Published in *Theoretical Computer Science*, **645**: 1–24, 2016.

Born and raised distributively: Fully distributed non-interactive adaptively-secure threshold signatures with short shares[☆]

Benoît Libert

Ecole Normale Supérieure de Lyon, Lyon, France

Marc Joye

Technicolor and Ecole normale supérieure, Los Altos, CA, USA

Moti Yung

Snapchat and Columbia University, New York, NY, USA

Abstract

Threshold cryptography is a fundamental distributed computational paradigm for enhancing the availability and the security of cryptographic public-key schemes. It does it by dividing private keys into n shares handed out to distinct servers. In threshold signature schemes, a set of at least $t+1 \leq n$ servers is needed to produce a valid digital signature. Availability is assured by the fact that any subset of $t+1$ servers can produce a signature when authorized. At the same time, the scheme should remain robust (in the fault tolerance sense) and unforgeable (cryptographically) against up to t corrupted servers; *i.e.*, it adds quorum control to traditional cryptographic services and introduces redundancy. Originally, most practical threshold signatures have a number of demerits: They have been analyzed in a static corruption model (where the set of corrupted servers is fixed at the very beginning of the attack); they require interaction; they assume a trusted dealer in the key generation phase (so that the system is not fully distributed); or they suffer from certain overheads in terms of storage (large share sizes).

In this paper, we construct practical *fully distributed* (the private key is born distributed), non-interactive schemes —where the servers can compute their partial signatures without communication with other servers— with adaptive security (*i.e.*, the adversary corrupts servers dynamically based on its full view of the history of the system). Our schemes are very efficient in terms of computation, communication, and scalable storage (with private key shares of size $O(1)$, where certain solutions incur $O(n)$ storage costs at each server). Unlike other adaptively secure schemes, our schemes are erasure-free (reliable erasure is hard to assure and hard to administer properly in actual systems). To the best of our knowledge, such a fully distributed highly constrained scheme has been an open problem in the area. In particular, and of special interest, is the fact that Pedersen's traditional distributed key generation (DKG) protocol can be safely employed in the initial key generation phase when the system is born —although it is well-known not to ensure uniformly distributed public keys. An advantage of this is that this protocol only takes one round optimistically (in the absence of faulty player).

Keywords: Threshold signatures, fully distributed schemes, non-interactivity, adaptive security, efficiency, availability, fault tolerance, distributed key generation, erasure-freeness.

[☆] This is the full version of a paper published at PODC 2014. It includes all proofs that were not included in the proceedings version.

Email addresses: `benoit.libert@gmail.com` (Benoît Libert), `marc.joye@gmail.com` (Marc Joye), `moti@cs.columbia.edu` (Moti Yung)

1. Introduction

Threshold cryptography [29, 30, 15, 28] is a paradigm where cryptographic keys are divided into $n > 1$ shares to be stored by distinct servers, which increases the system's availability and resilience to failures. In (t, n)-threshold cryptosystems, private-key operations require the cooperation of at least $t + 1$ out of n servers (any subset is good). By doing so, the system remains secure against adversaries that break into up to t servers. (The mechanism can be viewed as extending Shamir's secret sharing of one value [66] to sharing of the capability to apply cryptographic function efficiently.) The public-key portion of the function (*e.g.*, signature verification key) does not change from its usual format.

Threshold primitives are widely used in distributed protocols. Threshold homomorphic encryption schemes are utilized in voting systems (see, *e.g.*, [22, 23]) and multiparty computation protocols [24]. Threshold signatures enhance the security of highly sensitive private keys, like those of certification authorities (*e.g.*, [18]). They can also serve as tools for distributed storage systems [48, 64]. RSA and Elgamal-type constructions have been at the core of many threshold protocols the last two decades (see, *e.g.*, [28, 39, 40, 47]). A *fully distributed* public-key system is one where the public (and the distributed private) keys are jointly generated by the same servers which end up holding the private key's shares (*e.g.*, via a threshold secret sharing [66]). Efficient distributed key generation (DKG) protocols were put forth for both RSA [12, 34, 33, 26] and discrete-logarithm-based systems [61, 41, 35, 16, 43].

Non-interactive threshold signatures. For a long time, RSA-based threshold signatures have been the only solutions to enable non-interactive distributed signature generation. By "non-interactive", we mean that each server can compute its own partial signature without any online conversation with other servers: each server should send a single message to an entity, called *combiner*, which gathers the signature shares so as to obtain a full signature. Unlike threshold versions of Schnorr and DSA signatures [42, 39], threshold RSA signatures are well-suited to non-interactive signing protocols as they are deterministic. Hence, they do not require the servers to jointly generate a randomized signature component in a first round before starting a second round. Practical robust non-interactive threshold signatures were described by Shoup [67] under the RSA assumption and by Katz and Yung [50] assuming the hardness of factoring. Boldyreva [10] showed a threshold version of Boneh-Lynn-Shacham signatures [14], which provided an alternative non-interactive scheme with robustness and short signatures. The latter construction [10] was subsequently generalized by Wee [68]. These solutions are only known to resist static attacks, where the set of corrupted servers is chosen by the adversary at the very beginning of the attack, before even seeing the public key.

Adaptive corruptions. More realistically than the static model, the adaptive corruption model allows adversaries to choose whom to corrupt at any time, based on their entire view so far. Adaptive adversaries are known to be strictly (see, *e.g.*, [25]) stronger. The first adaptively secure threshold signatures were independently described in 1999 by Canetti *et al.* [16] and by Frankel *et al.* [35, 36]. These constructions rely on a technique, called "single inconsistent player" (SIP), which inherently requires interaction. The SIP technique basically consists in converting a t-out-of-n secret sharing into a t-out-of-t secret sharing in such a way that, in the latter case, there is only one server whose internal state cannot be consistently revealed to the adversary. Since this player is chosen at random by the simulator among the n severs, it is only corrupted with probability less than $1/2$ and, upon this undesirable event, the simulator can simply rewind the adversary back to one of its previous states. After this backtracking operation, the simulator uses different random coins to simulate the view of the adversary, hoping that the inconsistent player will not be corrupted again (and the expected number of rewinding-s is bounded by 2).

Jarecki and Lysyanskaya [49] extended the SIP technique in order to eliminate the need for servers to reliably erase intermediate computation results. However, their adaptively secure version of the Canetti-Goldwasser threshold cryptosystem [17] requires a substantial amount of interaction at each private-key operation. The same holds for the adaptively secure threshold signatures of Lysyanskaya and Peikert [56] and the universally composable protocols of Abe and Fehr [1].

In 2006, Almansa, Damgård and Nielsen [4] showed a variant of Rabin's threshold RSA signatures [63] and proved them adaptively secure using the SIP technique and ideas from [35, 36]. Similar techniques were used in [69] to construct adaptively secure threshold Waters signatures [70]. While the SIP technique provides adaptively secure threshold signatures based on RSA or the Diffie-Hellman assumption, these fall short of minimizing the amount of interaction. The constructions of [35, 36] proceed by turning a (t, n) polynomial secret sharing into a (t, t) additive secret sharing by first selecting a pool of at least t participants. However, if only one of these fails to provide a valid contribution to the signing process, the whole protocol must be restarted from scratch. The protocol of Almansa *et al.* [4] is slightly different in that; like [63], it proceeds by sharing an RSA private key in an additive (n, n) fashion (*i.e.*, the private RSA exponent d is split into shares $d_1, \ldots, d_n$ such that $d = \sum_{i=1}^{n} d_i$). In turn, each additive share d_i is shared in a (t, n) fashion using a polynomial verifiable secret sharing and each share $d_{i,j}$ of d_i is distributed to another server j. This is done in such a way that, if one participant fails to provide a valid RSA signature share $H(M)^{d_i}$, the missing signature share can be re-constructed by running the reconstruction algorithm of the verifiable secret sharing scheme that was used to share d_i. The first drawback of this approach is that it is only non-interactive when all players are honest as a second round is needed to reconstruct missing multiplicative signature shares $H(M)^{d_i}$. Another disadvantage is that players have to store $\Theta(n)$ values, where n is the number of servers, as each player has to store a polynomial share of other players' additive share. Ideally, we would like a solution where each player only stores $O(1)$ elements, regardless of the number of players.

Recently, Libert and Yung [54, 55] gave several constructions of adaptively secure threshold encryption schemes with chosen-ciphertext security. They also suggested an adaptively secure and non-interactive threshold variant [54] of a signature scheme due to Lewko and Waters [51]. The use of bilinear maps in composite order groups makes the scheme of [54] expensive when it comes to verifying signatures: as discussed by Freeman [38], computing a bilinear map in composite order groups is at least 50 times slower than evaluating the same bilinear map in prime-order groups at the 80-bit security level (things can only get worse at higher security levels). The techniques of Lewko [52] can be used to adapt the construction of [54] to the setting of prime-order groups. In the resulting construction, each signature consists of 6 group elements. The use of asymmetric bilinear maps (see [19]) allows reducing the signature size to 4 group elements. Unfortunately, the techniques of [52, 19] assume a trusted dealer and, if implemented in a distributed manner, their key generation phase is likely to be communication-expensive (resorting to generic multiparty secure computations). In particular, they seem hardly compatible with a round-optimal DKG protocol. The reason is that [19] requires to generate public keys containing pairs of matrices of the form $g^{\mathbf{A}} \in \mathbb{G}^{n \times n}$ and $g^{\mathbf{A}^{-1}} \in \mathbb{G}^{n \times n}$, for some matrix $\mathbf{A} \in \mathbb{Z}_p^{n \times n}$, and it is not clear how these non-linear operations can be achieved in a round-optimal distributed manner (let alone with adaptive security). Finally, the solutions of [54] require reliable erasures due to the use of the dual system encryption technique [71, 51] and the existence of several distributions of partial signatures.

Our contributions. We consider the problem of devising a fully distributed, non-interactive, robust, and adaptively secure construction which is as efficient as the centralized schemes obtained from [54, 19] and does not rely on erasures. In particular, we want to retain private-key shares of $O(1)$ size, no matter how many players are involved in the protocol. Here, "fully distributed" implies that the public key is jointly generated by all players —so that no trusted dealer is needed— while guaranteeing the security of the scheme against an adaptive adversary. As mentioned above, we wish to avoid the costly and hard-to-control use of reliable erasures. This means that, whenever the adversary corrupts a player, it learns the entire history of that player.

At the same time, the distributed key generation phase should be as communication-efficient as possible. Ideally, a single communication round should be needed when the players follow the protocol. Finally, we would like to avoid interaction during the distributed signing process. To the best of our knowledge, no existing solution combines all the aforementioned highly constraining properties. We thus provide the first candidates.

Our constructions are derived from linearly homomorphic structure-preserving signatures (LHSPS). As defined by Abe *et al.* [2, 3], structure-preserving signatures (SPS) are signature schemes where messages and public keys live in an abelian group over which a bilinear map is efficiently computable. Recently, Libert *et al.* [53] considered SPS schemes with additive homomorphic properties: given signatures on linearly independent vectors of group elements, anyone can publicly compute a signature on any linear combination of these vectors. In order to sign a message $M \in \{0, 1\}^*$ in a distributed manner, our idea is to hash M onto a vector of group elements which is signed using the LHSPS scheme of [53]. In the random oracle model, we prove that the resulting system is a secure digital signature even if the underlying LHSPS scheme satisfies a weak security definition. Since the LHSPS signing algorithm is deterministic and further presents certain homomorphic properties over the key space, the resulting signature is also amenable for non-interactive distributed signature generation. In the threshold setting, we take advantage of specific properties of the LHSPS scheme of [53] to prove that the scheme provides security against adaptive corruptions in the absence of secure erasures.

More surprisingly, we prove that the scheme remains adaptively secure if the public key is generated using Pedersen's DKG protocol [61]. The latter basically consists in having all players verifiably share a random value using Feldman's verifiable secret sharing (VSS) [32] before computing the shared secret as the sum of all well-behaved players' contributions. While very efficient (as only one round is needed in the absence of faulty players), this protocol is known [41] *not* to guarantee the uniformity of the resulting public key. Indeed, even a static adversary can bias the distribution by corrupting only two players. Nonetheless, the adversary does not have much control on the distribution of the public key and Pedersen's protocol can still be safely used in some applications, as noted by Gennaro *et al.* [42, 43]. For example, it was recently utilized by Cortier *et al.* [21] in the context of voting protocols. However, these safe uses of Pedersen's protocol were in the static corruption setting and our scheme turns out to be its first application in an adaptive corruption model. To our knowledge, it is also the first adaptively secure threshold signature where the DKG phase takes only one round when all players follow the specifications.

As an extension of our first scheme, we describe a variant supporting signature aggregation: as suggested by Boneh *et al.* [13], a set of n signatures for distinct public keys $PK_1, \ldots, PK_s$ on messages $M_1, \ldots, M_s$ can be aggregated into a single signature σ which convinces a verifier that, for each i, M_i was signed by the private key underlying PK_i. In the threshold setting, this property allows for de-centralized certification authorities while enabling the compression of certification chains.

As a final contribution, we give a non-interactive adaptively secure threshold signature scheme in the standard model that retains all the useful properties (including the erasure-freeness) of our first realization. In particular, Pedersen's protocol can still be used in the key generation phase if a set of uniformly random common parameters —which can be shared by many public keys— is set up beforehand. As is natural for standard-model constructions, this scheme is somewhat less efficient than its random-oracle-based counterpart but it remains sufficiently efficient for practical applications.

Like Gennaro *et al.* [43] and Cortier *et al.* [21], we prove security via a direct reduction from the underlying number-theoretic assumption instead of reducing the security of our schemes to that of their centralized version. We emphasize that our proof technique is different from those of [43, 21], where the reduction runs Pedersen's DKG protocol on behalf of honest players and embeds a discrete logarithm instance in the contribution of honest players to the public key, using a proper simulation of Feldman's verifiable secret sharing. In the adaptive corruption setting, this would at least require an adaptively secure variant of Feldman's VSS, such as [1], and thus extra communications. Instead, our reduction always faithfully runs the protocol on behalf of honest players, and thus always knows their internal state so as to perfectly answer corruption queries. Yet, we can use the adversary's forgery to break the underlying hardness assumption by taking advantage of key homomorphic properties of the scheme, which allow us to turn a forgery for the jointly generated public key —of possibly skewed distribution— into a forgery for some uniformly random key.

2. Background

2.1. Definitions for threshold signatures

A non-interactive (t, n)-threshold signature scheme consists of a tuple $\Sigma = (\mathsf{Dist\text{-}Keygen}, \mathsf{Share\text{-}Sign}, \mathsf{Share\text{-}Verify}, \mathsf{Verify}, \mathsf{Combine})$ of efficient algorithms or protocols such that:

$\mathsf{Dist\text{-}Keygen}(\mathsf{params}, \lambda, t, n)$: This is an interactive protocol involving n players $P_1, \ldots, P_n$, which all take as input common public parameters params, a security parameter $\lambda \in \mathbb{N}$ as well as a pair of integers $t, n \in \mathsf{poly}(\lambda)$ such that $1 \leq t \leq n$. The outcome of the protocol is the generation of a public key PK, a vector of private key shares $\mathbf{SK} = (SK_1, \ldots, SK_n)$ where P_i only obtains SK_i for each $i \in \{1, \ldots, n\}$, and a public vector of verification keys $\mathbf{VK} = (VK_1, \ldots, VK_n)$.

$\mathsf{Share\text{-}Sign}(SK_i, M)$: is a possibly randomized algorithm that takes in a message M and a private key share SK_i. It outputs a signature share σ_i.

$\mathsf{Share\text{-}Verify}(PK, \mathbf{VK}, M, (i, \sigma_i))$: is a deterministic algorithm that takes as input a message M, the public key PK, the verification key $\mathbf{VK}$ and a pair (i, σ_i) consisting of an index $i \in \{1, \ldots, n\}$ and signature share σ_i. It outputs 1 or 0 depending on whether σ_i is deemed as a valid signature share or not.

Combine$(PK, \mathbf{VK}, M, \{(i, \sigma_i)\}_{i \in S})$: takes as input a public key PK, a message M and a subset $S \subset \{1, \ldots, n\}$ of size $|S| = t + 1$ with pairs $\{(i, \sigma_i)\}_{i \in S}$ such that $i \in \{1, \ldots, n\}$ and σ_i is a signature share. This algorithm outputs either a full signature σ or $\perp$ if $\{(i, \sigma_i)\}_{i \in S}$ contains ill-formed partial signatures.

Verify(PK, M, σ): is a deterministic algorithm that takes as input a message M, the public key PK and a signature σ. It outputs 1 or 0 depending on whether σ is deemed valid share or not.

We shall use the same communication model as in, *e.g.*, [41, 42, 43], which is partially synchronous. Namely, communications proceed in synchronized rounds and sent messages are always received within some time bound in the same round. All players have access to a public broadcast channel, which the adversary can use as a sender and a receiver. However, the adversary cannot modify messages sent over this channel, nor prevent their delivery. In addition, we assume private and authenticated channels between all pairs of players.

In the adaptive corruption setting, the security of non-interactive threshold signatures can be defined as follows.

Definition 1. *A non-interactive threshold signature scheme Σ is* adaptively secure against chosen-message attacks *if no probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ has non-negligible advantage in the game hereunder. At any time, we denote by $C \subset \{1, \ldots, n\}$ and $\mathcal{G} := \{1, \ldots, n\} \setminus C$ the dynamically evolving subsets of corrupted and honest players, respectively. Initially, we set $C = \emptyset$.*

1. *The game begins by running the DKG protocol* Dist-Keygen(params, λ, t, N) *during which the challenger plays the role of honest players P_i and the adversary $\mathcal{A}$ is allowed to corrupt players at any time. When $\mathcal{A}$ chooses to corrupt a player P_i, the challenger sets $\mathcal{G} = \mathcal{G} \setminus \{i\}$, $C = C \cup \{i\}$ and returns the internal state of P_i. Moreover, $\mathcal{A}$ is allowed to act on behalf of P_i from this point forward. The protocol ends with the generation of a public key PK, a vector of private key shares* $\mathbf{SK} = (SK_1, \ldots, SK_n)$ *and the corresponding verification keys* $\mathbf{VK} = (VK_1, \ldots, VK_n)$. *At the end of this phase, the public key PK and $\{SK_i\}_{i \in C}$ are available to the adversary $\mathcal{A}$.*
2. *On polynomially many occasions, $\mathcal{A}$ adaptively interleaves two kinds of queries.*
   - *Corruption query: At any time, $\mathcal{A}$ can choose to corrupt a server. To this end, $\mathcal{A}$ chooses $i \in \{1, \ldots, n\}$ and the challenge returns SK_i before setting $\mathcal{G} = \mathcal{G} \setminus \{i\}$ and $C = C \cup \{i\}$.*
   - *Signing query: For any $i \in \mathcal{G}$, $\mathcal{A}$ can also submit a pair (i, M) and ask for a signature share on an arbitrary message M on behalf of player P_i. The challenger responds by computing $\sigma_i \leftarrow$* Share-Sign(SK_i, M) *and returning σ to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a message $M^\star$ and a signature $\sigma^\star$. We define $\mathcal{V} = C \cup \mathcal{S}$, where $\mathcal{S} \subset \{1, \ldots, n\}$ is the subset of players for which $\mathcal{A}$ made a signing query of the form $(i, M^\star)$. The adversary wins if the following conditions hold: (i) $|\mathcal{V}| < t + 1$; (ii)* Verify$(PK, M^\star, \sigma^\star) = 1$.

$\mathcal{A}$'s advantage is defined as its probability of success, taken over all coin tosses.

Since we focus on non-interactive schemes, Definition 1 allows the adversary to individually query each partial signing oracle whereas usual definitions only provide the adversary with an oracle that runs the distributed signing protocol on behalf of all honest players. We also remark that Definition 1 allows the adversary to obtain some partial signatures on the forgery message $M^\star$ as long as its output remains a non-trivial forgery. In a weaker (but still compelling) definition, partial signing queries for $M^\star$ would be completely disallowed. In the following, we will stick to the stronger definition.

2.2. Hardness assumptions

We first recall the definition of the Decision Diffie-Hellman problem.

Definition 2. *In a cyclic group $\mathbb{G}$ of prime order p, the* Decision Diffie-Hellman Problem *(DDH) in $\mathbb{G}$, is to distinguish the distributions (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c), with $a, b, c \overset{R}{\leftarrow} \mathbb{Z}_p$. The Decision Diffie-Hellman assumption is the intractability of DDH for any PPT distinguisher.*

We use bilinear maps $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ over groups of prime order p. We will work in asymmetric pairings, where we have $\mathbb{G} \neq \hat{\mathbb{G}}$ so as to allow the DDH assumption to hold in $\mathbb{G}$ (see, *e.g.*, [65]). In certain asymmetric pairing configurations, DDH is even believed to hold in both $\mathbb{G}$ and $\hat{\mathbb{G}}$. This assumption is called *Symmetric eXternal Diffie-Hellman* (SXDH) assumption and it implies that no isomorphism between $\hat{\mathbb{G}}$ and $\mathbb{G}$ be efficiently computable.

