the dimension $d$ and the modulus $q$. This value is used in the following steps.
- Convert the hexadecimal value to a binary string: The input hexadecimal value is converted into a binary string representation with a fixed length of 16 bits.
- Prepare the $\rho$ vector: Iterate over the binary string in chunks of 16 bits and convert each chunk into a decimal value. Each decimal value is then reduced modulo $q$ and added to the $\rho$ vector, resulting in a vector of $d$ elements.
- Generate the $\Delta_1$ vector: Generate a vector $\Delta_1$ of $d$ elements by assigning random values from a Gaussian distribution with mean 0 and standard deviation $\sigma$.
- Generate the $\omega_1$ vector: Generate a binary vector $\omega_1$ of $s$ elements by randomly assigning either 0 or 1 to each element.
- Compute the lattice point $\tau_1$: Calculate $\tau_1$ as the sum of the element-wise multiplication between the $\rho$ and $\omega_1$ vectors, added to the product of half the modulus $q$ and the sum of the elements in the $\Delta_1$ vector. Finally, reduce the result modulo $q$.
- Return the lattice point: The algorithm returns the computed lattice point $\tau_1$, the vectors $\Delta_1$ and $\omega_1$ for future use.

**Algorithm 5** Server-side Lattice Computation

```
procedure LATTICECOMPUTATION($\tau_1$)
    $d \leftarrow 256$
    $q \leftarrow 2^{14}$
    $s \leftarrow 2d$
    $\sigma \leftarrow \sqrt{\frac{d}{q}}$
    $\Delta_2 \leftarrow []$
    for $i \leftarrow 0$ to $d-1$ do
        $\Delta_2$.append(random.gaussian($0, \sigma$))
    end for
    $\omega_2 \leftarrow []$
    for $i \leftarrow 0$ to $s-1$ do
        $\omega_2$.append(random.int($0, 1$))
    end for
    $\tau_2 \leftarrow (\sum_{i=0}^{d-1} \tau_1[i] \cdot \omega_2[i]) + \left(\frac{q}{2} \cdot \sum_{i=0}^{d-1} \Delta_2[i]\right) \mod q$
    return $\tau_2, \Delta_2, \omega_2$
end procedure
```

The procedure in Algorithm 5 is similar to Algorithm 4, performs server-side computations to transform a given lattice point represented by the $\tau_1$ vector into a new lattice point. The algorithm initializes the parameters, including the dimension $d$, modulus $q$, and lattice size $s$, which define the properties of the lattice. It then generates two vectors, $\Delta_2$ and $\omega_2$, of size $d$ and $s$ respectively. The $\Delta_2$ vector is created by assigning random values from a Gaussian distribution with mean 0 and standard deviation $\sigma$, which is computed as the square root of $d/q$. The $\omega_2$ vector is formed by randomly selecting either 0 or 1 for each element. It proceeds to compute the new lattice point $\tau_2$. This is accomplished by performing element-wise multiplication between the $\tau_1$ vector and the $\omega_2$ vector, followed by summing the products. Additionally, half the modulus $q$ is multiplied by the sum of the elements in the $\Delta_2$ vector. The resulting values are combined and reduced modulo $q$ to obtain the new lattice point $\tau_2$. Finally, the algorithm returns the computed new lattice point $\tau_2$, along with the $\Delta_2$ and $\omega_2$ vectors. These values are used in subsequent steps of the proposed system.

**Algorithm 6** Real Lattice Point Generation

```
procedure REALLATTICEPOINT($\tau_2, \Delta_1, \omega_1$)
    $d \leftarrow 256$
    $q \leftarrow 2^{14}$
    $dot\_product \leftarrow 0$
    for $i \leftarrow 0$ to length$(\omega_1) - 1$ do
        $dot\_product \leftarrow dot\_product + \omega_1[i] \times \Delta_1[i]$
    end for
    $q_0 \leftarrow \lfloor q/2 \rfloor$
    $diff \leftarrow (\tau_2 - q_0 \times dot\_product) \mod q$
    $inv\_q_0 \leftarrow$ ExtendedEuclidean$(q_0, q)$
    $result \leftarrow (inv\_q_0 \times diff) \mod q$
    $\rho \leftarrow []$
    for $i \leftarrow 0$ to $d - 1$ step 16 do
        $chunk \leftarrow$ Substring$(result, i, i + 15)$
        $decimal\_value \leftarrow$ ConvertToDecimal$(chunk, 2)$
        $\rho$.Append$(decimal\_value)$
    end for
    return $\rho$
end procedure
```

