Post-Quantum Secure Deterministic Wallet: Stateless, Hot/Cold Setting, and More Secure

Mingxing Hu

Shanghai Jiao Tong University, Shanghai, China
mxhu2018@sjtu.edu.cn

Abstract. Since the invention of Bitcoin, cryptocurrencies have gained huge popularity. Crypto wallet, as the tool to store and manage the cryptographic keys, is the primary entrance for the public to access cryptocurrency funds. Deterministic wallet is an advanced wallet mechanism that has been proposed to achieve some appealing virtues, such as low-maintenance, easy backup and recovery, supporting functionalities required by cryptocurrencies, and so on. However, the existing deterministic wallet schemes especially in the quantum world still have a long way to be practical. The first barrier is how to build a deterministic wallet scheme without relying on the state, i.e., stateless. The stateful deterministic wallet scheme must internally maintain and keep refreshing synchronously a parameter named state which makes the implementation in practice become more complex. And once one of the states is leaked, thereafter the security notion of unlinkability is cannot be guaranteed (referred to as the weak security notion of forward unlinkability). The second barrier is how to derive the session secret keys from the master secret key in one-way. There are security shortfalls in previous works, they suffer a fatal vulnerability when a minor fault happens (say, one derived key is compromised somehow), then the damage is not limited to the leaked derived key, instead, it spreads to the master key and the whole system collapses. The third barrier is how to build a post-quantum secure deterministic wallet scheme supporting hot/cold setting, which is important since nearly all popular cryptocurrencies relied on the hardness problems that can be broken by quantum adversaries, and the hot/cold setting is a widely adopted method to effectively reduce the exposure chance of secret keys and hence improving the security of the system. The last barrier is how to build a deterministic wallet scheme with standard security notion of unforgeability. It is motivated by previous works which are based on a weaker/nonstandard unforgeability notion, in which the adversary is only allowed to query and forge the signatures w.r.t. the public keys that were assigned by the challenger.

In this work, we present a new deterministic wallet scheme in quantum world, which is stateless, supports hot/cold setting, satisfiies stronger security notions, and is more efficient. In particular, we reformalize the syntax and security models for deterministic wallets, capturing the functionality and security requirements (including full unlinkability and standard unforgeability) imposed by the practice in cryptocurrency. Then we propose a deterministic wallet construction and prove its security in

the quantum random oracle model. Finally, we show our wallet scheme is more practicable by analyzing an instantiation of our wallet scheme based on the signature scheme Falcon.

Keywords: Deterministic wallets · Post-Quantum · Lattice-Based cryptography · Blockchain · Cryptocurrency.

1 Introduction

Since the introduction of Bitcoin [33], cryptocurrencies has been undergoing a tremendous development as they provide a revolutionary payment paradigm. In most cryptocurrencies, balance updates are executed via transactions between coin-addresses, and *digital signature* [34,38] is employed to enable users to own and spend their coins. More specifically, each coin is assigned to a coin-address which technically is represented by a public key of a digital signature scheme, implying that the coin belongs to the owner of the public key. If a coin owner wants to spend the coin on a public key pk, he needs to generate a transaction tx and a signature σ such that (tx, σ) is a valid (message, signature) pair w.r.t. pk, authenticating the spending of the coin by this transaction. In this setting, the secret keys naturally became a highly attractive target for attacks since the users control funds via secret keys. Therefore, key management plays a crucial role in cryptocurrencies and it needs to work like a *wallet* for the coins.

Deterministic Wallet and Its Merits. *Deterministic wallet* [9] has been accepted as one of the most prominent solutions in the community for protecting the keys' security in cryptocurrency. At a high level, a deterministic wallet consists of two entities (hot wallet, cold wallet) and a deterministic key derivation mechanism. Deploying wallet systems in hot/cold setting is a widely adopted approach [15,3,13,17,42,27], which effectively reduce the exposure chance of secret keys and achieve better safety of the coins. More precisely, the hot wallet is permanently connected to the network, while the cold wallet stores the secret key and comes online only rarely (e.g., when a large amount of money has to be transferred). After an initialization phase, there is a master key pair (mpk, msk), where the master secret key msk is stored on the cold wallet, while the hot wallet keeps the corresponding master public key mpk. The deterministic key derivation mechanism allows both cold and hot wallets to derive matching secret and public session keys without interacting with each other. As the name of *deterministic* wallet implies, the deterministic property means that all keys in a wallet are deterministically generated from a "seed" so that when necessary (e.g., the crash of the device hosting the wallet) the wallet owner can recover all the keys from the seed. As Das et al. [15] formalized the concept of deterministic wallets, it provides two security guarantees. The first security guarantee is *wallet unforgeability* which states that once coins are sent to the cold wallet must remain secure even if the hot wallet is corrupted. The second one is *wallet unlinkability*, which guarantees that transactions sending money to different public session keys that were derived from the same master public key should be unlinkable.

Limits on the Stateful Deterministic Wallets. Since the notion of the deterministic wallet has been formally formalized in the work [15], this work and its following works [17,3] which includes the state-of-the-art work [3] all relied on a parameter named *state*, i.e., their works are stateful. This parameter makes the wallet system fall into problems. On efficiency, the stateful wallet systems must internally maintain and keep refreshing synchronously the state between the hold wallet and cold wallet, and must additionally set a mechanism to confirm if the intermediate state is leaked and the original state is erased securely, which significantly increases the difficulty of implementation. More precisely, after the initialization phase of wallet systems, there is an initial state St_0 and a pair of master key pairs (mpk, msk), if there is a need for session keys derivation w.r.t. an identity such as ID_1, the hot (resp., cold) wallet takes as input the mpk (resp., msk), St_0, and ID_1, outputs a new state St_1 and session key pk_{ID_1} (resp., sk_{ID_1}) by the key derivation algorithm $\mathtt{PKDer}(mpk, St_0, ID_1)$ (resp., $\mathtt{SKDer}(msk, St_0, ID_1)$). If there is once more need for session keys derivation w.r.t. an identity such as ID_2, just takes as input the refreshed St_1 and ID_2 and repeatedly invoke the key derivation procedure again. For simplicity, we can abstract the key derivation as below:

$$(pk_{ID_i}, St_i) \leftarrow \mathtt{PKDer}(mpk, St_{i-1}, ID_i), \quad (sk_{ID_i}, St_i) \leftarrow \mathtt{SKDer}(msk, St_{i-1}, ID_i)$$

More specifically, the new state St_i is computed by a random oracle function H that is called by the above key derivation algorithms, that is $(\cdot, St_i) \leftarrow \mathsf{H}(St_{i-1}, ID_i)$. Then recall that we mentioned the key derivation procedure is deterministic, i.e., the algorithms $(\mathtt{PKDer}, \mathtt{SKDer})$ are deterministic. So we now can observe the shortfalls exposing on the security of unlinkability, that is once one of the above-used states leaked, assuming this leaked state is St_i, then the adversary can use H to compute all the states $\{St_i, St_{i+1}, St_{i+2}, \cdots\}$ that prepare to use in future, and hence the unlinkability will be broken by checking if $(pk_{ID_i}, St_i) = \mathtt{PKDer}(mpk, St_{i-1}, ID_i)$ holds when the adversary observes the (pk_{ID_i}, St_i) are used in blockchain. Consequently, in this setting, the unlinkability only holds in prior to any hot wallet corruption, which is referred as the weak notion of unlinkability named *forward unlinkability* [15]. In summary, the stateful deterministic wallet systems severely limit the efficiency and the security notion of unlinkability is weak.

Security Shortfalls of Previous Works. Security is always the primary concern on the wallet systems since there were examples such as [23,26,18,42] pointed out that the previous works [36,40,37,14] suffers a realistic attack named *privilege escalation attack* [18], and presented the patched schemes. But no existing works analyze the deterministic wallets [15,3] and show the patched schemes. The privilege escalation attack is that once an attacker obtains a session secret key and the master public key somehow, he could reveal the master secret key and compromise the wallet completely and steal all the related coins. We are inspired by this attack and found the deterministic wallet schemes [15,3] also suffer the vulnerability. For instance, in the work [15], the derived key $sk' = msk \cdot \rho_{ID}$ where ρ_{ID} is a randomness and ρ_{ID} is invertible. Recall that the adversary

knows all the states since their unforgeability model allows the adversary to obtain the initial state St_0, therefore, the adversary can compute the target ρ_{ID} by $(\rho_{ID_i}, St_i) \leftarrow \mathsf{H}(St_{i-1}, ID_i)$, then the master secret key is revealed by $msk = sk' \cdot \rho_{ID}^{-1}$. This attack is also worked in the work [3] except the difference is that the session secret key derivation in [15] is multiplicative rerandomization (derivation), i.e., $sk' = msk \cdot \rho_{ID}$ while in [3] is additive rerandomization, i.e., $sk' = msk + \rho_{ID}$. Therefore, the fundamental cause of this attack is the process of session secret key derivation (rerandomization) is not one-way, i.e., invertible. In summary, a one-way support key derivation mechanism is an urgent need for deterministic wallet schemes. And we note that it is quite challenging since it not only needs to support one-way derivation but also needs to be compatible with the properties of deterministic wallet systems such as deterministic derivation and in the meanwhile, there is no interaction between the hot and cold wallet.

Lacking Post-Quantum Secure Deterministic Wallet in Hot/Cold Setting. As we described above, deploying wallet systems in hot/cold setting is widely adopted in almost all the works [15,3,13,17,42,27] since it can effectively reduce the risk of secret key exposure. The state-of-the-art work [3] presented the first post-quantum secure deterministic wallet and defined in the hot/cold setting, but fall short in building practical hot/cold deterministic wallet. This problem originated from the key derivation mechanism. Their deterministic wallet system is based on a signature scheme with rerandomizable keys which can be taken as a standard signature scheme augmented with two key rerandomization algorithms $(\mathtt{RandPK}, \mathtt{RandSK})$. As the name suggests, the algorithm `RandPK` (resp., `RandSK`) takes as input a pubic key pk (resp., secret key sk) and a randomness ρ then outputs a randomized public key $pk' \leftarrow \mathtt{RandPK}(pk, \rho)$ (resp., secret key $sk' \leftarrow \mathtt{RandSK}(sk, \rho)$). And these two algorithms are called by the key derivation algorithms of wallet scheme, i.e., `RandPK` and `RandSK` are called by `PKDer` (in hot wallet) and `SKDer` (in cold wallet), respectively. The point is that these rerandomization algorithms (`RandPK`, `RandSK`) are required to communicate when deploying in wallet systems, however, it is contradicted with the property of deterministic wallet that hot and cold wallets do not communicate with each other except when they are being initialized.

Consider a scenario, in which the rerandomization algorithms (`RandPK`, `RandSK`) have to synchronize after each invocation of `RandPK` or `RandSK` algorithm. Given msk and ρ, the algorithm `RandSK` uses ρ together with a counter `ctr` in order to deterministically generate a randomness ρ'. Then it computes the rerandomized secret key $sk' = msk + \rho'$ and outputs sk' only after verifying that it has the correct distribution, otherwise, it increases the counter as $\mathsf{ctr} = \mathsf{ctr} + 1$ and repeats this process. The algorithm `RandPK` algorithm needs to receive the corresponding `ctr` from `RandSK` in order to generate the correct rerandomized public key corresponding to sk'. Consequently, rigorously speaking, this post-quantum secure deterministic wallet scheme supports the hot/cold setting with a probability. We note that most cryptographic primitives used by cryptocurrencies today can be broken by quantum adversaries. Most notably, the ECDSA signature scheme

that is implemented by nearly all popular cryptocurrencies relies on the hardness of computing discrete logarithms, and hence can be broken by Shor's algorithm [Sho94]. Therefore, in order to make deterministic wallets more practical, we should not only design them in hot/cold setting but also in quantum world.

Weak/Non-Standard Security Notions of Prior Works. As we mentioned above, since the notion of deterministic wallet has been formally formalized by Das et al. [15], this work and its following works [17,3] which includes the state-of-the-art work of Alkadri et al. [3], their security notion of unlinkability is weak or nonstandard due to the relying on state. Not only that, their security notion of unforgeability is also weak/nonstandard. More precisely, their unforgeability models only allow the adversary to query the signing oracle and forge the signature with the randomness that is assigned by the challenger rather than adversarially chosen randomness, since the used randomness throughout the entire unforgeability game is sampled by a specified underlying oracle that is controlled by the challenger. In other words, the adversary can not query/forge the signatures with respect to the desired session public key, i.e., the adversary can not know the session public key that corresponds to the queried/forged message-signature tuple in advance. But in practice, the adversary knows all the states in the system since he can obtain the initial state when the hot wallet is compromised, and hence the adversary can compute the randomness and session public key corresponding to any identity, then the adversary can observe and forge signatures with respect to any session public key on the blockchain. Below we explain the details. Recall the definition of the unforgeability of deterministic wallet, which guarantees that once funds are transferred to the cold wallet they remain secure even if the hot wallet is compromised. Therefore the adversary obtains the initial state St_0 when the hot wallet is compromised. Then the adversary can compute the session public key $(pk_{ID_i}, St_i) \leftarrow \texttt{PKDer}(mpk, St_{i-1}, ID_i)$ with respect to any identity ID_i. More precisely, the $\texttt{PKDer}$ algorithm first computes the $(\rho_{ID_i}, St_i) \leftarrow \mathsf{H}(St_{i-1}, ID_i)$ by a random oracle function H then obtains the resulting pk_{ID_i} by calling $\texttt{RandPK}$ with input ρ_{ID_i}. Now the adversary can observe the activities with respect to this session public key pk_{ID_i}, which includes payments paid to pk_{ID_i} and the coins on pk_{ID_i} is spent by issuing a signature, in the meantime, the adversary can judge the time was right for forging a signature. Consequently, by the description and details as shown above, we can conclude that this weak security notion of unforgeability is non-standard and can not fully capture the practical requirements of deterministic wallets.

