

Threshold Structure-Preserving Signatures with Randomizable Key

Ahmet Ramazan Ağırtaş¹^a, Emircan Çelik²^b, Sermin Kocaman³^c, Fatih Sulak⁴^d
and Oğuz Yayla²^e

¹*Nethermind Research, London, U.K.*

²*Middle East Technical University, Ankara, Turkey*

³*FAME Crypt, Ankara, Turkey*

⁴*Atılım University, Ankara, Turkey*

Keywords: Structure-Preserving Signatures, Threshold Signatures, Randomizable Keys, Privacy.

Abstract: Digital signatures confirm message integrity and signer identity, but linking public keys to identities can cause privacy concerns in anonymized settings. Signatures with randomizable keys can break this link, preserving verifiability without revealing the signer. While effective for privacy, complex cryptographic systems need to be modular structured for efficient implementation. Threshold structure-preserving signatures enable modular, privacy-friendly protocols. This work combines randomizable keys with threshold structure-preserving signatures to create a valid, modular, and unlinkable foundation for privacy-preserving applications.


1 INTRODUCTION


Digital signatures are cryptographic techniques that provide a method to authenticate the integrity of the message and the identity of the signer. The message, once signed by the signer, can be verified using the signer's public key. However, in some certain scenarios, such as anonymized networks or privacy-preserving applications, the inherent connection between the public key and the signer's identity poses a problem (Balumuri et al., 2024). Revealing this information conflicts with preserving anonymity. In such situations, balancing authenticity with anonymity becomes a challenging task. To tackle this issue, Signatures with Randomizable Keys (SwRK) have drawn most of the attention to dissociate the signer's identity from their public key. In this scheme, the signer signs a message under the randomized secret signing key while ensuring that the signature remains valid. Although different terminology is employed throughout the literature, a comprehensive overview of previous work on this topic, connecting key-homomorphic sig-


natures, signatures with re-randomizable keys, signatures with key blinding, and key-updatable signatures, can be found in (Celi et al., 2023). While SwRK is useful for privacy-preserving protocols, it is also important for the protocol to be structured in a modular design to allow for the simple implementation of complex primitives. Fortunately, the concept of Structure Preserving Signatures (SPS) offers a way to construct modular protocols. SPS are signatures based on pairings, wherein verification keys, signatures, and messages are elements in a bilinear group, and the verification equation is derived from a pairing-product equation (Abe et al., 2010)


Related Work. In 2014, Abe et al. (Abe et al., 2014) proposed a re-randomizable SPS scheme that enables the randomization of both the signature and key after generation, while still preserving the structure of the signature. This scheme achieves direct randomization of the signature without the necessity of a message. In 2016, Ghadafi (Ghadafi, 2016) proposed shorter SPS than existing SPS schemes.


Many SPS in the literature are inadequate for thresholding due to nonlinear processes or even require significant overhead. However, in 2023, Crites et al. (Crites et al., 2023) proposed a Threshold Structure-Preserving Signature (TSPS) by defining indexed Diffie-Hellman message space. They address

^a <https://orcid.org/0000-0002-4574-0067>

^b <https://orcid.org/0009-0007-9707-2199>

^c <https://orcid.org/0000-0001-8334-8587>

^d <https://orcid.org/0000-0002-5220-3630>

^e <https://orcid.org/0000-0001-8945-2780>

thresholdizing by employing indexing that converts each scalar message m into an index id to generate partial signatures in a compatible format for aggregation in a threshold scheme.

Our Contribution. In this paper, we propose a novel structure preserving signature scheme with key randomization property. We also propose its threshold variant. Additionally we present cost analysis of our proposed schemes. Finally we provide a proof of concept implementation of our schemes with experimental results.

Paper Organization. This paper is organized as follows: Section 2 provides the necessary building blocks for our construction. Section 3 introduces our proposed Threshold Structure-Preserving Signature with Randomizable Keys (TSPSwRK) and its non-threshold counterpart IM-SPSwRK, followed by a performance analysis in Section 4. Finally, Section 5 offers the conclusion of the paper and future work.

