W-OTS$^+$ up my Sleeve! A Hidden Secure Fallback for Cryptocurrency Wallets

David Chaum[1], Mario Larangeira[2], Mario Yaksetig[1,3], and William Carter[1]

[1] xx network, Cayman Islands
{david, mario, will}@xx.network
[2] Tokyo Institute of Technology and IOHK, Japan
mario@c.titech.ac.jp/mario.larangeira@iohk.io
[3] University of Porto, Portugal
mario.yaksetig@fe.up.pt

Abstract. We introduce a new key generation mechanism where users can generate a "back up key", securely nested inside the secret key of a signature scheme. Our main motivation is that in case of leakage of the secret key, established techniques based on zero-knowledge proofs of knowledge are void since the key becomes public. On the other hand, the "back up key", which is secret, can be used to generate a "proof of ownership", *i.e.*, only the real owner of this secret key can generate such a proof. To the best of our knowledge, this extra level of security is novel, and could have already been used in practice, if available, in digital wallets for cryptocurrencies that suffered massive leakage of account private keys. In this work, we formalize the notion of "Proof of Ownership" and "Fallback" as new properties. Then, we introduce our construction, which is compatible with major designs for wallets based on ECDSA, and adds a W-OTS$^+$ signing key as a "back up key". Thus offering a quantum secure *fallback*. This design allows the hiding of any quantum secure signature key pair, and is not exclusive to W-OTS$^+$. Finally, we briefly discuss the construction of multiple generations of proofs of ownership.

Keywords: Digital currencies · Hash-based signatures · Post-Quantum Cryptography.

1 Introduction

Digital wallets allow users to securely store secret cryptographic keys which can be used to spend cryptocurrency funds. These wallets, and corresponding keys, are becoming increasingly important as hackers attempt to exploit eventual security flaws and, as a result, steal funds controlled by such wallets. In practice, users rely on a few approaches. The most

straightforward technique is to resort to secure hardware, *i.e.* hardware wallets [1]. Another popular practice among practitioners is the technique of hot/cold wallets [11], where, briefly, there is a hot wallet permanently connected to the network, typically initiated with the public key and can generate addresses for receiving funds. The cold wallet, on the other hand, stores the secret key and is kept without network connection. This separation ensures that it is harder for attackers to gain access to secret keys as they are kept offline. Despite these security enhanced wallets, we observe that in the case of massive key leakage, including in the cold wallet, any attempt of confirming the ownership of the leaked key is impractical, if possible.

Massive leaks have already happened. We showcase our work by highlighting the hack involving the Trinity wallet [24], which resulted in the theft of roughly 1.5M USD. Trinity, an open-source software wallet which enables users to manage their IOTA tokens, suffered from a hack so severe that the IOTA Foundation decided to halt the coordinator node and, as a result, temporarily stopped the confirmations of all transactions on the network. To perform this attack, the adversary gained the ability to load malicious code into the local Trinity wallet instances running on the computers of the target users and retrieved the secret seeds—along with the encryption passwords—to a malicious server owned by the attacker. The adversary then waited for the release of a new software update, which when installed resulted in overwriting the local cache of each compromised user and cleaned the traces of the exploit. After performing this attack, the adversary effectively gained access to secret keys that were—at least temporarily—on hot storage, resulting in a massive leakage without a practical solution for the users to prove ownership of their secret keys.

On a different threat vector, attacks against cold wallets storing elliptic curve secret keys are believed to be possible with the uprising of quantum computing. Major cryptocurrencies are based heavily on the ECDSA signature scheme. Therefore, an adversary capable of breaking the elliptic curve discrete logarithm problem (ECDLP) can extract the secret keys behind a wallet address, even though such keys never left the cold storage.

Structure in the ECDSA secret key. In both attacks mentioned earlier, the target is the secret key, *i.e.* the secret information kept by the wallet. Prior to the leakage, standard technique to prove ownership can be constructed by employing Zero Knowledge Proof of Knowledge (ZKPK) Protocols. The security derives directly from the zero knowledge

and soundness properties of ZKPK. However, in the case of a *massive leakage*, any party can generate such proof. Therefore new techniques should be developed.

The main technical challenge is to combine two cryptographic schemes by adding the creation of "some structure" in the ECDSA secret key, which allows for the introduction of some sort of "proof of ownership" that prevents or at least minimizes the damage of situations like the IOTA Hack, while also providing quantum resistance, in the case of the massive leakage. Ideally, this new design should also be compatible with the current address system of cryptocurrencies by not significantly changing the ECDSA design.

We address the issue of guaranteeing backward compatibility with ECDSA based wallets by adding a nested "back up key" to generate a quantum secure "proof of ownership". In other words, we propose a technique to embed a nested private key (in addition to the ordinary private key) to be used only in situations when it is necessary to prove ownership.

1.1 Previous Work: Hash-based Signatures

We briefly describe hash-based signatures and focus on the one-time use constructions, since these are the ones most closely related to our proposal and offer quantum resistance.

Typically, every signature scheme requires the use of a cryptographic hash function. Hash-based signature schemes rely solely on hash functions and, as a result, do not require any additional cryptographic or computational assumption. Since there are cryptographically-secure hash functions that are considered unfeasible to invert (later we review a more formal definition), users can provide a preimage that serves as proof of ownership of a specific public key.

Lamport [20] proposed a signature scheme that relies only on the security of one-way functions and can be used to sign multiple bits at once. For simplicity, we illustrate the example of the signing of a single bit, where the signer first generates two secret key values (x, y), and publishes the corresponding pair of hash values as the public key $PK = (\mathcal{H}(x), \mathcal{H}(y))$. The signer then releases the secret value x in case the bit to be signed is 0, or releases the secret value y in case the bit is 1. One of the main limitations of this scheme, however, is the fact that it can only produce one-time signatures (OTS).

Shortly after Lamport's publication, Winternitz [21] proposed a scheme known as the Winternitz one-time signature (W-OTS), that allowed the

signing of several bits at once as opposed to individual bits. In this scheme, the public key is $\mathcal{H}^w(x)$ instead of the pair $(\mathcal{H}(x), \mathcal{H}(y))$. If the message byte to be signed is, for example, 20, then the signature output is $\mathcal{H}^{20}(x)$, such that $\mathcal{H}^i(x)$ means i nested hashes of x. Moreover, to prevent an attacker from modifying the signature, the signer also releases a checksum value associated with the signed byte. This checksum is designed to prevent the adversary to attempt to produce a forgery by increasing any of the bytes without invalidating the resulting signature.

Hülsing [16] published an upgrade called W-OTS$^+$ that shortens the signatures size and increases the security of the original Winternitz scheme. This construction uses a chaining function in addition to a family of keyed functions, along with the XOR of a random value (or mask) before applying the one-way function to a specific ladder height.

1.2 Our Contribution

We start by defining two new properties we introduce. They are *fallback* and *proof of ownership*. These properties extend the functionality of a signature scheme by (1) allowing, considering a ECDSA scheme, the continued use of a signature scheme despite the leakage of the secret key, albeit using a different scheme (*i.e.* variant of W-OTS$^+$), and (2) prove the ownership of a leaked key, even when it becomes public.

More concretely, regarding (1), in addition to the verification and secret key, the generation algorithm of our constructions also outputs the "back up key" which can be used with the secret key as a separate (quantum secure) signature scheme for the fallback situation. In such situation, our construction is usable for existing wallets, and relies solely on symmetric primitives which, when instantiated with the correct security parameter, are conjectured secure even against adversaries with quantum capabilities or adversaries with access to elliptic curve secret key material stored on hot wallets. Our construction is easily extendable and relevant in a hot/cold wallet setting where the hot wallet—permanently connected to the network—contains the elliptic curve public key and, if needed, the actual elliptic curve key pair. The cold wallet, on the other hand, is kept without any network connection and stores the quantum-secure key pair, including the "back up key".

Regarding (2), we observe that a variant of the W-OTS$^+$ signature scheme nested into the main signature scheme can be used to prove the ownership of an ECDSA secret key. Briefly, by design, the "back up key" is the secret key of the internally nested scheme, *i.e.*, W-OTS$^+$, while the ECDSA secret key is derived from the public key of the W-OTS$^+$

variant. Given that we have an internal signature scheme, the proof of ownership for the, potentially leaked ECDSA secret key, is the W-OTS$^+$ like signature. We emphasize that the combination of two different signature schemes is the main technical challenge of this work, and required a new breakthrough which is the new signature variant we propose: the Extended W-OTS$^+$, *i.e.* eW-OTS$^+$. Note that, likewise we adapt the W-OTS$^+$ signature scheme, other hash based signature schemes can also be adapted in a similar fashion.

The ECDSA secret key is generated by combining the eW-OTS$^+$ verification key ℓ tuple into a L-tree structure, similarly to other existing proposals for other hash based signatures [10]. The resulting value is then treated as the ECDSA secret key, making our practical key generation mechanism especially suited for digital wallets, requiring no change on existing blockchain system designs currently in use. We analyze the security of our construction starting by studying our proposed signature scheme eW-OTS$^+$ in the light of the existing attacks against symmetric cryptographic primitives, including quantum ones as described in [15,23]. Finally, we implemented a prototype with full test coverage and compared our results with reference implementations.

In summary, in this work we:

- introduce new properties for a digital scheme named *fallback* and *proof of ownership*;
- propose a new variant of the W-OTS$^+$ based signature scheme: Extended W-OTS$^+$;
- construct a protocol, named $\mathcal{S}_{leeve}$, for generation and verification of a (single) proof of ownership π and formalize its security based on the Extended W-OTS$^+$;
- report on the results of the experiments of our prototype, which implements the main routines of our construction;
- discuss how to extend our $\mathcal{S}_{leeve}$ construction for multiple proofs of ownership.