1.1 Our Contribution

In this work, our main contribution is to present a new deterministic wallet scheme with merits: post-quantum security, stateless, supporting hot/cold setting, satisfying stronger security notions, and more efficient. In particular, we reformalize the syntax and security models for deterministic wallets and the underlying signature scheme, capturing the functionality and security (wallet unlinkability and wallet unforgeability) requirements that the cryptocurrency

practice imposes on deterministic wallets. It is worth mentioning that the underlying signature scheme is the first post-quantum secure signature scheme with rerandomizable public keys (RKS), which is achieved by giving a generic construction from a lattice-based hash-and-sign signature scheme. Then we propose a deterministic wallet construction and prove its security in quantum random oracle model. Finally, we show and evaluate our wallet scheme is more compact and practicable by analyzing an instantiation of our wallet scheme based on Falcon [20] (signature scheme of NIST finalist). The merits of stateless, supporting hot/cold setting, more secure, and more efficiency of our deterministic wallet scheme will empower its applications in practice.

New Definitions: Natural, Stateless, Stronger. To fully capture the security requirements of deterministic wallets in practice, we reformalize the syntax and the security models of deterministic wallet scheme and the underlying RKS signature scheme. On the syntax, our wallet scheme is stateless, i.e., do not need to internally maintain and keep refreshing synchronously a state between the hot and cold wallets. And we augment the formalization of deterministic wallet from [15,3] that is

$$\mathtt{DW} := (\mathtt{DW.KeyGen}, \mathtt{DW.PKDer}, \mathtt{DW.SKDer}, \mathtt{DW.Sign}, \mathtt{DW.Ver})$$

and the RKS signature scheme from [3] that is

$$\mathtt{RKS} := (\mathtt{RKS.KeyGen}, \mathtt{RKS.RandPK}, \mathtt{RKS.RandSK}, \mathtt{RKS.Sign}, \mathtt{RKS.Ver})$$

with a randomness generation algorithm `DW.RandGen` and `RKS.RandGen`, respectively. Under this setting, the key derivation algorithms (`DW.PKDer`, `DW.SKDer`) only need to take as input a master public/secret key and a randomness such as $pk_{ID} \leftarrow \mathtt{DW.PKDer}(mpk, \rho_{ID})$ rather than $(pk_{ID}, St_i) \leftarrow \mathtt{DW.PKDer}(mpk, ID, St_{i-1})$ as prior works [15,3] which needs to take as input an identity ID and an original state St_{i-1}. Therefore it can observe that our syntax is more natural. Moreover, the deterministic wallet is stateless in our system model. More specifically, after the initialization phase, there is a pair of master keys and a chaincode from key generation algorithm, i.e., $(mpk, msk, ch) \leftarrow \mathtt{DW.KeyGen}$, then directly use the ch to derive session keys for each identity. Rather than doing as prior works, it generates $(mpk, msk, St_0) \leftarrow \mathtt{DW.KeyGen}$, then the state St_0 is refreshed as St_1 after deriving a pair of session keys, thereafter there must be a refreshing operation on the current state St_i to St_{i-1} when the key derivation mechanism is called.

On the formalization of security models, we formalize the security notions of *full unlinkability* and *standard unforgeability* for deterministic wallets. We first recall the weak notion of *forward unlinkability* [15] (and also be adopted in [3,17]) in which the unlinkability only holds in prior to any hot wallet corruption, the unlinkability only holds in prior to any hot wallet corruption, it means only the keys generated prior to a hot wallet breach (i.e., when the adversary learns the state) cannot be linked to mpk. Therefore, in this setting, not only the initial state St_0 should be kept secret but also the intermediate state St_i should be

used with care. Compared with that, in our full unlinkability model, only the parameter of chaincode ch needs to be kept secret, then all the derived session public keys remain secure. As for the *standard unforgeability*, it guarantees that once funds are sent to the cold wallet must remain secure even if the hot wallet is corrupted, and it is stronger when compared with the unforgeability notion of prior works [15,3,17], since it allows the adversary to query the signing oracle and forge the signature with adversarially chosen randomness.

A New Key Derivation Mechanism: Support One-Way Derivation and Hot/Cold Setting. The *key derivation mechanism* is the most important part of a wallet system, since the performances and properties of a wallet system heavily depend on the key derivation mechanism. In this work, we present a new key derivation mechanism that enable the deterministic wallet to derive keys in one-way and support deploying in hot/cold setting. We achieve that by arming with appropriate lattice techniques. The key ingredient is a lattice basis delegation algorithm ($\mathsf{BasisDel}$) [1]. The $\mathsf{BasisDel}$ is used in the scenario: For a lattice $\mathbf{L}$ with basis $\mathbf{B}$, to delegate a short basis as the key to a child, the parents employ $\mathsf{BasisDel}$ with input $(\mathbf{L}, \mathbf{B})$ to create a new lattice $\mathbf{L}'$ with a random short basis $\mathbf{B}'$. *This delegation process is one-way*, i.e., a child node cannot use its secret key to recover the key of its parent or its siblings. We explain the delegation technique at a high level: let $\mathbf{L}$ be a lattice in $\mathbb{Z}^m$ and let $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$ be a short basis of $\mathbf{L}$. Let $\mathbf{R}$ be a public non-singular matrix in $\mathbb{Z}^{m \times m}$. Then we have the set $\mathbf{B}' = \{\mathbf{R}\mathbf{b}_1, \ldots, \mathbf{R}\mathbf{b}_m\}$ is a basis of the lattice $\mathbf{L}' := \mathbf{L}\mathbf{R}^{-1}$. Moreover, we can use standard tools such as basis randomization algorithm to randomize the basis without increasing the norm of $\mathbf{B}'$ by much. The end result is a random short basis $\mathbf{B}'$ of $\mathbf{L}'$. When instantiating in deterministic wallet setting, we can consider setting the parent lattice $\mathbf{L}$ as mpk, $\mathbf{B}$ as msk, $\mathbf{L}'$ as derived session public key pk', $\mathbf{B}'$ as derived session secret key sk', and set $\mathbf{R}$ as the randomness ρ. Since the key derivation is one-way, our deterministic wallet is immune to the privilege escalation attack.

We can observe the above delegation algorithm has *natural symmetry* property, i.e., when the correct randomness $\mathbf{R}$ is generated both hot and cold wallet can derive the matching session keys by $\mathbf{R}$. This property is essential for deterministic wallets since it requires that there is no interaction between the hold and cold wallet in the process of session key derivation. The state-of-the-art work [3] can not support the natural symmetry property, because the process of session key derivation requires interaction between the hold and cold wallet, so it falls short in building a practical hot/cold deterministic wallet. As we described above, the hot wallet in order to generate the correct session public key pk' corresponds to the session public key sk' in cold wallet, the hot wallet needs to receive the counter ctr from cold wallet. The reason for that is the adopting of rejection sampling [28] in secret key derivation algorithm, i.e., if the session secret key $sk' = msk + \rho'$ has the incorrect distribution then generate the randomness ρ' repeatedly and send the corresponding ctr to hot wallet. In other words, the generated randomness ρ' is worked (the sk' has the correct distribution) in a

probability p that is $p \geq \frac{1}{M}$ where $M = O(1)$. By contrast, in our setting, as shown by Agrawal et al. [1], once the randomness, i.e., the desired matrix $\mathbf{R}$ is generated, it is worked in an overwhelming probability, namely, we can obtain a valid pair of session keys (pk', sk') with overwhelming probability by simply multiply $\mathbf{R}$ and (mpk, msk) that is $(pk' := mpk \cdot \mathbf{R}^{-1}, sk' := \mathsf{Rand}(\mathbf{R} \cdot msk))$ where $\mathsf{Rand}()$ is a randomization operation by a standard tool. As lattice-based assumptions are conjectured to be secure under quantum computer attacks, we obtain the first post-quantum secure deterministic wallet supporting hot/cold setting.

Practical Instantiation and Deployment over Blockchains. In this work, we show our wallet scheme is more practicable by analyzing an instantiation of our wallet scheme based on the signature scheme Falcon [20], and analyze the deployment over blockchains. Falcon is a lattice-based signature scheme of NIST finalist, which is so compact that has the smallest size of "pk + sig", i.e., the size on the sum of public key and signature is smallest among the candidate algorithms for post-quantum (PQ) standardization [35]. Therefore, Falcon is fitted neatly to instantiate wallet schemes, since on the blockchains, we know that the *transaction throughput* of a transaction tx must usually consist of a public key pk, a signature σ, and the raw transaction part raw, i.e., we can simply write that as tx's = raw's + pk's + σ's. To better illustrate our performance, we show a comparison with the state-of-the-art deterministic wallet of Alkadri et al. [3]. Below we show the comparison results in Table 1 while the concrete analysis is given in Section 6.

Table 1. Comparison with prior work

Scheme	More secure	Hot/Cold setting	Security level	Sizes [B]	Transaction throughput	Cycle counts [k-cycles]
[3]	✗	✗	95 bits	pk: 14880 sig: 2592	≈17.5 KB	Sign: 3089.9 Ver: 814.3
Our	✓	✓	158 bits	pk: 897 sig: 666	≈1.66 KB	Sign: 1368.5 Ver: 95.6

To show the comparison comprehensively, we add two items that are "More secure" and "Hot/Cold setting" since both are indispensable for a practical wallet. The "More secure" means this work achieves *stateless*, *immune privilege escalation attack*, *full unlinkability*, and *standard unforgeability*. Note that the work [3] is not "More secure" since none of these notions are achieved as we described above, while our work achieved all of that. From the comparison results, we can observe that our scheme has a more compact performance than the prior work [3] since our works achieved more efficiency on the time-consuming and parameter sizes even under a higher security level.

1.2 Related Work

Research on Wallet Systems. Table 1 gives a comprehensive comparison between our work and the state-of-the-art deterministic wallet [3]. Below we would like to give further details on the comparison with the related work.

With the quick and explosive development of cryptocurrencies, the concept of hot/cold wallets have gained more attention because it is an important tool and primary entrance for its users to access cryptocurrency funds since it stores and manages the cryptographic keys. At the same time, however, the community has noticed the vulnerability on deterministic wallet schemes that once the master public key and one session secret key are compromised, the master secret key and the whole wallet will be compromised. Before the security model for hot/cold wallets had been formalized by Das et al. [15], there are also works aimed to present secure deterministic wallet schemes. For instance, Liu et al. [26] proposed a "identity-based signature" like signature scheme whose inherent properties such as session-key-insulated and master public key privacy-preserving can be employed to eliminate the privilege escalation attack effectively. Then it was transformed to lattice setting by [25], but was proved in random oracle, rather than quantum random oracle. And recently, they transform the signature scheme to a deterministic wallet scheme [27] but did not formalize the models for deterministic wallet and not quantum secure. Until recently, Das et al. [15] first formalized the security model for hot/cold wallets and gave the construction, but it needs to rely on state, i.e., stateful, and its security notions are weaker than ours, and not quantum secure. More recently, Erwig and Riahi [17] presented a novel deterministic wallet which build from a new cryptographic primitive named *adaptor signatures*, but it also adopt the approaches of Das et al. [15], so it is also stateful and weaker security guarantees, and cannot against quantum adversary. There are also researches [23,18,16,14,42] of *hierarchical* variants, i.e., hierarchical deterministic wallets, however, to the best of our knowledge, no hierarchical deterministic wallet in quantum world has been introduced so far.

Other Related Work. Lattice basis delegation technique is the key ingredient among the lattice techniques that our wallet scheme armed. Below we explain why we employ the lattice basis delegation technique by Agrawal et al. [1] rather than other works [10,31]. As aforementioned, the security notion of unlinkability requires that the master public key can not be revealed from session public keys. However, the delegation algorithms from the works [10,31] directly concatenate the master public key in plaintext, rather than randomized the master public key as [1], when deriving session public keys, so the unlinkability is trivially broken. Furthermore, the delegation algorithms of both works [10,31] do not satisfy the "natural symmetry" property which is essential for deterministic wallets as we mentioned above.

2 Preliminaries

Notation. We denote vectors as lower-case bold letters (e.g. $\mathbf{x}$), and matrices by upper-case bold letters (e.g. $\mathbf{A}$). We denote as $e \xleftarrow{\$} \mathcal{S}$ the uniform sampling of the variable e from the set $\mathcal{S}$. We write $[l]$ for a positive integer l to denote the set $\{1, \ldots, l\}$. We say that a function in n is *negligible*, written $\mathrm{negl}(n)$, if it vanishes faster than the inverse of any polynomial in n. We say probability $p(n)$ is *overwhelming* if $1 - p(n)$ is negligible. We denote the horizontal concatenation of two matrices $\mathbf{A}$ and $\mathbf{B}$ as $\mathbf{A} \mid \mathbf{B}$. For a matrix $\mathbf{A}$ we denote some matrix norms: $\|\mathbf{A}\|_1$ denotes the ℓ_1-norm of $\mathbf{A}$, $\|\mathbf{A}\|$ denotes the ℓ_2-norm of the longest column of $\mathbf{A}$, $\|\mathbf{A}\|_\infty$ denotes the ℓ_∞-norm of $\mathbf{A}$, $\|\mathbf{A}\|_{\mathrm{GS}}$ denotes the result of applying Gram-Schmidt orthogonalization to the columns of $\mathbf{A}$. Unless otherwise stated, all our algorithms are probabilistic, we use $y \leftarrow \mathsf{A}(x)$ to denote that algorithm A outputs y when running on input x, and write $y \leftarrow \mathsf{A}(x, \rho)$ to denote algorithm A outputs y when running on input x and randomness ρ. Note that in this way, algorithm A becomes a deterministic algorithm. We use the notation $\mathsf{A}(x)$ to denote the set of all possible outputs of (probabilistic) algorithm A on input x. We model hash functions as classical random oracles [7] or quantum random oracles [8]. We denote the classical random oracle by the symbol H and the quantum random oracle by the notation $|\mathsf{H}\rangle$. In our proofs, we will also consider reprogrammed random oracles. For a (quantum) random oracle H we write $\mathsf{H}_{x \to y}$ for the (quantum) random oracle that is reprogrammed on input x to y.

