Structure-Preserving Threshold Signatures

Mahdi Sedaghat[1], Daniel Slamanig[2], Markulf Kohlweiss[3], and Bart Preneel[1]

[1] imec-COSIC, KU Leuven, Leuven, Belgium
ssedagha@esat.kuleuven.be, bart.preneel@esat.kuleuven.be
[2] AIT Austrian Institute of Technology, Vienna, Austria
daniel.slamanig@ait.ac.at
[3] University of Edinburgh and IOHK, Edinburgh, UK
mkohlwei@inf.ed.ac.uk

Abstract. The by now broadly accepted reliance of society on online services, led to a push for decentralization to mitigate the societal and technical risks caused by single points of failure (PoF). One such PoF are cryptographic keys. Thus there is renewed interest in threshold cryptography to distribute the generation and use of such keys. Structure-preserving signatures (SPS) are an important building block for privacy-preserving cryptographic protocols such as electronic cash and (delegatable) anonymous credentials. However, to date, no structure-preserving threshold signatures (SPTS) are available. This is unfortunate, as another PoF is centralized identity management, which could be mitigated by anonymous credentials.

In this work we aim to close this gap by introducing a notion and constructions of (non-) interactive SPTS. While it is relatively easy to devise interactive SPTS supporting static corruptions, e.g., based on the SPS of Ghadafi (CT-RSA'16), constructing non-interactive SPTS is a much more delicate task. Due to their structural properties, starting from existing SPS does not yield secure schemes. Thus, we take a different path and first introduce the notion of message-indexed SPS, a variant of SPS that is parameterized by a message indexing function. Inspired by Pointcheval-Sanders (PS) signatures (CT-RSA'16) and the SPS of Ghadafi, we then present a message-indexed SPS, which is non-interactive threshold-friendly. We prove its security in the random oracle model based on a variant of the generalized PS assumption. Based on our message-indexed SPS we then propose the first non-interactive message-indexed SPTS, which we prove to be secure under adaptive corruption. Finally, we discuss applications of SPTS to privacy-preserving primitives.

Keywords: Threshold Signatures, Structure-Preserving Signatures, Message-Indexed Structure-Preserving Signatures, Groth-Sahai Proof System, Threshold-Issuance Anonymous Credentials, Threshold Group Signatures.

1 Introduction

There is a push towards robust and privacy-friendly, decentralized systems. However, in the decentralized setting, the management of cryptographic keys becomes a challenging issue. Threshold cryptography [DDFY94,Des90,DF90] is the method of choice to improve availability of keying material and to reduce the trust in single entities. Threshold cryptography allows a secret key to be shared among $n > 1$ parties [Sha79,Bla79] such that the task involving the key can only be performed, if at least $t \leq n$ parties collaborate. Threshold primitives such as threshold encryption [SG98,CGJ$^+$99], threshold signatures [Sho00,DK01], and threshold verifiable unpredictable functions [GJM$^+$21] enable distributed protocols, e.g., e-voting systems [CGS97,CFSY96] or multi-party computation [CDN01,DN03], to avoid single points of failure. Consequently, there is increased interest in threshold cryptography standards, e.g., by NIST [BDV$^+$20], and products, e.g., by Unbound and Sepior[4]

Due to the rise of cryptocurrencies, blockchain technology, and self-sovereign identity management, threshold signatures [Des90] attract significant research interest, e.g., [MR01,Bol03,Lin17] and [DKLs19,CGG$^+$20,KMOS21]. We recall that a (n, t)-threshold signature scheme distributes the signing key among n signers and any subgroup of size at least t can jointly generate a signature. Unforgeability holds as long as fewer than t key shares are known to the adversary.

A threshold signature is said to be non-interactive if the signers can generate their own partial signatures independently without communicating with the other signers. Compared to threshold variants of Schnorr [GJKR03,KG20] and (EC)DSA [GJKR96,DKLs19,CGG$^+$20], due to their deterministic behavior threshold variants of RSA [RSA78] or BLS signatures [BLS01] tend to be the most appropriate schemes for non-interactive threshold signatures [Sho00,DK01,Bol03,BL22]. Besides the distributed use of the secret keys, an aspect of threshold cryptography worth emphasizing is Distributed Key Generation (DKG) [Ped92]. It avoids the presence of a trusted party to initially generate the key shares: in DKG the secret key can be generated in a distributed way without anyone learning the entire key.

Structure-preserving (threshold) signatures. Structure-preserving signatures (SPS) [AFG$^+$10] are signatures constructed over bilinear groups, i.e., groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, all of prime order p, equipped with a non-degenerate and efficiently computable pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. One requires that messages and signatures only include source group elements (elements from $\mathbb{G}_1$ and $\mathbb{G}_2$) and the signature verification only checks group membership and pairing product equations. After their invention, SPS have been extensively studied with a focus on short signatures [AGHO11,AGOT14,Gha16,Gha17], lower bounds [AGHO11,AGO11,AAOT18], as well as (tight) security under well-known assumptions [ACD$^+$12,HJ12,KPW15,LPY15,JR17,GHKP18,AJO$^+$19]. SPS are compatible with the Groth and Sahai (GS) NIZK proofs [GS08] and, more generally, help avoid the expensive extraction of exponents in security proofs. This makes them attractive for a wide variety of privacy-preserving applications, such as group signatures [AFG$^+$10,LPY15], traceable signatures [ACHO11], blind signatures [AFG$^+$10,FHS15], attribute-based signatures [EGK14], malleable signatures [ALP12], anonymous credentials [Fuc11,CDHK15], anonymous e-cash [BCF$^+$11] or access control encryption [WC21,SP21].

[4] `https://www.unboundsecurity.com/`, `https://sepior.com`.

While many of the aforementioned applications of SPS are an attractive target for thresholdization, as of now there is no known threshold construction of SPS that could serve as their basis.

1.1 Our Contributions

Our contributions can be summarized as follows:

- *Message-Indexed Structure-Preserving Signatures.* In Sect. 3, we introduce the notion of message-indexed SPS along with its existential unforgeability under indexed chosen message attack (EUF-CiMA) security. This is an important step in making SPS threshold-friendly. Then, we propose a concrete EUF-CiMA secure SPS scheme which we prove secure in the random oracle model under a new variant of the generalized PS assumption [KSAP21]. We show the security of this assumption under the Strong Discrete Logarithm (SDL) assumption [BCN+10] in the Algebraic Group Model (AGM) [FKL18]. Our signatures are highly efficient and consist only of two group elements. This is achieved by relying on an indexed Diffie-Hellman message space, which allows us to de-randomize signatures and to bypass known impossibility results for unilateral SPS schemes [AGHO11,Gha16].
- *Structure-Preserving Threshold Signatures.* In Sect. 4, we introduce the notion of structure-preserving threshold signatures (SPTS) and propose an efficient and practical non-interactive SPTS based on our EUF-CiMA secure SPS scheme. Our SPTS is proven to be unforgeable under adaptive corruptions.
- *Applications.* In Sect. 5, we discuss applications of the proposed SPTS scheme to anonymous credential systems with threshold-issuance like Coconut [SAB+19] and Coconut^{++} [RP22], and threshold dynamic group signatures [CDL+20].

1.2 Outline and Overview

Desirable properties for SPTS. In addition to desirable properties such as *non-interactive* signing, *uniqueness* of signatures, and *not* just *one-time* signature security, we require that structure-preserving threshold signatures satisfies the following criteria: i) verification keys consist of source group elements ($\mathbb{G}_1$ and $\mathbb{G}_2$) of a bilinear group, ii) the signature only consists of source group elements, iii) to verify a signature, only source group messages are needed iv) the signature components are threshold-friendly, and v) only source group membership testing and pairing product equations of the form $\prod_i \prod_j e(G_i, H_j)^{c_{i,j}} = 1_{\mathbb{G}_T}$ need to be executed in the verification algorithm, where $G_i \in \mathbb{G}_1$ and $H_j \in \mathbb{G}_2$ and $c_{i,j} \in \mathbb{Z}_p$.

Existing Schemes Close to Our Requirements. Table. 1.2 lists threshold signature schemes and existing SPS that are close to what we want to achieve. Unfortunately, they all fail to satisfy some of the aforementioned requirements.

Table 1. The List of Related Existing Threshold Signatures and Structure-Preserving Signatures. SP, SG, VK, TF and PPE stand for Structure-Preserving, Source Group, Verification Key, Threshold-Friendly and Pairing Product Equation respectively. ✓: Applicable, ✗: Not satisfied.

Type.	**Scheme**	Non-Interactive	Uniqueness	Not One-time signature	VK of SG elements	Signature of SG elements	Message of SG elements	TF Signature components	PPE in Verification
Threshold Signatures	[Bol03,BL22]	✓	✓	✓	✓	✓	✗[a]	✓	✗
	[LJY14, ‡ 1]	✓	✓	✓	✓	✓	✗[a]	✓	✗
	[LJY14, ‡ 2]	✓	✓	✓	✓	✓	✗[a]	✓	✗
	[GJM+21]	✓	✓	✓	✓	✓	✗[a]	✓	✗
SP Signatures	[LPJY13]	✓	✓	✗	✓	✓	✓	✓	✓
	[Gha16]	✗	✗	✓	✓	✓	✓	✓	✓

[a] The message space is not specially for a SG message element and can take any arbitrary bit-length messages.

More precisely, the (adaptively-secure) threshold variant of BLS [Bol03,BL22] maps any arbitrary messages to the group using a structure-destroying hash-to-curve function modeled as a random oracle. Libert et al. [LJY14] propose adaptively secure non-interactive threshold signatures that are based on linearly-homomorphic SPS (LHSPS) [LPJY13]. Despite the fact that this construction is the closest to our requirements, the resulting threshold signature is not structure-preserving. They either need to rely on random oracles to hash scalar messages to group elements [LJY14, ‡ 1] or, when avoiding random oracles, a bit-wise encoding of the scalar message is required [LJY14, ‡ 2]. Gurkan et al. [GJM$^+$21] proposed a so-called (threshold) structure-preserving Verifiable Unpredictable Function (VUF), which is essentially a unique (threshold) signature [MRV99]. However, their construction is not structure-preserving in the sense of SPS, as like BLS it hashes arbitrary messages to the group using a random oracle.

Taking a closer look at the existing SPS constructions, there are two promising constructions: LHSPS [LPJY13], but as discussed in [LJY14, Sect. 2], the resulting threshold variant of this scheme is only a one-time signature. A one-time signature is a digital signature that can sign only one message per key pair. Ghadafi's SPS [Gha16] is a threshold-friendly SPS construction, but as we discuss in this paper it only results in an interactive SPTS (and is not unique).

Challenges and Techniques. In a (n,t)-SPTS, a secret signing key, $s \in \mathbb{Z}_p$, is shared among n parties, $\{s_i\}_{i\in[1,n]}$, where up to $t-1$ parties are allowed to be corrupted. To enable the reconstruction of a signature from partial signatures of at least t parties, the components should be threshold-friendly. Loosely speaking, a partial signature element such as A^{s_i} is threshold-friendly, when one can implicitly compute s as a linear combination of s_i and Lagrange coefficients L_i by computing a product function $\prod(A^{s_i})^{L_i}$. For instance, partial signature elements like A^{1/s_i} are not threshold-friendly as it is not possible to compute the linear combination of s_i in the exponent. Due to the constraints imposed by being structure preserving, a partial signature must only contain source group elements and must not hash messages during verification.

Our starting point for overcoming these challenges are Pointcheval-Sanders (PS) signatures [PS16,PS18] which neither make use of key inversion during signing nor hashing during verification. An interesting aspect of PS signatures is that the signing randomness represents a random basis element. However, the messages of both PS variants are scalars and thus do not satisfy our requirements. Fortunately, Ghadafi [Gha16] proposed an SPS that is very similar to PS signatures [PS16]. Meanwhile as we discuss later in the threshold variant of Ghadafi's SPS to have a uniquely reconstructed signature, we would need to use a pre-determined random exponents among the signers. This, however, contradicts non-interactivity and would result in an interactive (at least two-round) SPTS. Interactive signing, however, requires coordination and communication among all the parties, i.e., all need to be online: this is often not desirable or even not permitted [CGG+20] and can lead to undetected problems due to (complicated) sub-protocols [TS21]. Consequently, protocols that offer non-interactive signing are preferable. Here one promising starting point is to exploit a technique used in Coconut [SAB+19] and by Camenisch et al. [CDL+20] to non-interactively agree on a common random source group element via hashing in the signing phase. In [SAB+19] the input to the hash function is a commitment of the given message, while in [CDL+20] pre-determined indices are assumed.

We first propose a structure-preserving variant of PS which can be viewed as a modification of Ghadafi's SPS and show it's security under a new notion of existential unforgeability that we call chosen indexed-message attacks (EUF-CiMA). This indexing can be seen as a parameterized way of instantiating the generation of a common random source group element. This makes the underlying SPS scheme threshold-friendly and suitable to finally construct a SPTS. Signature of this scheme only consist of two source group elements and are thus one element shorter than Ghadafi's SPS [Gha16] and identical in size to PS signatures [PS16] in a weaker notion of security that we call EUF-CiMA.

Bypassing impossibility of unilateral SPS. Unilateral SPS are ones where signatures exclusively contain elements of one source group. It is known that it is impossible to even build unilateral SPS that are secure against random message attacks [AGHO11]. However, in an asymmetric bilinear setting and over a Diffie-Hellman message space [AFG+10], where the message space is dual in both source groups, Ghadafi [Gha16] has shown that building a unilateral SPS is indeed possible. While this scheme is not suitable for a non-interactive SPTS, our approach also circumvents the impossibility in a similar way, but we work on a so called indexed Diffie-Hellman message space.

Adaptive vs. Static Corruption. Threshold-issuance anonymous credential systems are a primary application of the proposed SPTS construction. Despite the fact that Coconut [SAB+19] does not provide a rigorous security analysis on their construction, recently Rial and Piotrowska [RP22] provide a full security analysis of a modified Coconut scheme, called Coconut^{++}. However, this scheme is only secure under static-corruptions, i.e., where the adversary must choose all corrupted parties at the beginning of the protocol (after observing their public keys). However, in reality corruptions can happen over time and a protection against stronger adversaries, i.e., against adaptive corruptions, is required. Recently Bacho and Loss [BL22] proposed a new technique to prove the unforgeability of threshold BLS signatures under adaptive corruptions based on One-More Discrete Logarithm (OMDL) assumption. We adopt their proof technique in this work and show that the proposed SPTS is existentially unforgeable against chosen indexed-message attacks with adaptive corruption.

2 Preliminaries and Definitions

Throughout, let $p \in \mathbb{P}$ denote a prime number with bit length polynomial in the security parameter of $\lambda \in \mathbb{N}$ with unary representation of 1^λ. For all positive polynomials $f(\lambda)$, a function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}^+$ is called negligible if $\exists\ \lambda_0 \in \mathbb{N}$ s.t. $\forall\ \lambda > \lambda_0$ we have: $\mathsf{negl}(\lambda) < 1/f(\lambda)$. We assume a field of prime order $\mathbb{Z}$ and denote $\mathbb{Z}_p^{\leq d}[X]$ as a set of univariate polynomials with degree $\leq d$. We use $Y \leftarrow\!\!\$\, F(X)$ to denote a probabilistic function F that on input X samples the output Y. We use $x \leftarrow\!\!\$\, \mathbb{Z}_p$ to denote a uniformly random integer x is sampled from $\mathbb{Z}_p$. We denote the set of integers $\{1, \ldots, n\}$ for an integer $n > 1$ by $[1, n]$. The algorithms are randomized unless expressly stated. "PPT" refers to "Probabilistic Polynomial Time". The vector of A is denoted by $\vec{A}$. We denote the output of a security game G_ζ between a challenger and a PPT adversary $\mathcal{A}$ by $G_\zeta^{\mathcal{A}}$ and it is said $\mathcal{A}$ wins the game if $G_\zeta^{\mathcal{A}} = 1$ with an advantage of $Adv_{\zeta,\mathcal{A}}^{G}(\lambda) = \Pr[G_\zeta^{\mathcal{A}} = 1]$.

Remark 2.1. For a given cyclic group $\mathbb{G}$ with generator $g \in \mathbb{G}$, messages of source group elements are denoted by M and the discrete logarithm of the message under the basis of the generator of group are denoted by m, where $m := \mathsf{dlog}_g(M)$. To compress group element representation for $b \in \{1, 2\}$ and $k \in \{n, m\}$ we use $g_{bk} \in \mathbb{G}_1^n \times \mathbb{G}_2^m$ to denote the set of group elements $(g_{11}, \ldots, g_{1n}, g_{21}, \ldots, g_{2m})$.

Definition 2.1 (Bilinear Groups [BF01]). *A bilinear group generator* $\mathcal{BG}(1^\lambda)$ *returns a tuple* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$*, such that* $\mathbb{G}_1$*,* $\mathbb{G}_2$ *and* $\mathbb{G}_T$ *are finite groups of the same* *prime order p, $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are the generators and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing. We set $g_T = e(g_1, g_2)$ that is the generator of the target group $\mathbb{G}_T$. In the multiplicative notion, the tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$ is called a bilinear group setting if these conditions hold:*

1. *$e(g_1, g_2) \neq 1_{\mathbb{G}_T}$ (non-degenerate).*
2. *e is an efficiently computable bilinear map.*
3. *$\forall\ a, b \in \mathbb{Z}_p$: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} = e(g_1^b, g_2^a)$ (bilinearity).*

In this paper we rely on bilinear groups with no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ [GPS08], aka Type-III pairings. They are to date the most efficient choice for relevant security levels [CM11].

Diffie-Hellman Message Space. Over an asymmetric bilinear group, a pair $(M_1, M_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ is called a Diffie-Hellman (DH) message [Fuc09,AFG+10] if there exists $m \in \mathbb{Z}_p$ s.t. $M_1 = g_1^m$ and $M_2 = g_2^m$. One can efficiently verify whether $(M_1, M_2) \in \mathcal{M}_{\mathsf{DH}}$ by checking $e(M_1, g_2) = e(g_1, M_2)$.

In this paper, we adapt the DH message space to a tuple $(id, M_1, M_2) \in \mathbb{I} \times \mathbb{G}_1 \times \mathbb{G}_2$ which uses a random basis h computed using a random oracle instead of g_1, that is:

Definition 2.2 (Indexed Diffie-Hellman Message Space). *Let H be a random oracle. $\mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$ is an indexed DH message space, if the following two properties hold:*

1. *For every $(id, M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$ there exists $m \in \mathbb{Z}_p$ s.t. for $h = \mathsf{H}(id)$, $M_1 = h^m$, $M_2 = g_2^m$.*
2. *For all $(id, M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$, $(id', M_1', M_2') \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$, $id = id' \Rightarrow (M_1, M_2) = (M_1', M_2') \in \mathcal{M}_{\mathsf{iDH}}$. That is, no two messages use the same index.*

The first condition is efficiently decidable by checking $e(M_1, g_2) = e(h, M_2)$. Note that in addition one needs to guarantee that no two messages use the same index. This is the responsibility of the signer. We denote the tuple $(id, M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$ by M while the pair $(M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}$ without index id is denoted by $\tilde{M}$.

