Tweakable $\mathcal{S}_{\text{leeve}}$: A Novel $\mathcal{S}_{\text{leeve}}$ Construction Based on Tweakable Hash Functions

David Chaum, Mario Larangeira, and Mario Yaksetig

Abstract Recently, Chaum et al. (ACNS'21) introduced $\mathcal{S}_{\text{leeve}}$, which describes an extra security layer for signature schemes, *i.e.*, ECDSA. This distinctive feature is a new key generation mechanism, allowing users to generate a "back up key" securely nested inside the secret key of a signature scheme. Using this novel construction, the "back up key", which is secret, can be used to generate a "proof of ownership", *i.e.*, only the rightful owner of this secret key can generate such a proof. This design offers a quantum secure *fallback*, *i.e.*, a brand new quantum resistant signature, ready to be used, nested in the ECDSA secret key. In this work, we rely on the original $\mathcal{S}_{\text{leeve}}$ definition to generalize the construction to a modular design based on *Tweakable Hash Functions*, thus yielding a cleaner design of the primitive. Furthermore, we provide a thorough security analysis taking into account the security of the ECDSA signature scheme, which is lacking in the original work. Finally, we provide an analysis based on formal methods using *Verifpal* assuring the security guarantees our construction provides.

Keywords Provable security · Digital wallet · Hash-based signatures

This work was supported by JSPS KAKENHI Grant Number JP21K11882.

D. Chaum
xx Network, Grand Cayman, Cayman Islands
e-mail: david@xx.network

M. Larangeira (✉)
Tokyo Institute of Technology, Tokyo, Japan
e-mail: mario@c.titech.ac.jp; mario.larangeira@iohk.io

IOHK, Battery Road, Singapore

M. Yaksetig
University of Porto, Porto, Portugal
e-mail: mario.yaksetig@fe.up.pt

P. Pardalos et al. (eds.), *Mathematical Research for Blockchain Economy*, Lecture Notes in Operations Research, https://doi.org/10.1007/978-3-031-18679-0_10

1 Introduction

The ECDSA signature scheme is widely used; however it achieved new levels of exposure after it found new applications in electronic wallets for cryptocurrencies such as Bitcoin [22], Ethereum [25] and Cardano/Ouroboros [2, 11, 19]. This intensive exposure drove the research community to channel its efforts to propose new attacks on the signature scheme/wallets [1, 24].

Recently, Chaum et al. [8] proposed $\mathcal{S}_{\text{leeve}}$, a signature based new cryptographic primitive in order to mitigate damages during massive leaks of wallet private information. In a nutshell, the goal of [8] is to allow the rightful user to prove its (correct) ownership in the face of the situation that its secret key becomes public. In such a situation, proving the knowledge of the correct secret key, via zero knowledge protocols, for example, is of no use as, potentially, anyone could generate such proof. Furthermore, $\mathcal{S}_{\text{leeve}}$ leaves at the disposal of the user a second signature scheme. More concretely, $\mathcal{S}_{\text{leeve}}$ leverages a regular ECDSA scheme to have a nested "back up key" to generate the proof of ownership, or even be fully discarded for a (post quantum) signature scheme; a hash based signature scheme. In theory, wallets implementing $\mathcal{S}_{\text{leeve}}$ can be easily switched to be quantum resistant, since in addition to ECDSA, they would also contain a post quantum signature as the fallback feature.

$\mathcal{S}_{leeve}$ Design Limitations. The novel approach in [8], in particular the construction the original authors introduced, deals with the aggregation of a W-OTS$^+$ public key into a single value to be used in the ECDSA as the secret key. Their solution was to adapt the L-Tree data structure [10] in order to execute the integration. This approach works for their purpose; however its design seems fairly limited and ad hoc, *i.e.* left and right branches of the L-Tree requires pair of values which needs to be added to the key pairs. More modern approaches exist and seem more suitable to such integration between ECDSA and hash based signatures, such as relying on Tweakable Hash Functions [3]. The security analysis of [8] introduces two new properties: *proof of ownership* and *fallback*; however it does not detail the impacts in the signature scheme. Namely, the introduction of a back up key nested into the ECDSA signature scheme is not shown to have any side effects on its security. In fact, the ECDSA security in [8] is not assured by a computational problem. Transactions generated by wallets rely on signatures, therefore this state of uncertainty is not ideal for the security of regular users. Moreover, a closer look on the ECDSA security literature shows that it is more involved than a naive reader would previously expect [6, 7, 13, 14].

History of the ECDSA Security. Brown [6, 7] has shown that the ECDSA is strong unforgeable (when the adversary receives only one signature per message) in a chosen message attack considering a proof technique based on the Random Oracle Model (ROM) and Generic Group Model (GGM). Fersch et al. [13] pointed out, that indeed ECDSA is strong unforgeable in these models; however in the real world, when no assumption is assumed in the group (thus not in the GGM), that is not the case. The

reason for the discrepancy is the modelling of the group in the conversion function of the scheme, *i.e.*, mapping the group elements to the field $\mathbb{Z}_q$ for a large prime q.

The works [13, 14] sidestep the briefly mentioned limitations of the proof technique by dropping the GGM, while still relying on the ROM. Both works show that ECDSA is indeed secure; however when the adversary is given only one signature per message employing a proof method relying on a "Generic ElGamal Framework", which subsumes several variants of DSA, including the ECDSA. Perhaps, surprisingly, the proven security is based in the Semi Logarithm Problem (SLP) [6] instead of the more standard Discrete Logarithm Problem (DLP) as one would expect. Attempts to show the security of $\mathcal{S}_{\text{leeve}}$ must take into account these developments, and that is what we do within this work.

Our Contributions. Succinctly, the main contributions of this work are

- Section 3 (and Appendix A) introduces a clean and modular construction to quantum-secure fallback and proof of ownership of ECDSA under the $\mathcal{S}_{\text{leeve}}$ definition and based on Tweakable Hash Functions;
- Section 4 presents a proof of security with respect to the original $\mathcal{S}_{\text{leeve}}$ definition, for proof of ownership and fallback, regarding the signatures generated by $\mathcal{S}_{\text{leeve}}$ with respect to the unforgeability security of the ECDSA scheme, and based on the computational problem SLP (dependence of a computational problem is crucial in provable security standard);
- Section 5 introduces benchmarks of an open-source, fully audited, and deployed implementation currently in use on existing blockchain platform;
- Section 6 shows the security of the construction using Verifpal, a formal methods analysis tool, and provides the first ever analysis of a hash-based signature scheme using formal methods analysis tools, highlighting the existing challenges in this type of modelling.

The most remarkable differences between the work from [8] and ours is (1) the use of Tweakable Hash functions, which [8] does not use. Therefore, as in [8], our construction works with basic wallet scripts, *e.g.*, multisig. Their construction relies heavily on L-Tree as used in [16], therefore our construction takes the more modern approach, (2) the security guarantees and analysis we introduce.

