Adaptively Secure BLS Threshold Signatures from DDH and co-CDH

Sourav Das and Ling Ren

University of Illinois at Urbana Champaign
{souravd2, renling}@illinois.edu

Abstract. Threshold signatures are one of the most important cryptographic primitives in distributed systems. A popular choice of threshold signature scheme is the BLS threshold signature introduced by Boldyreva (PKC'03). Some attractive properties of Boldyreva's threshold signature are that the signatures are unique and short, the signing process is non-interactive, and the verification process is identical to that of non-threshold BLS. These properties have resulted in its practical adoption in several decentralized systems. However, despite its popularity and wide adoption, up until recently, the Boldyreva scheme has been proven secure only against a static adversary. Very recently, Bacho and Loss (CCS'22) presented the first proof of adaptive security for the Boldyreva scheme, but they have to rely on strong and non-standard assumptions such as the hardness of one-more discrete log (OMDL) and the Algebraic Group Model (AGM). In this paper, we present the first adaptively secure threshold BLS signature scheme that relies on the hardness of DDH and co-CDH in asymmetric pairing groups in the Random Oracle Model (ROM). Our signature scheme also has non-interactive signing, compatibility with non-threshold BLS verification, and practical efficiency like Boldyreva's scheme. These properties make our protocol a suitable candidate for practical adoption with the added benefit of provable adaptive security.

1 Introduction

Threshold signatures schemes [Des88, DF89, GJKR07] protect the signing key by sharing it among a group of signers so that an adversary must corrupt a threshold number of signers to be able to forge signatures. The increasing demand for decentralized applications has resulted in large-scale adoptions of threshold signature schemes. Many state-of-the-art Byzantine fault tolerant protocols utilize threshold signatures to lower communication costs [MXC+16, YMR+19, AMS19, LLTW20, GKKS+22, GHM+17]. Efforts to standardize threshold cryptosystems are already underway [BP23].

A popular choice of threshold signature is the BLS signature, introduced by Boldyreva [Bol03] building on the work of Boneh–Lynn–Shacham [BLS01]. Boldyreva's BLS threshold signature scheme is popular because its verification is identical to standard non-threshold BLS signature, its signing process is non-interactive, the signatures are unique and small (a single elliptic curve group element), and the scheme is very efficient in terms of both computation and communication. These properties have resulted in practical adoptions of Boldyreva's BLS threshold signature for applications in the decentralized setting [dra23, ic23, ska23, arp23].

Static vs. Adaptive Security. However, despite its popularity and wide adoption, until recently, Boldyreva's scheme has been proven secure only against a static adversary. A static adversary must decide the set of signers to corrupt at the start of the protocol. In contrast, an adaptive adversary can decide which signers to corrupt during the execution of the protocol based on its view of the execution. Clearly, an adaptive adversary is a safer and more realistic assumption for the decentralized setting.

Designing an adaptively secure threshold signature scheme (BLS or otherwise) is challenging, let alone keeping it compatible with a non-threshold signature scheme. The generic approach to transforming a statically secure protocol into an adaptive one by guessing the set of parties an adaptive adversary may corrupt incurs an unacceptable exponential (in the number of parties) security loss. Existing adaptively secure threshold signature schemes in the literature have to make major sacrifices such as relying on parties to erase their internal states [CGJ+99, LY13], inefficient cryptographic primitives like non-committing encryptions [JL00, LP01], or strong and non-standard assumptions such as one more discrete logarithm (OMDL)

in the algebraic group model (AGM) [BL22, CKM23]. To make matters worse, for Boldyreva's variant of BLS signatures in particular, the recent work of Bacho-Loss [BL22] proves that a strong assumption such as OMDL is necessary.

Our results. We present an adaptively secure BLS threshold signature scheme. Our scheme retains the attractive properties of Boldyreva's scheme: signing is non-interactive, verification is identical to non-threshold BLS, and the scheme is simple and efficient.

The adaptive security proof of our signature scheme assumes the hardness of the decisional Diffie-Hellman (DDH) problem in a source group and the hardness of the co-computational Diffie-Hellman (co-CDH) problem in asymmetric pairing groups in the random oracle model (ROM). To put things into perspective, note that the standard non-threshold BLS signature assumes hardness of computational Diffie-Hellman (CDH) in pairing groups* in the ROM. Thus, our scheme only relies on DDH besides what standard non-threshold BLS signature already relies on. Moreover, if one is content with proving our scheme statically secure, we only need CDH in the ROM, as in the standard BLS signature.

In terms of efficiency, our scheme is only slightly more expensive than the Boldyreva scheme [Bol03]. The signing key of each signer consists of three field elements compared to one in Boldyreva. The threshold public keys consist of n group elements in total, identical to Boldyreva. Here n is the total number of signers. Our per-signer signing cost and partial signature verification cost of the aggregator are also small. We implement our scheme in Golang and compare its performance with Boldyreva's scheme. Our evaluation confirms that our scheme adds very small overheads.

We also describe a distributed key generation (DKG) protocol to secret share the signing key of our scheme. Our DKG adds minimal overhead compared to existing DKG schemes.

All of the above properties combined make our scheme a suitable candidate for a drop-in replacement for BLS signature in deployment systems and a worthwhile trade-off for the added benefit of provable adaptive security at modest performance cost.

Paper organization. We discuss the related work in §2 and present a technical overview of our scheme in §3. In §4, we give the required preliminaries. We then describe our threshold signature scheme in two parts: First, in §5 we describe our threshold signature scheme assuming a trusted key generation functionality to generate the signing keys. We then analyze its security in §6. Second, in §7, we describe a DKG protocol, which signers can use to set up the signing keys for our scheme in a distributed manner. Then in §8, we prove the adaptive security of our threshold signature when combined with our DKG protocol. We discuss the implementation and evaluation details in §9, and conclude with a discussion in §10.

2 Related works

Threshold signature schemes were first introduced by Desmedt [Des88]. Since then, numerous threshold signature schemes with various properties have been proposed. Most of the natural and popular threshold signature schemes are proven secure only against a static adversary [Des88, GJKR96, GJKR07, Sho00, Bol03, CGG$^+$20, KG21, CKM21, BCK$^+$22, RRJ$^+$22, CGRS23, TZ23, Sho23, BHK$^+$24, GS24]. The difficulty in proving adaptive security usually lies in the reduction algorithm's inability to generate consistent internal states for all parties. As a result, the reduction algorithm needs to know which parties will be corrupt, making the adversary static [BCK$^+$22]. We will next review threshold signatures with adaptive security. We classify them into *interactive* and *non-interactive* schemes.

Interactive threshold signatures. In an interactive threshold signature, signers interact with each other to compute the signature on a given message. The first adaptively secure threshold signatures were independently described by Canetti et al. [CGJ$^+$99] and Frankel et al. [FMY99a, FMY99b]. They prove adaptive security of their threshold signature scheme by introducing the "single inconsistent player" (SIP) technique. In the SIP approach, there exists only one signer whose internal state cannot be consistently revealed to the

*The standard non-threshold BLS signature scheme can also work with symmetric pairing groups and hence the CDH assumption instead of co-CDH.

adversary. Since this inconsistent signer is chosen at random, it is only corrupt with probability less than $1/2$ for $n > 2t$. These schemes also rely on secure erasure.

Lysyanskaya-Peikert [LP01] and Abe and Fehr [AF04] use the SIP technique along with expensive cryptographic primitives such as threshold homomorphic encryptions and non-committing encryptions, respectively, to design adaptively secure threshold signatures without relying on erasures. Later works [ADN06, WQL09] extend the SIP technique to Rabin's threshold RSA signature [Rab98] and the Waters [Wat05] signatures. A major downside of all these works is the high signing cost. For every message, signers need to run a protocol similar to a DKG protocol. Concurrently and independently, [BLT+24] presents a three-round adaptively secure threshold signature scheme assuming the hardness of DDH.

Non-interactive threshold signatures. A non-interactive threshold signature requires each signer to send a single message to sign. Practical, robust, non-interactive threshold signatures were described by Shoup [Sho00] under the RSA assumption and by Katz and Yung [KY02] assuming the hardness of factoring. Boldyreva [Bol03] presented a non-interactive threshold BLS signature scheme. Until recently, these schemes were proven secure against static adversaries only.

Bacho and Loss [BL22] recently proved adaptive security for Boldyreva's scheme based on the One More Discrete Logarithm (OMDL) assumption in the Random Oracle Model (ROM) and Algebraic Group Model (AGM). Their method addresses the challenge of revealing internal states of corrupt nodes to the adversary by giving the reduction adversary limited access to discrete logarithm oracle. (This approach has since been extended to the interactive threshold Schnorr signature [CKM23].) Bacho-Loss [BL22] also proves that reliance on OMDL is necessary for proving Boldyreva's BLS signature adaptively secure. This implies that a new protocol is needed to prove adaptive security under more standard assumptions.

Libert et al., [LJY14] presented a pairing-based, non-interactive threshold signature scheme assuming the hardness of the double-pairing assumption. However, their signature scheme is incompatible with standard BLS signature verification and thus cannot be a drop-in replacement for BLS in deployment systems. The signature size of their scheme is also twice as large as a BLS signature. Very recently, [DCX+23, GJM+24] also present pairing-based non-interactive threshold signatures with adaptive security. However, their signatures are also incompatible and more than $5\times$ larger than BLS signatures.

3 Technical Overview

We need to introduce several new ideas to design a new BLS threshold signature scheme and prove it adaptively secure. First, we introduce a new way of embedding the co-CDH input into a simulation of our scheme. Since we want our final signature to be a standard BLS signature, and BLS signatures are deterministic, these changes are delicate. Moreover, we embed the co-CDH challenge in such a way that during simulation, it remains indistinguishable from an honest execution of the protocol. This should hold, even if we use a DKG to generate the signing keys. We address this as follows. In our security proof, the reduction adversary can simulate the DKG and the threshold signature scheme to the adversary by faithfully running the protocol on behalf of all but one honest signer, i.e., we work with the single inconsistent party (SIP) technique. Second, we use a new approach to program two random oracles in a correlated way while ensuring that it remains indistinguishable from uniformly random to a computationally bounded adversary. This step is crucial for the reduction adversary to simulate signing queries.

Boneh-Lynn-Sacham (BLS) signature scheme [BLS01]. Before we describe our techniques, we briefly recall the non-threshold BLS signature scheme. Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ be a tuple of prime order pairing groups with scalar field $\mathbb{F}$. Let $\mathcal{M}$ be the finite message space of the signature scheme. Let $g \in \mathbb{G}$ be a uniformly random generator of $\mathbb{G}$ and $\mathsf{H} : \mathcal{M} \to \hat{\mathbb{G}}$ be a hash function modeled as a random oracle. The signing key $\mathsf{sk} = s \in \mathbb{F}$ is a random field element, and $\mathsf{pk} = g^s \in \mathbb{G}$ is the corresponding public verification key. The signature σ on a message m is then $\mathsf{H}(m)^{\mathsf{sk}} \in \hat{\mathbb{G}}$. Any verifier validates a signature σ' on a message m by checking that $e(\mathsf{pk}, \mathsf{H}(m)) = e(g, \sigma')$, where $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ is the bilinear pairing operation. The BLS signature is proven secure assuming the hardness of CDH in the ROM [BLS01].

Our core ideas. We will illustrate our core ideas using a simplified threshold signature scheme, which we do not know how to prove adaptively secure. We describe our final protocol and its adaptive security proof in §5 and §6, respectively.

Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ be a tuple of prime order asymmetric pairing groups with scalar field $\mathbb{F}$. Let $g, h \in \mathbb{G}$ be two uniformly random generators of $\mathbb{G}$ and $\hat{g}$ be a generator of $\hat{\mathbb{G}}$. As in the non-threshold BLS signature scheme, let $\mathsf{sk} = s \in \mathbb{F}$ be the secret signing key and $\mathsf{pk} = g^s \in \mathbb{G}$ be the public verification key. To get an (n, t) threshold signature scheme, the secret signing key s is then shared among n signers using a degree t polynomial $s(x)$. Additionally, signers also receive a share on a uniformly random polynomial $r(x)$ with the constraint that $r(0) = 0$. Precisely, the signing key of signer i is $\mathsf{sk}_i = (s(i), r(i))$ and the public verification key of signer i is $\mathsf{pk}_i = g^{s(i)} h^{r(i)} \in \mathbb{G}$.

With this initial setup, signers sign any message $m \in \mathcal{M}$, for a finite message space $\mathcal{M}$, as follows. Let $\mathsf{H}_0, \mathsf{H}_1$ be two random oracles where $\mathsf{H}_b : \mathcal{M} \to \hat{\mathbb{G}}$ for $b \in \{0, 1\}$. The partial signature from signer i on a message m is then $\sigma_i = \mathsf{H}_0(m)^{s(i)} \mathsf{H}_1(m)^{r(i)} \in \hat{\mathbb{G}}$. Upon receiving $t + 1$ valid partial signatures from a set of signers T, the aggregator computes the threshold signature by interpolating them in the exponent, i.e., it computes the aggregated signature $\sigma = \prod_{i \in T} \sigma_i^{L_i}$ for appropriate Lagrange coefficients L_i. It is easy to see that since $r(0) = 0$, the interpolation yields a standard BLS signature $\sigma = \mathsf{H}_0(m)^s \mathsf{H}_1(m)^0 = \mathsf{H}_0(m)^s$.

An avid reader will note that the partial signatures are no longer verifiable using a pairing check. Indeed, signers in our protocol instead use a Σ-protocol to prove the correctness of their partial signatures.

Naturally, the important question is how this modified BLS threshold signature helps us prove adaptive security. (We reiterate that the goal of this section is to give intuition, and we do not know how to prove this exact scheme adaptively secure.) At a very high level, the additional parameter h, the additional polynomial $r(x)$, and the additional random oracle $\mathsf{H}_1(\cdot)$ provide the reduction adversary with extra avenues to embed the co-CDH input and extract a solution to the co-CDH input from a signature forgery. We will elaborate on this next.

Let $\mathcal{A}_{\text{co-CDH}}$ be the reduction algorithm and $\mathcal{A}$ be the adversary that breaks the unforgeability of our scheme. $\mathcal{A}_{\text{co-CDH}}$ will run our threshold signature scheme with a rigged public key $\mathsf{pk} = g^s h^r \in \mathbb{G}$ with $r \neq 0$. Concretely, we work with $r = 1$, i.e., $\mathsf{pk} = g^s h$, but any non-zero value of r will also work. $\mathcal{A}_{\text{co-CDH}}$ will carefully interact with $\mathcal{A}$ so that $\mathcal{A}$ does not realize that the public key is rigged. Then, by definition, $\mathcal{A}$ will forge a BLS signature on some message m, i.e., $e(\mathsf{pk}, \mathsf{H}_0(m)) = e(g, \sigma)$. Now given a co-CDH input tuple $(g, \hat{g}, g^a, \hat{g}^a, \hat{g}^b)$, if we set $h = g^a$ and program the random oracle in a way such that $\mathsf{H}_0(m) = \hat{g}^b$, then $\sigma = \hat{g}^{(s+a)b}$. This implies that if $s \in \mathbb{F}$ is known, then we can efficiently compute $\hat{g}^{ab}$ given σ.

Let $s(x), r(x)$ be degree t polynomials for Shamir secret sharing of $s = s(0)$ and $r(0) = 1$. We will discuss in §6 how $\mathcal{A}_{\text{co-CDH}}$ interacts with $\mathcal{A}$ while ensuring that $\mathcal{A}_{\text{co-CDH}}$ knows $s(x)$ and $r(x)$, and $r(0) = 1$. Furthermore, in Appendix 8, we will discuss how $\mathcal{A}_{\text{co-CDH}}$ achieves this even when we use a DKG key to generate the signing keys while relying on just a single inconsistent party. This implies that since $\mathcal{A}_{\text{co-CDH}}$ knows both $s(x), r(x)$, it can reveal the internal state of any party that $\mathcal{A}$ corrupts, except the inconsistent party to $\mathcal{A}$. Unless $\mathcal{A}$ corrupts the inconsistent party, $\mathcal{A}$'s view in a real protocol instance and an instance rigged by $\mathcal{A}_{\text{co-CDH}}$ are computationally indistinguishable.

The final part of our protocol is how $\mathcal{A}_{\text{co-CDH}}$ simulates the signing queries under the rigged public key. Consider a naive approach where we use the signing procedure of Boldyreva's scheme, i.e., the partial signature of signer i is $\mathsf{H}_0(m)^{s(i)}$. Then, the unique aggregated signature is $\sigma = \mathsf{H}_0(m)^s$. However, since $r(0) = 1$, unless $\mathsf{H}_0(m) = 1_{\hat{\mathbb{G}}}$, i.e., the identity of the group $\hat{\mathbb{G}}$, it will always be the case that $e(\mathsf{pk}, \mathsf{H}_0(m)) \neq e(g, \sigma)$, so $\mathcal{A}$ realizes that it is in a rigged instance. This is why we bring in an additional random oracle H_1 and have the partial signatures as $\sigma_i = \mathsf{H}_0(m)^{s(i)} \mathsf{H}_1(m)^{r(i)}$. The final aggregated signature is now $\sigma = \mathsf{H}_0(m)^s \mathsf{H}_1(m)$. If $\mathcal{A}_{\text{co-CDH}}$ programs the two random oracles in a correlated manner, the pairing check $e(\mathsf{pk}, \mathsf{H}_0(m)) = e(g, \sigma)$ will pass. Crucially, the correlated programming of the two random oracles must be undetectable to $\mathcal{A}$. In §6, we will prove this is indeed the case for our final scheme, assuming the hardness of DDH in $\hat{\mathbb{G}}$.