2.1 Shamir Secret Sharing (SSS)

A (n, t)-Shamir Secret Sharing (SSS) [Sha79] scheme divides a secret s among n shareholders such that each subset of t shareholders can reconstruct s but any smaller subset of them learn nothing about the secret. For this purpose, the dealer who knows the secret s forms a polynomial $f(x)$ of degree t with randomly chosen coefficients such that $f(0) = s$. Then the dealer securely provides each shareholder with $s_i = f(i), i \in \{1, \ldots, n\}$. Each subset of $\mathcal{T} \subset \{1, \ldots, n\}$ with size at least t can pool their shares to reconstruct the secret s using Lagrange polynomial interpolation as, $s = f(0) = \sum_{i \in \mathcal{T}} s_i L_i^{\mathcal{T}}(0)$, where $L_i^{\mathcal{T}}(x) = \prod_{j \in \mathcal{T}, j \neq i} \frac{x-j}{i-j}$.

2.2 Algebraic Group Model

Algebraic Group Model (AGM) was introduced by Fuchsbauer, Kiltz and Loss [FKL18] and it lies between the Generic Group Model (GGM) [Sho97] and Standard Model (SM). AGM is similar to SM, but differs from GGM in that cyclic groups are actually represented through an algebraic algorithm AGM, on the other hand, is similar to GGM but differs from SM as an algebraic algorithm can only produce group elements by employing group operations on the given group elements. Although an algebraic algorithm does not need to interact with an oracle to perform a computation, it must output a record of a group operation, known as a representation.

The AGM definition in [FKL18] only captures regular cyclic groups, whereas Mizuide et al. [MTT19] extends this definition to include symmetric pairing groups, where $\mathbb{G}_1 = \mathbb{G}_2$, such that the algebraic adversary is also allowed to output the target group elements and their representations. Recently, Couteau and Hartmann [CH20] defined the Algebraic Asymmetric Bilinear Group Model which extends the AGM definition for asymmetric pairings by allowing the adversary to output multiple elements from all three groups. We recall it in the following definition.

Definition 2.3 (Algebraic Adversaries in an Asymmetric Bilinear Group [CH20]). *For a given asymmetric bilinear group* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$*, an adversary* $\mathcal{A}_{alg}$ *who takes the vectors* $\vec{\zeta}_1 = (g_1^{x_1}, \ldots, g_1^{x_n}) \in \mathbb{G}_1^n$*,* $\vec{\zeta}_2 = (g_2^{y_1}, \ldots, g_2^{y_m}) \in \mathbb{G}_2^m$ *and* $\vec{\zeta}_T = (g_T^{z_1}, \ldots, g_T^{z_\ell}) \in \mathbb{G}_T^\ell$ *is called algebraic in an asymmetric bilinear group if it outputs:*

$$S = \left(g_1^{S_1^1}, \ldots, g_1^{S^1_{n'}}, g_2^{S_1^2}, \ldots, g_2^{S^2_{m'}}, g_T^{S_1^T}, \ldots, g_T^{S^T_{\ell'}} \right) \in \mathbb{G}_1^{n'} \times \mathbb{G}_2^{m'} \times \mathbb{G}_T^{\ell'} \ , \tag{1}$$

along with a representation vector of size $n \cdot n' + m \cdot m' + \ell'(\ell + n \cdot m)$*, as follows:*

$$\vec{S} = \left((\alpha_{ij})_{\substack{i \in [1,n'] \\ j \in [1,n]}}, (\beta_{ij})_{\substack{i \in [1,m'] \\ j \in [1,m]}}, (\gamma_{ijk})_{\substack{i \in [1,\ell'] \\ j \in [1,n] \\ k \in [1,m]}}, (\gamma'_{ij})_{\substack{i \in [1,\ell'] \\ j \in [1,\ell]}} \right) \in \mathbb{Z}_p \ , \tag{2}$$

such that, $S_i^1 = \sum_{j=1}^n \alpha_{ij} x_j$ *for* $i \in [1, n']$*,* $S_i^2 = \sum_{j=1}^m \beta_{ij} y_j$ *for* $i \in [1, m']$ *and* $S_i^T = \sum_{j=1}^n \sum_{k=1}^m \gamma_{ijk} x_j y_k + \sum_{j=1}^\ell \gamma'_{ij} z_i$ *for* $i \in [1, \ell']$*. We denote the outputs and their representations as* $(S; \vec{S}) \leftarrow\$ \mathcal{A}_{alg}(\vec{\zeta}_1, \vec{\zeta}_2, \vec{\zeta}_T)$*.*

With regards to the representations that the algebraic adversary $\mathcal{A}_{alg}$ can output, we need to provide some additional notations borrowing from Kim et al. [KSAP21]. Let $\mathcal{A}_{alg}$ take the vectors of group elements $\vec{\zeta}_1 = (g_{11}, \ldots, g_{1n}) = (g_1^{x_1}, \ldots, g_1^{x_n}) \in \mathbb{G}_1^n$, $\vec{\zeta}_2 = (g_{21}, \ldots, g_{2m}) = (g_2^{y_1}, \ldots, g_2^{y_m}) \in \mathbb{G}_2^m$ and $\vec{\zeta}_T = (g_{T1}, \ldots, g_{T\ell}) = (g_T^{z_1}, \ldots, g_T^{z_\ell}) \in \mathbb{G}_T^\ell$ as inputs. By Def. 2.3, when $\mathcal{A}_{alg}$ outputs the group elements $S =$

$\big(h_1^1, \ldots, h_{n'}^1, h_1^2, \ldots, h_{m'}^2, h_1^T, \ldots, h_{\ell'}^T\big)$, for each element $h_i^b \in \mathbb{G}_b$, $i \in [1, v']$, $\mathcal{A}_{alg}$ must also output the corresponding representation $\vec{h_i^b} = (h_{i1}^b, \ldots, h_{iv}^b) \in \mathbb{Z}_p^{(v' \cdot v)}$, s.t., $h_i^b = \prod_{j=1}^{v} g_{bj}^{h_{ij}^b}$, where $b = \{1, 2, T\}$, $v = \{n, m, \ell\}$, $v' = \{n', m', \ell'\}$. The element $h_{ij}^b \in \mathbb{Z}_p$ regarding the group element $g_{bj} \in \mathbb{G}_b$ for $j \in [1, v]$ can be denoted by $[h_i^b | g_{bj}]$. If each of $\mathsf{dlog}_{g_b}(g_{bj})$ is represented by a k-variant polynomial in ring $\mathbb{Z}_p[X_{b1}, \ldots, X_{bk}]$ for some $k \in \mathbb{N}$, then we can define a polynomial $P_{h_i^b}[\vec{X}_b]$ to denote $\mathsf{dlog}_{g_b}(h_i^b) = \sum_{j=1}^{v} \big([h_i^b|g_{bj}] \cdot \mathsf{dlog}_{g_b}(g_{bj})\big)$, where $\vec{X}_b = (X_{b1}, \ldots, X_{bk})$. Note that the coefficients of polynomial $P_{h_i^b}[\vec{X}_b]$ are composed of the linear combination of the representation set elements $\vec{h}_i^b$. The equality of $\mathsf{dlog}_{g_b}(h_i^b) = z$, where $z \in \mathbb{Z}_p$, can be shown with a polynomial evaluation of $P_{h_i^b}[\vec{x}_b] = z$ at point $\vec{x}_b = (x_{b1}, \ldots, x_{bk})$, where $x_{bj} \in X_{bj}$. The equality of $P_{h_i^b}[\vec{x}_b] = 0$ means that the polynomial evaluates to 0 at point $\vec{x}_b$, while $P_{h_i^b}[\vec{X}_b] = 0$ means that $P_{h_i^b}[\vec{X}_b]$ is a zero-polynomial and its all coefficients are zero.

2.3 Assumptions

Definition 2.4 (ℓ-One-More Discrete Logarithm Problem [BNPS03]). *Let $(\mathbb{G}, p, g)$ be a cyclic group of prime order p with a generator g. Given a tuple $(g, g^{r_1}, g^{r_2}, \ldots, g^{r_\ell}) \in \mathbb{G}^\ell$, where $r_i \leftarrow\!\$\, \mathbb{Z}_p$ for $i \in [1, \ell]$, and a discrete logarithm oracle $\mathsf{Dlog}_g(g^x) \to x$, for all PPT adversaries $\mathcal{A}$ it is computationally hard to return a tuple $(r'_1, \ldots, r'_\ell) \in \mathbb{Z}_p^\ell$ such that (1) $\{r_i = r'_i\}_{i \in [1,\ell]}$ and (2) the oracle $\mathsf{Dlog}_g(.)$ is queried at most $\ell - 1$ times. We say ℓ-OMDL problem is ε-hard if the success probability of all PPT adversaries $\mathcal{A}$ is bounded by a negligible function ε.*

Intuitively, in the ℓ-OMDL assumption the adversary receives ℓ group elements and is given access to an oracle that computes the discrete logarithm (DL) of any given element with respect to a fixed basis. Eventually, the goal of the adversary is to compute the DL of all ℓ received challenges while making at most $\ell - 1$ calls to the oracle.

Definition 2.5 (Strong Discrete Logarithm (SDL) Assumption [BCN+10]). *Let $\mathcal{BG}(1^\lambda) = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$ is an asymmetric pairing group. Given a pair $(Z_1, Z_2) = (g_1^z, g_2^z)$, where $z \leftarrow\!\$\, \mathbb{Z}_p^*$, for all PPT adversaries, $\mathcal{A}$, it is computationally hard to find z.*

The PS assumption is an interactive assumption, defined by Pointcheval and Sanders [PS16] to construct an efficient randomizable signature. The assumption has been shown to hold in the GGM.

Definition 2.6 (PS Assumption [PS16]). *The PS assumption holds if no PPT adversary, $\mathcal{A}$, who takes asymmetric pairing setup, a tuple $(g_2^x, g_2^y) \in \mathbb{G}_2^2$ and PS oracle $\mathcal{O}^{PS}$ shown in Fig. 1, can find a tuple $(h^*, s^*, m^*) \in \mathbb{G}_1^2 \times \mathbb{Z}_p$ such that (1) $h^* \neq 1_{\mathbb{G}_1}$, $m^* \neq 0$, (2) $s^* = h^{x + ym^*}$, (3) $m^* \notin \mathcal{Q}$, where $\mathcal{Q}$ is the list of queried messages to the $\mathcal{O}^{PS}$ oracle.*

Note that in the PS assumption the validity of tuple (h^*, s^*, m^*) is decidable by checking: $e(s^*, g_2) = e(h^*, g_2^x(g_2^y)^{m^*})$. Kim et al. [KLAP20] introduced a generalized version of the PS assumption, GPS in short, that splits the PS oracle into two: the first oracle provides basis h sampled uniformly at random and the second oracle takes the message and basis h as inputs and generates the PS tuple.

Definition 2.7 (Generalized PS Assumption [KLAP20]). *Given a tuple $(g_2^x, g_2^y) \in \mathbb{G}_2^2$ and two oracles $\mathcal{O}_0^{GPS}$ and $\mathcal{O}_1^{GPS}$ defined in Fig. 1. The GPS assumption holds if no PPT adversary, $\mathcal{A}$, can find a tuple $(h^*, s^*, m^*) \in \mathbb{G}_1^2 \times \mathbb{Z}_p$ such that, (1) $h^* \neq 1_{\mathbb{G}_1},\ m^* \neq 0$, (2) $s^* = {h^*}^{x+ym^*}$, (3) $m^* \notin \mathcal{Q}_1$, where $\mathcal{Q}_1$ is the list of queried messages to $\mathcal{O}_1^{GPS}$ oracle by the adversary.*

Recently, Kim et al. [KSAP21] expanded the GPS assumption in the direction that all exponential values are substituted by group elements, called GPS_2. The security of GPS_2 assumption is proven in the AGM and reduced to the SDL problem.

Definition 2.8 (GPS_2 **Assumption [KSAP21]).** *Given a tuple $(g_2^x, g_2^y) \in \mathbb{G}_2^2$ and two oracles $\mathcal{O}_0^{GPS_2}$ and $\mathcal{O}_1^{GPS_2}$ defined in Fig. 1, the GPS_2 assumption holds if no algebraic adversary, $\mathcal{A}_{alg}$ can find a tuple $(M^*, h^*, s^*, f^*) \in \mathbb{G}_1^4$ such that, (1) $h^*, M^* \neq 1_{\mathbb{G}_1}$, (2) $s^* = {h^*}^x {M^*}^y$, (3) $\mathsf{dlog}_{g_1}(M^*) = \mathsf{dlog}_{h^*}(f^*)$, (4) $(\star, M^*) \notin \mathcal{Q}_1$.*

Connecting to the fact that the challenge in the GPS_2 assumption only contains the source group elements and the verification uses membership testing and pairing product equations, it fails to lead us to an SPS construction because of the unilateral impossibility result shown by [AGHO11]. We take one step ahead and define the GPS_3 assumption that avoids this drawback by relying on indexed Diffie-Hellman message spaces. Additionally, we modify the oracle $\mathcal{O}_0^{\mathrm{GPS}_2}$ such that $\mathcal{O}_0^{\mathrm{GPS}_3}$ oracle generates a basis h by taking an index id as input.

Definition 2.9 (GPS_3 **Assumption).** *Given a tuple $(g_2^x, g_2^y) \in \mathbb{G}_2^2$ and two oracles, $\mathcal{O}_0^{GPS_3}$ and $\mathcal{O}_1^{GPS_3}$ defined in Fig. 1. The GPS_3 assumption holds if no algebraic adversary, $\mathcal{A}_{alg}$, can find a tuple $(h^*, M_1^*, M_2^*, s^*) \in \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$ such that, (1) $h^* \neq 1_{\mathbb{G}_1}$ and $M_2^* \neq 1_{\mathbb{G}_2}$, (2) $s^* = {h^*}^x {M_1^*}^y$, (3) $\mathsf{dlog}_{h^*}(M_1^*) = \mathsf{dlog}_{g_2}(M_2^*)$, (4) $(\star, M_2^*) \notin \mathcal{Q}_1$. The GPS_3 assumption is called ε-hard if the success probability of all PPT adversaries is bounded by a negligible function ε, i.e. $Adv_{\mathcal{A}}^{GPS_3}(\lambda) \leq \varepsilon$.*

The GPS_3 assumption is decidable by checking the equality of two pairing product equations: $e(h^*, M_2^*) = e(M_1^*, g_2)$ and $e(h^*, g_2^x)e(M_1^*, g_2^y) = e(s^*, g_2)$.

$\mathbf{G}^{\mathrm{PS}}(1^\lambda)$

$$
\begin{array}{l}
(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2) \leftarrow \mathcal{BG}(1^\lambda) \\
x \leftarrow\$\, \mathbb{Z}_p \\
y \leftarrow\$\, \mathbb{Z}_p \\
(h^*, s^*, m^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathrm{PS}}}(g_2^x, g_2^y) \\
\textbf{return} \;\; (1)\; h^* \neq 1_{\mathbb{G}_1} \wedge\; m^* \neq 0 \;\wedge \\
\qquad\qquad\;\; (2)\; s^* = {h^*}^{x+ym^*} \;\wedge \\
\qquad\qquad\;\; (3)\; m^* \notin \mathcal{Q} \;\; .
\end{array}
$$

$\mathbf{G}^{\mathrm{GPS}}(1^\lambda)$

$$
\begin{array}{l}
(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2) \leftarrow \mathcal{BG}(1^\lambda) \\
x \leftarrow\$\, \mathbb{Z}_p \\
y \leftarrow\$\, \mathbb{Z}_p \\
(h^*, s^*, m^*) \leftarrow \mathcal{A}^{\mathcal{O}_0^{\mathrm{GPS}}, \mathcal{O}_1^{\mathrm{GPS}}}(g_2^x, g_2^y) \\
\textbf{return} \;\; (1)\; h^* \neq 1_{\mathbb{G}_1} \wedge\; m^* \neq 0 \;\wedge \\
\qquad\qquad\;\; (2)\; s^* = {h^*}^{x+m^*y} \;\wedge \\
\qquad\qquad\;\; (3)\; (\star, m^*) \notin \mathcal{Q}_1 \;\; .
\end{array}
$$

$\mathbf{G}^{\mathrm{GPS}_2}(1^\lambda)$

$$
\begin{array}{l}
(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2) \leftarrow \mathcal{BG}(1^\lambda) \\
x \leftarrow\$\, \mathbb{Z}_p \\
y \leftarrow\$\, \mathbb{Z}_p \\
(M^*, h^*, s^*, f^*) \leftarrow \mathcal{A}_{alg}^{\mathcal{O}_0^{\mathrm{GPS}_2}, \mathcal{O}_1^{\mathrm{GPS}_2}}(g_2^x, g_2^y) \\
\textbf{return} \;\; (1)\; h^* \neq 1_{\mathbb{G}_1} \wedge\; M^* \neq 1_{\mathbb{G}_1} \;\wedge \\
\qquad\qquad\;\; (2)\; s^* = {h^*}^x {f^*}^y \;\wedge \\
\qquad\qquad\;\; (3)\; \mathsf{dlog}_{h^*}(f^*) = \mathsf{dlog}_{g_1}(M^*) \;\wedge \\
\qquad\qquad\;\; (4)\; (\star, M^*) \notin \mathcal{Q}_1 \;\; .
\end{array}
$$

$\mathbf{G}^{\mathrm{GPS}_3}(1^\lambda)$

$$
\begin{array}{l}
(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2) \leftarrow \mathcal{BG}(1^\lambda) \\
x \leftarrow\$\, \mathbb{Z}_p \\
y \leftarrow\$\, \mathbb{Z}_p \\
(h^*, M_1^*, M_2^*, s^*) \leftarrow \mathcal{A}_{alg}^{\mathcal{O}_0^{\mathrm{GPS}_3}, \mathcal{O}_1^{\mathrm{GPS}_3}}(g_2^x, g_2^y) \\
\textbf{return} \;\; (1)\; M_1^* \neq 1_{\mathbb{G}_1} \wedge\; h^* \neq 1_{\mathbb{G}_1} \;\wedge \\
\qquad\qquad\;\; (2)\; s^* = {h^*}^x {M_1^*}^y \;\wedge \\
\qquad\qquad\;\; (3)\; \mathsf{dlog}_{h^*}(M_1^*) = \mathsf{dlog}_{g_2}(M_2^*) \;\wedge \\
\qquad\qquad\;\; (4)\; (\star, M_2^*) \notin \mathcal{Q}_1 \;\; .
\end{array}
$$

$\mathcal{O}^{\mathrm{PS}}(m) //\; m \in \mathbb{Z}_p$

$$
\begin{array}{l}
h \leftarrow\$\, \mathbb{G}_1 \\
\mathcal{Q} = \mathcal{Q} \cup \{m\} \\
\textbf{return}\; (h, h^{x+my})
\end{array}
$$

$\mathcal{O}_0^{\mathrm{GPS}}(.)$

$$
\begin{array}{l}
h \leftarrow\$\, \mathbb{G}_1 \\
\mathcal{Q}_0 = \mathcal{Q}_0 \cup \{h\} \\
\textbf{return}\; h
\end{array}
$$

$\mathcal{O}_0^{\mathrm{GPS}_2}(.)$

$$
\begin{array}{l}
r \leftarrow\$\, \mathbb{Z}_p \\
\mathcal{Q}_0 = \mathcal{Q}_0 \cup \{g_1^r\} \\
\textbf{return}\; g_1^r
\end{array}
$$

$\mathcal{O}_0^{\mathrm{GPS}_3}(id)$

$$
\begin{array}{l}
\textbf{if} \quad \mathcal{Q}_0[id] = \perp: \\
\qquad r \leftarrow\$\, \mathbb{Z}_p \\
\qquad \mathcal{Q}_0[id] \leftarrow g_1^r \\
\textbf{return}\; \mathcal{Q}_0[id]
\end{array}
$$

$\mathcal{O}_1^{\mathrm{GPS}}(h, m) //\; m \in \mathbb{Z}_p$

$$
\begin{array}{l}
\textbf{if}\; (h \notin \mathcal{Q}_0 \;\vee \\
\qquad (h, \star) \notin \mathcal{Q}_1): \\
\quad \textbf{return}\; \perp \\
s \leftarrow h^x M_1^y \\
\mathcal{Q}_1 = \mathcal{Q}_1 \cup \{(h, m)\} \\
\textbf{return}\; h^{x+my}
\end{array}
$$

$\mathcal{O}_1^{\mathrm{GPS}_2}(h, M_1, M_2) \; // M_1, M_2 \in \mathbb{G}_1$

$$
\begin{array}{l}
\textbf{if}\; (h \notin \mathcal{Q}_0 \;\vee \\
\qquad \mathsf{dlog}_h(M_1) \neq \mathsf{dlog}_{g_1}(M_2)): \\
\quad \textbf{return}\; \perp \\
\textbf{if}\; (h, \star) \in \mathcal{Q}_1: \\
\quad \textbf{return}\; \perp \\
\mathcal{Q}_1 = \mathcal{Q}_1 \cup \{(h, M_1)\} \\
\textbf{return}\; h^x M_1^y
\end{array}
$$

$\mathcal{O}_1^{\mathrm{GPS}_3}(h, M_1, M_2) // M_1 \in \mathbb{G}_1, M_2 \in \mathbb{G}_2$

$$
\begin{array}{l}
\textbf{if}\; (h \notin \mathcal{Q}_0 \;\vee \\
\qquad \mathsf{dlog}_h(M_1) \neq \mathsf{dlog}_{g_2}(M_2)): \\
\quad \textbf{return}\; \perp \\
\textbf{if}\; (h, \star) \in \mathcal{Q}_1: \\
\quad \textbf{return}\; \perp \\
\mathcal{Q}_1 = \mathcal{Q}_1 \cup \{(h, M_2)\} \\
\textbf{return}\; h^x M_1^y
\end{array}
$$

Fig. 1. Security Games and Oracles.