2.1 Quantum Random Oracle Model

In this section, we recall the quantum random oracle model and the existing results that we will use. The *quantum random oracle model* (QROM) is introduced by Boneh et al. [8] which is motivated by the observation that the *random oracle model* (ROM) is not appropriate in the post-quantum setting. In the real world, an adversary equipped with a quantum computer is able to implement the hash function and evaluate it in superposition. In QROM, parties with quantum computing power get access to the oracle $|\mathsf{H}\rangle$ where $|\mathsf{H}\rangle : |x, y\rangle \mapsto |x, y \oplus \mathsf{H}(x)\rangle$. Below we describe a result for quantum random oracles that are required for our proofs. As [3] mentioned, the one-way to hiding (O2H) lemma [39] is an important tool for security proofs in the quantum random oracle model. It gives bounds on the advantage of an adversary in distinguishing between different random oracles when the adversary is allowed to query them in superposition. Below we state the O2H lemma using the reformulation by Ambainis et al. [6].

Lemma 1 (One-way to hiding (O2H) [6]). *Let $\mathsf{G}, \mathsf{H} : \mathcal{X} \to \mathcal{Y}$ be random functions, let z be a random value, and let $\mathcal{S} \subset \mathcal{X}$ be a random set such that $\forall x \notin \mathcal{S}$, $\mathsf{G}(x) = \mathsf{H}(x)$. $(\mathsf{G}, \mathsf{H}, \mathcal{S}, z)$ may have arbitrary joint distribution. Furthermore, let $\mathcal{A}^{|\mathsf{H}\rangle}$ be a quantum oracle algorithm which queries $|\mathsf{H}\rangle$ at most q times. Let* Ev *be an arbitrary classical event. Define an oracle algorithm $\mathcal{B}^{|\mathsf{H}\rangle}$ as follows: Pick $i \xleftarrow{\$} [q]$. Run $\mathcal{A}^{|\mathsf{H}\rangle}(z)$ until just before its i-th round of queries to $|\mathsf{H}\rangle$. Measure*

the query in the computational basis, and output the measurement outcome. It holds that

$$\left| \Pr[\mathsf{Ev} : \mathcal{A}^{|\mathsf{H}\rangle}(z)] - \Pr[\mathsf{Ev}{:}\mathcal{A}^{|\mathsf{G}\rangle}(z)] \right| \leq 2q\sqrt{\Pr[x \in \mathcal{S} : \mathcal{B}^{|\mathsf{H}\rangle}(z) \Rightarrow x]}$$

2.2 Signature Schemes

In this section, we introduce the syntax and relevant security notions for signature schemes.

Definition 1 (Signature Scheme). *A signature scheme* $\mathtt{SIG}$ *is a triple of algorithms* $\mathtt{SIG} := (\mathtt{SIG.KeyGen}, \mathtt{SIG.Sign}, \mathtt{SIG.Ver})$*. The probabilistic algorithm of key generation* $\mathtt{SIG.KeyGen}$ *that outputs a key pair* (pk, sk) *of secret and public keys. The probabilistic algorithm of signing* $\mathtt{SIG.Sign}$ *takes as input a secret key* sk *and a message and returns a signature* σ*. The deterministic algorithm of verification* $\mathtt{SIG.Ver}$ *takes as input a public key* pk*, a signature* σ*, and a message* μ*. It returns 1 if signature* σ *is valid and 0 otherwise. The correctness requires that, let* λ *be the security parameter, for all key pairs* $(pk, sk) \leftarrow \mathtt{SIG.KeyGen}(1^\lambda)$*, all* $\mu \in \{0,1\}^*$*, and all* $\sigma \leftarrow \mathtt{SIG.Sign}(sk, \mu)$*, we have the following holds (over all the randomness in the experiment).*

$$\Pr\left[\mathtt{SIG.Ver}(pk, \mu, \mathtt{SIG.Sign}(sk, \mu)) = 1\right] \geq 1 - \mathrm{negl}(\lambda)$$

We will use an augmented notion of signature schemes with rerandomizable public keys from Alkadri et al. [3], which is a relaxed version of the notion of signature schemes with rerandomizable keys from Fleischhacker et al. [19] since it holds only for the generated public keys, but not in case of secret keys. In our setting, this notion is augmented with an additional algorithm $\mathtt{RKS.RandGen}$. The notion is described as follows.

Definition 2 (Signature Scheme with Rerandomizable Public Keys). *A signature scheme with rerandomizable public keys* $\mathtt{RKS}$ *is a tuple of algorithms* $\mathtt{RKS} := (\mathtt{RKS.KeyGen}, \mathtt{RKS.RandGen}, \mathtt{RKS.RandPK}, \mathtt{RKS.RandSK}, \mathtt{RKS.Sign}, \mathtt{RKS.Ver})$ *where algorithms* $(\mathtt{RKS.KeyGen}, \mathtt{RKS.Sign}, \mathtt{RKS.Ver})$ *satisfy the definition of a standard signature scheme as defined above (cf. Definition of* $\mathtt{SIG}$*). For randomness space* $\mathcal{R}$*, the randomness generation algorithm* $\mathtt{RKS.RandGen}$ *takes as input a parameter* γ *and outputs a randomness* $\rho \in \mathcal{R}$*. The public key rerandomization algorithm* $\mathtt{RKS.RandPK}$ *takes as input the public key* pk *and a randomness* $\rho \in \mathcal{R}$ *and outputs a randomized public key* pk'*, while the secret key rerandomization algorithm* $\mathtt{RKS.RandSK}$ *takes as input the secret key* sk *and a randomness* $\rho \in \mathcal{R}$ *and outputs a randomized secret key* sk'*.*

The $\mathtt{RKS}$ has a property that is rerandomizability of public keys, which ensures the distributions of the output rerandomized public keys are identical. In our setting, we adopt the relaxed version of that property, the distributions of the output rerandomized public keys are statistically indistinguishable.

Rerandomizability of public keys: *For all public keys* $(pk, \cdot) \leftarrow \texttt{RKS.KeyGen}(1^\lambda)$ *and all* $\rho \in \mathcal{R}$*, the distributions of* pk' *and* pk'' *are statistically indistinguishable, where* pk'' *from* $(pk'', \cdot) \leftarrow \texttt{RKS.KeyGen}(1^\lambda)$ *and* $pk' \leftarrow \texttt{RKS.RandPK}(pk, \rho)$*.*

Correctness: *For all* $\lambda \in \mathbb{N}$*,* $(pk, sk) \leftarrow \texttt{RKS.KeyGen}(1^\lambda)$*,* $\mu \in \mathcal{M}$*, and all* $\sigma \leftarrow \texttt{RKS.Sign}(sk, \mu)$*, it holds that*

$$\Pr\left[\texttt{RKS.Ver}(pk, \mu, \texttt{RKS.Sign}(sk, \mu)) = 1\right] \geq 1 - \mathrm{negl}(\lambda),$$

and for all $\rho \leftarrow \texttt{RKS.RandGen}(\gamma)$*, for a pair of rerandomized keys* $pk' \leftarrow \texttt{RKS.RandPK}(pk, \rho)$ *and* $sk' \leftarrow \texttt{RKS.RandSK}(sk, \rho)$*, it holds that*

$$\Pr\left[\texttt{RKS.Ver}(pk', \mu, \texttt{RKS.Sign}(sk', \mu)) = 1\right] \geq 1 - \mathrm{negl}(\lambda).$$

In this work we will use the standard security notion of existential unforgeability under adaptive chosen-message attacks [22] (EUF-CMA). Below we formalize the EUF-CMA security notion for a signature scheme SIG in Figure 1.

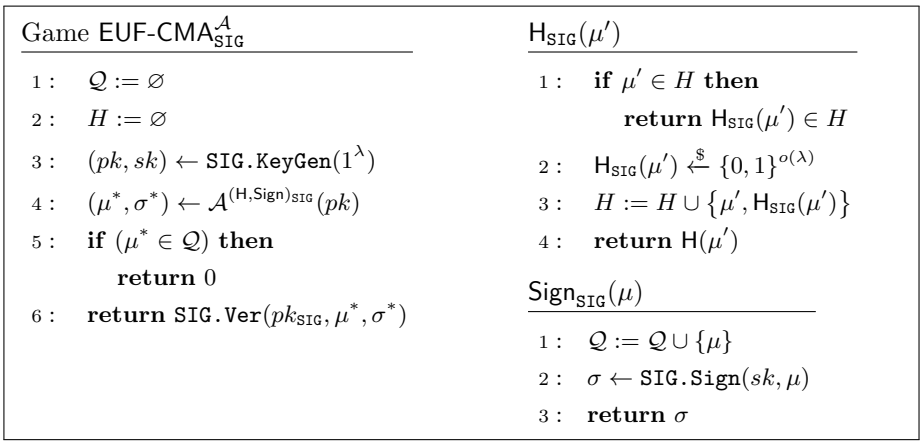

Fig. 1. The security game EUF-CMA of signature schemes

Definition 3 (EUF-CMA Security). *Let* $\mathsf{H}_{\mathtt{SIG}} : \{0,1\}^* \to \{0,1\}^{o(\lambda)}$ *be a hash function modeled as (quantum) random oracle. A signature scheme* SIG *is called* $(t, q_{\mathtt{Sign}}, q_{\mathsf{H}}, \varepsilon)$-EUF-CMA *in the (quantum) random oracle if for any adversary* $\mathcal{A}$ *running in time at most* t *and making at most* $q_{\mathtt{Sign}}$ *signature queries and at most* q_{H} *(superposition) queries to* $\mathsf{H}_{\mathtt{SIG}}$*, the game* $\mathsf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathtt{SIG}}$ *depicted in Figure 1 outputs 1 with probability at most* ε*, i.e.,* $\Pr[\mathsf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathtt{SIG}} = 1] \leq \varepsilon$*.*

Definition 4 (EUF-CMA-RK Security). *Let* $\mathsf{H}_{\mathtt{RKS}} : \{0,1\}^* \to \{0,1\}^{o(\lambda)}$ *be a hash function modeled as (quantum) random oracle. Let* $\mathcal{R}$ *be the randomness space. A signature scheme with rerandomizable (public) keys* RKS *is*

Game $\mathsf{EUF\text{-}CMA\text{-}RK}^{\mathcal{A}}_{\mathtt{RKS}}$

1 : $\mathcal{Q} := \varnothing$
2 : $H := \varnothing$
3 : $(pk, sk) \leftarrow \mathtt{RKS.KeyGen}(1^\lambda)$
4 : $(\mu^*, \sigma^*, \rho^*) \leftarrow \mathcal{A}^{(\mathsf{H},\mathsf{Sign})_{\mathtt{RKS}}}(pk)$
5 : **if** $(\mu^* \in \mathcal{Q})$ **or** $(\rho^* \notin \mathcal{R})$ **then** **return** 0
6 : $pk' \leftarrow \mathtt{RKS.RandPK}(pk, \rho^*)$
7 : **return** $\mathtt{RKS.Ver}(pk', \mu^*, \sigma^*)$

$\mathsf{H}_{\mathtt{RKS}}(\mu')$

(see Figure 1)

$\mathsf{Sign}_{\mathtt{RKS}}(\mu, \rho)$

1 : $\mathcal{Q} := \mathcal{Q} \cup \{\mu\}$
2 : **if** $(\rho \notin \mathcal{R})$ **then return** $\perp$
3 : $sk' \leftarrow \mathtt{RKS.RandSK}(sk, \rho)$
4 : $\sigma \leftarrow \mathtt{RKS.Sign}(sk', \mu)$
5 : **return** σ

Fig. 2. The security game EUF-CMA-RK of signature schemes with rerandomizable (public) keys

called $(t, q_{\mathtt{Sign}}, q_{\mathsf{H}}, \varepsilon)$-EUF-CMA-RK *in the (quantum) random oracle if for any adversary* $\mathcal{A}$ *running in time at most* t *and making at most* $q_{\mathtt{Sign}}$ *signature queries and at most* q_{H} *(quantum) random oracle queries to* $\mathsf{H}_{\mathtt{RKS}}$*, the game* $\mathsf{EUF\text{-}CMA\text{-}RK}^{\mathcal{A}}_{\mathtt{RKS}}$ *depicted in Figure 1 outputs 1 with probability at most* ε*, i.e.,* $\Pr[\mathsf{EUF\text{-}CMA\text{-}RK}^{\mathcal{A}}_{\mathtt{RKS}} = 1] \leq \varepsilon$.

2.3 Lattices and Gaussian distributions

Let $m \in \mathbb{Z}$ be a positive integer and $\mathbf{\Lambda} \subset \mathbb{R}^m$ be an m-dimensional full-rank lattice formed by the set of all integral combinations of m linearly independent basis vectors $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_m) \subset \mathbb{Z}^m$, i.e., $\mathbf{\Lambda} = \mathcal{L}(\mathbf{B}) = \left\{\mathbf{Bc} = \sum_{i=1}^m c_i \mathbf{b}_i : \mathbf{c} \in \mathbb{Z}^m\right\}$. For positive integers n, m, q, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and a vector $\mathbf{u} \in \mathbb{Z}_q^m$, the m-dimensional integer lattice $\mathbf{\Lambda}_q^\perp(\mathbf{A})$ is defined as $\mathbf{\Lambda}_q^\perp(\mathbf{A}) = \{\mathbf{e} \in \mathbb{Z}^m : \mathbf{Ae} = \mathbf{0} \pmod q\}$. $\mathbf{\Lambda}_q^{\mathbf{u}}(\mathbf{A})$ is defined as $\mathbf{\Lambda}_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{e} \in \mathbb{Z}^m : \mathbf{Ae} = \mathbf{u} \pmod q\}$. It holds that if $\mathbf{t} \in \mathbf{\Lambda}_q^{\mathbf{u}}(\mathbf{A})$ then $\mathbf{\Lambda}_q^{\mathbf{u}}(\mathbf{A}) = \mathbf{\Lambda}_q^\perp(\mathbf{A}) + \mathbf{t}$ and hence $\mathbf{\Lambda}_q^{\mathbf{u}}(\mathbf{A})$ is a coset of $\mathbf{\Lambda}_q^\perp(\mathbf{A})$. The discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^n,s,\mathbf{c}}$ over $\mathbb{Z}^n$ with standard deviation $s > 0$ and center $\mathbf{c} \in \mathbb{R}^n$ is defined as follows: For every $\mathbf{x} \in \mathbb{Z}^n$ the probability of $\mathbf{x}$ is given by $\mathcal{D}_{\mathbb{Z}^n,s,\mathbf{c}}(\mathbf{x}) = \rho_{s,\mathbf{c}}(\mathbf{x})/\rho_{s,\mathbf{c}}(\mathbb{Z}^n)$, where $\rho_{s,\mathbf{c}}(\mathbf{x}) = \exp(-\pi\|\mathbf{x}-\mathbf{c}\|/s^2)$ and $\rho_{s,\mathbf{c}}(\mathbb{Z}^n) = \sum_{\mathbf{x}\in\mathbb{Z}^n} \rho_{s,\mathbf{c}}(\mathbf{x})$. For simplicity, $\rho_{s,\mathbf{0}}$ and $\mathcal{D}_{\mathbb{Z}^n,s,\mathbf{0}}$ are abbreviated as ρ_s and $\mathcal{D}_{\mathbb{Z}^n,s}$, respectively.

