Fully Distributed Non-Interactive Adaptively-Secure Threshold Signature Scheme with Short Shares: Efficiency Considerations and Implementation*

Benoît Libert[1,2], Marc Joye[3], Moti Yung[4], and Fabrice Mouhartem[2]

[1] CNRS, Laboratoire d'Informatique du Parallélisme, France
[2] Ecole Normale Supérieure de Lyon, France
benoit.libert@ens-lyon.fr, fabrice.mouhartem@ens-lyon.fr
[3] OneSpan, Belgium
marc.joye@onespan.com
[4] Google Inc. and Columbia University, USA
moti@cs.columbia.edu

Abstract. Threshold cryptography enhances the availability and the security of cryptographic schemes 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 and unforgeable against up to t corrupted servers. Originally, most practical threshold signatures have a number of limitations: 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). We present a practical *fully distributed* non-interactive scheme—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 scheme is 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 scheme is erasure-free. Of particular 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 set up although it is well-known not to ensure uniformly distributed public keys. An advantage of this is that this protocol only takes one round 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.

1 Introduction

Threshold cryptography [31,32,17,30] 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. Threshold signatures enhance the security of highly sensitive private keys, like those of certification authorities (*e.g.*, [20]). They can also serve as tools for distributed storage systems [51,68]. RSA and Elgamal-type constructions have been at the core of many threshold protocols the last two decades (see, *e.g.*, [30,41,42,47]). A *fully distributed* public-key system is one where the public (and the distributed private) key are jointly generated by the same servers which end up holding the private key's shares (*e.g.*, via a threshold secret sharing [70]). Efficient distributed key generation (DKG) protocols were put forth for both RSA [14,36,35,28] and discrete-logarithm-based systems [65,43,37,18,45].

* This paper presents results initially published at PODC 2014.

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 [44,41], 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 [71] under the RSA assumption and by Katz and Yung [52] assuming the hardness of factoring. Boldyreva [12] showed a threshold version of Boneh-Lynn-Shacham signatures [16], which provided an alternative non-interactive scheme with robustness and short signatures. The latter construction [12] was subsequently generalized by Wee [72]. 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.*, [27]) stronger. The first adaptively secure threshold signatures were independently described in 1999 by Canetti *et al.* [18] and by Frankel *et al.* [37,38]. 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 an 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 to eliminate the need for servers to reliably erase intermediate computation results. However, their adaptively secure version of the Canetti-Goldwasser threshold cryptosystem [19] requires a substantial amount of interaction at each private key operation. The same holds for the adaptively secure threshold signatures of Lysyanskaya and Peikert [60] 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 [67] and proved them adaptively secure using the SIP technique and ideas from [37,38]. Similar techniques were used in [73] to construct adaptively secure threshold Waters signatures [74]. 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 [37,38] 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 [67], 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.

Libert and Yung [58,59] gave several constructions of adaptively secure threshold encryption schemes with chosen-ciphertext security. They also suggested an adaptively secure and non-interactive threshold variant [58] of a signature scheme due to Lewko and Waters [53]. The use of bilinear maps in composite order groups makes the scheme of [58] expensive when it comes to verifying signatures: as discussed by Freeman [40], 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 [54] can be used to adapt the construction of [58] 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 [21]) allows reducing the signature size to 4 group elements. Unfortunately, the techniques of [54,21] 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. Finally, the solutions of [58] require reliable erasures due to the use of the dual system encryption technique [75,53] and the existence of several distributions of partial signatures.

Our Contributions. We present a result published in [55,56], which considers the design of a practical threshold signature scheme which is as efficient as the centralized schemes obtained from [58,21] and features the following properties:

- It is fully distributed in that the public key is jointly generated by all players. No trusted dealer is required in the key generation phase.
- It is non-interactive in the sense that no communication is required among servers beyond the setup phase. Each server can locally compute its contribution to the signature generation process without talking to other servers and sends only one message to the combiner.
- From a security point of view, it is robust against malicious adversaries and remains secure in the adaptive corruption setting, where the adversary can decide whom to corrupt based on its entire view so far.
- It does not rely on the hard-to-control use of reliable erasures. Whenever the adversary corrupts a player, we assume that it learns the entire history of that player.
- Efficiency-wise, it retains private key shares of size $O(1)$, no matter how many players are involved in the protocol. In particular, servers do not have to share a backup share of other servers' shares.