GPS_3 Assumption in the Algebraic Group Model. Similar to [KSAP21], we define the GPS_3 in the AGM in Fig. 2. Compared to GPS_3 assumption in Def. 2.9, it has two main differences: 1 An extractor $\mathsf{Ext}(.)$ as a deterministic polynomial algorithm is defined in the second oracle, $\mathcal{O}_1^{\mathrm{GPS}_3}$. In the j^{th}-query, this extractor takes three source group elements $h_j, M_{j1}, M_{j2} \in \mathbb{G}_1^2 \times \mathbb{G}_2$ along with their representations $\vec{h}_j, \vec{M}_{j1}, \vec{M}_{j2}$ as inputs. It then returns a scalar m_j s.t. $M_{j1} = h_j^{m_j}$, $M_{j2} = g_2^{m_j}$, or it returns $\perp$ whenever the extraction is failed. This extractor succeeds to extract the scalar m_j because under some conditions, shown in Fig. 2, if the extraction fails then the SDL problem is no longer hard (we will discuss it in App. B). Thus the oracle $\mathcal{O}_1^{\mathrm{GPS}_3}$ in GPS_3 in the AGM can successfully respond to the algebraic adversary $\mathcal{A}_{alg}$ queries. 2 Additionally, the third condition in GPS_3 assumption of Def. 2.9, can be written as $\mathsf{dlog}_{g_2}(M_2^*) = \mathsf{dlog}_{h^*}(M_1^*) = \frac{\mathsf{dlog}_{g_1}(M_1^*)}{\mathsf{dlog}_{g_1}(h^*)}$ and these conditions can be checked by evaluating the polynomials $P_{M_2}^*(\vec{x}_2) = \mathsf{dlog}_{g_2}(M_2^*) = \frac{\mathsf{dlog}_{g_1}(M_1^*)}{\mathsf{dlog}_{g_1}(h^*)} = \frac{P_{M_1}^*(\vec{x}_1)}{P_h^*(\vec{x}_1)}$, where $\vec{x}_1$ and $\vec{x}_2$ are the vectors of the all points selected by the challenger to $\mathcal{A}_{alg}$ relative to all inputs $\mathcal{A}_{alg}$ received up to that point.

Theorem 2.1. *The* GPS_3 *assumption in the AGM, stated in Fig. 2, holds in the asymmetric algebraic bilinear group model, stated in Def. 2.3, under the hardness of the SDL assumption, stated in Def. 2.5.*

Proof (High-level). To prove this theorem we borrow the Kim et al. proof technique [KSAP21, Theorem 3.6] and define a challenger $\mathcal{B}_{alg}$ who can simulate the defined oracles in the GPS_3 assumption in the AGM. The defined extractor can successfully extract the scalar message m_j on the j^{th} query to $\mathcal{O}_1^{\mathrm{GPS}_3}$ oracle by having access to the representations of inputs to the oracle. As a contradiction, if the extractor fails then we can build an algebraic algorithm to solve the SDL problem and we conclude that $\mathcal{B}_{alg}$ can successfully simulate the security game. Then we demonstrate that no PPT algebraic adversary $\mathcal{A}_{alg}$ can output a valid challenge that satisfies all the conditions in the security game $\mathbf{G}_{\mathcal{A}_{alg}}^{\mathrm{GPS}_3}(1^\lambda)$ in Fig. 2. Note that GPS_3 assumption is stronger than GPS_2 assumption since the adversary should represent the message in both source groups. A formal proof of this theorem can be found in App. B.1. $\square$

2.4 Distributed Key Generation

Distributed Key Generation (DKG) protocols [Ped92] enables to securely generate key-pairs among parties without the need of a trusted dealer. To perform a secret key related operation, a cooperation of sufficiently large number of parties is required while any smaller subset is unable to learn any information about the secret key. The generated keys can be used for any threshold cryptosystem e.g. threshold signatures or threshold encryption. We formally define the DKG construction and list security requirements.

$\mathbf{G}^{\mathrm{GPS}_3}_{\mathcal{A}_{alg}}(1^\lambda)$

Run $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2) \leftarrow \mathcal{BG}(1^\lambda)$
$x \leftarrow\!\$\, \mathbb{Z}_p := \mathbf{X}, y \leftarrow\!\$\, \mathbb{Z}_p := \mathbf{Y}$
$\Big((h^*; \vec{h}^*), (M_1^*; \vec{M}_1^*), (M_2^*; \vec{M}_2^*), (s^*; \vec{s^*})\Big) \leftarrow \mathcal{A}_{alg}^{\mathcal{O}_0, \mathcal{O}_1}(g_2^x, g_2^y)$
If (1) $M_1^*, h^* \neq 1_{\mathbb{G}_1} \;\wedge\; M_2^* \neq 1_{\mathbb{G}_2} \;\wedge$
(2) $s^* = {h^*}^{x} {M_1^*}^{y} \;\wedge$
(3) $\mathsf{dlog}_{h^*}(M_1^*) = \mathsf{dlog}_{g_2}(M_2^*)$
$\leftrightarrow \boxed{P^*_{M_2}(\vec{x}_2) P^*_h(\vec{x}_1) - P^*_{M_1}(\vec{x}_1) = 0}$
(4) $\wedge\; (\star, M_2^*) \notin \mathcal{Q}_1$:
return 1
Else : **return** 0

$\mathsf{Ext}\Big((M_{j1}; \vec{M}_{j1}), (M_{j2}; \vec{M}_{j2})\Big)$

$//\vec{X}_1 = (\mathbf{X}, \mathbf{Y}, \mathbf{R}_1, \ldots, \mathbf{R}_{|\mathcal{Q}_0|})$
$//\vec{X}_2 = (\mathbf{X}, \mathbf{Y})$
If $(P_{M_{j1}}[\vec{X}_1] - \mathbf{R}_j P_{M_{j2}}[\vec{X}_2]) \neq 0$:
return $\perp$
Else :
return $[M_{j2} | g_2]$

$\mathcal{O}_0^{\mathrm{GPS}_3}(id_{m_j})$ //j-th query

If $\mathcal{Q}_0[id_{m_j}] = \perp$:
$r_j \leftarrow\!\$\, \mathbb{Z}_p := \mathbf{R}_j$
$\mathcal{Q}_0[id_{m_j}] \leftarrow g_1^{r_j} := h_j$
return $\mathcal{Q}_0[id_{m_j}]$

$\mathcal{O}_1^{\mathrm{GPS}_3}\Big((h_j; \vec{h}_j), (M_{j1}; \vec{M}_{j1}), (M_{j2}; \vec{M}_{j2})\Big)$//j-th query

If $h_j \notin \mathcal{Q}_0 \;\vee\; (h_j, \star) \in \mathcal{Q}_1$:
return $\perp$
For $r_j = \mathsf{dlog}_{g_1}(h_j$:
If $e(M_{j1}, g_2) \neq e(g_1^{r_j}, M_{j2})$:
return $\perp$
Else : $m_j \leftarrow \mathsf{Ext}\Big((M_{j1}; \vec{M}_{j1}), (M_{j2}; \vec{M}_{j2})\Big)$
$s_j = h_j^x M_{j1}^y = h_j^{x + m_j y}$
$\mathcal{Q}_1 = \mathcal{Q}_1 \cup \{(h_j, M_{j2})\}$
return s_j

Fig. 2. GPS_3 Assumption in the AGM.

Definition 2.10 (Distributed Key Generation (DKG) [Ped92]). *A (n, t)-DKG is an interactive protocol among a set of parties $(P_1, \ldots, P_n)$ to generate a tuple of public keys $(\mathsf{pk}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ along with a tuple of secret key shares $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ s.t. only party P_i knows share sk_i. A DKG is called $(t-1)$-Consistent if at the end of the protocol honest parties agree on a consistent public key pk and the vector of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ even if at most $t-1$ of the parties are corrupted. Moreover as long as the number of corrupted parties is less than t, there exists a polynomial of $f(x) \in \mathbb{Z}_p^{\leq t}[X]$ with degree $d \leq t$ such that we have: $\forall i \in [1, n]$: $\mathsf{sk}_i = f(i)$ and the global public key can be computed as $\mathsf{pk} = g^{f(0)}$, where g is the generator of a cyclic group $\mathbb{G}$.*

Next we recall the definition of oracle-aided algebraic simulatability from [BL22]. Informally, a DKG is called oracle-aided algebraic simulatable if there exists an efficient simulator Sim given some number of queries to a discrete logarithm oracle can simulate an execution of the DKG protocol with adaptive corruption. (the adversary can corrupt up to $t-1$ parties at any point of the experiment.)

Definition 2.11 ($(t-1,k)$-Oracle-Aided Algebraic Simulatability [BL22]). *A (n,t)-DKG protocol is called $(t-1,k)$-Oracle-Aided Algebraic Simulatability (OAAS) secure, if for all algebraic adversaries, $\mathcal{A}_{alg}$, that can corrupt at most $t-1$ parties, there exists an algebraic PPT simulator* Sim *that can make less than $k-1$ queries to a discrete logarithm oracle $\mathsf{Dlog}_g(.)$ and satisfy the following properties:*

- *In an adaptive corruption notion, for a given k-OMDL instance $\zeta = (\zeta_1, \ldots, \zeta_k) = (g^{r_1}, \ldots, g^{r_k})$, the algebraic simulator* Sim *can play the role of honest parties and it can successfully generates the global public key g^x with an algebraic representation $\vec{g^x} = (\tilde{a}_0, a_{0,1}, \ldots, a_{0,k})$, i.e. $g^x = g^{\tilde{a}_0} \prod_{j=1}^{k} \zeta_j^{a_{0,j}}$.*
- *For a given k-OMDL instance $\zeta = (\zeta_1, \ldots, \zeta_k) = (g^{r_1}, \ldots, g^{r_k})$, let for $i \in [1, k-1]$, $g_i \in \mathbb{G}$ denotes the i^{th} query to the discrete logarithm oracle $\mathsf{Dlog}_g(.)$ and $\vec{g_i} = (\tilde{a}_i, a_{i,1}, \ldots, a_{i,k})$ is the corresponding algebraic representation of g_i, i.e. $g_i = g^{\tilde{a}_i} \prod_{j=1}^{k} \zeta_j^{a_{i,j}}$. If* Sim *completes the simulation of an execution of the DKG, then the following Simulatability matrix is invertible over $\mathbb{Z}_p$.*

$$L := \begin{pmatrix} a_{0,1} & a_{0,2} & \cdots & a_{0,k} \\ a_{1,1} & a_{1,2} & \cdots & a_{1,k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k-1,1} & a_{k-1,2} & \cdots & a_{k-1,k} \end{pmatrix} \in \mathbb{Z}_p^{k \times k} \tag{3}$$

- *For all global public parameter $g^x \in \mathbb{G}$, the view of adversary $\mathcal{A}_{alg}$ when interacting with the algebraic simulator* Sim *on input ζ and the case that it is interacting with all honest parties are identically distributed.*

The minimum $k \in \mathbb{N}$ s.t. a (n,t)-DKG is $(t-1,k)$-OAAS secure is called the DKG's simulatability factor.

3 Message-Indexed Structure-Preserving Signatures

We give a high-level overview of our threshold-friendly SPS taking Ghadafi's construction (see App. A) as a starting point and show how it fails for building a non-interactive SPTS. Assume that the distributed key generation phase for secret keys x and y has already been completed: each signer $i \in [1,n]$ has access to a shared secret signing key $\mathsf{sk}_i = (\mathsf{sk}_{i1}, \mathsf{sk}_{i2})$, and the cooperation of at least t signers is required to obtain a valid aggregated signature. In a threshold variant of Ghadafi's SPS, in order to sign a Diffie-Hellman message $(M_1, M_2) \in \mathcal{M}_{\mathsf{DH}}$, each signer generates the partial signature σ_i, which consists of three elements $(R_i := g_1^{r_i}, S_i := M_1^{r_i}, T_i := R_i^{\mathsf{sk}_{i1}} S_i^{\mathsf{sk}_{i2}})$ with fresh randomness $r_i \leftarrow\$\, \mathbb{Z}_p$. To reconstruct the set of partial signatures from a list of signers $\mathcal{T}$, one can compute the Lagrange coefficient $L_i^{\mathcal{T}}(0)$ and obtain the first and second elements of the aggregated signature by computing the products, $R = \prod_{i\in\mathcal{T}} R_i^{L_i^{\mathcal{T}}(0)} = g_1^r$ and $S = \prod_{i\in\mathcal{T}} S_i^{L_i^{\mathcal{T}}(0)} = M_1^r$, where we denote $r = \sum_{i\in\mathcal{T}} r_i L_i^{\mathcal{T}}(0)$. One would expect to have $T = R^x S^y$. However, T cannot be computed as $\prod_{i\in\mathcal{T}} T_i^{L_i^{\mathcal{T}}(0)}$ and there is no efficient linear computation to reconstruct the secrets x and y given that the shares are multiplied by a distinct random integer. Therefore to overcome this challenge we follow the message indexing technique introduced by Sonnino et al. [SAB+19] and Camenisch et al. [CDL+20].

Indexing can be understood as requiring the existence of an injective function F that maps each message to an index. Then the partial signers of the message evaluate a hash-to-curve function H on the index to agree on a single basis $R = \mathsf{H}(id)$.

In our construction we will rename R to h to make explicit that it is a random basis of $\mathbb{G}_1$. That is, its discrete logarithm $\mathsf{dlog}_{g_1}(h)$ is unknown due to modeling H as a random oracle. As a consequence, we can no longer compute M_1^r without knowledge of the discrete logarithm $\mathsf{dlog}_h(M_1)$.

We overcome this problem by switching from a DH message space to an indexed DH message space $\mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}$, as defined in Def. 2.2.

This results in an apparent circularity with respect to the indexing technique. On the one hand, we require existence of an injective function F that maps (M_1, M_2) to id, on the other hand M_1 is computed as $M_1 = \mathsf{H}(id)^{\log_{g_2}(M_2)}$. This circularity is avoided by computing id from a partial message, or it's discrete logarithm.

$$\underbrace{m \in \mathbb{Z}_p \xrightarrow{f} id}_{\text{Message Indexing}} \overbrace{\xrightarrow{\mathsf{iDH}^{\mathsf{H}}(m,id)} (id, M_1, M_2) \in \mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}}^{\text{Indexed DH message space in ROM}}$$

Fig. 3. Towards a Message-Indexed SPS.

As illustrated in Fig. 3, to satisfy all conditions of the indexed DH message space we start from a scalar message and use an indexing function f to assign an index to each scalar message $m \in \mathbb{Z}_p$. In the next step, we use a hash-to-curve function $\mathsf{H} : \{0,1\}^* \to \mathbb{G}_1$ (modeled as a random oracle) to generate a unique basis h. Then the source group messages can be obtained using h. In an indexed-message SPS, the signing algorithm takes the source group messages of this form along with an index as inputs and then generates the underlying signature with access to hash-to-curve function $\mathsf{H}(.)$. Note that the index does not destroy the structure since the verifier does not need to know the index to verify a signature σ on message $\tilde{M} := (M_1, M_2)$.

$\mathsf{iDH}^{\mathsf{H}}(id, m)$	$\mathsf{H}(id)$
$h \leftarrow \mathsf{H}(id)$	**If** $\mathcal{Q}_{\mathsf{H}}[id] = \perp$:
$M_1 = h^m$	$\quad r \leftarrow\$ \mathbb{Z}_p$
$M_2 = g_2^m$	$\quad \mathcal{Q}_{\mathsf{H}}[id] \leftarrow g_1^r := h$
return $(id, M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$	**return** $\mathcal{Q}_{\mathsf{H}}[id]$

Fig. 4. Indexed Diffie-Hellman Message Space in the Random Oracle Model.

We adapt the notion of EUF-CMA security (See App. A) to existential unforgeability against chosen indexed-message attacks, denoted by EUF-CiMA. In this security notion, the adversary can make queries to the signing oracle by providing index/message pairs.