Properties of Gaussian Distributions: The following lemma from [21] captures some properties of Gaussian distributions. The property-1 gives a norm bound on the Gaussian sampled preimage vector. The property-2 will be used to make our RKS scheme satisfy the rerandomizability of public keys (see Section 5). The property-3, i.e., SampleGaussian algorithm is the basic algorithm which is called by our several algorithms such as SampleR, SamplePre, and RandBasis. The property-4, i.e., SamplePre algorithm will be mainly used in the signing algorithm of our LHS scheme (see Section 2.5).

Lemma 2 ([21]). *Let $q \geq 2$ and let $\mathbf{A}$ be a matrix in $\mathbb{Z}_q^{n\times m}$ with $m > n$. Let $\mathbf{B}$ be a basis of $\mathbf{\Lambda}_q^{\perp}(\mathbf{A})$ and standard deviation $s \geq \|\mathbf{B}\|_{\mathrm{GS}} \cdot \omega(\sqrt{\log m})$. Then for $\mathbf{c} \in \mathbb{R}^m$ and $\mathbf{u} \in \mathbb{Z}_q^n$:*

1. $\Pr_{\mathbf{e}\leftarrow D_{\mathbf{\Lambda}_{\mathbf{u}}^{\perp}(\mathbf{A}),s}}\left[\ \|\mathbf{e}\| \geq s\cdot\sqrt{m}\ \right] \leq \mathrm{negl}(n)$.
2. *For $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}$, the distribution of $\mathbf{u} = \mathbf{A}\mathbf{e} \in \mathbb{Z}_q^n$ is within negligible statistical distance of uniform over $\mathbb{Z}_q^n$.*
3. *There is a probabilistic algorithm $\mathsf{SampleGaussian}(\mathbf{A}, \mathbf{B}, s, \mathbf{c})$ that outputs $\mathbf{e} \in \mathbf{\Lambda}_q^{\perp}(\mathbf{A})$ drawn from a distribution statistically close to $\mathcal{D}_{\mathbf{\Lambda},s,\mathbf{c}}$.*
4. *There is a probabilistic algorithm $\mathsf{SamplePre}(\mathbf{A}, \mathbf{B}, \mathbf{u}, s)$ that outputs $\mathbf{e} \in \mathbf{\Lambda}_q^{\mathbf{u}}(\mathbf{A})$ drawn from a distribution statistically close to $\mathcal{D}_{\mathbf{\Lambda}_q^{\mathbf{u}}(\mathbf{A}),s,\mathbf{c}}$.*

Due to $\mathbf{\Lambda}_q^{\mathbf{u}}(\mathbf{A}) = \mathbf{\Lambda}_q^{\perp}(\mathbf{A})+\mathbf{t}$ for some $\mathbf{t} \in \mathbf{\Lambda}_q^{\mathbf{u}}(\mathbf{A})$, algorithm $\mathsf{SamplePre}(\mathbf{A}, \mathbf{B}, \mathbf{u}, s)$ works by calling $\mathsf{SampleGaussian}(\mathbf{A}, \mathbf{B}, s, \mathbf{t})$ and then subtracts $\mathbf{t}$ from the result.

The following lemma shows how to sample an essentially uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ with an associated basis $\mathbf{B}$ of $\mathbf{\Lambda}_q^{\perp}(\mathbf{A})$ with low Gram-Schmidt norm. The $\mathsf{BasisGen}$ algorithm will be used to sample the master public key mpk and master secret key msk, i.e., master key pair (mpk, msk) in the key generation phase of our wallet scheme.

Lemma 3 (Lattice Basis Generation Algorithm [5,31]). *There is a fixed constant $C > 1$ and a probabilistic polynomial-time algorithm $\mathsf{BasisGen}(1^n, 1^m, q)$ that, for polynomial bounded $m \geq Cn\log q$, the algorithm outputs $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{B} \in \mathbb{Z}^{m\times m}$ such that: $\mathbf{B}$ is a basis of $\mathbf{\Lambda}_q^{\perp}(\mathbf{A})$, the distribution of $\mathbf{A}$ is within negligible statistical distance of uniform, and $\|\mathbf{B}\|_{\mathrm{GS}} \leq L_{\mathrm{TG}} = O(\sqrt{n\log q})$.*

2.4 Lattice Basis Delegation

We now review the *lattice basis delegation* algorithm $\mathsf{BasisDel}$ of Agrawal et al. [1] which is the core procedure of the key derivation algorithm of our wallet scheme. The $\mathsf{BasisDel}$ algorithm consists of three parts: a full-rank matrix $\mathbf{R}$ in $\mathbb{Z}^{m\times m}$ where all the columns of $\mathbf{R}$ are low-norm, $\mathsf{ToBasis}$ algorithm from [30], and $\mathsf{RandBasis}$ algorithm from [10]. The following algorithm from [1] shows how to sample the desired $\mathbf{R}$.

Lemma 4 (SampleR Algorithm [1]). *Define $s_{\mathrm{R}} := L_{\mathrm{TG}} \cdot \omega(\sqrt{\log m})$. Let $\mathbf{T}$ be the canonical basis of the lattice $\mathbb{Z}^m$. There is a probabilistic polynomial-time algorithm $\mathsf{SampleR}(1^m)$ which outputs a full-rank matrix $\mathbf{R}$ in $\mathbb{Z}^{m\times m}$ by invoking $\mathsf{SampleGaussian}(\mathbb{Z}^m, \mathbf{T}, s_{\mathrm{R}}, \mathbf{0})$.*

The following algorithm shows that a full-rank set $\mathbf{S}$ of a lattice $\mathbf{\Lambda}$ can be converted into a basis $\mathbf{T}$ for $\mathbf{\Lambda}$ with an equally low Gram-Schmidt norm.

Lemma 5 (ToBasis Algorithm [30]). *There is a deterministic polynomial-time algorithm* $\mathsf{ToBasis}(\mathbf{S}, \mathbf{B})$ *which takes as input a basis* $\mathbf{B}$ *and a full-rank set (not necessarily a basis)* $\mathbf{S}$ *of lattice vectors in* $\mathbf{\Lambda} = \mathcal{L}(\mathbf{B})$*, outputs a basis* $\mathbf{T}$ *of* $\mathbf{\Lambda}$ *such that* $\|\mathbf{t}_i\|_{\mathrm{GS}} \leq \|\mathbf{s}_i\|_{\mathrm{GS}}$ *for all* i*.*

The following algorithm is used to randomize a lattice basis, which is useful for our wallet scheme when securely delegating the control on asset to another entity, because the resulting basis is still short, but is essentially statistically independent of the original basis.

Lemma 6 (RandBasis Algorithm [10]). *There is a probabilistic polynomial-time algorithm* $\mathsf{RandBasis}(\mathbf{B}, s)$ *which takes as input a basis* $\mathbf{B}$ *of* $\mathbf{\Lambda}_q^{\perp}(\mathbf{A})$ *and a standard deviation* $s \geq \|\mathbf{B}\|_{\mathrm{GS}} \cdot \omega(\sqrt{\log m})$*, outputs a new basis* $\mathbf{B}'$ *of* $\mathbf{\Lambda}_q^{\perp}(\mathbf{A})$ *such that* $\|\mathbf{B}'\|_{\mathrm{GS}} \leq s\sqrt{m}$ *holds with overwhelming probability. Furthermore, the distribution of* $\mathbf{B}'$ *is statistically independent with the original basis* $\mathbf{B}$*.*

Next, we introduce the lattice basis delegation algorithm $\mathsf{BasisDel}$ which is composed of the above-introduced matrix $\mathbf{R}$ and algorithms $\mathsf{ToBasis}$ and $\mathsf{RandBasis}$.

Lattice Basis Delegation Algorithm: $\mathsf{BasisDel}(\mathbf{A}, \mathbf{B}, \mathbf{R}, s)$

Inputs: A matrix $\mathbf{A}$ in $\mathbb{Z}_q^{n \times m}$, a basis $\mathbf{B}$ of $\mathbf{\Lambda}_q^{\perp}(\mathbf{A})$, a full-rank matrix $\mathbf{R}$ in $\mathbb{Z}^{m \times m}$, and a standard deviation s.

Outputs: Let $\mathbf{A}' = \mathbf{A}\mathbf{R}^{-1}$. The algorithm outputs a basis $\mathbf{B}'$ of $\mathbf{\Lambda}_q^{\perp}(\mathbf{A}')$.

The $\mathsf{BasisDel}$ algorithm runs as follows:

1. Let $\mathbf{B} = \left[\ \mathbf{b}_1 \mid \cdots \mid \mathbf{b}_m\ \right] \subseteq \mathbb{Z}^m$. Compute $\mathbf{S} = \left[\ \mathbf{R}\mathbf{b}_1 \mid \cdots \mid \mathbf{R}\mathbf{b}_m\ \right] \subseteq \mathbb{Z}^m$. Due to $\mathbf{B}$ is a basis of $\mathbf{\Lambda}_q^{\perp}(\mathbf{A})$, $\mathbf{R}$ is full-rank, and $\mathbf{A}'\mathbf{S} = \mathbf{0} \pmod q$, therefore, $\mathbf{S}$ is a set of independent vectors in $\mathbf{\Lambda}_q^{\perp}(\mathbf{A}')$.
2. Call $\mathsf{ToBasis}$ by taking as input the set $\mathbf{S}$ and an arbitrary basis of $\mathbf{\Lambda}_q^{\perp}(\mathbf{A}')$ and then outputs a basis $\bar{\mathbf{B}}$ whose Gram-Schmidt norm is no more than that of $\mathbf{S}$.
3. Call $\mathsf{RandBasis}$ by taking as input $\bar{\mathbf{B}}$ and standard deviation s then outputs the resulting basis $\mathbf{B}'$ of $\mathbf{\Lambda}_q^{\perp}(\mathbf{A}')$.

Lemma 7 (BasisDel Algorithm [1]**).** *There is a probabilistic polynomial-time algorithm* $\mathsf{BasisDel}(\mathbf{A}, \mathbf{B}, \mathbf{R}, s)$ *which takes as input a matrix* $\mathbf{A}$ *in* $\mathbb{Z}_q^{n \times m}$*, a basis* $\mathbf{B}$ *of* $\mathbf{\Lambda}_q^{\perp}(\mathbf{A})$*, a full-rank matrix* $\mathbf{R}$ *in* $\mathbb{Z}^{m \times m}$*, and a standard deviation* $s > \|\mathbf{B}\|_{\mathrm{GS}}\sqrt{nm\log q} \cdot \omega(\log^2 m)$*. Let* $\mathbf{A}' = \mathbf{A}\mathbf{R}^{-1}$*. The algorithm outputs a basis* $\mathbf{B}'$ *of* $\mathbf{\Lambda}_q^{\perp}(\mathbf{A}')$ *such that* $\|\mathbf{B}'\|_{\mathrm{GS}}/\|\mathbf{B}\|_{\mathrm{GS}} \leq m^{3/2}\omega(\log^2 m)$*. Furthermore, the distribution of* $\mathbf{B}'$ *is statistically close to the distribution* $\mathsf{RandBasis}(\mathbf{T}, s)$ *where* $\mathbf{T}$ *is an arbitrary basis of* $\mathbf{\Lambda}_q^{\perp}(\mathbf{A}')$ *satisfying* $\|\mathbf{T}\|_{\mathrm{GS}} \leq s/\omega(\sqrt{\log m})$*.*

2.5 Lattice-Based Hash-and-Sign Signatures

In this section, we review a generic construction of lattice-based hash-and-sign (LHS) signatures. The signature scheme $\texttt{LHS} := (\texttt{LHS.KeyGen}, \texttt{LHS.Sign}, \texttt{LHS.Ver})$ is formally described in Figure 3. The key generation algorithm `LHS.KeyGen` generates an instance of a computationally hard lattice problem called *short integer solution* (SIS) [2] or its special variants such as Ring-SIS [29] and Module-SIS [24]. In essence, the `LHS.KeyGen` algorithm is a lattice basis (trapdoor) generation algorithm $\mathsf{BasisGen}$ (cf. Lemma 3) which outputs $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{B} \in \mathbb{Z}^{m\times m}$ such that $\mathbf{B}$ is a lattice basis of $\mathbf{\Lambda}_q^{\perp}(\mathbf{A})$, the distribution of $\mathbf{A}$ is within negligible statistical distance of uniform, and $\mathbf{B}$ has a low Gram-Schmidt norm. The `LHS.Sign` algorithm mainly consists of two components, a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_q^n$ which is modeled as a random oracle, and a Gaussian preimage sampling algorithm $\mathsf{SamplePre}$ (cf. item 4 of Lemma 2). The $\mathsf{SamplePre}$ algorithm is able to sample a low-norm preimage $\mathbf{e} \leftarrow \mathsf{SamplePre}(\mathbf{B}, \mathsf{H}(\mu, \mathbf{r}))$ such that $\mathbf{Ae} = \mathsf{H}(\mu, \mathbf{r})$, and the distribution on $\mathbf{e}$ is statistically close to distribution $\mathcal{D}_{\mathbf{\Lambda}_q^{\mathbf{u}}(\mathbf{A}),s}$. A signature consists of two items $(\mathbf{e}, \mathbf{r})$: Gaussian sampled preimage $\mathbf{e}$ and a random vector $\mathbf{r}$. The verification algorithm `LHS.Ver` checks if $\mathbf{Ae} = \mathsf{H}(\mu, \mathbf{r})$, $\mathbf{r} \in \{0,1\}^{o(\lambda)}$, and if $\mathbf{e} \in \mathcal{D}_{\mathbf{\Lambda}_q^{\mathbf{u}}(\mathbf{A}),s}$ for a specified Gaussian parameter (standard deviation) s, otherwise the other $\mathbf{e}'$ is easy to find such that $\mathbf{Ae}' = \mathsf{H}(\mu, \mathbf{r})$ and hence the unforgeability is broken. For the correctness, by the property of $\mathsf{SamplePre}$ (cf. Item 4 of Lemma 2), $\mathbf{e} \in \mathcal{D}_{\mathbf{\Lambda}_q^{\mathbf{u}}(\mathbf{A}),s}$ and hence $\mathbf{Ae} = \mathbf{u} = \mathsf{H}(\mu, \mathbf{r})$ holds with overwhelming probability. Finally, the $\mathsf{EUF\text{-}CMA}$ security of lattice-based hash-and-sign signatures in QROM was analyzed in several works [8,20,11].