We showcase our protocol $\mathcal{S}_{leeve}$ as a tool for a catastrophic scenario as a massive leak of private information. As already happened [24], in order to minimize the damage, the system could be halted, until all the honest users are confirmed. The proof of ownership via $\mathcal{S}_{leeve}$ allows the users to confirm their authenticity, and its addresses, using a *back up key* stored separately (as will be formally introduced later when describing $\mathcal{S}_{leeve}$) and used only in situations like this. Furthermore, it is worth mentioning that although it does not help in the return of the potentially already

stolen funds, once the system is stopped, $\mathcal{S}_{leeve}$ allows the quick and safe identification of the honest owners of the addresses.

2 Preliminaries

It is convenient to quickly review the ECDSA construction for digital signature and W-OTS$^+$ signature construction from [16].

Definition 1 (ECDSA). *Given a hash function H, the ECDSA signature scheme is the tuple* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$*, defined as in Table 1:*

$\mathsf{Gen}(1^\lambda)$	$\mathsf{Sign}^H(\mathsf{m}, \mathtt{sk})$	$\mathsf{Verify}^H(\mathsf{m}, \mathtt{vk}, \sigma)$
$x \xleftarrow{\$} \mathbb{Z}_p$	$z \leftarrow H(\mathsf{m})$	Parse: $(r, s) \xleftarrow{p} \sigma$
$\mathtt{sk} \leftarrow x$	$t \xleftarrow{\$} \mathbb{Z}_p$	If $(r, s) \notin \mathbb{Z}_p$
$\mathtt{vk} \leftarrow g^x$	$(e_x, e_y) \leftarrow g^t$	Return 0
return $(\mathtt{vk}, \mathtt{sk})$	$r \leftarrow e_x \mod p$	$w \leftarrow s^{-1}$
	If $r = 0 \mod p$	$z \leftarrow H(\mathsf{m})$
	Pick another t	$u_1 \leftarrow zw \mod p$
	and start again	$u_2 \leftarrow rw \mod p$
	$s \leftarrow t^{-1} \cdot (z + r \cdot \mathtt{sk})$	$(e_x, e_y) \leftarrow g^{u_1} \cdot \mathtt{vk}^{u_2}$
	If $s = 0 \mod p$	If $(e_x, e_y) = (0, 0)$
	Pick another t	Return 0
	and start again	Return $r = e_x \mod p$
	Return $\sigma = (r, s)$	

Table 1. ECDSA construction.

The W-OTS$^+$ Construction. The Winternitz-OTS$^+$ signature schemes introduced by Hülsing [18] introduces an alternative signature scheme with quantum resistance. Their construction relies on a hash family and a chaining function which we now review.

Definition 2 (Family of Functions). *Given the security and the Winternitz parameters, respectively, $\lambda \in \mathbb{N}$ and $w \in \mathbb{N}, w > 1$, let a family of functions $\mathcal{H}_\lambda$ be $\{h_k : \{0,1\}^\lambda \to \{0,1\}^\lambda | k \in \mathcal{K}_\lambda\}$ with key space $\mathcal{K}_\lambda$.*

Definition 3 (Chaining Function). *Given a family of functions $\mathcal{H}_\lambda$, $x \in \{0,1\}^\lambda$, an iteration counter $i \in \mathbb{N}$, a key $k \in \mathcal{K}_\lambda$, for j $\lambda-$bit strings $\mathbf{r} = (r_1, \ldots, r_j) \in \{0,1\}^{\lambda \times j}$ with $j \geq i$, then we have the chaining function as follows*

$$c_k^i(x, \mathbf{r}) = \begin{cases} h_k(c_k^{i-1}(x, \mathbf{r}) \oplus r_i), \ 1 \leq i \leq j; \\ x, \ i = 0. \end{cases}$$

$\mathsf{Gen}_W^k(1^\lambda)$	$\mathsf{Sign}_W^k(\mathsf{m}, \mathtt{sk})$
Pick $(\ell + w - 1)$ λ-bit strings r_i	Compute $\mathsf{m} \to (\mathsf{m}_1, \ldots, \mathsf{m}_{\ell_1})$,
Set $\mathtt{sk}_i \leftarrow r_i$, for $1 \le i \le \ell$	for $\mathsf{m}_i \in \{0, \ldots, w-1\}$
Set $\mathtt{sk} = (\mathtt{sk}_1, \ldots, \mathtt{sk}_\ell)$	Compute checksum $C = \sum_{i=1}^{\ell_1}(w - 1 - \mathsf{m}_i)$,
Set $\mathbf{r} = (r_{\ell+1}, \ldots, r_{\ell+w-1})$	and its base w representation $(C_1, \ldots, C_{\ell_2})$,
Set $\mathtt{vk}_0 = (\mathbf{r}, k)$	for $C_i \in \{0, \ldots, w-1\}$
Set $\mathtt{vk}_i = c_k^{w-1}(\mathtt{sk}_i, \mathbf{r})$, $1 \le i \le \ell$	Parse $B = \mathsf{m} \| C$ as $(b_1, \ldots, b_{\ell_1+\ell_2})$
Set $\mathtt{vk} = (\mathtt{vk}_0, \mathtt{vk}_1, \ldots, \mathtt{vk}_\ell)$	Set $\sigma_i = c_k^{b_i}(\mathtt{sk}_i, \mathbf{r})$, for $1 \le i \le \ell_1 + \ell_2$
Return $(\mathtt{sk}, \mathtt{vk})$	Return $\sigma = (\sigma_1, \ldots, \sigma_{\ell_1+\ell_2})$

$\mathsf{Verify}_W^k(\mathsf{m}, \mathtt{vk}, \sigma)$
Compute $\mathsf{m} \to (\mathsf{m}_1, \ldots, \mathsf{m}_{\ell_1})$,
for $\mathsf{m}_i \in \{0, \ldots, w-1\}$
Compute checksum $C = \sum_{i=1}^{\ell_1}(w - 1 - \mathsf{m}_i)$,
and the base w representation $(C_1, \ldots, C_{\ell_2})$,
for $C_i \in \{0, \ldots, w-1\}$
Parse $B = \mathsf{m} \| C$ as $(b_1, \ldots, b_{\ell_1+\ell_2})$
Return 1, if the following equations hold
$\mathtt{vk}_0 = (\mathbf{r}, k)$
$\mathtt{vk}_i = c_k^{w-1-b_i}(\sigma_i, \mathbf{r}_{b_i+1,w-1})$ for $1 \le i \le \ell_1 + \ell_2$

Table 2. W-OTS$^+$ construction.

Additionally, we review the notation for the subset of randomness vector $\mathbf{r} = (r_1, \ldots, r_\ell)$. We denote by $\mathbf{r}_{a,b}$ the subset of $(r_a, \ldots, r_b)$.

Definition 4 (W-OTS$^+$). *Given the security parameter λ, a chaining function c, and $k \leftarrow \mathcal{K}$ from the key space $\mathcal{K}$, the W-OTS$^+$ signature scheme is the tuple $(\mathsf{Gen}_W, \mathsf{Sign}_W, \mathsf{Verify}_W)$, defined as in Table 2:*

The Security of W-OTS$^+$. The standard security notion for digital signature schemes is existential unforgeability under adaptive chosen message attacks (EU-CMA) which is defined using the following experiment. By $\mathrm{Dss}(1^\lambda)$ we denote a digital signature scheme (Dss) with security parameter λ, then we model the security by defining the security experiment $\mathsf{Exp}^{\mathsf{EU\text{-}CMA}}_{\mathsf{Dss}(1^\lambda)}(\mathcal{A})$, as follows:

Experiment $\mathsf{Exp}^{\mathsf{EU\text{-}CMA}}_{\mathsf{Dss}(1^\lambda)}(\mathcal{A})$

$(\mathsf{sk}, \mathsf{pk}) \longleftarrow \mathsf{keygen}(1^\lambda)$
$(M^*, \sigma^*) \longleftarrow \mathcal{A}^{\mathsf{Sign}(sk,\cdot)}(\mathsf{pk})$
Let $\{M_i, \sigma_i\}_1^q$ be the query-answer pairs of $\mathsf{Sign}(sk, \cdot)$
Return 1 iff $\mathsf{Verify}(pk, M^*, \sigma^*) = 1$ and $M^* \notin \{M_i\}_1{}^q$

We define the success probability of the adversary $\mathcal{A}$ in the above EU-CMA experiment as

$$\mathsf{Succ}^{\mathsf{EU\text{-}CMA}}_{\mathsf{Dss}(1^\lambda)}(\mathcal{A}) = \mathsf{Pr}[\mathsf{Exp}^{\mathsf{EU\text{-}CMA}}_{\mathsf{Dss}(1^\lambda)}(\mathcal{A}) = 1].$$

Definition 5 (EU-CMA). *Let $\lambda, t, q \in \mathbb{N}, t, q = poly(\lambda)$, Dss a digital signature scheme. We call Dss EU-CMA-secure, if the maximum success probability $\mathsf{InSec}^{EU\text{-}CMA}(Dss(1^\lambda); t, q)$ of all possibly probabilistic adversaries A, running in time $\leq t$, making at most q queries to Sign in the above experiment, is negligible in λ:*

$$\mathsf{InSec}^{EU\text{-}CMA}(Dss(1^\lambda); t, q) = max\ \{\mathsf{Succ}^{\mathsf{EU\text{-}CMA}}_{\mathsf{Dss}(1^\lambda)}(\mathcal{A})\} = negl(\lambda).$$

We note that our construction relies on the W-OTS$^+$ signature scheme, which is EU-CMA secure as long as the number of oracle queries of $\mathcal{A}$ is limited to one (i.e., $q = 1$).