Definition 3.1 (Existential Unforgeability under Chosen indexed Message Attack (EUF-CiMA)). *For a given asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$ a digital signature over the indexed Diffie-Hellman message space, $\mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$, is EUF-CiMA-secure, if for all PPT adversaries $\mathcal{A}$ the advantage on wining the defined security game defined in Fig. 5 is negligible, i.e.,*

$$Adv_{\mathbf{DS},\mathcal{A}}^{\mathit{EUF\text{-}CiMA}}(\lambda) := \Pr\left[\mathbf{G}_{\mathcal{A}}^{\mathsf{EUF-CiMA}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda) \ .$$

$\mathbf{G}_{\mathcal{A}}^{\text{EUF-CiMA}}(1^\lambda)$	$\mathcal{O}_{\mathsf{Sign}}(id, M_1, M_2)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$	**if** $(id, \star) \in \mathcal{Q}$:
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$	$\quad$ **return** $\perp$
$(M_1^*, M_2^*, \sigma^*) \leftarrow\$ \mathcal{A}^{\mathsf{H}, \mathcal{O}_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{vk})$	**Else** : $\sigma \leftarrow \mathsf{Sign}^{\mathsf{H}}(\mathsf{pp}, \mathsf{sk}, id, M_1, M_2)$
return $(\star, M_2^*) \notin \mathcal{Q} \ \wedge$	$\quad \mathcal{Q} \leftarrow \mathcal{Q} \cup \{(id, M_2)\}$
$\quad \mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, M_1^*, M_2^*, \sigma^*) = 1$	$\quad$ **return** σ

Fig. 5. Game $\mathbf{G}_{\mathcal{A}}^{\text{EUF-CiMA}}(1^\lambda)$.

A message-indexed SPS construction. In Fig. 6, we present our message-indexed SPS construction over the indexed Diffie-Hellman message space $\mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$. We then show that this construction is EUF-CiMA-secure under the hardness of the GPS_3 assumption, stated in Def. 2.9. Note that the signer has access to the indexed Diffie-Hellman message space in the ROM, described in Fig. 4.

$(\mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$

Parse (1^λ)
$(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2) \leftarrow \mathcal{BG}(1^\lambda)$
$\mathsf{pp} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$
return (pp)

$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$

Parse (pp)
$x, y \leftarrow\!\$\, \mathbb{Z}_p^*$
$\mathsf{sk} := (\mathsf{sk}_1, \mathsf{sk}_2) := (x, y)$
$\mathsf{vk} := (\mathsf{vk}_1, \mathsf{vk}_2) := (g_2^x, g_2^y)$
return $(\mathsf{sk}, \mathsf{vk})$

$(\sigma, \perp) \leftarrow \mathsf{Sign}^{\mathsf{H}}(\mathsf{pp}, \mathsf{sk}, id, M_1, M_2)$

Parse $(\mathsf{pp}, \mathsf{sk})$
$h \leftarrow\!\$\, \mathsf{H}(id)$
if $e(h, M_2) = e(M_1, g_2)$:
  return $\sigma := (h, s) = (h, h^x M_1^y)$
Else : **return** $\perp$

$(0, 1) \leftarrow \mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, M_1, M_2, \sigma)$

Parse $(\mathsf{pp}, \mathsf{vk}, \sigma)$
return $\big(h \neq 1_{\mathbb{G}_1} \wedge M_1 \neq 1_{\mathbb{G}_1} \wedge$
$e(h, M_2) = e(M_1, g_2) \wedge$
$e(h, \mathsf{vk}_1) e(M_1, \mathsf{vk}_2) = e(s, g_2)\big)$

Fig. 6. A Message-Indexed SPS Construction.

Theorem 3.1. *The message-indexed structure-preserving signature in Fig. 6 is correct and* EUF-CiMA*-secure, stated in Def. 3.1, under the hardness of* GPS_3 *assumption, stated in Def. 2.9.*

Proof (High-level). To informally demonstrate the proof of this theorem we have:

Correctness. If $e(h, M_2) = e(M_1, g_2) = e(g_1^r, g_2^m) = e(g_1^{rm}, g_2) = e(M_1, g_2)$, then the signature can be written as, $\sigma := (h, s) = (h, h^{x+my})$ and we have, $e(h, \mathsf{vk}_1) e(M_1, \mathsf{vk}_2) = e(h, g_2^x) e(h^m, g_2^y) = e(h, g_2)^{x+my} = e(h^{x+my}, g_2) = e(h^x M_1^y, g_2) = e(s, g_2)$.

EUF-CiMA Security. Over an indexed Diffie-Hellman message space, the random oracle, $\mathsf{H}(.)$, and the signing oracles in the EUF-CiMA security definition are exactly the same as the $\mathcal{O}_0^{\mathrm{GPS}_3}$ and $\mathcal{O}_1^{\mathrm{GPS}_3}$ oracles in the GPS_3 assumption, respectively, except the fact that the signing oracle takes the index id instead of the basis h as input. The challenger given the index id, queries $\mathcal{O}_0^{\mathrm{GPS}_3}$ oracle to obtain the corresponding basis h and then runs the oracle $\mathcal{O}_1^{\mathrm{GPS}_3}$ under the received queries. Thus we can conclude the security of the proposed MI-SPS construction is equivalent to the GPS_3 assumption. More precisely, under the existence of a PPT adversary, $\mathcal{A}$, against the EUF-CiMA security of the proposed MI-SPS, we can build a PPT algorithm $\mathcal{B}$ that can use $\mathcal{A}$ as a subroutine to break the hardness of GPS_3 problem. Thus as a contradiction, we conclude the proposed MI-SPS scheme is EUF-CiMA-secure. Note that the public parameters and oracles are identical and it is easy to show that the success probability of $\mathcal{A}$ and $\mathcal{B}$ is the same. □

Indexing function instantiations. In the indexed Diffie-Hellman message space, the indexing function can be instantiated depending on the requirements of applications: (1) message and the signature are public, (2) message and signature are hidden as for instance in anonymous credential systems. In the first case, the indexing function can be simply instantiated by defining the scalar message as the index, i.e. for each scalar message $m \in \mathbb{Z}_p$, we have $id := m \leftarrow f(m)$. Although it does not meet the SPS conditions of signing algorithm taking scalar message as input, it does not break the main requirements as we are not signing the scalar messages and verification does not need the index.

In the second use-cases, Sonnino et al. in Coconut [SAB$^+$19] took a different approach and commits to the scalar message along with a proof of well-formedness of the commitment. Meanwhile Coconut's authors do not provide a security model to analyze the security of their construction. Recently, Rial and Piotrowska [RP22] did a security analysis on Coconut and have shown that this scheme with some modification, called Coconut^{++} in [BSKD22], which they call PS signature in the ROM, can be provably secure. Camenisch et al. [CDL$^+$20] took a different observation for indexing the messages and instead of generating the basis by a function, they assume the existence of a pre-defined and publicly available indexing function. More precisely, there is a unique index value for each message that are known to each signer. The corresponding basis can be obtained by evaluating the hash-to-curve function at the given index. The authors note that if the size of the message space is in polynomial and known in advance, then this approach is secure since this is equivalent to including this unique basis in the public parameters. However, for most message spaces encountered in practice this is impractical.

4 Structure-Preserving Threshold Signature

In this section, we define the syntax and security notions of non-interactive (n, t)-SPTS schemes and then propose an efficient instantiation. Generally, in a (n, t)-SPTS, the signing key is distributed among n parties and the generation of any signature requires the cooperation of a subset of players of size at least t. Moreover, any adversary who learns $t-1$ or fewer partial signatures cannot forge signatures.

4.1 Definition and Security Requirements

Definition 4.1 (Structure-Preserving Threshold Signature). *For a given security parameter λ and asymmetric bilinear group $\mathcal{BG}(1^\lambda)$, a (n, t)-SPTS over message space $\mathcal{M}$ consists of a tuple of* $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Par\text{-}Sign}, \mathsf{Par\text{-}Verify}, \mathsf{Reconst}, \mathsf{Verify})$ *PPT algorithms defined as follows:*

- $(\mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$*: The setup algorithm takes the security parameter 1^λ as input and returns the public parameters* pp *as output.*
- $(\vec{\mathsf{sk}}, \vec{\mathsf{vk}}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}, n, t)$*: The key generation as a probabilistic algorithm takes the public parameters* pp *and two integers $t, n \in \mathsf{poly}(1^\lambda)$ such that $1 \leq t \leq n$ as inputs.*

It returns two vectors of size n of signing/verification keys $\vec{\mathsf{sk}} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ and $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ such that each party P_i for $i \in [n]$ receives a pair of $(\mathsf{sk}_i, \mathsf{vk}_i)$ along with a global verification key vk while the master secret key, sk, keeps hidden.

- $(\sigma_i) \leftarrow \mathsf{Par\text{-}Sign}(\mathsf{pp}, \mathsf{sk}_i, M)$*: The partial signing algorithm takes the public parameters pp, the i^{th} secret signing key sk_i and a message $M \in \mathcal{M}$ as inputs and returns the partial signature σ_i as output.*
- $(0,1) \leftarrow \mathsf{Par\text{-}Verify}(\mathsf{pp}, \mathsf{vk}_i, M, \sigma_i)$*: The partial verification algorithm is a deterministic algorithm that takes the i^{th} verification key vk_i, message $M \in \mathcal{M}$ and partial signature σ_i as inputs. If σ_i is a valid partial signature, it returns 1, otherwise it responds by 0. We refer to well-formed partial signatures as those that pass this verification.*
- $(\sigma, \perp) \leftarrow \mathsf{Reconst}(\mathsf{pp}, \{i, \sigma_i\}_{i \in \mathcal{T}})$*: The reconstruction algorithm takes public parameters and a set of well-formed partial signatures $\{i, \sigma_i\}$ over subset $\mathcal{T} \subseteq \{1, \ldots, n\}$ as inputs. It outputs an aggregated signature σ if $|\mathcal{T}| \geq t$, else it returns $\perp$.*
- $(0,1) \leftarrow \mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, M, \sigma)$*: This deterministic algorithm takes the verification key vk, message $M \in \mathcal{M}$ and an aggregated signature σ as inputs. It outputs either 1 (accept) or 0 (reject).*

Two main security properties for a SPTS are: Correctness and threshold existentially unforgeable against chosen indexed message attack (Threshold EUF-CiMA) defined as follows.

Definition 4.2 (Correctness). *A (n,t)-SPTS scheme, Ψ_{SPTS}, is called correct, if we have:*

$$\Pr\left[\begin{matrix}\forall\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\vec{\mathsf{sk}}, \vec{\mathsf{vk}}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}, n, t), M \in \mathcal{M}, |\mathcal{T}| \geq t : \\ \mathsf{Verify}\left(\mathsf{pp}, \mathsf{vk}, M, \mathsf{Reconst}\left(\mathsf{pp}, \{\mathsf{Par\text{-}Sign}(\mathsf{pp}, \mathsf{sk}_i, M)\}_{i \in \mathcal{T}}\right)\right) = 1\end{matrix}\right] \geq 1 - \mathsf{negl}(\lambda) \ .$$

We define the threshold unforgeability of non-interactive SPTS constructions over the indexed Diffie-Hellman message spaces in the adaptive corruption setting in the following definition. Throughput, for a given set of players with indices $\mathcal{P} = \{1, \ldots, n\}$, the evolving sets of corrupted and honest parties are denoted by $\mathcal{C} \subset \mathcal{P}$ and $\mathcal{H} = \mathcal{P} \setminus \mathcal{C}$, respectively, such that $\mathcal{C}$ is initialized with an empty list. We assume there exists a challenger $\mathcal{B}$ that plays the role of honest parties.

Definition 4.3 ((ε, q_h, q_s)-Threshold EUF-CiMA). *For a given bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$, a non-interactive (n,t)-SPTS scheme, Ψ_{SPTS}, over the indexed Diffie-Hellman message space $\mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}$, is called (ε, q_h, q_s)-Threshold EUF-CiMA-secure, if for all adaptive PPT adversaries $\mathcal{A}$ we have a negligible advantage in the following security game.*

1. ***Initialization:*** *The game starts with the key generation phase and at the end of this phase, $\mathcal{A}$ has access to the set of public parameters including the verification keys of whole players P_i for $i \in [1, n]$, i.e. $\vec{\mathsf{vk}} = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}$, and the global verification key vk. Additionally, it initializes an empty sets of $\mathcal{Q}_{\mathsf{H}} = \emptyset$, $\mathcal{Q}_{\mathsf{S}} = \emptyset$ and $\mathcal{S}_{(id, M_2)} = \emptyset$ for all messages $(M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}$.*

2. ***Query:*** *On polynomially bounded number of queries, the adversary $\mathcal{A}$ has access to the random oracle, corruption and the partial signing oracles as follows:*
   - *Random oracle,* $\mathsf{H}(id)$*: It takes the index id as input and if* $\mathcal{Q}_\mathsf{H}[id] = \perp$*, it samples* $r \leftarrow\!\$\, \mathbb{Z}_\mathsf{p}^*$ *and sets* $\mathcal{Q}_\mathsf{H}[id] \leftarrow g_1^r$*. It returns* $\mathcal{Q}_\mathsf{H}[id]$ *as output.*
   - *Corruption oracle,* $\mathcal{O}_C(j)$*: As an adaptive notion of security,* $\mathcal{A}$ *at any point of experiment can corrupt up to* $t-1$ *parties. By querying to this oracle and submitting a party identifier* $j \in \mathcal{P}$*,* $\mathcal{A}$ *receives the internal state of* P_j*. It updates* $\mathcal{C} = \mathcal{C} \cup \{j\}$ *and* $\mathcal{H} = \mathcal{H} \setminus \{j\}$ *s.t.* $|\mathcal{C}| < t$*.*
   - *Partial Signing oracle,* $\mathcal{O}_{\mathsf{PSign}}(k, M)$*:* $\mathcal{A}$ *queries* $M := (id, M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$ *along with an honest party identifier* $k \in \mathcal{H}$ *to this oracle. If* $(id, \star) \in \mathcal{Q}_\mathsf{S}$ *it returns* $\perp$*, else it executes* $(\sigma_k) \leftarrow \mathsf{Par\text{-}Sign}(\mathsf{pp}, \mathsf{sk}_k, M)$ *and returns* σ_k *back to* $\mathcal{A}$*. It also updates* $\mathcal{S}_{(id,M_2)} = \mathcal{S}_{(id,M_2)} \cup \{k\}$ *and* $\mathcal{Q}_\mathsf{S} = \mathcal{Q}_\mathsf{S} \cup \{(id, M_2)\}$*.*
3. ***Forgery:*** *In this phase, the adversary* $\mathcal{A}$ *returns a forged signature* σ^* *on challenge message* $\tilde{M}^* := (M_1^*, M_2^*)$ *to the challenger.*
4. ***Probability:*** *The adversary has the following advantage to win this security game:*

$$Adv_{\mathcal{A}}^{\mathsf{TSPS}}(\lambda) = \Pr\begin{bmatrix}(\mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda), (\vec{\mathsf{sk}}, \vec{\mathsf{vk}}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}, n, t), \\ (M_1^*, M_2^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{H}, \mathcal{O}_\mathsf{C}, \mathcal{O}_{\mathsf{PSign}}}(\mathsf{pp}, \vec{\mathsf{vk}}, \mathsf{vk}) : \\ \mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, M_1^*, M_2^*, \sigma^*) = 1 \wedge |\mathcal{S}_{(\star, M_2^*)} \cup \mathcal{C}| < t\end{bmatrix}.$$

We say a (n, t)*-SPTS scheme is* $(\varepsilon, q_\mathsf{H}, q_S)$*-threshold* EUF-CiMA*-secure if for all PPT adversaries* $\mathcal{A}$*, querying at most* q_H *and* q_S *numbers of queries to the random oracle and partial signing oracle, respectively, we have* $Adv_{\mathcal{A}}^{\mathsf{TSPS}}(\lambda) \leq \varepsilon$*.*

Intuitively, this security notion guarantees no adaptive PPT adversary, $\mathcal{A}$, that has corrupted at most $t-1$ signers and has access to a partial signing oracle to ask a polynomially bounded number of queries to the partial signature from at least $n-t+1$ honest signers on messages and indices of its choice cannot produce signature on a fresh message that is not queried to the partial signing oracle before. There is a weaker notion, called static corruption, which implies that the adversary should provide the list of corrupted parties, $\mathcal{C}$, at the initialization phase and cannot alter it later.

4.2 The Proposed Message-Indexed SPTS Construction

For a given security parameter λ, the proposed (n, t)-SPTS construction over indexed Diffie-Hellman message space, $\mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$, defined in Def. 2.2, includes the following PPT algorithms:

- $(\mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$: It takes the security parameter 1^λ as input and runs the asymmetric bilinear group generator $\mathcal{BG}(1^\lambda) = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$, where $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are the generators. It returns the public parameters $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$ as output.

– $(\vec{\mathsf{vk}}, \vec{\mathsf{sk}}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}, n, t)$: It takes pp and integers $t, n \in \mathsf{poly}(\lambda)$ such that $1 \leq t \leq n$ as inputs. It runs the Pedersen's DKG protocol [GJKR99,GJKR03] and acts as follows:
  1. Each player P_i for $i \in [1, n]$, samples two initial random integers $x_{i0}, y_{i0} \leftarrow\$ \mathbb{Z}_p^*$ and does the following steps:
     a) It samples $2t$ random integers $\{x_{ij}, y_{ij}\}_{j=1}^{t}$ and forms the polynomials $F_i[X] = x_{i0} + x_{i1}X + \ldots + x_{it}X^t \in \mathbb{Z}_p^t[X]$ and $G_i[X] = y_{i0} + y_{i1}X + \ldots + y_{it}X^t \in \mathbb{Z}_p^t[X]$ with degree t and commits the coefficients by publishing, $V_{xij} = g_2^{x_{ij}}, V_{yij} = g_2^{y_{ij}}\ \forall j \in [0, t]$.
     b) It sends $F_i(\ell)$ and $G_i(\ell)$ to ℓ^{th} player, P_ℓ, s.t. $\ell \in [1, n] \setminus \{i\}$ and keeps $F_i(i)$ and $G_i(i)$ by own.
  2. Player P_i checks the consistency of the received shares, $F_\ell(i), G_\ell(i)$, from player P_ℓ by computing the equations $g_2^{F_\ell(i)} = \prod_{j=0}^{t} V_{x\ell j}^{i^j}$ and $g_2^{G_\ell(i)} = \prod_{j=0}^{t} V_{y\ell j}^{i^j}$. If these equations hold, player P_i accepts the shares, otherwise it rejects and then complaints against the faulty player P_ℓ.
  3. Any faulty party that received at least t complaints is called disqualified while at the end of this phase at least t parties from the set of qualified players, $\mathcal{Q} \subset \{1, \ldots, n\}$ do the next steps.
  4. The global verification key is determined as $\mathsf{vk} := (\mathsf{vk}_1, \mathsf{vk}_2) := (\prod_{i\in\mathcal{Q}} V_{xi0}, \prod_{i\in\mathcal{Q}} V_{yi0}) = (g_2^{\sum_{i\in\mathcal{Q}} x_{i0}}, g_2^{\sum_{i\in\mathcal{Q}} y_{i0}}) = (g_2^x, g_2^y)$. Additionally, we denote the corresponding secret key by $\mathsf{sk} := (\mathsf{sk}_1, \mathsf{sk}_2) := (x, y) := (\sum_{i\in\mathcal{Q}} x_{i0}, \sum_{i\in\mathcal{Q}} y_{i0})$.
  5. Each qualified player P_i defines its private key share $\mathsf{sk}_i := (\mathsf{sk}_{i1}, \mathsf{sk}_{i2}) := (\sum_{\ell\in\mathcal{Q}} F_\ell(i), \sum_{\ell\in\mathcal{Q}} G_\ell(i))$.
  6. The corresponding verification key vk_i is obtained by computing, $\mathsf{vk}_i := (\mathsf{vk}_{i1}, \mathsf{vk}_{i2}) := (\prod_{\ell\in\mathcal{Q}} \prod_{j=0}^{t} (V_{x\ell j})^{i^j}, \prod_{\ell\in\mathcal{Q}} \prod_{j=0}^{t} (V_{y\ell j})^{i^j}) := (g_2^{F(i)}, g_2^{G(i)})$, where $F[X] = \sum_{\ell\in\mathcal{Q}} F_\ell[X]$ and $G[X] = \sum_{\ell\in\mathcal{Q}} G_\ell[X]$.
  7. For any disqualified parties P_j s.t. $j \in \{1, \ldots, n\} \setminus \mathcal{Q}$ we define $\mathsf{sk}_j = (0, 0)$ and corresponding verification key $\mathsf{vk}_j = (1_{\mathbb{G}_2}, 1_{\mathbb{G}_2})$.