For convenience, we also use the following problem in asymmetric pairing configurations.

Definition 3 ([3]). *The* Double Pairing problem *(DP) in $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ is, given $(\hat{g}_z, \hat{g}_r) \in_R \hat{\mathbb{G}}^2$, to find a pair $(z, r) \in \mathbb{G}^2 \setminus \{(1_{\mathbb{G}}, 1_{\mathbb{G}})\}$ that satisfies $e(z, \hat{g}_z) \cdot e(r, \hat{g}_r) = 1_{\mathbb{G}_T}$. The Double Pairing assumption asserts that the DP problem is infeasible for any PPT algorithm.*

The DP problem is known [3] to be at least as hard as DDH in $\hat{\mathbb{G}}$. Given $(\hat{g}_z, \hat{g}_r, \hat{g}_z{}^{\theta_1}, \hat{g}_r{}^{\theta_2})$, a solution (z, r) allows deciding whether $\theta_1 = \theta_2$ or not by testing if the equality $e(z, \hat{g}_z{}^{\theta_1}) \cdot e(r, \hat{g}_r{}^{\theta_2}) = 1_{\mathbb{G}_T}$ holds.

2.3. Linearly homomorphic structure-preserving signatures

Structure-preserving signatures [2, 3] are signature schemes that allow signing elements of an abelian group while preserving their algebraic structure, without hashing them first. In [53], Libert *et al.* described structure-preserving signatures with linearly homomorphic properties. Given signatures on several vectors $\vec{M}_1, \ldots, \vec{M}_n$ of group elements, anyone can publicly derive a signature on any linear combination of $\vec{M}_1, \ldots, \vec{M}_n$. They suggested the following scheme, which is a one-time LHSPS (namely, it only allows signing one linear subspace) based on the DP assumption.

Keygen(λ, N): Given a security parameter λ and the dimension $N \in \mathbb{N}$ of the subspace to be signed, choose bilinear group $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$. Then, choose $\hat{g}_z, \hat{g}_r \overset{R}{\leftarrow} \hat{\mathbb{G}}$. For $k = 1$ to N, pick $\chi_k, \gamma_k \overset{R}{\leftarrow} \mathbb{Z}_p$ and compute $\hat{g}_k = \hat{g}_z{}^{\chi_k} \hat{g}_r{}^{\gamma_k}$. The private key is $\mathsf{sk} = \{\chi_k, \gamma_k\}_{i=1}^N$ while the public key consists of $\mathsf{pk} = (\hat{g}_z,\ \hat{g}_r,\ \{\hat{g}_k\}_{k=1}^N)$.

Sign$(\mathsf{sk}, (M_1, \ldots, M_N))$: To sign a vector $(M_1, \ldots, M_N) \in \mathbb{G}^N$ using $\mathsf{sk} = \{\chi_k, \gamma_k\}_{k=1}^N$, compute and output $\sigma = (z, r) \in \mathbb{G}^2$, where $z = \prod_{k=1}^N M_k^{-\chi_k}$ and $r = \prod_{k=1}^N M_k^{-\gamma_k}$.

SignDerive$(\mathsf{pk}, \{(\omega_i, \sigma^{(i)})\}_{i=1}^\ell)$: Given the public key pk and ℓ tuples $(\omega_i, \sigma^{(i)})$, parse $\sigma^{(i)}$ as $\sigma^{(i)} = (z_i, r_i) \in \mathbb{G}^3$ for $i = 1$ to ℓ. Then, compute and return $\sigma = (z, r)$, where $z = \prod_{i=1}^\ell z_i^{\omega_i}$ and $r = \prod_{i=1}^\ell r_i^{\omega_i}$.

Verify$(\mathsf{pk}, \sigma, (M_1, \ldots, M_N))$: Given a purported signature $\sigma = (z, r) \in \mathbb{G}^2$ and a vector $(M_1, \ldots, M_N)$, return 1 if and only if $(M_1, \ldots, M_N) \neq (1_{\mathbb{G}}, \ldots, 1_{\mathbb{G}})$ and (z, r) satisfies $1_{\mathbb{G}_T} = e(z, \hat{g}_z) \cdot e(r, \hat{g}_r) \cdot \prod_{k=1}^N e(M_k, \hat{g}_k)$.

A useful property of the scheme is that, if the DP assumption holds, it is computationally hard to come up with two distinct signatures on the same vector, *even* if the private key is available.

3. A practical adaptively secure non-interactive threshold signature

The construction notably relies on the observation that, as shown in Appendix Appendix C.1, any one-time linearly homomorphic SPS can be turned into a fully secure ordinary signature by introducing a random oracle. The public key is simply that of a linearly homomorphic SPS for vectors of dimension $n > 1$. Messages are signed by hashing them to a vector $\vec{H} \in \mathbb{G}^n$ and generating a one-time homomorphic signature on $\vec{H}$. The security reduction programs the random oracle in such a way that all signed messages are hashed into a proper subspace of $\mathbb{G}^n$ whereas, with some probability, the adversary forges a signature on a message which is hashed outside this subspace. Hence, a forgery for this message translates into an attack against the underlying linearly homomorphic SPS.

In the threshold setting, our system can be seen as an adaptively secure variant of Boldyreva's threshold signature [10], which builds on the short signatures of Boneh, Lynn, and Shacham [14].

The DKG phase uses Pedersen's protocol [61] (or, more precisely, a variant with two generators). Each player verifiably shares a random secret using Pedersen's verifiable secret sharing [62] —where verification is enabled by having all parties broadcast commitments to their secret polynomials— and the final secret key is obtained by summing up the shares of non-disqualified players. When all parties follow the protocol, a single communication round is needed. Moreover, we do not need to rely on zero-knowledge proofs or reliable erasures at any time.

In order to sign a message using his private key share, each player first hashes the message M to obtain a vector $(H_1, H_2) \in \mathbb{G}^2$ of two group elements, which can be signed using the linearly homomorphic structure-preserving signature of Section 2.3. We actually build on the observation that any one-time linearly homomorphic SPS implies a fully secure digital signature in the random oracle model. In the threshold setting, we take advantage of two specific properties in the underlying homomorphic signature. First, it is also key homomorphic[1] and thus amenable for non-interactively distributing the signing process. Second, in the security proof of [53], the reduction always knows the private key, which allows consistently answering adaptive corruption queries.

3.1. Description

In the description below, we assume that all players agree on public parameters params consisting of asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with generators $\hat{g}_z, \hat{g}_r \in_R \hat{\mathbb{G}}$ and a hash function $H : \{0,1\}^* \to \mathbb{G}^2$ that ranges over $\mathbb{G} \times \mathbb{G}$. This hash function is modeled as a random oracle in the security analysis. While no party should know $\log_{\hat{g}_z}(\hat{g}_r)$, we do not need an extra round to generate $\hat{g}_r$ in a distributed manner as it can simply be derived from a random oracle.

$\mathsf{Dist\text{-}Keygen}(\mathsf{params}, \lambda, t, n)$: Given common public parameters $\mathsf{params} = \{(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), \hat{g}_z, \hat{g}_r, H\}$, a security parameter λ and integers $t, n \in \mathbb{N}$ such that $n \geq 2t + 1$, each player P_i conducts the following steps.

1. Each player P_i shares two pairs $\{(a_{ik0}, b_{ik0})\}_{k=1}^2$. To this end, he does the following:
   (a) For each $k \in \{1, 2\}$, choose random polynomials $A_{ik}[X] = a_{ik0} + a_{ik1}X + \cdots + a_{ikt}X^t$, $B_{ik}[X] = b_{ik0} + b_{ik1}X + \cdots + b_{ikt}X^t \in \mathbb{Z}_p[X]$ of degree t and broadcast
   $$\hat{W}_{ik\ell} = \hat{g}_z^{\,a_{ik\ell}} \hat{g}_r^{\,b_{ik\ell}} \qquad \forall \ell \in \{0, \ldots, t\} \ .$$
   (b) For $j = 1$ to n, send $\{(A_{ik}(j), B_{ik}(j))\}_{k=1}^2$ to P_j.
2. For each set of shares $\{(A_{jk}(i), B_{jk}(i))\}_{k=1}^2$ received from another player P_j, P_i verifies that
   $$\hat{g}_z^{A_{jk}(i)} \hat{g}_r^{B_{jk}(i)} = \prod_{\ell=0}^{t} \hat{W}_{jk\ell}^{\,i^\ell} \qquad \text{for } k = 1, 2 \ . \tag{1}$$
   If these equalities do not both hold, P_i broadcasts a complaint against the faulty sender P_j.
3. Any player who received strictly more than t complaints is immediately disqualified. Each player P_i who received a complaint from another player P_j responds by returning the correct shares $\{(A_{ik}(j), B_{ik}(j))\}_{k=1}^2$. If any of these new shares does not satisfy (1), P_i is disqualified. Let $Q \subset \{1, \ldots, n\}$ be the set of non-disqualified players at the end of Step 3.
4. The public key is obtained as $PK = \{\hat{g}_k\}_{k=1}^2$, where $\hat{g}_k = \prod_{i \in Q} \hat{W}_{ik0} = \hat{g}_z^{\sum_{i \in Q} a_{ik0}} \hat{g}_r^{\sum_{i \in Q} b_{ik0}}$. Each P_i locally defines his private key share
   $$SK_i = \{(A_k(i), B_k(i))\}_{k=1}^2 = \{(\textstyle\sum_{j \in Q} A_{jk}(i), \sum_{j \in Q} B_{jk}(i))\}_{k=1}^2$$
   and anyone can publicly compute his verification key $VK_i = (\hat{V}_{1,i}, \hat{V}_{2,i})$ as
   $$VK_i = (\hat{g}_z^{\,A_1(i)} \hat{g}_r^{\,B_1(i)}, \hat{g}_z^{\,A_2(i)} \hat{g}_r^{\,B_2(i)}) = \left(\prod_{j \in Q} \prod_{\ell=0}^{t} \hat{W}_{j1\ell}^{\,i^\ell}, \ \prod_{j \in Q} \prod_{\ell=0}^{t} \hat{W}_{j2\ell}^{\,i^\ell} \right) \ .$$
   For any disqualified player $i \in \{1, \ldots, n\} \setminus Q$, the i-th private key share is implicitly set as $SK_i = \{(0,0)\}_{k=1}^2$ and the corresponding verification key is $VK_i = (1_{\hat{\mathbb{G}}}, 1_{\hat{\mathbb{G}}})$.

[1] Namely, the private key space forms an additive group such that, for any message M, given any two signatures $\sigma_1 \leftarrow \mathsf{Sign}(sk_1, M)$ and $\sigma_2 \leftarrow \mathsf{Sign}(sk_2, M)$, anyone can compute a valid signature on M for the private key $sk_1 + sk_2$.

This completes the generation of the private key shares $\mathbf{SK} = (SK_1, \ldots, SK_n)$, the vector of verification keys $\mathbf{VK} = (VK_1, \ldots, VK_n)$ and the public key, which consists of

$$PK = (\mathsf{params},\ (\hat{g}_1, \hat{g}_2)) \ .$$

When the protocol ends, the obtained private key shares $\{A_k(i)\}_{k=1}^2$ and $\{B_k(i)\}_{k=1}^2$ lie on t-degree polynomials $A_k[X] = \sum_{j \in Q} A_{jk}[X]$ and $B_k[X] = \sum_{j \in Q} B_{jk}[X]$. Each player also holds an additive share $\{(a_{ik0}, b_{ik0})\}_{k=1}^2$ of the secret key $\{(A_k(0), B_k(0)) = (\sum_{i \in Q} a_{ik0}, \sum_{i \in Q} b_{ik0})\}_{k=1}^2$ but these shares will not be used in the scheme.

Share-Sign(i, SK_i, M): To generate a partial signature on a message $M \in \{0,1\}^*$ using his private key share $SK_i = \{(A_k(i), B_k(i))\}_{k=1}^2$, P_i first computes the hash value $(H_1, H_2) = H(M) \in \mathbb{G} \times \mathbb{G}$ and generates the partial signature $\sigma_i = (z_i, r_i) \in \mathbb{G}^2$ as

$$z_i = \prod_{k=1}^{2} H_k^{-A_k(i)}, \qquad r_i = \prod_{k=1}^{2} H_k^{-B_k(i)} \ .$$

Share-Verify$(PK, \mathbf{VK}, M, (i, \sigma_i))$: Given a candidate partial signature $\sigma_i = (z_i, r_i) \in \mathbb{G}^2$ and the verification key $VK_i = (\hat{V}_{1,i}, \hat{V}_{2,i})$, the partial verification algorithm first computes $(H_1, H_2) = H(M) \in \mathbb{G}^2$. It returns 1 if the equality $e(z_i, \hat{g}_z) \cdot e(r_i, \hat{g}_r) \cdot \prod_{k=1}^2 e(H_k, \hat{V}_{k,i}) = 1_{\mathbb{G}_T}$ holds and 0 otherwise.

Combine$(PK, \mathbf{VK}, M, \{(i, \sigma_i)\}_{i \in S})$: Given a $(t+1)$-set with valid shares $\{(i, \sigma_i)\}_{i \in S}$, parse the signature share σ_i as $(z_i, r_i) \in \mathbb{G}^2$ for each $i \in S$. Then, compute $(z, r) = (\prod_{i \in S} z_i^{\Delta_{i,S}(0)},\ \prod_{i \in S} r_i^{\Delta_{i,S}(0)})$ by Lagrange interpolation in the exponent. Return the pair $(z, r) \in \mathbb{G}^2$.

Verify(PK, M, σ): Given a purported signature $\sigma = (z, r) \in \mathbb{G}^2$, compute $(H_1, H_2) = H(M) \in \mathbb{G} \times \mathbb{G}$ and return 1 if and only if the following equality holds:

$$e(z, \hat{g}_z) \cdot e(r, \hat{g}_r) \cdot e(H_1, \hat{g}_1) \cdot e(H_2, \hat{g}_2) = 1_{\mathbb{G}_T} \ .$$

If the scheme is instantiated using Barreto-Naehrig curves [5] at the 128-bit security level, each signature consists of 512 bits. For the same security level, RSA-based threshold signatures like [67, 4] require 3076 bits. The scheme is also very efficient from a computational standpoint. Each server only has to compute two multi-exponentiations with two base elements and two "hash-on-curve" operations. The verifier has to compute a product of four pairings.

At the end of the key generation phase, each player only needs to store a private key share $SK_i = \{(A_k(i), B_k(i))\}_{k=1}^2$ of constant-size —whereas solutions like [4] incur the storage of $O(n)$ elements at each player— and can erase all intermediate values, including the polynomials $A_{ik}[X]$ and $B_{ik}[X]$. However, we insist that the security analysis does not require reliable erasures. When a player is corrupted, we assume that the adversary learns the entire history of this player.

3.2. Security

Although the public key is not guaranteed to be uniform due to the use of Pedersen's DKG protocol, the key homomorphic property allows the reduction to turn the adversary's forgery into a valid signature with respect to some uniformly random public key obtained by multiplying honest users' contributions to the public key. This is sufficient for solving a given Double Pairing instance.

The security proof proceeds with a sequence of games which can be outlined as follows. The first game is the real game where the challenger assumes the role of all honest players in the distributed key generation phase. Since it controls a majority of players, the challenger knows the polynomials $\{A_{jk}[X], B_{jk}[X]\}_{j \in Q, k \in \{1,2\}}$ and the private key shares $\{SK_j\}_{j \in Q}$ of all non-disqualified players —either because it obtained at least $t+1$ polynomial shares $\{(A_{jk}(i), B_{jk}(i))\}_{i \in \mathcal{G}, k \in \{1,2\}}$ for each $j \in Q$ or because it chose the polynomials itself— at the end of the Dist-Keygen protocol.

In subsequent games, the challenger applies Coron's proof technique for Full Domain Hash signatures [20]. At each random oracle query $H(M)$, it flips a coin $\vartheta_M \in \{0,1\}$ that takes the value 0 with probability $q_s/(q_s+1)$ and the value 1 with probability $1/(q_s+1)$, where q_s is the number of signing queries. If $\vartheta_M = 1$, the challenger defines

$H(M)$ to be a random vector of $\mathbb{G}^2$. If $\vartheta_M = 0$, the message M is hashed to a subspace of dimension 1. We prove that, although H does no longer behave as an actual random oracle, this change should not affect the adversary's view if the DDH assumption holds in $\mathbb{G}$. Coron's analysis [20] shows that, with probability $\Omega(1/q_s)$, the following conditions are fulfilled:

(i) The adversary only obtains partial signatures on messages $M_1, \ldots, M_{q_s}$ that are hashed in a one-dimensional subspace;

(ii) The adversary's forgery involves a message $M^\star$ such that $(H_1^\star, H_2^\star) = H(M^\star)$ is linearly independent of the vectors $\{(H_{1,i}, H_{2,i}) = H(M_i)\}_{i=1}^{q_s}$.

Condition (i) ensures that the adversary obtains little information about the private key shares $\{SK_i\}_{i \in \mathcal{G}}$ and the additive shares $\{a_{ik0}, b_{ik0}\}_{i \in \mathcal{G}, k \in \{1,2\}}$ of honest players. Hence, if the challenger computes the additive contribution of honest players to a signature on the vector $(H_1^\star, H_2^\star) = H(M^\star)$, this contribution is completely unpredictable by the adversary due to Condition (ii). With overwhelming probability, this contribution does *not* coincide with the one that can be extracted (using the additive shares $\{a_{jk0}, b_{jk0}\}_{j \in Q \backslash \mathcal{G},\, k \in \{1,2\}}$ that the reduction knows from the key generation phase) from the adversary's forgery $(z^\star, r^\star)$ using the key homomorphic property of the scheme. The challenger thus obtains two distinct linearly homomorphic signatures on the vector $(H_1^\star, H_2^\star)$, which allows solving an instance of the Double Pairing problem.

Theorem 1. *The scheme provides adaptive security under the SXDH assumption in the random oracle model. Namely, for any PPT adversary $\mathcal{A}$, there exist DDH distinguishers $\mathcal{B}_1$ and $\mathcal{B}_2$ with comparable running time in the groups $\mathbb{G}$ and $\hat{\mathbb{G}}$, respectively.*

Proof. The proof proceeds with a sequence of three games. The latter begins with Game 0, which is the real game, and ends with Game 2, where any PPT adversary is shown to contradict the Double Pairing assumption. For each $j \in \{0, 1, 2\}$, S_j denotes the event that the adversary wins in Game j.

We assume w.l.o.g. that the adversary $\mathcal{A}$ always queries the random oracle H before any signing query for the same message M. The challenger can always enforce this by making random oracle queries for itself. We also assume that random oracle queries are distinct.

Game 0: This is the real game. Namely, the challenger runs the Dist-Keygen protocol on behalf of all uncorrupted players. Whenever the adversary $\mathcal{A}$ decides to corrupt a player P_i, the challenger sets $C = C \cup \{i\}$, $\mathcal{G} = \mathcal{G} \backslash \{i\}$ and faithfully reveals the internal state of P_i, which includes P_i's private key share $SK_i = \{(A_k(i), B_k(i))\}_{k=1}^2$ and his polynomials $\{A_{ik}[X], B_{ik}[X]\}_{k=1}^2$ if the corruption query occurs after Step 1.a of Dist-Keygen. Whenever a player P_i is corrupted, $\mathcal{A}$ receives full control over P_i and may cause him to arbitrarily deviate from the protocol. Queries to the random oracle H are answered by returning uniformly random group elements in $\mathbb{G}^2$. Partial signature queries (i, M) are answered by returning the values $(z_i, r_i) = \left(\prod_{k=1}^2 H_k^{-A_k(i)}, \prod_{k=1}^2 H_k^{-B_k(i)}\right)$. At the end of the game, $\mathcal{A}$ outputs a message-signature pair $(\sigma^\star = (z^\star, r^\star), M^\star)$. We assume that the adversary queries $H(M^\star)$ before producing its forgery. We denote by S_0 the event that $\sigma^\star = (z^\star, r^\star)$ is a valid signature.