Finally, we review a crucial property for the hash function which is a building block of the W-OTS$^+$ signature scheme.

Definition 6 (Second preimage resistance). *Given a hash function family $\mathcal{H}_n$, we define the success probability of an adversary $\mathcal{A}$ against the second-preimage resistance of $\mathcal{H}_n$ as*

$$\begin{aligned} Succ^{SPR}_{\mathcal{H}_n}(\mathcal{A}) = \mathsf{Pr}[K \xleftarrow{\$} \{0,1\}^k; M \xleftarrow{\$} \{0,1\}^m; \\ M' \xleftarrow{\$} \mathcal{A}(K, M) : M' \neq M \wedge H_K(M) = H_K(M')]. \end{aligned}$$

3 New Properties: Proof of Ownership and Fallback

Our protocol relies on a Digital Signature. We assume there is a generation algorithm $\mathsf{Gen}_\pi(1^\lambda)$ which outputs the pairs of keys, $\mathtt{vk}$ and $\mathtt{sk}$, and backup information $\mathtt{bk}$. Whereas the pair is the regular verification key, used for verifying a signature, and the secret-key used for issuing a signature, that allows the issuing of a *ownership proof* π, with the backup information $\mathtt{bk}$, with respect to $\mathtt{vk}$. More concretely, we require adding two extra algorithms, $(\mathsf{Proof}, \mathsf{Verify\text{-}proof})$, to the tuple $(\mathsf{Gen}_\pi, \mathsf{Sign}, \mathsf{Verify})$, turning into our protocol named $\mathcal{S}_{leeve}$. Given $\mathsf{Gen}_\pi(1^\lambda) \rightarrow (\mathtt{vk}, \mathtt{sk}, \mathtt{bk})$, we have

- $\mathsf{Proof}(\mathtt{bk}, c) \rightarrow \pi$: it is a PPT algorithm that on input of the backup information $\mathtt{bk}$ and the challenge c, it outputs the ownership proof π;

– $\mathsf{Verify\text{-}proof}(\mathtt{vk}, \mathtt{sk}, \pi, c) \rightarrow \{0, 1\}$: it is a deterministic algorithm that on input of a public-key $\mathtt{vk}$, secret-key $\mathtt{sk}$, a ownership proof π and a challenge c, it outputs either 0, for an invalid proof, or 1 for a valid one.

We remark that $\mathtt{sk}$ is used as a regular secret key with Sign and Verify. Given the earlier formulation, we now introduce the property of *Proof of Ownership*.

Definition 7 (Proof of Ownership). *For any probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, it holds*

$$\Pr[(\mathtt{vk}, \mathtt{sk}, \mathtt{bk}) \leftarrow \mathsf{Gen}_\pi(1^\lambda) : (c^*, \pi^*) \leftarrow \mathcal{A}(\mathtt{sk}, \mathtt{vk}) \\ \wedge \mathsf{Verify\text{-}proof}(\mathtt{vk}, \mathtt{sk}, \pi^*, c^*) = 1] < negl(\lambda)$$

for all the probabilities are computed over the random coins of the generation and proof verification algorithms and the adversary.

Remark 1 (Prove of knowledge is not enough). Note that an alternative method to prove ownership of a secret-key, fairly straightforward in discrete logarithm based signatures, relies on regular Zero Knowledge Proof of Knowledge Protocol (ZKP), when the signer simply proves the *knowledge* of the secret key. However we argue that, in the case where the secret key is leaked, the security guarantees are voided in such method. On the other hand, the early introduced definition requires a proof of ownership, despite the secret key being already in possession of the adversary, thus showing that *knowledge* of the secret key is not enough.

We now formally introduce the property which allows the permanent switch from the secret key $\mathtt{sk}$, *e.g.* in the case it is hopelessly public, to a brand new "back up secret key" $\mathtt{bk}$, that is, the new, and still protected, secret key is only known to the signer. In addition to the new secret key $\mathtt{bk}$, there is also a brand new signature scheme where the new verification key is the assumed leaked secret key $\mathtt{sk}$.

Remark 2 (Informal meaning of proof). Our earlier definition for Proof of Ownership is formally not a "proof" in the sense of ZKP. For example, it is easy to see we skip completeness and zero-knowledge like security properties. Still following the analogy, our introduced property is equivalent to the ZKP "soundness", and that is enough for our purposes.

Definition 8 (Fallback). *We say that the scheme* $(\mathsf{Gen}_\pi, \mathsf{Sign}, \mathsf{Verify})$*, with secret and verification key respectively* $\mathtt{sk}$ *and* $\mathtt{vk}$ *such that* $\mathsf{Gen}_\pi(1^\lambda) \rightarrow$

$(\mathtt{vk}, \mathtt{sk}, \mathtt{bk})$*, has fallback if there are sign and verification algorithms* Sign_π *and* Verify_π *such that* $\mathtt{sk}$ *and* $\mathtt{bk}$ *can be used as verification and secret keys respectively, along with* Sign_π *and* Verify_π *to satisfy Definition 5.*

Remark 3 (Use case for current blockchains-Tranfer of funds). It is worth mention that the current blockhain designs are not compatible with hash based signatures such as W-OTS$^+$. However our design could be used to authenticate to a third party, as, for example, in the case of [24]. Another alternative is to rely on the fallback feature, and the proof of ownership, to transfer the potentially endangered funds to an address or account of a newly created public/private key. Note in such a case, the ECDSA secret key could be exposed since the fund would be securely transferred to a new and safe pair of keys.

4 Protocol Design Overview

Our construction for the Proof of Ownership as presented in Section 3 is heavily based on the W-OTS$^+$ Signature Scheme. Before presenting how to construct such proofs, we detail the adaptation of the original construction, described in Definition 4, in order to introduce the Extended W-OTS$^+$ which will be used in combination with ECDSA.

4.1 Adaptation of W-OTS$^+$

Roughly speaking, our construction allows users to generate a quantum secure key pair and, from those values, derive an elliptic curve wallet to be used for cryptocurrency transactions. For simplicity of explanation, we assume the quantum-secure key material to be a W-OTS$^+$ key pair and the elliptic curve wallet to use the ECDSA algorithm. We note, however, that our construction can be easily extended to use other cryptographic primitives.

The L-Tree Data Structure. We rely on the data structure introduced by Dahmen et al. [10] to keep the W-OTS$^+$ public key. The L-Tree of height h stores 2^h leaves (such that $2^h \geq \ell + 1$, the size of W-OTS$^+$ public key). Each node of the tree is denoted by $y_i[j]$, for node index from left to right is $j = 0, \ldots, 2^i - 1$ and $i = 0, \ldots, h$, and the root is the node $y_0[0]$. The nodes of the tree are computed using a hash function H_x selected from a keyed hash family $\mathcal{H} = \{H_x : \{0,1\}^{2n} \rightarrow \{0,1\}^n\}_{x \in \mathcal{K}}$. On a given level i and node j of the tree, each input is computed by the concatenation of the left and right children nodes outputs, after each was

bit wise XORed by the masks $v_i[0]$ and $v_i[1]$, for n-bit strings chosen at uniformly at random. More formally, for $i = h, \ldots, 0$ and $j = 0, \ldots, 2^i - 1$, we have

$$y_i[j] = H_x((y_{i+1}[2j] \oplus v_i[0]) \| (y_{i+1}[2j+1] \oplus v_i[1])).$$

The Typical Combination of W-OTS$^+$ and L-Tree. Since its introduction in [10], L-Trees have been used in combination with hash based signature schemes [17]. For simplicity, we describe a typical combination between W-OTS$^+$ and L-Tree. Later we adapt this construction to suit our *Extended* W-OTS$^+$ proposal. The L-Tree construction introduces three extra sets of values for the verification key W-OTS$^+_{\mathtt{vk}}$, in addition to $\mathtt{vk} = (\mathtt{vk}_0, \mathtt{vk}_1, \ldots, \mathtt{vk}_\ell)$ as given by Table 2. They are

- The hash family index x;
- The XOR masks $v_i[0]$ and $v_i[1]$ for $i = 0, \ldots, h$;
- The root value $y_0[0]$.

In order to create a new wallet, a user randomly generates a cryptographicaly secure seed value, to be used to derive the W-OTS$^+$ public seed, the W-OTS$^+$ secret keys $(sk_1, ..., sk_\ell)$, and the hash key x. Once the derivation step is completed, clients use the chaining function to obtain all the W-OTS$^+$ ladder values. The top ladder values are the components of the W-OTS$^+$ public key, which are compressed into a single value using the earlier described L-Trees. Let this value be $L_{v,x}(\mathtt{vk}_0, \mathtt{vk}_1, \ldots, \mathtt{vk}_\ell)$ for the set of h XOR masks and hash family index x.