At the same time, the distributed key generation phase should be as communication-efficient as possible. Our goal is to have a single communication round when the players follow the protocol. 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). Structure-preserving signatures (SPS) [2,3] are signature schemes where messages and public keys live in an abelian group over which a bilinear map is efficiently computable. Libert *et al.* [57] 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 [57]. In the (programmable) random oracle model, we prove in [55,56] 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 [57] 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 [65]. The latter basically consists in having all players verifiably share a random value using Feldman's verifiable secret sharing (VSS) [34] 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 [43] *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 observed in [44,45,23]. 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 specification.

In the journal version of the paper [56], we describe a variant supporting signature aggregation: as suggested by Boneh *et al.* [15], 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 signing 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.

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 inputs a message M, the public key PK, the verification key $\mathbf{VK}$ and a pair (i, σ_i) consisting of an index i and signature share σ_i. It outputs 1 or 0 depending on whether σ_i is deemed as a valid signature share or not.

$\mathsf{Combine}(PK, \mathbf{VK}, M, \{(i, \sigma_i)\}_{i \in S})$ 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.

$\mathsf{Verify}(PK, M, \sigma)$ 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.*, [43,44,45], 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 PPT adversary $\mathcal{A}$ has non-negligible advantage in the game hereunder. At any time, we denote by $\mathcal{C} \subset \{1, \ldots, n\}$ and $\mathcal{G} := \{1, \ldots, n\} \setminus \mathcal{C}$ the dynamically evolving subsets of corrupted and honest players, respectively. Initially, we set $\mathcal{C} = \emptyset$.*

1. *The game begins with an execution of* $\mathsf{Dist\text{-}KeyGen}(\mathsf{params}, \lambda, t, N)$ *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 $\mathcal{G} = \mathcal{G} \setminus \{i\}$, $\mathcal{C} = \mathcal{C} \cup \{i\}$ 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 $\{SK_i\}_{i \in \mathcal{C}}$ 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 $\mathcal{C} = \mathcal{C} \cup \{i\}$.*
   - *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 $\sigma_i \leftarrow \mathsf{Share\text{-}Sign}(SK_i, M)$ and returning σ_i to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a message $M^\star$ and a signature $\sigma^\star$. We define $\mathcal{V} = \mathcal{C} \cup \mathcal{S}$, 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)* $\mathsf{Verify}(PK, M^\star, \sigma^\star) = 1$.

$\mathcal{A}$'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 $a, b, c \xleftarrow{R} \mathbb{Z}_p$. The Decision Diffie-Hellman assumption is the intractability of DDH for any PPT distinguisher.*

We use bilinear maps $e\colon \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ 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.*, [69]). 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 non-trivial $(z, r) \in \mathbb{G}^2 \setminus \{(1_\mathbb{G}, 1_\mathbb{G})\}$ 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 $e(z, \hat{g}_z{}^{\theta_1}) \cdot e(r, \hat{g}_r{}^{\theta_2}) = 1_{\mathbb{G}_T}$.

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 [57], Libert *et al.* described structure-preserving signatures with linearly homomorphic properties. Given signatures on several vectors $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$ of group elements, anyone can publicly derive a signature on any linear combination of $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$. They suggested the following scheme, which is a one-time LHSPS (namely, it only allows signing one linear subspace) based on the DP assumption.

$\mathsf{Keygen}(\lambda, N)$ 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. Then, conduct the following steps.

1. Choose $\hat{g}_z, \hat{g}_r \xleftarrow{R} \hat{\mathbb{G}}$.
2. For $k = 1$ to N, pick $\chi_k, \gamma_k \xleftarrow{R} \mathbb{Z}_p$ 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 $\mathsf{pk} = (\hat{g}_z, \hat{g}_r, \{\hat{g}_k\}_{k=1}^N)$.

$\mathsf{Sign}(\mathsf{sk}, (M_1, \ldots, M_N))$ 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 $z = \prod_{k=1}^N M_k^{-\chi_k}$, $r = \prod_{k=1}^N M_k^{-\gamma_k}$.