In the following, we define $A_k[X] = \sum_{i \in Q} A_{ik}[X]$ and $B_k[X] = \sum_{i \in Q} B_{ik}[X]$ as well as $(a_{k0}, b_{k0}) = (\sum_{i \in Q} a_{ik0}, \sum_{i \in Q} b_{ik0})$ for each $k \in \{1, 2\}$. We remark that, at the end of the Dist-Keygen protocol, the challenger knows the polynomials $\{A_{jk}[X], B_{jk}[X]\}_{k=1}^2$ and the additive shares $\{(a_{jk0}, b_{jk0})\}_{k=1}^2$ of all non-disqualified players $j \in Q$. Indeed, for each $j \in Q \cap C$ such that P_j was corrupted before Step 1.a of the distributed key generation phase, it obtained at least $t + 1$ shares $\{A_{jk}(i), B_{jk}(i)\}_{k=1}^2$, which is sufficient for reconstructing $\{A_{jk}[X], B_{jk}[X]\}_{k=1}^2$. As for other players P_j such that $j \in Q$, the challenger honestly chose their sharing polynomials at Step 1.a of Dist-Keygen.

Game 1: This game is identical to Game 0 but with the following difference. For each random oracle query $H(M)$, the challenger $\mathcal{B}$ flips a biased coin $\vartheta_M \in \{0, 1\}$ that takes the value 1 with probability $1/(q_s + 1)$ and the value 0 with probability $q_s/(q_s + 1)$. When the game ends, $\mathcal{B}$ considers the event E that either of the following conditions holds:

- For the message $M^\star$, the coin $\vartheta_{M^\star} \in \{0, 1\}$ flipped for the hash query $H(M^\star)$ was $\vartheta_{M^\star} = 0$.
- There exists signing query (i, M) with $M \neq M^\star$ for which $\vartheta_M = 1$.

If event E occurs (which $\mathcal{B}$ can detect at the end of the game), $\mathcal{B}$ halts and declares failure. An analysis similar to that of Coron [20] shows that

$$\Pr[\neg E] = \frac{1}{q_s+1} \cdot \left(\frac{q_s}{q_s+1}\right)^{q_s} = \frac{1}{q_s+1}\left(1 - \frac{1}{q_s+1}\right)^{q_s} \geq \frac{1}{e\,(q_s+1)}$$

where e is the base for the natural logarithm. The transition from Game 0 to Game 1 is thus a transition based on a failure event of large probability [27] and we thus have $\Pr[S_1] = \Pr[S_0] \cdot \Pr[\neg E] \geq \Pr[S_0]/e(q_s+1)$.

Game 2: We modify the distribution of random oracle outputs. Specifically, the challenger $\mathcal{B}$ chooses generators $g, h \stackrel{R}{\leftarrow} \mathbb{G}$ at the beginning of the game and uses them to answer random oracle queries. The treatment of each hash query $H(M)$ depends on the random coin $\vartheta_M \in \{0,1\}$.

- If $\vartheta_M = 0$, the challenger $\mathcal{B}$ chooses a random $\alpha_M \stackrel{R}{\leftarrow} \mathbb{Z}_p$, and programs the random oracle so as to have $H(M) = (g^{\alpha_M}, h^{\alpha_M})$. Note that the resulting hash value $H(M) \in \mathbb{G}^2$ is no longer uniform in $\mathbb{G}^2$ as it now lives in the one-dimensional space spanned by the vector $(g,h) \in \mathbb{G}^2$.
- If $\vartheta_M = 1$, $\mathcal{B}$ chooses a uniformly random pair $(g_M, h_M) \in \mathbb{G}^2$ and programs $H(M)$ so as to have $H(M) = (g_M, h_M)$.

Lemma 1 below shows that Game 2 and Game 1 are computationally indistinguishable if the DDH assumption holds in the group $\mathbb{G}$. It follows that $|\Pr[S_2] - \Pr[S_1]| \leq \mathbf{Adv}^{\mathrm{DDH}_1}(\mathcal{B})$.

In Game 2, we claim that $\Pr[S_2] \leq \mathbf{Adv}^{\mathrm{DP}}(\mathcal{B}) + 1/p$ as $\mathcal{B}$ can be turned into an algorithm solving the DP problem.

Indeed, with probability $1/e(q_s+1)$, the hash value $H(M^\star) = (H_1^\star, H_2^\star) \in \mathbb{G}^2$ is uniformly random for the message $M^\star$ involved in the forgery $(z^\star, r^\star)$ whereas, for each signed message M such that $M \neq M^\star$, $H(M) = (H_1, H_2)$ lives in the one-dimensional subspace spanned by (g,h). We also note that, while the adversary is allowed to submit queries of the form $(i, M^\star)$ to the partial signing oracle, these queries do not reveal any more information than if the challenger were simply handing over the corresponding private share SK_i. We thus treat these partial signing queries for $M^\star$ as corruption queries. When $\mathcal{A}$ halts, the challenger determines which players have generated a partial signature on $M^\star$ and moves them from $\mathcal{G}$ to C. Note that, for these updated sets $\mathcal{G}$ and C, it still knows the polynomials $\{(A_{jk}[X], B_{jk}[X])\}_{k=1}^2$ for all $j \in C$. Let us define the aggregated additive shares

$$\begin{aligned} a_{k,\mathcal{G}} &= \sum_{j\in\mathcal{G}} a_{jk0}\,, & b_{k,\mathcal{G}} &= \sum_{j\in\mathcal{G}} b_{jk0}\,, & & \\ a_{k,Q\cap C} &= \sum_{j\in Q\cap C} a_{jk0}\,, & b_{k,Q\cap C} &= \sum_{j\in Q\cap C} b_{jk0}\,, & k &\in \{1,2\}\ . \end{aligned}$$

We remark that all pairs $\{(a_{k,\mathcal{G}}, b_{k,\mathcal{G}})\}_{k=1}^2$ are uniformly distributed in $\mathbb{Z}_p^2$ since they are obtained by summing additive shares that were honestly chosen by the challenger.

We also argue that $a_{2,\mathcal{G}}$ is independent of $\mathcal{A}$'s view. To see this, let us consider what an unbounded $\mathcal{A}$ can learn during the game. Corruption queries reveal $\{A_{j2}(i)\}_{j\in\mathcal{G}, i\in C}$, which is insufficient to infer anything about $a_{2,\mathcal{G}} = \sum_{j\in\mathcal{G}} A_{j2}(0)$ since $|C| \leq t$. For each $M \neq M^\star$, signing queries are answered by returning

$$(z_i, r_i) = (H_1^{-A_1(i)} H_2^{-A_2(i)},\ H_1^{-B_1(i)} H_2^{-B_2(i)}) = \Big((g^{-A_1(i)} \cdot h^{-A_2(i)})^{\alpha_M},\ (g^{-B_1(i)} \cdot h^{-B_2(i)})^{\alpha_M}\Big)\ .$$

Note that the information supplied by r_i is redundant since, for a given pair (H_1, H_2) and a given $z_i \in \mathbb{G}$, there is only one $r_i \in \mathbb{G}$ satisfying $e(z_i, \hat{g}_z) \cdot e(r_i, \hat{g}_r) \cdot \prod_{k=1}^2 e(H_k, \hat{V}_{k,i}) = 1_{\mathbb{G}_T}$. Since $\mathcal{A}$ knows $\{(A_{jk}[X], B_{jk}[X])\}_{k=1}^2$ for each $j \in Q \cap C$, it can obtain

$$z_{i,\mathcal{G}} = (g^{-\sum_{j\in\mathcal{G}} A_{j1}(i)} \cdot h^{-\sum_{j\in\mathcal{G}} A_{j2}(i)})^{\alpha_M}\ . \qquad (2)$$

However, these partial signatures (z_i, r_i) on $M \neq M^\star$ only provide $\mathcal{A}$ with redundant information about $(\sum_{j\in\mathcal{G}} A_{j1}(i)$, $\sum_{j\in\mathcal{G}} A_{j2}(i), \sum_{j\in\mathcal{G}} B_{j1}(i), \sum_{j\in\mathcal{G}} B_{j2}(i))$. The only thing that $\mathcal{A}$ really learns from (2) is the value $\sum_{j\in\mathcal{G}}(A_{j1}(i) + \omega \cdot A_{j2}(i))$, where $\omega = \log_g(h)$. In addition, during Step 2 of the Dist-Keygen protocol, relation (1) also provides the adversary with $\hat{g}_z{}^{A_{jk}(i)}\hat{g}_r{}^{B_{jk}(i)}$ for each $i \in \{1,\ldots,n\}$, $j \in \mathcal{G}$ and $k \in \{1,2\}$. Still, the only way to leverage these pieces of information is to interpolate them and get $a_{1,\mathcal{G}} + \omega \cdot a_{2,\mathcal{G}}$ as well as $\{a_{k,\mathcal{G}} + \rho \cdot b_{k,\mathcal{G}}\}_{k=1}^{2}$, where $\rho = \log_{\hat{g}_z}(\hat{g}_r)$, which leaves $\mathcal{A}$ with a system of 3 equations in 4 unknowns $\{(a_{k,\mathcal{G}}, b_{k,\mathcal{G}})\}_{k=1}^{2}$. As a consequence, $a_{2,\mathcal{G}}$ remains completely undetermined in $\mathcal{A}$'s view as long as $|\mathcal{C}| \leq t$.

The lack of adversarial information about $a_{2,\mathcal{G}}$ allows solving the DP problem as follows. For the target message $M^\star$, we can write $(H_1^\star, H_2^\star) = (g^{\alpha_{M^\star}}, h^{\alpha_{M^\star}+\gamma})$, for some random $\alpha_{M^\star}, \gamma \in_R \mathbb{Z}_p$. This implies that, if the challenger computes a product $(z^\dagger, r^\dagger)$ of its own partial signatures on the message $M^\star$ using the sum $(a_{1,\mathcal{G}}, a_{2,\mathcal{G}}, b_{1,\mathcal{G}}, b_{2,\mathcal{G}})$ of its additive shares, this product can be written as

$$\begin{aligned}(z^\dagger, r^\dagger) &= (H_1^{\star\,-a_{1,\mathcal{G}}} \cdot H_2^{\star\,-a_{2,\mathcal{G}}},\ H_1^{\star\,-b_{1,\mathcal{G}}} \cdot H_2^{\star\,-b_{2,\mathcal{G}}}) \\ &= \big((g^{a_{1,\mathcal{G}}} \cdot h^{a_{2,\mathcal{G}}})^{-\alpha_{M^\star}} \cdot h^{-\gamma \cdot a_{2,\mathcal{G}}},\ (g^{b_{1,\mathcal{G}}} \cdot h^{b_{2,\mathcal{G}}})^{-\alpha_{M^\star}} \cdot h^{-\gamma \cdot b_{2,\mathcal{G}}}\big), \end{aligned} \tag{3}$$

where $z^\dagger$ is completely unpredictable by $\mathcal{A}$. Indeed, in the right-hand-side member of (3), $\mathcal{A}$ can information-theoretically determine the term $(g^{a_{1,\mathcal{G}}} \cdot h^{a_{1,\mathcal{G}}})^{\alpha_{M^\star}}$ by interpolating the discrete logarithms $\sum_{j\in\mathcal{G}}(A_{j,1}(i)+\omega A_{j,2}(i))$ obtained from (2) (note that, although (g, h) are not explicitly given to $\mathcal{A}$, they can be inferred, in the same way as exponents α_M and $\alpha_{M^\star}$, by observing hash values). However, the uniformly random term $h^{-\gamma \cdot a_{2,\mathcal{G}}}$ remains completely independent of $\mathcal{A}$'s view.

Now, the challenger can use the adversary's forgery $(z^\star, r^\star)$ to compute

$$(z^\diamond, r^\diamond) = \Big(z^\star \cdot H_1^{\star\ a_{1,Q\cap C}} \cdot H_2^{\star\ a_{2,Q\cap C}},\ r^\star \cdot H_1^{\star\ b_{1,Q\cap C}} \cdot H_2^{\star\ b_{2,Q\cap C}}\Big),$$

which, if we define $\hat{g}_{1,\mathcal{G}} = \hat{g}_z{}^{a_{1,\mathcal{G}}} \cdot \hat{g}_r{}^{b_{1,\mathcal{G}}}$ and $\hat{g}_{2,\mathcal{G}} = \hat{g}_z{}^{a_{2,\mathcal{G}}} \cdot \hat{g}_r{}^{b_{2,\mathcal{G}}}$, is easily seen to satisfy

$$e(z^\diamond, \hat{g}_z) \cdot e(r^\diamond, \hat{g}_z) \cdot e(H_1^\star, \hat{g}_{1,\mathcal{G}}) \cdot e(H_2^\star, \hat{g}_{2,\mathcal{G}}) = 1_{\mathbb{G}_T}$$

since $\hat{g}_1 = \hat{g}_{1,\mathcal{G}} \cdot \hat{g}_z{}^{\ a_{1,Q\cap C}} \cdot \hat{g}_r{}^{\ b_{1,Q\cap C}}$ and $\hat{g}_2 = \hat{g}_{2,\mathcal{G}} \cdot \hat{g}_z{}^{\ a_{2,Q\cap C}} \cdot \hat{g}_r{}^{\ b_{2,Q\cap C}}$.

From (3), we see that $(z^\dagger, r^\dagger)$ also satisfies $e(z^\dagger, \hat{g}_z) \cdot e(r^\dagger, \hat{g}_z) \cdot e(H_1^\star, \hat{g}_{1,\mathcal{G}}) \cdot e(H_2^\star, \hat{g}_{2,\mathcal{G}}) = 1_{\mathbb{G}_T}$ by construction. Given that $z^\dagger$ is independent of $\mathcal{A}$'s view, the quotient $(z^\dagger/z^\diamond, r^\dagger/r^\diamond)$ forms a non-trivial solution to the DP instance $(\hat{g}_z, \hat{g}_r)$ with probability $1 - 1/p$. Such a solution easily allows building a distinguisher for the DDH problem in $\hat{\mathbb{G}}$. We thus find the upper bound

$$\mathbf{Adv}(\mathcal{A}) \leq e\,(q_s + 1) \cdot \Big(\mathbf{Adv}^{\mathrm{DDH}_1}(\mathcal{B}) + \mathbf{Adv}^{\mathrm{DDH}_2}(\mathcal{B}) + \frac{1}{p}\Big), \tag{4}$$

where q_s is the number of signing queries and e is the base for the natural logarithm. □

We remark that the proof of Theorem 1 goes through if, during the key generation phase, each player P_i additionally publicizes $(Z_{i0}, R_{i0}) = (g^{-a_{i10}}h^{-a_{i20}}, g^{-b_{i10}}h^{-b_{i20}})$, for public $g, h \in \mathbb{G}$, which satisfies $e(Z_{i0}, \hat{g}_z) \cdot e(R_{i0}, \hat{g}_r) \cdot e(h, \hat{W}_{i10}) \cdot e(g, \hat{W}_{i20}) = 1_{\mathbb{G}_T}$ and thus forms a LHSPS on (g, h) for the public key $\{\hat{W}_{ik0}\}_{k=1}^{2}$. Indeed, if we consider the information that each player initially reveals about its local additive shares $(a_{i10}, a_{i20}, b_{i10}, b_{i20}) \in \mathbb{Z}_p^4$, it amounts to the discrete logarithms of $(\hat{W}_{i10}, \hat{W}_{i20}, Z_{i0})$. The only extra information revealed by Z_{i0} is thus $a_{i10} + \omega \cdot a_{i20}$, where $\omega = \log_g(h)$, which leaves a_{i20} undetermined. While an unbounded adversary can compute the sum $a_{1,\mathcal{G}} + \omega \cdot a_{2,\mathcal{G}}$ in Game 2, it still has no information about $a_{2,\mathcal{G}} = \sum_{j\in\mathcal{G}} a_{j20}$. In Appendix Appendix F, we use this observation to show a simple modification of the scheme that supports signature aggregation.

3.3. Adding proactive security

The scheme readily extends to provide proactive security [60, 47, 37] against mobile adversaries that can potentially corrupt all the players at some point as long as it never controls more than t players at any time. By having the players refreshing all shares (without changing the secret) at discrete time intervals, the scheme remains secure against an adversary corrupting up to t players during the same period. This is achieved by having all players run a new instance of Pedersen's DKG protocol where the shared secret is $\{(0,0)\}_{k=1}^{2}$ and locally add the resulting shares to their local shares before updating $\{VK_i\}_{i=1}^{n}$ accordingly.

The techniques of [46, Section 4] can also be applied to detect parties holding a corrupted share (due to a crash during an update phase or an adversarial behavior) and restore the correct share.

4. A construction in the standard model

This section gives a round-optimal construction in the standard model. We remark that, under the Decision Linear assumption [11], any one-time LHSPS in symmetric bilinear groups can be turned into a full-fledged digital signature, as shown in the full version of the paper. In the threshold setting, we need to rely on specific properties of the underlying LHSPS in order to achieve adaptive security without relying on a trusted dealer.

The scheme relies on the Groth-Sahai non-interactive witness indistinguishable (NIWI) proof systems [45], which are recalled in Appendix Appendix A. In its centralized version, a signature consists of a NIWI proof of knowledge —somewhat in the spirit of Okamoto's signature scheme [59]— of a one-time linearly homomorphic signature on a fixed vector $g \in \mathbb{G}$ of dimension $n = 1$. To generate this proof, the signer forms a Groth-Sahai [45] common reference string (CRS) $(\vec{f}, \vec{f_M})$ using the bits of the message M, according to a technique suggested by Malkin *et al.* [57]. Due to the witness indistinguishability property of Groth-Sahai proofs, no information leaks about the private key of the underlying one-time homomorphic signature. For this reason, when the adversary creates a fake signature, the reduction is able to extract a different homomorphic signature than the one it can compute. Hence, it obtains two distinct signatures on the same vector, which allows solving an instance of the DP problem.

The scheme can also be seen as a threshold version of (a variant of) the signature presented in [57]. In order to distribute the signing process, we take advantage of the homomorphic properties of Groth-Sahai proofs. More precisely, we use the fact that linear pairing product equations and their proofs can be linearly combined in order to obtain a valid proof for the desired statement when performing a Lagrange interpolation in the exponent. In order to avoid interaction during the signing process, we leverage the property that the centralized signature scheme is key homomorphic.

However, we have to prove that the scheme remains adaptively secure when the DKG phase uses Pedersen's protocol [61]. To this end, we take further advantage of the key homomorphic property. In the security proof, we show that, if the adversary can forge a signature for a non-uniform public key PK, we can turn this forgery into one for another public key PK', which is uniformly distributed.

In the following notations, for each $h \in \mathbb{G}$ and any vector $\vec{g} = (g_1, g_2) \in \mathbb{G}^2$, we denote by $E(\vec{g}, \hat{h})$ the vector $(e(g_1, \hat{h}), e(g_2, \hat{g})) \in \mathbb{G}_T^2$.

Here, we assume public parameters params made of asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with generators $g \in_R \mathbb{G}$, $\hat{g}_z, \hat{g}_r \in_R \hat{\mathbb{G}}$ and vectors $\vec{f} = (f, h) \in \mathbb{G}^2$ and $\vec{f_i} = (f_i, h_i) \overset{R}{\leftarrow} \mathbb{G}^2$ for $i = 0$ to L, where $L \in \mathsf{poly}(\lambda)$.

$\mathsf{Dist\text{-}Keygen}(\mathsf{params}, \lambda, t, n)$: This protocol proceeds as in the scheme of Section 3. Namely, given common parameters $\mathsf{params} = \{(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, \hat{g}_z, \hat{g}_r, \vec{f}, \{\vec{f_i}\}_{i=0}^{L}\}$, a security parameter λ and integers $t, n \in \mathbb{N}$ such that $n \geq 2t + 1$, each player P_i conducts the following steps.