  The key generation phase completes by returning two vectors of size n of signing keys $\vec{\mathsf{sk}} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ and verification keys $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ along with a global verification key vk.

– $(\sigma_i, \perp) \leftarrow \mathsf{Par\text{-}Sign}(\mathsf{pp}, \mathsf{sk}_i, M)$: The partial signature algorithm is run by a qualified party $P_i \in \mathcal{Q}$ and takes pp, sk_i and message $M := (id, M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$ as inputs. It executes $h \leftarrow \mathsf{H}(id)$ and if $e(h, M_2) = e(M_1, g_2)$, then generates the partial signature as $\sigma_i = (h, s_i) = (h, h^{\mathsf{sk}_{i1}} M_1^{\mathsf{sk}_{i2}})$ and returns σ_i as output, otherwise it responds by $\perp$.

- $(0, 1) \leftarrow \mathsf{Par\text{-}Verify}(\mathsf{pp}, \mathsf{vk}_i, \tilde{M}, \sigma_i)$: Given message pair $\tilde{M} = (M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}$, a partial signature σ_i and partial verification key vk_i as inputs, it checks the membership of $M_1, s_i \in \mathbb{G}_1$, $h \neq 1_{\mathbb{G}_1}$ and $M_2 \neq 1_{\mathbb{G}_2}$. Then it computes and checks the pairing product equations, $e(h, M_2) = e(M_1, g_2)$ and $e(h, \mathsf{vk}_{i1}) e(M_1, \mathsf{vk}_{i2}) = e(s_i, g_2)$. If all these conditions hold, then the partial verification algorithm outputs 1 (accept), else it returns 0 (reject).

– $(\sigma, \perp) \leftarrow \mathsf{Reconst}(\mathsf{pp}, \{i, \sigma_i\}_{i \in \mathcal{T}})$: To obtain an aggregated signature, this algorithm takes a set of partial signatures $\{\sigma_i\}_{i \in \mathcal{T}}$ and if $|\mathcal{T}| < t$, it returns $\perp$. Otherwise, it computes $\sigma := (h, s) := \left(h, \prod_{i \in \mathcal{T}} s_i^{L_i^{\mathcal{T}}(0)}\right)$, where $L_i^{\mathcal{T}}(0)$ is the Lagrange coefficient for the i^{th} index corresponding to set $\mathcal{T}$ and returns σ as output.
– $(0, 1) \leftarrow \mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \tilde{M}, \sigma)$: The verification algorithm takes a message pair $\tilde{M} := (M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}$, an aggregated signature σ and global verification key vk as inputs, it checks the membership of $h, M_1, s \in \mathbb{G}_1$, $h \neq 1_{\mathbb{G}_1}$ and $M_2 \neq 1_{\mathbb{G}_2}$, and then checks the pairing product equations, $e(h, M_2) = e(M_1, g_2)$ and $e(h, \mathsf{vk}_1)e(M_1, \mathsf{vk}_2) = e(s, g_2)$. If all these conditions hold, then the verification algorithm outputs 1 (accept), else it returns 0 (reject).

Correctness: First we show that the reconstruction algorithm for a set of valid partial signatures $\{i, \sigma_i\}_{i \in \mathcal{T}}$, s.t. $|\mathcal{T}| \geq t$ on message $M := (id, M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$ results a valid aggregated signature $\sigma = (h, s)$.

$$s = \prod_{i \in \mathcal{T}} s_i^{L_i^{\mathcal{T}}(0)} = \prod_{i \in \mathcal{T}} \left(h^{\mathsf{sk}_{i1}} M_1^{\mathsf{sk}_{i2}}\right)^{L_i^{\mathcal{T}}(0)} = h^{\sum_{i \in \mathcal{T}} \mathsf{sk}_{i1} L_i^{\mathcal{T}}(0)} M_1^{\sum_{i \in \mathcal{T}} \mathsf{sk}_{i2} L_i^{\mathcal{T}}(0)} = h^{\mathsf{sk}_1} M_1^{\mathsf{sk}_2} \ .$$

Next, we show that the verification phase for the above aggregated signature σ on message $\tilde{M} = (M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}$ succeeds.

$$\begin{aligned} &e(h, M_2) = e(h, g_2^m) = e(h^m, g_2) = e(M_1, g_2) \ , \\ &e(h, \mathsf{vk}_1)e(M_1, \mathsf{vk}_2) = e(h, g_2^x)e(M_1, g_2^y) = e(h^x M_1^y, g_2) = e(s, g_2) \ . \end{aligned}$$

Theorem 4.1. *The Pedersen DKG, used in the proposed message-indexed SPTS construction is $(t-1, k)$-Oracle-aided Algebraic secure, stated in Def. 2.10, with $k \leq nt$.*

Proof. The proof can be found in [BL22, Theorem 4.5]. □

Theorem 4.2. *The proposed Non-Interactive (n, t)-SPTS construction over the index Diffie-Hellman message space is (ε, q_h, q_s)-Threshold* EUF-CiMA *secure, stated in Def. 4.3, in the algebraic group model and random oracle model under the (ε_1)-hardness of t-*OMDL *assumption, i.e. $(t-1, t)$-Oracle-aided algebraic security of the DKG, and the ε_2-hardness of* GPS_3 *assumption, stated in Def. 2.9, such that,*

$$\varepsilon \leq \varepsilon_2 + 4\left(1 - \left(\frac{(t-1)!\,(n-t+1)!}{n!}\right)\right)\varepsilon_1 - q_h^2/p \ .$$

Proof (High-level). We provide a detailed proof in App. B.2, but give a high-level overview of it here. We prove this theorem by defining a sequence of indistinguishable security games and assume the existence of an algebraic adversary $\mathcal{A}_{alg}$ who can break the threshold EUF-CiMA security of the proposed SPTS construction and then we build an algorithm $\mathcal{B}_{alg}$ that plays the role of an attacker against the t-OMDL problem and GPS_3 problem. We start with the actual security definition and then slightly modify it to multiple games s.t. in the last game the challenger can simulate the parameters and the oracles with the adaptive-corruption setting. Thus once $\mathcal{A}_{alg}$ returns a valid forgery σ^* associated to the random oracle query $\mathsf{H}(.)$ along with their algebraic representations, then the algorithm $\mathcal{B}_{alg}$ can use $\mathcal{A}_{alg}$ as a subroutine to break the hardness of the underlying problems. We summarize the defined security games and their main security justifications in Table. 2.

Table 2. The overview of the games to prove the threshold $\mathsf{EUF\text{-}CiMA}$ security of the proposed SPTS construction.

Games	**Public Parameters**	**Random Oracle**	**Corruption Oracle**	**Signing Oracle**	**Justification**
$\boldsymbol{G}_0$	$\mathsf{KGen}(\mathsf{pp}, n, t)$	$\mathsf{H}(.)$	Static	$\mathsf{Par\text{-}Sign}(\mathsf{pp}, \mathsf{sk}_i, M, id)$	-
$\boldsymbol{G}_1$	GPS_3 instance	$\mathcal{O}_0^{\mathrm{GPS}_3}(.)$	Static	$\mathcal{O}_1^{\mathrm{GPS}_3}(.)$	Hardness of GPS_3 problem
$\boldsymbol{G}_2$	GPS_3 instance OMDL instance	$\mathcal{O}_0^{\mathrm{GPS}_3}(.)$	Adaptive $\mathsf{Dlog}(.)$	$\mathcal{O}_1^{\mathrm{GPS}_3}(.)$ $\mathsf{Dlog}(.)$	Hardness of GPS_3 problem Hardness of OMDL problem

We follow a strong notion of unforgeability known as adaptive-corruption that the adversary $\mathcal{A}_{alg}$ can corrupt up to $t-1$ parties at any point of the experiment. As it is shown in [BL22], the Pedersen DKG is $(t-1,t)$-OAAS secure and in the last game to simulate the corruption oracle, the simulator Sim can query the discrete logarithm oracle provided by t-OMDL assumption. We simplify the existing proof techniques of Rial et al. [RP22] and define the global verification key of the threshold signature the same as the actual non-threshold signature construction meanwhile we improve their security notion to the adaptive-corruption setting. To generate the verification key of the honest parties without the knowledge of their secret keys, we utilize the Lagrange polynomial basis to obtain the honest parties share in the exponent. To simulate the partial signing oracle we can use the same technique to compute a partial signature of an honest party by querying to $\mathcal{O}_1^{\mathrm{GPS}_3}$ oracle. Moreover, we define an additional security game $\boldsymbol{G}_3$ that is identical to $\boldsymbol{G}_2$ while the adversary loses the games if it queries a randomly pre-selected identifier to the corruption oracle. Additionally, game $\boldsymbol{G}_4$ which is the same as previous game, except we consider the case that the adversary loses the game if there is a collision in the random oracle. $\square$

5 Applications

As already mentioned in Sect. 1, the compatibility of SPS with Groth-Sahai (GS) NIZK proofs [GS08] makes them an attractive building block for more complex cryptographic protocols. Since GS proofs are straight-line extractable, they are particularly interesting for constructions targeting security in composable security frameworks such as the universal composability (UC) framework. Moreover, the combination with efficient NIZK makes SPS attractive for privacy-preserving applications such as group signatures [AFG+10,LPY15], traceable signatures [ACHO11], blind signatures [AFG+10,FHS15] or anonymous credentials [Fuc11,CDHK15].

In all the aforementioned applications, users basically obtain signatures from some entity. These entities are prone to compromise of the signing key representing a single point of attack and failure. Replacing the use of SPS with SPTS in all these application scenarios helps to reduce on the one hand the trust in a single authority and and increases the availability of the respective signing service. Consequently, SPTS can be considered a general replacement for the SPS used in all of the aforementioned applications.

We will now discuss two specific instantiations of such primitives and how our concrete SPTS can be valuable.

Threshold Issuance Anonymous Credentials. Anonymous credentials (ACs) are an important privacy-preserving authentication technique which allows users to prove the possession of attributes while preserving their anonymity from verifiers. Sonnino et al. proposed Coconut [SAB+19], a so-called Threshold Issuance AC (TIAC) system, that enables a subset of credential issuers to jointly issue credentials. They rely on a threshold variant of PS signatures [PS16]. While Coconut is practical from a performance point of view and already found practical applications [BSKD22], it unfortunately lacks a rigorous security analysis. Recently, Rial and Piotrowska [RP22] conducted a security analysis of Coconut modeled via an attribute-based access control with threshold issuance functionality in the UC model, which requires some changes to the original scheme. However, they rely the assumption that a variant of the PS signatures remains secure in the ROM, which is not formally substantiated. Moreover, they only consider security in a static corruption model, i.e., the adversary needs to specify in the beginning which of the n signing entities it wants to corrupt. Our SPTS can firstly be chosen as a provably secure drop in replacement for the scheme used in [BSKD22] and secondly allows to lift this construction to stronger security guarantees by allowing adaptive corruptions.

Threshold Dynamic Group Signatures. Group signatures allow users to anonymously sign messages on behalf of a managed group without revealing their identity. But there is a dedicated entity (called the opener) who is able to reveal the identity of the exact signer when given a group signature. Camensich et al. [CDL+20] have recently introduced threshold dynamic group signatures where issuing and opening is distributed among several entities. As already mentioned earlier, their construction also relies on a variant of PS signatures. Like in the TIAC systems discussed above, their security model only considers static corruptions. Consequently, we can use our TSPS as a drop in replacement which allows to lift the construction to stronger security guarantees by allowing adaptive corruptions.

6 Conclusion and Future Work

In this work, we introduced a notion of a structure-preserving threshold signature (SPTS) and present an efficient SPTS construction. We formally proved that the proposed SPTS is secure under adaptive corruptions based on a new variant of generalized PS assumption in the algebraic group and random oracle model. We believe this work can open a new line of research for structure-preserving multi-party protocols like structure-preserving threshold encryption. Despite the fact that the indexing method makes the underlying scheme threshold-friendly, a SPTS without using this method is an interesting open question. The security of the proposed construction is based on an interactive assumption and as a future work we consider it interesting to investigate security based on non-interactive assumptions such as a lifted variant of q-MSDH assumption s.t. the experiments challenge only contains source group elements.

Acknowledgments. We would like to thank Behzad Abdolmaleki and Daniele Cozzo for their helpful discussions and valuable comments on an earlier version of this work. Mahdi Sedaghat and Bart Preneel were supported in part by the Research Council KU Leuven C1 on Security and Privacy for Cyber-Physical Systems and the Internet of Things with contract number C16/15/058 and by CyberSecurity Research Flanders with reference number VR20192203. Daniel Slamanig was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871473 (Kraken) and No. 861696 (Labyrinth) and by the Austrian Science Fund (FWF) and netidee SCIENCE under grant agreement P31621-N38 (Profet). The work of Markulf Kohlweiss was done while visiting COSIC.

References

AAOT18. Masayuki Abe, Miguel Ambrona, Miyako Ohkubo, and Mehdi Tibouchi. Lower bounds on structure-preserving signatures for bilateral messages. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 3–22. Springer, Heidelberg, September 2018.

ACD+12. Masayuki Abe, Melissa Chase, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 4–24. Springer, Heidelberg, December 2012.

ACHO11. Masayuki Abe, Sherman S. M. Chow, Kristiyan Haralambiev, and Miyako Ohkubo. Double-trapdoor anonymous tags for traceable signatures. In Javier Lopez and Gene Tsudik, editors, *ACNS 11*, volume 6715 of *LNCS*, pages 183–200. Springer, Heidelberg, June 2011.

AFG+10. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, August 2010.

AGHO11. Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 649–666. Springer, Heidelberg, August 2011.

AGO11. Masayuki Abe, Jens Groth, and Miyako Ohkubo. Separating short structure-preserving signatures from non-interactive assumptions. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 628–646. Springer, Heidelberg, December 2011.

AGOT14. Masayuki Abe, Jens Groth, Miyako Ohkubo, and Mehdi Tibouchi. Unified, minimal and selectively randomizable structure-preserving signatures. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 688–712. Springer, Heidelberg, February 2014.

AJO+19. Masayuki Abe, Charanjit S. Jutla, Miyako Ohkubo, Jiaxin Pan, Arnab Roy, and Yuyu Wang. Shorter QA-NIZK and SPS with tighter security. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 669–699. Springer, Heidelberg, December 2019.

ALP12. Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Computing on authenticated data: New privacy definitions and constructions. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 367–385. Springer, Heidelberg, December 2012.

BB08. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

BCF+11. Olivier Blazy, Sébastien Canard, Georg Fuchsbauer, Aline Gouget, Hervé Sibert, and Jacques Traoré. Achieving optimal anonymity in transferable e-cash with a judge. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT 11*, volume 6737 of *LNCS*, pages 206–223. Springer, Heidelberg, July 2011.

BCN+10. Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 381–398. Springer, Heidelberg, September 2010.

BDV+20. Luís TAN Brandão, Michael Davidson, Apostol Vassilev, et al. Nist roadmap toward criteria for threshold schemes for cryptographic primitives. In *National Institute of Standards and Technology Internal or Interagency Report 8214A*, 2020.

BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

BL22. Renas Bacho and Julian Loss. On the adaptive security of the threshold BLS signature scheme. Cryptology ePrint Archive, Report 2022/534, 2022. `https://eprint.iacr.org/2022/534`.

Bla79. G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48:313–317, 1979.

BLS01. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

BNPS03. Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003.

Bol03. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, January 2003.

BSKD22. Mathieu Baudet, Alberto Sonnino, Mahimna Kelkar, and George Danezis. Zef: Low-latency, scalable, private payments. Cryptology ePrint Archive, Report 2022/083, 2022. `https://eprint.iacr.org/2022/083`.

CDHK15. Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 262–288. Springer, Heidelberg, November / December 2015.

CDL+20. Jan Camenisch, Manu Drijvers, Anja Lehmann, Gregory Neven, and Patrick Towa. Short threshold dynamic group signatures. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 401–423. Springer, Heidelberg, September 2020.

CDN01. Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 280–299. Springer, Heidelberg, May 2001.

CFSY96. Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-autority secret-ballot elections with linear work. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 72–83. Springer, Heidelberg, May 1996.

CGG+20. Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1769–1787. ACM Press, November 2020.

CGJ+99. Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 98–115. Springer, Heidelberg, August 1999.

CGS97. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 103–118. Springer, Heidelberg, May 1997.

CH20. Geoffroy Couteau and Dominik Hartmann. Shorter non-interactive zero-knowledge arguments and ZAPs for algebraic languages. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 768–798. Springer, Heidelberg, August 2020.

CM11. Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings - the role of Ψ revisited. *Discret. Appl. Math.*, 159(13):1311–1322, 2011.

DDFY94. Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *26th ACM STOC*, pages 522–533. ACM Press, May 1994.

Des90. Yvo Desmedt. Making conditionally secure cryptosystems unconditionally abuse-free in a general context. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 6–16. Springer, Heidelberg, August 1990.

DF90. Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, Heidelberg, August 1990.

DK01. Ivan Damgård and Maciej Koprowski. Practical threshold RSA signatures without a trusted dealer. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 152–165. Springer, Heidelberg, May 2001.

DKLs19. Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *2019 IEEE Symposium on Security and Privacy*, pages 1051–1066. IEEE Computer Society Press, May 2019.

DN03. Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 247–264. Springer, Heidelberg, August 2003.

EGK14. Ali El Kaafarani, Essam Ghadafi, and Dalia Khader. Decentralized traceable attribute-based signatures. In Josh Benaloh, editor, *CT-RSA 2014*, volume 8366 of *LNCS*, pages 327–348. Springer, Heidelberg, February 2014.

FHS15. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 233–253. Springer, Heidelberg, August 2015.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

Fuc09. Georg Fuchsbauer. Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320, 2009. `https://eprint.iacr.org/2009/320`.

Fuc11. Georg Fuchsbauer. Commuting signatures and verifiable encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 224–245. Springer, Heidelberg, May 2011.