4 Preliminaries

Notations. For any integer a, we use $[a]$ to denote the ordered set $\{1, 2, \ldots, a\}$. For any set S, we use $s \leftarrow\$ S$ to indicate that s is sampled uniformly randomly from S. We use $|S|$ to denote the size of set S. Throughout the paper, we will use "$\leftarrow$" for probabilistic assignment and ":=" for deterministic assignment. We use λ to denote the security parameter. A machine is probabilistic polynomial time (PPT) if it is a probabilistic algorithm that runs in $\mathsf{poly}(\lambda)$ time. We also use $\mathsf{negl}(\lambda)$ to denote functions negligible in λ. We use the terms *party* (resp. *parties*) and *signer* (resp. *signers*) interchangeably.

4.1 Model

We consider a set of n signers denoted by $\{1, 2, \ldots, n\}$. We consider a PPT adversary $\mathcal{A}$ who can corrupt up to $t < n$ out of the n signers. Corrupted signers can deviate arbitrarily from the protocol specification. Note that with $t \geq n/2$, i.e., with a dishonest majority, it is impossible to achieve both unforgeability and guaranteed output delivery [KL07]. We focus on unforgeability over guaranteed output delivery for the dishonest majority case.

When the signing keys of our signature scheme are generated by a trusted setup, we assume the network is asynchronous. However, for simplicity, we will assume lock-step synchrony for our DKG protocol, i.e., parties execute the protocol in synchronized rounds, and a message sent at the start of a round arrives by the end of that round. Moreover, our DKG assumes an honest majority, i.e., $t < n/2$. Furthermore, during DKG, we let signers access a broadcast channel to send a value to all signers. We can efficiently realize such a broadcast channel by running a Byzantine broadcast protocol [LSP82, DS83, BGP92, MR21]. We note that the synchrony assumption is not necessary since asynchronous DKG protocols exist [KKMS20, DYX+22]. Similarly, we can remove the honest majority assumption using ideas from [CL24].

4.2 Shamir Secret Sharing, Bilinear Pairing, and Assumptions

Shamir secret sharing. The Shamir secret sharing [Sha79] embeds the secret s in the constant term of a polynomial $p(x) = s + a_1 x + a_2 x^2 + \cdots + a_d x^d$, where other coefficients $a_1, \cdots, a_d$ are chosen uniformly randomly from a field $\mathbb{F}$. The i-th share of the secret is $p(i)$, i.e., the polynomial evaluated at i. Given $d+1$ distinct shares, one can efficiently reconstruct the polynomial and the secret s using Lagrange interpolation. Also, s is information-theoretically hidden from an adversary that knows d or fewer shares.

Definition 1 (Bilinear Pairing). *Let $\mathbb{G}, \hat{\mathbb{G}}$ and $\mathbb{G}_T$ be three prime order cyclic groups with scalar field $\mathbb{F}$. Let $g \in \mathbb{G}$ and $\hat{g} \in \hat{\mathbb{G}}$ be generators. A pairing is an efficiently computable function $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ satisfying the following properties.*

1. *bilinear: For all $u, u' \in \mathbb{G}$ and $\hat{v}, \hat{v}' \in \hat{\mathbb{G}}$ we have*

$$e(u \cdot u', \hat{v}) = e(u, \hat{v}) \cdot e(u', \hat{v}), \quad \text{and} \quad e(u, \hat{v} \cdot \hat{v}') = e(u, \hat{v}) \cdot e(u, \hat{v}')$$

2. *non-degenerate: $g_T := e(g, \hat{g})$ is a generator of $\mathbb{G}_T$.*

We refer to $\mathbb{G}$ and $\hat{\mathbb{G}}$ as the source groups and refer to $\mathbb{G}_T$ as the target group.

We require that the decisional Diffie-Hellman (DDH) assumption holds for $\hat{\mathbb{G}}$ and the co-computational Diffie-Hellman (co-CDH) assumption holds for $(\mathbb{G}, \hat{\mathbb{G}})$.

Assumption 1 (DDH) *Let GGen be a group generation algorithm, which takes as input 1^λ and outputs the description of a prime order group $\hat{\mathbb{G}}$ with scalar field $\mathbb{F}$ of prime order p. The description also contains a*

generator $\hat{g} \in \hat{\mathbb{G}}$, and a description of the group operation. We say that the decisional Diffie-Hellman (DDH) assumption holds relative to GGen*, if for all PPT adversary $\mathcal{A}$, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := \left| \Pr\left[\mathcal{A}(\hat{\mathbb{G}},\mathbb{F},p,\hat{g},\hat{g}^a,\hat{g}^b,\hat{g}^{ab}) = 1 \;\middle|\; \begin{array}{r}(\hat{\mathbb{G}},\mathbb{F},p,\hat{g}) \leftarrow \mathsf{GGen}(1^\lambda),\\ a,b \leftarrow\!\$\ \mathbb{F}\end{array}\right] \right.$$
$$\left. - \Pr\left[\mathcal{A}(\hat{\mathbb{G}},\mathbb{F},p,\hat{g},\hat{g}^a,\hat{g}^b,\hat{g}^c) = 1 \;\middle|\; \begin{array}{r}(\hat{\mathbb{G}},\mathbb{F},p,\hat{g}) \leftarrow \mathsf{GGen}(1^\lambda),\\ a,b,c \leftarrow\!\$\ \mathbb{F}\end{array}\right] \right| = \varepsilon_{\mathrm{DDH}}$$

Assumption 2 (co-CDH) *Let GGen' be a group generation algorithm, which takes as input 1^λ and outputs the description of prime order groups $(\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T)$ with the scalar field $\mathbb{F}$ of order p, and a bilinear pairing operation $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$. The description also contains generators $(g,\hat{g}) \in (\mathbb{G},\hat{\mathbb{G}})$ and a description of the group operation. We say that the co-computational Diffie-Hellman (co-CDH) assumption holds relative to GGen', if for all PPT adversary $\mathcal{A}$, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{A},\mathsf{GGen}'}(\lambda) := \Pr\left[\mathcal{A}(\mathbb{G},\hat{\mathbb{G}},\mathbb{F},p,g,\hat{g},g^a,\hat{g}^b,\hat{g}^b) = \hat{g}^{ab} \;\middle|\; \begin{array}{r}(\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,\mathbb{F},p,g,\hat{g}) \leftarrow \mathsf{GGen}'(1^\lambda),\\ a,b \leftarrow\!\$\ \mathbb{F}\end{array}\right] = \varepsilon_{\mathrm{CDH}}$$

Remark on pairing group types. Looking ahead, the final threshold signatures in our schemes are in $\hat{\mathbb{G}}$, and hence, we require DDH to be hard in $\hat{\mathbb{G}}$. This implies that the pairing groups must be asymmetric, i.e., $\mathbb{G} \neq \hat{\mathbb{G}}$. There are two types of asymmetric pairing groups: type-II and type-III [GPS08]. A type-II pairing group supports one-directional efficient homomorphism. In our context, we can work with a type-II group $(\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T)$ with bilinear pairing operation $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ that supports an efficient homomorphism $\Phi : \mathbb{G} \to \hat{\mathbb{G}}$, but not the other way around. Note that even with such one-directional efficient homomorphism, DDH can still be hard in $\hat{\mathbb{G}}$. Thus, we can use both type-II and type-III pairing groups for our threshold signature scheme.

4.3 Threshold Signature

In this section, we introduce the syntax and security definitions for threshold signature schemes. We focus on schemes that have non-interactive signing and deterministic verification. Our security definitions are based on those of [BS23].

Definition 2 (Non-Interactive Threshold Signature). *Let t, n with $t < n$ be natural numbers. A non-interactive (n,t)-threshold signature scheme* TS *for a finite message space $\mathcal{M}$ is a tuple of polynomial time algorithms* $\mathsf{TS} = (\mathsf{Setup},\mathsf{KGen},\mathsf{PSign},\mathsf{PVer},\mathsf{Comb},\mathsf{Ver})$ *defined as follows:*

1. $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ *takes as input a security parameter and outputs public parameters* pp *(which are given implicitly as input to all other algorithms).*
2. $\mathsf{KGen}() \to \mathsf{pk}, \{\mathsf{pk}_i, \mathsf{sk}_i\}_{i\in[n]}$ *outputs a public key* pk*, a vector of threshold public keys $\{\mathsf{pk}_1,\ldots,\mathsf{pk}_n\}$, and a vector of secret key shares $\{\mathsf{sk}_1,\ldots,\mathsf{sk}_n\}$. The j-th signer receives $(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{sk}_j)$.*
3. $\mathsf{PSign}(\mathsf{sk}_i, m) \to \sigma_i$ *takes as input a secret key share sk_i, and a message $m \in \mathcal{M}$. It outputs a signature share σ_i.*
4. $\mathsf{PVer}(\mathsf{pk}_i, m, \sigma_i,) \to 0/1$ *takes as input a threshold public key share pk_i, a message m, and a signature share σ_i. It outputs 1 (accept) or 0 (reject).*
5. $\mathsf{Comb}(S, m, \{(\mathsf{pk}_i, \sigma_i)\}_{i\in S}) \to \sigma/\perp$ *takes as input a set S with $|S| \geq t+1$, a message m, and a set of tuples $(\mathsf{pk}_i, \sigma_i)$ consisting of public keys and signature shares of signers in S. It outputs either a signature σ or $\perp$.*
6. $\mathsf{Ver}(\mathsf{pk}, m, \sigma) \to 0/1$ *takes as input a public key* pk*, a message m, and a signature σ. It outputs 1 (accept) or 0 (reject).*

Game $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{pk}, \{\mathsf{pk}_i, \mathsf{sk}_i\}_{i\in[n]} \leftarrow \mathsf{KGen}(\mathsf{pp})$
Let $\mathcal{C} := \emptyset, \mathcal{H} := [n]$
$\mathsf{inp} := \mathsf{pp}, \mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}$
// $Q[m]$, initially $\{\}$, denotes the set of signers $\mathcal{A}$ queries for the partial signatures on m
$(m, \sigma) \leftarrow \mathcal{A}^{\text{CORR,PSIG}}(\mathsf{inp})$
if $|Q[m] \cup \mathcal{C}| \leq t \wedge \mathsf{Ver}(m, \mathsf{pk}, \sigma) = 1$:
  return 1
return 0

Oracle CORR(i):

if $\mathcal{C} \geq t$: **return** $\perp$
$\mathcal{C} := \mathcal{C} \cup \{i\}; \quad \mathcal{H} := \mathcal{H} \setminus \{i\}$
return sk_i

Oracle PSIG(i, m):

if $i \in \mathcal{H}$:
  $Q[m] := Q[m] \cup \{i\}$
  Let $\sigma_i \leftarrow \mathsf{PSign}(m, \mathsf{sk}_i)$
  return σ_i
return $\perp$

Game $\mathsf{RB\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{pk}, \{\mathsf{pk}_i, \mathsf{sk}_i\}_{i\in[n]} \leftarrow \mathsf{KGen}(\mathsf{pp})$
Let $\mathcal{C} := \emptyset, \mathcal{H} := [n]$
$\mathsf{inp} := \mathsf{pp}, \mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}$
// Verification of honest partial signatures are always successful
$i, m \leftarrow \mathcal{A}^{\text{CORR,PSIG}}(\mathsf{inp})$
$\sigma_i \leftarrow \mathsf{PSign}(\mathsf{sk}_i, m)$
if $\mathsf{PVer}(\mathsf{pk}_i, m, \sigma_i) \neq 1$:
  return 1
// Combining valid partial signature must yield valid threshold signatures
$S, m', \{\sigma_i\}_{i\in S} \leftarrow \mathcal{A}^{\text{CORR,PSIG}}(\mathsf{inp})$
assert $|S| \geq t + 1$
assert $\mathsf{PVer}(\mathsf{pk}_i, m', \sigma_i) = 1, \forall i \in S$
$\sigma := \mathsf{Comb}(S, m', \{\mathsf{pk}_i, \sigma_i\}_{i\in S})$
if $\mathsf{Ver}(\mathsf{pk}, m', \sigma) \neq 1$:
  return 1
return 0

Fig. 1: The unforgeability security game $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ and the robustness security game $\mathsf{RB\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ for a non-interactive (n, t)-threshold signature $\mathsf{TS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{PSign}, \mathsf{Comb}, \mathsf{Ver})$ with an adaptive adversary $\mathcal{A}$.

We require a non-interactive (n, t)-threshold signature scheme to satisfy *Unforgeability* and *Robustness* properties we describe next.

We formalize the unforgeability property using the $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game in Figure 1. Let $\mathcal{A}$ be the adversary in the $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game. $\mathcal{A}$ gets as input the public parameters pp, an honestly generated public key pk and threshold public keys $\{\mathsf{pk}_i\}_{i\in[n]}$. At any point in time, $\mathcal{A}$ can query the partial signature on a message m from any honest signer i by querying the oracle $\text{PSIG}(i, m)$. The game also maintains a list Q to store the subset of parties $\mathcal{A}$ has queried for partial signatures, i.e., for any message m, $Q[m]$ stores the subset of honest signers $\mathcal{A}$ has queried for partial signatures on m. Initially, $Q[m] = \{\}$ for every message m.

$\mathcal{A}$ can corrupt up to t signers throughout the protocol using the CORR oracle. Upon corrupting any party, say party $i \in [n]$, $\mathcal{A}$ learns its signing key sk_i. Our protocol also has the property that the internal state used in all partial signings by a signer is efficiently computable from the signing key of the signer and the public messages sent by the signer. Thus, upon corruption, revealing only the signing key of the signer is sufficient.

Finally, when $\mathcal{A}$ outputs a valid forgery (m^*, σ^*), we say that $\mathcal{A}$ wins if $\mathcal{A}$ queried for partial signatures on m^* from at most $t - |\mathcal{C}|$ signers, i.e., $|Q[m] \cup \mathcal{C}| \leq t$.

With the $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game defined in Figure 1, we define the unforgeability under chosen message attack property as follows.

Definition 3 (Unforgeability Under Chosen Message Attack). *Let* $\mathsf{TS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{PSign}, \mathsf{Comb}, \mathsf{Ver})$ *is a* (n, t)*-threshold signature scheme. Consider the game* $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ *defined in Figure 1. We say that* TS *is* $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ *secure, if for all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible, i.e.,*

$$\varepsilon_\sigma := \mathsf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{A},\mathsf{TS}}(\lambda) := \Pr[\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}(\lambda) \Rightarrow 1] = \mathsf{negl}(\lambda) \tag{1}$$

We formalize the robustness property using the $\mathsf{RB\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game in Figure 1. Intuitively, the robustness property ensures that the protocol behaves as expected for honest parties, even in the presence of an adaptive adversary that corrupts up to t parties. More precisely, it says that: (i) PVer should always accept honestly

generated partial signatures; and (ii) if we combine $t+1$ valid partial signatures (accepted by PVer) using the Comb algorithm, the output of Comb should be accepted by Ver, except with a negligible probability. The latter requirement ensures that maliciously generated partial signatures cannot prevent an honest aggregator from efficiently computing a threshold signature (except with a negligible probability). Note that $\mathcal{A}$ can generate the partial signatures in an arbitrary manner. Also, looking ahead, our scheme achieves robustness even if $\mathcal{A}$ corrupts all parties.

Definition 4 (Robustness Under Chosen Message Attack). *Let* $\mathsf{TS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{PSign}, \mathsf{Comb}, \mathsf{Ver})$ *is a* (t, n)*-threshold signature scheme. Consider the game* $\mathsf{RB\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ *defined in Figure 1. We say that* TS *is* $\mathsf{RB\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ *secure, if for all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible, i.e.,*

$$\mathsf{Adv}^{\mathsf{RB\text{-}CMA}}_{\mathcal{A},\mathsf{TS}}(\lambda) := \Pr[\mathsf{RB\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}(\lambda) \Rightarrow 1] = \mathsf{negl}(\lambda) \tag{2}$$

4.4 Boldyreva's BLS threshold signature scheme [Bol03]

For a security parameter λ, let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, \mathbb{F}, p, g) \leftarrow \mathsf{GGen}(1^\lambda)$ with bilinear pairing operation $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$. The public parameters of Boldyreva's (n, t)-threshold signature scheme for a message space $\mathcal{M}$ are $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{F}, p, g, \mathsf{H})$, where $\mathsf{H} : \mathcal{M} \to \hat{\mathbb{G}}$ is a hash function modeled as a random oracle. The signature scheme works as follows:

- $\mathsf{KGen}()$ samples a uniformly random polynomial $s(x) \in \mathbb{F}[X]$ of degree t. The signing key of i-th signer is $\mathsf{sk}_i := s(i)$, the public key $\mathsf{pk} := g^{s(0)}$, and the threshold public keys are $\{\mathsf{pk}_i := g^{\mathsf{sk}_i}\}_{i\in[n]}$.
- $\mathsf{PSign}(\mathsf{sk}_i, m)$ computes the partial signature with respect to secret key sk_i as $\sigma_i := \mathsf{H}(m)^{\mathsf{sk}_i} \in \hat{\mathbb{G}}$.
- $\mathsf{PVer}(\mathsf{pk}_i, m, \sigma_i)$ retruns 1 if $e(\mathsf{pk}_i, \mathsf{H}(m)) = e(g, \sigma_i)$, and 0 otherwise.
- $\mathsf{Comb}(S, m, \{(\mathsf{pk}_i, \sigma_i)\})$ first checks that $|S| \geq t+1$ and then runs $\mathsf{PVer}(\mathsf{pk}_i, \sigma_i, m)$ for all $i \in S$. If any one of these calls outputs 0, then return $\perp$. Otherwise, return $\sigma := \prod_{i\in S} \sigma_i^{L_{i,S}}$, where $L_{i,S} := \prod_{i\in S} \left(\frac{j}{j-i}\right)$ is the i-th Lagrange coefficient for the set S.
- $\mathsf{Ver}(\mathsf{pk}, m, \sigma)$ returns 1 if $e(\mathsf{pk}, \mathsf{H}(m)) = e(g, \sigma)$, and 0 otherwise.