$\texttt{LHS.KeyGen}(1^\lambda)$	$\texttt{LHS.Sign}(sk, \mu)$	$\texttt{LHS.Ver}(pk, \mu, (\mathbf{e}, \mathbf{r}))$
1 : $(\mathbf{A}, \mathbf{B}) \leftarrow \mathsf{GenTrap}(1^\lambda)$	1 : $\mathbf{r} \xleftarrow{\$} \{0,1\}^{o(\lambda)}$	1: **if** $\mathbf{e} \in \mathcal{D}_{\mathbf{\Lambda}_q^{\mathbf{u}}(\mathbf{A}),s}$
2 : $pk := \mathbf{A}, sk := \mathbf{B}$	2 : $\mathbf{u} = \mathsf{H}(\mu, \mathbf{r})$	$\wedge\ \mathbf{r} \in \{0,1\}^{o(\lambda)}$
3 : **return** (pk, sk)	3 : $\mathbf{e} \leftarrow \mathsf{SamplePre}(\mathbf{B}, \mathbf{u})$	$\wedge\ \mathbf{Ae} = \mathsf{H}(\mu, \mathbf{r})$
	4 : **return** $(\mathbf{e}, \mathbf{r})$	**then return** 1
		2: **return** 0

Fig. 3. A formal description of a generic hash-and-sign signature scheme from lattice assumptions.

3 The Definitions for Deterministic Wallets

In this section, we formalize the syntax and security models for deterministic wallets. At a high level, a deterministic wallet consists of two entities (cold wallet and hot wallet), and a key derivation mechanism. Specifically, upon initialization of the scheme, the cold wallet generates a master key pair (mpk, msk) and a

chaincode ch and forwards (mpk, ch) to the hot wallet. Then cold wallet and hot wallet can use the key derivation algorithms `SKDer` and `PKDer`, respectively, to derive an arbitrary number of session key pairs, without further interaction, from the master key pair (mpk, msk) and an identity set. The correctness requires that if the cold wallet and the hot wallet derive session key pairs on the same set of identities $ID_0, \ldots, ID_{N-1} \in \{0,1\}^*$ and in the same order, any signature created under one of the signing keys from cold wallet should be correctly verified under the corresponding verification key from hot wallet.

Definition 5. *A deterministic wallet scheme is a tuple of algorithms* $\mathtt{DW} := (\mathtt{DW.KeyGen}, \mathtt{DW.RandGen}, \mathtt{DW.PKDer}, \mathtt{DW.SKDer}, \mathtt{DW.Sign}, \mathtt{DW.Ver})$*, which are defined as follows:*

- `DW.KeyGen`: *The master key generation algorithm takes as input a security parameter* λ *and outputs a master key pair* (mpk, msk) *as well as a chaincode* ch*.*
- `DW.RandGen`: *The randomness generation algorithm takes as input a chaincode* ch *and an identity* ID*, then outputs a randomness* ρ_{ID}*.*
- `DW.PKDer`: *The public key derivation algorithm takes as input a master public key* mpk *and a randomness* ρ_{ID}*, then outputs a session public key* pk_{ID}*.*
- `DW.SKDer`: *The secret key derivation algorithm takes as input a master secret key* msk *and a randomness* ρ_{ID}*, then outputs a session secret key* sk_{ID}*.*
- `DW.Sign`: *The signing algorithm takes as input a session secret key* sk_{ID} *for some identity* ID *and a message* μ *and outputs a signature* σ*.*
- `DW.Ver`: *The verification algorithm takes as input a session public key* pk_{ID} *for some* ID*, a message* μ*, and a signature* σ *and outputs 1 if* σ *is a valid signature for* μ *under public key* pk_{ID}*. It outputs 0 otherwise.*

Definition 6 (Correctness). *For* $N \in \mathbb{N}$*, any* $(mpk, msk, ch) \leftarrow \mathtt{DW.KeyGen}(1^\lambda)$*, any* $\boldsymbol{ID} := (ID_1, \ldots, ID_N) \in \{0,1\}^*$*, and any* $\rho_{ID_i} \leftarrow \mathtt{DW.RandGen}(ch, ID_i)$*, we define the sequence* (pk_i, sk_i) *for* $1 \leq i \leq N$ *as*

$$pk_i := \mathtt{DW.PKDer}(mpk, \rho_{ID_i}), \quad sk_i := \mathtt{DW.SKDer}(msk, \rho_{ID_i}).$$

`DW` *is correct if for all* $\mu \in \{0,1\}^*$*, it holds that*

$$\Pr\left[\mathtt{DW.Ver}(pk_i, \mu, \mathtt{DW.Sign}(sk_i, \mu)) = 1\right] \geq 1 - \mathrm{negl}(\lambda)$$

A deterministic wallet scheme `DW` should satisfy the following two security properties - *wallet unlinkability* and *wallet unforgeability.*

3.1 Wallet Unlinkability

As works [15,3] described, the unlinkability property guarantees that the different public session keys that were derived from the same master public key

should be unlinkable. Formally, it requires that, given the master public key, the distribution of session public keys is computationally indistinguishable from session public keys that are generated from a fresh master public key and chain code. The full unlinkability game for deterministic wallet DWUNL is presented in Figure 4. Initially, the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ runs its subprocedure $\mathcal{A}_1$ on input a master public key mpk generated by $\mathtt{DW.KeyGen}(1^\lambda)$ and subsequently interacts with oracles PK and $\mathsf{Sign}_{\mathtt{DW}}$ that reflects $\mathcal{A}$'s ability. The oracle PK takes as input an identity ID and outputs the corresponding session public key pk_{ID}, and stores pk_{ID} in an array $Keys[ID]$, which captures that $\mathcal{A}$ has the ability to observe transactions stored on the blockchain that transfer money to pk_{ID}. The oracle $\mathsf{Sign}_{\mathtt{DW}}$ takes as input a message μ and a randomness ρ, and outputs the corresponding signature if $\rho \in \mathcal{R}$. It captures that $\mathcal{A}$ has the ability to obtain signatures with respect to a rerandomized public key that is assigned by himself. After the query phase of $\mathcal{A}_1$, a challenge identity ID^* is provided. Note that we use the array $Keys[ID]$ to store session public key pk_{ID} w.r.t. the key value ID. If there is no record for ID^*, the game proceeds to the challenge phase, otherwise return 0. Then the adversary $\mathcal{A}_2$ will be given a challenge public key $pk^b_{ID^*}$ which is either real or random, namely, it depends on tuple (mpk, msk, ch) or a tuple $(\widehat{mpk}, \widehat{msk}, \widehat{ch})$ that was sampled freshly and independently with prior one. $\mathcal{A}_2$'s goal is to distinguish these two scenarios.

Game DWUNL

1 : $(mpk, msk, ch) \leftarrow \mathtt{DW.KeyGen}(1^\lambda)$
2 : $ID^* \leftarrow \mathcal{A}_1^{\mathsf{Sign}_{\mathtt{DW}}, \mathsf{PK}}(mpk)$
3 : **if** $(Keys[ID^*] \neq \perp)$ **then**
  return 0
4 : $\rho_{ID^*} \leftarrow \mathtt{DW.RandGen}(ch, ID^*)$
5 : $pk^0_{ID^*} \leftarrow \mathtt{DW.PKDer}(mpk, \rho_{ID^*})$
6 : $sk^0_{ID^*} \leftarrow \mathtt{DW.SKDer}(msk, \rho_{ID^*})$
7 : $(\widehat{mpk}, \widehat{msk}, \widehat{ch}) \leftarrow \mathtt{DW.KeyGen}(1^\lambda)$
8 : $\widehat{\rho_{ID^*}} \leftarrow \mathtt{DW.RandGen}(\widehat{ch}, ID^*)$
9 : $pk^1_{ID^*} \leftarrow \mathtt{DW.PKDer}(\widehat{mpk}, \widehat{\rho_{ID^*}})$
10 : $sk^1_{ID^*} \leftarrow \mathtt{DW.SKDer}(\widehat{msk}, \widehat{\rho_{ID^*}})$
11 : $b \xleftarrow{\$} \{0, 1\}$
12 : $Keys[ID^*] \leftarrow (pk^b_{ID^*}, sk^b_{ID^*})$
13 : $b' \leftarrow \mathcal{A}_2^{\mathsf{Sign}_{\mathtt{DW}}, \mathsf{PK}}(mpk, pk^b_{ID^*})$
14 : **return** $b' = b$

Oracle $\mathsf{PK}(ID)$

1 : $\rho_{ID} \leftarrow \mathtt{DW.RandGen}(ch, ID)$
2 : $pk_{ID} \leftarrow \mathtt{DW.PKDer}(mpk, \rho_{ID})$
3 : $Keys[ID] \leftarrow \{pk_{ID}\}$
4 : **return** pk_{ID}

Oracle $\mathsf{Sign}_{\mathtt{DW}}(\mu, \rho)$

1 : **if** $(\rho \notin \mathcal{R})$ **then**
  return $\perp$
2 : $sk \leftarrow \mathtt{DW.SKDer}(msk, \rho)$
3 : $\sigma \leftarrow \mathtt{DW.Sign}(sk, \mu)$
4 : **return** σ

Fig. 4. Unlinkability game DWUNL for deterministic wallets.

Definition 7. *A deterministic wallet scheme* DW *is PQ-full-unlinkable if for any quantum adversary* $\mathcal{A}$*, the advantage in game* DWUNL *(cf. Figure 4) is negligible.*

3.2 Wallet Unforgeability

At a high-level, the unforgeability guarantees that once funds are transferred to the cold wallet they remain secure even if the hot wallet is compromised, and the adversary can observe transfers of coins that are sent from the cold wallet. The standard unforgeability game for deterministic wallet DWUF is presented in Figure 5. Initially, the adversary $\mathcal{A}$ obtains a master public key mpk and the chain code ch as input. $\mathcal{A}$ is allowed to access the oracles PK and $\mathsf{Sign}_{\mathtt{DW}}$ which are defined as same as in the game DWUNL, with the difference that PK now does not need to keep track of all queried identities and $\mathsf{Sign}_{\mathtt{DW}}$ now additionally keeps track of all queried messages. Finally, $\mathcal{A}$ outputs a forgery tuple $(\mu^*, \sigma^*, \rho^*)$. We say $\mathcal{A}$ wins the DWUF game if (μ^*, ρ^*) has not been queried to oracle $\mathsf{Sign}_{\mathtt{DW}}$, and σ^* is a valid signature for μ^* w.r.t. pk^*. In addition, as prior works [15,3], we also adopt the *public key prefixing* method to against the *related key attacks* [32,15], i.e., a signature on a message μ is computed as $\mathtt{Sign}(sk, (hpk, \mu))$ where $hpk := \mathsf{F}(pk)$ and F is a collision-resistant hash function.

Definition 8. *A deterministic wallet scheme* DW *is PQ-unforgeable if for any quantum adversary* $\mathcal{A}$*, the advantage in game* DWUF *(cf. Figure 5) is negligible.*

Game DWUF

$\mathcal{Q} := \varnothing$
$(mpk, msk, ch) \leftarrow \mathtt{DW.KeyGen}(1^\lambda)$
$(\mu^*, \sigma^*, \rho^*) \leftarrow \mathcal{A}^{\mathsf{Sign}_{\mathtt{DW}}, \mathsf{PK}}(mpk, ch)$
if $(\mu^*, \rho^*) \in \mathcal{Q}$ **then**
  return 0
$pk^* \leftarrow \mathtt{DW.KeyGen}(mpk, \rho^*)$
if $\mathtt{DW.Ver}(pk^*, \mu^*, \sigma^*) = 0$ **then**
  return 0
return 1

Oracle $\mathsf{PK}(ID)$

$\rho_{ID} \leftarrow \mathtt{DW.RandGen}(ch, ID)$
$pk_{ID} \leftarrow \mathtt{DW.PKDer}(mpk, \rho_{ID})$
return pk_{ID}

Oracle $\mathsf{Sign}_{\mathtt{DW}}(\mu, \rho)$

if $(\rho \notin \mathcal{R})$ **then**
  return $\perp$
$\mathcal{Q} := \mathcal{Q} \cup \{\mu, \rho\}$
$sk \leftarrow \mathtt{DW.SKDer}(msk, \rho)$
$\sigma \leftarrow \mathtt{DW.Sign}(sk, \mu)$
return σ

Fig. 5. Unforgeability game DWUF for deterministic wallets.