2 NOTATION AND BACKGROUND

For the syntax, λ is the security parameter 1^λ is its unary representation, and $negl(\lambda)$ denotes negligible function. While $x \xleftarrow{\$} X$ is utilized to indicate that x is uniformly sampled from the set X , the cardinality of a set X is represented by the symbol $|X|$. Also, $A(x) \rightarrow y$ denotes that y is an output of the algorithm A on input x , whereas $x \leftarrow y$ denotes the straightforward assignment process. We denote linear pairing function $e(.,.)$ as described in (Galbraith et al., 2008).

Signatures with Randomizable Keys. Signature with Randomizable Keys (SwRK) consists of eight-tuple algorithms, namely PPGen, KGen, SignGen, Verify, RandPK, RandSK, Adapt, and VerKey, which was defined in (Celi et al., 2023).

Indexed-Message Structure-Preserving Signature. Indexed-Message Structure-Preserving Signature (IM-SPS) over the indexed Diffie-Hellman message space (Crites et al., 2023) builds four-tuple algorithms Setup, KGen, SignGen, and Verify, which can be described as follows.

1. $\text{Setup}(1^\lambda) \rightarrow pp$: This PPT algorithm takes the security parameter 1^λ as input, and it outputs public parameters $pp = ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2), H)$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of order p utilized in

bilinear map e , g_1 and g_2 are generators of \mathbb{G}_1 and \mathbb{G}_2 , respectively. Also, $H: \{0, 1\}^* \rightarrow \mathbb{G}_1$ is a hash function that maps arbitrary length binary inputs to the group \mathbb{G}_1 .

2. $\text{KGen}(pp) \rightarrow (sk, vk)$:
 - (a) Sample $sk = (sk_0, sk_1) = (x, y) \xleftarrow{\$} \mathbb{Z}_p^*$
 - (b) Compute $vk = (vk_0, vk_1) = (g_2^x, g_2^y)$
3. $\text{SignGen}(pp, sk, (id, M_1, M_2)) \rightarrow \sigma$:
 - (a) $(id, M_1, M_2) = (id, H(id)^m, g_2^m) \in \mathcal{M}_{iDH}^H$ as defined in (Crites et al., 2023), where $m \in \mathbb{Z}_p$
 - (b) Check $e(h, M_2) = e(M_1, g_2)$, where $h \leftarrow H(id)$. If this is not the case, abort.
 - (c) Compute the signature $\sigma = (h, s) = (h, h^x \cdot M_1^y)$.
4. $\text{Verify}(pp, vk, (M_1, M_2), \sigma) \rightarrow 0/1$:
 - (a) Accept if all the following conditions hold:
 - i. $h \neq 1_{\mathbb{G}_1}$,
 - ii. $M_1 \neq 1_{\mathbb{G}_1}$,
 - iii. $e(h, M_2) = e(M_1, g_2)$,
 - iv. $e(h, vk_0)e(M_1, vk_1) = e(s, g_2)$.
 - (b) Reject, otherwise.

Threshold SPS. Threshold Structure-Preserving Signatures (TSPS) for indexed message spaces (Crites et al., 2023) builds six-tuple algorithms Setup, KGen, PSignGen, PSignVerify, Reconst, and Verify, which can be described as follows.

1. $\text{Setup}(1^\lambda) \rightarrow pp$: This PPT algorithm takes the security parameter 1^λ as input, and outputs public parameters $pp = ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2), H)$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order p , utilized in an efficient bilinear map e , g_1, g_2 are generators of $\mathbb{G}_1, \mathbb{G}_2$, respectively, and $H: \{0, 1\}^* \rightarrow \mathbb{G}_1$ is a hash function that maps arbitrary length binary inputs to the group \mathbb{G}_1 .
2. $\text{KGen}(pp, n, t) \rightarrow (\vec{sk}, \vec{vk}, vk)$:
 - (a) Run Pedersen's Distributed Key Generation (PedDKG) protocol (Pedersen, 1991) as follows:
 - Each node N_i samples random two integers $(x_{i0}, y_{i0}) \xleftarrow{\$} \mathbb{Z}_p^*$ for $i \in [1, n]$, and forms two polynomials with degree t by sampling $2t$ random integer coefficient $\{x_{i\ell}, y_{i\ell}\}_{\ell=1}^t \xleftarrow{\$} \mathbb{Z}_p^*$:

$$f_i(X) = x_{i0} + x_{i1}X + \dots + x_{it}X^t \in \mathbb{Z}_p[X]$$

$$g_i(X) = y_{i0} + y_{i1}X + \dots + y_{it}X^t \in \mathbb{Z}_p[X]$$

Each N_i commits the coefficients by broadcasting $B_{i\ell} = g_2^{x_{i\ell}}$, $C_{i\ell} = g_2^{y_{i\ell}} \forall \ell \in [0, t]$. Then,