Gha16. Essam Ghadafi. Short structure-preserving signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 305–321. Springer, Heidelberg, February / March 2016.

Gha17. Essam Ghadafi. More efficient structure-preserving signatures - or: Bypassing the type-III lower bounds. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *ESORICS 2017, Part II*, volume 10493 of *LNCS*, pages 43–61. Springer, Heidelberg, September 2017.

GHKP18. Romain Gay, Dennis Hofheinz, Lisa Kohl, and Jiaxin Pan. More efficient (almost) tightly secure structure-preserving signatures. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 230–258. Springer, Heidelberg, April / May 2018.

GJKR96. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold DSS signatures. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 354–371. Springer, Heidelberg, May 1996.

GJKR99. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 295–310. Springer, Heidelberg, May 1999.

GJKR03. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure applications of Pedersen's distributed key generation protocol. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 373–390. Springer, Heidelberg, April 2003.

GJM$^+$21. Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Aggregatable distributed key generation. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 147–176. Springer, Heidelberg, October 2021.

GMR88. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

GPS08. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008. Applications of Algebra to Cryptography.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

HJ12. Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 590–607. Springer, Heidelberg, August 2012.

JR17. Charanjit S. Jutla and Arnab Roy. Improved structure preserving signatures under standard bilinear assumptions. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 183–209. Springer, Heidelberg, March 2017.

KG20. Chelsea Komlo and Ian Goldberg. FROST: flexible round-optimized schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, volume 12804 of *Lecture Notes in Computer Science*, pages 34–65. Springer, 2020.

KLAP20. Hyoseung Kim, Youngkyung Lee, Michel Abdalla, and Jong Hwan Park. Practical dynamic group signature with efficient concurrent joins and batch verifications. Cryptology ePrint Archive, Report 2020/921, 2020. `https://eprint.iacr.org/2020/921`.

KMOS21. Yashvanth Kondi, Bernardo Magri, Claudio Orlandi, and Omer Shlomovits. Refresh When You Wake Up: Proactive Threshold Wallets with Offline Devices. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 608–625, 2021.

KPW15. Eike Kiltz, Jiaxin Pan, and Hoeteck Wee. Structure-preserving signatures from standard assumptions, revisited. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 275–295. Springer, Heidelberg, August 2015.

KSAP21. Hyoseung Kim, Olivier Sanders, Michel Abdalla, and Jong Hwan Park. Practical dynamic group signatures without knowledge extractors. Cryptology ePrint Archive, Report 2021/351, 2021. `https://eprint.iacr.org/2021/351`.

Lin17. Yehuda Lindell. Fast secure two-party ECDSA signing. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 613–644. Springer, Heidelberg, August 2017.

LJY14. Benoît Libert, Marc Joye, and Moti Yung. Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM PODC*, pages 303–312. ACM, July 2014.

LPJY13. Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Linearly homomorphic structure-preserving signatures and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 289–307. Springer, Heidelberg, August 2013.

LPY15. Benoît Libert, Thomas Peters, and Moti Yung. Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 296–316. Springer, Heidelberg, August 2015.

MR01. Philip D. MacKenzie and Michael K. Reiter. Two-party generation of DSA signatures. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 137–154. Springer, Heidelberg, August 2001.

MRV99. Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, October 1999.

MTT19. Taiga Mizuide, Atsushi Takayasu, and Tsuyoshi Takagi. Tight reductions for Diffie-Hellman variants in the algebraic group model. In Mitsuru Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 169–188. Springer, Heidelberg, March 2019.

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

PS16. David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Heidelberg, February / March 2016.

PS18. David Pointcheval and Olivier Sanders. Reassessing security of randomizable signatures. In Nigel P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 319–338. Springer, Heidelberg, April 2018.

RP22. Alfredo Rial and Ania M. Piotrowska. Security analysis of coconut, an attribute-based credential scheme with threshold issuance. Cryptology ePrint Archive, Report 2022/011, 2022. `https://eprint.iacr.org/2022/011`.

RSA78. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

SAB+19. Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In *NDSS 2019*. The Internet Society, February 2019.

SG98. Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 1–16. Springer, Heidelberg, May / June 1998.

Sha79. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

Sho00. Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer, Heidelberg, May 2000.

SP21. Mahdi Sedaghat and Bart Preneel. Cross-domain attribute-based access control encryption. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *Cryptology and Network Security*, pages 3–23. Springer International Publishing, 2021.

TS21. Dmytro Tymokhanov and Omer Shlomovits. Alpha-rays: Key extraction attacks on threshold ecdsa implementations. Cryptology ePrint Archive, Report 2021/1621, 2021. `https://ia.cr/2021/1621`.

WC21. X. Wang and S. M. Chow. Cross-Domain Access Control Encryption: Arbitrary-Policy, Constant-Size, Efficient. In *2021 2021 IEEE Symposium on Security and Privacy (SP)*, pages 388–401, Los Alamitos, CA, USA, may 2021. IEEE Computer Society.

A Omitted Definitions

A.1 Digital Signatures

Digital signatures as an electronic analogue of a written signature ensure communication privacy, the integrity of data, the authenticity of digital messages/senders, and the non-repudiation of the sender. In what follows we formally define DS schemes and their security requirements.

Definition A.1 (Digital Signatures). *A digital signature, $\Psi_{\mathbf{DS}}$, over message space $\mathcal{M}$ consists of following PPT algorithms:*

- $(\mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$*: Setup is a probabilistic algorithm which takes the security parameter 1^λ as input and returns the set of public parameters* pp *as output.*
- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$*: Key generation is a probabilistic algorithm which takes* pp *as input and outputs the pair of signing/verification keys* $(\mathsf{sk}, \mathsf{vk})$*.*
- $(\sigma) \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, m)$*: Signing algorithm takes* pp*, secret key* sk *and a message $m \in \mathcal{M}$ as inputs and returns signature σ as output.*
- $(0, 1) \leftarrow \mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \sigma, m)$*: Verification is a deterministic algorithm which takes* pp*, a signature σ, the message $m \in \mathcal{M}$ and verification key* vk *as inputs, responds with either 0 (reject) or 1 (accept).*

The primary security requirements for digital signature are *Correctness* and *unforgeability against chosen message attack*, which define as follows:

Definition A.2 (Correctness). *A digital signature, Ψ_{DS}, is called correct, if for a given public parameters* pp*, we have:*

$$\Pr\left[\forall\ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}), m \in \mathcal{M} : \mathsf{Verify}\left(\mathsf{vk}, m, \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, m)\right) = 1\ \right] \geq 1 - \mathsf{negl}(\lambda)\ .$$

Definition A.3 (Existential Unforgeability under Chosen Message Attack (EUF-CMA) [GMR88]). *A digital signature, Ψ_{DS}, is ε-EUF-CMA-secure if for all PPT adversaries $\mathcal{A}$ on winning the security game in Fig. 7 we have, $Adv^{\mathsf{EUF\text{-}CMA}}_{\mathsf{DS},\mathcal{A}}(\lambda) = \Pr[\mathbf{G}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}}(1^\lambda) = 1] \leq \varepsilon$.*

$\mathbf{G}_{\mathcal{A}}^{\text{EUF-CMA}}(1^\lambda)$	$\mathcal{O}_{\mathsf{Sign}}(m)$
$1: \ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$	$1: \ \text{Initialize } \mathcal{Q} = \emptyset$
$2: \ (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$	$2: \ \sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, m)$
$3: \ (m^*, \sigma^*) \leftarrow\$ \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{vk})$	$3: \ \mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
$4: \ \textbf{return } (m^* \notin \mathcal{Q} \ \wedge \ \mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, m^*, \sigma^*) = 1)$	$4: \ \textbf{return } \sigma$

Fig. 7. Game $\mathbf{G}_{\mathcal{A}}^{\text{EUF-CMA}}(1^\lambda)$.

There is a weaker security notion known as EUF-wCMA [BB08] which requires the adversary to provide the challenger with the list of messages $\{m_1, \ldots, m_q\}$ at the beginning of the game (before receiving the public parameters and verification key).

Pointcheval-Sanders Signatures [PS16]. The PS signature (CT-RSA'16) works on an asymmetric bilinear setting and for a single scalar message, i.e. $k = 1$, consists of the following PPT algorithms:

- $(\mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$: The setup algorithm takes the security parameter 1^λ as input and returns the global public parameters $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{p}, e, g_1, g_2)$, where $g_1 \leftarrow\$ \mathbb{G}_1$ and $g_2 \leftarrow\$ \mathbb{G}_2$ are randomly selected generators as output.
- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$: Takes the global public parameters pp as input, samples two random integers $x, y \leftarrow\$ \mathbb{Z}_p^*$, returns the verification key as $\mathsf{vk} = (\mathsf{vk}_1, \mathsf{vk}_2) := (g_2^x, g_2^y)$ and keeps the signing key $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2) = (x, y)$ secret.
- $(\sigma) \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, m)$: Takes the secret signing key sk and a integer message $m \in \mathbb{Z}_p$ as inputs. It samples $r \leftarrow\$ \mathbb{Z}_p^*$ uniformly at random and then computes $\sigma = (h, s) = (g_1^r, h^{x+my})$ and returns the signature σ as output.
- $(0, 1) \leftarrow \mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \sigma, m)$: To verify a signature σ, this algorithm takes pp, the verification key vk and message m as inputs. If $h \neq 1_{\mathbb{G}_1}$ and the pairing product equation $e(h, \mathsf{vk}_1 \mathsf{vk}_2^m) = e(s, g_2)$ holds, then it returns 1 (accept), otherwise it returns 0 (reject).

The correctness of this construction is straightforward and it is EUF-CMA-secure under the PS assumption, stated in Def. 2.6.

Ghadafi's SPS. Next, for a given security parameter λ, we outline the Ghadafi's SPS construction [Gha16] over a Diffie-Hellman message space $\mathcal{M}_{\mathsf{DH}}$, as follows:

- $(\mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$: The setup algorithm takes the security parameter 1^λ as input and returns the global public parameters $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{p}, e, g_1, g_2)$, where $g_1 \leftarrow\$ \mathbb{G}_1$ and $g_2 \leftarrow\$ \mathbb{G}_2$ are randomly selected generators as output.

– $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$: The key generation algorithm takes the global public parameters pp as input, samples two random integers $x, y \leftarrow\!\!\$\, \mathbb{Z}_p^*$, returns the verification key as $\mathsf{vk} = (\mathsf{vk}_1, \mathsf{vk}_2) := (g_2^x, g_2^y)$ and keeps the secret signing key $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2) = (x, y)$ secure.
– $(\sigma) \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, M_1, M_2)$: The signing algorithm takes the secret signing key sk and a DH message $(M_1, M_2) \in \mathcal{M}_{\mathsf{DH}}$ such that $e(M_1, g_2) = e(g_1, M_2)$ as inputs and samples $r \leftarrow\!\!\$\, \mathbb{Z}_p^*$ uniformly at random. Then it computes $R = g_1^r$, $S = M_1^r$ and $T = R^x S^y$, and returns the signature $\sigma = (R, S, T)$ as output.
– $(0, 1) \leftarrow \mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \sigma, M_1, M_2)$: To verify a signature σ, this algorithm takes pp, the verification key vk and a message $(M_1, M_2) \in \mathcal{M}_{\mathsf{DH}}$ as inputs. If $R \neq 1_{\mathbb{G}_1}, S, T \in \mathbb{G}_1$ and both pairing equations $e(R, M_2) = e(S, g_2)$ and $e(T, g_2) = e(R, \mathsf{vk}_1)e(S, \mathsf{vk}_2)$ hold, then it returns 1 (accept), otherwise it returns 0 (reject).

B Omitted Proofs

B.1 Proof of Theorem 2.1

Proof. Over an asymmetric bilinear setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$, let $\mathcal{B}_{Alg}$ is the challenger of GPS_3 assumption in the AGM and receives the SDL instance $(Z_1, Z_2) = (g_1^z, g_2^z)$ as input. It simulates the GPS_3 instance $(X, Y) := (g_2^x, g_2^y) = (Z_2^{a_0} g_2^{b_0}, Z_2^{a_0'} g_2^{b_0'})$ by setting $x := a_0 z + b_0$ and $y := a_0' z + b_0'$, where a_0, b_0, a_0', b_0' are sampled uniformly at random from $\mathbb{Z}_p$. Let the maximum number of queries to oracle $\mathcal{O}_0^{\text{GPS}_3}$ and $\mathcal{O}_1^{\text{GPS}_3}$ are denoted by q_0 and q_1, respectively. Then $\mathcal{B}_{alg}$ simulates the defined oracles as follows:

– **Oracle** $\mathcal{O}_0^{\text{GPS}_3}(id_i)$: To simulate the i^{th} query s.t. $i \in [1, q_0]$, if $\mathcal{Q}_0[id_i] = \perp$, $\mathcal{B}_{alg}$ samples $a_i, b_i \leftarrow\!\!\$\, \mathbb{Z}_p$ and assigns $\mathcal{Q}_0[id_i] \leftarrow Z_1^{a_i} g_1^{b_i}$ that implicitly sets $r_i = a_i z + b_i$.
– **Oracle** $\mathcal{O}_1^{\text{GPS}_3}\left((h_j; \vec{h}_j), (M_{j1}; \vec{M}_{j1}), (M_{j2}; \vec{M}_{j2})\right)$: To simulate the j^{th} query for $j \geq 2$ to this oracle we assume that the challenger has successfully simulated the $k - 1$ previously queries to this oracle. Thus the algebraic adversary $\mathcal{A}_{alg}$ has access to $\left\{s_\ell = g_1^{r_\ell(x + m_\ell y)}\right\}_{\ell=1}^{j-1}$, where $r_\ell = \mathsf{dlog}_{g_1}(h_\ell)$ and m_ℓ is the extracted scalar message by the defined extractor $\mathsf{Ext}(.)$. In this case, $\mathcal{A}_{alg}$ makes the j^{th}-query to the oracle $\mathcal{O}_1^{\text{GPS}_3}$ by providing the tuple $\left((h_j; \vec{h}_j), (M_{j1}; \vec{M}_{j1}), (M_{j2}; \vec{M}_{j2})\right)$ that can be determined by the following polynomials:

$$\mathbf{P}_{M_{j1}}[\vec{X}_1] = [M_{j1}|g_1] + \sum_{\ell=1}^{q_0} \mathbf{R}_\ell [M_{j1}|h_\ell] + \sum_{\ell=1}^{j-1} \mathbf{R}_\ell [M_{j1}|s_\ell](\mathbf{X} + m_\ell \mathbf{Y}) \ , \tag{4a}$$

$$\mathbf{P}_{M_{j2}}[\vec{X}_2] = [M_{j2}|g_2] + \mathbf{X}[M_{j2}|X] + \mathbf{Y}[M_{j2}|Y] \ , \tag{4b}$$

where $\vec{X}_1 = (\mathbf{X}, \mathbf{Y}, \mathbf{R}_1, \ldots, \mathbf{R}_{|\mathcal{Q}_0|})$ and $\vec{X}_2 = (\mathbf{X}, \mathbf{Y})$. These two polynomials are defined based on the fact that, $\mathcal{A}_{alg}$ has access to the answers received from the oracles, $\{h_\ell = g_1^{r_\ell}\}_{\ell=1}^{q_0}$ and $\left\{s_\ell = g_1^{r_\ell(x+m_\ell y)}\right\}_{\ell=1}^{j-1}$ of the first source group elements and the GPS_3 instances of the second source group elements. As it is shown in Fig. 2, the oracle $\mathcal{O}_1^{\text{GPS}_3}$ does not fail if $e(h_j, M_{j2}) = e(M_{j1}, g_2)$, which it implies the polynomial $P_j[\vec{X}_T] = P_{M_{j1}}[\vec{X}_1] - \mathbf{R}_j P_{M_{j2}}[\vec{X}_2]$, where $\vec{X}_T = \vec{X}_1 \cdot \vec{X}_2$, should be vanished on at least one point like $\vec{x}_1 = (x, y, r_1, \ldots, r_{q_0})$ and $\vec{x}_2 = (x, y)$. We define the event E as the case that $P_j[\vec{X}_T] = 0$ and there are two possible cases: 1 The event E fulfils, i.e. $P_j[\vec{X}_T] = 0$, then the defined extractor can successfully extract the scalar messages m_j for $1 \leq j \leq q_1$. 2 The event does not fulfil, $\neg E$, i.e. $P_j[\vec{X}_T] \neq 0$, then we can define an algebraic adversary $\mathcal{D}$ that can solve the SDL problem with a non-negligible advantage. We formally discuss these two cases in the following claims.

Claim 0.1: If $P_j[\vec{X}_T] = 0$, then the extractor can successfully extracts the scalar messages m_j.

Proof. Similar to the proof of [KSAP21, Theorem 3.6, Claim.1], the condition $P_j[\vec{X}_T] = 0$ implies the equality $P_{M_{j1}}[\vec{X}_1] = \mathbf{R}_j P_{M_{j2}}[\vec{X}_2]$ must hold. Thus based on the received representations from $\mathcal{A}_{alg}$, we can write:

$$\begin{aligned}&[M_{j1}|g_1] + \sum_{\ell=1}^{q_0\setminus\{j\}} \mathbf{R}_\ell [M_{j1}|h_\ell] + \mathbf{R}_j [M_{j1}|h_j] + \sum_{\ell=1}^{j-1} \mathbf{R}_\ell [M_{j1}|s_\ell](\mathbf{X} + m_\ell \mathbf{Y}) = \\ &\mathbf{R}_j \left([M_{j2}|g_2] + \mathbf{X}[M_{j2}|X] + \mathbf{Y}[M_{j2}|Y]\right) \ .\end{aligned} \tag{5}$$

Which this implies:

$$[M_{j2}|g_2] = [M_{j1}|h_j] \text{ (Due to } \mathbf{R}_j) \ , \tag{6a}$$
$$[M_{j1}|g_1] = 0 \text{ (Due to } \mathbf{R}_j) \ , \tag{6b}$$
$$[M_{j2}|X] = 0 \text{ (Due to } \mathbf{R}_j\mathbf{X}) \ , \tag{6c}$$
$$[M_{j2}|Y] = 0 \text{ (Due to } \mathbf{R}_j\mathbf{Y}) \ , \tag{6d}$$
$$\{[M_{j1}|h_\ell] = 0\}_{\forall\ \ell\in[1,q_0],\ell\neq j} \text{ (Due to } \mathbf{R}_\ell) \ , \tag{6e}$$
$$\{[M_{j1}|s_\ell] = 0\}_{\forall\ \ell\in[1,j-1]} \text{ (Due to } \mathbf{R}_\ell\mathbf{X}) \ . \tag{6f}$$

Finally, based on the above equations we can write, $M_{j2} = g_2^{[M_{j2}|g_2]}$ and $M_{j1} = h_j^{[M_{j1}|h_j]}$ and therefore the extractor returns $m_j := [M_{j2}|g_2] = [M_{j1}|h_j]$ as the scalar message. □

Claim 0.2: If $P_j[\vec{X}_T] \neq 0$, i.e. the extractor fails, the SDL problem is not hard.