$\mathsf{Sign\text{-}Derive}(\mathsf{pk}, \{(\omega_i, \sigma^{(i)})\}_{i=1}^\ell)$ Given pk and ℓ tuples $(\omega_i, \sigma^{(i)})$, parse $\sigma^{(i)}$ as $\sigma^{(i)} = (z_i, r_i) \in \mathbb{G}^2$ for $i = 1$ to ℓ. Then, compute and return $\sigma = (z, r)$, where $z = \prod_{i=1}^\ell z_i^{\omega_i}$ and $r = \prod_{i=1}^\ell r_i^{\omega_i}$.

$\mathsf{Verify}(\mathsf{pk}, \sigma, (M_1, \ldots, M_N))$ Given $\sigma = (z, r) \in \mathbb{G}^2$ 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 the full version of this paper [56], 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 $\boldsymbol{H} \in \mathbb{G}^n$ and generating a one-time homomorphic signature on $\boldsymbol{H}$. 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 [12], which builds on the short signatures of Boneh, Lynn, and Shacham [16].

The DKG phase uses Pedersen's protocol [65] (or, more precisely, a variant with two generators). Each player verifiably shares a random secret using Pedersen's verifiable secret sharing [66]—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[5] and thus amenable for non-interactively

[5] Namely, the private key space forms an additive group such that, for any message M, given any two signatures $\sigma_1 \leftarrow \mathsf{Sign}(\mathsf{sk}_1, M)$ and $\sigma_2 \leftarrow \mathsf{Sign}(\mathsf{sk}_2, M)$, anyone can compute a valid signature on M for the private key $sk_1 + sk_2$.

distributing the signing process. Second, in the security proof of [57], 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 with generators $\hat{g}_z, \hat{g}_r \in_R \hat{\mathbb{G}}$ and a hash function $H\colon \{0,1\}^* \to \mathbb{G}^2$ 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.

Dist-KeyGen(params, λ, t, n) Given 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 random 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 against who was issued at least t complaints against it is immediately disqualified. Each player P_i who received a complaint from another player P_j responds by broadcasting 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 $\mathcal{Q} \subset \{1, \ldots, n\}$ 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 $\hat{g}_k = \prod_{i \in \mathcal{Q}} \hat{W}_{ik0} = \hat{g}_z^{\sum_{i \in \mathcal{Q}} a_{ik0}} \hat{g}_r^{\sum_{i \in \mathcal{Q}} b_{ik0}}$. Each P_i locally defines his private key share
   $$SK_i = \{(A_k(i), B_k(i))\}_{k=1}^2 = \left\{ \left(\textstyle\sum_{j \in \mathcal{Q}} A_{jk}(i), \sum_{j \in \mathcal{Q}} B_{jk}(i) \right) \right\}_{k=1}^2$$
   and anyone can publicly compute his verification key $VK_i = (\hat{V}_{1,i}, \hat{V}_{2,i})$ as
   $$VK_i = (\hat{g}_z^{A_1(i)} \hat{g}_r^{B_1(i)}, \hat{g}_z^{A_2(i)} \hat{g}_r^{B_2(i)}) = \left(\textstyle\prod_{j \in \mathcal{Q}} \prod_{\ell=0}^{t} \hat{W}_{j1\ell}^{\,i^\ell},\ \prod_{j \in \mathcal{Q}} \prod_{\ell=0}^{t} \hat{W}_{j2\ell}^{\,i^\ell} \right)\ .$$
   For any disqualified player $i \in \{1, \ldots, n\} \setminus \mathcal{Q}$, the i-th private key share is implicitly set as $SK_i = \{(0,0)\}_{k=1}^2$ and the corresponding verification key is $VK_i = (1_{\hat{\mathbb{G}}}, 1_{\hat{\mathbb{G}}})$.