2 Background

As preparation, let n be the security parameter, and PPT denote *probabilistic polynomial-time*. We rely on the standard notion of negligible function. That is $\mathsf{negl}(n)$ is said to be negligible if and only if for all $c \in \mathbb{N}$, there is a n_0 such that for all $n \geq n_0$, $\mathsf{negl}(n) < n^{-c}$.

Now we can review the $\mathcal{S}_{\text{leeve}}$ and Tweakable Hash Function definitions.

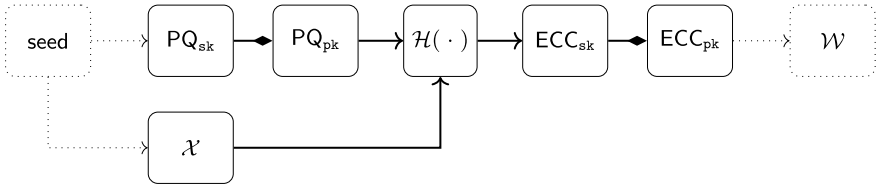

Fig. 1 Overview of the $\mathcal{S}_{\text{leeve}}$ construction, where the user generates a post-quantum (PQ) key pair $(\mathsf{PQ}_{\mathtt{sk}}, \mathsf{PQ}_{\mathtt{pk}})$ along with a hash key $\mathcal{X}$ from the local randomness seed, which is used as an input when hashing the fallback public key. The result of the hashing operation is used as an elliptic curve secret key, $\mathsf{ECC}_{\mathtt{sk}}$, then used as elliptic curve cryptography (ECC) trapdoor and obtain the elliptic curve public key $\mathsf{ECC}_{\mathtt{pk}}$. Diamond arrows represent a trapdoor, and normal arrows showcase the values acting either as input or output

Overview of the $\mathcal{S}_{leeve}$ construction. The principle behind the construction is that users first generate a public-private key pair that is quantum resistant along with a secret key value, hash the quantum-resistant public key along with the secret key value, and use this output as a secret key to be used as an elliptic curve secret key. Upon obtaining the elliptic curve secret key, users can trivially generate the corresponding public key. To finalize, users may have to perform additional steps to obtain a wallet address associated with an elliptic curve public key. For example, on blockchain platforms like Bitcoin [22], Ethereum [25] and Cardano/Ouroboros [2, 11, 19], users hash their ECDSA public key to obtain their wallet address. The construction is designed to be modular such that users can easily use the best suitable cryptographic assumptions for each of the modules. Figure 1 illustrates an overview diagram of the construction.

$\mathcal{S}_{leeve}$ Desired Properties. The design [8] is due to the need to integrate a quantum-secure fallback into the ECDSA scheme. Namely, the question it addresses is: *If an adversary breaks the Elliptic Curve based DLP (ECDLP), compromising the security of a cryptocurrency, can users redeem (or rollover) their assets in a safe manner without the risk of theft from this adversary?*

Before addressing the required properties for our design, we highlight similar research in this area, such as [18] and [17], that provide a different alternative solution to this question. These approaches rely on a user Alice publishing one hash commit of both the elliptic curve public key and the fallback public key. At a later point in time, Alice signs a reveal transaction using the fallback secret key which reveals both the elliptic curve public key and the fallback key. This transaction then proves that Alice is the true owner of a specific wallet address.

The scheme in [8] requires some different properties, which are now enumerated. First, and intuitively, users should have the ability to integrate a (quantum-secure) fallback for traditional cryptocurrency wallets, which typically rely on the ECDSA scheme. Ideally, this solution should not incur in any type of additional communication costs and should not assume an interactive protocol if it is not strictly necessary. This segregation and lack of interactivity with any other parties is particularly rele-

vant as it allows a user to, upon completion of the key generation, quickly and simply store the fallback key in a cold wallet without requiring any signature until a quantum threat appears. Second, the users should have the ability to leak the fallback public key in a manner that does not expose the ECDSA secret key. For example, Alice should be able to disclose that she owns a wallet address $\mathcal{W}_A$, and a fallback public key to inform all in the system that she may need to provide a signature that can be verified under such fallback key. The reveal of the fallback public key should not translate to a compromise of the ECDSA secret key as then any entity in the system could produce signatures and attempt to perform transactions on behalf of Alice.

Third, the design should be modular, easy to use, and compatible with currently used cryptocurrency wallets. Therefore, the design should have the possibility of supporting any elliptic curve based wallet, any post-quantum secure fallback, and should support the use of mnemonics and other features that improve usability for the end user. Ideally, the security proofs for each of the components should be modular such that changing the used schemes in the different parts of the design does not affect other parts of the construction.

Lastly, one of the main properties of this construction is *fork voiding* in a blockchain system. Upon redeeming the digital assets into the fallback public key, users should fully expose the ECDSA secret key such that the value of the assets stored on the original chain naturally converges towards zero, thus it incentives users to abandon the initial chain towards the new and safer fork.

$\mathcal{S}_{leeve}$ and its Security Properties. The $\mathcal{S}_{\text{leeve}}$ primitive is composed by the tuple (Gen_π, Sign, Verify, Proof, Verify-Proof). The generation algorithm outputs the pairs of keys, vk and sk, and the backup key bk. The first pair is the regular verification key, used for verifying a signature, and the secret-key used for issuing a signature. While the last key is used to issue the *Proof of Ownership* π, with respect to vk as follows

Definition 1 *($\mathcal{S}_{\text{leeve}}$ [8])* A fallback scheme $\mathcal{S}_{\text{leeve}} = (\mathsf{Gen}_\pi, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Verify\text{-}Proof})$ is a set of PPT algorithms:

- Gen_π (1^n) on input of a security parameter n outputs a private signing key sk, a public verification key vk and the back up key bk;
- Sign (sk, m) outputs a signature σ under sk for a message m using the designated main signature scheme, in our example this is an ECDSA signature;
- Verify (vk, σ, m) outputs 1 iff σ is a valid signature on m under vk;
- Proof(bk, c) on input of the backup information bk and the challenge c, it outputs the ownership proof π. In our example, this is a W-OTS$^+$ signature on the challenge c using the fallback key bk;
- Verify-Proof(vk, sk, π, c) is a deterministic algorithm that on input of a public-key vk, secret-key sk, an ownership proof π and a challenge c, it outputs either 0, for an invalid proof, or 1 for a valid one.

The two main security properties of $\mathcal{S}_{\text{leeve}}$ are (1) the capability of issuing a proof to confirm the ownership of the secret key, even in the face of a massive leakage, when

the secret key becomes public, and (2) the capability to smoothly switch to another signature scheme, namely a quantum resistant one. Briefly, we formally review both properties.

Definition 2 *(Proof of Ownership* [8]*)* For any PPT algorithm $\mathcal{A}$ and security parameter n, it holds

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

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

Definition 3 *(Fallback* [8]*)* 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 $\mathsf{Gen}_\pi(1^n) \rightarrow (\mathtt{vk}, \mathtt{sk}, \mathtt{bk})$, 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 as fully independent signature scheme.