The procedure presented in Algorithm 6 is used to generate the original lattice point $\rho$ from the provided parameters $\tau_2$, $\Delta_1$, and $\omega_1$. The algorithm performs the following steps:

- Initialize the dimension $d$ to 256 and modulus $q$ to $2^{14}$, which defines the properties of the lattice.
- Compute the dot product between the vectors $\Delta_1$ and $\omega_1$ by iterating over their elements and accumulating the product of corresponding elements.
- Compute $q_0$ as half of the modulus $q$.
- Calculate the difference between $\tau_2$ and $q_0$ times the dot product. Reduce the result modulo $q$ to ensure it falls within the lattice.
- Compute the modular inverse $inv\_q_0$ of $q_0$ using the Extended Euclidean algorithm.
- Multiply $inv\_q_0$ with the difference calculated earlier and reduce the result modulo $q$. This step ensures that the final result falls within the lattice and can be represented as a valid lattice point.
- Construct the $\rho$ vector by splitting the final result into chunks of 16 bits.
- Convert each chunk from binary to decimal representation.
- Append the decimal values to the $\rho$ vector.
- Return the $\rho$ vector, representing the original lattice point.

## 4.2. Working Mechanism

The proposed methodology is divided into three distinct components, namely the client-side computation, server-side computation and client-side key generation.

**Client-side Computation:**

1) Compute $\eta$ by applying the hash function (see Algorithm 3) to the username and password1.
2) Compute $\mu$ by applying the hash function (see Algorithm 3) to password1 and password2.
3) Compute $\tau_1$, $\Delta_1$, and $\omega_1$ by using the hash-to-lattice-point algorithm (see Algorithm 4) with $\eta$ as input.
4) Send a POST request to the server, including the hashed username and $\tau_1$.

**Server-side Computation:**

1) If the hashed username exists in the server's database, retrieve the stored values $\Delta_2$ and $\omega_2$ associated with it.
2) If the hashed username does not exist, compute $\tau_2$ by using the point-to-lattice-point algorithm (see Algorithm 5) with $\tau_1$ as input.
3) Store the computed $\Delta_2$ and $\omega_2$ in the database, mapping them to the hashed username.
4) Send a response to the client, including $\tau_2$.

**Client-side Key Generation:**

1) Compute $\rho$ by using the real lattice point algorithm (see Algorithm 6) with $\tau_2$, $\Delta_1$, and $\omega_1$ as input.
2) Generate a secret $\phi$ by using $\rho$ and $\mu$ as two shares of a (2, 3) LTSSS.
3) Generate a third share from the secret for backup purposes.
4) Generate a private and public key pair using the Kyber512 algorithm, with $\phi$ as input.

On the client-side, the computation begins with the application of Algorithm 3 i.e., a hash function to the username and password1, resulting in a hashed value called $\eta$. Additionally, another hash computation is performed using password1 and password2, producing a hashed value called $\mu$. These hash functions ensure the confidentiality and integrity of sensitive information. Furthermore, the client applies Algorithm 4 i.e., a lattice computation function to $\eta$, resulting in the computation of $\tau_1$, $\Delta_1$, and $\omega_1$. These values represent points on a lattice structure and serve as the foundation for subsequent operations. The client then sends a POST request to the server, transmitting the hashed username and $\tau_1$. This request establishes a communication channel between the client and server.