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 private key shares $\{A_k(i)\}_{k=1}^2$ and $\{B_k(i)\}_{k=1}^2$ lie on t-degree polynomials $A_k[X] = \sum_{j \in \mathcal{Q}} A_{jk}[X]$ and $B_k[X] = \sum_{j \in \mathcal{Q}} B_{jk}[X]$. Each player also holds an additive share $\{(a_{ik0}, b_{ik0})\}_{k=1}^2$ of the secret key $\{(A_k(0), B_k(0)) = (\sum_{i \in \mathcal{Q}} a_{ik0}, \sum_{i \in \mathcal{Q}} b_{ik0})\}_{k=1}^2$ 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

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

Share-Verify$(PK, \mathbf{VK}, M, (i, \sigma_i))$ Given the 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 algorithm first computes $(H_1, H_2) = H(M) \in \mathbb{G}^2$. It returns 1 if $e(z_i, \hat{g}_z) \cdot e(r_i, \hat{g}_r) \cdot \prod_{k=1}^2 e(H_k, \hat{V}_{k,i}) = 1_{\mathbb{G}_T}$ 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 [7] at the 112-bit security level,[6] each signature consists of 512 bits. For the same security level, RSA-based threshold signatures like [71,4] require 2692 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 constant-size private key share $SK_i = \{(A_k(i), B_k(i))\}_{k=1}^2$—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 $\{A_{jk}[X], B_{jk}[X]\}_{j \in \mathcal{Q}, k \in \{1,2\}}$ and the private key shares $\{SK_j\}_{j \in \mathcal{Q}}$ 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 $j \in \mathcal{Q}$ 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 [22]. 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 [22] shows that, with probability $\Omega(1/q_s)$, the following conditions

[6] Given the recent attacks [50] on Barreto-Naehrig curves, and more generally curves with a smooth extension degree, the security of BN curves has been reevaluated [6]. Which explains this non-standard security level.

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 $\{a_{jk0}, b_{jk0}\}_{j\in\mathcal{Q}\setminus\mathcal{G}}$, $k \in \{1,2\}$ 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.* (The proof is given in Appendix B.)

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 the full version of this work [56], 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 [64,47,39] 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 [48, 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.

3.4 Implementation Results

Table 1. Implementation results for a set of 51 users with absolute majority to allow recombinations.

Algorithm	Dist-KeyGen	Share-Sign	Combine	Verify
Timings (ms)	202763	112	493	13

A proof-of-concept implementation is available at URL https://gitlab.inria.fr/fmouhart/threshold-signature. It is written in C++ using the Relic library [5] to manipulate pairing-friendly groups but is not fully optimized (especially the Dist-KeyGen protocol). The timings listed in Table 1

were obtained on a *Intel Core*^{TM} *i7-5600U* running at 2.60GHz on a single core. The Combine algorithm includes the Share-Verify of the signature for each user who has not been disqualified. This implementation gives a rough estimate (order of magnitude) of the performance this threshold signature scheme can achieve. The Dist-KeyGen and the Share-Sign algorithms are evaluated for the overall process done in a sequential manner, not for an individual user.