N_i sends $f_i(j), g_i(j)$ to N_j , such that $j \in [1, n] \setminus \{i\}$.

- Node N_i confirms the consistency of the received shares $f_j(i), g_j(i)$ from N_j by calculating $g_2^{f_j(i)} = \prod_{\ell=0}^t B_{j\ell}^{i\ell}$ and $g_2^{g_j(i)} = \prod_{\ell=0}^t C_{j\ell}^{i\ell}$. If these conditions are met, N_i accepts the shares; otherwise, it rejects them and complaints against malicious node N_j .
- A node is deemed disqualified if it receives at least t of complaints. At the conclusion of this phase, at least t nodes from the set of qualified nodes Q proceed to the following steps.
- The global verification key vk is determined as follows:

$$vk := (vk_{00}, vk_{01}) = (\prod_{i \in Q} B_{i0}, \prod_{i \in Q} C_{i0}) \\ = (g_2^{\sum_{i \in Q} x_{i0}}, g_2^{\sum_{i \in Q} y_{i0}}) = (g_2^x, g_2^y)$$

Also, its corresponding secret key will be $sk := (\sum_{i \in Q} x_{i0}, \sum_{i \in Q} y_{i0})$

- Each qualified node N_i defines its secret key sk_i and its corresponding verification key vk_i as follows:

$$sk_i := (sk_{i0}, sk_{i1}) = (\sum_{j \in Q} f_j(i), \sum_{j \in Q} g_j(i)) = (x_i, y_i) \\ vk_i := (vk_{i0}, vk_{i1}) = (\prod_{j \in Q} \prod_{\ell=0}^t (B_{j\ell})^{i\ell}, \prod_{j \in Q} \prod_{\ell=0}^t (C_{j\ell})^{i\ell}) \\ = (g_2^{f_j(i)}, g_2^{g_j(i)}),$$

where $f(X) = \sum_{j \in Q} f_j(X)$ and $g(X) = \sum_{j \in Q} g_j(X)$.

- Set two vectors of size n of signing keys $\vec{sk} \leftarrow (sk_1, sk_2, \dots, sk_n)$, and verification keys $\vec{vk} \leftarrow (vk_1, vk_2, \dots, vk_n)$
- $\text{PSignGen}(pp, sk_i, (id, M_1, M_2)) \rightarrow (\sigma_i, \perp)$:
 - $(id, M_1, M_2) = (id, H(id)^m, g_2^m) \in \mathcal{M}_{iDH}^H$, where $m \in \mathbb{Z}_p$
 - Check $e(h, M_2) = e(M_1, g_2)$, where $h \leftarrow H(id)$. If it does not hold, abort.
 - Compute the partial signatures $\sigma_i = (h_i, s_i) = (h, h^{sk_{i0}} \cdot M_1^{sk_{i1}})$.
- $\text{PSignVerify}(pp, vk_i, (M_1, M_2), \sigma_i = (h_i, s_i)) \rightarrow (0/1)$:
 - Accept if all the following conditions hold:
 - $h_i \neq 1_{\mathbb{G}_1}$,
 - $M_1 \neq 1_{\mathbb{G}_1}$,
 - $e(h_i, M_2) = e(M_1, g_2)$,
 - $e(h_i, vk_{i0})e(M_1, vk_{i1}) = e(s_i, g_2)$.
 - Reject, otherwise.