On the server-side, the server checks if the hashed username exists in its database. If it does, the server retrieves the corresponding stored values $\Delta_2$ and $\omega_2$. These values ensure consistency in the subsequent computations. If the hashed username does not exist, the server computes $\tau_2$ using Algorithm 5 i.e., the server-side lattice computation with $\tau_1$ as input. The server stores the computed $\Delta_2$ and $\omega_2$

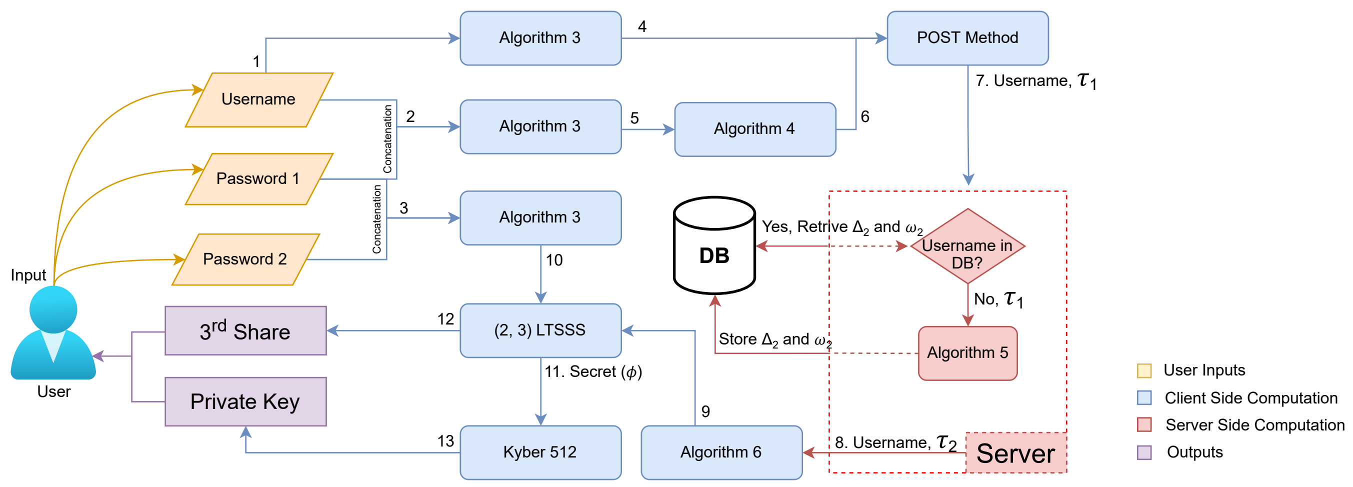

Figure 3. Working mechanism of the proposed solution

in its database, associating them with the hashed username for future reference. A response is then sent back to the client, including $\tau_2$, establishing a secure channel for further communication.

At the end, on the client-side, the client computes $\rho$ by using the Algorithm 6 i.e., real lattice point generation function with $\tau_2$, $\Delta_1$, and $\omega_1$ as input. This computation converts the received lattice point into a usable form. The client generates a backup key or third share, a secret $\phi$ by utilizing $\rho$ and $\mu$ as shares of a (2, 3) LTSSS. This ensures that multiple shares are needed to reconstruct the secret, enhancing security. A third share is generated for backup purposes. Additionally, the client generates a private and public key pair using the Kyber512 algorithm with $\phi$ as input. This key pair forms the foundation for secure data exchange.

Users can securely store the backup key using methods such as offline storage (e.g., printed paper), secure local storage, or cloud storage. This backup key ensures access to the private key and the ability to perform essential transactions even in scenarios where the server experiences downtime or disruptions.

The proposed method also incorporates a noteworthy feature that allows users to rekey their private key while retaining the same username and passwords. When users opt for rekeying, the server assigns new random values for $\Delta_2$ and $\omega_2$, which are subsequently utilized in generating a fresh private key. It is essential to emphasize that the values $\Delta_2$ and $\omega_2$ stored in the database hold no valuable information pertaining to the private key itself; rather, they serve as inputs for the key generation process. The inclusion of this rekeying functionality offers an elevated level of security and flexibility to users. In the event of suspected private key compromise or a desire to enhance security measures, users can effortlessly initiate the rekeying process. By doing so, any potential vulnerabilities associated with the previous private key are effectively mitigated. The freshly generated private key, derived from the updated $\Delta_2$ and $\omega_2$ values, operates independently from its predecessor and introduces an enhanced level of security. By separating the private key generation process from the username and passwords, the proposed method ensures that even if the stored $\Delta_2$ and $\omega_2$ values are compromised, they hold no information that could be exploited to reveal the private key. This reinforces the safeguarding of user assets and bolsters the overall security of the wallet.