Tweakable Hash Functions. Introduced to allow better abstraction of hash-based signature scheme. By decoupling the computations of hash chains, hash trees, and nodes, protocol designers can separate the analysis of the high-level construction from exactly how the computation is done. Therefore abstracting the computation away in hash-based schemes only requires analyzing the hashing construction. The standard definition is as follows.

Definition 4 *(Tweakable Hash Function* [3]*)* Let $\mathcal{P}$ the public parameters space, $\mathcal{T}$ the tweak space, and $n, \alpha \in \mathbb{N}$. A Tweakable Hash Function is an efficient function mapping an α-bit message M to an n-bit hash value MD using a function key called public parameter $P \in \mathcal{P}$ and a tweak $T \in \mathcal{T}$. Therefore, we have $\mathsf{Th} : \mathcal{P} \times \mathcal{T} \times \{0,1\}^\alpha \rightarrow \{0,1\}^n, \quad \mathsf{MD} \leftarrow \mathsf{Th}(P, T, M)$.

A tweakable hash function takes public parameters P and context information in the form of a tweak T in addition to the message. The public parameters might be thought as a function key or index. The tweak might be interpreted as a nonce. We use the term public parameter for the function key to emphasize it is intended to be public. Thus we explicitly assume an extra property for Th.

Definition 5 *(Indistinguishability)* For the security parameter n, and the tweakable hash function Th, we say that Th has the Computational Indistinguishability from Uniformly Random Distribution Property, if for every PPT distinguisher $\mathcal{D}$, and arbitrary choices of the parameters P, T and M, the following holds $|Pr[x \leftarrow \mathsf{Th}(P, T, M), \mathcal{D}(x) = 1] - Pr[x \leftarrow \mathcal{U}, \mathcal{D}(x) = 1]| \leq \mathsf{negl}(n)$, for the uniform distribution $\mathcal{U}$.

3 The Tweakable $\mathcal{S}_{\text{leeve}}$

We now describe our $\mathcal{S}_{\text{leeve}}$ construction, with W-OTS$^+$ as the fallback, and a tweakable hash function for the public key integration, *i.e.* Tweakable $\mathcal{S}_{\text{leeve}}$.

Definition 6 *(Family of Functions)* Given the security and the Winternitz parameters, respectively, $n \in \mathbb{N}$ and $w \in \mathbb{N}, w > 1$, let a family of functions $\mathcal{H}_n$ be $\{h_k : \{0,1\}^n \rightarrow \{0,1\}^n | k \in \mathcal{K}_n\}$ with key space $\mathcal{K}_n$.

Definition 7 *(Chaining Function)* Given a family of functions $\mathcal{H}_n$, $x \in \{0,1\}^n$, an iteration counter $i \in \mathbb{N}$, a key $k \in \mathcal{K}_n$, for $jn-$bit strings $\mathbf{r} = (r_1, \ldots, r_j) \in \{0,1\}^{n \times j}$ with $j \geq i$, then we have the chaining function as follows

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

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 $(r_a, \ldots, r_b)$, and for our construction to be presented next, we rely on a Key-Derivation Function KDF which follows the recently announced set of recommendations [9].

Protocol Description. $\mathcal{S}_{\text{leeve}}$ is 5-tuple set of PPT algorithms (Gen_π, Sign, Verify, Proof, Verify-Proof). We describe the generic version of (Gen_π, Sign, Verify) in Table 1, based on the formalism of [14] which is convenient for our security analysis in Sect. 4. The algorithms Proof and Verify-Proof are given in Tables 2 and 3, respectively.

Table 1 Gen$\mathcal{S}_{\text{leeve}}$ is based on the GenElGamal Framework [6, 14] and it relies on the Th which is indistinguishable from the uniform distribution as per Definition 5, and Proof and Verify-Proof are the concrete algorithms

$\mathsf{Gen}_\pi^k(1^n)$	$\mathsf{Sign}^H(\mathsf{m}, \mathtt{sk})$	$\mathsf{Verify}^H(\mathsf{m}, \mathtt{vk}, \sigma)$
Pick a random public seed P	$h \leftarrow H(\mathsf{m})$	Parse: $(s, t) \xleftarrow{p} \sigma$
Pick $(\ell + w - 1)$ n-bit strings r_i	$r \xleftarrow{\$} \mathbb{Z}_p$; $R \leftarrow g^r$	$h \leftarrow H(\mathsf{m})$
Set $\mathtt{bk}_i \leftarrow r_i$, for $1 \leq i \leq \ell$	If $R = 1$:Return $\perp$	If $(s, h, t) \notin \mathbb{D}$: Return 0
Set $\mathbf{r} = (r_{\ell+1}, \ldots, r_{\ell+w-1})$	$t \leftarrow f(R)$	$\hat{R} \leftarrow \mathsf{V}_{g,x}^E(s, h, t)$
Set $\mathtt{vk}_i = c_k^{w-1}(\mathtt{bk}_i, \mathbf{r})$, $1 \leq i \leq \ell$	$s \leftarrow \mathsf{S}_{\mathtt{sk}}^E(h, t, r)$	If $\hat{R} = 1$: Return 0
Pick a random hash key $\mathcal{X}$	If $(s, h, t) \notin \mathbb{D}$: Return $\perp$	$\hat{t} \leftarrow f(\hat{R})$
Pick a random tweak T	Return $\sigma = (s, t)$	If $t \neq \hat{t}$: Return 0
W-OTS$^+_{\mathsf{pk}} = \mathsf{Th}(P, T, \mathtt{vk}_1, \ldots, \mathtt{vk}_\ell)$		Return 1
$\mathtt{sk} \leftarrow ((\mathbf{r}, k), \mathsf{Th}(P, \mathcal{X}, \text{W-OTS}^+_{\mathsf{pk}}))$		
$\mathtt{vk} \leftarrow g^{\mathtt{sk}_1}$		
Return $(\mathtt{vk}, \mathtt{sk}, \mathtt{bk})$		

Table 2 Proof algorithm, which is the eW-OTS$^+$ Signature Scheme from [8]. The changes introduced by our construction are necessary in order to be used in combination with ECDSA signatures

Proof$(c, \mathtt{bk})$
Parse $\mathtt{bk} \rightarrow (\mathtt{bk}_0, \mathtt{bk}_1, \ldots, \mathtt{bk}_\ell)$
Parse $\mathtt{bk}_0 \rightarrow (\mathcal{T}, \mathcal{P}, \mathcal{X})$
Set $\pi_0 = \mathtt{bk}_0$
Compute $c \rightarrow (c_1, \ldots, c_{\ell_1})$,
for $c_i \in \{0, \ldots, w-1\}$
Compute checksum $C = \sum_{i=1}^{\ell_1}(w - 1 - c_i)$,
w-base representation $(C_1, \ldots, C_{\ell_2})$,
for $C_i \in \{0, \ldots, w-1\}$
Parse $B = c \Vert C$ as $(b_1, \ldots, b_{\ell_1+\ell_2})$
Set $\pi_i = c_k^{b_i}(\mathtt{bk}_i, \mathbf{r})$, for $1 \leq i \leq \ell_1 + \ell_2$
Return $\pi = (\pi_0, \pi_1, \ldots, \pi_{\ell_1+\ell_2})$