L-Tree and Extended W-OTS$^+$. We now propose a new construction for W-OTS$^+$, denoted Extended W-OTS$^+$ (eW-OTS$^+$). The motivation of the novel design is to allow the nesting of the W-OTS$^+$ public key into a regular ECDSA secret key, and yet allow the construction of proofs of ownership. This combination of keys will be presented later. The main differences between W-OTS$^+$ and the eW-OTS$^+$ designs are (1) the key generation algorithm incorporates the typical construction with L-Tree earlier described into the key generation, (2) the regular W-OTS$^+$ public key is changed to $\mathtt{pk}$, and (3) the secret key tuple has an extra term, *i.e.* $\mathtt{sk}_0$, instead of the regular terms $\mathtt{sk}_1, \ldots, \mathtt{sk}_\ell$. eW-OTS$^+$ is introduced because we assume that the nested W-OTS$^+$ public key is in the public domain and, without this extension, any adversary would be able to obtain the ECDSA secret key value by simply hashing the W-OTS$^+$ public key. The full construction is given by Definition 9.

Definition 9 (eW-OTS$^+$). *Given the security parameter λ, a chaining function c, and $k \leftarrow \mathcal{K}$ from the key space $\mathcal{K}$, an unkeyed hash function $\mathcal{H}$,*

$\mathsf{Gen}^k_{eW}(1^\lambda)$	$\mathsf{Sign}^k_{eW}(\mathsf{m}, \mathtt{sk})$
Pick $(\ell + w - 1)$ λ-bit strings r_i	Parse $\mathtt{sk} \rightarrow (\mathtt{sk}_0, \mathtt{sk}_1, \ldots, \mathtt{sk}_\ell)$
Pick a hash index family x	Parse $\mathtt{sk}_0 \rightarrow (v, x)$
Pick h pairs $v = (v_1[0], v_1[1], \ldots, v_h[0], v_h[1])$	Set $\sigma_0 = \mathtt{sk}_0$
$\mathtt{sk}_0 = (v_1[0], v_1[1], \ldots, v_h[0], v_h[1], x)$	Compute $\mathsf{m} \rightarrow (\mathsf{m}_1, \ldots, \mathsf{m}_{\ell_1})$,
Set $\mathtt{sk}_i \leftarrow r_i$, for $1 \leq i \leq \ell$	for $\mathsf{m}_i \in \{0, \ldots, w-1\}$
Set $\mathtt{sk} = (\mathtt{sk}_0, \mathtt{sk}_1, \ldots, \mathtt{sk}_\ell)$	Compute checksum $C = \sum_{i=1}^{\ell_1}(w - 1 - \mathsf{m}_i)$,
Set $\mathbf{r} = (r_{\ell+1}, \ldots, r_{\ell+w-1})$	w-base representation $(C_1, \ldots, C_{\ell_2})$,
Set $\mathtt{vk}_0 = (\mathbf{r}, k)$	for $C_i \in \{0, \ldots, w-1\}$
Set $\mathtt{vk}_i = c_k^{w-1}(\mathtt{sk}_i, \mathbf{r})$, $1 \leq i \leq \ell$	Parse $B = \mathsf{m} \| C$ as $(b_1, \ldots, b_{\ell_1+\ell_2})$
Set $L = L_{v,x}(\mathtt{vk}_1, \ldots, \mathtt{vk}_\ell)$	Set $\sigma_i = c_k^{b_i}(\mathtt{sk}_i, \mathbf{r})$, for $1 \leq i \leq \ell_1 + \ell_2$
Set $\mathtt{pk} = (\mathtt{vk}_0, L, \mathtt{sk}_0)$	Return $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_{\ell_1+\ell_2})$
Return $(\mathtt{sk}, \mathtt{pk})$	

$\mathsf{Verify}^k_{eW}(\mathsf{m}, \mathtt{pk}, \sigma)$
Parse $\mathtt{pk} \rightarrow (\mathtt{pk}_0, \mathtt{pk}_1, \mathtt{pk}_2)$
Parse $\mathtt{pk}_0 \rightarrow (\mathbf{r}, k)$
Parse $\sigma \rightarrow (\sigma_0, \sigma_1, \ldots, \sigma_{\ell_1+\ell_2}), \sigma_0 \rightarrow (v, x)$
Compute $\mathsf{m} \rightarrow (\mathsf{m}_1, \ldots, \mathsf{m}_{\ell_1})$,
for $\mathsf{m}_i \in \{0, \ldots, w-1\}$
Compute checksum $C = \sum_{i=1}^{\ell_1}(w - 1 - \mathsf{m}_i)$,
and the base w representation $(C_1, \ldots, C_{\ell_2})$,
for $C_i \in \{0, \ldots, w-1\}$
Parse $B = \mathsf{m} \| C$ as $(b_1, \ldots, b_{\ell_1+\ell_2})$
Set $\mathtt{vk}_i = c_k^{w-1-b_i}(\sigma_i, \mathbf{r}_{b_i+1,w-1})$ for $1 \leq i \leq \ell_1 + \ell_2$
Compute the L-Tree root as $L_{v,x}(\mathtt{vk}_1, \ldots, \mathtt{vk}_{\ell_1+\ell_2})$
Return 1, if the following equations hold
$\mathtt{pk}_1 = L_{v,x}(\mathtt{vk}_1, \ldots, \mathtt{vk}_{\ell_1+\ell_2})$
$\mathtt{pk}_2 = \sigma_0$

Table 3. Extended W-OTS$^+$ Signature Scheme with the changes from the original W-OTS$^+$ construction (Table 2) highlighted in blue. The changes introduced by our construction are necessary in order to be used in combination with ECDSA signatures.

then the eW-OTS$^+$ signature scheme is the tuple $(\mathsf{Gen}_{eW}, \mathsf{Sign}_{eW}, \mathsf{Verify}_{eW})$*, defined as in Table 3:*

Note that the Extended W-OTS$^+$ construction has as key pair $(\mathtt{sk}, \mathtt{pk})$ which differs from the regular construction $(\mathtt{sk}, \mathtt{vk})$ of W-OTS$^+$. The reader will certainly notice the need for the notation change in the public key from $\mathtt{vk}$ to $\mathtt{pk}$ in the next section, when the combination between ECDSA and eW-OTS$^+$ is described and we use both terms.

The ECDSA Key Pair from eW-OTS$^+$. We assume that the elliptic curve wallet is generated in a one-way manner, which means that if the ECDSA wallet is compromised, then the user can prove ownership of the

wallet by providing a signature from the eW-OTS$^+$ key pair, which is assumed secure. More formally $\mathtt{pk} = (\mathtt{vk}_0, L, \mathtt{sk}_0)$, as defined in Table 3, is the input in a unkeyed hash function $\mathcal{H}$, resulting in $\mathcal{H}(\mathtt{vk}_0, \mathcal{H}(L, \mathtt{sk}_0))$, which is the ECDSA private key, $\mathtt{sk}$, and can be used to calculate the public key using the trapdoor function of the signature scheme. The ECDSA public and secret key are, respectively, $\mathtt{sk}_{ECDSA} = \mathcal{H}(\mathtt{vk}_0, \mathcal{H}(L, \mathtt{sk}_0))$ and $\mathtt{vk}_{ECDSA} = g^{\mathcal{H}(\mathtt{vk}_0, \mathcal{H}(L, \mathtt{sk}_0))}$. Figure 1 illustrates a simplified diagram of our construction. Typically cryptocurrencies, such as Bitcoin [22], Ethereum [26], Cardano [2] and even general frameworks [19] for wallet address are built by hashing the ECDSA public key. Therefore, to integrate our construction in certain settings, an additional hash of the elliptic curve public key value is necessary, *i.e.* $\mathcal{H}(\mathtt{vk}_{ECDSA})$.

4.2 Ownership Proof Generation and Verification

As described in Section 3, the signature scheme that offers Proof of Ownership is a tuple $(\mathsf{Gen}_\pi, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Verify\text{-}proof})$, such that $\mathsf{Gen}_\pi(1^\lambda) \rightarrow (\mathtt{vk}, \mathtt{sk}, \mathtt{bk})$, $\mathsf{Proof}(\mathtt{bk}, c) \rightarrow \pi$ and $\mathsf{Verify\text{-}proof}(\mathtt{vk}, \mathtt{sk}, \pi, c) \rightarrow \{0, 1\}$. In order to construct such scheme we combine the ECDSA and eW-OTS$^+$ designs. The generation and verification of signatures are respectively carried by Sign and Verify as regular ECDSA signatures. The proof of ownership is put forth by the eW-OTS$^+$ design via the Proof and $\mathsf{Verify\text{-}proof}$ algorithms. Put simply, the tuple $(\mathtt{vk}, \mathtt{sk})$ is the regular ECDSA key pair, such that $\mathtt{sk}$ is generated from the underpinning eW-OTS$^+$ public key $\mathtt{pk}$. Whereas $\mathtt{bk}$ is the eW-OTS$^+$ secret key $(\mathtt{sk}_0, \mathtt{sk}_1, \ldots, \mathtt{sk}_\ell)$ corresponding to $\mathtt{pk}$.

It remains to introduce the Gen_π algorithm to generate the tuple $(\mathtt{vk}, \mathtt{sk}, \mathtt{bk})$.