- $\text{Reconst}(pp, \vec{vk}, (M_1, M_2), \{\sigma_i\}_{i \in \mathcal{T}}) \rightarrow (\sigma, \perp)$:
 - Abort if at least one of the following conditions holds:
 - $\exists i \in \mathcal{T} \mid \text{PSignVerify}(pp, vk_i, (M_1, M_2), \sigma_i) = 0$,
 - $\exists i, j \in \mathcal{T}, i \neq j \mid h_i \neq h_j$.
 - Otherwise, set $h \leftarrow h_i$ and compute the signature $\sigma = (h, s) = (h, \prod_{i \in \mathcal{T}} s_i^{\lambda_i})$, where λ_i is an appropriate Lagrange coefficient.
- $\text{Verify}(pp, vk, (M_1, M_2), \sigma) \rightarrow (0/1)$:
 - Accept, if all of the following conditions hold:
 - $h \neq 1_{\mathbb{G}_1}$
 - $M_1 \neq 1_{\mathbb{G}_1}$
 - $e(h, M_2) = e(M_1, g_2)$
 - $e(h, vk_{00})e(M_1, vk_{01}) = e(s, g_2)$,
 - Reject, otherwise.

The three primary security properties for TSPS defined over indexed message spaces are *partial verification correctness*, *evaluation correctness*, and *threshold existential unforgeability against chosen indexed message attacks*. (Threshold EUF-CiMA)2

3 SPS AND THRESHOLD SPS WITH RANDOMIZABLE KEYS

Now, we introduce the randomizable key-enabled variant of the SPS and TSPS schemes.

To clarify the foundational structure of our protocol, we initially present its non-threshold version, the IM-SPS scheme in Section 2. We give the details of IM-SPS with a randomizable key, IM-SPSwRK in the following. It is equivalent to IM-SPS, with the added specification of SwRK algorithms RandPK , RandSK , and Adapt . While RandSK introduces randomness to the secret key, RandPK applies same randomness to the public key. Furthermore, Adapt modifies an existing signature to align with the randomized key, while keeping the original signing key confidential.

- $\text{Setup}(1^\lambda)$
 - $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2) \leftarrow \text{Gen}(1^\lambda)$
 - $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$
 - $pp \leftarrow ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2), H)$
 - return pp
- $\text{KeyGen}(pp, n, t)$
 - $sk \leftarrow (sk_0, sk_1) = (x, y) \xleftarrow{\$} \mathbb{Z}_p^*$
 - $vk \leftarrow (vk_0, vk_1) = (g_2^x, g_2^y)$
 - return (sk, vk)

3. RandSK($pp, sk, \rho = (r_0, r_1)$)
 - (a) $sk' = (sk'_0, sk'_1) = (x+r_0, y+r_1)$
 - (b) return sk'
4. RandPK($pp, vk, \rho = (r_0, r_1)$)
 - (a) $vk' = (vk'_0, vk'_1) = (g_2^x \cdot g_2^{r_0}, g_2^y \cdot g_2^{r_1})$
 - (b) return vk'
5. SignGen($pp, sk, (id, M_1, M_2)$)
 - (a) $(id, M_1, M_2) = (id, H(id)^m, g_2^m), h \leftarrow H(id)$
 - (b) If $e(h, M_2) \neq e(M_1, g_2)$: return \perp
 - (c) Else: $\sigma = (h, s) = (h, h^x \cdot M_1^y)$
 - (d) return σ
6. Adapt($pp, (id, M_1, M_2), \sigma, \rho, sk$)
 - (a) $\sigma' = (h, s') = (h, h^{sk'_0} \cdot M_1^{sk'_1})$
 $= (h, h^{x+r_0} \cdot H(id)^{m \cdot (y+r_1)})$
 - (b) return σ'
7. Verify($pp, vk', (M_1, M_2), \sigma'$)
 - (a) Parse $\sigma' = (h, s')$
 - (b) If $h \neq 1_{\mathbb{G}_1} \wedge M_1 \neq 1_{\mathbb{G}_1}$
 $\wedge e(h, M_2) = e(M_1, g_2)$
 $\wedge e(h, vk'_0) e(M_1, vk'_1) = e(s', g_2)$
 - (c) return 1
 - (d) Else: return 0

3.1 Threshold SPSwRK

In this section we give the details of our proposed Threshold Structure-Preserving Signatures with Randomizable Keys (TSPSwRK). TSPSwRK is equivalent to TSPS, with the added specification of SwRK algorithms RandPK, and Adapt. In addition, the threshold version requires RandGen, RandSKShare, and RandPKShare algorithms.