Table 3 The verification of the proof π adapts the verification procedure for eW-OTS$^+$ by adding an extra check on the ECDSA verification key $\mathtt{vk}$

Verify-Proof$(\mathtt{vk}, \mathtt{sk}, c, \pi)$
Parse $\mathtt{sk} \rightarrow (\mathtt{sk}_0, \mathtt{sk}_1)$
Parse $\mathtt{sk}_0 \rightarrow (\mathbf{r}, k)$
Parse $\pi \rightarrow (\pi_0, \pi_1, \ldots, \pi_{\ell_1+\ell_2}), \pi_0 \rightarrow (T, P, \mathcal{X})$
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 $B = c \Vert C$ 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 $1 \leq i \leq \ell_1 + \ell_2$
Set W-OTS$^+_{\mathsf{pk}} = \mathsf{Th}(P \Vert T \Vert \mathtt{vk}_1, \ldots, \mathtt{vk}_{\ell_1+\ell_2})$
Return 1, if the following equations hold
$\mathtt{sk}_1 = \mathsf{Th}(P \Vert \mathcal{X} \Vert \text{W-OTS}^+_{\mathsf{pk}})$
$\mathtt{vk} = g^{\mathtt{sk}_1}$

3.1 The Generic Sleeve: Gen$\mathcal{S}_{leeve}$

In order to formulate the definition for Gen$\mathcal{S}_{\text{leeve}}$, we review more basic definitions to cast it in more generic terms and bases its security on a computational problem, *i.e.* SLP.

Our security analysis relies on the work of [14] which is the state of the art in the understanding of the security of the ECDSA. Their proof bases the analysis in the Semi Logarithm Problem (SLP) with respect to the *Conversion Function* f. Such a

function was introduced in the *GenElGamal Framework* which subsumes ECDSA and other ElGamal based schemes. The proposed framework is parameterized by a *Defining Equation E* for a set $\mathbb{D}$ which gives the distribution of the values to be used in the signature generation, consequently, generating the different "flavors" of the ElGamal/DSA schemes.[1] For a better readability and completeness of this work, we now review these definitions.

Conversion Function. A component of the GenElGamal Framework is the conversion function f. More concretely, the conversation function maps the group members from $\mathbb{G}$ to $\mathbb{Z}_q$. The SLP is with respect to f and, in its simplest form, can be stated as given a pair of group members g and $X = g^x$, it is required to output s and t such that $t = f((g \cdot X^t)^{\frac{1}{s}})$. Its more general form is given by the next definition.

Definition 8 *(SLP* [6]*)* Let $(\mathbb{G}, g, q)$ be a prime-order group and let $f : \mathbb{G}^* \to \mathbb{Z}_q$ and $\rho_0, \rho_1 : \mathbb{Z}_q^2 \to \mathbb{Z}_q$ be functions. We say that an algorithm $\mathcal{I}(\tau, \epsilon)$-breaks the SLP in $\mathbb{G}$ with respect to f, ρ_0 and ρ_1 if it runs in time at most τ and achieves probability $\epsilon = \Pr[X \leftarrow \mathbb{G}; (u, v) \leftarrow \mathcal{I}(g, X) : v = f(g^{\rho_0(u,v)} \cdot X^{\rho_1(u,v)})]$.

The Defining Equation. The sign and verification procedures for the ECDSA and $\mathcal{S}_{\text{leeve}}$ variants can be defined in a modular and general fashion. The technique to make the variants is crucially dependent on the sampling values; Each variant has a different distribution. The Defining Equation rules the distribution, thus we review the definition.

Definition 9 *(Defining Equation)* Let $\mathbb{D} \subset \mathbb{Z}_q^3$ be a set. An equation $E = E(s, h, t, r, x)$ over $\mathbb{D} \times (\mathbb{Z}_q^*)^2$ is said to be defining (a signature scheme) if E has the form $E(s, h, t, r, x) = C_0(s, h, t) + r \cdot C_r(s, h, t) + x \cdot C_x(s, h, t)$, where C_0 and C_x are functions $\mathbb{D} \to \mathbb{Z}_q$, and C_r is a function $\mathbb{D} \to \mathbb{Z}_q^*$. With other words, E is defining if it is affine linear in x and r, and E can always be solved for r.

The concrete example of Defining Equation is $E(s, h, t, r, x) = h - rs + tx$ for the Defining Set $\mathbb{D} = \mathbb{Z}_q^* \times \mathbb{Z}_q \times \mathbb{Z}_q^*$ as given by [14].

Definition 10 *(Sign and Verification Function)* Let E be a defining equation. Then we define the signing function $\mathsf{S}^E(h, t, r, \mathtt{sk}) = \mathsf{S}^E_{\mathtt{sk}}(h, t, r)$ as follows: if there exists a unique s such that $E(s, h, t, r, \mathtt{sk})$ is satisfied, S^E returns s; otherwise, the function returns $\perp$. Further, we define the verification function $\mathsf{V}^E(g, s, h, t, \mathtt{sk}) = \mathsf{V}^E_{g,\mathtt{sk}}(s, h, t)$ with respect to a prime-order group $(\mathbb{G}, g, q)$ as follows: if r is the (unique) solution of $E(s, h, t, r, \mathtt{sk})$ then V^E returns g^r.

As remarked by [14], the affine linear form of E makes possible to efficiently evaluate V^E given just the tuple $(s, h, t, g^{\mathtt{sk}})$, *i.e.*, without knowing $\mathtt{sk}$ explicitly. Now we are ready to define our generic construction.

Definition 11 *(Gen$\mathcal{S}_{\text{leeve}}$ Framework)* Given a hash function H, and the S and V, respectively the Sign and Verification Functions, the Conversion Function f,

[1] For a complete list of the supported schemes, we refer the reader to the full list in [14].

the Defining Equation E and Set $\mathbb{D}$, and the Generic $\mathcal{S}_{\text{leeve}}$ scheme is the tuple $(\mathsf{Gen}_\pi, \mathsf{Sign}^H, \mathsf{Verify}^H, \mathsf{Proof}, \mathsf{Verify\text{-}Proof})$, such that k is the parameter of the family of function, the three first algorithms are given as follows.