The Generation of the "back up key". Intuitively, the proof of ownership of the key, requires similar properties of an identification scheme between a prover and a verifier instantiated by a particular signature scheme. In our construction, the identification scheme is based on the earlier introduced eW-OTS$^+$ design. More concretely, given a challenge as a message provided by the verifier, the prover only needs to sign this message with $\mathtt{bk}$. As described earlier, let the ECDSA key pair be $\mathtt{sk} = \mathcal{H}(\mathtt{pk})$, for $\mathtt{pk} = (\mathtt{vk}_0, L, \mathtt{sk}_0)$, and $\mathtt{vk} = g^{\mathcal{H}(\mathtt{vk}_0, \mathcal{H}(L, \mathtt{sk}_0))}$ for an unkeyed hash function $\mathcal{H}$. Therefore the "back up secret key" $\mathtt{bk}$ is $(\mathtt{sk}_0, \mathtt{sk}_1, \ldots, \mathtt{sk}_\ell)$, the eW-OTS$^+$ secret-key, is illustrated in Table 3. For completeness, we present the construction of algorithm $\mathsf{Gen}_\pi(1^\lambda) \rightarrow (\mathtt{vk}, \mathtt{sk}, \mathtt{bk})$. Note that

in the following construction, the key pair $(\mathtt{vk}, \mathtt{sk})$ can be used as regular signature (*i.e.* for the ECDSA Signature), that is with algorithms $(\mathsf{Sign}, \mathsf{Verify})$ as in Table 1.

Generation and Verification of the Proof π. Whereas the regular ECDSA signatures are generated and verified via the pair of algorithms $(\mathsf{Sign}, \mathsf{Verify})$ and the keys $(\mathtt{sk}, \mathtt{vk})$ generated via construction Table 4. The proof generation and verification are done via the algorithms for W-OTS$^+$ described by Table 3. More concretely, the algorithm $\mathsf{Proof}^k(\mathtt{bk}, c)$, for a challenge c, is implemented by the algorithm Sign_{eW}, whereas the proof verification $\mathsf{Verify\text{-}proof}(\mathtt{vk}, \mathtt{sk}, \pi, c)$ is based on an adaptation of the signature verification $\mathsf{Verify}^k_{eW}(\mathtt{pk}, \sigma, \mathsf{m})$. The full description of the procedure is on Table 5.

We argue that the construction can be extended to provide multiple proofs of ownership by adding more eW-OTS$^+$ instances "underneath" the main one presented in Table 4. For the purpose of this work and also for page limitation, it is not necessary to fully describe the algorithms. However we present an informal description of the construction later in Section 9.

Practical Considerations. The back up key $\mathtt{bk}$ is not necessary to the regular use in combination with the ECDSA scheme, *i.e.* the blockchain

$\mathsf{Gen}^k_\pi(1^\lambda)$
Pick uniform random strings $(\ell + w - 1)$ λ-bit strings r_i
Set $\mathtt{bk}_i \leftarrow r_i$, for $1 \leq i \leq \ell$
Pick a hash index family x
Pick n-bit random masks $v_i[0]$ and $v_i[1]$, for $i = 0, \ldots, \log \ell$
Set $\mathtt{bk}_0 = (v_1[0], v_1[1], \ldots, v_{\log \ell}[0], v_{\log \ell}[1], x)$
Set $\mathtt{bk} = (\mathtt{bk}_0, \mathtt{bk}_1, \ldots, \mathtt{bk}_\ell)$
Set $\mathbf{r} = (r_{\ell+1}, \ldots, r_{\ell+w-1})$
Set $\mathtt{vk}_0 = (\mathbf{r}, k)$
Set $\mathtt{vk}_i = c_k^{w-1}(\mathtt{bk}_i, \mathbf{r})$, $1 \leq i \leq \ell$
Set nodes $y_i[j]$ for $j = 0, \ldots, \ell - 1$ and $i = \log \ell, \ldots, 0$ as $y_{\log \ell}[0] = \mathcal{H}_x(\mathtt{vk}_1), \ldots, y_{\log \ell}[\ell - 1] = \mathcal{H}_x(\mathtt{vk}_\ell)$ $y_i[j] = H_x((y_{i+1}[2j] \oplus v_i[0]) \| (y_{i+1}[2j+1] \oplus v_i[1]))$
Set $L = y_0[0]$
Set $\mathtt{sk} = (\mathtt{vk}_0, L, \mathtt{bk}_0)$
Set $\mathtt{vk} = g^{\mathcal{H}(\mathtt{vk}_0, \mathcal{H}(L, \mathtt{bk}_0))}$
Return $(\mathtt{sk}, \mathtt{vk}, \mathtt{bk})$

Table 4. The algorithm Gen_π, likewise the eW-OTS$^+$ construction, adds a L data structure into its procedure, and outputs also the "back up secret key" $\mathtt{bk}$.

Verify-proof$(\mathtt{vk}, \mathtt{sk}, c, \pi)$
Parse $\mathtt{sk} \rightarrow (\mathtt{sk}_0, \mathtt{sk}_1, \mathtt{sk}_2)$
Parse $\mathtt{sk}_0 \rightarrow (\mathbf{r}, k)$
Parse $\pi \rightarrow (\pi_0, \pi_1, \ldots, \pi_{\ell_1+\ell_2}), \pi_0 \rightarrow (v, x)$
Compute $c \rightarrow (c_1, \ldots, c_{\ell_1})$,
for $c_i \in \{0, \ldots, w-1\}$
Compute checksum $C = \sum_{i=1}^{\ell_1}(w-1-c_i)$,
and the base w representation $(C_1, \ldots, C_{\ell_2})$,
for $C_i \in \{0, \ldots, w-1\}$
Parse $B = c\|C$ as $(b_1, \ldots, b_{\ell_1+\ell_2})$
Set $\mathtt{vk}_i = c_k^{w-1-b_i}(\pi_i, \mathbf{r}_{b_i+1,w-1})$ for $1 \leq i \leq \ell_1 + \ell_2$
Compute the L-Tree root as $L = L_{v,x}(\mathtt{vk}_1, \ldots, \mathtt{vk}_{\ell_1+\ell_2})$
Return 1, if the following equations hold
$\mathtt{sk}_1 = L$
$\mathtt{sk}_2 = \pi_0$
$\mathtt{vk} = g^{\mathcal{H}(\mathtt{sk}_0, \mathcal{H}(L, \pi_0))}$

Table 5. The verification of the proof π adapts the verification procedure for eW-OTS$^+$ by adding an extra check on the ECDSA verification key `vk`.

use. In order to guarantee a secure and legit use of the `bk`, that is the generation of proof of ownership in case of a catastrophic leakage, `bk` should be kept in a separate storage, *i.e.* cold storage.

5 Ownership, Fallback and eW-OTS$^+$ Security

Here we argue about the properties of our construction for $\mathcal{S}_{leeve}$, providing *Fallback* and generation of *Proof of Ownership*. However, first we describe the security level provided by our design based on the eW-OTS$^+$ construction of Table 3, and we consider it has a security level λ if a successful attack is expected to require on average $2^{\lambda-1}$ evaluations of the used hash function family.

Unforgeability of eW-OTS$^+$. More concretely, we base the security of the extended W-OTS$^+$ on the existential unforgeability of the underlying W-OTS$^+$ signature scheme and the (multi-target) second preimage resistance of the used hash function. Recall that the existential unforgeability under chosen message attack (EU-CMA) for one-time signature schemes is defined when the number of signature queries is limited to 1 [16]. Then we have the following theorem.

Theorem 1. *Given the EU-CMA security of W-OTS$^+$, the Extended W-OTS$^+$ from Table 3 is existentially unforgeable under adaptive chosen*

message attacks, if $\mathcal{H}_n$ is from a second-preimage resistant hash function family.

Proof. W-OTS$^+$ uses a family of functions $\mathcal{F}_n : \{f_k : \{0,1\}^n \to \{0,1\}^n | k \in \mathcal{K}_n\}$ with a key space $\mathcal{K}_n$. We know from [16] that, to attack the EU-CMA property, an adversary $\mathcal{A}$ must be able to break the following security level, λ_1, such that $\lambda_1 \geqslant n - \log_2(w^2\ell + w)$.

Alternatively, $\mathcal{A}$ may attempt to subvert the underlying hash function $\mathcal{H}$ we introduce in our construction.

To successfully attack this additional step introduced in the extended W-OTS$^+$ and produce a forged signature, $\mathcal{A}$ must break the second-preimage resistance property of $\mathcal{H}$ and find a colliding $L(\text{W-OTS}^+_{\mathtt{vk}})'$ that matches the target hash output.

We show in Appendix A that the cost of this attack for an n-bit hash is 2^n. Additionally, we know that in a real-world cryptocurrency setting, the adversary has the advantage of being able to perform multi-target attacks on any of the existing d outputs, which results in the following security level of $\lambda_2 \leqslant n_1 - log_2(d)$. Given the above tight bounds, we obtain the security level (λ) of the extended W-OTS$^+$, which is $\lambda \leq min\ \{\lambda_1, \lambda_2\}$.

For simplicity, we assume that $n = n_1$. We, therefore, obtain that the best attack against the extended W-OTS$^+$ construction is the same attack against the original W-OTS$^+$. As a result, if the adversary is able to break the EU-CMA property of the extended W-OTS$^+$, then it can break the unforgeability of the original W-OTS$^+$. Therefore, our construction is no weaker than the original as long as the output of the used hash function $\mathcal{H}$ is $n_1 \geqslant n$. □

Security regarding Ownership and Fallback. Now we describe the security of our scheme ($\mathsf{Gen}_\pi, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Verify\text{-}proof}$). Given the eW-OTS$^+$ construction from Table 9, let Sign_{eW} be the algorithm Proof, Gen_π is given by Table 4, while $\mathsf{Verify\text{-}proof}$ is given by Table 5, finally Sign and Verify are the ECDSA algorithms for signing and verifying signatures, respectively. For readability, let ($\mathsf{Gen}_\pi, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Verify\text{-}proof}$) be known as $\mathcal{S}_{leeve}$ (as already mentioned in Section 3). Then we can claim the following properties of $\mathcal{S}_{leeve}$.