## 5. Results

The proposed method generates several key results that contribute to its superiority over existing wallets and enhance its security against quantum computers.

- **User-Friendly Authentication:** Instead of relying on complex recovery phrases, this method uses familiar credentials, namely a username and password, for authentication. This approach simplifies the user experience and makes it easier for individuals to access their wallets securely.
- **Quantum-Resistance:** The method addresses the emerging threat of quantum computers by employing quantum-resistant cryptographic algorithms. The wallet generates private and public keys using the Kyber512 algorithm, which is designed to withstand attacks from quantum computers. This ensures that the generated keys remain secure even in the face of advancements in quantum computing technology.
- **Robust Key Size:** The wallet generates a Kyber512 private key, which has a key size of 512 bits. The larger key size provides significantly higher cryptographic strength compared to the 128 or 256-bit keys commonly used in recovery phrase-based wallets. This increased key size contributes to the wallet's overall security and makes it more resistant to brute-force and cryptographic attacks.

- **Rekeying Capability:** One notable advantage of this method is the ability to rekey the private key using the same username and password. In traditional recovery phrase-based wallets, the recovery phrase is the sole means of regaining access to the wallet. However, in this method, users can rekey their private key by providing their username and password. This feature offers convenience and flexibility, allowing users to regain access to their wallets using familiar credentials without solely relying on a recovery phrase.

## 6. Implications

The utilization of LWE-based cryptography within the cryptocurrency wallet algorithm presents a compelling avenue for achieving a high level of security against known attacks targeting the LWE problem. The LWE problem has exhibited formidable resistance to a diverse range of attacks, spanning both classical and quantum algorithms, rendering it an attractive option for cryptographic applications. The incorporation of Secure Sockets Layer (SSL) and Multi-Factor Authentication (MFA) serves as an effective deterrent against the vulnerability posed by Man-in-The-Middle (MiTM) attacks. SSL serves to facilitate a secure and encrypted communication channel between two entities, thereby impeding malevolent actors from intercepting and manipulating data while in transit. MFA, on the other hand, amplifies the security of user credentials by fortifying the authentication process through the stipulation of at least two identification modes prior to access being granted to their account.

Nevertheless, It is crucial to acknowledge that although these measures are efficacious, they are not entirely impervious to the risks of MITM attacks. The possibility of exploiting SSL or MFA implementation vulnerabilities, or employing tactics such as phishing by perpetrators, remains a potential concern. The comprehensive security of the wallet algorithm is also reliant on the security of other constituent elements, including the authentication mechanism and the storage of shared secrets. Notably, the deployment of a two-password system for generating the shared secret may engender certain usability challenges that necessitate careful consideration.

Users are required to remember two distinct passwords for the two stages of password hashing, presenting a plausible usability barrier. This may result in users struggling to remember complex passwords for both stages or forgetting one of the passwords entirely, ultimately resulting in self-lockout from the wallet. It is crucial to design password requirements and recovery mechanisms that balance security and usability to mitigate this issue. Despite these challenges, the two-password system implemented in the wallet algorithm may be more user-friendly than current wallet systems that rely on 12/24 word seed-phrases. Such phrases may be difficult for users to recall and susceptible to brute-force attacks if not chosen carefully.

Apart from usability challenges, it is also important to consider the potential for attacks on the LWE-based cryptography in the future. While the LWE problem is currently deemed to be secure against a wide range of attacks, the possibility of new attacks or quantum computing advances rendering the system vulnerable cannot be entirely precluded. To address this issue, the wallet algorithm incorporates additional security measures, such as PQC and ZKP. PQC techniques have been devised to withstand attacks by quantum computers, which can break many traditional cryptographic schemes. Meanwhile, ZKP techniques enable cryptographic proofs to be furnished without revealing sensitive information, ensuring the continued security of the system, even if the attacker attains partial access to the system.

The use of LWE-based cryptography in the cryptocurrency wallet algorithm represents a promising approach for achieving high security. Nevertheless, the system's overall security is contingent upon the security of multiple components, including the password authentication mechanism and the storage of shared secrets. It is imperative to balance usability and security when designing these elements to ensure that the system is secure and user-friendly. Furthermore, it is vital to continually monitor and update the system's security in response to emerging threats and advances in cryptographic research.