1. Each player P_i shares a random pair (a_{i0}, b_{i0}) according to the following step:
   (a) Pick random polynomials $A_i[X] = a_{i0} + a_{i1}X + \cdots + a_{it}X^t$, $B_i[X] = b_{i0} + b_{i1}X + \cdots + b_{it}X^t$ of degree t and broadcast $\hat{W}_{i\ell} = \hat{g}_z{}^{a_{i\ell}} \hat{g}_r{}^{b_{i\ell}}$ for all $\ell \in \{0, \ldots, t\}$.
   (b) For $j = 1$ to n, send $(A_i(j), B_i(j))$ to P_j.
2. For each received shares $(A_j(i), B_j(i))$, player P_i verifies that
$$\hat{g}_z^{A_j(i)} \hat{g}_r^{B_j(i)} = \prod_{\ell=0}^{t} \hat{W}_{j\ell}^{\,i^\ell} \quad . \tag{5}$$
   If the latter equality does not hold, P_i broadcasts a complaint against P_j.
3. Any player receiving more than t complaints is disqualified. Each player P_i who received a complaint from another player P_j responds by returning the correct shares $(A_i(j), B_i(j))$. If any of these new shares fails to satisfy (5), the faulty P_i is expelled. Let $Q \subset \{1, \ldots, n\}$ be the set of non-disqualified players at the end of Step 3.
4. The public key PK is obtained as $PK = \hat{g}_1$, where $\hat{g}_1 = \prod_{i \in Q} \hat{W}_{i0} = \hat{g}_z^{\sum_{i \in Q} a_{i0}} \hat{g}_r^{\sum_{i \in Q} b_{i0}}$. Each P_i locally defines his private key share as $SK_i = (A(i), B(i)) = \left(\sum_{j \in Q} A_j(i), \sum_{j \in Q} B_j(i)\right)$ and anyone can publicly compute his verification key as $VK_i = \hat{V}_i = \hat{g}_z{}^{A(i)} \hat{g}_r{}^{B(i)} = \prod_{j \in Q} \prod_{\ell=0}^{t} \hat{W}_{j\ell}^{\,i^\ell}$. Any disqualified player $i \in \{1, \ldots, n\} \setminus Q$ is implicitly assigned the share $SK_i = (0, 0)$ and the matching verification key $VK_i = 1_{\hat{\mathbb{G}}}$.

The vector of private key shares is $\mathbf{SK} = (SK_1, \ldots, SK_n)$ and the corresponding vector of verification keys $\mathbf{VK} = (VK_1, \ldots, VK_n)$. The public key consists of $PK = (\mathsf{params},\ \hat{g}_1)$.

Share-Sign(SK_i, M): To generate a partial signature on a L-bit message $M = M[1] \ldots M[L] \in \{0,1\}^L$ using $SK_i = (A(i), B(i))$, define $(z_i, r_i) = (g^{-A(i)}, g^{-B(i)})$.

1. Using the bits $M[1] \ldots M[L]$ of $M \in \{0,1\}^L$, define the vector $\vec{f}_M = \vec{f}_0 \cdot \prod_{i=1}^{L} \vec{f}_i^{M[i]}$ so as to assemble a Groth-Sahai CRS $\mathbf{f}_M = (\vec{f}, \vec{f}_M)$.
2. Using the CRS $\mathbf{f}_M = (\vec{f}, \vec{f}_M)$, compute Groth-Sahai commitments $\vec{C}_{z,i} = (1_{\mathbb{G}}, z_i) \cdot \vec{f}_1^{\nu_{z,1}} \cdot \vec{f}_M^{\nu_{z,2}}$ and $\vec{C}_{r,i} = (1_{\mathbb{G}}, r_i) \cdot \vec{f}^{\nu_{r,1,i}} \cdot \vec{f}_M^{\nu_{r,2,i}}$ to the group elements z_i and r_i, respectively. Then, generate a NIWI proof $\vec{\hat{\pi}}_i = (\hat{\pi}_{1,i}, \hat{\pi}_{2,i}) \in \hat{\mathbb{G}}^2$ that committed elements $(z_i, r_i) \in \mathbb{G}^2$ satisfy the verification equation $1_{\mathbb{G}_T} = e(z_i, \hat{g}_z) \cdot e(r_i, \hat{g}_r) \cdot e(g, \hat{V}_i)$. This proof is obtained as

$$\vec{\hat{\pi}}_i = (\hat{\pi}_{1,i}, \hat{\pi}_{2,i}) = \big(\hat{g}_z^{\,-\nu_{z,1,i}} \cdot \hat{g}_r^{\,-\nu_{r,1,i}},\ \hat{g}_z^{\,-\nu_{z,2,i}} \cdot \hat{g}_r^{\,-\nu_{r,2,i}}\big)$$

Return $\sigma_i = (\vec{C}_{z,i}, \vec{C}_{r,i}, \vec{\hat{\pi}}_i) \in \mathbb{G}^4 \times \hat{\mathbb{G}}^2$.

Share-Verify($PK, \mathbf{VK}, M, (i, \sigma_i)$): Given $M \in \{0,1\}^L$ and a candidate σ_i, parse σ_i as $\sigma_i = (\vec{C}_{z,i}, \vec{C}_{r,i}, \vec{\hat{\pi}}_i)$. Define $\vec{f}_M = \vec{f}_0 \cdot \prod_{i=1}^{L} \vec{f}_i^{M[i]}$ and return 1 if $\vec{\hat{\pi}}_i = (\hat{\pi}_{1,i}, \hat{\pi}_{2,i})$ satisfies

$$E((1_{\mathbb{G}}, g), \hat{V}_i)^{-1} = E(\vec{C}_{z,i}, \hat{g}_z) \cdot E(\vec{C}_{r,i}, \hat{g}_r) \cdot E(\vec{f}, \hat{\pi}_{1,i}) \cdot E(\vec{f}_M, \hat{\pi}_{2,i})$$

and 0 otherwise.

Combine($PK, \mathbf{VK}, M, \{(i, \sigma_i)\}_{i \in S}$): Given a $(t+1)$-set with valid shares $\{(i, \sigma_i)\}_{i \in S}$, parse each signature share σ_i as $(\vec{C}_{z,i}, \vec{C}_{r,i}, \vec{\hat{\pi}}_i) \in \mathbb{G}^4 \times \hat{\mathbb{G}}^2$, where $\vec{\hat{\pi}}_i = (\hat{\pi}_{1,i}, \hat{\pi}_{2,i})$, for all $i \in S$. Then, compute $(\vec{C}'_z, \vec{C}'_r, \hat{\pi}'_1, \hat{\pi}'_2)$ as

$$\Big(\prod_{i \in S} \vec{C}_{z,i}^{\Delta_{i,S}(0)},\ \prod_{i \in S} \vec{C}_{r,i}^{\Delta_{i,S}(0)},\ \prod_{i \in S} \hat{\pi}_{1,i}^{\Delta_{i,S}(0)},\ \prod_{i \in S} \hat{\pi}_{2,i}^{\Delta_{i,S}(0)}\Big)$$

by Lagrange interpolation in the exponent. Finally, re-randomize $(\vec{C}'_z, \vec{C}'_r, \hat{\pi}'_1, \hat{\pi}'_2)$ and output the resulting re-randomized full signature $\sigma = (\vec{C}_z, \vec{C}_r, \hat{\pi}_1, \hat{\pi}_2)$.

Verify(PK, M, σ): Given a message $M \in \{0,1\}^L$ and a purported signature σ, parse σ as $(\vec{C}_z, \vec{C}_r, \vec{\hat{\pi}}) \in \mathbb{G}^4 \times \hat{\mathbb{G}}^2$. Define $\vec{f}_M = \vec{f}_0 \cdot \prod_{i=1}^{L} \vec{f}_i^{M[i]}$ and return 1 if and only if $\vec{\hat{\pi}} = (\hat{\pi}_1, \hat{\pi}_2)$ satisfies

$$E((1_{\mathbb{G}}, g), \hat{g}_1)^{-1} = E(\vec{C}_z, \hat{g}_z) \cdot E(\vec{C}_r, \hat{g}_r) \cdot E(\vec{f}, \hat{\pi}_1) \cdot E(\vec{f}_M, \hat{\pi}_2) \ .$$

The scheme can be simplified by having each player set his private key share as $SK_i = (g^{-A(i)}, g^{-B(i)})$ so as to spare two exponentiations in the signing phase. In the description, we defined SK_i as $(A(i), B(i))$ to insist that no reliable erasures are needed. At each corruption query, the adversary obtains $(A(i), B(i))$ and, not only $(g^{-A(i)}, g^{-B(i)})$. In any case, each player only needs to store two elements of $\mathbb{Z}_p$.

At the 128-bit security level, if each element of $\mathbb{G}$ (resp. $\hat{\mathbb{G}}$) has a 256-bit (resp. 512 bit) representation on Barreto-Naehrig curves [5], we only need 2048 bits per signature.

Theorem 2. *The scheme provides adaptive security under the SXDH assumption in the standard model. Namely, for any PPT adversary $\mathcal{A}$, there exist DDH distinguishers $\mathcal{B}_1$ and $\mathcal{B}_2$ with comparable running time in the groups $\mathbb{G}$ and $\hat{\mathbb{G}}$, respectively.*

Proof. The proof uses a sequence of three games that begins with Game 0, which is the real game, and ends with Game 2, where even any PPT adversary $\mathcal{A}$ allows breaking the Double Pairing assumption. For each $j \in \{0,1,2\}$, we denote by S_j the event that the adversary wins in Game j.

Game 0: This is the actual game. Specifically, the challenger assumes the role of all uncorrupted players during the Dist-Keygen protocol. When $\mathcal{A}$ chooses to corrupt a player P_i, the challenger sets $C = C \cup \{i\}$, $\mathcal{G} = \mathcal{G}\backslash\{i\}$ and reveals the internal state of P_i, which includes P_i's private key share $SK_i = (A(i), B(i))$ and the polynomials $A_i[X], B_i[X]$ if the corruption occurs after Step 1.a of Dist-Keygen. When a player P_i is corrupted, it may arbitrarily deviate from the protocol. Partial signature queries (i, M) are answered by honestly running the partial signing algorithm. At the end of the game, $\mathcal{A}$ outputs a message-signature pair $(\sigma^\star = (\vec{C}_z^\star, \vec{C}_r^\star, \vec{\pi}^\star), M^\star)$. We denote by S_0 the event that $\sigma^\star = (\vec{C}_z^\star, \vec{C}_r^\star, \vec{\pi}^\star)$ is a valid forgery.

We define $A[X] = \sum_{i \in Q} A_i[X]$, $B[X] = \sum_{i \in Q} B_i[X]$ and $(a_0, b_0) = (\sum_{i \in Q} a_{i0}, \sum_{i \in Q} b_{i0})$. At the end of the Dist-Keygen protocol, the challenger knows the polynomials $A_j[X], B_j[X]$ and the additive shares (a_{j0}, b_{j0}) of all non-disqualified players $j \in Q$. Indeed, since there is always a majority of honest players, the challenger receives at least $t+1$ shares $(A_j(i), B_j(i))$ for each $j \in Q \cap C$. As for the remaining players P_j such that $j \in \mathcal{G} \subset Q$, their polynomials were honestly chosen by the challenger at Step 1.a of Dist-Keygen.

Game 1: In this game, we modify the generation of the public parameters. Namely, the vectors $(\vec{f}, \{\vec{f}_i\}_{i=0}^L)$ are chosen as $\vec{f} = (f, h)$ and $\{\vec{f}_i\}_{i=0}^L$, where

$$\begin{aligned} \vec{f}_0 &= \vec{f}^{\xi_{0,1}} \cdot (1, h)^{\xi_{0,2}} \cdot (1, h)^{\mu \cdot \zeta - \rho_0}, && \qquad (6) \\ \vec{f}_i &= \vec{f}^{\xi_{i,1}} \cdot (1, h)^{\xi_{i,2}} \cdot (1, h)^{-\rho_i}, && \qquad i \in \{1, \ldots, L\} \end{aligned}$$

with $\mu \overset{R}{\leftarrow} \{0, \ldots, L\}$, $\xi_{0,1}, \xi_{1,1}, \ldots, \xi_{L,1} \overset{R}{\leftarrow} \mathbb{Z}_p$, $\xi_{0,2}, \xi_{1,2}, \ldots, \xi_{L,2} \overset{R}{\leftarrow} \mathbb{Z}_p$ and $\rho_0, \rho_1, \ldots, \rho_L \overset{R}{\leftarrow} \{0, \ldots, \zeta - 1\}$, with $\zeta = 2q$ and where q is the number of partial signature queries. This change is only conceptual since $\{\vec{f}_i\}_{i=0}^L$ have the same distribution as in Game 0. We thus have $\Pr[S_1] = \Pr[S_0]$.

Game 2: In this game, we raise an event F_2 and let the simulator $\mathcal{B}$ abort if it occurs. Let $M_1, \ldots, M_q$ be the messages submitted to the partial signing oracle during the game. For each message $M = M[1] \ldots M[L] \in \{0, 1\}^L$, we consider the function $J(M) = \mu \cdot \zeta - \rho_0 - \sum_{i=1}^L \rho_i \cdot M[i]$. We define F_2 to be the event that either

- $J(M^\star) \neq 0$;
- There exists a message $M_j \in \{M_1, \ldots, M_q\} \backslash \{M^\star\}$ such that $J(M_j) = 0$.

We remark that $\rho_0, \rho_1, \ldots, \rho_L$ are chosen independently of $\mathcal{A}$'s view. In fact, the challenger could equivalently define $\{\vec{f}_i\}_{i=0}^L$ at the beginning of the game and only choose $\{\rho_i\}_{i=0}^L$ – along with values $\{\xi_{2,i}\}_{i=0}^L$ that are consistent with $\{\vec{f}_i\}_{i=0}^L$ – at the very end of the game, when $M^\star, M_1, \ldots, M_q$ have been defined. The same analysis as [70, 8] shows that $\Pr[S_2 \wedge \neg F_2] \geq \Pr[S_1]^2 / (27 \cdot q \cdot (L+1))$.

This follows from the fact that, for any set of queries, a lower bound on the probability of event $\neg F_2$ is $1/(2q(L+1))$.

Game 3: We modify the distribution of public parameters. Namely, $\vec{f} = (f, h)$ is chosen as before but, instead of generating $\{\vec{f}_i\}_{i=0}^L$ as in Game 2, we choose them as

$$\begin{aligned} \vec{f}_0 &= \vec{f}^{\xi_{0,1}} \cdot (1, h)^{\mu \cdot \zeta - \rho_0} && \qquad (7) \\ \vec{f}_i &= \vec{f}^{\xi_{i,1}} \cdot (1, h)^{-\rho_i}, && \qquad i \in \{1, \ldots, L\} \end{aligned}$$

which amounts to setting $\xi_{0,2} = \xi_{1,2} = \ldots = \xi_{L,2} = 0$. If the DDH assumption holds in $\mathbb{G}$, the above modification should have no noticeable impact on $\mathcal{A}$'s behavior. More precisely, if events $S_2 \wedge \neg F_2$ and $S_3 \wedge \neg F_2$ occur with noticeably different probabilities in Game 2 and Game 3, we can build a DDH distinguisher $\mathcal{B}$. The latter takes as input a DDH instance (f, h, f^α, Z), where $\alpha \overset{R}{\leftarrow} \mathbb{Z}_p$ and $Z = h^\alpha$ or $Z \in_R \mathbb{G}$. Using the random self-reducibility of DDH, we can create $L+1$ DDH instances (by extending a technique used in [58]) —which are all Diffie-Hellman tuples or random tuples depending whether $Z = h^\delta$ or not— and embed them in $\{\vec{f}_i\}_{i=0}^L$ in such a way that $\{\vec{f}_i\}_{i=0}^L$ is distributed as in Game 2 (resp. Game 3) if $Z \in_R \mathbb{G}$ (resp. $Z = h^\delta$). For this reason, we can write $|\Pr[S_3 \wedge \neg F_2] - \Pr[S_2 \wedge \neg F_2]| \leq \mathbf{Adv}^{\mathrm{DDH}_1}(\mathcal{B})$.

In Game 3, we can show that $\Pr[S_3 \wedge \neg F_2] \leq \mathbf{Adv}(\mathcal{B})^{\mathrm{DP}}(\lambda) + 1/p$ as the challenger $\mathcal{B}$ can be turned into an algorithm solving the DP problem.

Indeed, if $\neg F_2$ occurs, the adversary's forgery $\sigma^\star = (\vec{C}_z^\star, \vec{C}_r^\star, \vec{\pi}^\star)$ is a NIWI proof generated for a perfectly binding Groth-Sahai CRS $(\vec{f}, \vec{f}_{M^\star})$, which means that $\mathcal{B}$ can use $\log_f(h)$ to extract a pair $(z^\star, r^\star)$ satisfying $e(z^\star, \hat{g}_z) \cdot e(r^\star, \hat{g}_r) = e(g, \hat{g}_1)^{-1}$ from the commitments $\vec{C}_z^\star, \vec{C}_r^\star$. At the same time, for each signed message $M \neq M^\star$, the NIWI proof $(\vec{C}_z, \vec{C}_r, \vec{\pi})$ is generated for a perfectly hiding CRS $(\vec{f}, \vec{f}_M)$, so that no information about (z_i, r_i) – and thus the private key shares $(A(i), B(i))$ – leaks for these queries.

Also, while $\mathcal{A}$ is allowed queries of the form $(i, M^\star)$ to the partial signing oracle, these do not reveal any more information than the corresponding private share $SK_i = (A(i), B(i))$ itself. We thus consider partial signing queries for $M^\star$ as if they were corruption queries. Namely, at the end of the game, $\mathcal{B}$ determines which players have generated a partial signature for $M^\star$ and moves them from $\mathcal{G}$ to C. For these updated sets $\mathcal{G}$ and C, however, $\mathcal{B}$ still has the polynomials $(A_j[X], B_j[X])$ at disposal for all $j \in C$. Let us consider the aggregated additive shares

$$a_{\mathcal{G}} = \sum_{j \in \mathcal{G}} a_{j0}\,, \qquad b_{\mathcal{G}} = \sum_{j \in \mathcal{G}} b_{j0}\,,$$
$$a_{Q \cap C} = \sum_{j \in Q \cap C} a_{j0}\,, \qquad b_{Q \cap C} = \sum_{j \in Q \cap C} b_{j0}\,,$$

which are the contribution of good and corrupted players to the shared private key. We remark that the pair $(a_{\mathcal{G}}, b_{\mathcal{G}})$ is uniformly distributed in $\mathbb{Z}_p^2$ since it is the sum of uniformly random additive shares chosen by the challenger.

It is easy to see that, conditionally on $\neg F_2$, the term $a_{\mathcal{G}}$ is independent of $\mathcal{A}$'s view as long as $|C| \leq t$. This follows from the following facts: (i) Partial signing queries for $M \neq M^\star$ reveal nothing about $a_{\mathcal{G}}$ due to the perfect WI property of the proof system; (ii) $\mathcal{A}$ obtains at most $t-1$ shares of each pair (a_{j0}, b_{j0}), for $j \in \mathcal{G}$; (iii) During Step 2 of the $\mathsf{Dist\text{-}Keygen}$ protocol, while relation (5) provides the adversary with $\hat{g}_z{}^{A_j(i)} \hat{g}_r{}^{B_j(i)}$ for each $j \in \mathcal{G}$ and $i \in \{1, \ldots, n\}$, the only extractable useful information is $a_{\mathcal{G}} + \log_{\hat{g}_z}(\hat{g}_r) \cdot b_{\mathcal{G}}$. As a consequence, $a_{\mathcal{G}}$ remains independent of $\mathcal{A}$'s view as long as $|C| \leq t$.

It comes that $\mathcal{B}$ can solve the DP problem as follows. If we define $\hat{g}_{1,\mathcal{G}} = \hat{g}_z{}^{a_{\mathcal{G}}} \hat{g}_r{}^{b_{\mathcal{G}}}$, the challenger can use the sum $(a_{\mathcal{G}}, b_{\mathcal{G}})$ of its additive shares to compute a pair

$$(z^\dagger, r^\dagger) = (g^{-a_{\mathcal{G}}},\ g^{-b_{\mathcal{G}}}),$$

where $z^\dagger$ is completely unpredictable by $\mathcal{A}$, such that $e(z^\dagger, \hat{g}_z) \cdot e(r^\dagger, \hat{g}_r) = e(g, \hat{g}_{1,\mathcal{G}})^{-1}$.

Moreover, $\mathcal{B}$ can also use the group elements $(z^\star, r^\star)$ extracted from $\mathcal{A}$'s forgery to compute

$$(z^\diamond, r^\diamond) = \left(z^\star \cdot g^{\ a_{Q \cap C}},\ r^\star \cdot g^{\ b_{Q \cap C}}\right),$$

which, since $\hat{g}_1 = \hat{g}_{1,\mathcal{G}} \cdot \hat{g}_z{}^{\ a_{Q \cap C}} \cdot \hat{g}_r{}^{\ b_{Q \cap C}}$, is easily seen to also satisfy

$$e(z^\diamond, \hat{g}_z) \cdot e(r^\diamond, \hat{g}_z) = e(g, \hat{g}_{1,\mathcal{G}})^{-1} \quad .$$

Since $z^\dagger$ is independent of $\mathcal{A}$'s view, the pair $(z^\dagger/z^\diamond, r^\dagger/r^\diamond)$ forms a non-trivial solution to the DP instance $(\hat{g}_z, \hat{g}_r)$ with probability $1 - 1/p$. Since the DP assumption implies the DDH assumption in $\hat{\mathbb{G}}$, we can upper bound the adversary's advantage as

$$\mathbf{Adv}(\mathcal{A}) \leq \left(27q\,(L+1) \cdot \left[\mathbf{Adv}^{\mathrm{DDH}_1}(\mathcal{B}) + \mathbf{Adv}^{\mathrm{DDH}_2}(\mathcal{B}) + \frac{1}{p}\right]\right)^{1/2},$$

which proves the announced result. □

Acknowledgements

The first author was supported in part by the ERC Starting Grant ERC-2013-StG-335086-LATTAC and in part by the "Programme Avenir Lyon Saint-Etienne de l'Université de Lyon" in the framework of the programme "Investissements d'Avenir" (ANR-11-IDEX-0007). Part of this work while the first author was in Technicolor (France) and the last author was in Google Inc. (USA).