4 The Construction for Deterministic Wallets

In this section, we introduce our generic construction, and proofs of deterministic wallets in a quantum world. We assume in the following a signature scheme with

rerandomizable public keys RKS := (RKS.KeyGen, RKS.RandGen, RKS.RandPK, RKS.RandSK, RKS.Sign, RKS.Ver). A generic construction of a deterministic wallet scheme DW := (DW.KeyGen, DW.RandGen, DW.PKDer, DW.SKDer, DW.Sign, DW.Ver) is depicted in Figure 6.

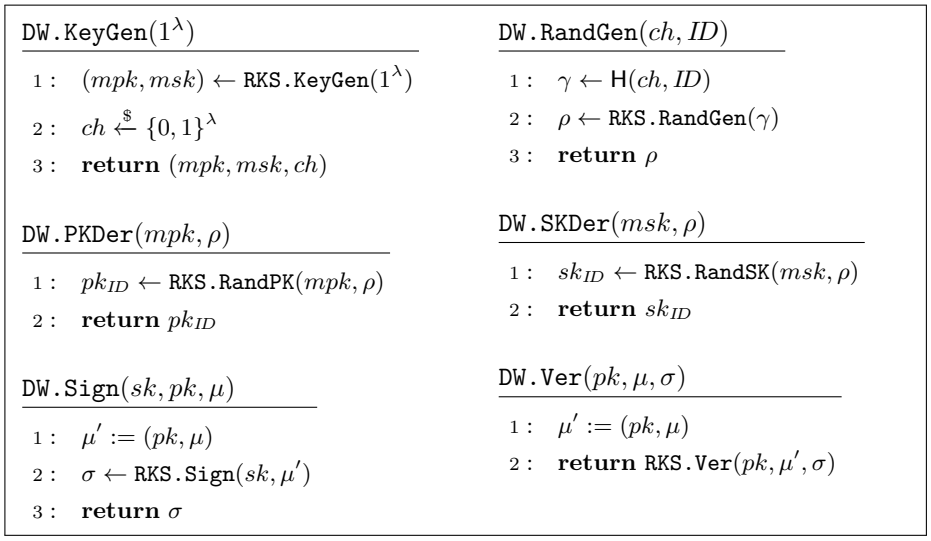

Fig. 6. Construction of a deterministic wallet DW from a signature scheme with rerandomizable keys RKS and a random oracle H.

4.1 Security Proofs

In this section we prove that the DW scheme achieves both unlinkability and unforgeability against quantum adversaries.

Theorem 1. *Let* RKS *be a signature scheme with rerandomizable public keys (cf. Definition 2) and* H *be a random oracle. Then the deterministic wallet scheme* DW *built from* RKS *and* H *(cf. Figure 6) is* DWUNL *secure according to Definition 7, i.e., against quantum adversaries which have access to* $|\mathsf{H}\rangle$.

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary which makes q queries to its oracles $|\mathsf{H}\rangle$. We prove the theorem via the following two games.

Game G_0: This game behaves exactly as the game DWUNL (cf. Figure 4) instantiated with DW (cf. Figure 6).

Game G_1: This game is the same as G_0, except that the randomness ρ, prior to running $\mathcal{A}_2$ (Line 11 in Figure 4), is sampled at random, i.e., independent of the random oracle. In this way, the only difference between two games G_0 and G_1

lies in the random oracle, since the randomness ρ in both games is distributed identical. From the point of view of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the random oracle in game G_1 is $|\mathsf{H}_{\mathcal{S}\to\$}\rangle$, i.e., the random oracle that is reprogrammed the chain code in $\mathcal{S}$ to random value. Hence, we can bound the advantage in distinguishing G_0 and G_1 by the advantage in distinguishing the random oracles $|\mathsf{H}\rangle$ and $|\mathsf{H}_{\mathcal{S}\to\$}\rangle$. Applying the O2H Lemma (cf. Lemma 1) yields

$$\left|\Pr\left[\mathcal{A}^{|\mathsf{H}\rangle} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{|\mathsf{H}_{\mathcal{S}\to\$}\rangle} \Rightarrow 1\right]\right| \leq 2q\sqrt{\Pr\left[ch = ch' \in \mathcal{S} : \mathcal{B}^{|\mathsf{H}\rangle} \Rightarrow ch'\right]}$$

where $\mathcal{B}$ is the adversary specified in Lemma 1, $\mathcal{S}$ is the set contains all chain codes prior to running $\mathcal{A}_2$. In our setting, $|\mathcal{S}| = 1$ since there is only one chain code ch throughout the system. This yields

$$\Pr\left[ch = ch' \in \mathcal{S} : \mathcal{B}^{|\mathsf{H}\rangle} \Rightarrow ch'\right] \leq \frac{|\mathcal{S}|}{2^\lambda} = \frac{1}{2^\lambda}.$$

Combining the above equations yields that the advantage in distinguishing G_0 and G_1 is negligible in the security parameter λ.

It remains to bound the advantage of $\mathcal{A}$ in game G_1, where the same argument from the classical proof applies. In G_1, the challenge public key $pk^b_{ID^*}$ given to $\mathcal{A}_2$ is independent of the random oracle (as the random oracle is not used in G_1 anymore for deriving keys). Hence, it is irrelevant whether the adversary makes any query (classical or quantum) to the random oracle. As $\mathtt{RKS}$ is a signature scheme with rerandomizable public keys, the challenge public keys $pk^0_{ID^*}$ and $pk^1_{ID^*}$ are identically distributed, which yields that the adversarial advantage is 0. Combining the above proves the theorem.

Theorem 2. *The deterministic wallet scheme* $\mathtt{DW}$ *depicted in Figure 6 is* DWUF *secure in the* QROM *if the signature scheme with rerandomizable public keys* $\mathtt{RKS}$ *depicted in Figure 7 is* $\mathsf{EUF\text{-}CMA\text{-}RK}$ *secure in the* QROM*.*

Proof. Let $\mathcal{A}$ be an adversary which makes q_{H} queries to $|\mathsf{H}\rangle$. And assume $\mathcal{A}$ has non-negligible advantage ϵ in winning the game $\mathsf{DWUF}^{\mathcal{A}}_{\mathtt{DW}}$ in G_0. Then we give a game hop G_1 in which $\mathcal{A}$ loses the game if there is a collision of session keys for different ID, but with a negligible probability. Finally, we construct an algorithm $\mathcal{B}$ that runs $\mathcal{A}$ as subroutine in order to win the game $\mathsf{EUF\text{-}CMA\text{-}RA}^{\mathcal{B}}_{\mathtt{RKS}}$ (cf. Definition 4) against the $\mathtt{RKS}$ scheme.

Game G_0: This game behaves exactly as the game DWUF (cf. Figure 5) instantiated with $\mathtt{DW}$ (cf. Figure 6).

Game G_1: This game behaves as G_0 but aborts when there is a collision of keys for different identities. As [3] mentioned, the bound from [15] is applicable. This yields that the advantage of $\mathcal{A}$ in G_1 is $\epsilon - \mathrm{negl}(\lambda)$.

Bounding the advantage. We now show how to transform an adversary $\mathcal{A}$ playing G_1 into an adversary $\mathcal{B}$ playing $\mathsf{EUF\text{-}CMA\text{-}RK}$ (where, the underlying

signature scheme is $\mathtt{RKS}$). At the start, $\mathcal{B}$ receives a public key pk, samples a chain code $ch \xleftarrow{\$} \{0,1\}^\lambda$, and initializes an empty set $\mathcal{Q} := \varnothing$. It invokes $\mathcal{A}$ on input $(mpk = pk, ch)$.

Simulation of Quantum Random Oracle $|\mathsf{H}\rangle$. When $\mathcal{A}$ makes a query (ID, ch) to $|\mathsf{H}\rangle$, $\mathcal{B}$ simulates it using a $2q_{\mathsf{H}}$-wise independent function which is indistinguishable for an adversary making q_{H} queries [43].

Simulation of PK *Oracle.* When $\mathcal{A}$ queries its oracle PK on ID, $\mathcal{B}$ computes $pk_{ID} \leftarrow \mathtt{RKS.RandPK}(pk, \rho_{ID})$, where $\rho_{ID} \leftarrow \mathtt{RKS.RandGen}(\gamma_{ID})$ and $\gamma_{ID} \leftarrow \mathsf{H}(ch, ID)$. Finally, $\mathcal{B}$ sends pk_{ID} to $\mathcal{A}$.

Simulation of $\mathsf{Sign}_{\mathtt{DW}}$ *Oracle.* When $\mathcal{A}$ makes a query (μ, ρ) to its oracle $\mathsf{Sign}_{\mathtt{DW}}$, $\mathcal{B}$ checks if $\rho \in \mathcal{R}$, otherwise aborts. $\mathcal{B}$ computes $pk \leftarrow \mathtt{DW.PKDer}(mpk, \rho)$ then sets $\mu' = (\mu, pk)$, and queries its own oracle $\mathsf{Sign}_{\mathtt{RKS}}$ on (μ', ρ) and obtains the response σ. Then $\mathcal{B}$ adds the tuple (μ, ρ) to $\mathcal{Q}$ and sends σ to $\mathcal{A}$. When $\mathcal{A}$ outputs a forgery $(\mu^*, \sigma^*, \rho^*)$, $\mathcal{B}$ checks if $(\mu^*, \rho^*) \in \mathcal{Q}$, then $\mathcal{B}$ computes the $pk^* \leftarrow \mathtt{DW.PKDer}(mpk, \rho^*)$, and sets $\hat{\mu}^* = (\mu^*, pk^*)$. Finally, $\mathcal{B}$ outputs $(\hat{\mu}^*, \sigma^*, \rho^*)$.

We now show that the output of $\mathcal{B}$ is a valid forgery whenever the output of $\mathcal{A}$ is. Since $(\mu^*, \sigma^*, \rho^*)$ is a valid forgery by $\mathcal{A}$, we know that $(\mu^*, \rho^*) \notin \mathcal{Q}$ and hence $(\hat{\mu}^*, \rho^*) \notin \mathcal{Q}_{\mathsf{c}}$ where $\mathcal{Q}_{\mathsf{c}}$ is the set that was initialized by the challenger. Therefore, validity of the forgery by $\mathcal{A}$ yields validity of the forgery by $\mathcal{B}$. Assuming the security of the underlying signature scheme $\mathtt{RKS}$, we have that this advantage be negligible. This yields that ϵ is negligible, resulting in a contradiction with the ϵ is non-negligible as described in G_0. This completes the proof.

5 Lattice-Based Signatures with Rerandomizable PK

In this section, we propose a lattice-based construction of a signature scheme with honestly rerandomizable public keys $\mathtt{RKS}$ (cf. Definition 2) in Section 5.1, and prove the security in Section 5.2.

5.1 Construction

In the following, we describe the signature scheme with rerandomizable public keys $\mathtt{RKS}$. The scheme extends the generic construction of lattice-based hash-and-sign signatures $\mathtt{LHS}$ from Section 2.5. In addition, we define the following functions and algorithms:

- GenR is an algorithm that on input $(\mathsf{dim}, s, \mathsf{bnd}, \mathsf{rnd})$, then outputs a full-rank matrix $\mathbf{R} = [\ \mathbf{r}_1 \mid \cdots \mid \mathbf{r}_{\mathsf{dim}}\ ] \in \mathcal{D}_{\mathbb{Z}^{\mathsf{dim}\times\mathsf{dim}},s}$ such that $\|\mathbf{R}\|_{\mathrm{GS}} \leq \mathsf{bnd}$ by using a randomness rnd. As [1] mentioned, GenR algorithm in practice can be built from a "standard" random function $h : \{0,1\}^* \to \{0,1\}^t$ by using h as a coin generator for the sampling process in $\mathsf{SampleR}$ algorithm (cf. Lemma 4).
- $\mathsf{F} : \{0,1\}^* \to \{0,1\}^{o(\lambda)}$ is a collision-resistant hash function that is used to hash the public key for preventing related key attacks [32].

Let $\mathcal{R} := \mathcal{D}_{\mathbb{Z}^{m\times m},s}$ be the randomness space. The matrix $\mathbf{R} \in \mathcal{R}$, output from GenR, plays the key role in the keys randomization. Specifically, given a valid LHS key pair ($pk := \mathbf{A}, sk := \mathbf{B}$), then we immediately obtain a randomized and valid key pair ($pk' := \mathbf{AR}^{-1}, sk' := \mathbf{RB}$) since $\mathbf{AR}^{-1}\mathbf{RB} = \mathbf{AB} = \mathbf{0} \pmod q$. By the basis randomization algorithm $\mathsf{RandBasis}$ that is called in the $\mathsf{BasisDel}$ algorithm (cf. Lemma 7), the delegation cannot be inverted, i.e., the master key cannot be revealed from the randomized key. To guarantee the norm of the randomized secret key is sufficiently short, we generate the matrix $\mathbf{R}$ by $\mathsf{GenR}(m, s, B/\beta, \gamma)$ for a predefined positive number B and where β we set as the Gram-Schmidt norm bound of $\mathbf{R}$ i.e., $\|\mathbf{R}\|_{\mathrm{GS}} \leq \beta$.

Below we describe our signature scheme with rerandomizable public keys. The respective algorithms are formalized in Figure 7.

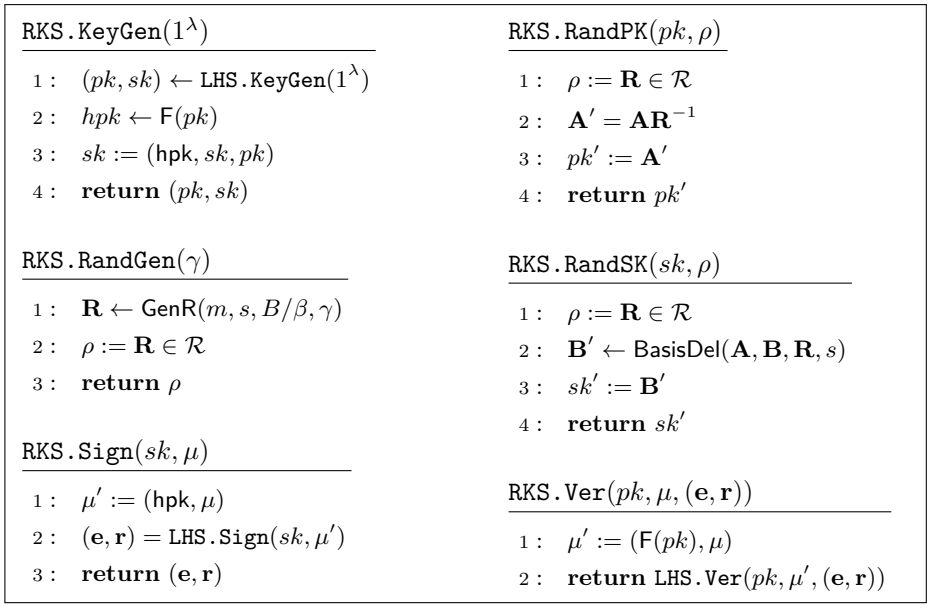

Fig. 7. Construction of lattice-based signature scheme with rerandomizable public keys.