## 7. Conclusion

Our proposed algorithm for improving the security of cryptocurrency wallets using LWE-based cryptography offers several advantages over existing solutions. Unlike many current wallets that rely on a 12/24 word seed-phrase, our algorithm uses a two-password system for generating the shared secret. This not only provides stronger security against quantum attacks, but also makes the wallet more user-friendly by reducing the risk of users losing their recovery phrase. Our algorithm also offers the added advantage of rekeying the private key using the same username and password, further enhancing the wallet's security.

Moreover, using LWE-based cryptography provides a high level of security against known attacks. The algorithm also provides post-quantum security using PQC and ZKP, ensuring the wallet remains secure even in future attacks on LWE-based cryptography. However, we recognize that using a two-password system may be inconvenient for some users, as they need to remember two passwords and ensure they are sufficiently complex and secure. To address this, we plan to explore alternative approaches that improve the user experience while maintaining a high level of security, such as using a single password or biometric based authentication, social recovery mechanisms.

## References

[Andryukhin(2019)] AA Andryukhin. 2019. Phishing attacks and preventions in blockchain based projects. In *2019 International Conference on Engineering Technologies and Computer Science (EnT)*. IEEE, 15–19.

[Augot et al.(2015)] Daniel Augot, Lejla Batina, Daniel J. Bernstein, Joppe W. Bos, Johannes A. Buchmann, Wouter Castryck, Orr Dunkelman, Tim Güneysu, Shay Gueron, Andreas Hülsing, Tanja Lange, Christian Rechberger, Peter Schwabe, Nicolas Sendrier, Frederik Vercauteren, and Bo-Yin Yang. 2015. Initial recommendations of long-term secure post-quantum systems.

[Belchior et al.(2022)] Rafael Belchior, André Vasconcelos, Sérgio Guerreiro, and Miguel Correia. 2022. A Survey on Blockchain Interoperability: Past, Present, and Future Trends. *Comput. Surveys* 54, 8 (2022). https://doi.org/10.1145/3471140

[Bernstein(2009a)] Daniel J. Bernstein. 2009a. Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete? (5 2009). https://cr.yp.to/hash/collisioncost-20090517.pdf

[Bernstein(2009b)] Daniel J. Bernstein. 2009b. *Introduction to post-quantum cryptography*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1–14. https://doi.org/10.1007/978-3-540-88702-7_1

[Bernstein(2010)] Daniel J. Bernstein. 2010. Grover vs. McEliece. (3 2010). https://cr.yp.to/codes/grovercode-20100303.pdf

[Blum et al.(1988)] Manuel Blum, Paul Feldman, and Silvio Micali. 1988. Non-Interactive Zero-Knowledge and Its Applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing* (Chicago, Illinois, USA) *(STOC '88)*. Association for Computing Machinery, New York, NY, USA, 103–112. https://doi.org/10.1145/62212.62222

[Bui et al.(2019)] Thanh Bui, Siddharth Prakash Rao, Markku Antikainen, and Tuomas Aura. 2019. Pitfalls of open architecture: How friends can exploit your cryptocurrency wallet. In *Proceedings of the 12th European Workshop on Systems Security*. 1–6.

[Chaum et al.(1988)] David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. 1988. An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. In *Advances in Cryptology — EUROCRYPT' 87*, David Chaum and Wyn L. Price (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 127–141.

[Chen et al.(2020a)] Huashan Chen, Marcus Pendleton, Laurent Njilla, and Shouhuai Xu. 2020a. A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses. *Comput. Surveys* 53, 3 (2020). https://doi.org/10.1145/3391195

[Chen et al.(2020b)] Huashan Chen, Marcus Pendleton, Laurent Njilla, and Shouhuai Xu. 2020b. A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Computing Surveys (CSUR)* 53, 3 (2020), 1–43.

[Courtois and Mercer(2017)] Nicolas T Courtois and Rebekah Mercer. 2017. Stealth address and key management techniques in blockchain systems. In *ICISSP 2017-Proceedings of the 3rd International Conference on Information Systems Security and Privacy*. 559–566.