Proof. Based on the fact that $x := a_0 z + b_0$, $y := a_0' z + b_0'$ and $\{r_\ell := a_\ell z + b_\ell\}_{\ell=1}^{q_0}$, we can convert the variables $\mathbf{X}$, $\mathbf{Y}$ and $\{\mathbf{R}_\ell\}_{\ell=1}^{q_0}$ to $\mathbf{A_0}\mathbf{Z} + \mathbf{B_0}$, $\mathbf{A_0'}\mathbf{Z} + \mathbf{B_0'}$ and $\{\mathbf{A}_\ell \mathbf{Z} + \mathbf{B}_\ell\}_{\ell=1}^{q_0}$ and define a univariate polynomial $G_j^*[\vec{Z}_T]$ from the polynomial $P_j^*[\vec{X}_T]$. If $G_j^*[\vec{Z}_T] \neq 0$, then the equality $G_j^*[\vec{z}_T] = P_j^*(\vec{x}_T)$ implies that $G_j^*[\vec{z}_T]$ has at least one root like $\vec{z}_T$. Like the analysis in the proof of [KSAP21, Theorem 3.6, Claim.2], we have $\Pr[G_j^*[\vec{Z}_T] = 0] \leq \Pr[a_3 = 0] \leq 3/p$ (Schwartz-Zippel lemma), where a_3 is the leading coefficient of $G_j^*[\vec{Z}_T]$. Additionally, in the case of $G_j^*[\vec{Z}_T] \neq 0$, there exists a vector $\vec{z}$ as the root for this polynomial that can be the solution of SDL problem. Thus we can write:

$$\Pr[\neg E] \leq \Pr[P_j^*[\vec{X}_T] \neq 0 \wedge G_j^*[\vec{Z}_T] \neq 0] + \Pr[G_j^*[\vec{Z}_T] = 0] \leq Adv_{\mathcal{D}}^{SDL}(\lambda) + 3/p \ . \tag{7}$$

Thus as a contradiction, if the extractor fails then we can define an algebraic algorithm $\mathcal{D}$ that can solve the SDL problem with a non-negligible advantage. We can conclude the challenger of the GPS_3 problem can successfully simulate the defined oracles for the adversary $\mathcal{A}_{alg}$. □

W.l.o.g we assume that the adversary $\mathcal{A}_{alg}$ has queried the maximum number of queries q_0 and q_1 to the provided oracles as above s.t. $q_1 \leq q_0$. It finally outputs $\Big((h^*;\vec{h}^*), (M_1^*;\vec{M}_1^*), (M_2^*;\vec{M}_2^*), (s^*;\vec{s^*})\Big)$ based on the received responses from the oracles and also public parameters. From the received representations we can write:

$$P_h^*[\vec{X}_1] = [h^*|g_1] + \sum_{\ell=1}^{q_0} \mathbf{R}_\ell[h^*|h_\ell] + \sum_{\ell=1}^{q_1} \mathbf{R}_\ell[h^*|s_\ell](\mathbf{X} + m_\ell\mathbf{Y}) \ , \tag{8a}$$

$$P_{M_1}^*[\vec{X}_1] = [M_1^*|g_1] + \sum_{\ell=1}^{q_0} \mathbf{R}_\ell[M_1^*|h_\ell] + \sum_{\ell=1}^{q_1} \mathbf{R}_\ell[M_1^*|s_\ell](\mathbf{X} + m_\ell\mathbf{Y}) \ , \tag{8b}$$

$$P_{M_2}^*[\vec{X}_2] = [M_2^*|g_2] + \mathbf{X}[M_2^*|X] + \mathbf{Y}[M_2^*|Y] \ , \tag{8c}$$

$$P_s^*[\vec{X}_1] = [s^*|g_1] + \sum_{\ell=1}^{q_0} \mathbf{R}_\ell[s^*|h_\ell] + \sum_{\ell=1}^{q_1} \mathbf{R}_\ell[s^*|s_\ell](\mathbf{X} + m_\ell\mathbf{Y}) \ . \tag{8d}$$

Similar to the proof of [KSAP21, Theorem 3.6] if all conditions in the security game $\mathbf{G}_{\mathcal{A}_{alg}}^{\mathrm{GPS}_3}$ in Fig. 2 are satisfied then we can define the following events:

1. Event E_1: $\mathbf{G}_{\mathcal{A}_{alg}}^{\mathrm{GPS}_3} = 1$ and also the extractor $\mathsf{Ext}(.)$ does not fail.
2. Event E_2: The polynomial $P_2^*[\vec{X}_1] = P_s^*[\vec{X}_1] - \Big(\mathbf{X}P_h^*[\vec{X}_1] + \mathbf{Y}P_{M_1}^*[\vec{X}_1]\Big)$ is the zero-polynomial.
3. Event E_3: The polynomial $P_3^*[\vec{X}_T] = P_{M_2}^*[\vec{X}_2]P_h^*[\vec{X}_1] - P_{M_1}^*[\vec{X}_1]$ is the zero-polynomial.

Claim 0.3: $\Pr[E_1 \wedge E_2 \wedge E_3] = 0$.

Proof. The proof of this claim is similar to the proof of [KSAP21, Theorem 3.6, Claim.3]. $\square$

Claim 0.4: $\Pr[E_1 \wedge \neg E_2] + \Pr[E_1 \wedge E_2 \wedge \neg E_3] \leq Adv_{\mathcal{D}}^{SDL}(\lambda) + 7/p$.

Proof. The proof of this claim is similar to the proof of [KSAP21, Theorem 3.6, Claim.4]. $\square$

Thus we can write:

$$\begin{aligned}&\Pr[E_1] = \Pr[E_1 \wedge E_2 \wedge E_3] + \Pr[E_1 \wedge \neg E_2] + \Pr[E_1 \wedge E_2 \wedge \neg E_3] = \\ &\Pr[E_1 \wedge \neg E_2] + \Pr[E_1 \wedge E_2 \wedge \neg E_3] \leq Adv_{\mathcal{D}}^{SDL}(\lambda) + 7/p \ .\end{aligned} \tag{9}$$

We can conclude:

$$\begin{aligned}&Adv_{\mathcal{B}_{alg}}^{\mathrm{GPS}_3}(\lambda) = \Pr[\mathbf{G}_{\mathcal{A}_{alg}}^{\mathrm{GPS}_3} = 1 \wedge \neg E] + \Pr[\mathbf{G}_{\mathcal{A}_{alg}}^{\mathrm{GPS}_3} = 1 \wedge E] \leq \\ &Adv_{\mathcal{D}}^{SDL}(\lambda) + 3/p + Adv_{\mathcal{D}}^{SDL}(\lambda) + 7/p \leq 2Adv_{\mathcal{D}}^{SDL}(\lambda) + 10/p \ .\end{aligned} \tag{10}$$

$\square$

B.2 Proof of Theorem 4.2

Proof. The proof technique is inspired by the simulatability property of threshold signatures introduced by Gennaro et al. [GJKR96] and to achieve adaptive security we borrow the proof technique used in the recent work of Bacho and Loss [BL22]. We use a sequence of games and show that each is computationally indistinguishable from the previous. The first game, $\boldsymbol{G}_0$, is the real security game defined in Def. 4.3. We show that the proposed (n, t)-SPTS construction is (ε, q_h, q_s)-Threshold $\mathsf{EUF\text{-}CiMA}$ secure, if the underlying DKG construction is $(t-1, t)$-OAAS secure, i.e. the t-OMDL assumption is (ε_1)-hard along with the GPS_3 problem is ε_2-hard (the underlying MI-SPS is $\mathsf{EUF\text{-}CiMA}$ secure). Let an algebraic adversary, $\mathcal{A}_{alg}$, which forges Ψ_{SPTS} successfully with a non-negligible advantage, we build an algebraic algorithm $\mathcal{B}_{alg}$ against the hardness of t-OMDL assumption and the hardness of GPS_3 assumption (the unforgeability of the proposed MI-SPS) that uses $\mathcal{A}_{alg}$ as a subroutine and has the success probability of $\leq \Pr[\mathrm{GPS}_3^{\mathcal{B}} = 1] + 4\left(1 - \left(\frac{(t-1)!\,(n-t+1)!}{n!}\right)\right)\Pr[t\text{-}\mathsf{OMDL}^{\mathcal{B}} = 1] - q_h^2/p$, in the AGM+ROM. This can be demonstrated in the following steps:

• **Game $\boldsymbol{G}_0$**: This is the actual unforgeability game with static corruption and we assume the existence of a challenger $\mathcal{B}_{alg}$ who is taking the role of all uncorrupted players in the distributed key generation phase and honestly answers the required oracles.

Initialization. For a given asymmetric bilinear group setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$, $\mathcal{B}_{alg}$ initializes $\mathcal{S}_{(id,M_2)} = \emptyset$ for all $(id, \tilde{M}) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$, $\mathcal{C} = \emptyset$ and $\mathcal{H} = \mathcal{P} \setminus \mathcal{C}$, $\mathcal{Q}_{\mathsf{H}} = \emptyset$ and $\mathcal{Q}_{\mathsf{S}} = \emptyset$. Additionally as a trusted dealer $\mathcal{B}_{alg}$ samples a global secret key $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2) =$

$(x, y) \leftarrow\!\$\, \mathbb{Z}_p^2$ and defines two polynomials $F(x) \in \mathbb{Z}_p^t[X]$ and $G(x) \in \mathbb{Z}_p^t[X]$ of degree t s.t. $F(0) = x$ and $G(0) = y$ and then generates the global verification key $\mathsf{vk} = (g_2^x, g_2^y)$ along with $\mathsf{vk}_j = (g_2^{F(j)}, g_2^{G(j)})$ for and parties P_j s.t. $j \in [1, n]$. It also receives the list of corrupted parties $\mathcal{C}$ from $\mathcal{A}_{alg}$ (static corruption).

Oracles: According to the definition of threshold $\mathsf{EUF\text{-}CiMA}$ security, $\mathcal{B}_{alg}$ should provide the corruption and partial signing oracles along with a random oracle for the adversary $\mathcal{A}$ as follows:

– **Random oracle**, $\mathsf{H}(id)$: It takes an index $id \in \mathbb{I}$ as input and if $\mathcal{Q}_{\mathsf{H}}[id] = \perp$, it samples $r \leftarrow\!\$\, \mathbb{Z}_p$ and assigns $\mathcal{Q}_{\mathsf{H}}[id] \leftarrow g_1^r$. It returns $\mathcal{Q}_{\mathsf{H}}[id]$ as output. W.l.o.g let the algebraic adversary $\mathcal{A}_{alg}$ before any partial signing query on an indexed DH message $(id, M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$, she queries the random oracle $\mathsf{H}(.)$ with id and it answers with h s.t. $\mathsf{dlog}_h(M_1) = \mathsf{dlog}_{g_2}(M_2) = m$.
– **Oracle** $\mathcal{O}_C(j)$: If a player P_j is corrupted by adversary $\mathcal{A}_{alg}$, the challenger updates $\mathcal{H} = \mathcal{H} \backslash \{j\}$ and $\mathcal{C} = \mathcal{C} \cup \{j\}$ and then reveals the internal state of P_j which contains its secret signing key shares $\mathsf{sk}_j = (F(j), G(j))$. Note that $\mathcal{B}_{alg}$ has access to all coefficients of these two polynomials and can compute the secret key of all parties including corrupted and honest parties.
– **Oracle** $\mathcal{O}_{\mathsf{PSign}}(k, M)$: Adversary $\mathcal{A}_{alg}$ has access to the partial signing oracle which is provided by the challenger $\mathcal{B}_{alg}$. For queries $M := (id, M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$ and an honest party identifier $k \in \mathcal{H}$, if $(id, \star) \notin \mathcal{Q}_{\mathsf{S}}$, then it honestly runs the partial signing algorithm $\sigma_k \leftarrow \mathsf{Par\text{-}Sign}(\mathsf{pp}, \mathsf{sk}_k, M)$, updates $\mathcal{S}_{(id,M_2)} = \mathcal{S}_{(id,M_2)} \cup \{k\}$ and $\mathcal{Q}_{\mathsf{S}} = \mathcal{Q}_{\mathsf{S}} \cup \{(id, M_2)\}$, and returns σ_k to $\mathcal{A}_{alg}$.

Forgery: At the end of this game, $\mathcal{A}$ returns forge signature σ^* on message $\tilde{M}^* = (M_1^*, M_2^*) \in \mathcal{M}_{\mathsf{iDH}}$. We denote the event of σ^* being a valid signature on message $\tilde{M}^*$ by $G_0^{\mathcal{A}} = 1$.

• **Game G_1**: This game is identical to the previous game except the fact that the challenger issues the public parameters and provides the oracles with the help of the challenger of GPS_3 problem. Similar to the previous game, in this game we assume the static corruption and later we lift it to the adaptive corruption under the hardness of t-OMDL problem.

Initialization. $\mathcal{B}_{alg}$ receives the GPS_3 instance, $(g_2^{r_0}, g_2^{v_0})$ from the challenger, and the list of corrupted parties, $\mathcal{C}$, from $\mathcal{A}_{alg}$ s.t. $|\mathcal{C}| \leq t - 1$ and sets $\mathcal{H} = \mathcal{P} \setminus \mathcal{C}$. $\mathcal{B}_{alg}$ assigns the global verification key as $\mathsf{vk}_0 \leftarrow (g_2^{r_0}, g_2^{v_0})$ and acts as follows:

– To define the pair of secret/verification keys of the corrupted parties P_i s.t. $i \in \mathcal{C}$, the challenger $\mathcal{B}_{alg}$ samples the random integers $x_i, y_i \leftarrow\!\$\, \mathbb{Z}_p$ and computes their secret keys as $\mathsf{sk}_i = (\mathsf{sk}_{i1}, \mathsf{sk}_{i2}) := (x_i, y_i)$ along with their corresponding verification keys $\mathsf{vk}_i = (\mathsf{vk}_{i1}, \mathsf{vk}_{i2}) := (g_2^{x_i}, g_2^{y_i})$.
– To generate the verification key of the honest parties P_k s.t. $k \in \mathcal{H}$, $\mathcal{H} = \mathcal{P} \setminus \mathcal{C}$, $\mathcal{B}_{alg}$ proceeds as follows:

1. For all $i \in \mathcal{T} := \mathcal{C} \cup \{0\}$ it computes the Lagrange polynomials evaluated at point k as below:
$$L_i^{\mathcal{T}}(k) = \frac{\prod_{j\in\mathcal{T}, j\neq i}(j-k)}{\prod_{j\in\mathcal{T}, j\neq i}(j-i)} \mod p \ . \tag{11}$$
2. To generate the verification key of an honest party P_k, it takes the verification keys of corrupted parties and the global verification key vk_0 and then it computes $\mathsf{vk}_k := (\mathsf{vk}_{k1}, \mathsf{vk}_{k2}) = (\prod_{i\in\mathcal{T}} \mathsf{vk}_{i1}^{L_i^{\mathcal{T}}(k)}, \prod_{i\in\mathcal{T}} \mathsf{vk}_{i2}^{L_i^{\mathcal{T}}(k)})$.

This completes the initialization phase by publishing the verification key of signers including honest and corrupted parties, $\vec{\mathsf{vk}}$, along with the global verification key vk_0.

Oracles: Like the previous game, $\mathcal{B}_{alg}$ provides the following oracles:

- **Random oracle**, $\mathsf{H}(id)$: It takes an index $id \in \mathbb{I}$ as input and call the $\mathcal{O}_0^{\mathrm{GPS}}$ oracle with id. The oracle returns $\mathcal{Q}_0[id]$ as output and $\mathcal{B}_{alg}$ delivers $\mathcal{Q}_{\mathsf{H}}[id] \leftarrow \mathcal{Q}_0[id]$ to the adversary.
- **Oracle** $\mathcal{O}_{\mathsf{C}}(j)$: For the list of corrupted parties $P_j, j \in \mathcal{C}$ s.t. $|\mathcal{C}| < t$, $\mathcal{B}_{alg}$ returns the secret key of each party P_j as the internal state to $\mathcal{A}_{alg}$ computed in the previous phase (static corruption).
- **Oracle** $\mathcal{O}_{\mathsf{PSign}}(k, M)$: For a given $M := (id, M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$ along with an honest party identifier $k \in \mathcal{H}$, $\mathcal{B}_{alg}$, if $(id, \star) \notin \mathcal{Q}_{\mathsf{S}}$ then it acts as follows:
  1. $\mathcal{B}$ queries id to the random oracle $\mathsf{H}(.)$ to obtain the basis of h. It then queries $\mathcal{O}_1^{\mathrm{GPS}_3}$ oracle under (h, M_1, M_2) and receives the signature $\sigma_0 = (h, s_0)$ as the response.
  2. For all corrupted parties $i \in \mathcal{C}$, $\mathcal{B}$ computes the partial signatures $\sigma_i = (h, s_i) = (h, h^{\mathsf{sk}_{i1}} M_1^{\mathsf{sk}_{i2}})$.
  3. For all $i \in \mathcal{T} := \mathcal{C} \cup \{0\}$, $\mathcal{B}_{alg}$ computes the Lagrange polynomials evaluated at point k the same as equation 11.
  4. It computes $\sigma_k = (h, s_k) = (h, \prod_{i\in\mathcal{T}} s_i^{L_i^{\mathcal{T}}(k)})$ and returns σ_k back.
  5. It then updates $\mathcal{S}_{(id,M_2)} = \mathcal{S}_{(id,M_2)} \cup \{k\}$ and $\mathcal{Q}_{\mathsf{S}} = \mathcal{Q}_{\mathsf{S}} \cup \{(id, M_2)\}$.

Forgery phase: In this step, $\mathcal{A}$ returns a tuple of forged signature (h^*, M_1^*, M_2^*, s^*) such that it passes the verification phase and $|\mathcal{S}_{(\star, M_2^*)} \cup \mathcal{C}| < t$. As we already define the verification key of the SPTS scheme the same as the GPS_3 problem instance then the challenger $\mathcal{B}_{alg}$ transfers the received forge from $\mathcal{A}_{alg}$ as a valid forgery to the underlying challenger of GPS_3 problem, then we have $\left|\Pr[G_0^{\mathcal{A}} = 1] - \Pr[G_1^{\mathcal{A}} = 1]\right| \leq \Pr[\mathrm{GPS}_3^{\mathcal{B}} = 1]$.

• **Game G_2**: This game is identical to the previous game, except the fact that the challenger $\mathcal{B}_{alg}$ in this game generates the public parameters based on the t-OMDL and GPS_3 assumptions instances and answers the oracles according to these assumptions as well.