Corollary 1. *$\mathcal{S}_{leeve}$ generates a single proof of ownership π as per Definition 7.*

Proof. (Sketch) The proof π is an eW-OTS$^+$ signature on a challenge c. Given the security of eW-OTS$^+$ stated by Theorem 1, thereby π generated by $\mathcal{S}_{leeve}$ satisfies Definition 7. □

Now we show that tuple $(\mathsf{Gen}_\pi, \mathsf{Sign}, \mathsf{Verify})$, parsed from $\mathcal{S}_{leeve}$ provides a Fallback signature scheme.

Theorem 2. *The tuple* $(\mathsf{Gen}_\pi, \mathsf{Sign}, \mathsf{Verify})$*, derived from* $\mathcal{S}_{leeve}$*, has the Fallback property as per Definition 8.*

Proof. (Sketch) Following Definition 8, we need to show that there are algorithms Sign_π and Verify_π, such that `sk` and `bk` can derive regular verification and secret signatures. By considering the original construction $\mathcal{S}_{leeve}$, we have that $\mathsf{Sign}_\pi = \mathsf{Proof}$ and $\mathsf{Verify}_\pi = \mathsf{Verify\text{-}proof}$, and this ends the proof. □

6 Applications

We divide this section into two parts. First, we describe a concrete example of how to proceed upon the suspicion of the existence of a successful attack against the computational assumptions that ensure the security of the ECDSA algorithm. Secondly, we introduce different real-world use cases that allow users to prove ownership of their wallet in the event where the ECDSA secret key is leaked, but the $\mathcal{S}_{leeve}$ backup key remains safe.

Hard Fork. If an attacker $\mathcal{A}$ is able to steal the secret keys behind a cryptocurrency wallet, then $\mathcal{A}$ is able to steal all the funds associated with that wallet. Since a $\mathcal{S}_{leeve}$ proof-of-ownership does not convince $\mathcal{A}$ to return the stolen funds, at first glance it may appear that there is no reason to have a fallback mechanism as all these funds are gone.

$\mathcal{S}_{leeve}$ is better suited for situations where the signature scheme associated with the quantum-secure backup can be used as a direct replacement for the original scheme. In a blockchain, this signature transition is only possible by making significant changes in the protocol, which create an alternative blockchain. This process is known as a hard fork. Using our construction, any blockchain can perform a signature scheme transition and allow any user to claim ownership of potentially stolen funds. As quantum algorithms become practical, blockchain platforms can recommend their users to create new wallet addresses using the $\mathcal{S}_{leeve}$ structure such that, when a hard fork is required, any user can produce a proof-of-ownership to transfer the funds to an address containing a new public key.

$\mathcal{S}_{leeve}$ becomes even more applicable in token sales settings where the smart contracts enforce lockup periods to restrict buyers from selling

purchased tokens. If there is evidence of a quantum attack against a blockchain, users can utilize the $\mathcal{S}_{leeve}$ construction to prove ownership of potentially compromised tokens and redeem them in an alternative manner while the lockup period is still active, thus ensuring that no theft occurs.

Revoking Wallet Addresses. It is extremely important for users to have the ability to revoke a wallet address they own. Therefore, user Alice should have the ability notify the network that a specific wallet address is to be considered invalid and rejected by the nodes when attempting to make a payment. Alice, in this example, has her ECDSA secret key stolen and revokes her stolen wallet address by creating a proof-of-ownership, using her backup key, to inform the network that her address is compromised and, simultaneously introduce a new wallet address to contain the funds associated with the initial wallet.

Insurance. An insurance company may, for example, need to refund a group of protected customers after a set of ECDSA private keys are leaked and the associated funds stolen. Any user whose key is present in this leak, if in possession of their $\mathcal{S}_{leeve}$ backup key, can remotely prove that they are the true owners of a specific wallet address and prove to the insurance company that they are entitled to the refund. The insurance company knows that an adversary is not able to produce such a forgery unless both keys are compromised.

7 Discussion

In this section we briefly discuss selected issues and analyze open problems as well as some future work challenges.

Fail-stop Signatures. Traditional digital signatures allow a user Alice to produce signatures such that everyone who knows the public key of the signer Alice can verify such signature. Such signatures are computationally secure for the signer as they can be forged by an adversary with quantum capabilities. Once a signature is forged, it is difficult for the honest signer Alice to convince third parties that she did not produce the forged signature.

Fail-stop signatures [25] solve this problem by offering the signer a method to prove that a forgery took place. After receiving such a proof, the system should be stopped. As a result, the signer is protected from

an arbitrarily powerful forgery since all participants, or an eventual system operator, know the signature scheme is broken, and should halt the system.

A possible enhancement for $\mathcal{S}_{leeve}$ is to alter the key generation to support the the integration of fail-stop signatures. Instead of generating an ECDSA keypair from a hash-based key, users can generate a fail-stop keypair as this would allow a user to prove that a rogue signature is indeed a forgery and therefore instantiate the backup key to prove the true ownership of a key pair.

Tweakable Hash Functions. In hash-based signature schemes, it is important to use constructions that use security notions such as second-preimage and preimage resistance instead of collision resistance. Different hash-based schemes focus on different ways to achieve these more specific security notions as they substantially enhance the security level of the produced signatures. However, the main idea behind the different constructions to achieve these specific security notions is similar enough that it is possible to create an abstraction, such that it is not necessary to provide a new security analysis for each of the alternatives to move towards (second) preimage resistance.

The work from [5] introduces an abstraction—called tweakable hash functions—which allows protocol designers to unify the description of hash-based signature schemes, separating the exact details of how the scheme computes tree nodes typically used in hash-based constructions. This division allows for the separation of the analysis of the high-level construction from the analysis of how this computation is done. As a result, changing the way nodes are computed in a hash-based signature scheme only requires analyzing the hashing construction as a tweakable hash function.

One optimization we introduce is the use of tweakable hash functions to compress all the W-OTS$^+$ top ladder values into a single root value, which results in a more simplified implementation with better performance.

Cold Storage. Using $\mathcal{S}_{leeve}$ does not necessarily imply that both the ECDSA secret key and the backup key should be stored in different cold storage units. For example, a quantum adversary can gain rogue access to an ECDSA secret key by breaking the discrete log problem using only public information such as the public keys that are present in a blockchain.

In this setting, the fallback key remains securely stored and can be freely used by the wallet owner.

To increase the security of $\mathcal{S}_{leeve}$ it is possible, however, to use different storage for the secret key and the backup key to ensure that if a wallet owner moves the ECDSA secret key to a hot wallet, and such a wallet is compromised, then the owner remains protected as the adversary $\mathcal{A}$ should not be able to gain access to the cold wallet containing the $\mathcal{S}_{leeve}$ backup key.

Backwards Compatibility. Ideally, users should be able to use the ideas behind $\mathcal{S}_{leeve}$ to use the seed phrase of a hierarchical deterministic wallet and retroactively prove ownership of a specific wallet address. The feasibility of this remains undefined and represents an interesting future work challenge as it would allow any user to utilize this approach and have the ability to prove ownership of wallet address with guaranteed backwards compatibility with any wallet that supports the use of seed phrases to generate hierarchical deterministic wallets.

We note that our construction preserves the structure of both the ECDSA private and public keys, and if the user actually relies on two different cold storage solutions—one for the ECDSA key and the other for the $\mathcal{S}_{leeve}$ backup key—then it is possible to achieve backwards compatibility as the storage of the ECDSA key pair does not require any particular or different treatment.

To support the $\mathcal{S}_{leeve}$ backup key, however, both the wallets and the blockchain require protocol modifications. Wallets require modification to have the ability to generate hash-based signatures, while the blockchain needs to be modified to have the ability of verifying these hash-based signatures.

Compatibility with different post-quantum signature schemes. $\mathcal{S}_{leeve}$ is designed in a modular manner that allows the hiding of any quantum secure signature key pair, and is not exclusive to W-OTS$^+$. In this paper, we particularly focus on W-OTS$^+$ as a fallback for ECDSA because it corresponds to the real-world use case that inspired the creation of this construction. Platforms, however, have the flexibility to use different signature schemes accordingly.