The Setup algorithm of TSPSwRK accepts the security parameter 1^λ and outputs the public parameters pp for the TSPS. The public parameters consist of groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of orders p used in bilinear map e , along with the generators g_1 and g_2 for \mathbb{G}_1 , and \mathbb{G}_2 , respectively. Additionally, there is a hash function that maps to \mathbb{G}_1 allowing messages to be securely represented as elements within the group.

KeyGen algorithm allows to each node N_i to independently generate its secret key sk_i and its corresponding verification key vk_i . After the generation of individual secret keys $\vec{sk} = (sk_1, \dots, sk_n)$ and their corresponding verification keys $\vec{vk} = (vk_1, \dots, vk_n)$, the algorithm generates the main verification key vk .

RandGen algorithm is run with a Distributed Key Generation (DKG) protocol without the involvement of a trusted entity. Even though Pedersen DKG is

used in (Crites et al., 2023), in scenarios where a trusted authority is responsible for randomizing the signature and public key, a Verifiable Random Function (VRF) (Micali et al., 1999) can be employed. Alternatively, a Distributed Verifiable Random Function (DVRF) (Ağirtaş et al., 2025) can be also utilized by the signers to achieve the same purpose.

RandSKShare and RandPKShare algorithm are executed by each node to randomize its secret key as sk'_i and its corresponding public verification key vk'_i , respectively. Also, RandPK algorithm generates the randomized main verification key vk' . Furthermore, since the Pedersen Distributed Key Generation (DKG) algorithm intrinsically includes its own proofs, a distinct VerKey algorithm is not established to validate correctness of the randomization process.

PSignGen algorithm generates the partial signatures σ_i on the indexed message (id, M_1, M_2) under the node's secret key sk_i , while Adapt algorithm outputs the randomized partial signature σ'_i under the node's randomized secret key sk'_i .

Reconst algorithm generates the randomized main signature σ aggregating threshold t randomized partial signatures σ'_i after verifying the correctness of them via PSignVerify. At the end of the scheme, the randomized main signature σ' is verified under randomized main verification key vk' .

1. Setup(1^λ)
 - (a) $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2) \leftarrow \text{Gen}(1^\lambda)$
 - (b) $H: \{0, 1\}^* \rightarrow \mathbb{G}_1$
 - (c) $pp \leftarrow ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2), H)$
 - (d) return pp
2. KeyGen(pp, n, t)
 - (a) Run PedDKG
 - (b) **for** $i \in [1, n]$:
 - $sk_i \leftarrow (sk_{i0}, sk_{i1}) = (x_i, y_i)$
 - $vk_i \leftarrow (vk_{i0}, vk_{i1}) = (g_2^{x_i}, g_2^{y_i})$
 - (c) **end for**
 - (d) $\vec{sk} \leftarrow (sk_1, \dots, sk_n); \vec{vk} \leftarrow (vk_1, \dots, vk_n)$
 - (e) $vk \leftarrow (vk_{00}, vk_{01}) = (g_2^x, g_2^y)$
 - (f) return (\vec{sk}, \vec{vk}, vk)
3. RandGen()
 - (a) Run PedDKG
 - (b) **for** $i \in [1, n]$:
 - $\rho_i = (r_{i0}, r_{i1});$
 - $(g_2)^{\rho_i} = (g_2^{r_{i0}}, g_2^{r_{i1}})$
 - (c) **end for**
 - (d) $(g_2)^p = (g_2^{r_0}, g_2^{r_1})$
 - (e) return $(g_2)^p$
4. RandSKShare(pp, sk_i, ρ)