`RKS.KeyGen`: The key generation algorithm `RKS.KeyGen` takes as input a security parameter and then invokes the algorithm `LHS.KeyGen` to obtain a key pair (pk, sk) where $pk := \mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $sk := \mathbf{B} \in \mathbb{Z}^{m\times m}$. Then, it computes $\mathsf{hpk} := \mathsf{F}(pk)$, and prepends (pk, hpk) to sk. Finally, it outputs (pk, sk).

`RKS.RandGen`: On input a randomness $\gamma \in \{0,1\}^{o(\lambda)}$, the randomness generation algorithm `RKS.RandGen` computes $\rho := \mathbf{R} \in \mathcal{R}$ by $\mathsf{GenR}(m, s, B/\beta, \gamma)$. Finally, it outputs ρ.

$\texttt{RKS.RandPK}$: On input a public key $pk = \mathbf{A}$ and a randomness $\rho := \mathbf{R} \in \mathcal{R}$, the public key randomization algorithm $\texttt{RKS.RandPK}$ computes $\mathbf{A}' = \mathbf{A}\mathbf{R}^{-1}$ and sets $pk' = \mathbf{A}'$. Finally, it outputs pk'.

$\texttt{RKS.RandSK}$: On input a secret key $sk = (\mathbf{A}, \mathbf{B}, \mathsf{hpk})$ and a randomness $\rho := \mathbf{R} \in \mathcal{R}$, the secret key randomization algorithm $\texttt{RKS.RandSK}$ computes the $sk' = \mathbf{B}'$ by the basis delegation algorithm $\mathsf{BasisDel}(\mathbf{A}, \mathbf{B}, \mathbf{R}, s)$. Finally, it outputs sk'.

$\texttt{RKS.Sign}$: On input a secret key $sk = (\mathbf{A}, \mathbf{B}, \mathsf{hpk})$ and a message μ, the signing algorithm sets $\mu' = (\mathsf{hpk}, \mu)$ and invokes the algorithm $\texttt{LHS.Sign}(sk, \mu')$ to obtain a signature $(\mathbf{e}, \mathbf{r})$. Finally, it outputs $(\mathbf{e}, \mathbf{r})$.

$\texttt{RKS.Ver}$: On input a public key pk, a message μ, and a signature $(\mathbf{e}, \mathbf{r})$, the verification algorithm $\texttt{RKS.Ver}$ sets $\mu' = (\mathsf{F}(pk), \mu)$ and outputs the bit obtained by running the algorithm $\texttt{LHS.Ver}(vk, \mu', (\mathbf{e}, \mathbf{r}))$.

The correctness of the $\texttt{RKS}$ scheme directly follows from the correctness of the $\texttt{LHS}$ scheme.

5.2 Security Proof

In this section, we analyze the $\mathsf{EUF\text{-}CMA\text{-}RK}$ security of the scheme $\texttt{RKS}$ in the QROM. More precisely, we reduce its $\mathsf{EUF\text{-}CMA\text{-}RK}$ security to the $\mathsf{EUF\text{-}CMA}$ security of the lattice-based hash-and-sign signature scheme $\texttt{LHS}$ described in Section 2.5. Note that rerandomizability of public keys (see Definition 2) follows from the property of Gaussian distribution that is item 2 of Lemma 2.

Theorem 3. *The signature scheme with rerandomizable public keys* $\texttt{LHS}$ *depicted in Figure 7 is* $\mathsf{EUF\text{-}CMA\text{-}RK}$ *secure in the* QROM *if scheme* $\texttt{LHS}$ *described in Figure 3 is* $\mathsf{EUF\text{-}CMA}$ *secure in the* QROM.

Proof. Let $\mathcal{A}$ be an adversary that is able to generate valid forgeries under $\texttt{RKS}$ scheme, i.e., $\mathcal{A}$ is able to win the unforgeability game of $\mathsf{EUF\text{-}CMA\text{-}RK}^{\mathcal{A}}_{\texttt{RKS}}$ (cf. Definition 4). We construct an algorithm $\mathcal{B}$ (depicted in Figure 8) that runs $\mathcal{A}$ as subroutine in order to win the game $\mathsf{EUF\text{-}CMA}^{\mathcal{B}}_{\texttt{LHS}}$ (cf. Definition 3) against the $\texttt{LHS}$ scheme. By the security model w.r.t. $\mathsf{EUF\text{-}CMA\text{-}RK}$, $\mathcal{A}$ has quantum access to a random oracle $\mathsf{H}_{\texttt{RKS}}$ and classical access to signing oracle $\texttt{Sign}_{\texttt{RKS}}$. And by the security model w.r.t. $\mathsf{EUF\text{-}CMA}$, the reduction $\mathcal{B}$ has quantum access to the random oracle $\mathsf{H}_{\texttt{LHS}}$ and classical access to signing oracle $\texttt{Sign}_{\texttt{LHS}}$. $\mathcal{B}$ obtains as input a public key $pk_{\texttt{LHS}} = \mathbf{A}$, then simulates to $\mathcal{A}$ as described in the following.

Setup. $\mathcal{B}$ first samples $\boldsymbol{\gamma} \xleftarrow{\$} \{0,1\}^{\lambda}$ and uses the public key $pk_{\texttt{LHS}}$ from the $\mathsf{EUF\text{-}CMA}$ game as the public key $pk_{\texttt{RKS}}$ in its simulation of $\mathsf{EUF\text{-}CMA\text{-}RK}$. In other words, it runs $\mathcal{A}$ on input $pk_{\texttt{RKS}} = pk_{\texttt{LHS}}$ and $\boldsymbol{\gamma}$ in $\mathsf{EUF\text{-}CMA\text{-}RK}$.

Simulation of Random Oracle Queries. When $\mathcal{A}$ queries on random oracle $\mathsf{H}_{\texttt{RKS}}$, $\mathcal{B}$ forwards the queries to $\mathsf{H}_{\texttt{LHS}}$ and then forwards the response from $\mathsf{H}_{\texttt{LHS}}$ as the response of $\mathsf{H}_{\texttt{RKS}}$ to $\mathcal{A}$.

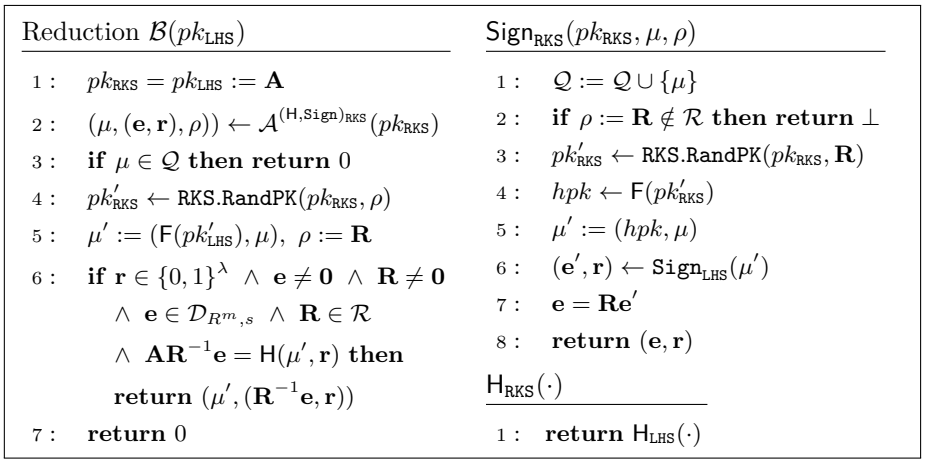

Fig. 8. Reduction from the EUF-CMA security of lattice-based hash-and-sign signature scheme LHS (Figure 1) to EUF-CMA-RK security of signature scheme with honestly rerandomizable public keys RKS (Figure 2).

Simulation of Signing Queries. When $\mathcal{A}$ queries signing oracle $\mathtt{Sign}_{\mathtt{RKS}}$ on input $(pk_{\mathtt{RKS}}, \mu, \rho)$, after set $\mathcal{Q} := \mathcal{Q} \cup \{\mu\}$, $\mathcal{B}$ first parses $\rho := \mathbf{R}$ and checks if $\mathbf{R} \in \mathcal{R}$ otherwise aborts. Then, $\mathcal{B}$ computes the randomized public key $pk'_{\mathtt{RKS}}$ and $hpk \leftarrow \mathsf{F}(pk'_{\mathtt{RKS}})$, sets $\mu' = (hpk, \mu)$ and then obtain $(\mathbf{e}', \mathbf{r}) \leftarrow \mathtt{Sign}_{\mathtt{LHS}}(\mu')$ by querying its own challenge signing oracle. Finally, $\mathcal{B}$ sets $\mathbf{e} = \mathbf{R}\mathbf{e}'$ and sends $(\mathbf{e}, \mathbf{r})$ to $\mathcal{A}$. Due to $(\mathbf{e}', \mathbf{r}) \leftarrow \mathtt{Sign}_{\mathtt{LHS}}(\mu')$, therefore, it holds that $\mathbf{A}\mathbf{e}' = \mathsf{H}(\mu', \mathbf{r})$. We know $pk'_{\mathtt{RKS}} := \mathbf{A}' = \mathbf{A}\mathbf{R}^{-1}$, therefore, it holds that $\mathbf{A}' \cdot \mathbf{R}\mathbf{e}' = \mathsf{H}(\mu', \mathbf{r})$.

In the real world and simulation, the item $\mathbf{r}$ of the signature tuple $(\mathbf{e}, \mathbf{r})$ is randomly selected, so we focus on the difference of $\mathbf{e}$ between the real world and simulation. Let $(\mathbf{e}_\mathtt{r}, s_\mathtt{r})$ and $(\mathbf{e}_\mathtt{s}, s_\mathtt{s})$ be the preimage $\mathbf{e}$ in real world and simulation, respectively. In the real world, $\mathbf{e}_\mathtt{r}$ is sampled by the Gaussian preimage sampling algorithm SamplePre with respect to the randomized trapdoor $sk' := \mathbf{B}'$, i.e., $\mathbf{e}_\mathtt{r} \leftarrow \mathsf{SamplePre}(\mathbf{B}', \mathbf{u})$. In the simulation, $\mathbf{e}_\mathtt{s}$ is firstly sampled by SamplePre with respect to the original trapdoor $sk := \mathbf{B}$ and then multiply $\mathbf{R}$, i.e., $\mathbf{e}_\mathtt{s} = \mathbf{R}\bar{\mathbf{e}}$ where $\bar{\mathbf{e}} \leftarrow \mathsf{SamplePre}(\mathbf{B}, \mathbf{u})$. By Lemma 7, it needs to set $s_\mathtt{r} > \|\mathbf{B}\|_{\mathrm{GS}}\sqrt{nm\log q} \cdot \omega(\log^2 m)$. By item 1 of Lemma 2, $\|\mathbf{e}_\mathtt{r}\| \leq s_\mathtt{r}\sqrt{m}$. Then for the norm bound of $\|\mathbf{e}_\mathtt{s}\|$, by Lemma 2, it needs to set $s_\mathtt{s} \geq \|\mathbf{B}\|_{\mathrm{GS}} \cdot \omega(\sqrt{\log m})$, and by item 1 of Lemma 2, $\|\bar{\mathbf{e}}_\mathtt{r}\| \leq s_\mathtt{s}\sqrt{m}$, then by Lemma 4, the norm of each entry $\mathbf{r}_i$ of $\mathbf{R}$ is bounded as $\|\bar{\mathbf{r}}_i\| \leq s_\mathrm{R}\sqrt{m}$. Therefore, the norm bound on $\mathbf{e}_\mathtt{s} = \mathbf{R}\bar{\mathbf{e}}$ is $s_\mathtt{s}s_\mathrm{R}m$, i.e., $\|\mathbf{e}_\mathtt{s}\| \leq s_\mathtt{s}s_\mathrm{R}m$. By Lemma 3 and 4, $s_\mathrm{R} = O(\sqrt{n\log q}) \cdot \omega(\sqrt{\log m})$. Therefore, $\|\mathbf{e}_\mathtt{s}\| \leq \|\mathbf{B}\|_{\mathrm{GS}}\sqrt{n\log q} \cdot \omega(\log^2 m) \cdot m$, i.e., the norm bound of $\mathbf{e}_\mathtt{s}$ is as same as $\mathbf{e}_\mathtt{r}$. Therefore, we can conclude that the simulated signatures are correctly distributed.

Exploiting the Forgery. When $\mathcal{A}$ provides the forgery $(\mu, (\mathbf{e}, \mathbf{r})), \rho)$, $\mathcal{B}$ parses $\rho = \mathbf{R}$, computes the randomized public key $pk'_{\mathtt{RKS}} \leftarrow \mathtt{RKS.RandPK}(pk_{\mathtt{RKS}}, \rho)$, and

set $\mu' = (\mathsf{F}(pk'_{\mathtt{LHS}}), \mu)$. $\mathcal{B}$ aborts if the forgery with respect to a queried message, i.e., $\mu \in \mathcal{Q}$, or the randomness ρ is illegal, i.e., $\rho \notin \mathcal{R}$. $\mathcal{B}$ also aborts if the forgery is not valid by checking as Line 7 of Figure 8. Otherwise $\mathcal{B}$ outputs $(\mathbf{R}^{-1}\mathbf{e}, \mathbf{r})$ as the forgery under LHS.