4 Security Analysis

This sections introduces the security analysis of the Tweakable $\mathcal{S}_{\text{leeve}}$ in three complementary ways. The next sections cover, respectively, the following:

1. Security with respect of generic attacks and fallback security, as these were introduced in [8];
2. Lemma 1 proposal that shows Tweakable $\mathcal{S}_{\text{leeve}}$ has equivalent security as the ECDSA in terms of unforgeability of signatures, *i.e.* EUF-CMA;
3. The security of the Gen$\mathcal{S}_{\text{leeve}}$ (introduced in Sect. 3.1), in the same fashion of [14], *i.e.* Generic ECDSA, and show Gen$\mathcal{S}_{\text{leeve}}$ to be secure with respect to the Semi Logarithm Problem (SLP).

4.1 Generic Attack Security and Unforgeability of Fallback Scheme

The authors of $\mathcal{S}_{\text{leeve}}$ describe in [8] the security level of the construction against generic attacks targeted at the underlying hash function and prove the unforgeability of the fallback scheme. Additionally, they prove that, for an appropriate choice of parameters, the best attack against the fallback scheme (i.e., eW-OTS$^+$ the W-OTS$^+$ variant introduced in [8]) is the same attack against the original W-OTS$^+$. We use these results as a reference as we consider the same fallback scheme and note that, by replacing the assumptions of the underlying hash function with a tweakable hash function, the security results remain well-defined.

4.2 Tweakable $\mathcal{S}_{leeve}$ is at Least as Secure as an ECDSA One

The security of the ECDSA scheme is given by [14]. However $\mathcal{S}_{\text{leeve}}$ introduces a new key generation method, which is not considered in the security proof of [8]. Concretely, the generation method relies on the tweakable hash function in order to generate the ECDSA secret key `sk`; however it is not clear if such modification on the ECDSA scheme introduces weaknesses. We address this gap now.

The Unforgeability of $\mathcal{S}_{leeve}$. In addition to the listed properties of Sect. 2, $\mathcal{S}_{\text{leeve}}$ is also suitable to similar security definitions as the ones for signature schemes. The

Table 4 Unforgeability for $\mathcal{S}_{\text{leeve}}$, *i.e.* three keys are generated. The above game is One-Message Existential Unforgeability with Chosen Message Attack game, *i.e.* (EUF-CMA1). For the general form, *i.e.* the standard (EUF-CMA), the Sign Procedure does not abort when the message is in the list $\mathcal{L}$. For the key only (UF-KOA) variant of the game, the adversary does not access the Sign Procedure

Procedure Init(n)	Procedure Sign(m)
$\mathcal{L} \leftarrow \emptyset$	If $m \in \mathcal{L}$: Abort
$(\mathtt{vk}, \mathtt{sk}, \mathtt{bk}) \leftarrow \mathsf{Gen}_\pi(1^n)$	$\sigma \leftarrow \mathsf{Sign}(\mathtt{sk}, m)$
Return $\mathtt{vk}$	If $\sigma = \perp$: Return $\perp$
	$\mathcal{L} \leftarrow \mathcal{L} \cup \{m\}$
Procedure Fin(m^*, σ^*)	Return σ
If $\mathsf{m}^* \in \mathcal{L}$: Abort	
If $\mathsf{Verify}(\mathtt{vk}, \mathsf{m}^*, \sigma^*) = 0$: Abort	
Return 1	

difference is the generation of the keys, which $\mathcal{S}_{\text{leeve}}$ introduces an extra one, the back up key. Table 4 defines the security notion, derived from standard EUF-CMA security for signature schemes. The difference is only the extra back up key.

The goal of the next lemma is to show that the EUF-CMA security of $\mathcal{S}_{\text{leeve}}$, constructed with a suitable tweakable hash function, and ECDSA, instantiated with uniformly random sampling for the secret key, are equivalent.

Lemma 1 *Assume ECDSA is EUF-CMA secure and the generation algorithm* Gen_π *from Table 1 is constructed with a tweakable hash function* Th *indistinguishable from the uniform distribution as per Definition 5 for the security parameter n. Then* $\mathcal{S}_{leeve}$ *is EUF-CMA as given by the security game of Table 4.*

Proof *(sketch)* Assume the existence of a $\mathcal{S}_{\text{leeve}}$ forger $\mathcal{F}$ which wins the game from Table 4 by outputting a forgery (m^*, σ^*) with non-negligible probability. Then we construct a PPT distinguisher algorithm $\mathcal{D}$ which breaks the indistinguishability property of Th with high probability. We construct $\mathcal{D}$ as follows:

- $\mathcal{D}$ performs the security game given by Definition 5, and receives as input the string x;
- $\mathcal{D}$ modifies the generation algorithm Gen_π from Table 1, by using the received string x to generate the public, key. In the modified game the public key is $\mathtt{vk}' = g^{\mathtt{sk}'}$ for $\mathtt{sk}' \leftarrow x$;
- $\mathcal{D}$ simulates the EUF-CMA security game of Table 4 to $\mathcal{F}$ using $(\mathtt{vk}', sk')$;
- With high probability $\mathcal{F}$ outputs (m^*, σ^*), then $\mathcal{D}$ uses the verification algorithm Verify to perform the following and stop:
  - If $\mathsf{Verify}(m^*, \sigma^*) = 1$, then output 1
  - Else, output 0;

Now we estimate the success probability of $\mathcal{F}$ in the EUF-CMA game of Table 4, by considering three points:

- From the indistinguishability property of Th, we know $|Pr[x \leftarrow \mathsf{Th}(P, T, M), \mathcal{D}(x) = 1] - Pr[x \leftarrow \mathcal{U}, \mathcal{D}(x) = 1]|$ is negligible for arbitrary choices of P, T and M as given by Definition 5 and initial hypothesis;
- Following from the $\mathsf{EUF\text{-}CMA}$ security of ECDSA, we have that $\Pr[x \leftarrow \mathcal{U}, \mathsf{Verify}(m^*, \sigma^*) = 1]$ is negligible for the uniform random distribution $\mathcal{U}$;
- Finally, note that $\Pr[x \leftarrow \mathsf{Th}(P, T, v), \mathcal{D}(x) = 1]$ and $\Pr[x \leftarrow \mathsf{Th}(P, T, v), \mathsf{Verify}(m^*, \sigma^*) = 1]$ are equal by design of $\mathcal{D}$ and success probability of $\mathcal{F}$.

Therefore, $|Pr[x \leftarrow \mathsf{Th}(P, T, M), \mathcal{D}(x) = 1] - Pr[x \leftarrow \mathcal{U}, \mathcal{D}(x) = 1]| \leq \mathsf{negl}(n)$, and $|\Pr[x \leftarrow \mathsf{Th}(P, T, M), \mathsf{Verify}(m^*, \sigma^*) = 1] - \mathsf{negl}(n)| \leq \mathsf{negl}(n)$. Hence $\Pr[x \leftarrow \mathsf{Th}(P, T, v), \mathsf{Verify}(m^*, \sigma^*) = 1]$ must be negligible and $\mathcal{S}_{\text{leeve}}$ is also $\mathsf{EUF\text{-}CMA}$, thereby giving the lemma. □

The earlier lemma only relates the security of $\mathcal{S}_{\text{leeve}}$ and ECDSA. In order to thoroughly prove the hardness of breaking $\mathcal{S}_{\text{leeve}}$ it is convenient to consider a computational problem. That is what we do next.