[1] M. Abe, S. Fehr. Adaptively secure Feldman VSS and applications to universally-composable threshold cryptography. In *Crypto '04*, *LNCS* 3152, pp. 317–334, Springer, 2004.

[2] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, M. Ohkubo. Structure-preserving signatures and commitments to group elements. In *Crypto '10*, *LNCS* 6223, pp. 209–236, Springer, 2010.

[3] M.Abe, K. Haralambiev, M. Ohkubo. Signing on elements in bilinear groups for modular protocol design. In Cryptology ePrint Archive: Report 2010/133, 2010.

[4] J. Almansa, I. Damgård, J.-B. Nielsen. Simplified threshold RSA with adaptive and proactive security. In *Eurocrypt '06*, *LNCS* 4004, pp. 593–611, Springer, 2006.

[5] P. Barreto, M. Naehrig. Pairing-friendly elliptic curves of prime order. In *SAC '05*, *LNCS* 3897, pp. 319–331, Springer, 2005.

[6] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, H. Shacham. Randomizable proofs and delegatable anonymous credentials. In *Crypto '09*, *LNCS* 5677, pp. 108–125, Springer, 2009.

[7] M. Bellare, C. Namprempre, G. Neven. Unrestricted aggregate signatures. In *ICALP '07*, *LNCS* 4596, pp. 411–422, Springer, 2007.

[8] M. Bellare, T. Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme. In *Eurocrypt '09*, *LNCS* 5479, pp. 407–424, Springer, 2009.

[9] M. Bellare, P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS '93*, pp. 62–73, ACM Press, 1993.

[10] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-group signature scheme. In *PKC '03*, *LNCS* 2567, pp. 31–46, Springer, 2003.

[11] D. Boneh, X. Boyen, H. Shacham. Short group signatures. In *Crypto '04*, *LNCS* 3152, pp. 41–55, Springer, 2004.

[12] D. Boneh, M. Franklin. Efficient Generation of shared RSA keys. In *Crypto '97*, *LNCS* 1924, pp. 425–439, Springer, 1997.

[13] D. Boneh, C. Gentry, B. Lynn, H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Eurocrypt '03*, *LNCS* 2656, pp. 416–432, Springer, 2003.

[14] D. Boneh, B. Lynn, H. Shacham. Short signatures from the Weil pairing. *J. Cryptology* 17(4), pp. 297–319, 2004. Earlier version in *Asiacrypt '01*, *LNCS* 2248, pp. 514–532, Springer, 2001.

[15] C. Boyd. Digital Multisignatures. In *Cryptography and Coding*, pp. 241–246, Oxford University Press, 1989.

[16] R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Adaptive security for threshold cryptosystems. In *Crypto '99*, *LNCS* 1666, pp. 98–115, Springer, 1999.

[17] R. Canetti, S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *Eurocrypt '99*, *LNCS* 1592, pp. 90–106, Springer, 1999.

[18] Distributed CA for Visa-MC SET Infrastructure announcement, 1997 see: `http://www.geocities.ws/rayvaneng/w0597_09.htm`

[19] J. Chen, H.-W. Lim, S. Ling, H. Wang, H. Wee. Shorter IBE and signatures via asymmetric pairings. In *Pairing '12*, *LNCS* 7708, pp. 122–140, Springer, 2012.

[20] J.-S. Coron. On the exact security of full domain hash. In *Crypto '00*, *LNCS* 1880, pp. 229–235, Springer, 2000.

[21] V. Cortier, D. Galindo, S. Glondu, M. Izabachène. Distributed ElGamal à la Pedersen: Application to Helios. In *WPES '13*, pp. 131–142, ACM Press, 2013.

[22] R. Cramer, M. Franklin, B. Schoenmakers, M. Yung. Multi-autority secret-ballot elections with linear work. In *Eurocrypt '96*, *LNCS* 1070, pp. 72–83, Springer, 1996.

[23] R. Cramer, R. Gennaro, B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Eurocrypt '97*, *LNCS* 1233, pp. 103–118, Springer, 1997.

[24] R. Cramer, I. Damgård, J.-B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Eurocrypt '01*, *LNCS* 2045, pp. 280–299, Springer, 2001.

[25] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, T. Rabin. Efficient multi-party computations secure against an adaptive adversary. In *Eurocrypt '99*, *LNCS* 1592, pp. 311–326, Springer, 1999.

[26] I. Damgård, G. Mikkelsen. Efficient, robust and constant-round distributed RSA key generation. In *TCC '10*, *LNCS* 5978, pp. 183–200, Springer, 2010.

[27] A. Dent. A Note On Game-Hopping Proofs. Cryptology ePrint Archive: Report 2006/260.

[28] A. De Santis, Y. Desmedt, Y. Frankel, M. Yung. How to share a function securely. In *STOC '94*, pp. 522-533, ACM Press, 1994.

[29] Y. Desmedt. Society and group oriented cryptography: A new concept. In *Crypto '87*, *LNCS* 293, pp. 120–127, Springer, 1987.

[30] Y. Desmedt, Y. Frankel. Threshold cryptosystems. In *Crypto '89*, *LNCS* 435, pp. 307–315, Springer, 1990.

[31] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Crypto '84*, *LNCS* 196, pp. 10–18, Springer, 1984.

[32] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *FOCS '87*, pp. 427–437, IEEE Press, 1987.

[33] P.-A. Fouque, J. Stern. Fully Distributed Threshold RSA under standard assumptions. In *Asiacrypt '01*, *LNCS* 2248, pp. 310–330, Springer, 2001.

[34] Y. Frankel, P. MacKenzie, M. Yung. Robust efficient distributed RSA-key generation. In *STOC '98*, pp. 663–672, ACM Press, 1998.

[35] Y. Frankel, P. MacKenzie, M. Yung. Adaptively-secure distributed public-key systems. In *ESA '99*, *LNCS* 1643, pp. 4–27, Springer, 1999.

[36] Y. Frankel, P. MacKenzie, M. Yung. Adaptively-secure optimal-resilience proactive RSA. In *Asiacrypt '99*, *LNCS* 1716, pp. 180–194, Springer, 1999.

[37] Y. Frankel, P. Gemmell, P. MacKenzie, M. Yung. Optimal resilience proactive public-key cryptosystems. In *FOCS '97*, pp. 384–393, IEEE Press, 1997.

[38] D. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *Eurocrypt '10*, *LNCS* 6110, pp. 44-61, Springer, 2010.

[39] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Robust threshold DSS signatures. In *Eurocrypt '96*, *LNCS* 1070, pp. 354–371, Springer, 1996.

[40] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Robust and efficient sharing of RSA functions. In *Crypto '96*, *LNCS* 1109, pp. 157–172, Springer, 1996.

[41] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Eurocrypt '99*, *LNCS* 1592, pp. 295–310, Springer, 1999.

[42] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Secure applications of Pedersen's distributed key generation protocol. In *CT-RSA '03*, *LNCS* 2612, pp. 373–390, Springer, 2003.

[43] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *J. Cryptology* 20(1), pp. 51–83, Springer, 2007.

[44] R. Gennaro, S. Halevi, H. Krawczyk, T. Rabin. Threshold RSA for dynamic and ad-hoc groups. In *Eurocrypt '08*, *LNCS* 4965, pp. 88–107, 2008.

[45] J. Groth, A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Eurocrypt '08*, *LNCS* 4965, pp. 415–432, Springer, 2008.

[46] A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung. Proactive secret sharing Or: How to cope with perpetual leakage. In *Crypto '95*, *LNCS* 963, pp. 339-352, Springer, 1995.

[47] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung. Proactive public key and signature systems. In *ACM-CCS '97*, pp. 100–110, ACM Press, 1997.

[48] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, B. Zhao. OceanStore: An architecture for global-scale persistent storage. In *ASPLOS '00*, pp. 190-201, ACM Press, 2000.

[49] S. Jarecki, A. Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In *Eurocrypt '00*, *LNCS* 1807, pp. 221–242, Springer, 2000.

[50] J. Katz, M. Yung. Threshold cryptosystems based on factoring. In *Asiacrypt '02*, *LNCS* 2501, pp. 199-205, Springer, 2002.

[51] A. Lewko, B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC '10*, *LNCS* 5978, pp. 455–479, Springer, 2010.

[52] A. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *Eurocrypt '12*, *LNCS* 5978, pp. 318–33, Springer, 2012.

[53] B. Libert, T. Peters, M. Joye, M. Yung. Linearly homomorphic structure-preserving signatures and their applications. In *Crypto '13*, *LNCS* 8043, pp. 289–307, Springer, 2013.

[54] B. Libert, M. Yung. Adaptively secure non-interactive threshold cryptosystems. *Theoretical Computer Science*, vol. 478, pp. 76–100, March 2013. Extended abstract in *ICALP 2011*, *LNCS* 6756, pp. 588–600, Springer, 2011.

[55] B. Libert, M. Yung. Non-interactive CCA2-secure threshold cryptosystems with adaptive security: New framework and constructions. In *TCC '12*, *LNCS* 7194, pp. 75–93, Springer, 2012.

[56] A. Lysyanskaya, C. Peikert. Adaptive security in the threshold setting: From cryptosystems to signature schemes. In *Asiacrypt '01*, *LNCS* 2248, pp. 331–350, Springer, 2001.

[57] T. Malkin, I. Teranishi, Y. Vahlis, M. Yung. Signatures resilient to continual leakage on memory and computation. In *TCC '11*, *LNCS* 6597, pp. 89–106, Springer, 2011.

[58] M. Naor, O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS '97*, pp. 458–467, IEEE Press, 1997.

[59] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Crypto '92*, *LNCS* 740, pp. 31–53, Springer, 2011.

[60] R. Ostrovksy, M. Yung. How to withstand mobile virus attacks. In *PODC '91*, pp. 51–59, ACM Press, 1991.

[61] T. Pedersen. A threshold cryptosystem without a trusted party. In *Eurocrypt '91*, *LNCS* 547, pp. 522–526, Springer, 1991.

[62] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. *Crypto '91*, *LNCS* 576, pp. 129–140, Springer, 1991.

[63] T. Rabin. A simplified approach to threshold and proactive RSA. In *Crypto '98*, *LNCS* 1462, pp. 89–104, Springer, 1998.

[64] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, J. Kubiatowicz. Pond: The OceanStore prototype. In *2003 USENIX Workshop on File and Storage Technologies*, Usenix Association, 2003.

[65] M. Scott. Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Cryptology ePrint Archive: Report 2002/164.

[66] A. Shamir. How to share a secret. In *Commun. ACM* 22(11), pp. 612–613, 1979.

[67] V. Shoup. Practical threshold signatures. In *Eurocrypt 2000*, *LNCS* 1807, pp. 207–220, Springer, 2000.

[68] H. Wee. Threshold and revocation cryptosystems via extractable hash proofs. In *Eurocrypt '11*, *LNCS* 6632, pp. 589–609, Springer, 2011.

[69] Z. Wang, H. Qian, Z. Li. Adaptively secure threshold signature scheme in the standard model. In *Informatica* 20(4), pp. 591-612, 2009.

[70] B. Waters. Efficient identity-based encryption without random oracles. In *Eurocrypt '05*, *LNCS* 3494, pp. 114–127, Springer, 2005.

[71] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *Crypto '09*, *LNCS* 5677, pp. 619–636, Springer, 2009.

Appendix A. Groth-Sahai non-interactive proof systems

In [45], Groth and Sahai described efficient non-interactive witness indistinguishable (NIWI) proof systems of which one instantiation relies on the SXDH assumption. This instantiation uses prime order groups and a common reference string containing three vectors $\vec{f_1}, \vec{f_2} \in \mathbb{G}^2$, where $\vec{f_1} = (g, f_1)$, $\vec{f_2} = (h, f_2)$, for some $g, h, f_1, f_2 \in \mathbb{G}$. To commit to a group element $X \in \mathbb{G}$, the prover chooses $r, s \overset{R}{\leftarrow} \mathbb{Z}_p$ and computes $\vec{C} = (1, X) \cdot \vec{f_1}^{\,r} \cdot \vec{f_2}^{\,s}$. On a perfectly sound common reference string, we have $\vec{f_2} = \vec{f_1}^{\,\xi}$, for some $\xi \in \mathbb{Z}_p$. Commitments $\vec{C} = (g^{r+\xi s}, f_1^{r+\xi s} \cdot X)$ are extractable as their distribution coincides with that of an Elgamal ciphertexts [31] and the committed X can be extracted using $\beta = \log_g(f_1)$. In the witness indistinguishability (WI) setting, the vector $\vec{f_2}$ is chosen so that $(\vec{f_1}, \vec{f_2})$ are linearly independent vectors and $\vec{C}$ is a perfectly hiding commitment. Under the DDH assumption in $\mathbb{G}$, the two kinds of CRS can be exchanged for one another without the adversary noticing.

To convince the verifier that committed variables satisfy a set of relations, the prover computes one commitment per variable and one proof element per equation. Such NIWI proofs can be efficiently generated for linear pairing-product equations, which are relations of the type

$$\prod_{i=1}^{n} e(\mathcal{X}_i, \mathcal{A}_i) = t_T \,, \tag{A.1}$$

for variables $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathbb{G}$ and constants $\mathcal{A}_1, \ldots, \mathcal{A}_n \in \hat{\mathbb{G}}$, $t_T \in \mathbb{G}_T$, $a_{ij} \in \mathbb{Z}_p$, for $i, j \in \{1, \ldots, n\}$.

Under the SXDH assumption, proving an an equation like (A.1) requires two elements of $\hat{\mathbb{G}}$.

In [6], Belenkiy *et al.* showed that Groth-Sahai proofs are perfectly randomizable. Given commitments $\{\vec{C}_{\mathcal{X}_i}\}_{i=1}^{n}$ and a NIWI proof $\vec{\pi}_{\mathrm{PPE}}$ that committed $\{\mathcal{X}\}_{i=1}^{n}$ satisfy (A.1), anyone can publicly compute re-randomized commitments $\{\vec{C}_{\mathcal{X}'_i}\}_{i=1}^{n}$ and a re-randomized proof $\vec{\pi}'_{\mathrm{PPE}}$ of the same statement. Moreover, $\{\vec{C}_{\mathcal{X}'_i}\}_{i=1}^{n}$ and $\vec{\pi}'_{\mathrm{PPE}}$ are distributed as freshly generated commitments and proof.

Appendix B. Definition and template of linearly homomorphic structure-preserving signatures

Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ be groups of prime order p such that a bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ can be efficiently computed.

A signature scheme is *structure-preserving* [3] if messages, signatures and public keys live in the groups $\mathbb{G}$ or $\hat{\mathbb{G}}$. In linearly homomorphic structure-preserving signatures, the message space $\mathcal{M}$ consists of pairs $\mathcal{M} := \mathcal{T} \times \mathbb{G}^N$, for some $N \in \mathbb{N}$, where $\mathcal{T}$ is a tag space.

Definition 4. *A* linearly homomorphic structure-preserving signature scheme *over* $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ *is a tuple of efficient algorithms* $\Sigma = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{SignDerive}, \mathsf{Verify})$ *for which the message space consists of* $\mathcal{M} := \mathcal{T} \times \mathbb{G}^N$*, for some integer* $n \in \mathsf{poly}(\lambda)$ *and some set* $\mathcal{T}$*, and with the following specifications.*

$\mathsf{Keygen}(\lambda, N)$ *is a randomized algorithm that takes in a security parameter* $\lambda \in \mathbb{N}$ *and an integer* $N \in \mathsf{poly}(\lambda)$ *denoting the dimension of vectors to be signed. It outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$*, where* pk *includes the description of a tag space* $\mathcal{T}$*, where each tag serves as a file identifier.*

$\mathsf{Sign}(\mathsf{sk}, \tau, \vec{M})$ *is a possibly randomized algorithm that takes as input a private key* sk*, a file identifier* $\tau \in \mathcal{T}$ *and a vector* $\vec{M} = (M_1, \ldots, M_N) \in \mathbb{G}^N$*. It outputs a signature* $\sigma \in \mathbb{G}^{n_s}$*, for some* $n_s \in \mathsf{poly}(\lambda)$*.*

$\mathsf{SignDerive}(\mathsf{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^{\ell})$ *is a (possibly randomized) derivation algorithm. It inputs a public key* pk*, a file identifier* τ *as well as* ℓ *pairs* $(\omega_i, \sigma^{(i)})$*, each of which consists of a coefficient* $\omega_i \in \mathbb{Z}_p$ *and a signature* $\sigma^{(i)} \in \mathbb{G}^{n_s}$*. It outputs a signature* $\sigma \in \mathbb{G}^{n_s}$ *on the vector* $\vec{M} = \prod_{i=1}^{\ell} \vec{M}_i^{\,\omega_i}$*, where* $\sigma^{(i)}$ *is a signature on* $\vec{M}_i$*.*

$\mathsf{Verify}(\mathsf{pk}, \tau, \vec{M}, \sigma)$: *is a deterministic verification algorithm that takes as input a public key* pk*, a file identifier* $\tau \in \mathcal{T}$*, a signature* σ *and a vector* $\vec{M} = (M_1, \ldots, M_N)$*. It outputs* 0 *or* 1 *depending on whether* σ *is deemed valid or not.*

In a *one-time* linearly homomorphic SPS, the tag τ can be omitted in the specification as a given key pair $(\mathsf{pk}, \mathsf{sk})$ only allows signing one linear subspace.

As in all linearly homomorphic signatures, the security requirement is that the adversary be unable to create a valid triple $(\tau^\star, \vec{M}^\star, \sigma^\star)$ for a new file identifier $\tau^\star$ or, if $\tau^\star$ is recycled from one or more honestly generated signatures, for a vector $\vec{M}^\star$ outside the linear span of the vectors that have been legitimately signed for the tag $\tau^\star$.

An important property is that the $\mathsf{SignDerive}$ algorithm must operate on vectors that are all labeled with the same tag.

Template of One-Time LHSPS. In [53], it was observed that any linearly homomorphic structure-preserving signature satisfies a certain template. In the case of one-time LHSPS, this template can be simplified as follows in bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$.

$\mathsf{Keygen}(\lambda, N)$: Given λ and the dimension $N \in \mathbb{N}$ of the vectors to be signed, choose constants n_z, m. Among these, n_s will determine the signature length while m will be the number of verification equations. Then, choose $\{\hat{F}_{j,\mu}\}_{j\in\{1,\ldots,m\},\mu\in\{1,\ldots,n_s\}}$, $\{\hat{G}_{j,k}\}_{k\in\{1,\ldots,N\},\ j\in\{j,\ldots,m\}}$ in the group $\hat{\mathbb{G}}$. The public key is $\mathsf{pk} = (\{\hat{F}_{j,\mu}\}_{j\in\{1,\ldots,m\},\mu\in\{1,\ldots,n_s\}}, \{\hat{G}_{j,k}\}_{k\in\{1,\ldots,N\},\ j\in\{j,\ldots,m\}})$ while sk contains information about the representation of public elements w.r.t. specific bases.

$\mathsf{Sign}(\mathsf{sk}, (M_1, \ldots, M_N))$: A vector $(M_1, \ldots, M_N) \in \mathbb{G}^N$ is signed by outputting a tuple of the form $\sigma = (Z_1, \ldots, Z_{n_s}) \in \mathbb{G}^{n_s}$.

$\mathsf{SignDerive}(\mathsf{pk}, \{(\omega_i, \sigma^{(i)})\}_{i=1}^{\ell})$: parses each $\sigma^{(i)}$ as $(Z_1^{(i)}, \ldots, Z_{n_s}^{(i)})$ and computes

$$Z_\mu = \prod_{i=1}^{\ell} {Z_\mu^{(i)}}^{\omega_i}, \qquad \mu \in \{1, \ldots, n_s\} \ .$$

After a possible extra re-randomization step, it outputs $(Z_1, \ldots, Z_{n_s})$.