References

1. M. Abe, S. Fehr. Adaptively secure Feldman VSS and applications to universally-composable threshold cryptography. In *Crypto 2004*, *LNCS* 3152, pp. 317–334, 2004. doi:10.1007/978-3-540-28628-8_20
2. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, M. Ohkubo. Structure-preserving signatures and commitments to group elements. In *Crypto 2010*, *LNCS* 6223, pp. 209–236, 2010. doi:10.1007/978-3-642-14623-7_12
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. http://eprint.iacr.org/2010/133
4. J. Almansa, I. Damgård, J.-B. Nielsen. Simplified threshold RSA with adaptive and proactive security. In *Eurocrypt 2006*, *LNCS* 4004, pp. 593–611, 2006. doi:10.1007/11761679_35
5. D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic.
6. R. Barbulescu and S. Duquesne. Updating key size estimations for pairings. *Journal of Cryptology*, to appear. doi:10.1007/s00145-018-9280-5
7. P. Barreto, M. Naehrig. Pairing-friendly elliptic curves of prime order. In *SAC 2005*, *LNCS* 3897, pp. 319–331, 2005. doi:10.1007/11693383_22
8. M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, H. Shacham. Randomizable proofs and delegatable anonymous credentials. In *Crypto 2009*, *LNCS* 5677, pp. 108–125, 2009. doi:10.1007/978-3-642-03356-8_7
9. M. Bellare, C. Namprempre, G. Neven. Unrestricted aggregate signatures. In *ICALP 2007*, *LNCS* 4596, pp. 411–422, 2007. doi:10.1007/978-3-540-73420-8_37
10. M. Bellare, T. Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme. In *Eurocrypt 2009*, *LNCS* 5479, pp. 407–424, 2009. doi:10.1007/978-3-642-01001-9_24
11. M. Bellare, P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS '93*, pp. 62–73, 1993. doi:10.1145/168588.168596
12. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman group signature scheme. In *PKC 2003*, *LNCS* 2567, pp. 31–46, 2003. doi:10.1007/3-540-36288-6_3
13. D. Boneh, X. Boyen, H. Shacham. Short group signatures. In *Crypto 2004*, *LNCS* 3152, pp. 41–55, 2004. doi:10.1007/978-3-540-28628-8_3
14. D. Boneh, M. Franklin. Efficient generation of shared RSA keys. In *Crypto '97*, *LNCS* 1924, pp. 425–439, 1997. doi:10.1007/BFb0052253
15. D. Boneh, C. Gentry, B. Lynn, H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Eurocrypt 2003*, *LNCS* 2656, pp. 416–432, 2003. doi:10.1007/3-540-39200-9_26
16. D. Boneh, B. Lynn, H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology* 17(4):297–319, 2004. Earlier version in *Asiacrypt'01*, *LNCS* 2248, pp. 514–532, 2001. doi:10.1007/s00145-004-0314-9
17. C. Boyd. Digital multisignatures. In *Cryptography and Coding* (H.J. Beker and F.C. Piper, Eds.), Oxford University Press, pp. 241–246, 1989.
18. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Adaptive security for threshold cryptosystems. In *Crypto '99*, *LNCS* 1666, pp. 98–115, 1999. doi:10.1007/3-540-48405-1_7
19. R. Canetti, S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *Eurocrypt '99*, *LNCS* 1592, pp. 90–106, 1999. doi:10.1007/3-540-48910-X_7
20. Distributed CA for Visa-MC SET Infrastructure announcement, 1997. http://www.geocities.ws/rayvaneng/w0597_09.htm
21. J. Chen, H.-W. Lim, S. Ling, H. Wang, H. Wee. Shorter IBE and signatures via asymmetric pairings. In *Pairing 2012*, *LNCS* 7708, pp. 122–140, 2012. doi:10.1007/978-3-642-36334-4_8
22. J.-S. Coron. On the exact security of full domain hash. In *Crypto 2000*, *LNCS* 1880, pp. 229–235, 2000. doi:10.1007/3-540-44598-6_14
23. V. Cortier, D. Galindo, S. Glondu, M. Izabachène. Distributed ElGamal à la Pedersen: Application to Helios. In *WPES 2013*, pp. 131–142, 2013. doi:10.1145/2517840.2517852
24. R. Cramer, M. Franklin, B. Schoenmakers, M. Yung. Multi-autority secret-ballot elections with linear work. In *Eurocrypt '96*, *LNCS* 1070, pp. 72–83, 1996. doi:10.1007/3-540-68339-9_7
25. R. Cramer, R. Gennaro, B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Eurocrypt '97*, *LNCS* 1233, pp. 103–118, 1997. doi:10.1007/3-540-69053-0_9
26. R. Cramer, I. Damgård, J.-B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Eurocrypt 2001*, *LNCS* 2045, pp. 280–299, 2001. doi:10.1007/3-540-44987-6_18