Boldyreva's scheme is secure in the presence of a *static* adversary assuming hardness of computational Diffie-Hellman assumption in the random oracle model [Bol03, BCK+22].

5 Adaptively Secure BLS Threshold Signature

In this section, we will describe our adaptively secure (n, t)-threshold signature scheme assuming that KGen is run by a trusted party.

$\underline{\mathsf{Setup}(1^\lambda)}$: Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, \mathbb{F}, p) \leftarrow \mathsf{GGen}(1^\lambda)$ be pairing groups with scalar field $\mathbb{F}$ of prime order p and bilinear pairing operation $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$. Let $g, h, v \in \mathbb{G}$ are three uniformly random independent generators of $\mathbb{G}$. Let $\mathsf{H}_0, \mathsf{H}_1 : \mathcal{M} \to \hat{\mathbb{G}}$ and $\mathsf{H}_{\mathsf{FS}} : \{0,1\}^* \to \hat{\mathbb{G}}$ be three distinct hash functions modeled as random oracles. The public parameters of our scheme are then $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{F}, g, h, v, \mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_{\mathsf{FS}})$.As we discussed earlier, we assume that all the algorithms below implicitly take the public parameters as input.

$\underline{\mathsf{KGen}()}$: Sample three uniformly random polynomials $s(x), r(x)$ and $u(x)$ of degree t each with the constraint that $r(0) = u(0) = 0$. The signing key of signer i is then $\mathsf{sk}_i := (s(i), r(i), u(i))$. Let $\mathsf{pk} := g^{s(0)} h^{r(0)} v^{u(0)} = g^{s(0)}$ be the public verification key, and $\mathsf{pk}_i := g^{s(i)} h^{r(i)} v^{u(i)}$ be party i's threshold public key.

$\underline{\mathsf{PSign}(\mathsf{sk}_i, m)}$: The partial signature of signer i on a message m is the tuple (σ_i, π_i), where $\sigma_i := \mathsf{H}_0(m)^{s(i)} \mathsf{H}_1(m)^{r(i)}$, and π_i is a non-interactive zero-knowledge (NIZK) proof of the correctness of σ_i with respect to pk_i. Signer i computes π_i using the Σ-protocol in Figure 3. We use the Fiat-Shamir heuristic to make the signing phase non-interactive.

$\underline{\mathsf{PVer}(\mathsf{pk}_i, m, \sigma_i)}$: On input the threshold public key pk_i and the partial signature tuple (σ_i, π_i), and the message m validates σ_i by running the Σ-protocol verifier $\mathcal{V}$, and accepts if and only if $\mathcal{V}$ accepts.

$\underline{\mathsf{Setup}(1^\lambda)}$:

1: $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, \mathbb{F}, p) \leftarrow \mathsf{GGen}(1^\lambda)$ be pairing groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order p, scalar field $\mathbb{F}$ and bilinear pairing operation $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$.
2: Let $g, h, v \in \mathbb{G}$ be three uniformly random independent generators of $\mathbb{G}$.
3: Let $\mathsf{H}_0, \mathsf{H}_1 : \mathcal{M} \to \hat{\mathbb{G}}$ and $\mathsf{H}_{\mathsf{FS}} : \{0,1\}^* \to \mathbb{F}$ be three hash functions modeled as random oracle.
4: **return** $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{F}, g, h, v, \mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_{\mathsf{FS}})$.

// We assume all algorithms implicitly take the output of Setup as input. We use H_{FS} in SigmaProve and SigmaVer.

$\underline{\mathsf{KGen}()}$:

5: Let $s(\cdot), r(\cdot), u(\cdot) \leftarrow\$ \mathbb{F}[x]$ be three polynomials of degree t with $r(0) = u(0) = 0$.
6: Let $\mathsf{pk} := g^{s(0)} h^{r(0)} v^{u(0)} = g^{s(0)}$
7: **for** each $i \in [n]$:
8: Let $\mathsf{sk}_i := (s(i), r(i), u(i))$
9: Let $\mathsf{pk}_i := g^{s(i)} h^{r(i)} v^{u(i)}$
10: **return** $(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{sk}_j)$ to signer j for all $j \in [n]$

$\underline{\mathsf{PSign}(\mathsf{sk}_i = (s_i, r_i, u_i), m)}$:

11: Let $\sigma_i := \mathsf{H}_0(m)^{s_i} \mathsf{H}_1(m)^{r_i}$
12: Let $\pi_i := \mathsf{SigmaProve}(\mathsf{pk}_i, m, \sigma_i, \mathsf{sk}_i)$
13: **return** σ_i, π_i

$\underline{\mathsf{PVer}(\mathsf{pk}_i, m, (\sigma_i, \pi_i))}$:

14: **return** $\mathsf{SigmaVer}(\mathsf{pk}_i, m, \sigma_i, \pi_i)$

$\underline{\mathsf{Comb}(S, m, \{(\mathsf{pk}_i, (\sigma_i, \pi_i))\}_{i\in S}}$:

15: **assert** $|S| \geq t + 1$
16: **for** each $i \in S$:
17: **assert** $\mathsf{PVer}(\mathsf{pk}_i, m, (\sigma_i, \pi_i))$
18: Let $L_{i,S}$ be the i-th Lagrange coefficients for S
19: **return** $\sigma := \prod_{i\in S} \sigma_i^{L_{i,S}}$

$\underline{\mathsf{Ver}(\mathsf{pk}, m, \sigma)}$:

21: **if** $e(\mathsf{pk}, \mathsf{H}_0(m)) = e(g, \sigma)$:
22: **return** 1
23: **return** 0

Fig. 2: Adaptively secure (n, t) BLS threshold signature with trusted key generation.

Input: $(g, h, v, \mathsf{pk}) \in \mathbb{G}^4$, $(\hat{g}_0, \hat{g}_1) = (\mathsf{H}_0(m), \mathsf{H}_1(m))$ for some $m \in \mathcal{M}$, $\sigma \in \hat{\mathbb{G}}$
Witness: $(s, r, u) \in \mathbb{F}^3$
The prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$ that it knows $s, r, u \in \mathbb{F}$ such that $\mathsf{pk} = g^s h^r v^u$ and $\sigma = \hat{g}_0^s \hat{g}_1^r$.

// We assume that both algorithms implicitly take of $g, h, v, \mathsf{H}_0, \mathsf{H}_1$ as input

$\underline{\mathsf{SigmaProve}(\mathsf{pk}, m, \sigma, (s, r, u))}$:

1: Let $\hat{g}_0 := \mathsf{H}_0(m)$ and $\hat{g}_1 := \mathsf{H}_1(m)$
2: Sample $a_s, a_r, a_u \leftarrow\$ \mathbb{F}$. Let $x := g^{a_s} h^{a_r} v^{a_u}$, and $y := \mathsf{H}_0(m)^{a_s} \mathsf{H}_1(m)^{a_r}$.
3: Let $c := \mathsf{H}_{\mathsf{FS}}(x, y, \mathsf{pk}, \sigma, \hat{g}_0, \hat{g}_1)$, for hash function $\mathsf{H}_{\mathsf{FS}} : \{0,1\}^* \to \mathbb{F}$ modeled as a random oracle.
4: Let $z_s := a_s + s \cdot c$, $z_r := a_r + r \cdot c$ and $z_u := a_u + u \cdot c$.
5: **return** $\pi := (x, y, z_s, z_r, z_u)$.

$\underline{\mathsf{SigmaVer}(\mathsf{pk}, m, \sigma, \pi = (x, y, z_s, z_r, z_u))}$:

6: Let $\hat{g}_0 := \mathsf{H}_0(m)$ and $\hat{g}_1 := \mathsf{H}_1(m)$
7: Let $c := \mathsf{H}_{\mathsf{FS}}(x, y, \mathsf{pk}, \sigma, \hat{g}_0, \hat{g}_1)$
8: **if** $g^{z_s} h^{z_r} v^{z_u} = x \cdot \mathsf{pk}^c$ and $\hat{g}_0^{z_s} \hat{g}_1^{z_r} = y \cdot \sigma^c$:
9: **return** 1
10: **return** 0

Fig. 3: Σ-protocol for computing and verifying the correctness proof for partial signatures.

$\underline{\mathsf{Comb}(S, m, \{(\mathsf{pk}_i, (\sigma_i, \pi_i))\}_{i\in S}}$: Upon receiving a set of signers S with $|S| \geq t + 1$, a message m, and the corresponding threshold public-key and partial signatures tuples $\{(\mathsf{pk}_i, (\sigma_i, \pi_i))\}_{i\in S}$, first validates each of the partial signature using PVer. If any of these partial signatures verification fails, i.e., returns 0, the Comb algorithm returns $\perp$. Otherwise, the Comb algorithm computes the threshold signature σ as:

$$\sigma := \prod_{i\in T} \sigma_i^{L_{i,S}} \tag{3}$$

where $L_{i,S}$ is the i-th Lagrange coefficient with respect to the set S.

$\underline{\mathsf{Ver}(\mathsf{pk}, m, \sigma)}$: The verification procedure of our scheme is identical to that of the standard BLS signature: on input the public key pk and the signature σ on a message m, a verifier accepts if $e(\mathsf{pk}, \mathsf{H}_0(m)) = e(g, \sigma)$.

Remark. Note that signers do not use $u(i)$ while computing σ_i. It is in the public verification key (and hence used in computing π_i) as an artifact to make our adaptive security proof go through.

6 Proofs of Adaptive Security

We first analyze the properties of the Σ-protocol in Figure 3, which we then use to prove the robustness and adaptive security of our threshold signature scheme.

6.1 Properties of the Σ-protocol

We require the Σ-protocol to satisfy the standard *completeness, knowledge-soundness*, and *zero-knowledge* properties [Dam02]. Briefly, the completeness property guarantees that an honest prover will always be able to convince an honest verifier about true statements. The knowledge soundness property ensures that, for every prover who convinces an honest verifier about a statement with a non-negligible probability, there exists an efficient extractor who interacts with the prover to compute the witness. Finally, the zero-knowledge property ensures that the proof reveals no information other than the statement's truth. We remark that achieving zero-knowledge against honest verifiers is sufficient for our purposes. The completeness of our Σ-protocol is straightforward. The knowledge soundness and honest-verifier zero-knowledge properties also follow from standard Σ-protocol analysis.

Knowledge soundness. We prove knowledge soundness by extractability. For any PPT prover $\mathcal{P}$, let $\mathcal{E}$ be the extractor. Then $\mathcal{E}$ interacts with $\mathcal{P}$ with two different challenges c and c' on the same first message to receive two pairs of valid responses (z_s, z_r, z_u) and (z'_s, z'_r, z'_u). Then, we have:

$$g^{z_s - z'_s} h^{z_r - z'_r} v^{z_u - z'_u} = \mathsf{pk}^{c-c'} \quad \text{and} \quad \mathsf{H}_0(m)^{z_s - z'_s} \mathsf{H}_1(m)^{z_r - z'_r} = \sigma^{c-c'}$$
$$\implies s = \frac{z_s - z'_s}{c - c'}; \quad r = \frac{z_r - z'_r}{c - c'}; \quad u = \frac{z_u - z'_u}{c - c'}$$

Let $\varepsilon_{\mathsf{ext}}$ be the success probability of the extractor $\mathcal{E}$. Then, it follows from the generalized forking lemma [BN06] that $\varepsilon_{\mathsf{ext}} \geq \varepsilon^2 / q_{\mathsf{FS}} - \varepsilon / |\mathbb{F}|$ where ε is the probability that an adversary $\mathcal{A}$ outputs a valid response while making at most q_{FS} random oracle queries to H_{FS}.

Honest verifier zero-knowledge (HVZK). Let $\mathcal{S}$ be the simulator. $\mathcal{S}$ samples uniformly random $(c, z_s, z_r, z_u) \in \mathbb{F}^4$ and computes x and y as

$$x := g^{z_s} h^{z_r} v^{z_u} \cdot \mathsf{pk}^{-c} \quad \text{and} \quad y =: \mathsf{H}_0(m)^{z_s} \mathsf{H}_1(m)^{z_r} \cdot \sigma^{-c} \tag{4}$$

$\mathcal{S}$ then programs the random oracle such that $\mathsf{H}_{\mathsf{FS}}(x, y, \mathsf{pk}, \sigma, m) = c$ and outputs $\pi = (c, z_s, z_r, z_u)$ as the proof. Clearly, the simulated transcript is identically distributed to the real-protocol transcript.

6.2 Robustness

Before we prove the robustness of our scheme, we prove the following helper lemma.

Lemma 1. *If any signer i with threshold public key $\mathsf{pk}_i = g^{s(i)} h^{r(i)} v^{u(i)}$ outputs a partial signature σ_i on a message m along with a valid Σ-protocol proof π_i as per Figure 3, then assuming hardness of discrete logarithm in $\mathbb{G}$, σ_i is well-formed, i.e., $\sigma_i = \mathsf{H}_0(m)^{s(i)} \mathsf{H}_1(m)^{r(i)}$.*

Proof. For valid Σ-protocol proof π_i, let $\mathcal{E}$ be the extractor from §6.1 and let s', r', u' be the extracted witness. We need to prove $(s', r', u') = (s(i), r(i), u(i))$.

For the sake of contradiction, assume this is not the case. Then, we can construct an adversary $\mathcal{A}_{\mathrm{DL}}$ that breaks the discrete logarithm in $\mathbb{G}$ as follows. On input a discrete logarithm instance $(g, y) \in \mathbb{G}^2$, $\mathcal{A}_{\mathrm{DL}}$ samples $\theta \in \{0, 1\}$ and sets either $h = y$ or $v = y$ depending on the value of θ. $\mathcal{A}_{\mathrm{DL}}$ picks the other parameter as g^α for some known uniformly random $\alpha \in \mathbb{F}$. $\mathcal{A}_{\mathrm{DL}}$ next faithfully emulates the trusted key generation with

$\mathcal{A}$ with some chosen polynomials $s(\cdot), r(\cdot), v(\cdot)$. $\mathcal{A}_{\mathrm{DL}}$ also faithfully emulates the corruption, partial signature queries, and random oracle queries.

Now $(s', r', u') \neq (s(i), r(i), u(i))$ for any signer i implies that

$$g^{s'-s(i)} h^{r-r(i)} v^{u'-u(i)} = 1_{\mathbb{G}} \tag{5}$$

where $1_{\mathbb{G}}$ is the identity element of $\mathbb{G}$.

Say $h = g^{\alpha_h}$ and $v = g^{\alpha_v}$ for some $\alpha_h, \alpha_v \in \mathbb{F}$, and let $\delta_s := s' - s(i)$, $\delta_r := r' - r(i)$, and $\delta_u := u' - u(i)$. Then, equation (5), implies that $\delta_s + \delta_r \alpha_h + \delta_u \alpha_v = 0$. If either δ_r or δ_u is non-zero, then we can compute α_h or α_v, respectively, as:

$$\delta_r \neq 0 \implies \alpha_h = (-\delta_s - \alpha_v \delta_u) \cdot \delta_r^{-1}; \quad \delta_u \neq 0 \implies \alpha_v = (-\delta_s - \alpha_h \delta_r) \cdot \delta_u^{-1} \tag{6}$$

Finally, $(\delta_r, \delta_u) = (0, 0)$, implies that $\delta_s = 0$. Since $\mathcal{A}_{\mathrm{DL}}$ uses y as either h or v uniformly at random, it implies that if the extractor $\mathcal{E}$ outputs $(s', r', u') \neq (s(i), r(i), u(i))$ with probability $\varepsilon_{\mathsf{ext}}$, then $\mathcal{A}_{\mathrm{DL}}$ outputs the discrete logarithm of y with respect to g, with probability at least $\varepsilon_{\mathsf{ext}}/2$. □

We will now prove that robustness, i.e., any PPT adversary $\mathcal{A}$ wins the $\mathsf{RB\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game in Figure 1 only with a negligible probability. More formally,

Theorem 3 (Robustness). *The non-interactive (n, t)-threshold signature scheme* $\mathsf{TS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{PSign}, \mathsf{PVer}, \mathsf{Comb}, \mathsf{Ver})$ *in Figure 2 is* $\mathsf{RB\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ *secure.*

Proof. There are two possible winning cases for an adversary $\mathcal{A}$ in the $\mathsf{RB\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game: (1) honestly computed partial signatures do not satisfy the validation check PVer (line 23 in the $\mathsf{RB\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game in Figure 1), and (2) every partial signature passes PVer but the honestly aggregated full signature does not satisfy the validation check Ver (line 29 in Figure 1).

Let us first analyze the first winning case. Note that the PVer algorithm in our protocol runs the verifier of the Σ-protocol in Figure 3. Then, the completeness property of the Σ-protocol guarantees that the Σ-protocol verifier always accepts honestly generated proofs. This implies that the winning condition in line 8 in Figure 1 never occurs for our protocol.