Informal Multiple Proofs Construction. The construction introduced in Section 4 allows only a single proof. The reason is that eW-OTS$^+$

signature scheme is one-time signature scheme. Here we informally describe a construction to allow the generation of several proofs. The basic change is in the generation of the secret-key tuple $\mathtt{bk}_0, \mathtt{bk}_1, \ldots, \mathtt{bk}_\ell$. Whereas in the previous constructions of Tables 3 and 4 the values in the tuple are picked at random, the extended version computes t tuples, where each set of values $(\mathtt{bk}_0^{(j-1)}, \mathtt{bk}_1^{(j-1)}, \ldots, \mathtt{bk}_\ell^{(j-1)})$ is generated from executing a *Key Derivation Function* (KDF) from the previous tuple $(\mathtt{bk}_0^{(j)}, \mathtt{bk}_1^{(j)}, \ldots, \mathtt{bk}_\ell^{(j)})$, for $j \leq t$. More concretely, $\mathtt{bk}_1^{(j-1)} = KDF(L^{(j)})$, and $\mathtt{bk}_i^{(j-1)} = KDF(L^{(j)}||salt^{(j-1)}||i)$ for $1 < i \leq \ell$, randomly chosen values $salt^{(j-1)}$ and the L-Tree root value $L^{(j)}$ of the underlying construction *i.e.* eW-OTS$^+$ instance with index j. Thus, for a t-backup key value construction, the generation is as follows:

- Pick $\mathtt{bk}_0^{(j)} = (x^{(j)}, v_1^{(j)}[0], v_1^{(j)}[1], \ldots, v_{\log \ell}^{(j)}[0], v_{\log \ell}^{(j)}[1])$, for $1 \leq j \leq t$;
- Pick $\mathtt{vk}_0^{(j)} = (\mathbf{r}^{(j)}, k^{(j)})$, for $1 \leq j \leq t$, and $(\mathbf{r}^{(j)}, k^{(j)})$ chosen as $(\mathbf{r}, k)$ in Table 4;
- Pick random values $(\mathtt{bk}_1^{(j)}, \ldots, \mathtt{bk}_\ell^{(j)})$;
- Given $\mathtt{bk}_0^{(j)}$ and $(\mathtt{bk}_1^{(j)}, \ldots, \mathtt{bk}_\ell^{(j)})$, compute $L^{(j)}$;
- Compute $\mathtt{bk}_1^{(j-1)} = KDF(L^{(i)})$, and $\mathtt{bk}_i^{(j-1)} = KDF(L^{(j)}||salt^{(j-1)}||i)$ for $1 < i \leq \ell$, $t \geq j \geq 1$ and randomly chosen values $salt^{(j-1)}$.

The intuition is to add $t-1$ eW-OTS$^+$ constructions "underneath" the upmost one. The public key of the underlying eW-OTS$^+$ instance, generates, via KDF (which can be constructed by a hash function), the secret key of the next (*i.e.* $\mathtt{bk}_1^{(j-1)}$, the last line of the above description).

The verification algorithm for such multiple construction has to take into account in which "level" (from t to 0, in the above description) the signature was generated, and be continually updated on each new signature generation. For comparison, the construction for a single proof only has one level. The "multilevel" p-th proof is of the form

$$\pi = ((\pi_0, \ldots, \pi_{\ell_1+\ell_2}), (\mathtt{vk}_0^{(1)}, \mathtt{sk}_0^{(1)}, L^{(1)}, salt^{(1)}), \ldots, (\mathtt{vk}_0^{(p+1)}, \mathtt{sk}_0^{(p+1)}, L^{(p+1)}, salt^{(p+1)})),$$

for $\mathtt{vk}_0^{(p)} = (v^{(p)}, x^{(p)})$ and $\mathtt{sk}_0^{(p)} = (\mathbf{r}^{(p)}, k^{(p)})$. Thus the verification procedure transverse the underneath structure of eW-OTS$^+$ instances from some point p, *i.e.* the p-th proof, up to the upmost one 1. Roughly the procedure is as follows:

- Compute $\mathtt{vk}_i^{(p-1)} = c_k^{w-1-b_i}(\pi_i, \mathbf{r}_{b_i+1,w-1}^{(p-1)})$;
- Compute $L^{(p-1)} = L_{v^{(p-1)}, x^{(p-1)}}(\mathtt{vk}_1^{(p-1)}, \ldots, \mathtt{vk}_{\ell_1+\ell_2}^{(p-1)})$.

For $p - 1 < j \leq 1$,
- Compute $\mathtt{sk}_i^{(j)} = KDF(L^{(j-1)}||salt^{(j-1)}||i)$;
- Compute $\mathtt{vk}_i^{(j)} = c_{k^{(j)}}^{w-1}(\mathtt{sk}_i^{(j)}, \mathbf{r}_{b_i+1,w-1}^{(j)})$;
- Compute $L^{(j)} = L_{v^{(j)},x^{(j)}}(\mathtt{vk}_1^{(j)}, \ldots, \mathtt{vk}_{\ell_1+\ell_2}^{(j)})$,

at this point the verification boils down to the correctness of the value $L = L^{(1)}$ as before.

8 Experimental Results

To validate our results, we implemented a single-threaded prototype in Golang [13].

We note that this implementation does not combine the W-OTS$^+$ public key values using an L-Tree structure. Instead, our implementation uses a tweakable hash function to combine all the W-OTS$^+$ ladder top values into a single root value. Since our construction has a very concrete application, we implemented an additional implementation variant that includes the BIP 39 [6] standard to generate the hidden W-OTS$^+$ fallback from a mnemonic seed. We verified the correctness of this code by comparing it with reference BIP39 implementations [7,14].

We ran our experiments on a 2.8 GHz Quad-Core Intel Core i7 with 16GB of RAM, running 64-bit macOS 10.15.6. Below, we expose a table containing the corresponding performance of our prototype.

	Execution time (ms)		
Algorithm	**Gen**	**Sign**	**Verify**
$\mathcal{S}_{leeve}$	3.87	0.024	1.472
$\mathcal{S}_{leeve}$ w/ BIP39	7.51	0.024	1.472
ECDSA (on secp256r1)	0.77	0.069	0.084

Table 6. Performance metrics of our custom implementation.

These timings demonstrate that the key generation component of our design is significantly slower than presently used key generation mechanisms. Depending on the protocol instantiation, our key generation is between 5 to 10 times slower than a normal ECDSA key generation algorithm. We highlight, however, that this is an expected result given the amount of additional steps introduced by our construction. We also note that the key generation can be easily accelerated by performing the

W-OTS$^+$ ladder calculations in parallel. Regarding key storage, our construction utilizes the same storage space as a normal ECDSA private key. For example, the wallet storage of a Bitcoin secret key would require 256-bits for both the $\mathcal{S}_{leeve}$ construction and for a normal wallet.

9 Conclusion

We proposed $\mathcal{S}_{leeve}$ as a new approach to integrate a quantum-secure fallback inside an elliptic curve private key. The core idea is to have a hidden hash-based signing key pair. The users can show they are the rightful owner of the cryptocurrency secret keys even in the presence of an adversary capable of breaking the elliptic curve discrete logarithm problem, which is not a possibility using any of the existing curve-based cryptocurrency wallets. Moreover in catastrophic scenarios, where a massive leakage has potentially happened, and system is halted, users can show to trusted third parties that they are the correct owners of the wallet.

Along with $\mathcal{S}_{leeve}$, we presented also novel ideas for security guarantees and security analysis, aspiring that they will stimulate additional discussion, and potential improvements in the cryptocurrency wallet research community. As another contribution to the above mentioned discussion, we argue that the $\mathcal{S}_{leeve}$ construction can be changed to scale, in the sense that it can be extended to provide multiples proofs of ownership, *i.e.* π_i for $t \geq i$ or to provide multiple signatures while in "fallback mode", or even $\mathcal{S}_{leeve}$ can be used combined with Fail-stop Signatures. As a final remark, we recall that although we presented a construction based on W-OTS$^+$ signature scheme, we believe other hash based signature schemes can be adapted in similar fashion.

References

1. Myrto Arapinis, Andriana Gkaniatsou, Dimitris Karakostas, and Aggelos Kiayias. A formal treatment of hardware wallets. In Ian Goldberg and Tyler Moore, editors, *FC 2019: 23rd International Conference on Financial Cryptography and Data Security*, volume 11598 of *Lecture Notes in Computer Science*, pages 426–445, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019. Springer, Heidelberg, Germany.
2. Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 913–930, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

3. Gustavo Banegas and Daniel J. Bernstein. Low-communication parallel quantum multi-target preimage search. In Carlisle Adams and Jan Camenisch, editors, *SAC 2017: 24th Annual International Workshop on Selected Areas in Cryptography*, volume 10719 of *Lecture Notes in Computer Science*, pages 325–335, Ottawa, ON, Canada, August 16–18, 2017. Springer, Heidelberg, Germany.
4. Daniel J. Bernstein and Andreas Hülsing. Decisional second-preimage resistance: When does SPR imply PRE? In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 33–62, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany.
5. Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS$^+$ signature framework. In Cavallaro et al. [9], pages 2129–2146.
6. Mnemonic code for generating deterministic keys. `https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki`. Accessed: 2020-01-20.
7. Mnemonic code converter. `https://iancoleman.io/bip39/`. Accessed: 2020-01-20.
8. Study finds less than 40% of btc addresses are economically relevant. `https://news.bitcoin.com/study-finds-less-than-40-of-btc-addresses-are-economically-relevant/`. Accessed: 2020-09-18.
9. Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors. *ACM CCS 2019: 26th Conference on Computer and Communications Security*. ACM Press, November 11–15, 2019.
10. Erik Dahmen, Katsuyuki Okeya, Tsuyoshi Takagi, and Camille Vuillaume. Digital signatures out of second-preimage resistant hash functions. In Johannes Buchmann and Jintai Ding, editors, *Post-quantum cryptography, second international workshop, PQCRYPTO 2008*, pages 109–123, Cincinnati, Ohio, United States, October 17–19 2008. Springer, Heidelberg, Germany.
11. Poulami Das, Sebastian Faust, and Julian Loss. A formal treatment of deterministic wallets. In Cavallaro et al. [9], pages 651–668.
12. Ethereum unique addresses chart. `https://etherscan.io/chart/address`. Accessed: 2020-09-18.
13. Implementation of wots up my sleeve. `https://github.com/yaksetig/sleeve`. Accessed: 2021-04-01.
14. Golang implementation of the bip39 spec. `https://godoc.org/github.com/tyler-smith/go-bip39`. Accessed: 2020-09-20.
15. Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th Annual ACM Symposium on Theory of Computing*, pages 212–219, Philadephia, PA, USA, May 22–24, 1996. ACM Press.
16. Andreas Hülsing. W-OTS+ - shorter signatures for hash-based signature schemes. In Youssef et al. [27], pages 173–188.
17. Andreas Hülsing, Joost Rijneveld, and Fang Song. Mitigating multi-target attacks in hash-based signatures. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 387–416, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany.
18. Michael Hutter and Peter Schwabe. NaCl on 8-bit AVR microcontrollers. In Youssef et al. [27], pages 156–172.