We note that the environment of $\mathcal{A}$ is perfectly simulated, and whenever $\mathcal{A}$ wins the game $\mathsf{EUF\text{-}CMA\text{-}RK}^{\mathcal{A}}_{\mathtt{RKS}}$, $\mathcal{B}$ wins the game $\mathsf{EUF\text{-}CMA}^{\mathcal{B}}_{\mathtt{LHS}}$.

6 Practical Instantiation and Deployment

In this section, we show a more compact instantiation of our work, then we analyze the deployment over blockchains. Motivated by the general fact that the *transaction throughput* is the most important measure for the wallet schemes when considering deployment in the practical scenario of cryptocurrency system, we consider employing the lattice-based signature Falcon [20] to instantiate our work since Falcon taken compactness as its design rationale and core competitiveness. For a simple transaction in most cryptocurrency networks that transfers coins from one party to another, the transaction's size is mainly dominated by the size of "pk + sig", i.e., the sum of the public key size and the signature size of the employed signature scheme. Falcon is fitted neatly to resolve that since it has the smallest size of "pk + sig" among the candidate algorithms for post-quantum standardization [35].

6.1 Instantiation from Falcon

In this section, we show how to instantiate the signature scheme with rerandomizable public keys RKS with Falcon. The Falcon signature is also based on the hash-and-sign paradigm as shown in Figure 3, and also employs lattice basis as a trapdoor to Gaussian sample the preimages as signatures. And we note that all the employed algorithms can be transformed to the Falcon setting. Below we first describe the lattice-based hash-and-sign signature LHS that instantiates from Falcon.

The LHS scheme that instantiates from Falcon is as same as ours (cf. Figure 3). The difference is that the trapdoor generation algorithm is instantiated by a polynomial ring solving procedure. More precisely, the public key is set as $pk := [\ 1 \mid h\ ]$, secret key is $sk := \begin{bmatrix} g & -f \\ G & -F \end{bmatrix}$ where f, g, F, G are polynomials over $\mathbb{Z}[X]/(X^n + 1)$ and $h \leftarrow g \cdot f^{-1}$. Therefore, it requires setting the dimension $m = 2$. For the Gaussian sampling algorithm SamplePre, as [31] mentioned[1], the Gaussian sampling algorithm can be transferred easily to the compact lattices defined over polynomial rings which include $\mathbb{Z}[X]/(X^n + 1)$.

[1] Even the algorithms that presented in [31] based on a new trapdoor whose quality is measured by the maximal singular value rather than the Gram-Schmidt norm, but the trapdoor can also be transformed to the classical lattice basis form (cf. Lemma 5.3 of [31])

Then we describe how the RKS scheme can be instantiated with Falcon. As described in Section 5.1, the GenR algorithm in practice can be built from standard random function $h : \{0,1\}^* \to \{0,1\}^t$ by employing h as a coin generator in the algorithm SampleR. Therefore, the point is how to instantiate the lattice basis delegation algorithm BasisDel since SampleR is a sub-algorithm of BasisDel (cf. Lemma 7). As given by Lemma 7, the BasisDel has three sub-algorithms: SampleR, ToBasis, and RandBasis. For simplicity, we describe that as below equation

$$\mathsf{BasisDel} \;=\; \mathsf{SampleR} \;+\; \mathsf{ToBasis} \;+\; \mathsf{RandBasis}$$

Recall the description given in Lemma 2, both SampleR and RandBasis algorithms calling the SampleGaussian algorithm. And RandBasis algorithm additionally calls the ToBasis algorithm. We have

$$\mathsf{RandBasis} = \mathsf{SampleGaussian} + \mathsf{ToBasis}, \qquad \mathsf{SampleR} = \mathsf{SampleGaussian}$$

For the SampleGaussian algorithm, as aforementioned, it can be efficiently transformed to the Falcon setting by the work [31]. For the ToBasis algorithm, the point is that if the input set of vectors is full-rank. By the result in [1], we can obtain a full-rank set of vectors by repeatedly calling the SampleGaussian algorithm fewer than two times in expectation for prime q.

In our setting, the master key pair $(pk := \mathbf{A}, sk := \mathbf{B})$ only be used for rerandomization while signatures are generated by using the delegated, i.e., the randomized key pairs $(pk' := \mathbf{A}', sk' := \mathbf{B}')$. In both settings of Falcon and ours, we employ the lattice basis as the secret key. And we use the delegated lattice basis as the signing key. Recall the process of rerandomization, the randomness is mainly derived from the parameter $\mathbf{R}$. Therefore, we next analyze if the norm of $\mathbf{R}$ is reasonably set so that the resulting signature scheme satisfies the norm bound in Falcon. In Falcon, the Gram-Schmidt norm bound on the basis is B. Recall in our setting, the Gram-Schmidt norm bound on the original basis is $\|\mathbf{B}\|_{\mathrm{GS}} \leq \beta$, and the bound on $\mathbf{R}$ is $\|\mathbf{R}\|_{\mathrm{GS}} \leq B/\beta$ since it was generated by $\mathsf{GenR}(m, s, B/\beta, \rho)$. Then we have the Gram-Schmidt norm bound on the delegated basis $\mathbf{B}' = \mathbf{B}\mathbf{R}^{-1}$ such that $\|\mathbf{B}'\|_{\mathrm{GS}} \leq B$. Therefore, we can set the parameters of our scheme as exactly the same as Falcon (Table 3.3 of [20]).

6.2 Deployment over Blockchains

In this section, we analyze the transaction throughput that can be achieved in a cryptocurrency system when employing the signature scheme with rerandomizable public keys RKS instantiated from Falcon. We use the analytical methods from [3] and show the comparison with that. To show the comparison fairly, we state the bit security in quantitative form as [4] which assumes the compared works under the same BKZ cost model [12].

On the blockchains, a transaction (tx) must usually consist of a public key pk and the signature σ that was signed by the sender such that the validity of the transaction can be verified. The raw transaction size (i.e., exclude the size of pk

and σ) of a cryptocurrency system such as Bitcoin is roughly 100 Bytes(B) [41]. Therefore, the size of a valid transaction should be estimated below

$$\mathsf{tx}\text{'s size} \quad = \quad 100\ \mathrm{B} \quad + \quad pk\text{'s} \quad + \quad \sigma\text{'s}$$

When instantiating our wallet scheme with Falcon, we can take the pk size = 897 B and signature size = 666 B (cf. Table 3.3 of [20]) for a post-quantum security level of 158 bits (under BKZ cost model). Therefore, the size of a tx would then result in 100 B + 897 B + 666 B $\approx$ 1.66 KB. Below we show the comparison with prior work [3] in Table 2, in which we also compare the performance on the signing and verify (on portable C reference and AVX2-optimized implementations) that was given in [4].

Table 2. Comparison with prior work

Scheme	Security	Sizes [B]	Transaction throughput	Cycle counts [k-cycles] C Reference	AVX2
[3]	95 bits	pk: 14880 sig: 2592	$\approx$17.5 KB	Sign: 3089.9 Ver: 814.3	1759.0 678.5
Our	158 bits	pk: 897 sig: 666	$\approx$1.66 KB	Sign: 1368.5 Ver: 95.6	1009.8 81.0

The work [3] was instantiated from the lattice-based signature scheme qTESLA[4] which was implemented with the CPU of 3.4GHz Intel Core i7-6700 (Skylake), while our work from Falcon which was implemented with the CPU of 3.3GHz Intel Core i7-6567U (Skylake). From the comparison result of Table 2, our scheme has a more compact performance than the prior work [3] since our works achieved more efficiency on the time-consuming and parameter sizes.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In: Rabin, T. (ed.) Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6223, pp. 98–115. Springer, Heidelberg (2010), https://doi.org/10.1007/978-3-642-14623-7_6
2. Ajtai, M.: Generating hard instances of lattice problems. In: STOC. pp. 99–108. ACM (1996)
3. Alkeilani Alkadri, N., Das, P., Erwig, A., Faust, S., Krämer, J., Riahi, S., Struck, P.: Deterministic wallets in a quantum world. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 1017–1031 (2020)
4. Alkim, E., Barreto, P.S., Bindel, N., Krämer, J., Longa, P., Ricardini, J.E.: The lattice-based digital signature scheme qtesla. In: International Conference on Applied Cryptography and Network Security. pp. 441–460. Springer (2020)

5. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: Albers, S., Marion, J. (eds.) STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings. vol. 3, pp. 75–86. Germany (2009), https://doi.org/10.4230/LIPIcs.STACS.2009.1832
6. Ambainis, A., Hamburg, M., Unruh, D.: Quantum security proofs using semi-classical oracles. In: Advances in Cryptology - CRYPTO 2019. pp. 269–295. Springer (2019)
7. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993. pp. 62–73. ACM, New York (1993), https://doi.org/10.1145/168588.168596
8. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7073, pp. 41–69. Springer, Heidelberg (2011), https://doi.org/10.1007/978-3-642-25385-0_3
9. Buterin, V.: Deterministic Wallets, Their Advantages and their Understated Flaws (2013), https://bitcoinmagazine.com/articles/deterministic-wallets-advantages-flaw-1385450276
10. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6110, pp. 523–552. Springer, Heidelberg (2010), https://doi.org/10.1007/978-3-642-13190-5_27
11. Chailloux, A., Debris-Alazard, T.: Tight and optimal reductions for signatures based on average trapdoor preimage sampleable functions and applications to code-based signatures. In: IACR International Conference on Public-Key Cryptography. pp. 453–479. Springer (2020)
12. Chen, Y., Nguyen, P.Q.: Bkz 2.0: Better lattice security estimates. In: ASIACRYPT. pp. 1–20. Springer (2011)
13. Das, P., Erwig, A., Faust, S., Loss, J., Riahi, S.: The exact security of bip32 wallets. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 1020–1042 (2021)
14. Das, P., Erwig, A., Faust, S., Loss, J., Riahi, S.: The exact security of bip32 wallets. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 1020–1042 (2021)
15. Das, P., Faust, S., Loss, J.: A formal treatment of deterministic wallets. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security. pp. 651–668. ACM New York, NY (2019)
16. Di Luzio, A., Francati, D., Ateniese, G.: Arcula: A secure hierarchical deterministic wallet for multi-asset blockchains. In: International Conference on Cryptology and Network Security. pp. 323–343. Springer (2020)
17. Erwig, A., Riahi, S.: Deterministic wallets for adaptor signatures. In: European Symposium on Research in Computer Security. pp. 487–506. Springer (2022)
18. Fan, C.I., Tseng, Y.F., Su, H.P., Hsu, R.H., Kikuchi, H.: Secure hierarchical bitcoin wallet scheme against privilege escalation attacks. International Journal of Information Security **19**(3), 245–255 (2020)

19. Fleischhacker, N., Krupp, J., Malavolta, G., Schneider, J., Schröder, D., Simkin, M.: Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In: Public-Key Cryptography–PKC 2016, pp. 301–330. Springer (2016)
20. Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon: Fast-fourier lattice-based compact signatures over ntru. Submission to the NISTs post-quantum cryptography standardization process **36**(5) (2018)
21. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Dwork, C. (ed.) Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008. pp. 197–206. ACM, New York (2008), https://doi.org/10.1145/1374376.1374407
22. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on computing **17**(2), 281–308 (1988)
23. Gutoski, G., Stebila, D.: Hierarchical deterministic bitcoin wallets that tolerate key leakage. In: International Conference on Financial Cryptography and Data Security. pp. 497–504. Springer (2015)
24. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography **75**(3), 565–599 (2015)
25. Liu, W., Liu, Z., Nguyen, K., Yang, G., Yu, Y.: A lattice-based key-insulated and privacy-preserving signature scheme with publicly derived public key. In: European Symposium on Research in Computer Security. pp. 357–377. Springer (2020)
26. Liu, Z., Yang, G., Wong, D.S., Nguyen, K., Wang, H.: Key-insulated and privacy-preserving signature scheme with publicly derived public key. In: 2019 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 215–230. IEEE (2019)
27. Liu, Z., Yang, G., Wong, D.S., Nguyen, K., Wang, H., Ke, X., Liu, Y.: Secure deterministic wallet and stealth address: Key-insulated and privacy-preserving signature scheme with publicly derived public key. IEEE Transactions on Dependable and Secure Computing **19**(5), 2934 (2022)
28. Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In: IAdvances in CryptologyASIACRYPT 2009. pp. 598–616. Springer (2009)
29. Micciancio, D.: Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In: FOCS. pp. 356–365. IEEE (2002)
30. Micciancio, D., Goldwasser, S.: Complexity of lattice problems: a cryptographic perspective, vol. 671. Kluwer Academic Publishers, Boston (2002)
31. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7237, pp. 700–718. Springer, Heidelberg (2012), https://doi.org/10.1007/978-3-642-29011-4_41
32. Morita, H., Schuldt, J.C., Matsuda, T., Hanaoka, G., Iwata, T.: On the security of the schnorr signature scheme and dsa against related-key attacks. In: ICISC 2015. pp. 20–35. Springer (2015)
33. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2009), http://bitcoin.org/bitcoin.pdf

34. NIST: FIPS pub 186-4 - Digital Signature Standard (DSS) (2021), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
35. NIST: Post-Quantum Cryptography (PQC) Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates (2022), https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4
36. Noether, S., Mackenzie, A., et al.: Ring confidential transactions. Ledger **1**, 1–18 (2016)
37. Pieter, W.: Bip32: hierarchical deterministic wallets (2012), https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki
38. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2), 120–126 (1978)
39. Unruh, D.: Revocable quantum timed-release encryption. Journal of the ACM (JACM) **62**(6), 1–76 (2015)
40. Van Saberhagen, N.: Cryptonote v 2.0 (2013)
41. WIKI: Bitcoin wiki transaction format. In: Accessed: 2020-05-04. (2019), https://en.bitcoin.it/wiki/Transaction
42. Yin, X., Liu, Z., Yang, G., Chen, G., Zhu, H.: Secure hierarchical deterministic wallet supporting stealth address. In: European Symposium on Research in Computer Security. Springer (2022)
43. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model pp. 758–775 (2012)