Now let us consider the second winning case. Lemma 1 ensures that assuming hardness of discrete logarithm in $\mathbb{G}$, the aggregator only aggregates well-formed partial signatures. Thus, we get

$$\begin{aligned}\sigma &= \prod_{i \in S} \sigma^{L_{i,S}} = \prod_{i \in S} \mathsf{H}_0(m)^{s(i) L_{i,S}} \mathsf{H}_1(m)^{r(i) L_{i,S}} \\ &= \mathsf{H}_0(m)^{\sum_{i \in S} s(i) L_{i,S}} \mathsf{H}_1(m)^{\sum_{i \in S} r(i) L_{i,S}} = \mathsf{H}_0(m)^s \mathsf{H}_1(m)^0 = \mathsf{H}_0(m)^s.\end{aligned}$$

Note that $\sigma = \mathsf{H}_0(m)^s$ always satisfies the final verification check Ver.

Thus, we get that assuming the hardness of discrete logarithm in $\mathbb{G}$, any PPT adversary $\mathcal{A}$ wins the $\mathsf{RB\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game only with a negligible probability. □

6.3 Helper Lemmas for Unforgeability

Our unforgeability proof crucially relies on the following lemma from Naor-Reingold [NR04, Lemma 4.4]. We refer the reader to [NR04] for its proof.

Lemma 2 (Lemma 4.4 in [NR04]). *For any security parameter λ, let $(\hat{\mathbb{G}}, \mathbb{F}, p, \hat{g}) \leftarrow \mathsf{GGen}(1^\lambda)$ be a cyclic group of prime order p with scalar field $\mathbb{F}$ and generator $\hat{g} \in \hat{\mathbb{G}}$. For all $q_{\mathsf{H}} \leq \mathsf{poly}(\lambda)$, assuming hardness of decisional Diffie-Hellman (DDH) assumption in $\hat{\mathbb{G}}$, the following two distributions are indistinguishable.*

$$\mathcal{D}_0 := \hat{g}, \hat{g}^\alpha, \{(\hat{g}^{\beta_i}, \hat{g}^{\gamma_i})\}_{i \in [q_{\mathsf{H}}]} \text{ for } \alpha \leftarrow\$ \mathbb{F} \text{ and } \forall i \in [q_{\mathsf{H}}]\ (\beta_i, \gamma_i) \leftarrow\$ \mathbb{F}^2 \tag{7}$$

$$\mathcal{D}_1 := \hat{g}, \hat{g}^\alpha, \{(\hat{g}^{\beta_i}, \hat{g}^{\alpha \cdot \beta_i})\}_{i \in [q_{\mathsf{H}}]} \text{ for } \alpha \leftarrow\$ \mathbb{F} \text{ and } \forall i \in [q_{\mathsf{H}}]\ \beta_i \leftarrow\$ \mathbb{F} \tag{8}$$

More precisely, if an adversary $\mathcal{A}$ can distinguish between a sample from $\mathcal{D}_0$ and $\mathcal{D}_1$ with probability ε, then $\mathcal{A}$ can break the DDH assumption with probability at least $\varepsilon - 1/|\mathbb{F}|$. This implies $\varepsilon \leq \varepsilon_{\mathrm{DDH}} + 1/|\mathbb{F}|$.

We use the abovementioned lemma to prove the following.

Lemma 3. *For security parameter λ, let $(\hat{\mathbb{G}}, \mathbb{F}, p, \hat{g}) \leftarrow \mathsf{GGen}(1^\lambda)$ be a cyclic group of prime order p with scalar field $\mathbb{F}$ and generator $\hat{g} \in \hat{\mathbb{G}}$. For all $q_\mathsf{H} \leq \mathsf{poly}(\lambda)$ and any fixed $k \in [q_\mathsf{H}]$, let the distribution $\mathcal{D}_{1,k}$ be defined as follows:*

$$\mathcal{D}_{1,k} := g, \{(g^{\beta_i}, g^{\gamma_i})\}_{i\in[q_\mathsf{H}]} \text{ for } \alpha \leftarrow\$ \mathbb{F} \text{ and } \begin{cases} \forall i \neq k,\ \beta_i \leftarrow\$ \mathbb{F}, \gamma_i := \alpha \cdot \beta_i \\ i = k,\ (\beta_i, \gamma_i) \leftarrow\$ \mathbb{F}^2 \end{cases}$$

Then, assuming hardness of DDH in $\hat{\mathbb{G}}$, the distributions $\mathcal{D}_0$ (defined in Lemma 2) and $\mathcal{D}_{1,k}$ are indistinguishable except with probability at most $\varepsilon_{\text{DDH}} + 1/|\mathbb{F}|$.

Proof. Define $\mathcal{D}_{0,k}$ to be identical to $\mathcal{D}_0$ for notational convenience. For any fixed k, given a sample $(g, g^\alpha, \{(g^{\beta_i}, g^{\gamma_i})\}$ from $\mathcal{D}_\theta$ for either $\theta \in \{0, 1\}$ we can get a sample from $\mathcal{D}_{\theta,k}$ by substituting g^{γ_k} in the given sample with a uniformly random element in $\hat{\mathbb{G}}$ and dropping the term g^α.

6.4 Unforgeability with an Adaptive Adversary

We will prove the unforgeability assuming the hardness of the DDH in $\hat{\mathbb{G}}$ and the hardness of co-CDH in $(\mathbb{G}, \hat{\mathbb{G}})$. Let $\mathcal{A}_{\text{co-CDH}}$ be the reduction adversary. Upon input a co-CDH instance $(g, \hat{g}, g^a, \hat{g}^a, \hat{g}^b)$, $\mathcal{A}_{\text{co-CDH}}$ interacts with $\mathcal{A}$ such that when $\mathcal{A}$ forges a signature, $\mathcal{A}_{\text{co-CDH}}$ uses the forgery to compute $\hat{g}^{ab}$. We summarize $\mathcal{A}_{\text{co-CDH}}$ interaction with $\mathcal{A}$ in Figure 4 and describe it next.

Simulating the public parameters. On a co-CDH input $(g, \hat{g}, g^a, \hat{g}^a, \hat{g}^b)$, $\mathcal{A}_{\text{co-CDH}}$ samples $\alpha_v \leftarrow\$ \mathbb{F}$, sets $h := g^a, v := g^{\alpha_v}$, and sends (g, h, v) to $\mathcal{A}$. $\mathcal{A}_{\text{co-CDH}}$ provides $\mathcal{A}$ access to the random oracles using lazy programming, i.e., $\mathcal{A}_{\text{co-CDH}}$ programs random oracles on any input only upon a query.

Simulating the KGen functionality. $\mathcal{A}_{\text{co-CDH}}$ samples $s, u \leftarrow\$ \mathbb{F}$ and three uniformly random degree t polynomials $s(\cdot), r(\cdot), u(\cdot) \in \mathbb{F}[x]$, but crucially with the constraints $s(0) = s$, $u(0) = u$, and $r(0) = 1$ for the multiplicative identity 1 in $\mathbb{F}$. $\mathcal{A}_{\text{co-CDH}}$ then computes the public key and threshold public keys as follows:

$$\mathsf{pk} := g^{s(0)} h^{r(0)} v^{u(0)} = g^s h v^u; \quad \text{and} \quad \left\{\mathsf{pk}_i := g^{s(i)} h^{r(i)} v^{u(i)}\right\}_{i\in[n]} \tag{11}$$

$\mathcal{A}_{\text{co-CDH}}$ then sends $\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}$ to $\mathcal{A}$.

Simulating corruption queries. Let $\mathcal{H}$ and $\mathcal{C} = [n] \setminus \mathcal{H}$ be the set of honest and malicious parties, respectively. Anytime during the signing phase, if $\mathcal{A}$ corrupts signer $i \in [n]$, $\mathcal{A}_{\text{co-CDH}}$ checks whether $|\mathcal{C}| < t$ or not. If the check is successful, $\mathcal{A}_{\text{co-CDH}}$ faithfully reveals the secret signing key $\mathsf{sk}_i := (s(i), r(i), u(i))$ of signer i, and updates $\mathcal{C} := \mathcal{C} \cup \{i\}$ and $\mathcal{H} := \mathcal{H} \setminus \{i\}$. $\mathcal{A}_{\text{co-CDH}}$ lets $\mathcal{A}$ only corrupt up to t signers. Otherwise, $\mathcal{A}_{\text{co-CDH}}$ outputs $\perp$.

Simulating threshold signature. $\mathcal{A}_{\text{co-CDH}}$ simulates the signing queries by programming the random oracles as follows. Let $\alpha = a + \alpha_v u$. Note that H_0 is always queried on the forged message, at least by $\mathcal{A}_{\text{co-CDH}}$ during the signature verification. Moreover, whenever $\mathcal{A}$ queries H_θ for either $\theta \in \{0, 1\}$ on any message, $\mathcal{A}_{\text{co-CDH}}$ internally queries $\mathsf{H}_{1-\theta}$ on the same message. Let q_H be an upper bound on the number of queries by $\mathcal{A}$ to H_0 and H_1 combined. $\mathcal{A}_{\text{co-CDH}}$ samples $\hat{k} \leftarrow\$ [q_\mathsf{H}]$. On the k-th random oracle query on message m_k, depending upon the value of k, $\mathcal{A}_{\text{co-CDH}}$ programs the random oracles as follows.

$$\begin{aligned} k \neq \hat{k} &\implies \mathsf{H}_0(m_k) := \hat{g}^{\beta_k};\ \mathsf{H}_1(m_k) := \hat{g}^{\alpha\cdot\beta_k} \text{ for } \beta_k \leftarrow\$ \mathbb{F} \\ k = \hat{k} &\implies \mathsf{H}_0(m_k) := \hat{g}^b; \quad \mathsf{H}_1(m_k) := \hat{g}' \text{ for } \hat{g}' \leftarrow\$ \hat{\mathbb{G}} \end{aligned}$$

Let $m_{\hat{k}}$ be the queried message for $k = \hat{k}$. Then, except for message $m_{\hat{k}}$, $\mathcal{A}_{\text{co-CDH}}$ always responds to partial signing queries as per the honest protocol. For message $m_{\hat{k}}$, $\mathcal{A}_{\text{co-CDH}}$ faithfully responds to up to $t - |\mathcal{C}|$ partial signing queries and aborts if $\mathcal{A}$ queries for more partial signatures on $m_{\hat{k}}$.

Input: co-CDH tuple $(g, g^a, \hat{g}, \hat{g}^a, \hat{g}^b) \in \mathbb{G}^3 \times \hat{\mathbb{G}}$.

KGen simulation:

1. Let $\alpha_v \leftarrow\$ \mathbb{F}$. Let $h := g^a$ and $v := g^{\alpha_v}$.
2. Let $s, u \leftarrow\$ \mathbb{F}$. Sample three uniformly random degree t polynomials $s(x), r(x), u(x) \in \mathbb{F}[x]$ with the constraints $s(0) = s$, $u(0) = u$ and $r(0) = 1$. Here 1 is the multiplicative identity element of the field $\mathbb{F}$.
3. Compute $\mathsf{pk} := g^{s(0)} h^{r(0)} v^{u(0)} = g^s h v^u$, and for each $i \in [n]$, $\mathsf{pk}_i := g^{s(i)} h^{r(i)} v^{u(i)}$
4. For each $i \in [n]$, let $\mathsf{sk}_i := (s(i), r(i), u(i))$. Send $\mathsf{pk}, \{\mathsf{pk}_j\}_{j \in [n]}$ to $\mathcal{A}$.

Corruption simulation:

5. Let $\mathcal{H}$ and $\mathcal{C} = [n] \setminus \mathcal{H}$ be the set of honest and malicious parties, respectively.
6. When $\mathcal{A}$ submits a corruption query i, if $|\mathcal{C}| \geq t$, respond with $\perp$. Otherwise, send sk_i to $\mathcal{A}$. Update $\mathcal{H} := \mathcal{H} \setminus \{i\}$ and $\mathcal{C} := \mathcal{C} \cup \{i\}$.

Threshold signature simulation:

7. For each query to H_{FS} on input x, Return $\mathsf{H}_{\mathsf{FS}}(x)$ if $\mathsf{H}_{\mathsf{FS}}(x) \neq \perp$. Otherwise, return $\mathsf{H}_{\mathsf{FS}}(x) := y \leftarrow\$ \mathbb{F}$.
8. Let $\alpha := a + \alpha_v u$, thus $\hat{g}^\alpha := \hat{g}^a \cdot \hat{g}^{\alpha_v u}$. // $\mathcal{A}_{\text{co-CDH}}$ does not know α.
9. Let q_{H} be an upper bound on the total number of random oracle queries to H_0 and H_1, combined.
10. Sample $\hat{k} \leftarrow\$ [q_{\mathsf{H}}]$.
11. On k-th random oracle query to H_θ for either $\theta \in \{0, 1\}$ on message m_k:
    (a) If $\mathsf{H}_\theta(m_k) \neq \perp$, return $\mathsf{H}_\theta(m_k)$. Otherwise,
    (b) If $k \neq \hat{k}$, program the random oracles as follows and return $\mathsf{H}_\theta(m_k)$.

$$\mathsf{H}_0(m_k) := \hat{g}^{\beta_k};\ \mathsf{H}_1(m_k) := (\hat{g}^\alpha)^{\beta_k} \text{ for } \beta_k \leftarrow\$ \mathbb{F} \tag{9}$$

    (c) If $k = \hat{k}$, set the random oracles as follows and return $\mathsf{H}_\theta(m_k)$.

$$\mathsf{H}_0(m_k) := \hat{g}^b; \quad \mathsf{H}_1(m_k) := \hat{g}' \text{ for } \hat{g}' \leftarrow\$ \hat{\mathbb{G}} \tag{10}$$

12. Let $m_{\hat{k}}$ be the queried message for $k = \hat{k}$. Then, except for message $m_{\hat{k}}$, respond to partial signing queries as per the honest protocol.
13. For message $m_{\hat{k}}$, faithfully respond to up to $t - |\mathcal{C}|$ partial signing queries and abort if $\mathcal{A}$ queries for more partial signatures on $m_{\hat{k}}$.

Compute co-CDH output:

14. When $\mathcal{A}$ outputs a valid forgery $(m_{\hat{k}}, \sigma)$, output $\sigma \cdot (\hat{g}^b)^{-(s+\alpha_v u)}$ as the co-CDH solution.

Fig. 4: $\mathcal{A}_{\text{co-CDH}}$'s interaction with $\mathcal{A}$ to compute the co-CDH solution, when signers use the KGen functionality to setup the signing keys.

Computing the co-CDH solution. When $\mathcal{A}$ outputs a valid forgery $(m_{\hat{k}}, \sigma)$, $\mathcal{A}_{\text{co-CDH}}$ uses its knowledge of (s, u) and computes the co-CDH solution as follows:

$$\hat{g}_{\text{cdh}} := \sigma \cdot (\hat{g}^b)^{-b(s+\alpha_v u)} \tag{12}$$

Lemma 4. *If $(m_{\hat{k}}, \sigma)$ is a valid forgery, then $\hat{g}_{\text{cdh}}$ is the valid co-CDH solution.*

Proof. Since $(m_{\hat{k}}, \sigma)$ is a valid forgery, the following holds.

$$e(\mathsf{pk}, \mathsf{H}_0(m_{\hat{k}})) = e(g, \sigma) \implies e(g^s h v^u, \hat{g}^b) = e(g, \sigma) \tag{13}$$

Let $\hat{h} = \hat{g}^a$ and $\hat{v} = \hat{g}^{\alpha_v}$. Then, from equation (13), we get that:

$$\sigma = \left(\hat{g}^s \hat{h} \hat{v}^u\right)^b \implies \sigma \cdot \hat{g}^{-b(s+\alpha_v u)} = \hat{h}^b = \hat{g}^{ab} = \hat{g}_{\text{cdh}} \qquad \square$$

Next, we illustrate that assuming the hardness of DDH in $\hat{\mathbb{G}}$, if $\mathcal{A}$ forges a signature in the $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game, then $\mathcal{A}$ also forges a signature during its interaction with $\mathcal{A}_{\text{co-CDH}}$, just with a slightly lower probability.

We will illustrate this via a sequence of games. Game $\mathbf{G}_0$ is the real protocol execution, and game $\mathbf{G}_7$ is the interaction of $\mathcal{A}$ with $\mathcal{A}_{\text{co-CDH}}$. Here on, for any game $\mathbf{G}_i$, we will use "$\mathbf{G}_i \Rightarrow 1$" as a shorthand for the event that a PPT adversary $\mathcal{A}$ forges a signature in game $\mathbf{G}_i$.

GAME $\mathbf{G}_0$: This game is the security game $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ for our threshold signature scheme, where the game follows the honest protocol. Here, the game provides $\mathcal{A}$ access to any random oracle using the standard lazy simulation technique.