27. 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, 1999. doi:10.1007/3-540-48910-X_22
28. I. Damgård, G. Mikkelsen. Efficient, robust and constant-round distributed RSA key generation. In *TCC 2010*, *LNCS* 5978, pp. 183–200, 2010. doi:10.1007/978-3-642-11799-2_12
29. A. Dent. A note on game-hopping proofs. Cryptology ePrint Archive: Report 2006/260. http://eprint.iacr.org/2006/260
30. A. De Santis, Y. Desmedt, Y. Frankel, M. Yung. How to share a function securely. In *STOC '94*, pp. 522-533, 1994. doi:10.1145/195058.195405
31. Y. Desmedt. Society and group oriented cryptography: A new concept. In *Crypto '87*, *LNCS* 293, pp. 120–127, 1987. doi:10.1007/3-540-48184-2_8
32. Y. Desmedt, Y. Frankel. Threshold cryptosystems. In *Crypto'89*, *LNCS* 435, pp. 307–315, 1989. doi:10.1007/0-387-34805-0_28
33. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Crypto '84*, *LNCS* 196, pp. 10–18, 1985. doi:10.1007/3-540-39568-7_2
34. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *FOCS '87*, pp. 427–437, 1987. doi:10.1109/SFCS.1987.4
35. P.-A. Fouque, J. Stern. Fully distributed threshold RSA under standard assumptions. In *Asiacrypt 2001*, *LNCS* 2248, pp. 310–330, 2001. doi:10.1007/3-540-45682-1_19
36. Y. Frankel, P. MacKenzie, M. Yung. Robust efficient distributed RSA-key generation. In *STOC '98*, pp. 663–672, 1998. doi:10.1145/276698.276882
37. Y. Frankel, P. MacKenzie, M. Yung. Adaptively-secure distributed public-key systems. In *ESA '99*, *LNCS* 1643, pp. 4–27, 1999. doi:10.1007/3-540-48481-7_2
38. Y. Frankel, P. MacKenzie, M. Yung. Adaptively-secure optimal-resilience proactive RSA. In *Asiacrypt '99*, *LNCS* 1716, pp. 180–194, 1999. doi:10.1007/978-3-540-48000-6_15
39. Y. Frankel, P. Gemmell, P. MacKenzie, M. Yung. Optimal resilience proactive public-key cryptosystems. In *FOCS '97*, pp. 384–393, 1997. doi:10.1109/SFCS.1997.646127
40. D. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *Eurocrypt 2010*, *LNCS* 6110, pp. 44-61, 2010. doi:10.1007/978-3-642-13190-5_3
41. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Robust threshold DSS signatures. In *Eurocrypt '96*, *LNCS* 1070, pp. 354–371, 1996. doi:10.1007/3-540-68339-9_31
42. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Robust and efficient sharing of RSA functions. In *Crypto '96*, *LNCS* 1109, pp. 157–172, 1996. doi:10.1007/3-540-68697-5_13
43. 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, 1999. doi:10.1007/3-540-48910-X_21
44. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Secure applications of Pedersen's distributed key generation protocol. In *CT-RSA 2003*, *LNCS* 2612, pp. 373–390, 2003. doi:10.1007/3-540-36563-X_26
45. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology* 20(1):51–83, 2007. doi:10.1007/s00145-006-0347-3
46. R. Gennaro, S. Halevi, H. Krawczyk, T. Rabin. Threshold RSA for dynamic and ad-hoc groups. In *Eurocrypt 2008*, *LNCS* 4965, pp. 88–107, 2008. doi:10.1007/978-3-540-78967-3_6
47. A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung. Proactive public key and signature systems. In *ACM-CCS '97*, pp. 100–110, 1997. doi:10.1145/266420.266442
48. 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, 1995. doi:10.1007/3-540-44750-4_27
49. S. Jarecki, A. Lysyanskaya. Adaptively Secure Threshold cryptography: Introducing concurrency, removing erasures. In *Eurocrypt 2000*, *LNCS* 1807, pp. 221–242, 2000. doi:10.1007/3-540-45539-6_16
50. T. Kim and R. Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In *Crypto 2016*, *LNCS* 9814, pp. 543–571. Springer, 2016. doi:10.1007/978-3-662-53018-4_20
51. 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 2000*, pp. 190-201, 2000. doi:10.1145/356989.357007
52. J. Katz, M. Yung. Threshold cryptosystems based on factoring. In *Asiacrypt 2002*, *LNCS* 2501, pp. 199-205, 2002. doi:10.1007/3-540-36178-2_12
53. A. Lewko, B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC 2010*, *LNCS* 5978, pp. 455–479, 2010. doi:10.1007/978-3-642-11799-2_27
54. A. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *Eurocrypt 2012*, *LNCS* 5978, pp. 318–33, 2012. doi:10.1007/978-3-642-29011-4_20
55. B. Libert, M. Joye, M. Yung. Born and raised distributively: Fully distributed non-interactive adaptively secure threshold signatures with short shares. In *PODC 2014*, pp. 303–312, 2014. doi:10.1145/2611462.2611498
56. B. Libert, M. Joye, M. Yung. Born and raised distributively: Fully distributed non-interactive adaptively secure threshold signatures with short shares. In *Theoretical Computer Science* 645:1–24, 2016. doi:10.1016/j.tcs.2016.02.031
57. B. Libert, T. Peters, M. Joye, M. Yung. Linearly homomorphic structure-preserving signatures and their applications. In *Crypto 2013*, *LNCS* 8043, pp. 289–307, 2013. doi:10.1007/978-3-642-40084-1_17