4.3 The Security of Gen$\mathcal{S}_{\text{leeve}}$

From now we take the approach of [14] in order to build a full proof of the unforgeability of the generic $\mathcal{S}_{\text{leeve}}$ variant based on the assumed hard computational problem. Namely, show the security of Gen$\mathcal{S}_{\text{leeve}}$ with respect to SLP. What we do now is to review the main definitions from [14] combined with the ones introduced in Sect. 3.1.

Definition 12 *(h-decomposable)* Let $E = E(s, h, t, r, x)$ be a defining equation with corresponding set $\mathbb{D}$. We say that E is h-decomposable (with respect to $\mathbb{D}$) if there exist functions $\nu_0, \nu_1 : \mathbb{Z}_q \to \mathbb{Z}_q$ and $\rho_0, \rho_1 : \mathbb{Z}_q^2 \to \mathbb{Z}_q$ such that $\nu_0, \nu_1 \neq 0$ if $h \neq 0$ and $r = \nu_0(h) \cdot \rho_0(s, t) + x \cdot \nu_1(h) \cdot \rho_1(s, t)$ for all $(s, h, t) \in \mathbb{D}$ and $r, x \in \mathbb{Z}_q^*$ satisfying $E(s, h, t, r, x)$.

For completeness, in the next definition we consider the standard notion for δ statistical distance. That is, for any two ensembles $\{X(x, k)\}_{x \in \{0,1\}^*, k \in \mathbb{N}}$ and $\{Y(x, k)\}_{x \in \{0,1\}^*, k \in \mathbb{N}}$, for index k and input x, the value $|\Pr[X(x, k) = 1] - \Pr[Y(x, k) = 1]|$ is at most δ.

Definition 13 *(δ-Simulatability)* Let $(E, \mathbb{G}, H, f, \mathbb{D})$ be an instantiation of Gen$\mathcal{S}_{\text{leeve}}$ as in Definition 11. It is said that the instantiation is δ-simulatable if there exists a function $\mathsf{Sim}^E : \mathbb{Z}_q^3 \times \mathbb{Z}_q^2 \cup \{\perp\}$ that is computable in about the same time as S^E such that for all $\mathtt{sk} \in \mathbb{Z}_q^*$ the statistical distance between the outputs of the two protocols depicted by Table 5 is at most δ.

The generic security is derived from the work on [14]. Namely, the next two theorems which are defined according to the number of random oracle and signature queries, respectively $\mathcal{Q}_H$ and $\mathcal{Q}_s$ and the big-O notation O. For completeness we present them altered to Gen$\mathcal{S}_{\text{leeve}}$. However we refer the reader to the full work for the proofs of the theorems, which are the same for Gen$\mathcal{S}_{\text{leeve}}$.

Table 5 $\mathsf{P}_{\mathsf{Sim}}$ shows that, given a procedure Sim, it is possible to generate a tuple (s, h, t) statistically close without knowing the secret key $\mathtt{sk}$

$\mathsf{P}_{real}(\mathtt{sk}, g)$	$\mathsf{P}_{\mathsf{Sim}}(\mathtt{vk}, g)$
$r \xleftarrow{\$} \mathbb{Z}_p$	$a, b \xleftarrow{\$} \mathbb{Z}_p$
$R \leftarrow g^r$	$R \leftarrow \mathtt{vk}^a g^b$
If $R = 1$: Return $\perp$	If $R = 1$: Return $\perp$
$t \leftarrow f(R)$	$t \leftarrow f(R)$
$h \xleftarrow{\$} \mathbb{Z}_q$	$(s, h) \leftarrow \mathsf{Sim}^E(a, b, t)$
$s \leftarrow \mathsf{S}^E_{\mathtt{sk}}(h, t, r)$	If $(s, h, t) \notin \mathbb{D}$: Return $\perp$
If $(s, h, t) \notin \mathbb{D}$: Return $\perp$	Return (s, h, t)
Return (s, h, t)	

Theorem 1 *[14] Let $(E, \mathbb{G}, H, f, \mathbb{D})$ be a δ-simulatable of Gen$\mathcal{S}_{leeve}$. Then if H is modeled as random oracle, for every forger $\mathcal{F}$ that $(\tau, \mathcal{Q}_s, \mathcal{Q}_H, \varepsilon)$-breaks the one-per-message unforgeabillity if this instantiation there also exists a forger $\mathcal{F}'$ that $(\tau', 0, \mathcal{Q}_H, \varepsilon')$-breaks the key-only unforgeability of this instantiation, where $\varepsilon' \geq \varepsilon/(e^2(\mathcal{Q}_s + 1)) - \mathcal{Q}_s\delta$ and $\tau' = \tau + \mathsf{O}(\mathcal{Q}_H)$.*

Theorem 2 *[14] Let $(\mathbb{G}, g, q)$ be a prime-order group, let E be a defining equation with corresponding set $\mathbb{D}$, and let $f : \mathbb{G}^* \to \mathbb{Z}_q$ and $H : \{0, 1\}^* \to \mathbb{Z}_q$ be functions. If E is h-decomposable with functions ρ_0 and ρ_1, and H is modelled as a random oracle, then the SLP in $\mathbb{G}$ with respect to f, ρ_0, ρ_1 is non-tightly equivalent to the key-only unforgeability of Gen$\mathcal{S}_{leeve}$ when instantiated with $(E, \mathbb{G}, H, f, \mathbb{D})$.*

That is, for any adversary $\mathcal{I}$ that (τ, ε)-breaks SLP, there exists a forger $\mathcal{F}$ that (τ', ε)-breaks the key-only unforgeability of Generic Sleeve, where $\tau \approx \tau'$.

Conversely, for any forger $\mathcal{F}$ that $(\tau, \mathcal{Q}_H, \varepsilon)$-breaks the key-only unforgeability of Gen$\mathcal{S}_{leeve}$, there exists an adversary $\mathcal{I}$ that $(\tau', \varepsilon/\mathcal{Q}_H - 1/q)$-breaks SLP, where $\tau \approx \tau'$ and $\mathcal{Q}_H$ is the number of random oracle queries posed by $\mathcal{F}$.

Sections 4.1, 4.2 and 4.3 fully cover the security of the Tweakable $\mathcal{S}_{\text{leeve}}$, regarding ECDSA security, and Gen$\mathcal{S}_{\text{leeve}}$ with respect to SLP.

We now focus on the experimental results.

5 Implementation and Performance

This section describes our open-source implementation, the audit results along with the associated fixes, and details of the Verifpal formal analysis model.