We also make a purely conceptual change to the game. Let (m^*, σ^*) be the forgery. Then, we assume that $\mathcal{A}$ always queries $\mathsf{H}_0(m^*)$ before outputting the forgery. This is without loss of generality and does not change the advantage of $\mathcal{A}$ because one could build a wrapper adversary that internally runs $\mathcal{A}$ but queries $\mathsf{H}_0(m^*)$ before outputting. Then by definition, we have:

$$\mathsf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{A},\mathsf{TS}}(\lambda) = \Pr[\mathbf{G}_0 \Rightarrow 1] = \varepsilon_\sigma.$$

GAME $\mathbf{G}_1$: Let q_{H} be the upper bound on the total number of random oracle queries to H_0 and H_1. For each $k \in [q_{\mathsf{H}}]$, let m_k be the input to the k-th random oracle query. This game is identical to $\mathbf{G}_0$, except that we sample $\hat{k} \leftarrow\$ [q_{\mathsf{H}}]$, and the game aborts if the $\mathcal{A}$ forges a message m_k for $k \neq \hat{k}$ or queries for more than $t - |\mathcal{C}|$ partial signatures for $m_{\hat{k}}$. Clearly, if no abort occurs, games $\mathbf{G}_0$ and $\mathbf{G}_1$ are the same. Furthermore, the view of $\mathcal{A}$ is independent of $\hat{k}$. Thus, we get:

$$\Pr[\mathbf{G}_1 \Rightarrow 1] \geq \frac{1}{q_{\mathsf{H}}} \cdot \Pr[\mathbf{G}_0 \Rightarrow 1] \tag{14}$$

GAME $\mathbf{G}_2$: This game is identical to $\mathbf{G}_1$, except that we sample $\alpha_h, \alpha_v \leftarrow\$ \mathbb{F}$ and set $h := g^{\alpha_h}$ and $v := g^{\alpha_v}$. Clearly, the view of $\mathcal{A}$ in $\mathbf{G}_1$ is identical to its view in $\mathbf{G}_2$, hence $\Pr[\mathbf{G}_1 \Rightarrow 1] = \Pr[\mathbf{G}_2 \Rightarrow 1]$.

GAME $\mathbf{G}_3$: In this game, we change how we program the random oracles H_0 and H_1. In particular, we program the random oracles $\mathsf{H}_0, \mathsf{H}_1$ in a correlated manner to ensure a distribution identical to how $\mathcal{A}_{\text{co-CDH}}$ programs these random oracles in Figure 4. The rest of the steps are identical to game $\mathbf{G}_2$.

More specifically, in game $\mathbf{G}_3$, we sample $u \leftarrow\$ \mathbb{F}$ and let $\alpha := \alpha_h + \alpha_v u$. Then, for the k-th random oracle query, depending upon whether $k = \hat{k}$, we program the random oracles as follows:

$$k \neq \hat{k} \implies \mathsf{H}_0(m_k) := \hat{g}^{\beta_k};\ \mathsf{H}_1(m_k) := \hat{g}^{\alpha \cdot \beta_k} \text{ for } \beta_k \leftarrow\$ \mathbb{F} \tag{15}$$

$$k = \hat{k} \implies \mathsf{H}_0(m_k) := \hat{g}^{\beta};\quad \mathsf{H}_1(m_k) := \hat{g}' \text{ for } \hat{g}' \leftarrow\$ \hat{\mathbb{G}} \tag{16}$$

We next bound the probability $\Pr[\mathbf{G}_3 \Rightarrow 1]$ as follows.

Lemma 5. *Let ε_{DDH} be the advantage of breaking DDH in $\hat{\mathbb{G}}$ as defined in Assumption 1, then $|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \varepsilon_{\text{DDH}} + 1/|\mathbb{F}|$.*

Proof. Observe that, in game $\mathbf{G}_2$, we program the random oracles H_0 and H_1 with a sample from $\mathcal{D}_0$ defined in Lemma 2. Similarly, in game $\mathbf{G}_3$, we program the random oracles H_0 and H_1 exactly with a sample from the distribution $\mathcal{D}_{1,\hat{k}}$ defined in Lemma 3. Apart from the output of the random oracles H_0 and H_1, the rest of the view is identically distributed in $\mathbf{G}_2$ and $\mathbf{G}_3$. Recall from Lemma 3, assuming hardness of DDH in $\hat{\mathbb{G}}$, samples from distributions $\mathcal{D}_0$ and $\mathcal{D}_{1,\hat{k}}$ are computationally indistinguishable. Thus, we get,

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \varepsilon_{\text{DDH}} + \frac{1}{|\mathbb{F}|} \tag{17}$$

GAME $\mathbf{G}_4$: This game is identical to $\mathbf{G}_3$, except that for each honest signer we use simulated NIZK proofs for correctness of partial signatures instead of actual NIZK proofs. Looking ahead, we switch to simulated NIZK proofs in this game to later argue in game $\mathbf{G}_6$ that the NIZK proofs do not reveal any information about the secret signing keys. This is crucial to argue the indistinguishability between game $\mathbf{G}_5$ and $\mathbf{G}_6$.

During the NIZK simulation, the game programs the random oracle H_{FS} on input $(x, y, \mathsf{pk}, \sigma, \hat{g}_0, \hat{g}_1)$ at a choice of its challenge. The game aborts if H_{FS} is already programmed at $(x, y, \mathsf{pk}, \sigma, \hat{g}_0, \hat{g}_1)$. Note that the NIZK protocol we use is perfect honest-verifier zero-knowledge (HVZK). Hence, conditioned on the successful programming of the random oracle H_{FS}, i.e., if the game does not abort, $\mathcal{A}$'s view in games $\mathbf{G}_3$ and $\mathbf{G}_4$ are identically distributed. Next, we will formally analyze the abort probability.

Let E be the event that at least one of our H_{FS} query collides with $\mathcal{A}$'s random oracle query. Then, we have,

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| = |\Pr[\mathbf{G}_3 \Rightarrow 1|E] - \Pr[\mathbf{G}_4 \Rightarrow 1|E]| \cdot \Pr[E] \leq \Pr[E].$$

Here, we use the fact that $|\Pr[\mathbf{G}_3 \Rightarrow 1|E] - \Pr[\mathbf{G}_4 \Rightarrow 1|E]| \leq 1$ and $\Pr[\mathbf{G}_3 \Rightarrow 1|\neg E] = \Pr[\mathbf{G}_4 = 1|\neg E]$.

We now analyze the probability of event E. For each NIZK simulation, the game needs to program H_{FS} at a input $(x, y, \mathsf{pk}, \sigma, \hat{g}_0, \hat{g}_1)$ for some uniformly random $x, y \leftarrow\!\!\$\ \mathbb{G}$. Since $\mathcal{A}$ makes at most q_{FS} queries to the random oracle H_{FS}, the probability that the game aborts during each NIZK simulation is at most $q_{\mathsf{FS}}/|\mathbb{F}|^2$. Since $\mathcal{A}$ makes at most q_s signing queries and we need to simulate at most n partial signatures per signing query, using a simple union bound, we get

$$\Pr[E] \leq \frac{q_{\mathsf{FS}} \cdot q_s \cdot n}{|\mathbb{F}|^2} = \varepsilon_{\text{nizk-fail}}. \tag{18}$$

Hence, we get $|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq \varepsilon_{\text{nizk-fail}}$.

GAME $\mathbf{G}_5$: In this game, we change how we sample the signing keys. To illustrate our modification, we will distinguish between the signing key polynomials of game $\mathbf{G}_4$ and $\mathbf{G}_5$. More precisely, let $s_4(x), r_4(x), u_4(x)$ and $s_5(x), r_5(x), u_5(x)$ be the signing key polynomials in game $\mathbf{G}_4$ and game $\mathbf{G}_5$, respectively. Then, in game $\mathbf{G}_5$ we sample the signing key polynomial $s_5(x) := s_4(x) + \alpha$ where $\alpha = \alpha_h + \alpha_v u$. The other two signing key polynomials remain unchanged, i.e., $r_5(x) = r_4(x)$ and $u_5(x) = u_4(x)$.

Observe that for any fixed α, since $s_4(x)$ is a random degree t polynomial, $s_5(x) = s_4(x) + \alpha$ is also a random degree t polynomial. Hence, $\mathcal{A}$'s view in game $\mathbf{G}_4$ is identical to its view in game $\mathbf{G}_5$, and $\Pr[\mathbf{G}_4 \Rightarrow 1] = \Pr[\mathbf{G}_5 \Rightarrow 1]$.

GAME $\mathbf{G}_6$: In this game, we change how we sample the signing keys again. More precisely, we sample signing key polynomials such that $s_6(x) := s_4(x)$, $r_6(x) := r_4(x) + 1$ and $u_6(x) := u_4(x) + u$, for uniformly random $u \in \mathbb{F}$ we use to define $\alpha = \alpha_h + \alpha_v u$.

The indistinguishability between $\mathcal{A}$'s view in game $\mathbf{G}_5$ and game $\mathbf{G}_6$ is another crucial step of our proof.

Lemma 6. $\Pr[\mathbf{G}_5 \Rightarrow 1] = \Pr[\mathbf{G}_6 \Rightarrow 1]$

Proof. Let $\mathsf{pk}_{\mathbf{G}_5}$ and $\mathsf{pk}_{\mathbf{G}_6}$ are the public keys in game $\mathbf{G}_5$ and $\mathbf{G}_6$, respectively. We first prove that $\mathsf{pk}_{\mathbf{G}_5}$ and $\mathsf{pk}_{\mathbf{G}_6}$ are identically distributed. Note that by design, we have $s_6(0) = s_4(0)$, $r_6(0) = 1$, and $u_6(0) = u$. This implies that,

$$\begin{aligned}\mathsf{pk}_{\mathbf{G}_5} = g^{s_5(0)} h^{r_5(0)} v^{u_5(0)} = g^{s_4(0)+\alpha} &= g^{s_4(0)+\alpha_h+\alpha_v u} \\ &= g^{s_4(0)} h v^u = g^{s_6(0)} h^{r_6(0)} v^{u_6(0)} = \mathsf{pk}_{\mathbf{G}_6}\end{aligned}$$

Next, for any signer i, let $\mathsf{pk}_{i,\mathbf{G}_5}$ and $\mathsf{pk}_{i,\mathbf{G}_6}$ be its threshold public keys in game $\mathbf{G}_5$ and $\mathbf{G}_6$, respectively. Then, since $h = g^{\alpha_h}$ and $v = g^{\alpha_v}$, we have:

$$\begin{aligned}\mathsf{pk}_{i,\mathbf{G}_5} = g^{s_5(i)} h^{r_5(i)} v^{u_5(i)} &= g^{s_4(0)+\alpha_h+\alpha_v u} \cdot h^{r_4(i)} \cdot v^{u_4(i)} \\ &= g^{s_4(i)} \cdot h^{r_4(i)+1} \cdot v^{u_4(i)+u} \\ &= g^{s_6(i)} \cdot h^{r_6(i)} \cdot v^{u_6(i)} = \mathsf{pk}_{i,\mathbf{G}_6}\end{aligned} \tag{19}$$

Similarly, for any signer i, for any message m_k for $k \neq \hat{k}$, let $\sigma_{i,\mathbf{G}_5}$ and $\sigma_{i,\mathbf{G}_6}$ be its partial signatures in $\mathbf{G}_5$ and $\mathbf{G}_6$, respectively. Recall from equation (15), for $k \neq \hat{k}$, we have that:

$$\mathsf{H}_0(m_k) = \hat{g}^{\beta_k} \quad \text{and} \quad \mathsf{H}_1(m_k) = \hat{g}^{\alpha \cdot \beta_k} \text{ for } \alpha = \alpha_h + u\alpha_v \text{ and } \beta_k \leftarrow\!\!\$\ \mathbb{F}$$

This implies that,

$$\begin{aligned}\sigma_{i,\mathbf{G}_5} &= \mathsf{H}_0(m_k)^{s_5(i)}\mathsf{H}_1(m_k)^{r_5(i)} = \mathsf{H}_0(m_k)^{s_4(i)+\alpha}\cdot \mathsf{H}_1(m_k)^{r_4(i)}\\ &= g^{\beta_k\cdot(s_4(i)+\alpha)}\cdot g^{\alpha\beta_k r_4(i)} = g^{\beta_k s_4(i)}\cdot g^{\alpha\beta_k(1+r_4(i))}\\ &= g^{\beta_k s_6(i)}\cdot g^{\alpha\beta_k r_6(i)} = \mathsf{H}_0(m)^{s(i)}\cdot \mathsf{H}_1(m)^{r+r(i)} = \sigma_{i,\mathbf{G}_7} \end{aligned} \tag{20}$$

Equations (19) and (20) imply that the threshold public keys and the partial signatures are identically distributed in games $\mathbf{G}_5$ and $\mathbf{G}_6$. Moreover, the simulated partial signature correctness NIZK proofs reveal no additional information about the signing keys of the honest signers, except what is revealed by the threshold public keys and the partial signatures.

Hence, it remains to show that the joint view of signing keys of the corrupt signers and the set of partial signatures on the forged message $m_{\hat{k}}$ in games $\mathbf{G}_5$ and $\mathbf{G}_6$ are identically distributed. Let $\mathcal{C}$ be the set of corrupt signers. Let $Q[m_{\hat{k}}] \subset \mathcal{H}$ be the subset of honest signers $\mathcal{A}$ queries for partial signatures on the forged message $m_{\hat{k}}$. We have $|Q(m_{\hat{k}}) \cup \mathcal{C}| \leq t$. Also, let $\hat{g}_0 = \mathsf{H}_0(m_{\hat{k}})$ and $\hat{g}_1 = \mathsf{H}_1(m_{\hat{k}})$. Then, for any fixed α, let $\mathcal{D}_5$ and $\mathcal{D}_6$ be the views of $\mathcal{A}$ in game $\mathbf{G}_5$ and $\mathbf{G}_6$, respectively, i.e.,

$$\begin{aligned}\mathcal{D}_5 &= \left(\left\{\hat{g}_0^{s_4(i)+\alpha}\cdot\hat{g}_1^{r_4(i)}\right\}_{i\in Q[m_{\hat{k}}]}, \{s_4(i)+\alpha,\quad r_4(i),\quad u_4(i)\}_{i\in\mathcal{C}}\right),\\ \mathcal{D}_6 &= \left(\left\{\hat{g}_0^{s_4(i)}\cdot\hat{g}_1^{r_4(i)+1}\right\}_{i\in Q[m_{\hat{k}}]}, \{s_4(i),\quad r_4(i)+1,\quad u_4(i)+u\}_{k\in\mathcal{C}}\right)\end{aligned}$$

We argue that $\mathcal{D}_5$ and $\mathcal{D}_6$ are identically distributed based on the following. Consider the following two distributions $\mathcal{D}_{5,t}$ and $\mathcal{D}_{6,t}$ as defined below:

$$\begin{aligned}\mathcal{D}_{5,t} &= \left(\{s_4(i)+\alpha,\quad r_4(k),\quad u_4(k)\}_{k\in\mathcal{C}\cup Q[m_{\hat{k}}]}\right)\\ \mathcal{D}_{6,t} &= \left(\{s_4(k),\quad r_4(k)+1,\quad u_4(k)+u\}_{k\in\mathcal{C}\cup Q[m_{\hat{k}}]}\right)\end{aligned}$$

Observe that the distributions $\mathcal{D}_{5,t}$ and $\mathcal{D}_{6,t}$ are Shamir's secret shares of three secrets using independent random polynomials. Since $|\mathcal{C}\cup Q[m_{\hat{k}}]| \leq t$, the perfect secrecy of Shamir's secret sharing implies that $\mathcal{D}_{5,t}$ and $\mathcal{D}_{6,t}$ are identically distributed. Observe that given a sample from either $\mathcal{D}_{5,t}$ or $\mathcal{D}_{6,t}$, one can efficiently compute a sample from $\mathcal{D}_5$ or $\mathcal{D}_6$, respectively. Hence, $\mathcal{D}_5$ and $\mathcal{D}_6$ are also identically distributed. Therefore, $\mathcal{A}$'s view in $\mathbf{G}_5$ and $\mathbf{G}_6$ are identically distributed, and hence $\Pr[\mathbf{G}_5 \Rightarrow 1] = \Pr[\mathbf{G}_6 \Rightarrow 1]$. □

GAME $\mathbf{G}_7$: This game is identical to $\mathbf{G}_6$, except that we use actual NIZK proofs for partial signatures. We switch back to real proofs in this game because $\mathcal{A}_{\text{co-CDH}}$ in Figure 4 uses real proofs during its interaction with $\mathcal{A}$. Finally, using an argument similar as in the advantage of $\mathcal{A}$ between $\mathbf{G}_4$ and $\mathbf{G}_5$, we get that:

$$|\Pr[\mathbf{G}_6 \Rightarrow 1] - \Pr[\mathbf{G}_7 \Rightarrow 1]| \leq \varepsilon_{\text{nizk-fail}}. \tag{21}$$

Observe that, if game $\mathbf{G}_7$ does not abort, then $\mathcal{A}$'s view in game $\mathbf{G}_7$ is identically distributed as its view in its interaction with $\mathcal{A}_{\text{co-CDH}}$, where $\mathcal{A}_{\text{co-CDH}}$ uses (g^a, g^b) from co-CDH input $(g, g^a, g^b, \hat{g}^a)$ as (h, g^β) in game $\mathbf{G}_7$. Additionally, $\mathcal{A}_{\text{co-CDH}}$ uses $\hat{g}^a$ to compute the random oracle outputs in step 11(b) in Figure 4. Hence, from the above sequence of games, we get that:

$$\begin{aligned}&|\Pr[\mathbf{G}_0 \Rightarrow 1] - \Pr[\mathbf{G}_7 \Rightarrow 1]| \leq \varepsilon_{\text{DDH}} + \frac{1}{|\mathbb{F}|} + 2\varepsilon_{\text{nizk-fail}} + \left(1 - \frac{1}{q_{\mathsf{H}}}\right)\cdot \Pr[\mathbf{G}_0 \Rightarrow 1]\\ &\implies \Pr[\mathbf{G}_7 \Rightarrow 1] \geq \frac{1}{q_{\mathsf{H}}}\cdot\varepsilon_\sigma - \varepsilon_{\text{DDH}} - \frac{1}{|\mathbb{F}|} - 2\varepsilon_{\text{nizk-fail}}.\end{aligned} \tag{22}$$

This implies that if adversary $\mathcal{A}$ outputs a forgery in the $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game of our signature scheme (i.e., $\mathbf{G}_0$) with probability ε_σ, then $\mathcal{A}$ outputs a forgery on $m_{\hat{k}}$ during its interaction with $\mathcal{A}_{\text{co-CDH}}$ (i.e., in $\mathbf{G}_7$) with probability at least $\varepsilon_\sigma/q_{\mathsf{H}} - \varepsilon_{\text{DDH}} - 1/|\mathbb{F}| - 2\varepsilon_{\text{nizk-fail}}$. Moreover, Lemma 4 implies that $\mathcal{A}_{\text{co-CDH}}$ can efficiently compute the co-CDH solution using the forgery on $m_{\hat{k}}$. Combining all the above, we get our main theorem, as stated below.