58. B. Libert, M. Yung. Adaptively secure non-interactive threshold cryptosystems. *Theoretical Computer Science* 478:76–100, 2013. Extended abstract in *ICALP 2011*, *LNCS* 6756, pp. 588–600, 2011. doi:10.1016/j.tcs.2013.01.001
59. B. Libert, M. Yung. Non-interactive CCA2-secure threshold cryptosystems with adaptive security: New framework and constructions. In *TCC 2012*, *LNCS* 7194, pp. 75–93, 2012. doi:10.1007/978-3-642-28914-9_5
60. A. Lysyanskaya, C. Peikert. Adaptive security in the threshold setting: From cryptosystems to signature schemes. In *Asiacrypt 2001*, *LNCS* 2248, pp. 331–350, 2001. doi:10.1007/3-540-45682-1_20
61. T. Malkin, I. Teranishi, Y. Vahlis, M. Yung. Signatures resilient to continual leakage on memory and computation. In *TCC 2011*, *LNCS* 6597, pp. 89–106, 2011. doi:10.1007/978-3-642-19571-6_7
62. M. Naor, O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS '97*, pp. 458–467, 1997. doi:10.1109/SFCS.1997.646134
63. T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Crypto '92*, *LNCS* 740, pp. 31–53, 1993. doi:10.1007/3-540-48071-4_3
64. R. Ostrovksy, M. Yung. How to withstand mobile virus attacks. In *PODC '91*, pp. 51–59, 1991. doi:10.1145/112600.112605
65. T. Pedersen. A threshold cryptosystem without a trusted party. *Eurocrypt '91*, *LNCS* 547, pp. 522–526, 1991. doi:10.1007/3-540-46416-6_47
66. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. *Crypto '91*, *LNCS* 576, pp. 129–140, 1991. doi:10.1007/3-540-46766-1_9
67. T. Rabin. A simplified approach to threshold and proactive RSA. In *Crypto '98*, *LNCS* 1462, pp. 89–104, 1998. doi:10.1007/BFb0055722
68. S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, J. Kubiatowicz. Pond: The OceanStore prototype. In *Fast 2003*, USENIX Workshop on File and Storage Technologies, 2003. http://www.usenix.org/events/fast03/tech/rhea.html
69. M. Scott. Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Cryptology ePrint Archive: Report 2002/164. http://eprint.iacr.org/2002/164
70. A. Shamir. How to share a secret. *Communications of ACM* 22(11):612–613, 1979. doi:10.1145/359168.359176
71. V. Shoup. Practical threshold signatures. In *Eurocrypt 2000*, *LNCS* 1807, pp. 207–220, 2000. doi:10.1007/3-540-45539-6_15
72. H. Wee. Threshold and revocation cryptosystems via extractable hash proofs. In *Eurocrypt 2011*, *LNCS* 6632, pp. 589–609, 2011. doi:10.1007/978-3-642-20465-4_32
73. Z. Wang, H. Qian, Z. Li. Adaptively secure threshold signature scheme in the standard model. *Informatica* 20(4):591-–612, 2009. https://www.mii.lt/Informatica/htm/INFO739.htm
74. B. Waters. Efficient identity-based encryption without random oracles. In *Eurocrypt 2005*, *LNCS* 3494, 2005. doi:10.1007/11426639_7
75. B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *Crypto 2009*, *LNCS* 5677, pp. 619–636, 2009. doi:10.1007/978-3-642-03356-8_36

A Definition of Linearly Homomorphic Structure-Preserving Signatures

Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ be groups of prime order p with an efficiently computable bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$.

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 $\Sigma = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Sign\text{-}Derive}, \mathsf{Verify})$ for which the message space is $\mathcal{M} := \mathcal{T} \times \mathbb{G}^N$, 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.*

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

$\mathsf{Sign\text{-}Derive}(\mathsf{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^{\ell})$**:** *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 $\boldsymbol{M} = \prod_{i=1}^{\ell} \boldsymbol{M}_i{}^{\omega_i}$, where $\sigma^{(i)}$ is a signature on $\boldsymbol{M}_i$.*

$\mathsf{Verify}(\mathsf{pk}, \tau, \boldsymbol{M}, \sigma)$ *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* $\boldsymbol{M} = (M_1, \ldots, M_N)$*. 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 $(\tau^\star, \boldsymbol{M}^\star, \sigma^\star)$ for a new file identifier $\tau^\star$ or, if $\tau^\star$ is recycled from one or more honestly generated signatures, for a vector $\boldsymbol{M}^\star$ 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.