Reference Implementation. We implemented a single-threaded version in Golang. In our implementation, W-OTS$^+$ uses SHA3 for public key compression and Blake2b for hash ladder calculations. We use the secp256k1 curve with ECDSA as

the main signature scheme and verified the correctness of our code, which integrates BIP39 [4], by comparing it with reference BIP39 implementations [5, 15]. Our implementation differs slightly from the original W-OTS$^+$ specification, which defines a secret key as ℓ random numbers and, instead, derives the secret key values from a single *seed* parameter by using a KDF. The W-OTS$^+_{\mathsf{pk}}$ is compressed using a tweakable hash function using the public seed, and the secret hash key value $\mathcal{X}$.

Audit. We expose the detailed results obtained from the official audit of the reference implementation and the subsequent fixes.

- **Scope:** The scope of the audit included the correctness of the cryptography and associated security, finding eventual timing leaks, usage of unsafe APIs, missing security checks, risk from dependencies, and poor randomness generation.
- **Security Issues:** No outstanding security issue appeared in the core cryptographic modules and the main security remarks are associated with a command line interface (CLI) tool created to improve the usability of the user. The audit results are openly available in [23].
- **Verifpal Implementation:** The code associated with the formal analysis tools is openly available on a Github repository in a special folder dedicated to the formal verification component [23].

Performance Metrics. We present performance metrics for our single-threaded implementation running on one Amazon c5.xlarge benchmark machine with an Intel Xeon Platinum 8124M 3.00GHz CPU and 8GiB RAM. Our code runs a $\mathcal{S}_{\text{leeve}}$ key generation in 1.81 ms, which comprises a W-OTS$^+$ key generation that takes 1.75 ms and an ECDSA key generation that takes 0.059 ms. These early results demonstrate that the key generation of the (tweakable) $\mathcal{S}_{\text{leeve}}$ construction is significantly slower than presently used key generation mechanism (i.e., ECDSA). These results are expected as the $\mathcal{S}_{\text{leeve}}$ construction introduces a significant amount of additional steps in the wallet generation process. Potential improvements may include calculating the W-OTS$^+$ hash-ladders in parallel and the use of different and potentially faster hash functions implementations.

6 Formal Methods Analysis

This section reports on the mathematical security proof of our construction, and outlines the Verifpal [20, 21] model we used to analyze the tweakable $\mathcal{S}_{\text{leeve}}$ along with some of the challenges that appeared throughout this process. We start by giving a brief summary on the Verifpal tool.

Verifpal. Verifpal is a software for verifying the security of cryptographic protocols. This tool is oriented towards real-world practitioners attempting to integrate formal verification into their line of work. To achieve this, Verifpal uses a new, intuitive language for modeling protocols that is considered easier to write and understand than the languages currently employed by existing tools.

Challenges to Modelling $\mathcal{S}_{leeve}$ in Verifpal. A commonly found problem in symbolic model protocol verifiers is that, for complex protocols, the different combinations of variables that the verifier must assess, quickly becomes too large to terminate in reasonable time. This is a challenge we faced in our modelling process as we initially attempted to model a W-OTS$^+$ fallback for ECDSA and the tool constantly issued memory fault errors when starting to perform the hash ladder iterations, which resulted in the stopping of the verification process in a faulty manner. Additionally, we highlight the lack of existence of the *XOR* logical function in the tool, which lead to design attempts with changed variants of the chaining function.

Verifpal Model of $\mathcal{S}_{leeve}$. To avoid the memory fault issues derived from iterating different attack scenarios involving a high number of hash function calls, we model a simpler Lamport signature scheme as a quantum-secure fallback instead of W-OTS$^+$.

Attacker model. All the interactions in the model go through an active attacker. Therefore, we assume the Dolev-Yao model [12] where the adversary is in charge of delivering the messages.

Results. The tool output that regardless of the compromise of the ECDSA secret key value, the queried values remain confidential, and only the true owner of the hash-based fallback key pair is able to produce a safety signature. We assume correctness of the Verifpal execution results, especially since there reslts match the results obtained in the security proof of $\mathcal{S}_{\text{leeve}}$.

7 Final Remarks

The $\mathcal{S}_{\text{leeve}}$ definition is a promising and novel scheme designed as an extension to existing wallets since, as quantum computers evolve, the security of most cryptocurrency wallets is at risk.

In this work, we improve on the original $\mathcal{S}_{\text{leeve}}$ construction by proposing the Tweakable $\mathcal{S}_{\text{leeve}}$. Thus we introduced a more modular approach that is simpler to analyze and implement. Moreover, we fill the missing gaps in the security proof of the original proposal, connecting it to the state-of-the-art of the ECDSA security. Namely, (1) our construction presents the same capabilities of the original $\mathcal{S}_{\text{leeve}}$, (2) it is at least as secure as the ECDSA signature scheme given a tweakable hash function whose output is computationally indistinguishable from the uniformly random distribution, and (3) our construction is generically secure, *i.e.* Gen$\mathcal{S}_{\text{leeve}}$ with respect to SLP.

Finally, we showcase our security results using the formal method analysis tool called *Verifpal*, which produced positive results matching the ones obtained in the mathematical proof of security. The distinctive extra level of security provided by $\mathcal{S}_{\text{leeve}}$ has the potential to help in the adoption of this new cryptographic primitive in the context of blockchain applications. Moreover this work illustrates that our construction is now open-source, audited, and features a more complete security

proof relating the construction with a concrete computational problem: a *must* in provable security practice.

Finally, this work illustrates a fruitful combination of theoretical work, from the protocol specification/construction, to the formal method analysis. Such thorough work which might raise the expectation of due diligence teams to include formal analysis when designing and evaluating cryptographic protocols.

A High-level Diagram of the Tweakable $\mathcal{S}_{\text{leeve}}$ Construction

This section exposes a high-level diagram of the sequence of performed steps in the key generation component of the $\mathcal{S}_{\text{leeve}}$ construction (Fig. 2).