Theorem 4 (Adaptively secure BLS threshold signature). *Let λ be the security parameter, and let $(\mathbb{F}, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p) \leftarrow \mathsf{GGen}(1^\lambda)$ be pairing groups of prime order p. For any n, t for $n = \mathsf{poly}(\lambda)$ and $t < n$, assuming hardness of decisional Diffie-Hellman (DDH) in $\hat{\mathbb{G}}$, and hardness of co-computational Diffie-Hellman (co-CDH) in $(\mathbb{G}, \hat{\mathbb{G}})$ in the random oracle model, any PPT adversary making at most q_{H} random oracle queries to H_0 and H_1 combined, q_{FS} queries to the random oracle H_{FS}, and at most q_s signature queries wins the $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game Figure 1 for our scheme in Figure 2 with probability at most ε_σ where:*

$$\varepsilon_\sigma \leq q_{\mathsf{H}} \cdot \left(\varepsilon_{\mathrm{DDH}} + \frac{1}{|\mathbb{F}|} + 2\varepsilon_{\text{nizk-fail}} + \varepsilon_{\mathrm{CDH}} \right),$$

$\varepsilon_{\text{nizk-fail}} = (q_{\mathsf{FS}} \cdot q_s \cdot n)/|\mathbb{F}|^2$, and $\varepsilon_{\mathrm{DDH}}$ and $\varepsilon_{\mathrm{CDH}}$ are the advantages of an adversary running in $T \cdot \mathsf{poly}(\lambda, n)$ time in breaking DDH in $\hat{\mathbb{G}}$ and co-CDH in $(\mathbb{G}, \hat{\mathbb{G}})$, respectively. This implies that ε_σ is negligible, and hence, our threshold signature scheme in §5 is unforgeable.

Remark. Note that the unforgeability property of our threshold signature scheme does not rely on the *soundness* property of the Σ-protocol signers use to prove the correctness of the partial signatures. We only rely on the knowledge-soundness property to achieve robustness of our scheme (see §6.2).

6.5 Unforgeability with static adversary

We now briefly argue that if we are content with proving our signature scheme statically secure, then we only need the hardness of CDH assumption in a pairing group $(\mathbb{G}, \hat{\mathbb{G}})$ in the ROM. For static security, we do not require asymmetric pairing groups. Thus, we will assume $\mathbb{G} = \hat{\mathbb{G}}$ in this analysis, and hence the CDH assumption instead of co-CDH. Moreover, we will only consider the TS-UF-0 threat model from [BCK+22]. Our security proof is similar to the static security proof of Boldyreva's scheme. We want to note that assuming the hardness of CDH in the random oracle model, Boldyreva's scheme has only been proven TS-UF-0 secure. We adopt TS-UF-0 for simplicity since static security is not the main focus of the paper.

Let $\mathcal{A}_{\text{static}}$ be the static adversary that breaks the unforgeability of our signature scheme, and let $\mathcal{A}_{\mathrm{CDH}}$ be the CDH adversary. Let $\mathcal{C}$ be the set of signers $\mathcal{A}_{\text{static}}$ corrupts at the beginning of the protocol, and $\mathcal{H} = [n] \setminus \mathcal{C}$ be the set of honest signers. Also, let $\mathcal{S} \subset \mathcal{H}$ be the subset of honest signers $\mathcal{A}_{\text{static}}$ will query for partial signatures on the forged message. By the definition of a static adversary, we require that $|\mathcal{C} \cup \mathcal{S}| \leq t$ and $\mathcal{A}_{\text{static}}$ declare the sets $\mathcal{C}, \mathcal{S}$ to $\mathcal{A}_{\mathrm{CDH}}$. $\mathcal{A}_{\mathrm{CDH}}$ on input a CDH input $(g, g^a, g^b) \in \mathbb{G}^3$ simulates the KGen functionality and the signature scheme with $\mathcal{A}_{\text{static}}$ as follows.

Simulating the KGen functionality. For simplicity, let us assume $|\mathcal{C} \cup \mathcal{S}| = t$. $\mathcal{A}_{\mathrm{CDH}}$ samples $h, v \leftarrow\$ \mathbb{G}$. Next, $\mathcal{A}_{\mathrm{CDH}}$ samples two random degree t polynomials $r(x), u(x)$ with the constraint $r(0) = u(0) = 0$. To compute the polynomial $s(x)$, $\mathcal{A}_{\text{static}}$ samples $s(j) \leftarrow\$ \mathbb{F}$ for each $j \in \mathcal{C} \cup \mathcal{S}$. $\mathcal{A}_{\mathrm{CDH}}$ sets the public key as $\mathsf{pk} = g^a$ and computes threshold public keys $\{\mathsf{pk}_i\} = \{g^{s(i)} h^{r(i)} v^{u(i)}\}_{i \in [n]}$ using interpolation in the exponent. $\mathcal{A}_{\mathrm{CDH}}$ then sends $\mathsf{pk}, \{\mathsf{pk}_i\}_{i \in [n]}, \{\mathsf{sk}_i\}_{i \in \mathcal{C}}$ to $\mathcal{A}_{\text{static}}$.

Simulating the signing queries. Throughout the simulation $\mathcal{A}_{\mathrm{CDH}}$ always faithfully responds to queries to H_1. Note that H_0 is always queried on the forged message, at least by $\mathcal{A}_{\mathrm{CDH}}$ during the signature verification. Let q_{H} be an upper bound on the number of random oracle queries to H_0, including the query during the signature verification. For static security, the number of queries to H_1 can be unbounded. $\mathcal{A}_{\mathrm{CDH}}$ samples $\hat{k} \leftarrow\$ [q_{\mathsf{H}}]$. On the k-th random oracle query on message m_k, depending upon the value of k, $\mathcal{A}_{\mathrm{CDH}}$ programs the random oracle as follows:

$$k \neq \hat{k} \implies \mathsf{H}_0(m_k) = g^{\beta_k} \text{ for } \beta_k \leftarrow\$ \mathbb{F}; \quad \text{and} \quad k = \hat{k} \implies \mathsf{H}_0(m_k) = g^b;$$

Let q_s be the maximum number of signing queries made by $\mathcal{A}_{\text{static}}$. We have $q_s \leq q_{\mathsf{H}}$. Then, whenever $k \neq \hat{k}$, $\mathcal{A}_{\mathrm{CDH}}$ uses its knowledge of β_k and polynomial $r(\cdot)$ to respond to partial signing queries correctly. Alternatively, when $k = \hat{k}$ and let $m_{\hat{k}}$ be the corresponding message, $\mathcal{A}_{\mathrm{CDH}}$ correctly responds to partial signing queries for each signer $j \in \mathcal{C} \cup \mathcal{S}$, using its knowledge of $s(j)$. If $\mathcal{A}_{\text{static}}$ queries for partial signatures on $m_{\hat{k}}$ from signers not in $\mathcal{C} \cup \mathcal{S}$, $\mathcal{A}_{\mathrm{CDH}}$ aborts.

Now, whenever $\mathcal{A}_{\text{static}}$ outputs a valid forgery $(m_{\hat{k}}, \sigma^*)$, $\mathcal{A}_{\mathrm{CDH}}$ outputs σ^* as the CDH solution. It is easy to see that $\sigma^* = g^{ab}$.

7 Distributed Key Generation (DKG) Definitions and Analysis

In our discussion so far, we proved the adaptive security of our threshold signature scheme assuming a trusted party generates the signing keys. In this section, we present a distributed key generation (DKG) protocol that signers can run to set up the signing keys of our threshold signature scheme instead of relying on the trusted party. DKG has the following interface.

DKG(): For any (n, t) non-interactive threshold signature scheme TS with $t < n/2$, DKG is an interactive protocol among n parties, which all take some public parameters as inputs. At the end of the protocol, signers output a public key pk, a vector of threshold public keys $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$. Each signer i additionally outputs a secret key share sk_i.

As in §5, concretely, in our DKG protocol, $\mathsf{sk}_i = (s(i), r(i), u(i))$, $\{\mathsf{pk}_j = g^{s(j)}h^{r(j)}v^{u(j)}\}_{j\in[n]}$, and the public verification key $\mathsf{pk} = g^{s(0)}h^{r(0)}v^{u(0)}$. Here, $s(\cdot), r(\cdot)$ and $u(\cdot)$ are three degree t polynomials with $r(0) = 0$ and $u(0) = 0$. This implies that $\mathsf{pk} = g^{s(0)}$.

First, we require the DKG protocol to satisfy *robustness*, which states that the keys output by the DKG protocol are well-formed, even in the presence of an adaptive adversary that can corrupt up to t out of n signers. More formally,

Definition 5 (DKG Robustness). *A DKG protocol* DKG *for any* (n, t) *non-interactive threshold signature scheme* TS *with* $t < n/2$ *is robust, if for all security parameters* λ *and all PPT adversary* $\mathcal{A}$ *that can adaptively corrupt up to* t *parties during the DKG protocol, the following holds:*

$$\Pr\left[\begin{array}{r|l} \exists\ s(x), r(x), u(x) \in \mathbb{F}[x]^t\ s.t. & \\ r(0) = u(0) = 0 \wedge \mathsf{pk} = g^{s(0)} \wedge & \\ \mathsf{sk}_i = (s(i), r(i), u(i)), & \mathsf{pk}, \{\mathsf{pk}_i, \mathsf{sk}_i\}_{i\in[n]} \leftarrow \mathsf{DKG}(n, t) \\ \mathsf{pk}_i = g^{s(i)}h^{r(i)}v^{u(i)}, \forall i \in [n] & \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Here, the probability is over the choice of the randomness of both $\mathcal{A}$ *and the honest parties.*

We also require the DKG protocol to satisfy the *single inconsistent party (SIP) simulatability*. Recall that the security proof of our threshold signature used a rigged public key with $r(0) = 1$ and uniformly random $u(0)$. However, with DKG, we do not have a trusted entity to set up the rigged public key. Instead, we will rely on the SIP technique [CGJ+99, FMY99a, FMY99b] to set up a rigged public key. In more detail, we will let one honest party deviate from the specified DKG protocol so that the final DKG output has the rigged structure we need. For this to go through, we need to ensure that $\mathcal{A}$ cannot distinguish between the execution with a single inconsistent party and the real execution of the protocol where all parties are honest. The SIP simulatability property below captures this.

Definition 6 (SIP Simulatability). *For security parameter* λ*, for all* (n, t) *with* $t < n/2$ *and all PPT adversary* $\mathcal{A}$ *that adaptively corrupts up to* t *parties, let* $\mathcal{S}_{\mathsf{DKG}}$ *be an efficient simulator that runs a DKG protocol with* $\mathcal{A}$ *with a single inconsistent party (SIP) such that the DKG output is rigged. A DKG protocol is SIP simulatable if* $\mathcal{A}$*'s view* $\mathsf{View}^{\mathcal{A}}_{\text{real}}$ *of the real protocol execution is indistinguishable from its view* $\mathsf{View}^{\mathcal{A}}_{\text{sim}}$ *in its interaction with* $\mathcal{S}_{\mathsf{DKG}}$*.*

We remark that the precise notion of *rigged* can vary depending upon the application. For our purpose, we require the simulated protocol to output a public key $\mathsf{pk} = g^s h v^u$ for $s, u \leftarrow\$\ \mathbb{F}$.

7.1 Design of our DKG protocol

We design our DKG protocol by augmenting the classic Pedersen DKG protocol, also referred to as the JF-DKG protocol [GJKR07]. We pick JF-DKG due to its simplicity and popularity. We believe we can use many other DKG protocols using a similar modification (see our discussion at the end of this section). We summarize our protocol in Figure 5 and describe it next.

Public parameters: $(g, h, v) \in \mathbb{G}^3, \mathbb{F}$

Sharing phase:

1. Each party i (as a dealer) chooses random polynomials $s_i(x), r_i(x)$ and $u_i(x)$ over $\mathbb{F}$ of degree t each, where

$$s_i(x) := s_{i,0} + s_{i,1}x + \cdots + s_{i,t}x^t; \qquad r_i(x) := r_{i,1}x + \cdots + r_{i,t}x^t; \qquad u_i(x) := u_{i,1}x + \cdots + u_{i,t}x^t \tag{23}$$

2. Party i computes $\mathsf{cm}_i := [g^{s_0}, g^{s_1}h^{r_1}v^{u_1}, \ldots g^{s_t}h^{r_t}v^{u_t}]$.
3. Party i computes π_i, the NIZK proof of knowledge of $s_{i,0}$ with respect to $g^{s_{i,0}}$.
4. Party i broadcasts (cm_i, π_i) to all.
5. Party i privately sends $s_i(j), r_i(j), u_i(j)$ to party j.

Agreement phase:

6. Each party j verifies the shares it receives from other parties by checking for $i = 1, \ldots, n$:

$$g^{s_i(j)}h^{r_i(j)}v^{u_i(j)} = \prod_{k\in[0,t]} \mathsf{cm}_i[k]^{j^k} \tag{24}$$

7. If the check fails for an index i, party j broadcasts a complaint against P_i
8. Party i (as a dealer) reveals $s_i(j), r_i(j), u_i(j)$ matching eq. (24). If any of the revealed shares fails this equation, party i is disqualified. Let $\mathcal{Q}$ be the set of non-disqualified parties.

Key-derivation phase:

9. The public key pk is computed as $\mathsf{pk} := \prod_{i\in\mathcal{Q}} \mathsf{cm}_i[0]$. The threshold public keys pk_j for each $j \in [n]$ are computed as:

$$\mathsf{pk}_j := \prod_{i\in\mathcal{Q}} \prod_{k\in[0,t]} \mathsf{cm}_i[k]^{j^k} \tag{25}$$

10. Each party j sets its signing key as $\mathsf{sk}_j := (\sum_{i\in\mathcal{Q}} s_i(j), \sum_{i\in\mathcal{Q}} r_i(j), \sum_{i\in\mathcal{Q}} u_i(j))$.
11. The shared secret key is $s =: \sum_{i\in\mathcal{Q}} s_i$.

Fig. 5: Our DKG protocol which is a modification of the JF-DKG [GJKR07].

Let $g, h, v \in \mathbb{G}$ be three uniformly random generators of $\mathbb{G}$ with a scalar field $\mathbb{F}$. We will describe our DKG protocol in three phases: *Sharing, Agreement* and *Key Derivation*.

Sharing phase. During the sharing phase, each party i, as a verifiable secret sharing (VSS) dealer, samples three random degree-t polynomials $s_i(x), r_i(x), u_i(x)$ with $r_i(0) = u_i(0) = 0$ such that

$$s_i(x) := s_{i,0} + s_{i,1}x + \cdots + s_{i,t}x^t; \qquad r_i(x) := r_{i,1}x + \cdots + r_{i,t}x^t; \qquad u_i(x) := u_{i,1}x + \cdots + u_{i,t}x^t$$

Party i then computes a commitment $\mathsf{cm}_i \in \mathbb{G}^{t+1}$ to these polynomials

$$\mathsf{cm}_i := [g^{s_{i,0}}, g^{s_{i,1}}h^{r_{i,1}}v^{u_{i,1}}, \cdots, g^{s_{i,t}}h^{r_{i,t}}v^{u_{i,t}}] \tag{26}$$

and a proof of knowledge π of discrete logarithm of $\mathsf{cm}_i[0] = g^{s_{i,0}}$ with respect to the generator g using the Schnorr identification scheme [Sch90] (steps 2 and 3).

Party i then publishes (cm_i, π_i) (step 4) using a broadcast channel. Intuitively, the proof π_i ensures that the constant terms of $r_i(x)$ and $u_i(x)$ are zero, except with a negligible probability. Also, party i sends each party j, via a private channel, the tuple $(s_i(j), r_i(j), u_i(j))$.

Agreement phase. The purpose of the agreement phase is for parties to agree on a subset of dealers, also referred to as the qualified set, who correctly participated in the sharing phase. To agree on the qualified set, each party j, upon receiving from dealer i the tuple (s', r', u') (via the private channel) and (cm_i, π_i) (via the broadcast channel), accepts them as valid shares if π_i is a valid proof and the following holds:

$$g^{s'}h^{r'}v^{u'} = \prod_{k\in[0,t]} \mathsf{cm}_i[k]^{j^k} \tag{27}$$

If either of the validation checks fails for any dealer i, the party broadcasts a complaint against the dealer i (step 7). The dealer i then responds to all the complaints against it by publishing the shares of all the

Input: co-CDH tuple $(g, g^a, \hat{g}, \hat{g}^a, \hat{g}^b) \in \mathbb{G}^3 \times \hat{\mathbb{G}}$.