$\mathsf{Verify}(\mathsf{pk}, \sigma, (M_1, \ldots, M_N))$: given a signature $\sigma = (Z_1, \ldots, Z_{n_s}) \in \mathbb{G}^{n_s}$ and $(M_1, \ldots, M_N)$, return 0 if $(M_1, \ldots, M_N) = (1_{\mathbb{G}}, \ldots, 1_{\mathbb{G}})$. Otherwise, return 1 if and only if the following equalities hold for $j = 1$ to m:

$$1_{\mathbb{G}_T} = \prod_{\mu=1}^{n_s} e(Z_\mu, \hat{F}_{j,\mu}) \cdot \prod_{k=1}^{N} e(M_k, \hat{G}_{j,k}), \qquad j \in \{1, \ldots, m\} \ . \tag{B.1}$$

Appendix C. Generic signatures from one-time linearly homomorphic signature

This section shows that any one-time LHSPS can be used to build fully secure signatures in the random oracle model and in the standard model.

Appendix C.1. generic construction in the random oracle model from the K-linear assumption

We first recall the following assumption which is a generalization of the DDH and Decision Linear assumptions.

Definition 5. *For $K > 0$, the* K-Linear assumption *states that, given $(g, g_1, \ldots, g_K, g_1^{a_1}, \ldots, g_K^{a_K}, \eta) \in \mathbb{G}^{2K+2}$, no PPT algorithm can decide if $\eta = g^{a_1+\cdots+a_K}$ or $\eta \in_R \mathbb{G}$.*

For $K = 1$ (resp. $K = 2$), the K-linear assumption coincides with the DDH (resp. DLIN) assumption.

Given any one-time linearly homomorphic signature $\Pi = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{SignDerive}, \mathsf{Verify})$ with homomorphic key generation that works over possibly asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$, we can use a random oracle to build the following fully secure signature scheme under the K-Linear assumption.

$\mathsf{Keygen}(\lambda)$: Given a security parameter λ, the system is set up by conducting the following steps.

1. Generate a LHSPS key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Pi.\mathsf{Keygen}(\lambda, K+1)$ for vectors of dimension $N = K + 1$. Parse pk as $\mathsf{pk} = ((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T),\ \{\hat{F}_{i,j,\mu}\}_{j\in\{1,\ldots,m\},\mu\in\{1,\ldots,n_s\}}, \{\hat{G}_{i,j,k}\}_{j\in\{j,\ldots,m\},k\in\{1,\ldots,K+1\}})$.
2. Choose a hash function $H : \{0,1\}^* \to \mathbb{G}^{K+1}$, which will be modeled as a random oracle in the security analysis.

Return the public key

$$PK = \mathsf{pk} = \big(\{\hat{F}_{j,\mu}\}_{j\in\{1,\ldots,m\},\mu\in\{1,\ldots,n_s\}}, \{\hat{G}_{j,k}\}_{j\in\{j,\ldots,m\},k\in\{1,\ldots,K+1\}}\big)$$

and the private key $SK = \mathsf{sk}$.

$\mathsf{Sign}(SK, M)$: To sign a message $M \in \{0,1\}^*$ using $SK = \mathsf{sk}$, conduct the following steps.

1. Compute $(H_1, \ldots, H_{K+1}) = H(M) \in \mathbb{G}^{K+1}$.

2. Using sk, compute and output

$$\sigma = (Z_1, \dots, Z_{n_s}) \leftarrow \Pi.\mathsf{Sign}(\mathsf{sk}, (H_1, \dots, H_{K+1})) \in \mathbb{G}^{n_s}.$$

$\mathsf{Verify}(PK, M, \sigma)$: Given $M \in \{0,1\}^L$ and a purported signature σ_i, parse σ as $(Z_1, \dots, Z_{n_s}) \in \mathbb{G}^{n_s}$ and return 1 if $\Pi.\mathsf{Verify}(\mathsf{pk}, (H_1, \dots, H_{K+1}), \sigma_i) = 1$, where $(H_1, \dots, H_{K+1}) = H(M) \in \mathbb{G}^{K+1}$. Otherwise, it returns 0.

It is immediate that the construction is correct. We prove that it is fully secure in the random oracle model if the underlying LHSPS is one-time secure.

Theorem 3. *The scheme provides existential unforgeability under chosen-message attacks in the random oracle model assuming that: (i) Π is a secure one-time LHSPS; (ii) The k-Linear assumption holds in $\mathbb{G}$.*

Proof. The proof proceeds with a sequence of games.

Game 0: This is the real attack game. The challenger generates a public key PK which is given to the adversary $\mathcal{A}$. After a series of adaptive signing queries, $\mathcal{A}$ produces a forgery $\sigma^\star$ on a message $M^\star$ that has never been submitted to the signing oracle. We define S_0 as the event that $(\sigma^\star, M^\star)$ is a valid forgery.

Game 1: This game is like Game 0 but we modify the treatment of random oracle queries. For each hash query $H(M)$, the challenger $\mathcal{B}$ flips a coin $\vartheta_M \in \{0,1\}$ that yields the value 1 with probability $1/(q_s+1)$ and 0 with probability $q_s/(q_s+1)$. At the end of the game, $\mathcal{B}$ considers the event E that either of the following conditions holds:

- For the target message $M^\star$, the coin $\vartheta_{M^\star} \in \{0,1\}$ flipped for the hash query $H(M^\star)$ was $\vartheta_{M^\star} = 0$.
- There was a partial signing query (i, M) for which $\vartheta_M = 1$.

If event E occurs, $\mathcal{B}$ halts and fails. Coron's analysis [20] shows that $\Pr[\neg E] = 1/e(q_s+1)$, which implies $\Pr[S_1] = \Pr[S_0] \cdot \Pr[\neg E] = \Pr[S_0]/e(q_s+1)$.

Game 2: This game is identical to Game 1 except for one difference. At the outset of the game, $\mathcal{B}$ picks $g, g_1, \dots, g_K \overset{R}{\leftarrow} \mathbb{G}$, $a_1, \dots, a_K \overset{R}{\leftarrow} \mathbb{Z}_p$ and defines vectors $\vec{g}_1 = (g_1^{a_1}, 1, \dots 1, g) \in \mathbb{G}^{K+1}$, $\vec{g}_2 = (1, g_2^{a_2}, 1, \dots 1, g) \in \mathbb{G}^{K+1}, \dots,$ $\vec{g}_K = (1, \dots, 1, g_K^{a_K}, g) \in \mathbb{G}^{K+1}$. Then, the distribution of random oracle outputs is modified as follow depending on the random coin $\vartheta_M \in \{0,1\}$.

- If $\vartheta_M = 0$, the challenger $\mathcal{B}$ defines $H(M) \in \mathbb{G}^{K+1}$ to be a random linear combination

$$(H_1, \dots, H_k) = \vec{g}_1^{\alpha_1} \cdots \vec{g}_K^{\alpha_K},$$

where $\alpha_1, \dots, \alpha_K \overset{R}{\leftarrow} \mathbb{Z}_p$. Note that $H(M)$ lives in a proper subspace of $\mathbb{G}^{K+1}$.

- If $\vartheta_M = 1$, $\mathcal{B}$ defines $H(M)$ as a random vector of $\mathbb{G}^{K+1}$.

Clearly, although H is no longer a true random oracle, it still looks like one in $\mathcal{A}$'s view if the K-Linear assumption holds. We have $|\Pr[S_2] - \Pr[S_1]| \leq \mathbf{Adv}^{K\text{-LIN}}(\mathcal{B})$.

In Game 2, $\mathcal{A}$ can be used by the challenger $\mathcal{B}$ to break the one-time security of Π. Specifically, $\mathcal{B}$ takes as input a public key pk for Π, which is also given as a challenge public key to $\mathcal{A}$.

Before starting its interaction with $\mathcal{A}$, our LHSPS adversary $\mathcal{B}$ invokes its own challenger to obtain linearly homomorphic signatures $\tilde{\sigma}_1, \dots, \tilde{\sigma}_K$ on the vectors $\vec{g}_1, \dots, \vec{g}_K \in \mathbb{G}^{K+1}$. If event $\neg E$ occurs, we know that $\mathcal{A}$ only obtains signatures on messages M such that $H(M)$ lives in the linear subspace spanned by $(\vec{g}_1, \dots, \vec{g}_K)$. In this case, $\mathcal{B}$ is always able to answer signing queries on its own. Indeed, it can use the coefficients $\alpha_1, \dots, \alpha_K \in \mathbb{Z}_p$ such that $H(M) = \prod_{i=1}^K \vec{g}_i^{\alpha_i}$ in order to obtain and return a signature $\sigma \leftarrow \Pi.\mathsf{SignDerive}(\mathsf{pk}, \{(\alpha_i, \tilde{\sigma}_i)\}_{i=1}^K)$ on M without any further help from its own signing oracle.

When the adversary outputs a valid forgery $(\sigma^\star, M^\star)$, we know that, if event $\neg E$ occurs, the vector $H(M^\star) = (H_1, \dots, H_{K+1})$ lands outside $\mathrm{span}(\vec{g}_1, \dots, \vec{g}_k)$, which means that $(\sigma^\star, H(M^\star))$ is a valid forgery for Π. □

Appendix C.2. Construction in the standard model from any one-time LHSPS

This section shows how to use the DLIN-based instantiation of Groth-Sahai proofs to build a fully secure digital signature from any one-time LHSPS in symmetric bilinear groups (*i.e.*, where $\mathbb{G} = \hat{\mathbb{G}}$).

In the standard model, the generic construction uses the same design principle as the scheme of Section 4 in that it proceeds by generating a non-interactive proof of knowledge of a one-time homomorphic signature on a fixed vector. The difference is that, while NIWI proofs for pairing product equations were sufficient in the scheme of Section 4, we need NIZK proofs here. Fortunately, although pairing product equations are not known to always admit efficient NIZK simulators, they can be realized while keeping constant-size signatures.

Given any one-time linearly homomorphic signature $\Pi = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{SignDerive}, \mathsf{Verify})$ in symmetric bilinear groups, we can construct a fully secure signature as follows.

$\mathsf{Keygen}(\lambda)$: The system is set up by conducting the following steps.

1. Generate a LHSPS key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Pi.\mathsf{Keygen}(\lambda, 1)$ for vectors of dimension $N = 1$. Parse pk as $\mathsf{pk} = ((\mathbb{G}, \mathbb{G}_T), \{F_{i,j,\mu}\}_{j\in\{1,\ldots,m\},\mu\in\{1,\ldots,n_s\}}, \{G_{i,j}\}_{j\in\{j,\ldots,m\}})$.
2. Choose generators $g, g_1, g_2 \overset{R}{\leftarrow} \mathbb{G}$ and define vectors $\vec{g}_1 = (g_1, 1, g)$, $\vec{g}_2 = (1, g_2, g)$. Choose $L + 1$ random vectors $\vec{f}_0, \vec{f}_1, \ldots, \vec{f}_L \overset{R}{\leftarrow} \mathbb{G}^3$.

Return the public key

$$PK = \Big((\mathbb{G}, \mathbb{G}_T),\ (\vec{g}_1, \vec{g}_2, \{\vec{f}_i\}_{i=0}^{L}),\ \mathsf{pk} = (\{F_{j,\mu}\}_{j\in\{1,\ldots,m\},\mu\in\{1,\ldots,n_s\}}, \{G_j\}_{j\in\{j,\ldots,m\}})\Big)$$

and the secret key $SK = \mathsf{sk}$.

$\mathsf{Sign}(SK, M)$: To generate a signature on a message $M = M[1] \ldots M[L] \in \{0, 1\}^L$ using the private key $SK = \mathsf{sk}$, do the following.

1. Generate $(Z_1, \ldots, Z_{n_s}) \leftarrow \Pi.\mathsf{Sign}(\mathsf{sk}, g)$ as a one-time homomorphic signature on the one-dimensional vector $g \in \mathbb{G}$.
2. Using the bits $M[1] \ldots M[L]$ of $M \in \{0, 1\}^L$, define the vector $\vec{f}_M = \vec{f}_0 \cdot \prod_{i=1}^{L} \vec{f}_i^{M[i]}$ so as to assemble a Groth-Sahai CRS $\mathbf{f}_M = (\vec{g}_1, \vec{g}_2, \vec{f}_M)$.
3. Using $\mathbf{f}_M = (\vec{f}, \vec{f}_M)$, compute Groth-Sahai commitments

$$\vec{C}_{Z,\mu} = (1_{\mathbb{G}}, Z_\mu) \cdot \vec{g}_1^{\nu_{1,\mu}} \cdot \vec{g}_2^{\nu_{2,\mu}} \cdot \vec{f}_M^{\nu_{3,\mu}}$$

to $\{Z_\mu\}_{\mu=1}^{n_s}$ for $\mu \in \{1, \ldots, n_s\}$. Then, generate NIZK proofs $\vec{\pi}_j \in \mathbb{G}^6$ that $(Z_1, \ldots, Z_{n_s})$ satisfy the verification equations

$$1_{\mathbb{G}_T} = e(g, G_j) \cdot \prod_{\mu=1}^{n_s} e(Z_\mu, F_{j,\mu}), \qquad j \in \{1, \ldots, m\}\ .$$

These NIZK proofs are obtained by proving that

$$1_{\mathbb{G}_T} = e(g, \Theta_j) \cdot \prod_{\mu=1}^{n_s} e(Z_\mu, F_{j,\mu}), \tag{C.1}$$

$$1_{\mathbb{G}_T} = e(g, \Theta_j) \cdot (g, G_j^{-1}), \qquad j \in \{1, \ldots, m\} \tag{C.2}$$

using extra variables $\Theta_j = G_j$ and their commitments $\vec{C}_{\Theta_j} = (1_{\mathbb{G}}, \Theta_j) \cdot \vec{g}_1^{\theta_{1,j}} \cdot \vec{g}_2^{\theta_{2,j}} \cdot \vec{f}_M^{\theta_{3,j}}$ as

$$\begin{aligned}\vec{\pi}_j &= (\pi_{j,1}, \pi_{i,2}, \pi_{j,3}, \pi_{j,4}, \pi_{j,5}, \pi_{j,6}) \\ &= \Big(g^{-\theta_{1,j}} \cdot \prod_{\mu=1}^{n_s} F_{j,\mu}^{-\nu_{1,\mu}},\ g^{-\theta_{2,j}} \cdot \prod_{\mu=1}^{n_s} F_{j,\mu}^{-\nu_{2,\mu}}, g^{-\theta_{3,j}} \cdot \prod_{\mu=1}^{n_s} F_{j,\mu}^{-\nu_{3,\mu}},\ g^{-\theta_{1,j}},\ g^{-\tau_{2,j}},\ g^{-\theta_{3,j}}\Big)\ .\end{aligned}$$

Return the partial signature $\sigma_i = (\vec{C}_{Z,1}, \ldots, \vec{C}_{Z,n_s}, \vec{C}_{\Theta_1}, \ldots, \vec{C}_{\Theta_m}, \vec{\pi_1}, \ldots, \vec{\pi_m}) \in \mathbb{G}^{3n_s+9m}$.

$\mathsf{Verify}(PK, M, \sigma)$: Given a L-bit message $M \in \{0,1\}^L$ and a purported signature σ, parse σ as $(\vec{C}_{Z,1}, \ldots, \vec{C}_{Z,n_s}, \vec{C}_{\Theta_1}, \ldots, \vec{C}_{\Theta_m}, \vec{\pi_1}, \ldots, \vec{\pi_m}) \in \mathbb{G}^{3n_s+9m}$, where $\vec{\pi_j} = (\pi_{j,1}, \pi_{j,2}, \pi_{j,3}, \pi_{j,4}, \pi_{j,5}, \pi_{j,6})$ for each $j \in \{1, \ldots, m\}$. Return 1 if it satisfies

$$1_{\mathbb{G}_T} = E(g, \vec{C}_{\Theta_j}) \cdot \prod_{\mu=1}^{n_s} E(F_\mu, \vec{C}_{Z,\mu}) \cdot E(\vec{g}_1, \pi_{j,1}) \cdot E(\vec{g}_2, \pi_{j,2}) \cdot E(\vec{f}_M, \pi_{j,3})$$
$$1_{\mathbb{G}_T} = E(g, \vec{C}_{\Theta_j}) \cdot E(g^{-1}, (1_{\mathbb{G}}, 1_{\mathbb{G}}, G_j)) \cdot E(\vec{g}_1, \pi_{j,4}) \cdot E(\vec{g}_2, \pi_{j,5}) \cdot E(\vec{f}_M, \pi_{j,6})$$

where $\vec{f}_M = \vec{f}_0 \cdot \prod_{i=1}^{L} \vec{f}_i^{M[i]}$.

The proof of security proceeds in the same way as the proof of Theorem 2. The main change is that we need NIZK proofs for pairing product equations here.

Theorem 4. *The scheme provides existential unforgeability assuming that* Π *is a secure one-time LHSPS and that the DLIN assumption holds in* $\mathbb{G}$*.*

Proof. The proof proceeds with a sequence of games.

Game 0: This is the actual game. We call S_0 the event that the adversary $\mathcal{A}$ wins.

Game 1: We modify the generation of the public key. Namely, the vectors $(\vec{g}_1, \vec{g}_2, \{\vec{f}_i\}_{i=0}^L)$ are chosen as $\vec{g}_1 = (g_1, 1, g)$, $\vec{g}_2 = (1, g_2, g)$ and $\{\vec{f}_i\}_{i=0}^L$, where

$$\vec{f}_0 = \vec{g}_1^{\xi_{0,1}} \cdot \vec{g}_1^{\xi_{0,2}} \cdot (1,1,g)^{\xi_{0,3}} \cdot (1,1,g)^{\mu\cdot\zeta-\rho_0} \tag{C.3}$$
$$\vec{f}_i = \vec{g}_1^{\xi_{i,1}} \cdot \vec{g}_2^{\xi_{i,2}} \cdot (1,1,g)^{\xi_{i,3}} \cdot (1,g)^{-\rho_i}, \qquad i \in \{1, \ldots, L\}$$

with $\mu \xleftarrow{R} \{0, \ldots, L\}$, $\xi_{0,1}, \xi_{1,1}, \ldots, \xi_{L,1} \xleftarrow{R} \mathbb{Z}_p$, $\xi_{0,2}, \xi_{1,2}, \ldots, \xi_{L,2} \xleftarrow{R} \mathbb{Z}_p$, $\xi_{0,3}, \xi_{1,3}, \ldots, \xi_{L,3} \xleftarrow{R} \mathbb{Z}_p$ and $\rho_0, \rho_1, \ldots, \rho_L \xleftarrow{R} \{0, \ldots, \zeta - 1\}$, with $\zeta = 2q$ and where q is the number of partial signature queries. We note that $\{\vec{f}_i\}_{i=0}^L$ have the same distribution as in Game 0, so that $\Pr[S_1] = \Pr[S_0]$.

Game 2: We raise an event F_2 and let the simulator $\mathcal{B}$ abort if it occurs. Let $M_1, \ldots, M_q$ be the messages submitted by the adversary to the partial signing oracle during the game. For each message $M = M[1] \ldots M[L] \in \{0,1\}^L$, we define the function $J(M) = \mu \cdot \zeta - \rho_0 - \sum_{i=1}^L \rho_i \cdot M[i]$ and define F_2 to be the event that either

- $J(M^\star) \neq 0$;
- There exists a message $M_j \in \{M_1, \ldots, M_q\}$ such that $J(M_j) = 0$.

We remark that $\rho_0, \rho_1, \ldots, \rho_L$ are completely independent of $\mathcal{A}$'s view. The same analysis as [70, 8] shows that $\Pr[S_2 \wedge \neg F_2] \geq \Pr[S_1]^2/(27 \cdot q \cdot (L+1))$.

Game 3: We change the distribution of public parameters. Namely, $\vec{g}_1$ and $\vec{g}_2$ are chosen as before but, instead of generating $\{\vec{f}_i\}_{i=0}^L$ as in Game 2, they are generated as

$$\vec{f}_0 = \vec{g}_1^{\xi_{0,1}} \cdot \vec{g}_2^{\xi_{0,2}} \cdot (1,1,g)^{\mu\cdot\zeta-\rho_0} \tag{C.4}$$
$$\vec{f}_i = \vec{g}_1^{\xi_{i,1}} \cdot \vec{g}_1^{\xi_{i,2}} \cdot (1,1,g)^{-\rho_i}, \qquad i \in \{1, \ldots, L\}$$

which amounts to setting $\xi_{0,3} = \xi_{1,3} = \ldots = \xi_{L,3} = 0$. Under the DLIN assumption in $\mathbb{G}$, this change should not noticeable modify the probability of event $S_2 \wedge \neg F_2$. Concretely, if events $S_2 \wedge \neg F_2$ and $S_3 \wedge \neg F_2$ occur with significantly different probabilities in Game 2 and Game 3, we can construct a DLIN distinguisher $\mathcal{B}$. The latter takes as input a DLIN instance $(g, g_1, g_2, g_1^a, g_2^b, Z)$, where $a, b \xleftarrow{R} \mathbb{Z}_p$ and $Z = g^{a+b}$ or $Z \in_R \mathbb{G}$.