19. Dimitris Karakostas, Aggelos Kiayias, and Mario Larangeira. Account management in proof of stake ledgers. In Clemente Galdi and Vladimir Kolesnikov, editors, *Security and Cryptography for Networks*, pages 3–23, Cham, 2020. Springer International Publishing.
20. Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.
21. Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.
22. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.
23. Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, Oct 1997.
24. Trinity attack incident part 1: Summary and next steps. `https://blog.iota.org/trinity-attack-incident-part-1-summary-and-next-steps-8c7ccc4d81e8`. Accessed: 2020-09-22.
25. Eugène van Heyst and Torben P. Pedersen. How to make efficient fail-stop signatures. In Rainer A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT'92*, volume 658 of *Lecture Notes in Computer Science*, pages 366–377, Balatonfüred, Hungary, May 24–28, 1993. Springer, Heidelberg, Germany.
26. Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.
27. Amr Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien, editors. *AFRICACRYPT 13: 6th International Conference on Cryptology in Africa*, volume 7918 of *Lecture Notes in Computer Science*, Cairo, Egypt, June 22–24, 2013. Springer, Heidelberg, Germany.

A Generic Attacks

The early presented constructions are hash based ones, therefore in this section we present an extensive list of computational complexities of various generic attacks against hash functions, while relating them with our constructions. Later we rely on these complexities to analyse and prove security of our proposed signature scheme.

Preimage resistance. The adversary $\mathcal{A}$ may obtain a hash digest and attempt to invert the one-way property of the used hash function. Assuming that the inputs are uniform random n-bit values, then this preimage attack costs 2^n in the classical setting. In the post-quantum setting, using Grover's algorithm, this attack costs $2^{n/2}$.

Second preimage resistance (SPR). The adversary may instead attempt to find a second preimage of an n-bit message. Assuming a non-compressing hash function, that is, there is at least an n-bit-to-n-bit preimage to hash mapping, then this attack costs 2^n in the classic setting, and $2^{n/2}$ in the post-quantum setting.

Enhanced target collision resistance (eTCR). The notion of eTCR implies that an adversary is allowed to choose a target message M. Upon choosing this target message, $\mathcal{A}$ learns the function $\mathcal{H}_K$ (by learning the key K) and the adversary wins after presenting a new message M' and a (possibly new) key K' such that $H_K(M) = H_{K'}(M')$.

A possible application of the eTCR game in our setting involves the adversary committing to a $L(\text{W-OTS}^+_{\mathtt{vk}})$ public key value and then obtaining the hash function key. There are two ways an adversary may attempt to break the eTCR property of a hash function. First, $\mathcal{A}$ may attempt to obtain a new $L(\text{W-OTS}^+_{\mathtt{vk}})'$ such that $H_K(L(\text{W-OTS}^+_{\mathtt{vk}})) = H_K(L(\text{W-OTS}^+_{\mathtt{vk}})')$. Second, $\mathcal{A}$ may attempt to obtain a new key K' and $L(\text{W-OTS}^+_{\mathtt{vk}})'$ such that $H_K(L(\text{W-OTS}^+_{\mathtt{vk}})) = H_{K'}(L(\text{W-OTS}^+_{\mathtt{vk}})')$.

If $\mathcal{A}$ owns the secret keys corresponding to the colliding $L(\text{W-OTS}^+_{\mathtt{vk}})'$, then $\mathcal{A}$ can forge a proof of ownership of the target wallet. This forgery costs at least 2^n pre-quantum, $2^{n/2}$ post-quantum (Grover's algorithm), and results in the adversary having the ability to prove ownership of an elliptic curve based wallet with a different fallback public key. We highlight, however, that even if the adversary can find a second preimage, it is not guaranteed that it corresponds to a $L(\text{W-OTS}^+_{\mathtt{vk}})'$ actually controlled by $\mathcal{A}$.

Multi-target attacks. The previous definitions assume an adversary attacking one single target. We assume a hash function with n-bit outputs is used d times and each of these d outputs is publicly posted (*e.g.*, on a blockchain). The adversary $\mathcal{A}$ may, therefore, attempt to invert any of these public d values, which results in an attack complexity of $2^{n-log_2(d)}$ instead of 2^n. In order to show the effectiveness of a multi-target attack, we consider the case where all the secret keys associated with the wallet addresses are publicly exposed and are generated using our hidden key construction.

This setting results in a leakage of approximately 2^{29} target wallet addresses, for example [12] [8], which results in an attack complexity cost of 2^{n-29}. Typically, ECDSA secret keys of 256 bits. Therefore, a multi-target attack in the setting we describe results in a direct loss of 29 bits in security, resulting in a cost of 2^{227} instead of 2^{256}. In a post-quantum setting[4], however, the adversary must perform $2^{n/2}/\sqrt{d}$, where $d < 2^{n/3}$.

Decisional second-preimage resistance (DSPR). In [4], Bernstein and Hülsing introduce DSPR, which defines the advantage in deciding, given a random input x, whether x has a second preimage.

An adversary could potentially use this definition to determine in advance whether or not it is worth attacking the SPR (or eTCR) of a hash function. If the DSPR advantage is non-negligible, then the adversary can choose a wallet target, and determine in advance whether or not there is a second-preimage. For example, if there is not a second-preimage associated with a target wallet address, then the adversary can select another target address as opposed to spending unnecessary computational resources trying to find a non-existent value. The paper, however, proves that DSPR is at least as hard to break as preimage resistance (PRE) or second preimage resistance (SPR) for uniform random hash functions from $\{0,1\}^n$ to $\{0,1\}^n$. This results in an attack cost of 2^n in the classical setting, and $2^{n/2}$ in the post-quantum setting.

The authors considered ways to attack DSPR for real hash functions, and concluded that there is no obvious way for a fast attack to achieve any advantage. Consequently, $\mathcal{A}$ cannot take advantage of the DSPR notion to gain any non-negligible advantage in creating forged proof(s)-of-ownership.

[4] We highlight the work of Banegas and Bernstein [3] that studies the existing overhead beyond the quantum queries and shows that even in a post-quantum setting, the collision-finding algorithms costs at least $2^{n/2}$, even if it requires a smaller number of queries.

B Simplified Description of the Construction

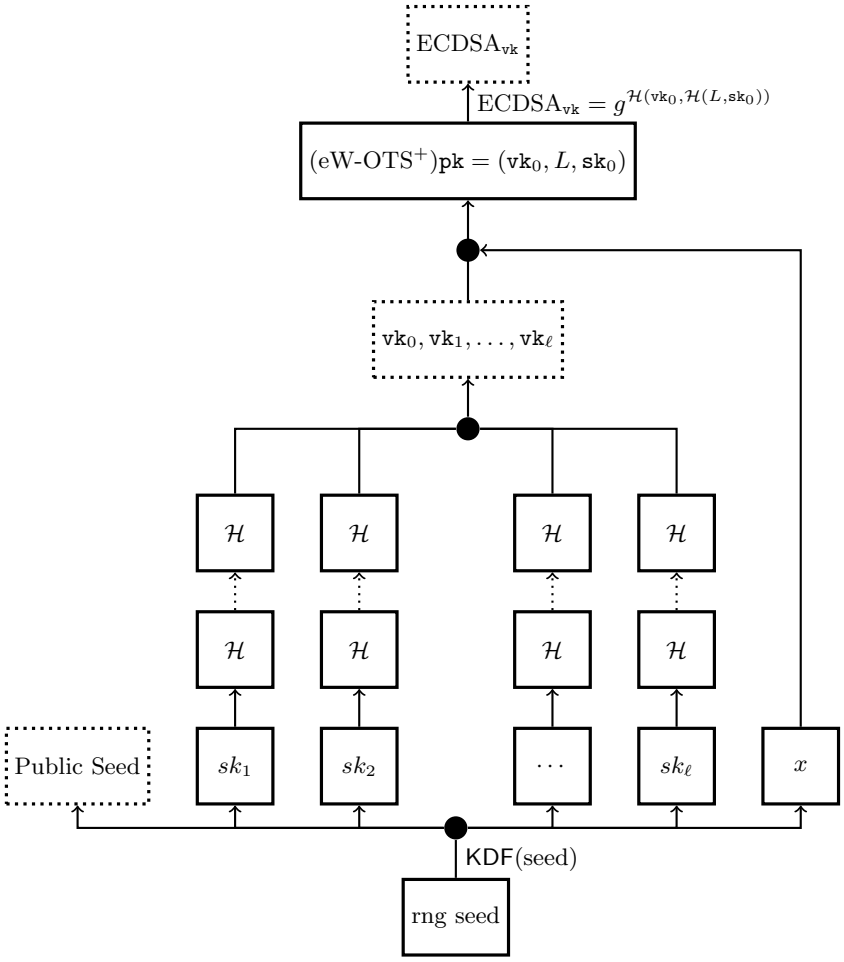

Fig. 1. Hidden key construction for eW-OTS$^+$. The dotted boxes are the potentially public values, while the normal boxes are the secret values. The diagram shows the commonly know as "ladders",*i.e.* the sequence of hash function executions up to the verification values, and "rng seed" generating randomness for the private hash key x.