DKG simulation:

1. Sample $\alpha_v \leftarrow\$ \mathbb{F}$. Let $h := g^a$ and $v := g^{\alpha_v}$.
2. Let $\mathcal{C}$ and $\mathcal{H}$ be the set of honest and malicious parties, respectively. Sample $\hat{i} \leftarrow\$ \mathcal{H}$, and let $\mathcal{H} := \mathcal{H} \setminus \{\hat{i}\}$.
3. For each honest party $i \in \mathcal{H}$, maintain its internal state st_i of the DKG protocol. The internal state consists of the private messages party i receives and secret values party i samples throughout the DKG protocol.
4. On behalf of each $i \in \mathcal{H}_{-\hat{i}}$, run the DKG protocol as per the specification.
5. Let $u \leftarrow\$ \mathbb{F}$. On behalf of $\hat{i}$, sample three degree t uniformly random polynomial $s_{\hat{i}}(x), r_{\hat{i}}(x), u_{\hat{i}}(x) \in \mathbb{F}[x]$ with the constraint that $r_{\hat{i}}(0) = 1$ and $u(0) = u$. Compute the commitment $\mathsf{cm}_{\hat{i}}$ as:
$$\mathsf{cm}_{\hat{i}} := [g^{s_{\hat{i},0}} h^{r_{\hat{i},0}} v^{u_{\hat{i},0}}, g^{s_{\hat{i},1}} h^{r_{\hat{i},1}} v^{u_{\hat{i},1}}, \ldots, g^{s_{\hat{i},t}} h^{r_{\hat{i},t}} v^{u_{\hat{i},t}}] \tag{29}$$
here $s_{\hat{i},j}, r_{\hat{i},j}$, and $u_{\hat{i},j}$ for each $j \in [0,t]$ is are the coefficient of x^j in $s_{\hat{i}}(x), r_{\hat{i}}(x)$, and $u_{\hat{i}}(x)$, respectively.
6. Compute $\pi_{\hat{i}}$, the proof of knowledge of exponent $\mathsf{cm}_{\hat{i}}[0]$ with respect to g, by running the NIZK simulator.
7. For the of rest of the DKG protocol, follow the honest protocol specification on behalf of every honest party.
8. Let $\mathcal{Q}$ be the qualified set of parties in the DKG protocol. Note that, since we are working in a synchronous network, $\mathcal{H} \subseteq \mathcal{Q}$ and hence $\hat{i} \in \mathcal{Q}$.

Corruption simulation.

8. When $\mathcal{A}$ corrupts a signer $i \in \mathcal{H}$:
   (a) If $i = \hat{i}$, rewind $\mathcal{A}$ to step 2 and rerun.
   (b) Otherwise, send $(\mathsf{st}_i, \mathsf{sk}_i)$, the internal state and the signing key of party i to $\mathcal{A}$, and update $\mathcal{H} := \mathcal{H} \setminus \{i\}$ and $\mathcal{C} := \mathcal{C} \cup \{i\}$.

Fig. 6: Simulator $\mathcal{S}_{\mathsf{DKG}}$ for our DKG protocol in Figure 5.

complaining parties. All parties then locally validate all the revealed shares for all the complaints. If any dealer i publishes an invalid response to any complaint or does not respond at all, then dealer i is disqualified (step 8). Let $\mathcal{Q}$ be the set of qualified dealers. Note that all honest parties will always be part of $\mathcal{Q}$.

Key-derivation phase. With a qualified set $\mathcal{Q}$, the final public key is $\mathsf{pk} = \prod_{i \in Q} \mathsf{cm}_i[0]$. The threshold public key pk_j of every party j is computed as in equation (25). The signing key sk_j of each party j is the sum of the j-th share of all dealers in $\mathcal{Q}$ as shown in step 10 of Figure 5. Let $s(x), r(x), u(x)$ be the polynomials defined as:
$$s(x) := \sum_{i \in Q} s_i(x); \qquad r(x) := \sum_{i \in Q} r_i(x); \qquad u(x) := \sum_{i \in Q} u_i(x). \tag{28}$$
Once the DKG protocol finishes, each party i outputs its signing key $\mathsf{sk}_i := (s(i), r(i), u(i))$, the public key $\mathsf{pk} := g^{s(0)} h^{r(0)} v^{u(0)} = g^{s(0)}$, and the threshold public keys $\{\mathsf{pk}_j := g^{s(j)} h^{r(j)} v^{u(j)}\}_{j \in [n]}$.

Using other DKG protocols. In Figure 5, we augment the JF-DKG protocol for our signature scheme. Our augmentation techniques are generic and can be used with many existing DKG protocols that follow the same three-phase structure [Ped91, CGJ+99, CS04, FS01, GJKR07, Gro21, KHG12, KKMS20, DYX+22]. Specifically, we can augment any such DKG protocol by having each VSS dealer: (i) share two additional zero-polynomials $r(\cdot)$ and $u(\cdot)$; and (ii) publish a NIZK proof π for the correctness of the zero-polynomial. Similarly, each VSS recipient will validate the shares it receives with the updated check in Figure 5.

7.2 Analysis of our DKG

We now prove that our DKG protocol satisfies the *robustness* and *SIP simulatibility* properties we define.

Lemma 7 (Robustness). *Assuming the hardness of discrete logarithm in $\mathbb{G}$, our DKG protocol in Figure 5, is robust as per Definition 5.*

Proof. An argument similar to the correctness analysis of [GJKR07, Theorem 1] ensures that assuming hardness of discrete logarithm in $\mathbb{G}$, individual signing keys $\mathsf{sk}_i = (s(i), r(i), u(i))$ output by honest parties

lies on some degree t polynomials $s(x), r(x)$ and $u(x)$. We will now argue that assuming hardness discrete logarithm in $\mathbb{G}$, $r(0) = u(0) = 0$.

Let $\mathcal{A}_{\mathrm{DL}}$ be the discrete logarithm adversary. On input a discrete logarithm instance $(g, y) \in \mathbb{G}^2$, $\mathcal{A}_{\mathrm{DL}}$ samples $\theta \in \{0, 1\}$ and sets either $h = y$ or $v = y$ depending on the value of θ. $\mathcal{A}_{\mathrm{DL}}$ picks the other parameter as g^{α} for some known $\alpha \leftarrow\$ \mathbb{F}$. Without loss of generality, let $h = g^{\alpha_h}$ and $v = g^{\alpha_v}$ for some $(\alpha_h, \alpha_v) \in \mathbb{F}^2$. $\mathcal{A}_{\mathrm{DL}}$ next faithfully runs the DKG protocol with $\mathcal{A}$.

Let $\mathcal{C} \subset [n]$ be the set of corrupt parties and $\mathcal{Q}$ be the set of qualified parties. For each $i \in \mathcal{C} \cap \mathcal{Q}$, let $s_i(x), r_i(x)$ and $u_i(x)$ be the degree t polynomials shared by party i. Let $s_i := s_i(0), r_i := r_i(0)$ and $u_i := u_i(0)$. Then, we will prove that assuming the hardness of discrete logarithm, for all parties $i \in \mathcal{C} \cap \mathcal{Q}$, $(r_i, u_i) = (0, 0)$ except with a negligible probability.

For the sake of contradiction, assuming it is not the case, i.e., there exists an $i \in \mathcal{C} \cap \mathcal{Q}$ such that $(r_i, u_i) \neq (0, 0)$. Then, $\mathcal{A}_{\mathrm{DL}}$ solves the discrete logarithm for y as follows. Let cm_i be the commitment vector output by party i, along with a NIZK proof-of-knowledge π_i. $\mathcal{A}_{\mathrm{DL}}$ then extracts the witness w such that $g^w = \mathsf{cm}_i[0]$ from $\mathcal{A}$ using the NIZK proof-of-knowledge extractor. Also, let $s_i(x), r_i(x), u_i(x)$ be the polynomial party i shares during the DKG protocol. Then, since $n - t > t$, $\mathcal{A}_{\mathrm{DL}}$ can extract $s_i(x), r_i(x), u_i(x)$ using a straight-line extractor. Let $s_i := s_i(0), r_i := r_i(0)$ and $u_i := u_i(0)$. Then, since $i \in \mathcal{Q}$, it is also the case that $\mathsf{cm}_i[0] = g^{s_i} h^{r_i} v^{u_i}$. Therefore,

$$w = s_i + \alpha_h r_i + \alpha_v u_i. \tag{30}$$

Now, depending on whether $r_i \neq 0$ or $u_i \neq 0$, $\mathcal{A}_{\mathrm{DL}}$ computes either α_h or α_v as:

$$r_i \neq 0 \implies \alpha_h = (w - s_i - u_i \alpha_v) \cdot r_i^{-1}; \qquad u_i \neq 0 \implies \alpha_v = (w - s_i - r_i \alpha_h) \cdot u_i^{-1} \tag{31}$$

We will now analyze the success probability of $\mathcal{A}_{\mathrm{DL}}$ assuming parties use the Schnorr identification scheme in $\mathbb{G}$ with Fiat-Shamir heuristic as the proof-of-knowledge protocol [Sch90, FS86]. Let $\mathsf{H}_{\mathsf{pok}}$ be the random oracle used in the proof-of-knowledge protocol and let q_{pok} be the upper bound on the number of queries $\mathcal{A}$ makes to the random oracle $\mathsf{H}_{\mathsf{pok}}$. This implies that, from generalized forking lemma [BN06], if $\mathcal{A}$ outputs an accepting proof π_i for a party $i \in \mathcal{C} \cap \mathcal{Q}$ with probability ε, then $\mathcal{A}_{\mathrm{DL}}$ successfully extracts w with probability at least $\varepsilon_{\mathsf{ext}} := \varepsilon^2 / q_{\mathsf{pok}} - \varepsilon / |\mathbb{F}|$.

Since $\mathcal{A}_{\mathrm{DL}}$ uses y as either h or v uniformly at random, it implies that if $\mathcal{A}$ successfully uses $(r_i, u_i) \neq (0, 0)$ with probability ε, then $\mathcal{A}_{\mathrm{DL}}$ outputs the discrete logarithm of y with respect to g, with probability at least $\varepsilon_{\mathsf{ext}}/2$. Hence, assuming the hardness of discrete logarithm, $r(0) = u(0) = 0$.

Lemma 8 (SIP Simulatability). *The DKG protocol in Figure 5 is SIP simulatable as per Definition 6.*

Proof. We will prove this via a sequence of games, where game $\mathbf{G}_0$ is the real protocol execution and $\mathbf{G}_4$ is the simulated protocol.

GAME $\mathbf{G}_0$: This is the real execution of our DKG protocol. Hence, $\mathcal{A}$'s view in this game is $\mathsf{View}^{\mathcal{A}}_{\mathrm{real}}$.

GAME $\mathbf{G}_1$: This game is identical to $\mathbf{G}_0$, except that we sample $\alpha_h, \alpha_v \leftarrow\$ \mathbb{F}$ and set $h := g^{\alpha_h}$ and $v := g^{\alpha_v}$. Clearly, the view of $\mathcal{A}$ in $\mathbf{G}_0$ is identical to its view in $\mathbf{G}_1$.

GAME $\mathbf{G}_2$: This game is identical to $\mathbf{G}_1$, except that we compute the proof-of-knowledge $\pi_{\hat{i}}$ for party $\hat{i}$ using the NIZK simulator. Note that the statement we want to simulate is independent of $\mathcal{A}$. Hence, we can compute the simulated proof before $\mathcal{A}$ makes any random oracle query. This implies that we can output the simulated NIZK proof, even if our random oracle query collides with $\mathcal{A}$'s random oracle queries. Combining this with the fact that Σ-protocols are perfect HVZK, we get that $\mathcal{A}$'s view in game $\mathbf{G}_1$ is identical to its view in game $\mathbf{G}_2$.

GAME $\mathbf{G}_3$: In this game, we change how we sample the secrets of party $\hat{i}$ during the sharing phase. To illustrate this change, we will distinguish between the polynomials party $\hat{i}$ shares in game $\mathbf{G}_2$ and $\mathbf{G}_3$. More precisely, let $(s_{\hat{i},2}(x), r_{\hat{i},2}(x), u_{\hat{i},2}(x))$ and $(s_{\hat{i},3}(x), r_{\hat{i},3}(x), u_{\hat{i},3}(x))$ be the polynomials party $\hat{i}$ shares during the sharing phase of the DKG protocols in game $\mathbf{G}_2$ and $\mathbf{G}_3$, respectively.

Let $\alpha := \alpha_h + \alpha_v u$ for some $u \leftarrow\$ \mathbb{F}$. Then, in game $\mathbf{G}_3$, party $\hat{i}$ shares polynomials such that:

$$s_{\hat{i},3}(x) := s_{\hat{i},2}(x) + \alpha; \qquad r_{\hat{i},3}(x) := r_{\hat{i},2}(x); \qquad u_{\hat{i},3}(x) := u_{\hat{i},2}(x)$$

Note for any fixed α, since $s_{\hat{i},2}(x)$ is a uniformly random polynomial of degree t, $s_{\hat{i},3}(x)$ is also uniformly random. Hence, $\mathcal{A}$'s view in game $\mathbf{G}_3$ is identical to its view in game $\mathbf{G}_2$.

GAME $\mathbf{G}_4$: In this game, we change how party $\hat{i}$ samples its secret polynomials again. More specifically, now party $\hat{i}$ sample secret key polynomials as below:

$$s_{\hat{i},4}(x) := s_{\hat{i},2}(x); \qquad r_{\hat{i},4}(x) := r_{\hat{i},2}(x) + 1; \qquad u_{\hat{i},4}(x) := u_{\hat{i},2}(x) + u \tag{32}$$

Based on a similar argument as Lemma 6, $\mathcal{A}$'s view in game $\mathbf{G}_4$ is identically distributed as its view in $\mathbf{G}_3$.

Note the view of $\mathcal{A}$ in game $\mathbf{G}_4$ identically distributed as its view in its interaction with $\mathcal{S}_{\mathsf{DKG}}$, and hence $\mathcal{A}$'s view in $\mathbf{G}_4$ is $\mathsf{View}^{\mathcal{A}}_{\text{sim}}$. This implies that $\mathsf{View}^{\mathcal{A}}_{\text{real}}$ and $\mathsf{View}^{\mathcal{A}}_{\text{sim}}$ are identically distributed. □

Running time of $\mathcal{S}_{\mathsf{DKG}}$. It is easy to see that, during each iteration of the simulation, $\mathcal{S}_{\mathsf{DKG}}$ runs in polynomial time. We will now argue that $\mathcal{S}_{\mathsf{DKG}}$ runs for more than λ iterations with probability at most $2^{-\lambda}$.

During every iteration of the simulation, all honest dealers (including the SIP) will be part of the qualified set, and the probability that $\mathcal{A}$ does not corrupt the SIP, i.e., $\hat{i} \notin \mathcal{C}$ is at least:

$$\Pr[\hat{i} \notin \mathcal{C}] \geq \frac{\binom{n-1}{t}}{\binom{n}{t}} = \frac{n-t}{n} \geq 1/2 \tag{33}$$

where the last inequality follows from the fact that $n > 2t$.

Equation (33) implies that in each iteration $\mathcal{S}_{\mathsf{DKG}}$ needs to rewind $\mathcal{A}$ with probability at most 1/2. This implies that $\mathcal{S}_{\mathsf{DKG}}$ runs for more than λ iterations with probability at most $2^{-\lambda}$.

Lemma 9 (Rigged Public Key). *Let $s(x), r(x), u(x)$ be the signing key polynomials in the simulated DKG protocol as per Figure 6. Then, assuming hardness of discrete-logarithm (DL) in $\mathbb{G}$, $(r(0), u(0)) = (1, u)$ and $\mathsf{pk} = g^s h v^u$, for s and u as defined in Figure 6.*

Proof. In Lemma 7, we prove that assuming hardness of DL in $\mathbb{G}$, in the real execution of the DKG protocol, for every party $j \in \mathcal{C} \cap \mathcal{Q}$, $(r_j(0), u_j(0)) = (0, 0)$. Moreover, Definition 6 implies that $\mathsf{View}^{\mathcal{A}}_{\text{real}}$ and $\mathsf{View}^{\mathcal{A}}_{\text{sim}}$ are identically distributed. Hence, we get that for every $j \in \mathcal{C}$, $(r_j(0), u_j(0)) = (0, 0)$ even in the simulated protocol. Lastly, since $\hat{i} \in \mathcal{Q}$, we get that $(r(0), u(0)) = (1, u)$, and $\mathsf{pk} = g^s h v^u$ for some $s = s(0)$. □

7.3 Signature Scheme with DKG.

Our threshold signature scheme with a DKG protocol is identical to Figure 2, except that signers generate their signing keys by running the DKG protocol in Figure 5.

8 Proof of Adaptive Security with DKG

The robustness of our threshold signature scheme with DKG follows from the DKG robustness (see Lemma 7) and an argument similar to the proof of Theorem 3. We focus on unforgeability next.