- (a) $sk'_i = (sk'_{i0}, sk'_{i1}) = (sk_{i0} + r_{i0}, sk_{i1} + r_{i1})$
- (b) return sk'_i
5. RandPKShare(pp, vk_i, ρ)
 - (a) $vk'_i = (vk'_{i0}, vk'_{i1}) = (g_2^{x_i} \cdot g_2^{r_{i0}}, g_2^{y_i} \cdot g_2^{r_{i1}})$
 - (b) return vk'_i
6. RandPK(pp, vk, ρ)
 - (a) $vk' = (vk'_{00}, vk'_{01}) = (g_2^x \cdot g_2^{r_0}, g_2^y \cdot g_2^{r_1})$
 - (b) return vk'
7. PSigGen($pp, sk_i, (id, M_1, M_2)$)
 - (a) $(id, M_1, M_2) = (id, H(id)^m, g_2^m), h \leftarrow H(id)$
 - (b) If $e(h, M_2) \neq e(M_1, g_2)$: return \perp
 - (c) Else: $\sigma_i = (h, s_i) = (h, h^{sk_{i0}} \cdot M_1^{sk_{i1}})$
 - (d) return σ_i
8. Adapt($pp, (id, M_1, M_2), \sigma_i, \rho_i, sk_i$)
 - (a) $\sigma'_i = (h, s'_i) = (h, h^{sk_{i0}} \cdot M_1^{sk_{i1}})$
 $= (h, h^{x_i + r_{i0}} \cdot H(id)^{m \cdot (y_i + r_{i1})})$
 - (b) return σ'_i
9. PSigVerify($pp, vk'_i, (M_1, M_2), \sigma'_i$)
 - (a) Parse $\sigma'_i = (h_i, s'_i)$
 - (b) If $h_i \neq 1_{\mathbb{G}_1} \wedge M_1 \neq 1_{\mathbb{G}_1}$
 $\wedge e(h_i, M_2) = e(M_1, g_2)$
 $\wedge e(h_i, vk'_{i0}) e(M_1, vk'_{i1}) = e(s'_i, g_2)$
 - (c) return 1
 - (d) else: return 0
10. Reconst($pp, \{vk'_i\}_{i \in \mathcal{T}}, (M_1, M_2), \{\sigma'_i\}_{i \in \mathcal{T}}$)
 - (a) Parse $\sigma'_i = (h_i, s'_i)$
 - (b) If $\exists i, j \in \mathcal{T}, i \neq j \mid h_i \neq h_j \vee \exists i \in \mathcal{T} \mid$
 $\text{PSigVerify}(pp, vk'_i, (M_1, M_2), \sigma'_i) = 0$
 return \perp
 - (c) else: $h \leftarrow h_i$
 - (d) return $\sigma' \leftarrow (h, s) = (h, \prod_{i \in \mathcal{T}} s_i'^{\lambda_i})$
11. Verify($pp, vk', (M_1, M_2), \sigma'$)
 - (a) Parse $\sigma' = (h, s')$
 - (b) If $h \neq 1_{\mathbb{G}_1} \wedge M_1 \neq 1_{\mathbb{G}_1}$
 $\wedge e(h, M_2) = e(M_1, g_2)$
 $\wedge e(h, vk'_{00}) e(M_1, vk'_{01}) = e(s', g_2)$
 - (c) return 1
 - (d) Else: return 0

Correctness. First, we demonstrate that it fulfills the criteria for *partial evaluation correctness*, ensuring that a correctly created partial signature via the PSigGen successfully passes the PSigVerify checks after Adapt algorithm. Indeed, for all $i \in [1, n]$ and correctly indexed message $(id, M_1, M_2) \in \mathcal{M}_{IDH}^H$, it holds that:

$$\begin{aligned} e(h_i, vk'_{i0}) e(M_1, vk'_{i1}) &= e(h_i, g_2^{x_i + r_{i0}}) e(h_i^m, g_2^{y_i + r_{i1}}) \\ &= e(s'_i, g_2) \end{aligned}$$

Next, we show that it satisfies evaluation correctness, ensuring that for a set of partial signatures $\{i, \sigma'_i\}_{i \in \mathcal{T}}, \mathcal{T} \subset [1, n], |\mathcal{T}| = t$ on an indexed message $(id, M_1, M_2) = (id, H(id)^m, g_2^m) \in \mathcal{M}_{IDH}^H$ with the same $h \leftarrow H(id)$, Reconst algorithm successfully generates the valid aggregated signature $\sigma = (h, s)$ as follows:

$$\begin{aligned} s &= \prod_{i \in \mathcal{T}} s_i'^{\lambda_i} = \prod_{i \in \mathcal{T}} (h^{sk'_{i0}} M_1^{sk'_{i1}})^{\lambda_i} \\ &= h^{\sum_{i \in \mathcal{T}} (x_i \cdot \lambda_i + r_{i0} \cdot \lambda_i)} M_1^{\sum_{i \in \mathcal{T}} (y_i \cdot \lambda_i + r_{i1} \cdot \lambda_i)} \\ &= h^{(\sum_{i \in \mathcal{T}} (x_i \cdot \lambda_i) + \sum_{i \in \mathcal{T}} (r_{i0} \cdot \lambda_i))} M_1^{(\sum_{i \in \mathcal{T}} (y_i \cdot \lambda_i) + \sum_{i \in \mathcal{T}} (r_{i1} \cdot \lambda_i))} \\ &= h^{x + r_0} M_1^{y + r_1} \end{aligned}$$

4 ANALYSIS

4.1 Computational Cost Analysis

Let the setup be same as in the 3.1. Since our scheme relies on bilinear maps, our computations are performed over the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, with pairing operation $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Table 1 shows the detailed computational cost of each phase in our proposed TSPSwRK scheme relative to a standard TSPS. We also provide a comparison with its non-threshold variant, IM-SPSwRK, with the original IM-SPS. In this table, while $\text{Exp}_{\mathbb{G}_i}$ represents the exponentiation in the Group \mathbb{G}_i , $\text{Mul}_{\mathbb{G}_i}$ denotes the multiplication in the Group \mathbb{G}_i , where $i \in \{1, 2, T\}$. Also, pairing represents the pairing operation $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, and Hash represents the hash operation in \mathbb{G}_1 .

4.2 Implementation and Evaluation

The PoC implementation of the TSPSwRK protocol can be found on ¹. All experiments were carried out on a 64-bit operating system with a 6-core CPU running at approximately 3.2 GHz and 16 GB of RAM. The system did not have any specialized hardware acceleration. The protocol was executed at least 10 times for each configuration, parameterized by the total number of participants (n) and the threshold (t). For each run, the execution time of protocol phases was recorded. The mean execution time for each component was then computed for each (n, t) configuration. Since the complexity of distributed key generation is $O(n^2)$, a large number of participants would not be practical. Figure 1 shows these results.

¹https://github.com/emir0x1/tsps_wrk/

Table 1: The Computational Cost Analysis.

Phases	IM-SPS	IM-SPSwRK	TSPS	TSPSwRK
Key Generation	2Exp_{G_2}	$4t + 2\text{Exp}_{G_2}, 4t - 4\text{Mul}_{G_2}$	$n + 1\text{Exp}_{G_2}$	$4t + 2\text{Exp}_{G_2}, 4t - 4\text{Mul}_{G_2}$
RandGen				$4t + 4\text{Exp}_{G_2}, 4t - 4\text{Mul}_{G_2}$
RandPKShare				$2\text{Exp}_{G_2}, 2\text{Mul}_{G_2}$
RandPK		$2\text{Exp}_{G_2}, 2\text{Mul}_{G_2}$		$2\text{Exp}_{G_2}, 2\text{Mul}_{G_2}$
SignGen, PSignGen	$2\text{pairing}, 2\text{Exp}_{G_1}, 1\text{Mul}_{G_1}$	$3\text{Exp}_{G_1}, 1\text{Mul}_{G_1}, 1\text{Exp}_{G_2}, 1\text{Hash}(H_1), 1\text{pairing}$	$2\text{pairing}, 1\text{Mul}_{G_1}, 1\text{Mul}_{G_1}$	$3\text{Exp}_{G_1}, 1\text{Mul}_{G_1}, 1\text{Exp}_{G_2}, 1\text{Hash}(H_1), 1\text{pairing}$
Adapt		$2\text{Exp}_{G_1}, 1\text{Mul}_{G_1}$		$2\text{Exp}_{G_1}, 1\text{Mul}_{G_1}$
PSignVerify			$5\text{pairing}, 1\text{Mul}_{G_1}$	$5\text{pairing}, 1\text{Mul}_{G_1}$
Reconst			$5\text{pairing}, t - 1\text{Mul}_{G_1}, t\text{Exp}_{G_1}, t\text{Exp}_{G_1}$	$5\text{pairing}, t\text{Exp}_{G_1}, t\text{Mul}_{G_1}$
Verify	$5\text{pairing}, 1\text{Mul}_{G_1}$	5pairing	$5\text{pairing}, 1\text{Mul}_{G_1}$	5pairing