[Di Angelo and Salzer(2020)] Monika Di Angelo and Gernot Salzer. 2020. Characteristics of wallet contracts on Ethereum. In *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE, 232–239.

[Ducas et al.(2013)] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. 2013. Lattice Signatures and Bimodal Gaussians. Cryptology ePrint Archive, Paper 2013/383. https://eprint.iacr.org/2013/383 https://eprint.iacr.org/2013/383.

[Erinle et al.(2023)] Yimika Erinle, Yathin Kethepalli, Yebo Feng, and Jiahua Xu. 2023. SoK: Design, Vulnerabilities, and Security Measures of Cryptocurrency Wallets. arXiv:2307.12874 [cs.CR]

[Eskandari et al.(2018)] Shayan Eskandari, Jeremy Clark, David Barrera, and Elizabeth Stobert. 2018. A first look at the usability of bitcoin key management. *arXiv preprint arXiv:1802.04351* (2018).

[Eyal(2022)] Ittay Eyal. 2022. On Cryptocurrency Wallet Design. In *3rd International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[Gershon(2013)] Eric Gershon. 2013. New qubit control bodes well for future of quantum computing. (1 2013). https://phys.org/news/2013-01-qubit-bodes-future-quantum.html

[Goldreich et al.(1997)] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. 1997. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology — CRYPTO '97*, Burton S. Kaliski (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 112–131.

[Götte and Scheuermann(2021)] Jan Sebastian Götte and Björn Scheuermann. 2021. Tech Report: Inerial HSMs Thwart Advanced Physical Attacks. *Cryptology ePrint Archive* (2021).

[Güneysu et al.(2012)] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. 2012. Practical Lattice-Based Cryptography: A Signature Scheme for Embedded Systems. In *Cryptographic Hardware and Embedded Systems – CHES 2012*, Emmanuel Prouff and Patrick Schaumont (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 530–547.

[Guo and Yu(2022)] Huaqun Guo and Xingjie Yu. 2022. A Survey on Blockchain Technology and its security. *Blockchain: Research and Applications* 3, 2 (2022), 100067.

[He et al.(2018)] Shuangyu He, Qianhong Wu, Xizhao Luo, Zhi Liang, Dawei Li, Hanwen Feng, Haibin Zheng, and Yanan Li. 2018. A social-network-based cryptocurrency wallet-management scheme. *IEEE Access* 6 (2018), 7654–7663.

[He et al.(2019)] Xiaojian He, Jinfu Lin, Kangzi Li, and Ximeng Chen. 2019. A novel cryptocurrency wallet management scheme based on decentralized multi-constrained derangement. *IEEE Access* 7 (2019), 185250–185263.

[Hoffstein et al.(1998)] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. 1998. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory*, Joe P. Buhler (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 267–288.

[Homoliak et al.(2018)] Ivan Homoliak, Dominik Breitenbacher, Alexander Binder, and Pawel Szalachowski. 2018. An air-gapped 2-factor authentication for smart-contract wallets. *arXiv preprint arXiv:1812.03598* (2018).

[Khadzhi et al.(2020)] Andrey S Khadzhi, Sergey V Zareshin, and Oleg V Tarakanov. 2020. A Method for Analyzing the Activity of Cold Wallets and Identifying Abandoned Cryptocurrency Wallets. In *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. IEEE, 1974–1977.

[Khan et al.(2019)] Abdul Ghaffar Khan, Amjad Hussain Zahid, Muzammil Hussain, and Usama Riaz. 2019. Security of cryptocurrency using hardware wallet and qr code. In *2019 International Conference on Innovative Computing (ICIC)*. IEEE, 1–10.

[Khorasgani et al.(2014)] Hamidreza Amini Khorasgani, Saba Asaad, Taraneh Eghlidos, and Mohammadreza Aref. 2014. A lattice-based threshold secret sharing scheme. In *2014 11th International ISC Conference on Information Security and Cryptology*. IEEE, 173–179.

[Li et al.(2020)] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. 2020. A survey on the security of blockchain systems. *Future Generation Computer Systems* 107 (2020), 841–853.