B Proof of Theorem 1

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 $\mathsf{Dist\text{-}KeyGen}$ protocol on behalf of all uncorrupted players. Whenever the adversary $\mathcal{A}$ decides to corrupt a player P_i, the challenger sets $\mathcal{C} = \mathcal{C} \cup \{i\}$, $\mathcal{G} = \mathcal{G} \backslash \{i\}$ 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 $\mathsf{Dist\text{-}KeyGen}$. 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 $(z_i, r_i) = (\prod_{k=1}^2 H_k^{-A_k(i)}, \prod_{k=1}^2 H_k^{-B_k(i)})$. 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 $A_k[X] = \sum_{i \in \mathcal{Q}} A_{ik}[X]$ and $B_k[X] = \sum_{i \in \mathcal{Q}} B_{ik}[X]$ as well as $(a_{k0}, b_{k0}) = (\sum_{i \in \mathcal{Q}} a_{ik0}, \sum_{i \in \mathcal{Q}} b_{ik0})$ for each $k \in \{1, 2\}$. We remark that, at the end of the $\mathsf{Dist\text{-}KeyGen}$ 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 $j \in \mathcal{Q}$. Indeed, for each $j \in \mathcal{Q} \cap \mathcal{C}$ 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 $j \in \mathcal{Q}$, the challenger honestly chose their sharing polynomials at step 1.a of $\mathsf{Dist\text{-}KeyGen}$.

Game 1: This game is identical to Game 0 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. The same analysis as that of Coron [22] shows that $\Pr[\neg E] = 1/(e(q_s + 1))$, 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 [29] and we thus have $\Pr[S_1] = \Pr[S_0] \cdot \Pr[\neg E] = \Pr[S_0]/(e(q_s + 1))$.

Game 2: We modify the distribution of random oracle outputs. Specifically, the challenger $\mathcal{B}$ chooses generators $g, h \xleftarrow{R} \mathbb{G}$ 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 $\alpha_M \xleftarrow{R} \mathbb{Z}_p$, 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 sets $H(M) = (g_M, h_M)$.

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 $\Pr[S_2] \leq \mathbf{Adv}(\mathcal{B})^{\mathrm{DP}}(\lambda) + 1/p$ as $\mathcal{B}$ implies a DP-solving algorithm.

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 $\mathcal{C}$. Note that, for these updated sets $\mathcal{G}$ and $\mathcal{C}$, it still knows the polynomials $\{(A_{jk}[X], B_{jk}[X])\}_{k=1}^2$ for all $j \in \mathcal{C}$. Let us define the aggregated additive shares

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

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, consider what an unbounded $\mathcal{A}$ learns. Corruption queries reveal $\{A_{j2}(i)\}_{j\in\mathcal{G}, i\in\mathcal{C}}$, which is insufficient to infer anything about $a_{2,\mathcal{G}} = \sum_{j\in\mathcal{G}} A_{j2}(0)$ since $|\mathcal{C}| \leq t$. For each $M \neq M^\star$, signing queries are answered by returning

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

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 $j \in \mathcal{Q} \cap \mathcal{C}$, it can obtain

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

However, these partial signatures (z_i, r_i) on $M \neq M^\star$ only provide $\mathcal{A}$ with redundant information about $\left(\sum_{j\in\mathcal{G}} A_{j1}(i), \sum_{j\in\mathcal{G}} A_{j2}(i), \sum_{j\in\mathcal{G}} B_{j1}(i), \sum_{j\in\mathcal{G}} B_{j2}(i)\right)$. 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

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

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

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

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

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

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

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

Lemma 1. *Under the DDH assumption in $\mathbb{G}$, Game 2 is 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, $\mathbf{SK}$ and $\mathbf{VK}$ 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 $z_M = x \cdot \alpha_M + \beta_M$. 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 $g_M, h_M \xleftarrow{R} \mathbb{G}^2$ 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. □