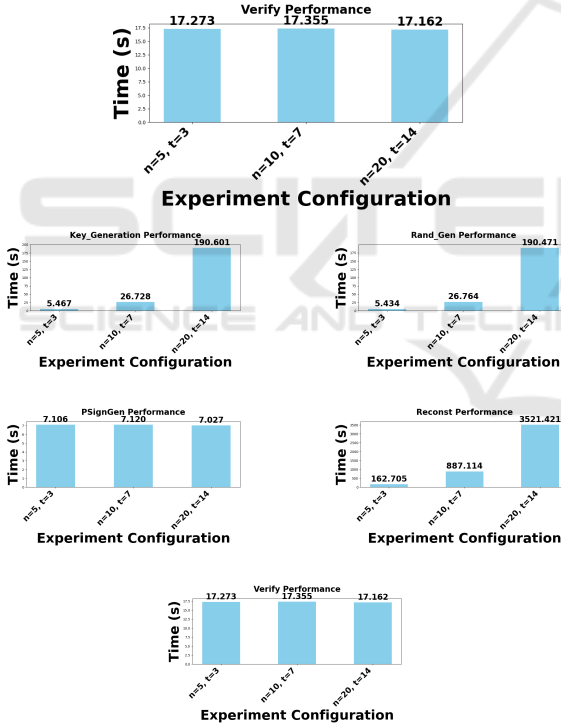


Figure 1: Performance metrics of TSPSwRK.

5 CONCLUSION AND FUTURE WORK

In this study, we proposed threshold structure preserving signatures with randomization property and also its threshold variant TSPSwRK. Additionally, cost analysis comparison of the protocols are pre-

sented. Further, proof of concept implementation of TSPSwRK protocol and performance analysis of the protocol is presented. We plan to improve the performance of the TSPSwRK by reducing the size of the shares (Krawczyk, 1994). Exploring the hierarchical threshold (Tassa, 2007) variant of TSPSwRK and application of Nested Shamir Secret sharing is also left for future work.

REFERENCES

- Abe, M., Groth, J., Ohkubo, M., and Tibouchi, M. (2014). Structure-preserving signatures from type II pairings. In *Advances in Cryptology—CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17–21, 2014, Proceedings, Part 1* 34, pages 390–407. Springer.
- Ağırtaş, A. R., Özer, A. B., Saygi, Z., and Yayla, O. (2025). Distributed verifiable random function with compact proof. In Dolev, S., Elhadad, M., Kutylowski, M., and Persiano, G., editors, *Cyber Security, Cryptology, and Machine Learning*, pages 119–134. Cham. Springer Nature Switzerland.
- Balumuri, S., Eaton, E., and Lamontagne, P. (2024). Quantum-safe public key blinding from MPC-in-the-head signature schemes. *Cryptology ePrint Archive*.
- Celi, S., Griffy, S., Hanzlik, L., Kempner, O. P., and Slamanig, D. (2023). Sok: Signatures with randomizable keys. *Cryptology ePrint Archive*.
- Crites, E., Kohlweiss, M., Preneel, B., Sedaghat, M., and Slamanig, D. (2023). Threshold structure-preserving signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 348–382. Springer.
- Galbraith, S. D., Paterson, K. G., and Smart, N. P. (2008). Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121. Applications of Algebra to Cryptography.
- Ghadafi, E. (2016). Short structure-preserving signatures. In *Cryptographers’ Track at the RSA Conference*, pages 305–321. Springer.
- Krawczyk, H. (1994). Secret sharing made short. In Stinson, D. R., editor, *Advances in Cryptology — CRYPTO’ 93*, pages 136–146, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Micali, S., Rabin, M., and Vadhan, S. (1999). Verifiable random functions. In *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*, pages 120–130. IEEE.
- Pedersen, T. P. (1991). Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer.
- Tassa, T. (2007). Hierarchical threshold secret sharing. *J. Cryptol.*, 20(2):237–264.