[Lixie(2021)] Lixie. 2021. Zero-knowledge proofs explained: Part 1. (10 2021). https://www.expressvpn.com/blog/zero-knowledge-proofs-explained/

[Lyubashevsky et al.(2013)] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2013. On Ideal Lattices and Learning with Errors over Rings. *J. ACM* 60, 6, Article 43 (nov 2013), 35 pages. https://doi.org/10.1145/2535925

[Mangipudi et al.(2022)] Easwar Vivek Mangipudi, Udit Desai, Mohsen Minaei, Mainack Mondal, and Aniket Kate. 2022. Uncovering Impact of Mental Models towards Adoption of Multi-device Crypto-Wallets. *Cryptology ePrint Archive* (2022).

[Nakamoto(2009)] Satoshi Nakamoto. 2009. Bitcoin: A Peer-to-Peer Electronic Cash System. *Cryptography Mailing list at https://metzdowd.com* (03 2009).

[Pal et al.(2021)] Om Pal, Bashir Alam, Vinay Thakur, and Surendra Singh. 2021. Key management for blockchain technology. *ICT express* 7, 1 (2021), 76–80.

[Peikert(2014)] Chris Peikert. 2014. Lattice Cryptography for the Internet. Cryptology ePrint Archive, Paper 2014/070. https://eprint.iacr.org/2014/070 https://eprint.iacr.org/2014/070.

[Quisquater et al.(1990)] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, and Soazig Guillou. 1990. How to Explain Zero-Knowledge Protocols to Your Children. In *Advances in Cryptology — CRYPTO' 89 Proceedings*, Gilles Brassard (Ed.). Springer New York, New York, NY, 628–631.

[Regev(2009)] Oded Regev. 2009. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *J. ACM* 56, 6, Article 34 (sep 2009), 40 pages. https://doi.org/10.1145/1568318.1568324

[Rezaeighaleh and Zou(2019)] Hossein Rezaeighaleh and Cliff C Zou. 2019. Deterministic sub-wallet for cryptocurrencies. In *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 419–424.

[Shbair et al.(2021)] Wazen M Shbair, Eugene Gavrilov, and Radu State. 2021. HSM-based Key Management Solution for Ethereum Blockchain. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 1–3.

[Shor(1997)] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 26, 5 (1997), 1484–1509. https://doi.org/10.1137/S0097539795293172 arXiv:https://doi.org/10.1137/S0097539795293172

[Spectrum(2008)] IEEE Spectrum. 2008. How quantum computers threaten our current cryptography system and what we can do about it. (11 2008). https://spectrum.ieee.org/qa-with-postquantum-computing-cryptography-researcher-jintai-ding

[Steinfeld et al.(2004)] Ron Steinfeld, Huaxiong Wang, and Josef Pieprzyk. 2004. Lattice-Based Threshold-Changeability for Standard Shamir Secret-Sharing Schemes. In *Advances in Cryptology - ASIACRYPT 2004*, Pil Joong Lee (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 170–186.

[Suratkar et al.(2020)] Saurabh Suratkar, Mahesh Shirole, and Sunil Bhirud. 2020. Cryptocurrency wallet: A review. In *2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)*. IEEE, 1–7.

[Urien(2021)] Pascal Urien. 2021. Innovative Countermeasures to Defeat Cyber Attacks Against Blockchain Wallets. *2021 5th Cyber Security in Networking Conference, CSNet 2021* (2021), 49–54. https://doi.org/10.1109/CSNet52717.2021.9614649

[Wu and Wang(2014)] Huixin Wu and Feng Wang. 2014. A Survey of Noninteractive Zero Knowledge Proof System and Its Applications. *The Scientific World Journal* 2014 (04 May 2014), 560484. https://doi.org/10.1155/2014/560484

[Zamani et al.(2020)] Efpraxia Zamani, Ying He, and Matthew Phillips. 2020. On the security risks of the blockchain. *Journal of Computer Information Systems* 60, 6 (2020), 495–506.

[Zhang et al.(2015)] Jiang Zhang, Zhenfeng Zhang, Jintai Ding, Michael Snook, and Özgür Dagdelen. 2015. Authenticated Key Exchange from Ideal Lattices. In *Advances in Cryptology - EUROCRYPT 2015*, Elisabeth Oswald and Marc Fischlin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 719–751.