Using the random self-reducibility of DLIN, we can create $L + 1$ DLIN instances – which are all linear tuples (resp. random tuples) if $Z = g^{a+b}$ (resp. $Z \in_R \mathbb{G}$) —and embed them in $\{\vec{f_i}\}_{i=0}^{L}$ in such a way that $\{\vec{f_i}\}_{i=0}^{L}$ is distributed as in Game 2 (resp. Game 3) if $Z \in_R \mathbb{G}$ (resp. $Z = g^{a+b}$). For this reason, we obtain the inequality $|\Pr[S_3 \wedge \neg F_2] - \Pr[S_2 \wedge \neg F_2]| \leq \mathbf{Adv}^{\mathrm{DLIN}}(\mathcal{B})$.

Game 4: We modify the signature generation oracle. Namely, at each signing query on a message $M = M[1] \dots M[L]$, the challenger assembles the Groth-Sahai CRS $(\vec{g}_1, \vec{g}_2, \vec{f}_M)$, where

$$\vec{f}_M = \vec{f}_0 \cdot \prod_{i=1}^{L} \vec{f}_i^{M[i]} = \vec{g}_1^{K_1(M)} \cdot \vec{g}_2^{K_2(M)} \cdot (1, 1, g)^{J(M)},$$

where $K_1(M) = \xi_{0,1} + \sum_{i=1}^{L} \xi_{i,1}$, $K_2(M) = \xi_{0,2} + \sum_{i=1}^{L} \xi_{i,2}$. Using the trapdoor information $(K_1(M), K_2(M), J(M))$, the challenger generates a simulated NIZK proof $\vec{\pi}_{i,j}$.

To generate a proof for Equation (C.1), $\mathcal{B}$ uses the assignment $\Theta_j = 1_{\mathbb{G}}$ and $Z_\mu = 1_{\mathbb{G}}$ for $\mu \in \{1, \dots, n_s\}$, $j \in \{1, \dots, m\}$, which allows honestly generating a proof for (C.1). As for equation (C.2), the challenger can use $(K_1(M), K_2(M), J(M))$ to generate a fake proof that $\Theta_j = G_j$ by trapdoor opening the commitment to $\vec{C}_{\Theta_j} = \vec{g}_1^{\theta_{1,j}} \cdot \vec{g}_2^{\theta_{2,j}} \cdot \vec{f}_M^{\theta_{3,j}}$ which was computed as a commitment to $1_{\mathbb{G}}$. Namely, it can compute

$$\pi_{j,4} = g^{\theta_{1,j}} \cdot G_j^{\frac{K_1(M)}{J(M)}}, \qquad \pi_{j,5} = g^{\theta_{2,j}} \cdot G_j^{\frac{K_2(M)}{J(M)}}, \qquad \pi_{j,6} = g^{\theta_{3,j}} \cdot G_j^{-\frac{1}{J(M)}},$$

which forms a valid proof for equation (C.2) and is always computable when $J(M) \neq 0$. Given that simulated NIZK proofs are perfectly indistinguishable from real proofs on a witness indistinguishable CRS, $\mathcal{A}$'s view is the same as in Game 3, so that $\Pr[S_4 \wedge \neg F_2] = \Pr[S_3 \wedge \neg F_2]$.

In Game 4, we claim that $\mathcal{B}$ can be turned into a LHSPS forger running in about the same time as $\mathcal{A}$ and for which $\Pr[S_4 \wedge \neg F_2] \leq \mathbf{Adv}^{\mathrm{LHSPS}}(\mathcal{B})$.

To break the security of Π, $\mathcal{B}$ interacts with its own challenger that provides it with $\mathsf{pk} = (\{F_{j,\mu}\}_{j \in \{1,\dots,m\}, \mu \in \{1,\dots,n_s\}}, \{G_j\}_{j \in \{j,\dots,m\}})$ as a public key for Π. Then, $\mathcal{B}$ generates $(\vec{g}_1, \vec{g}_2, \{\vec{f_i}\}_{i=0}^{L})$ as in Game 4 so as to form PK, which is given to $\mathcal{A}$.

The adversary's signing queries are answered by simulating NIZK proofs as in Game 4 in such a way that $\mathcal{B}$ never has to invoke its own signing oracle. When the adversary outputs a forgery $(\sigma^\star, M^\star)$, we know that $(\vec{g}_1, \vec{g}_2, \vec{f}_M)$ is an extractable Groth-Sahai CRS if event $\neg F_2$ occurs. Using the discrete logarithms $(\log_g(g_1), \log_g(g_2))$, $\mathcal{B}$ can thus extract a valid LHSPS from the Groth-Sahai commitments contained in $\sigma^\star$. □

Appendix D. Deferred proofs for Theorem 1

Lemma 1. *Under the DDH assumption in $\mathbb{G}$, Game 2 is computationally indistinguishable from Game 1.*

Proof. The proof is by contradiction and builds a straightforward DDH distinguisher $\mathcal{B}$ from an adversary $\mathcal{A}$ that has noticeably different behaviors in Game 1 and Game 2.

The reduction $\mathcal{B}$ receives as input a pair (g, g^x, g^y, T) and has to decide whether $T = g^{xy}$ or $T \in_R \mathbb{G}$. To this end, algorithm $\mathcal{B}$ begins by generating PK, **SK** and **VK** in the same way as in the real scheme. In addition, it defines $h = g^y$. Throughout the game, $\mathcal{B}$ always answers partial signing queries and corruption queries faithfully. However, the treatment of random oracle queries $H(M)$ depends on the value of the biased coin $\delta_M \in \{0, 1\}$. Namely, when $\delta_M = 0$, $\mathcal{B}$ uses the random self-reducibility of DDH and builds many DDH instances out of one.

- If $\delta_M = 0$, $\mathcal{B}$ chooses $\alpha_M, \beta_M \xleftarrow{R} \mathbb{Z}_p$, computes

$$(g_M, h_M) = ((g^x)^{\alpha_M} \cdot g^{\beta_M}, T^{\alpha_M} \cdot (g^y)^{\beta_M})$$

and programs the random oracle so as to have $H(M) = (H_1, H_2) = (g_M, h_M)$. Observe that, if $T = g^{xy}$, the pair $(H_1, H_2) = (g_M, h_M)$ has the same distribution as in Game 2 as it can be written $(H_1, H_2) = (g^{z_M}, h^{z_M})$ with $z_M = x \cdot \alpha_M + \beta_M$. In contrast, if $T \in_R \mathbb{G}$, we can write $T = g^{xy+z}$ for some random $z \in_R \mathbb{Z}_p$. In this case, we have $(H_1, H_2) = (g^{z_M}, h^{z_M + x \cdot \alpha_M})$, so that $(H_1, H_2) \in_R \mathbb{G}^2$.

- If $\delta_M = 1$, $\mathcal{B}$ draws $g_M, h_M \overset{R}{\leftarrow} \mathbb{G}^2$ and defines $H(M) = (g_M, h_M)$.

When $\mathcal{A}$ terminates, $\mathcal{B}$ outputs a random bit if event E has come about during the game. Otherwise, $\mathcal{B}$ outputs 1 if $b' = b$ and 0 otherwise.

Clearly, if $T = g^{xy}$, $\mathcal{A}$'s view is exactly the same as in Game 2. In contrast, if T is uniform in $\mathbb{G}$, $\mathcal{B}$ is rather playing Game 1 with the adversary. □

Appendix E. A construction based on the DLIN assumption

This section shows a variant of our first construction based on the Decision Linear assumption [11], which is believed to be strictly weaker than SXDH. The scheme retains adaptive security in the random oracle model even in groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ endowed with efficiently computable isomorphisms between $\mathbb{G}$ and $\hat{\mathbb{G}}$.

Definition 6 ([11]). *The* Decision Linear problem *(DLIN) in* $\mathbb{G}$*, is to distinguish the distributions* $(g^a, g^b, g^{ac}, g^{bd}, g^{c+d})$ *and* $(g^a, g^b, g^{ac}, g^{bd}, g^z)$*, with* $a, b, c, d \overset{R}{\leftarrow} \mathbb{Z}_p$*,* $z \overset{R}{\leftarrow} \mathbb{Z}_p$*.*

For convenience, we also consider the following problem.

Definition 7. *The* Simultaneous Double Pairing problem *(SDP) in* $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ *is, given a tuple* $(\hat{g}_z, \hat{g}_r, \hat{h}_z, \hat{h}_u) \in \hat{\mathbb{G}}^4$*, to find a non-trivial triple* $(z, r, u) \in \mathbb{G}^3 \backslash \{(1_{\mathbb{G}}, 1_{\mathbb{G}}, 1_{\mathbb{G}})\}$ *such that the equalities* $e(z, \hat{g}_z) \cdot e(r, \hat{g}_r) = 1_{\mathbb{G}_T}$ *and* $e(z, \hat{h}_z) \cdot e(u, \hat{h}_u) = 1_{\mathbb{G}_T}$ *are satisfied.*

Any algorithm solving the SDP problem immediately yields a DLIN distinguisher in the group $\hat{G}$. From a DLIN$_2$ instance $(\hat{g}, \hat{g}^a, \hat{g}^{ac}, \hat{g}^b, \hat{g}^{bd}, \eta)$, we create an SDP instance $(\hat{g}^{ac}, \hat{g}^a, \hat{g}^{bd}, \hat{g}^b)$. If the SDP solver finds (z, r, u) such that $e(z, \hat{g}^{ac}) \cdot e(r, \hat{g}^a) = 1_{\mathbb{G}_T}$ and $e(z, \hat{g}^{bd}) \cdot e(u, \hat{g}^b) = 1_{\mathbb{G}_T}$, we know that $r = z^{-c}$ and $u = z^{-d}$. To decide if $\eta = \hat{g}^{c+d}$, it is sufficient to check whether $e(r \cdot u, \hat{g}) \cdot e(z, \eta) = 1_{\mathbb{G}_T}$.

Here, we need public parameters params consisting of four generators $\hat{g}_z, \hat{g}_r, \hat{h}_z, \hat{h}_u \in_R \hat{\mathbb{G}}$. Again, $\hat{g}_r, \hat{h}_z, \hat{h}_u$ can be derived from a random oracle so as to avoid the need to generate them in a in a distributed manner while still making sure that no party knows their discrete logarithms.

$\mathsf{Dist\text{-}Keygen}(\mathsf{params}, \lambda, t, n)$: Given $\mathsf{params} = \{(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), \hat{g}_z, \hat{g}_r, \hat{h}_z, \hat{h}_u, , H\}$, a security parameter λ and integers $t, n \in \mathbb{N}$ such that $n \geq 2t + 1$, the players $P_1, \ldots, P_n$ proceed as follows.

1. Each player P_i shares random triples $\{(a_{ik0}, b_{ik0}, c_{ik0})\}_{k=1}^3$. To this end, he does the following:
   (a) For each $k \in \{1, 2, 3\}$, choose random polynomials $A_{ik}[X] = a_{ik0} + a_{ik1}X + \cdots + a_{ikt}X^t$, $B_{ik}[X] = b_{ik0} + b_{ik1}X + \cdots + b_{ikt}X^t$ and $C_{ik}[X] = c_{ik0} + c_{ik1}X + \cdots + c_{ikt}X^t \in \mathbb{Z}_p[X]$ of degree t and broadcast
   $$\hat{V}_{ik\ell} = \hat{g}_z^{\,a_{ik\ell}} \hat{g}_r^{\,b_{ik\ell}}, \qquad \hat{W}_{ik\ell} = \hat{h}_z^{\,a_{ik\ell}} \hat{h}_u^{\,c_{ik\ell}}, \qquad \forall \ell \in \{0, \ldots, t\} .$$
   (b) For $j = 1$ to n, send $\{(A_{ik}(j), B_{ik}(j), C_{ik}(j))\}_{k=1}^3$ to P_j.
2. For each received shares $\{(A_{jk}(i), B_{jk}(i), C_{jk}(i))\}_{k=1}^3$, player P_i verifies that
   $$\hat{g}_z^{A_{jk}(i)}\, \hat{g}_r^{B_{jk}(i)} = \prod_{\ell=0}^{t} \hat{V}_{jk\ell}^{\,i^\ell}, \qquad \hat{h}_z^{A_{jk}(i)} \hat{h}_u^{C_{jk}(i)} = \prod_{\ell=0}^{t} \hat{W}_{jk\ell}^{\,i^\ell}, \qquad \forall k \in \{1, 2, 3\} . \tag{E.1}$$
   If these equalities do not both hold, P_i broadcasts a complaint against P_j.
3. Any player who receives strictly more than t complaints is disqualified. Each player P_i who received a complaint from another player P_j responds by returning the correct shares $\{(A_{ik}(j), B_{ik}(j), C_{ik}(j))\}_{k=1}^3$. If any of these new shares does not satisfy (E.1), P_i is disqualified. Let $Q \subset \{1, \ldots, n\}$ be the set of players who are not disqualified at the end of Step 3.

4. The public key is obtained as $PK = \{\hat{g}_k, \hat{h}_k\}_{k=1}^3$, where $\hat{g}_k = \prod_{i \in Q} \hat{V}_{ik0} = \hat{g}_z^{\sum_{i \in Q} a_{ik0}} \cdot \hat{g}_r^{\sum_{i \in Q} b_{ik0}}$ and $\hat{h}_k = \prod_{i \in Q} \hat{W}_{ik0} = \hat{h}_z^{\sum_{i \in Q} a_{ik0}} \cdot \ hath_u^{\sum_{i \in Q} c_{ik0}}$. Each P_i locally defines his private key share
$$SK_i = \{(A_k(i), B_k(i), C_k(i))\}_{k=1}^3 = \{(\sum_{j \in Q} A_{jk}(i), \sum_{j \in Q} B_{jk}(i), \sum_{j \in Q} C_{jk}(i))\}_{k=1}^3$$
and anyone can publicly compute his verification key
$$VK_i = \Big(\{\hat{U}_{k,i} = \hat{g_z}^{A_k(i)} \hat{g_r}^{B_k(i)}\}_{k=1}^3, \{\hat{Z}_{k,i} = \hat{h_z}^{A_k(i)} \hat{h_u}^{C_k(i)}\}_{k=1}^3\Big)$$
as $(\hat{U}_{k,i}, \hat{Z}_{k,i}) = \Big(\prod_{j \in Q} \prod_{\ell=0}^t \hat{V}_{jk\ell}^{i^\ell}, \ \prod_{j \in Q} \prod_{\ell=0}^t \hat{W}_{jk\ell}^{i^\ell}\Big)$ for each $i \in Q$ and $k \in \{1, 2, 3\}$.
For any disqualified player $i \in \{1, \ldots, n\} \backslash Q$, the i-th private key share is implicitly set as $SK_i = \{(0, 0, 0)\}_{k=1}^3$ and the corresponding verification key is $VK_i = (\{1_{\hat{\mathbb{G}}}, 1_{\hat{\mathbb{G}}}\}_{k=1}^3)$.

The vector of private key shares is $\mathbf{SK} = (SK_1, \ldots, SK_n)$ while the corresponding vector of verification keys $\mathbf{VK} = (VK_1, \ldots, VK_n)$ and the public key, which consists of
$$PK = \Big(\mathsf{params}, \ \{\hat{g}_k, \hat{h}_k\}_{k=1}^3\Big) \ .$$

$\mathsf{Share\text{-}Sign}(i, SK_i, M)$: To generate a partial signature on a message $M \in \{0, 1\}^*$ using its private key share $SK_i = \{(A_k(i), B_k(i), C_k(i))\}_{k=1}^3$, the i-th server first computes the hash value $(H_1, H_2, H_3) = H(M) \in \mathbb{G}^3$ and generates the partial signature as
$$z_i = \prod_{k=1}^{3} H_k^{-A_k(i)}, \qquad r_i = \prod_{k=1}^{3} H_k^{-B_k(i)}, \qquad u_i = \prod_{k=1}^{3} H_k^{-C_k(i)} \ .$$
The partial signature consists of $\sigma_i = (z_i, r_i, u_i) \in \mathbb{G}^3$.

$\mathsf{Share\text{-}Verify}(PK, VK_i, M, (i, \sigma_i))$: Given the partial signatures $\sigma_i = (z_i, r_i, u_i) \in \mathbb{G}^3$ and the verification key $VK_i = (\{\hat{U}_{k,i}, \hat{Z}_{k,i}\}_{k=1}^3)$, compute $(H_1, H_2, H_3) = H(M) \in \mathbb{G}^3$ and return 1 iff
$$e(z_i, \hat{g}_z) \cdot e(r_i, \hat{g}_r) \cdot \prod_{k=1}^{3} e(H_k, \hat{U}_{k,i}) = 1_{\mathbb{G}_T} , \qquad e(z_i, \hat{h}_z) \cdot e(u_i, \hat{h}_u) \cdot \prod_{k=1}^{3} e(H_k, \hat{Z}_{k,i}) = 1_{\mathbb{G}_T} \ .$$

$\mathsf{Combine}(PK, \mathbf{VK}, M, \{(i, \sigma_i)\}_{i \in S})$: Given a $(t + 1)$-set with valid shares $\{(i, \sigma_i)\}_{i \in S}$, parse the signature share σ_i as $(z_i, r_i, u_i) \in \mathbb{G}^3$ for each $i \in S$. Then, compute
$$(z, r, u) = \Big(\prod_{i \in S} z_i^{\Delta_{i,S}(0)}, \ \prod_{i \in S} r_i^{\Delta_{i,S}(0)}, \ \prod_{i \in S} u_i^{\Delta_{i,S}(0)}\Big)$$
by Lagrange interpolation in the exponent. Return the pair $(z, r, u) \in \mathbb{G}^3$.

$\mathsf{Verify}(PK, M, \sigma)$: Given $\sigma = (z, r, u) \in \mathbb{G}^3$, compute $(H_1, H_2, H_3) = H(M) \in \mathbb{G}^3$ and return 1 if and only if (z, r, u) satisfy the equalities
$$e(z, \hat{g}_z) \cdot e(r, \hat{g}_r) \cdot \prod_{k=1}^{3} e(H_k, \hat{g}_k) = 1_{\mathbb{G}_T} , \qquad e(z, \hat{h}_z) \cdot e(u, \hat{h}_u) \cdot \prod_{k=1}^{3} e(H_k, \hat{h}_k) = 1_{\mathbb{G}_T} \ .$$

The scheme can be proved adaptively secure under the DLIN assumption. The proof is, *mutatis mutandis*, identical to the proof of Theorem 1 and omitted here.

Theorem 5. *The scheme provides adaptive security in the random oracle model if the DLIN assumption holds in both $\mathbb{G}$ and $\hat{\mathbb{G}}$. Concretely, for any PPT adversary $\mathcal{A}$, there exist efficient DLIN distinguishers $\mathcal{B}_1$ and $\mathcal{B}_2$ in the groups $\mathbb{G}$ and $\hat{\mathbb{G}}$, respectively, such that*
$$\mathbf{Adv}(\mathcal{A}) \leq e \cdot (q_s + 1) \cdot \Big(\mathbf{Adv}^{\mathrm{DLIN}_1}(\mathcal{B}_1) + \mathbf{Adv}^{\mathrm{DLIN}_2}(\mathcal{B}_2) + \frac{1}{p}\Big),$$
where q_s is the number of signing queries and e is the base for the natural logarithm.

Appendix F. A construction supporting signature aggregation

In this section, we show how to extend the scheme of Section 3 so as to enable unrestricted signature aggregation. We consider non-interactive distributed aggregate signatures which are threshold signatures augmented with two additional algorithms named Aggregate and Aggregate-Verify. The former takes as input a set of public keys $\{PK_j\}_{j=1}^{\ell}$ with the corresponding message-signature pairs $\{(M_j, \sigma^{(j)})\}_{j=1}^{\ell}$ and returns a constant-size aggregate signature σ. The corresponding aggregate verification algorithm inputs an aggregate signature σ and a set of ℓ pairs $\{(PK_j, M_j)\}_{j=1}^{\ell}$ and returns 1 if and only if it accepts σ as convincing evidence that signer j indeed signed M_j.