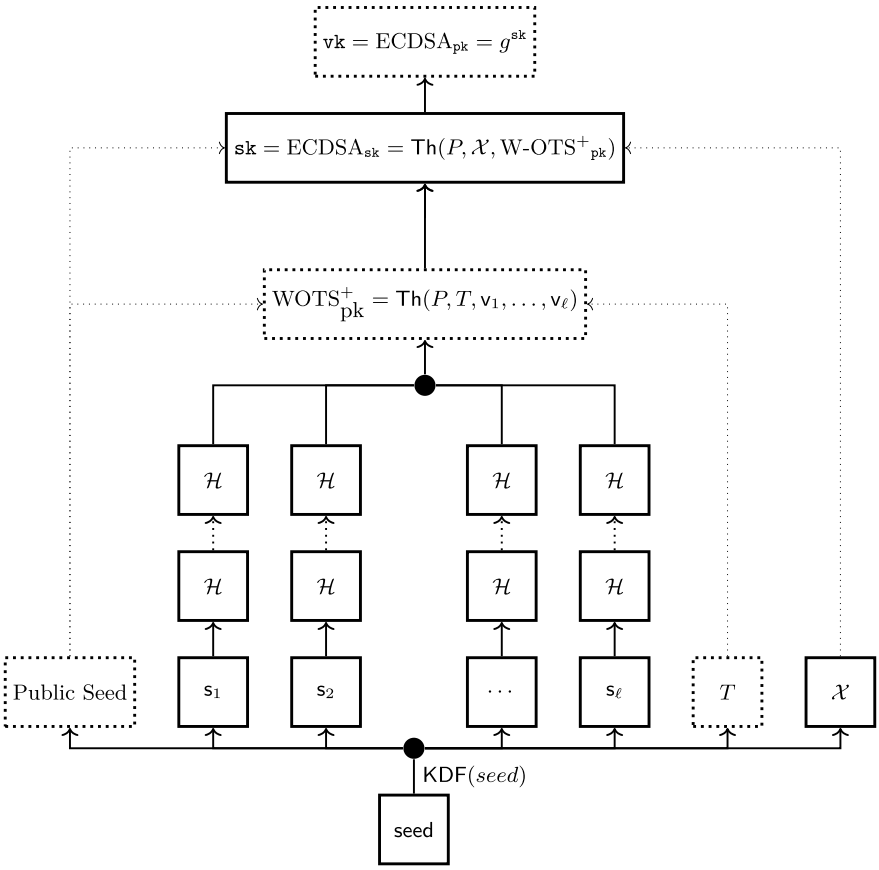

Fig. 2 $\mathcal{S}_{\text{leeve}}$ high-level diagram of the key generation

References

1. Aranha, D. F., Novaes, F. R., Takahashi, A., Tibouchi, M., & Yarom, Y. (2020). Ladderleak: Breaking ecdsa with less than one bit of nonce leakage. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (pp. 225–242). New York, NY, USA: CCS '20, Association for Computing Machinery.
2. Badertscher, C., Gazi, P., Kiayias, A., Russell, A., & Zikas, V. (2018). Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In D. Lie, M. Mannan, M. Backes, & X. Wang (Eds.), *ACM CCS* (pp. 913–930). ACM Press. https://doi.org/10.1145/3243734.3243848.
3. Bernstein, D. J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., & Schwabe, P. (2019). The SPHINCS$^+$ signature framework. In L. Cavallaro, J. Kinder, X. Wang, & J. Katz (Eds.), *ACM CCS* (pp. 2129–2146). ACM Press. https://doi.org/10.1145/3319535.3363229.
4. Mnemonic code for generating deterministic keys. Accessed September 10, 2021, from https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki.
5. Mnemonic code converter. Accessed September 10, 2021, from https://iancoleman.io/bip39/.
6. Brown, D. (2005). On the provable security of ECDSA, pp. 21–40. London Mathematical Society Lecture Note Series, Cambridge University Press.
7. Brown, D. R. (2005). Generic groups, collision resistance, and ecdsa. vol. 35, pp. 119–152. Springer.
8. Chaum, D., Larangeira, M., Yaksetig, M., & Carter, W. (2021). Wots+ up my sleeve! a hidden secure fallback for cryptocurrency wallets. In *International Conference on Applied Cryptography and Network Security* (pp. 195–219). Springer.
9. Chen, L. (2022). Recommendation for key derivation using pseudorandom functions-revision 1. NIST special publication. Accessed February 20, 2022, from https://doi.org/10.6028/NIST.SP.800-108r1-draft.
10. Dahmen, E., Okeya, K., Takagi, T., & Vuillaume, C. (2008). Digital signatures out of second-preimage resistant hash functions. In J. Buchmann, & J. Ding (Eds.), *Post-quantum Cryptography, Second International Workshop, PQCRYPTO* (pp. 109–123). Heidelberg: Springer. https://doi.org/10.1007/978-3-540-88403-3_8.
11. David, B., Gazi, P., Kiayias, A., & Russell, A. (2018). Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: J. B. Nielsen, & V. Rijmen (Eds.), *EUROCRYPT, Part II. LNCS* (vol. 10821, pp. 66–98). Heidelberg: Springer. https://doi.org/10.1007/978-3-319-78375-8_3.
12. Dolev, D., & Yao, A. (1983). On the security of public key protocols. *IEEE Transactions on Information Theory, 29*(2), 198–208.
13. Fersch, M., Kiltz, E., & Poettering, B. (2016). On the provable security of (ec)dsa signatures. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1651–1662). New York, NY, USA: CCS '16, Association for Computing Machinery.
14. Fersch, M., Kiltz, E., & Poettering, B. (2017). On the one-per-message unforgeability of (EC)DSA and its variants. In: Y. Kalai, & L. Reyzin (Eds.), *TCC 2017, Part II. LNCS* (vol. 10678, pp. 519–534). Heidelberg: Springer https://doi.org/10.1007/978-3-319-70503-3_17.
15. Golang implementation of the bip39 spec. Accessed September 10, 2021, from https://godoc.org/github.com/tyler-smith/go-bip39.
16. Hülsing, A. (2013). W-OTS+ - shorter signatures for hash-based signature schemes. In A. Youssef, A. Nitaj, & A. E. Hassanien (Eds.), *AFRICACRYPT 13. LNCS* (vol. 7918, pp. 173–188). Heidelberg: Springer. https://doi.org/10.1007/978-3-642-38553-7_10.
17. Ilie, D. I., Karantias, K., & Knottenbelt, W. J. (2020). Bitcoin crypto-bounties for quantum capable adversaries. *Cryptology ePrint Archive*, Paper 2020/186. https://eprint.iacr.org/2020/186.
18. Ilie, D. I., Knottenbelt, W. J., & Stewart, I. (2020). Committing to quantum resistance, better: A speed-and-risk-configurable defence for bitcoin against a fast quantum computing attack. *Cryptology ePrint Archive*, Paper 2020/187. https://eprint.iacr.org/2020/187.

19. Kiayias, A., Russell, A., David, B., & Oliynykov, R. (2017). Ouroboros: A provably secure proof-of-stake blockchain protocol. In: J. Katz, & H. Shacham (Eds.), *CRYPTO 2017, Part I. LNCS* (vol. 10401, pp. 357–388). Heidelberg: Springer. https://doi.org/10.1007/978-3-319-63688-7_12.
20. Kobeissi, N. (2021). Verifpal: Cryptographic Protocol Analysis for Students and Engineers. Accessed August 5, 2021, from https://verifpal.com.
21. Kobeissi, N., Nicolas, G., & Tiwari, M. (2020). Verifpal: Cryptographic protocol analysis for the real world. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop* (p. 159). New York, NY, USA: CCSW'20, Association for Computing Machinery.
22. Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf.
23. Sleeve. (2022). Accessed February 21, 2022, from https://github.com/xx-labs/sleeve/tree/main/verifpal_model.
24. Trinity attack incident part 1: Summary and next steps. Accessed September 22, 2020, from https://blog.iota.org/trinity-attack-incident-part-1-summary-and-next-steps-8c7ccc4d81e8.
25. Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper, 151*, 1–32.