Initialization: Let $(g_2^{r_0}, g_2^{v_0}) \in \mathbb{G}_2^2$ is the GPS_3 problem's instance, and $\zeta_1 = (g_2^{r_1}, \ldots, g_2^{r_t}) \in \mathbb{G}_2^t$ and $\zeta_2 = (g_2^{v_1}, \ldots, g_2^{v_t}) \in \mathbb{G}_2^t$ be the t-OMDL^1 and t-OMDL^2 instances, respectively. According to Theorem 4.1, the Pedersen DKG is $(t-1, t)$-OAAS secure, then

$\mathcal{B}_{alg}$ can build two algebraic simulators Sim_1 and Sim_2 in parallel that on a given t-OMDL^1 and t-OMDL^2 instances along with $t-1$ access to the discrete logarithm oracles, $\mathsf{Dlog}^1_{g_2}(.)$ and $\mathsf{Dlog}^2_{g_2}(.)$, acts as follows:

- Taken $(g_2^{r_0}, g_2^{v_0})$, ζ_1 and ζ_2, $\mathcal{B}_{alg}$ can interpolate two polynomials $F(x), G(x) \in \mathbb{Z}_p^t[X]$ of degree t with coefficients r_i and v_i for all $i \in [1,t]$ and constant terms of r_0 and v_0, respectively, i.e. $F(x) = r_0 + \sum_{i=1}^{t} r_i x^i$ and $G(x) = v_0 + \sum_{i=1}^{t} v_i x^i$.
- It then computes the shares as $\left(g_2^{r_0} \prod_{i=1}^{t} (g_2^{r_i})^{j^i}, g_2^{v_0} \prod_{i=1}^{t} (g_2^{v_i})^{j^i}\right) = \left(g_1^{\sum_{i=0}^{t-1} r_i \cdot j^i}, g_2^{\sum_{i=0}^{t} v_i \cdot j^i}\right) = \left(g_2^{F(j)}, g_2^{G(j)}\right)$, and issues the verification keys $\mathsf{vk}_j = (\mathsf{vk}_{j1}, \mathsf{vk}_{j2}) = \left(g_2^{F(j)}, g_2^{G(j)}\right)$, for all parties $j \in [1,n]$.
- It then computes the global verification key $\mathsf{vk}_0 = (\mathsf{vk}_{01}, \mathsf{vk}_{02}) = (g_2^{f(0)}, g_2^{g(0)}) = (g_2^{r_0}, g_2^{v_0})$ along with the set of verification keys $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n) = \left(\left(g_2^{f(1)}, g_2^{g(1)}\right), \ldots, \left(g_2^{f(n)}, g_2^{g(n)}\right)\right)$.

This completes the initialization phase by publishing the global verification key vk_0 and $\vec{\mathsf{vk}}$. We remark that since the Pedersen's DKG is OAAS secure then the distribution of public parameters in this game is identical to the previous one.

Oracles: By having access to the aforementioned instances, $\mathcal{B}_{alg}$ simulates the oracles by taking the advantage of existing oracles in the underlying assumptions as follows:

- **Random oracle**, $\mathsf{H}(id)$: It takes an index id as input and call the $\mathcal{O}_0^{\mathrm{GPS}}$ oracle with id. The oracle returns $\mathcal{Q}_0[id]$ as output and $\mathcal{B}_{alg}$ delivers $\mathcal{Q}_{\mathsf{H}}[id] \leftarrow \mathcal{Q}_0[id]$ to the adversary. W.l.o.g let the algebraic adversary $\mathcal{A}_{alg}$ before any partial signing query on an indexed DH message, $(id, M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$, it queries the random oracle $\mathsf{H}(.)$ with index id and it answers with h s.t. $\mathsf{dlog}_{g_2}(M_2) = \mathsf{dlog}_h(M_1) = m$.
- **Oracle** $\mathcal{O}_{\mathsf{C}}(j)$: The algebraic adversary $\mathcal{A}_{alg}$ has access to this oracle to corrupt up to $t-1$ parties by querying the index $j \in [1,n]$. To answer, $\mathcal{B}_{alg}$ queries the discrete logarithm oracles $\mathsf{Dlog}^1_{g_2}(g_2^{f(j)})$ and $\mathsf{Dlog}^2_{g_2}(g_2^{g(j)})$ and returns $\mathsf{sk}_j = (\mathsf{sk}_{j1}, \mathsf{sk}_{j2}) = (f(j), g(j)) = (x_j, y_j)$ as the internal state of the corrupted party P_j and updates $\mathcal{C} = \mathcal{C} \cup \{j\}$. $\mathcal{A}_{alg}$ can corrupt at most $t-1$ parties at any point of the game (adaptive corruption setting). For the simplicity, $\mathcal{B}_{alg}$ records the received responses from the discrete logarithm oracles in $\mathcal{Q}_{\mathcal{C}}[j] \leftarrow (f(j), g(j))$ that for all entries it is initialized by $\perp$.
- **Oracle** $\mathcal{O}_{\mathsf{PSign}}(k, M)$: For each given message $M := (id, M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}^{\mathsf{H}}$ along with an honest party identifier $k \in \mathcal{H}$, if $(id, \star) \notin \mathcal{Q}_{\mathsf{S}}$ then $\mathcal{B}_{alg}$ acts as follows:
  1. $\mathcal{B}$ queries id to the random oracle $\mathsf{H}(id)$ if $\mathcal{Q}_{\mathsf{H}}(id) = \perp$ then $\mathcal{B}$ aborts the query. Otherwise it queries $\mathcal{O}_1^{\mathrm{GPS}}(.)$ oracle under (h, M_1, M_2) and receives the signature $\sigma_0 = (h, s_0)$ as output. It then acts as follows:
     - $\mathcal{B}$ picks a random index $i' \in [1,t]$ s.t. $\mathcal{Q}_{\mathcal{C}}[i'] = \perp$ at the current point of the experiment, and then it continues querying to the discrete logarithm oracles

$\mathsf{Dlog}^1_{g_2}(.)$ and $\mathsf{Dlog}^2_{g_2}(.)$ for all parties in $\{P_1,\dots,P_t\}\setminus\{P_{i'}\}$ and updates $\mathcal{Q}_\mathcal{C}$. Note that after this phase, all indices in range $[1,t]$ are queried to the discrete logarithm oracles except the index i'.

- For all parties' identifier $i\in\mathcal{C}\setminus\{0,i'\}$, $\mathcal{B}$ computes the partial signatures $\left\{\sigma_i=(h,s_i)=(h,h^{f(i)}M_1^{g(i)})\right\}_{i\in\mathcal{C}\setminus\{0,i'\}}$.

2. For all $i\in\mathcal{T}:=\mathcal{C}\cup\{0\}$, it computes the Lagrange basis polynomials evaluated at point k the same as equation 11.
3. It then computes $\sigma_k=(h,s_k)=(h,\prod_{i\in\mathcal{C}\cup\{0\}}s_i^{L_i^\mathcal{T}(k)})$ and returns σ_k back as the partial signature generated by the party P_k.
4. It then updates $\mathcal{S}_{(id,M_2)}=\mathcal{S}_{(id,M_2)}\cup\{k\}$ and $\mathcal{Q}_\mathsf{S}=\mathcal{Q}_\mathsf{S}\cup\{(id,M_2)\}$.

Since the distribution of the provided public parameters and the oracles are the same as the previous game then we have $\Pr[G_1^\mathcal{A}=1]=\Pr[G_2^\mathcal{A}=1]$, and we can write, $\left|\Pr[G_0^\mathcal{A}=1]-\Pr[G_2^\mathcal{A}=1]\right|\le\Pr[\mathrm{GPS}_3^\mathcal{B}=1]$.

Forgery phase: In this step, $\mathcal{A}_{alg}$ returns a forged signature $\sigma^*=(h^*,s^*)$ on message $\tilde{M}^*:=(M_1^*,M_2^*)\in\mathcal{M}_{\mathsf{iDH}}$ s.t. it passes the verification phase and $|\mathcal{S}_{(\star,M_2^*)}\cup\mathcal{C}|<t$. We denote the event of σ^* being a valid signature on message (M_1^*,M_2^*) by $G_2^\mathcal{A}=1$. $\mathcal{A}_{alg}$ is an algebraic adversary and then she outputs the representation vectors, $\Big((h^*;\vec{h^*}),(M_1^*;\vec{M_1^*}),(M_2^*;\vec{M_2^*}),(s^*;\vec{s^*})\Big)$, and we have:

$$P_{h^*}[\vec{X}_1]=[h^*|g_1]+\sum_{\ell=1}^{q_h}\mathbf{R}_\ell[h^*|h_\ell]+\sum_{m_\ell\in\mathcal{Q}_\mathsf{S},j\in\mathcal{S}_{M_\ell}}\mathbf{R}_\ell[h^*|s_\ell](\mathbf{X_j}+m_\ell\mathbf{Y_j})\ , \tag{12a}$$

$$P_{M_1^*}[\vec{X}_1]=[f^*|g_1]+\sum_{\ell=1}^{q_h}\mathbf{R}_\ell[f^*|h_\ell]+\sum_{m_\ell\in\mathcal{Q}_\mathsf{S},j\in\mathcal{S}_{M_\ell}}\mathbf{R}_\ell[f^*|s_\ell](\mathbf{X_j}+m_\ell\mathbf{Y_j})\ , \tag{12b}$$

$$P_{M_2^*}[\vec{X}_2]=[M_2^*|g_2]+\sum_{\ell=0}^{n}\left(\mathbf{X}_\ell[M_2^*|X_\ell]+\mathbf{Y}_\ell[M_2^*|Y_\ell]\right)\ , \tag{12c}$$

$$P_{s^*}[\vec{X}_1]=[s^*|g_1]+\sum_{\ell=1}^{q_h}\mathbf{R}_\ell[s^*|h_\ell]+\sum_{m_\ell\in\mathcal{Q}_\mathsf{S},j\in\mathcal{S}_{M_\ell}}\mathbf{R}_\ell[s^*|s_\ell](\mathbf{X_j}+m_\ell\mathbf{Y_j})\ , \tag{12d}$$

where $\vec{X}_1=(\mathbf{X}_1,\dots,\mathbf{X}_{q_s+t-1},\mathbf{Y}_1,\dots,\mathbf{Y}_{q_s+t-1},\mathbf{R}_1,\dots,\mathbf{R}_{q_h})$ and $\vec{X}_2=(\mathbf{X}_0,\mathbf{X}_1,\dots,\mathbf{X}_n,\mathbf{Y}_0,\mathbf{Y}_1,\dots,\mathbf{Y}_n)$. Note that all these polynomials are defined based on the fact that the algebraic adversary has access to the public generators g_1,g_2 along with the global verification key $\mathsf{vk}_0=(g_2^{x_0},g_2^{y_0})$ and all n parties verification keys $\{\mathsf{vk}_i=(g_2^{x_0},g_2^{y_i})\}_{i=1}^n$ and also the received answers from random oracle $\{h_\ell=g_1^{r_\ell}\}_{\ell=1}^{q_h}$ and can compute $\{M_{1\ell}=g_1^{m_\ell r_\ell}\}_{\ell=1}^{q_h}$. It also has access to the answers received from the partial signing oracle under the index of honest and corrupted parties $j\in\mathcal{P}$, i.e. $\left\{s_{\ell,j}=g_1^{r_k(x_j+m_\ell y_j)}\right\}_{\ell=1,j=1}^{q_s,|\mathcal{S}_{M_\ell}|}$.

Assume id_{m_i} is the i^{th} query to the random oracle, $\mathsf{H}(.)$, and the index of the received challenge message is $i^* \in [1, q_h]$ (as we discussed the adversary has queried the random oracle before the forgery phase). Let the sampled randomnesses on indices i and index i^* are denoted by r_i and r^*, i.e. $\mathcal{Q}_\mathsf{H}[id_{m^*}] \leftarrow g_1^{r^*}$ and $\mathcal{Q}_\mathsf{H}[id_{m_i}] \leftarrow g_1^{r_i}$, respectively. We split the answers from random oracle and partial signing oracle to the queries of challenge index id_{m^*} and challenge message $\tilde{M}^*$ and since we are assuming that the adversary can successfully forge then from $s^* = h^{*^{(x+m^*y)}}$, we can write:

$$\begin{aligned}\mathbf{R}^*(\mathbf{X} + m^*\mathbf{Y}) = \overbrace{[s^*|g_1] + \sum_{\ell=1}^{q_h \setminus \{i^*\}} \mathbf{R}_\ell [s^*|h_\ell]}^{(\blacklozenge)} + \mathbf{R}^*[s^*|h^*] + \\ \underbrace{\sum_{m_\ell \in \mathcal{Q}_\mathsf{S}, j \in \mathcal{S}_{M_\ell}} \mathbf{R}_\ell [s^*|s_\ell](\mathbf{X_j} + m_\ell \mathbf{Y_j})}_{(\clubsuit)} + \sum_{j \in \mathcal{S}_{M_\ell}} \mathbf{R}^*[s_j^*|s^*](\mathbf{X_j} + m^*\mathbf{Y_j}) \ .\end{aligned} \tag{13}$$

As we already assumed that the adversary wins the forgery game with a non-negligible advantage and also the defined random oracle is collision resistance (we will discuss the probability of this event in game $\boldsymbol{G}_4$), then neither ($\clubsuit$) nor ($\blacklozenge$) parts of the equation include the term $\mathbf{R}^*$ and we can write $(\mathbf{X} + m^*\mathbf{Y}) = [s^*|h^*] + \sum_{j=1}^{n} [s_j^*|s^*](\mathbf{X_j} + m^*\mathbf{Y_j})$. Let $(\mathbf{X} + m^*\mathbf{Y})$ and $(\mathbf{X_j} + m^*\mathbf{Y_j})$ are denoted by x and x_j, respectively. We can define an event E as the event that $x \neq [s^*|h^*] + \sum_{j=1}^{n} ([s_j^*|s^*]x_j)$.

Lemma B.1. *Let $G_2^{\mathcal{A}} = 1$ denotes the event that the forged signature σ^* in game $\boldsymbol{G}_2$ on message $(M_1^*, M_2^*) \in \mathcal{M}_{\mathsf{iDH}}$ be valid and event E is defined as above. Then there are two PPT algebraic adversaries $\mathcal{A}_{alg}^1$ and $\mathcal{A}_{alg}^2$ playing either t-OMDL^1 or t-OMDL^2 (we denote both with t-OMDL) s.t.* $\Pr[t\text{–}\mathsf{OMDL}^{\mathcal{A}_1} = 1] = \Pr[G_4^{\mathcal{A}} = 1 \wedge \neg E]$ *and* $\Pr[t\text{–}\mathsf{OMDL}^{\mathcal{A}_2} = 1] \geq (1 - 1/p) \cdot \Pr[G_4^{\mathcal{A}} = 1 \wedge E]$.

Proof. The proof of this Lemma is the same as [BL22, Lemma 4.2]. □

Thus $\mathcal{B}_{alg}$ as an attacker against the t-OMDL picks $j^* \leftarrow\!\$\, \{1, 2\}$ and then internally emulates $\mathcal{A}_{alg}^{j^*}$ and for a $p \geq 2$ we can write:

$$\begin{aligned}&\Pr[t\text{-}\mathsf{OMDL}^{\mathcal{B}} = 1] = \\ &\Pr[t\text{-}\mathsf{OMDL}^{\mathcal{A}_1} = 1 \mid j^* = 1] \cdot \Pr[j^* = 1] + \Pr[t\text{-}\mathsf{OMDL}^{\mathcal{A}_2} = 1 \mid j^* = 2] \cdot \Pr[j^* = 2] = \\ &1/2 \left(\Pr[t\text{-}\mathsf{OMDL}^{\mathcal{A}_1} = 1] + \Pr[t\text{-}\mathsf{OMDL}^{\mathcal{A}_2} = 1]\right) \geq \\ &1/2 \left(1 - \frac{1}{p}\right) \left(\Pr[G_2^{\mathcal{A}} = 1 \wedge E] + \Pr[G_2^{\mathcal{A}} = 1 \wedge \neg E]\right) \geq 1/4 \Pr[G_2^{\mathcal{A}} = 1] \ .\end{aligned} \tag{14}$$

• **Game $\boldsymbol{G}_3$**: Let the set of corrupted parties before the first query to the partial signing oracle is denoted by $\mathcal{C}^{\leftarrow}$ and $\mathcal{C}^{\rightarrow}$ denotes the list corrupted parties after this point of experiment s.t. $\mathcal{C} = \mathcal{C}^{\leftarrow} \cup \mathcal{C}^{\rightarrow}$. This game is identical to the previous game, except the challenger in the first partial signing oracle query samples an index $i' \in [1, n] \backslash \mathcal{C}^{\leftarrow}$ uniformly at random. Once the adversary $\mathcal{A}_{alg}$ queries the corruption oracle on the index i' then $\mathcal{B}_{alg}$ aborts and $\mathcal{A}_{alg}$ loses the game. In a worse case, we assume the adversary does not query the corruption oracle before querying the partial signing oracle for the first time, i.e. $\mathcal{C}^{\leftarrow} = \emptyset$. The probability of this event is given by the inverse of the number of $(t-1)$-element combinations of n object taken without repetition. Thus we can write:

$$\Pr[G_3^{\mathcal{A}} = 1] \leq \left(1 - \left(\frac{(t-1)!\,(n-t+1)!}{n!}\right)\right) \Pr[G_2^{\mathcal{A}} = 1] \ . \tag{15}$$

As a simple example, for a full threshold setting, $t = n$, we have, $\Pr[G_3^{\mathcal{A}} = 1] \leq (1 - (1/n)) \Pr[G_2^{\mathcal{A}} = 1]$.

• **Game $\boldsymbol{G}_4$**: This game is the same as the previous game except once there are two distinct indices id_{m_1} and id_{m_2} s.t. $\mathsf{H}(id_{m_1}) = \mathsf{H}(id_{m_2})$ then $\mathcal{B}_{alg}$ aborts and $\mathcal{A}_{alg}$ loses the game. The probability of this event is $\Pr[G_4^{\mathcal{A}} = 1] \leq \Pr[G_3^{\mathcal{A}} = 1] - q_h^2/p$. Then we can write:

$$\Pr[G_4^{\mathcal{A}} = 1] \leq \left(1 - \left(\frac{(t-1)!\,(n-t+1)!}{n!}\right)\right) \Pr[G_2^{\mathcal{A}} = 1] - q_h^2/p \ . \tag{16}$$

From equation 14 we have:

$$\Pr[\boldsymbol{G}_4^{\mathcal{A}} = 1] \leq 4\left(1 - \left(\frac{(t-1)!\,(n-t+1)!}{n!}\right)\right) \Pr[t\text{-}\mathsf{OMDL}^{\mathcal{B}} = 1] - q_h^2/p \ . \tag{17}$$

In addition, as the distribution of output in games $\boldsymbol{G}_2^{\mathcal{A}}$, $\boldsymbol{G}_3^{\mathcal{A}}$ and $\boldsymbol{G}_4^{\mathcal{A}}$ are identical and the verification key of the SPTS scheme is the same as the GPS_3 instance (MI-SPS construction) then $\mathcal{B}_{alg}$ transfers the received forgery as a valid forgery to the Challenger of GPS_3 problem (challenger of MI-SPS scheme). Thus we can write:

$$\begin{aligned}
&\Pr[G_0^{\mathcal{A}} = 1] \leq \Pr[\text{GPS}_3^{\mathcal{B}} = 1] + \Pr[\boldsymbol{G}_2^{\mathcal{A}} = 1] \Rightarrow \\
&Adv_{\mathcal{A}}^{\mathsf{TSPS}}(\lambda) \leq \varepsilon_2 + 4\left(1 - \left(\frac{(t-1)!\,(n-t+1)!}{n!}\right)\right) \varepsilon_1 - q_h^2/p \ .
\end{aligned} \tag{18}$$

□