The security of an aggregate threshold signature is defined via a game which is identical to the game of Definition 1 except that, instead of outputting a single message-signature pair in stage 3, the adversary outputs an aggregate signature $\sigma^\star$ and a list of pairs $\{(PK_j, M_j^\star)\}_{j=1}^{\ell}$. Formally, the adversary is given public parameters params and interacts with the challenger during the generation of the challenge public key $PK^\star$. In the second phase, the adversary is granted access to an adaptive corruption oracle as well as a partial signing oracle for all servers that hold a share SK_i of the matching private key $SK^\star$. Eventually, the adversary outputs $\sigma^\star$ and $\{(PK_j, M_j^\star)\}_{j=1}^{\ell}$. It wins under the following conditions: (i) Aggregate-Verify($\{(PK_j, M_j)\}_{j=1}^{\ell}, \sigma^\star$) $= 1$; (ii) The set $\{(PK_j, M_j^\star)\}_{j=1}^{\ell}$ contains at least one pair $(PK_j, M_j^\star)$ such that $PK_j = PK^\star$ and, if $\mathcal{S} \subset \{1, \ldots, n\}$ denotes the set of players for which $\mathcal{A}$ obtained a partial signature on $M_j^\star$, then $|\mathcal{S} \cup C| \leq t$.

Here, we assume public parameters params containing two extra generators $g, h \in_R \mathbb{G}$ in addition to those of the scheme in Section 3. The distributed key generation phase is as in Section 3, with the difference that the players jointly generate a one-time linearly homomorphic signature on the vector $(g, h) \in \mathbb{G}^2$ and incorporate this signature in the public key. In the aggregate verification process, the verifier performs a sanity check on public keys and only accepts aggregate signatures where all involved public keys contain a linearly homomorphic signature on (g, h).

Looking ahead, these homomorphic signatures will be useful in the security proof when it comes to extract an ordinary signature from the adversary's fake aggregate signature. More specifically, they allow the reduction to recover the contribution of adversarially generated keys to the aggregate signature in order to remove it from the aggregate. To this end, another solution would have been to assume registered public keys (as done in Boldyreva's multi-signatures [10]) and force the adversary to reveal the underlying private key whenever it generates a public key for itself. We found it more natural to introduce a built-in proof of validity in each public key.

Dist-Keygen(params, λ, t, n): Given params $= \{(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, h, \hat{g}_z, \hat{g}_r, H\}$, a security parameter λ and integers $t, n \in \mathbb{N}$ with $n \geq 2t + 1$, the players proceed as follows.

1. Each player P_i does the following:
   (a) For each $k \in \{1, 2\}$, choose random polynomials $A_{ik}[X] = a_{ik0} + a_{ik1}X + \cdots + a_{ikt}X^t$, $B_{ik}[X] = b_{ik0} + b_{ik1}X + \cdots + b_{ikt}X^t \in \mathbb{Z}_p[X]$ of degree t and broadcast
   $$\hat{W}_{ik\ell} = \hat{g}_z^{\,a_{ik\ell}} \hat{g}_r^{\,b_{ik\ell}}, \qquad \forall \ell \in \{0, \ldots, t\} .$$
   Then, broadcast values
   $$Z_{i0} = g^{-a_{i10}} h^{-a_{i20}}, \qquad R_{i0} = g^{-b_{i10}} h^{-b_{i20}},$$
   which form a homomorphic signature on the vector (g, h) for the public key $(\hat{W}_{i10}, \hat{W}_{i20})$.
   (b) For $j = 1$ to n, send $\{(A_{ik}(j), B_{ik}(j))\}_{k=1}^{2}$ to P_j.
2. For each received shares $\{(A_{jk}(i), B_{jk}(i))\}_{k=1}^{2}$, player P_i verifies that
   $$\hat{g}_z^{A_{jk}(i)} \hat{g}_r^{B_{jk}(i)} = \prod_{\ell=0}^{t} \hat{W}_{jk\ell}^{\,i^\ell} \qquad \text{for } k = 1, 2 . \tag{F.1}$$
   and $e(Z_{j0}, \hat{g}_z) \cdot e(R_{j0}, \hat{g}_r) \cdot e(g, \hat{W}_{j10}) \cdot e(h, \hat{W}_{j20}) = 1_{\mathbb{G}_T}$. If equalities (F.1) do not both hold, P_i broadcasts a complaint against P_j.

3. Any player P_i who sent incorrect verification values (Z_{i0}, R_{i0}) or received more than t complaints is immediately disqualified. Each player P_i who received a complaint from another player P_j responds by returning the correct shares $\{(A_{ik}(j), B_{ik}(j))\}_{k=1}^{2}$. If any of these new shares fail to satisfy (F.1), P_i is disqualified. Let $Q \subset \{1, \ldots, n\}$ be the set of non-disqualified players at the end of Step 3.
4. The public key PK is obtained as $PK = (\{\hat{g}_k\}_{k=1}^{2}, Z, R)$, where

$$\hat{g}_k = \prod_{i \in Q} \hat{W}_{ik0} = \hat{g}_z^{\sum_{i \in Q} a_{ik0}} \hat{g}_r^{\sum_{i \in Q} b_{ik0}}, \qquad Z = \prod_{i \in Q} Z_{i0}, \qquad R = \prod_{i \in Q} R_{i0} \ .$$

Each P_i locally defines his private key share

$$SK_i = \{(A_k(i), B_k(i))\}_{k=1}^{2} = \{(\textstyle\sum_{j \in Q} A_{jk}(i), \sum_{j \in Q} B_{jk}(i))\}_{k=1}^{2}$$

and anyone can publicly compute his verification key $VK_i = (\hat{V}_{1,i}, \hat{V}_{2,i})$ as

$$VK_i = (\hat{g}_z{}^{A_1(i)} \hat{g}_r{}^{B_1(i)}, \hat{g}_z{}^{A_2(i)} \hat{g}_r{}^{B_2(i)}) = \left(\prod_{j \in Q} \prod_{\ell=0}^{t} \hat{W}_{j1\ell}^{\,i^\ell}, \ \prod_{j \in Q} \prod_{\ell=0}^{t} \hat{W}_{j2\ell}^{\,i^\ell} \right) \ .$$

Any disqualified player $i \in \{1, \ldots, n\} \backslash Q$ has his private key share (resp. its verification key) implicitly set as $SK_i = \{(0, 0)\}_{k=1}^{2}$ (resp. $VK_i = (1_{\hat{\mathbb{G}}}, 1_{\hat{\mathbb{G}}})$).

The public key finally consists of

$$PK = (\mathsf{params}, \ (\hat{g_1}, \hat{g_2}), \ Z, \ R) \ .$$

Share-Sign(i, SK_i, M): To create a partial signature on a message $M \in \{0, 1\}^*$ using his private key share $SK_i = \{(A_k(i), B_k(i))\}_{k=1}^{2}$, P_i first computes $(H_1, H_2) = H(PK \| M) \in \mathbb{G} \times \mathbb{G}$ and obtains the partial signature as $\sigma_i = (z_i, r_i) = (\prod_{k=1}^{2} H_k^{-A_k(i)}, \prod_{k=1}^{2} H_k^{-B_k(i)})$.

As we can see, Share-Sign is as in Section 3 with the sole difference that the public key is appended to the message which is actually being signed. Algorithms Share-Verify and Combine are identical to those of Section 3. We thus focus on the signature aggregation and verification algorithms. For convenience, we define the standard verification algorithm as a special case of the Aggregate-Verify algorithm when $\ell = 1$.

Aggregate$(\{(PK_j, \sigma^{(j)}, M_j)\}_{j=1}^{\ell})$: Given a set of triples $(PK_j, \sigma^{(j)}, M_j)$, parse each public key PK_j as $PK_j = (\mathsf{params}, (\hat{g_1}^{(j)}, \hat{g_2}^{(j)}), Z^{(j)}, R^{(j)})$ and return $\perp$ if it does not parse properly. Then, parse each signature $\sigma^{(j)}$ as $(z^{(j)}, r^{(j)})$ and return $\perp$ if there exists $j \in \{1, \ldots, \ell\}$ such that Verify$(PK_j, M_j, \sigma^{(j)}) = 0$. Otherwise, compute and return $(z, r) = (\prod_{j=1}^{\ell} z^{(j)}, \ \prod_{j=1}^{\ell} r^{(j)})$.

Aggregate-Verify$(\{(PK_j, M_j)\}_{j=1}^{\ell}, \sigma)$: Given a candidate aggregate signature $\sigma = (z, r) \in \mathbb{G}^2$, compute $(H_1^{(j)}, H_2^{(j)}) = H(PK_j \| M_j) \in \mathbb{G} \times \mathbb{G}$ for each $j \in \{1, \ldots, \ell\}$ and return 1 if and only if the following equality holds

$$e(z, \hat{g}_z) \cdot e(r, \hat{g}_r) \cdot \prod_{j=1}^{\ell} \prod_{k=1}^{2} e(H_j^{(j)}, \hat{g}_k^{(j)}) = 1_{\mathbb{G}_T} \ .$$

and, for each $j \in \{1, \ldots, \ell\}$, $PK_j = (\mathsf{params}, (\hat{g_1}^{(j)}, \hat{g_2}^{(j)}), Z^{(j)}, R^{(j)})$ satisfies the sanity check below

$$e(Z^{(j)}, \hat{g}_z) \cdot e(R^{(j)}, \hat{g}_r) \cdot e(g, \hat{g}_1^{(j)}) \cdot e(h, \hat{g}_2^{(j)}) = 1_{\mathbb{G}_T} \ .$$

Like Bellare *et al.* [7], we consider a security model where aggregate signatures may involve multiple messages from the same signer. The proof of Theorem 6 combines the proof of Theorem 1 and ideas of [7].

However, while Bellare *et al.* [7] prove the security of their unrestricted BGLS aggregate signatures via a reduction from the security of BLS signatures [14], we need a direct proof under the SXDH assumption. The reason is that, in the proof of [7, Lemma 3.3], the reduction interacts with a BLS challenger on input of a public key PK^{BLS} while emulating the aggregate adversary's challenger. Whenever the latter makes a random oracle query $H(PK\|M)$, the reduction forwards the query to its BLS challenger when $PK = PK^{\mathrm{BLS}}$ and handles the query itself when $PK \neq PK^{\mathrm{BLS}}$. In the threshold setting, this approach is problematic because random oracle queries may occur during the distributed key generation protocol, before the challenge public key $PK^\star$ is defined. Since $PK^\star$ is not necessarily uniform due to the use of Pedersen's protocol, we cannot trivially rule out hash queries of the form $H(PK^\star, M)$ during the key generation phase. While Gennaro *et al.* [42] properly addressed a similar problem in the static corruption setting, we found it simpler to give a direct proof under the underlying number theoretic assumptions here.

Theorem 6. *The above threshold aggregate signature is secure under adaptive corruptions in the random oracle under the SXDH assumption in* $(\mathbb{G}, \hat{\mathbb{G}}_T)$*. Concretely, any PPT adversary* $\mathcal{A}$ *implies DDH distinguishers* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *with comparable running times in the groups* $\mathbb{G}$ *and* $\hat{\mathbb{G}}$*, respectively.*

Proof. In order to prove the result, we consider a sequence of games which is similar to that in the proof of Theorem 1. For each $i \in \{0, 1\}$, we call S_i the event that the adversary wins Game i.

Game 0: This game the actual attack game during which all random oracle queries are answered by returning uniformly random group elements in $\mathbb{G}^2$. We assume that all hash queries are distinct. At the beginning, the challenger $\mathcal{B}$ assumes the role of all uncorrupted players when executing the Dist-Keygen protocol in interaction with the adversary $\mathcal{A}$. When $\mathcal{A}$ chooses to corrupt a player P_i, the challenger sets $C = C \cup \{i\}$, $\mathcal{G} = \mathcal{G} \backslash \{i\}$ and reveals the internal state of P_i. From this point forward, $\mathcal{A}$ has full control over P_i and may cause him to deviate from the protocol.

The distributed key generation phase thus ends with the generation of a challenge public key $PK^\star = (\{\hat{g}_k{}^\star, \hat{h}_k{}^\star\}_{k=1}^2, Z^\star, R^\star)$. At this point, the aggregate adversary $\mathcal{A}$ has also obtained the vector of verification keys $\mathbf{VK} = (VK_1, \ldots, VK_n)$, as well as the private key shares SK_i of all corrupted players $i \in C$ so far and their internal state, including their polynomials $A_{ik}[X]$ and $B_{ik}[X]$. In the next phase, partial signature queries (i, M) are answered by faithfully computing the pair $(z_i, r_i) = (\prod_{k=1}^2 H_k^{-A_k(i)}, \prod_{k=1}^2 H_k^{-B_k(i)})$.

When $\mathcal{A}$ terminates, it outputs a purported aggregate signature $\sigma^\star$ together with a list of pairs $\{(PK_j, M_j^\star)\}_{j=1}^\ell$. We assume that the adversary queries all hash values $\{H(PK_j, M_j^\star)\}_{j=1}^\ell$ before outputting its forgery. We denote by S_0 the event that $\sigma^\star = (z^\star, r^\star)$ is a valid forgery. Recall that S_0 implies that $\{(PK_j, M_j^\star)\}_{j=1}^\ell$ contains at least one pair $(PK_{j^\star}, M_{j^\star}^\star)$ such that $PK_{j^\star} = PK^\star$ and $|C \cup \mathcal{S}| \leq t$, where $\mathcal{S} \subset \{1, \ldots, n\}$ denotes the subset of players that have generated a partial signature on $M_{j^\star}^\star$. If $\{(PK_j, M_j^\star)\}_{j=1}^\ell$ contains several pairs satisfying the latter condition, we define $(PK_{j^\star}, M_{j^\star}^\star)$ to be the first one in lexicographical order.

At the end of the game, we define the set $\mathcal{L}$ as the union of $\{(PK_j, M_j^\star)\}_{j=1}^\ell \backslash \{(PK^\star, M_{j^\star}^\star)\}$ and the set $\{(PK^\star, M)\}_{j=1}^{q_s}$ of pairs $(PK^\star, M)$ such that $M \neq M_j^\star$ and $\mathcal{A}$ obtained a partial signature on M. Clearly, the size of $\mathcal{L}$ is at most $|\mathcal{L}| \leq q_s + n_{max} - 1$.

Game 1: This game is like Game 0 with one difference. For each hash query $\mathsf{M} = (PK\|M)$, the challenger $\mathcal{B}$ flips a coin $\vartheta_{\mathsf{M}} \in \{0, 1\}$ that takes the value 1 with probability $1/(q_s + n_{max})$ and the value 0 with probability $1 - 1/(q_s + n_{max})$. At the end of the game, $\mathcal{B}$ considers the event E that at least one of the following conditions holds:

- For the message $\mathsf{M}^\star = (PK^\star, M_{j^\star}^\star)$, the coin $\vartheta_{\mathsf{M}^\star} \in \{0, 1\}$ associated with the random oracle query $H(PK^\star, M_{j^\star}^\star)$ was $\vartheta_{\mathsf{M}^\star} = 0$.
- The list $\mathcal{L}$ contains an entry (PK, M) for which $\vartheta_{(PK\|M)} = 1$.

Note that $\mathcal{B}$ can recognize event E at the end of the game. If it occurs, $\mathcal{B}$ halts and reports failure. Coron's analysis [20] shows that $\Pr[\neg E] = 1/e(q_s + n_{max})$, where e is the base for the natural logarithm, so that we have $\Pr[S_1] = \Pr[S_0] \cdot \Pr[\neg E] = \Pr[S_0]/e(q_s + n_{max})$.

Game 2: We modify the input-output behavior of the random oracle and let the treatment of each hash query $H(PK\|M)$ depend on the random coin $\vartheta_{\mathsf{M}} \in \{0, 1\}$.

- If $\vartheta_{\mathsf{M}} = 0$, $\mathcal{B}$ chooses a random $\alpha_M \xleftarrow{R} \mathbb{Z}_p$ and defines $H(PK\|M) = (g^{\alpha_{\mathsf{M}}}, h^{\alpha_{\mathsf{M}}})$. Note that the resulting hash value $H(PK\|M) \in \mathbb{G}^2$ now lives in the linear subspace spanned by $(g, h) \in \mathbb{G}^2$.
- If $\vartheta_{\mathsf{M}} = 1$, $\mathcal{B}$ chooses a uniformly random pair $(g_{\mathsf{M}}, h_{\mathsf{M}}) \in \mathbb{G}^2$ and sets $H(PK\|M) = (g_{\mathsf{M}}, h_{\mathsf{M}})$.

The same argument as in the proof of Lemma 1 shows that Game 2 and Game 1 are computationally indistinguishable if the DDH assumption holds in the group $\mathbb{G}$. It follows that $|\Pr[S_2] - \Pr[S_1]| \leq \mathbf{Adv}^{\mathrm{DDH}_1}(\mathcal{B})$.

In Game 2, we can prove that $\Pr[S_2] \leq \mathbf{Adv}(\mathcal{B})^{\mathrm{DP}}(\lambda) + 1/p$ as $\mathcal{B}$ can be turned into an algorithm solving the DP problem, as in the proof of Theorem 1. We actually show that $\mathcal{B}$ can use the fake aggregate signature $(z^\star, r^\star)$ – which involves a message $M^\star_{j^\star}$ on behalf of $PK^\star$ although the sets C and $\mathcal{S}$ are such that $|C \cup \mathcal{S}| \leq t$ – produced by the adversary to extract a valid ordinary signature $(z^\diamond, r^\diamond)$ on $M_{j^\star}$ with respect to the challenger public key $PK^\star$. In turn, this forgery $(M^\star_{j^\star}, (z^\diamond, r^\diamond))$ can be used to break the DP assumption in the same way as in the proof of Theorem 1.

To extract, an ordinary signature $(z^\diamond, r^\diamond)$ from $(z^\star, r^\star)$, $\mathcal{B}$ proceeds in the same way as in the proof of [7, Lemma 3.3]. If $\neg E$ occurs, if we define the subset

$$\mathcal{W} := \{(PK_j, M^\star_j)\}_{j=1}^{\ell} \backslash \{(PK^\star, M^\star_{j^\star})\} \subset \mathcal{L},$$

then all pairs $(PK, M) \in \mathcal{W}$ have been assigned the bit $\vartheta_{(PK\|M)} = 0$, which means that $H(PK\|M)$ was defined as $(g^{\alpha_{\mathsf{M}}}, h^{\alpha_{\mathsf{M}}})$ for some random $\alpha_{\mathsf{M}} \in_R \mathbb{Z}_p$ chosen by $\mathcal{B}$. It implies that, as long as $PK = (\{\hat{g}_k, \hat{h}_k\}_{k=1}^{2}, Z, R)$ is a valid public key, the pair $(z_{\mathsf{M}}, r_{\mathsf{M}}) = (Z^{\alpha_{\mathsf{M}}}, R^{\alpha_{\mathsf{M}}})$ is a valid signature on M w.r.t. PK since (Z, R) is a homomorphic signature on (g, h) (this follows from the linearly homomorphic property of the scheme in Section 2.3). For all pairs $\mathsf{M} = (PK, M) \in \mathcal{W}$, $\mathcal{B}$ can thus compute a valid signature $(z_{\mathsf{M}}, r_{\mathsf{M}})$ and divide it from the aggregate forgery $(z^\star, r^\star)$. If $\tau \in \{1, \ldots, \ell\}$ denotes the number of occurrences of $(PK^\star, M^\star_{j^\star})$ in $\{(PK_j, M^\star_j)\}_{j=1}^{\ell}$, $\mathcal{B}$ can thus compute

$$(z^\diamond, r^\diamond) = \Big((z^\star \cdot \prod_{\mathsf{M} \in \mathcal{W}} z_{\mathsf{M}}^{-1})^{1/\tau}, (r^\star \cdot \prod_{\mathsf{M} \in \mathcal{W}} r_{\mathsf{M}}^{-1})^{1/\tau}\Big),$$

which is a valid ordinary signature on $M^\star_{j^\star}$ by construction.

We also know that, conditionally on event $\neg E$, for all messages $M \neq M^\star_{j^\star}$, the queried hash value $H(PK^\star\|M)$ falls in the subspace spanned by (g, h). This guarantees that $\mathcal{B}$ has not leaked too much information about the private key shares of honest players, which allows applying the same arguments as in the proof of Theorem 1 when it comes to argue that the DP problem can be solved with non-negligible probability in Game 2.

When putting the above altogether, the adversary's advantage can be bounded by

$$\mathbf{Adv}(\mathcal{A}) \leq e \cdot (q_s + n_{max}) \cdot \Big(\mathbf{Adv}^{\mathrm{DDH}_1}(\mathcal{B}_1) + \mathbf{Adv}^{\mathrm{DDH}_2}(\mathcal{B}_2) + \frac{1}{p}\Big),$$

where q_s is the number of signing queries, e is the base for the natural logarithm and n_{max} denotes the maximal number of message-public-key pairs in an aggregate. □