Similar to §6, we prove the unforgeability assuming the hardness of the DDH in $\hat{\mathbb{G}}$ and the hardness of co-CDH in $(\mathbb{G}, \hat{\mathbb{G}})$. Let $\mathcal{A}_{\text{co-CDH}}$ be the reduction adversary. On input a co-CDH instance $(g, \hat{g}, g^a, \hat{g}^a, \hat{g}^b)$, $\mathcal{A}_{\text{co-CDH}}$ simulates the DKG and threshold signature protocol for a PPT adversary $\mathcal{A}$, such that when $\mathcal{A}$ forges a signature, $\mathcal{A}_{\text{co-CDH}}$ uses the forgery to compute $\hat{g}^{ab}$. As we mentioned earlier, our security reduction will use the *single inconsistent party* (SIP) technique [CGJ+99, FMY99a, FMY99b] where there exists only one signer whose internal state cannot be consistently revealed to $\mathcal{A}$. We summarize $\mathcal{A}_{\text{co-CDH}}$'s interaction with $\mathcal{A}$ in Figure 7 and describe it next.

Input: co-CDH tuple $(g, g^a, \hat{g}, \hat{g}^a, \hat{g}^b) \in \mathbb{G}^3 \times \hat{\mathbb{G}}$.

DKG simulation: // We use the notations from Figure 6.

1. Run the DKG simulator $\mathcal{S}_{\mathsf{DKG}}$. Let $\hat{i} \in \mathcal{H}$ be the SIP, and let $r_{\hat{i}}(x)$ and $u_{\hat{i}}(x)$ be the polynomial $\hat{i}$ chooses during the DKG simulation, with $r_{\hat{i}}(0) = 1$ and $u_{\hat{i}}(0) = u$ for some uniformly random $u \leftarrow\$ \mathbb{F}$.
2. Let $\mathcal{Q}$ be the resulting qualified set as a result of simulating the DKG. Let $s(x), r(x)$, and $u(x)$ be degree t polynomials where:
$$s(x) = \sum_{i \in \mathcal{Q}} s_i(x); \quad r(x) = \sum_{i \in \mathcal{Q}} r_i(x); \quad u(x) = \sum_{i \in \mathcal{Q}} u_i(x); \tag{34}$$
Note that by design, the resulting keys of party i are $\mathsf{sk}_i = (s(i), r(i), u(i))$ and $\mathsf{pk}_i = g^{s(i)} h^{r(i)} v^{u(i)}$. Moreover, lemma 9 implies that $(r(0), u(0)) = (1, u)$ and $\mathsf{pk} = g^s h v^u$ where $s = s(0)$.

Corruption simulation:

3. When $\mathcal{A}$ corrupts a signer $i \in \mathcal{H}$.
   (a) If $i = \hat{i}$, rewind $\mathcal{A}$ to step 1 and rerun.
   (b) Otherwise, send $(\mathsf{st}_i, \mathsf{sk}_i)$, the internal state from the DKG protocol and the signing key of party i to $\mathcal{A}$, and update $\mathcal{H} := \mathcal{H} \setminus \{i\}$ and $\mathcal{C} := \mathcal{C} \cup \{i\}$.

Threshold signature simulation:

// Identical to the threshold signature simulation in steps 7 to 13 of Figure 4.

Compute co-CDH output:

4. Let $(m_{\hat{k}}, \sigma)$ be the valid forgery $\mathcal{A}$ outputs. Output $\sigma \cdot \hat{g}^{-b(s + \alpha_v u)}$ as the co-CDH solution.

Fig. 7: $\mathcal{A}_{\text{co-CDH}}$'s interaction with $\mathcal{A}$ to compute the co-CDH solution, when signers use the DKG protocol in Figure 5 to generate the signing keys.

The main idea again is that $\mathcal{A}_{\text{co-CDH}}$ will set up a rigged public key during the DKG protocol by running the simulator $\mathcal{S}_{\mathsf{DKG}}$, where: (i) lemma 8 ensures that $\mathcal{A}$'s view of the simulated protocol is indistinguishable from its view of the real protocol; and (ii) lemma 9 ensures that the simulated keys are rigged with $\mathsf{pk} = g^s h v^u$ for some uniformly random s and u. Finally, using the same argument as Lemma 4, whenever $\mathcal{A}$ forges a signature with the rigged public key, $\mathcal{A}_{\text{co-CDH}}$ will solve the co-CDH challenge.

Simulating the DKG protocol. $\mathcal{A}_{\text{co-CDH}}$ simulates the DKG protocol with $\mathcal{A}$ by running the DKG simulator $\mathcal{S}_{\mathsf{DKG}}$. Let $\mathcal{C}$ be the set of parties $\mathcal{A}$ has corrupted so far, and let $\mathcal{H} := [n] \setminus \mathcal{C}$ be the set of honest parties. Also, let $\hat{i} \in \mathcal{H}$ be the SIP chosen by $\mathcal{S}_{\mathsf{DKG}}$ during the simulation.

Let $\mathcal{Q}$ be the qualified set of parties during the DKG simulation. Since $\mathcal{H} \subseteq \mathcal{Q}$, we have $\hat{i} \in \mathcal{Q}$. Let $s(\cdot), r(\cdot), u(\cdot)$ be the degree t polynomials defined as in step 2 of Figure 7:

$$s(x) := \sum_{i \in \mathcal{Q}} s_i(x); \qquad r(x) := \sum_{i \in \mathcal{Q}} r_i(x); \qquad u(x) := \sum_{i \in \mathcal{Q}} u_i(x) \tag{35}$$

Since $2t < n$, $\mathcal{A}_{\text{co-CDH}}$ can extract the polynomials $s(\cdot), r(\cdot)$ and $u(\cdot)$ in its entirety using a straight-line extractor. Moreover, lemma 9 implies that assuming hardness of DL, we $r(0) = 1$ and $u(0) = u$.

Simulating threshold signature. $\mathcal{A}_{\text{co-CDH}}$ simulates the threshold signing phase exactly as in §6.4, except how it responds to additional corruption queries. More precisely, if $\mathcal{A}$ corrupts any party $i \in \mathcal{H}_{-\hat{i}}$ anytime during the signing phase, $\mathcal{A}_{\text{co-CDH}}$ reveals the $(\mathsf{st}_i, \mathsf{sk}_i)$ to $\mathcal{A}$ for DKG internal state st_i of party i. $\mathcal{A}_{\text{co-CDH}}$ also updates $\mathcal{C} := \mathcal{C} \cup \{i\}$ and $\mathcal{H} = \mathcal{H} \setminus \{i\}$. Alternatively, if $\mathcal{A}$ corrupts party $\hat{i}$, $\mathcal{A}_{\text{co-CDH}}$ rewinds $\mathcal{A}$ to the start of the simulation and restarts the simulation, including re-running $\mathcal{S}_{\mathsf{DKG}}$ with fresh randomness.

Breaking the co-CDH assumption. Let $(m_{\hat{k}}, \sigma)$ be a forgery output by $\mathcal{A}$. Recall that the public key in the simulated protocol is $g^s h v^u$ where $\mathcal{A}_{\text{co-CDH}}$ knows (s, u). $\mathcal{A}_{\text{co-CDH}}$ computes the co-CDH solution $\hat{g}_{\text{cdh}}$ as

$$\hat{g}_{\text{cdh}} := \sigma \cdot \hat{g}^{-b(s + \alpha_v u)} \tag{36}$$

The correctness of $\hat{g}_{\text{cdh}}$ follows from an argument similar to Lemma 4.

Next, we illustrate that assuming the hardness of DDH in $\hat{\mathbb{G}}$, if $\mathcal{A}$ forges a signature in the $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game when the signing keys are generated by using our DKG protocol from Figure 5, then $\mathcal{A}$ also forges a

signature during its interaction with $\mathcal{A}_{\text{co-CDH}}$, where $\mathcal{A}_{\text{co-CDH}}$ uses the DKG protocol in Figure 5 to generate the signing keys, just with a slightly lower probability.

As in §6.4, we will illustrate this via a sequence of games. Game $\mathbf{G}_0$ be the $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game, and game $\mathbf{G}_6$ is the interaction of $\mathcal{A}$ with $\mathcal{A}_{\text{co-CDH}}$ as per Figure 7. Also, as in §6.4, for any game $\mathbf{G}_i$, we will use "$\mathbf{G}_i \Rightarrow 1$" as a shorthand for the event that a PPT adversary $\mathcal{A}$ forges a signature in game $\mathbf{G}_i$.

GAME $\mathbf{G}_0$ **to** GAME $\mathbf{G}_4$: Similar to game $\mathbf{G}_0$ to $\mathbf{G}_4$ in §6.4, except the game runs the DKG protocol in Figure 5 with $\mathcal{A}$ instead of the KGen functionality to generate the signing keys. Hence, by a similar argument as in §6.4, we get that:

$$\Pr[\mathbf{G}_4 \Rightarrow 1] \geq \frac{\varepsilon_\sigma}{q_{\mathsf{H}}} - \frac{1}{|\mathbb{F}|} - \varepsilon_{\text{DDH}} - \varepsilon_{\text{nizk-fail}}. \tag{37}$$

Here, ε_σ is the winning probability of $\mathcal{A}$ in the $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game.

GAME $\mathbf{G}_5$: This game is identical to $\mathbf{G}_4$ except that we run the DKG simulator $\mathcal{S}_{\mathsf{DKG}}$ to set up the signing keys. Lemma 8 ensures that $\mathcal{A}$'s interaction with $\mathcal{S}_{\mathsf{DKG}}$ is identically distributed as its view in the real protocol execution. Moreover, an argument similar to Lemma 6 implies that $\mathcal{A}$'s view of the rest of the signing phase is identically distributed in the game $\mathbf{G}_4$ and $\mathbf{G}_5$. Combining these, we get that $\Pr[\mathbf{G}_4 = 1] = \Pr[\mathbf{G}_5 = 1]$.

GAME $\mathbf{G}_6$: This game is identical to $\mathbf{G}_5$, except that we use actual NIZK proofs for partial signatures. As in §6.4, we switch back to real proofs in this game because $\mathcal{A}_{\text{co-CDH}}$ in Figure 7 uses real proofs in its interaction with $\mathcal{A}$. Therefore, using an argument similar as in the advantage between $\mathbf{G}_3$ and $\mathbf{G}_4$ in §6.4, we get that:

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \varepsilon_{\text{nizk-fail}}. \tag{38}$$

Observe that, if game $\mathbf{G}_6$ does not abort, then $\mathcal{A}$'s view in game $\mathbf{G}_6$ is identically distributed as its view in its interaction with $\mathcal{A}_{\text{co-CDH}}$ in Figure 7, where $\mathcal{A}_{\text{co-CDH}}$ uses (g^a, g^b) from co-CDH input $(g, g^a, g^b, \hat{g}^a)$ as (h, g^β) in game $\mathbf{G}_6$. Additionally, $\mathcal{A}_{\text{co-CDH}}$ uses $\hat{g}^a$ to compute the random oracle outputs in step 11(b) in Figure 4. Hence, from the above sequence of games, we get that:

$$\Pr[\mathbf{G}_6 \Rightarrow 1] \geq \frac{1}{q_{\mathsf{H}}} \cdot \varepsilon_\sigma - \varepsilon_{\text{DDH}} - \frac{1}{|\mathbb{F}|} - 2\varepsilon_{\text{nizk-fail}}. \tag{39}$$

This implies that if adversary $\mathcal{A}$ outputs a forgery in the $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game of our signature scheme (i.e., $\mathbf{G}_0$) with probability ε_σ, then $\mathcal{A}$ outputs a forgery on $m_{\hat{k}}$ during its interaction with $\mathcal{A}_{\text{co-CDH}}$ in Figure 7 (i.e., in $\mathbf{G}_6$) with probability at least $\varepsilon_\sigma/q_{\mathsf{H}} - \varepsilon_{\text{DDH}} - 1/|\mathbb{F}| - 2\varepsilon_{\text{nizk-fail}}$. Moreover, Lemma 4 implies that $\mathcal{A}_{\text{co-CDH}}$ can efficiently compute the co-CDH solution using the forgery on $m_{\hat{k}}$. Combining all the above, we get our second main theorem, as stated below.

Theorem 5 (Adaptively secure BLS threshold signature with DKG). *Let λ be the security parameter, and let $(\mathbb{F}, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p) \leftarrow \mathsf{GGen}(1^\lambda)$ be pairing groups of prime order p. For any n, t for $n = \mathsf{poly}(\lambda)$ and $t < n/2$, let DKG be a DKG protocol that satisfies the properties we describe in §7.1. Then, assuming the hardness of decisional Diffie-Hellman (DDH) in $\hat{\mathbb{G}}$, and hardness of co-computational Diffie-Hellman (co-CDH) in $(\mathbb{G}, \hat{\mathbb{G}})$ in the random oracle model, any PPT adversary making at most q_{H} random oracle queries to H_0 and H_1 combined, q_{FS} queries to the random oracle H_{FS}, and at most q_s signature queries wins the $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game Figure 1 for our signature scheme in Figure 2 with probability at most ε_σ where:*

$$\varepsilon_\sigma \leq q_{\mathsf{H}} \cdot \left(\varepsilon_{\text{DDH}} + \frac{1}{|\mathbb{F}|} + 2\varepsilon_{\text{nizk-fail}} + \varepsilon_{\text{CDH}} \right),$$

$\varepsilon_{\text{nizk-fail}} = (q_{\mathsf{FS}} \cdot q_s \cdot n)/|\mathbb{F}|^2$, and ε_{DDH} and ε_{CDH} are the advantages of an adversary running in $T \cdot \mathsf{poly}(\lambda, n)$ time in breaking DDH in $\hat{\mathbb{G}}$ and co-CDH in $(\mathbb{G}, \hat{\mathbb{G}})$, respectively. This implies that ε_σ is negligible, and hence, our threshold signature scheme in §5 is unforgeable.

8.1 Unforgeability with static adversary

We now briefly argue that if we are content with proving our signature scheme statically secure, then we only need the hardness of CDH assumption in a pairing group $(\mathbb{G}, \hat{\mathbb{G}})$ in the ROM. Let $\mathcal{A}_{\text{static}}$ be the static adversary that breaks the unforgeability of our signature scheme, and let $\mathcal{A}_{\text{CDH}}$ be the CDH adversary. Then, except for the DKG simulation, $\mathcal{A}_{\text{CDH}}$ interacts with $\mathcal{A}_{\text{static}}$ exactly as in §6.5. $\mathcal{A}_{\text{CDH}}$ simulates the DKG protocol as follows.

Simulating the DKG protocol. For simplicity, let us assume $|\mathcal{C} \cup \mathcal{S}| = t$. $\mathcal{A}_{\text{CDH}}$ samples $h, v \leftarrow\$ \mathbb{G}$. Next, on behalf of each honest node $i \in \mathcal{H}$, $\mathcal{A}_{\text{CDH}}$ picks random degree t polynomials $r_i(x), u_i(x)$ with $r_i(0) = u_i(0) = 0$. To compute the polynomial $s_i(x)$, $\mathcal{A}_{\text{static}}$ samples $\alpha_i \leftarrow\$ \mathbb{F}$ and $s_i(j) \leftarrow\$ \mathbb{F}$ for each $j \in \mathcal{C} \cup \mathcal{S}$. $\mathcal{A}_{\text{CDH}}$ then uses $s_i(0) = s \cdot \alpha_i$ as the secret of dealer i. $\mathcal{A}_{\text{CDH}}$ computes the DKG commitments as: $\mathsf{cm}_i[0] = g^{s_i(0)}$ and $\mathsf{cm}_i[j] = g^{s_{i,j}}$ for each $j \in [t]$, using interpolation in the exponent. $\mathcal{A}_{\text{CDH}}$ then continue the rest of the simulation as per the standard approach [GJKR07].

9 Implementation and Evaluation

9.1 Evaluation Setup

We implement our threshold signature scheme in Go. Our implementation is publicly available at `https://github.com/sourav1547/adaptive-bls`. We use the `gnark-crypto` library [BPH+23] for efficient finite field and elliptic curve arithmetic for the BLS12-381 curve. We also use (for both our implementation and the baselines) the multi-exponentiation of group elements using Pippenger's method [BDLO12] for efficiency. We evaluate our scheme and baselines on a *t3.2xlarge* Amazon Web Service (AWS) instance with 32 GB RAM, 8 virtual cores, and 2.50GHz CPU.

Baselines. We implement two variants of Boldyreva's BLS threshold signatures as baselines. The variants differ in how the aggregator validates the partial signatures. The Boldyreva-I variant is the standard variant we describe in §4.4. In Boldyreva-II, along with the partial signatures, signers also attach a Σ-protocol proof attesting to the correctness of the partial signatures. Instead of pairings, the aggregator uses the Σ-protocol proof to check the validity of the partial signatures, resulting in faster verification time. We refer readers to Burdges et al. [BCLS22] for more details on Boldyreva-II. For Σ-protocols in both Boldyreva-II and our scheme, we use the standard optimization where the proof omits the first message of the prover and instead includes the Fiat-Shamir challenge [CS97].

We evaluate the *signing time* and *partial signature verification time* of our scheme. The signing time refers to the time a signer takes to sign a message and compute the associated proofs. The partial signature verification time measures the time the aggregator takes to verify a single partial signature. Note that after partial signature verification, the aggregation time of our threshold signature is identical to the aggregation time of Boldyreva's scheme, but for completeness, we also measure the total *aggregation time*. Our final verification time is identical to Boldyreva's scheme (and standard BLS).

9.2 Evaluation Results

We report our results in Table 1. Through our evaluation, we seek to illustrate that our scheme only adds a small overhead compared to Boldyreva's scheme [Bol03] to achieve adaptive security.

Signing time. As expected, the per signer signing time of Boldyreva-II is slightly higher than Boldyreva-I, since a signer in Boldyreva-II also computes the Σ-protocol proof. Similarly, our per signer signing cost is 3.3× higher than Boldyreva-II as our Σ-protocol involves more computation than Boldyreva-II.

Partial signature verification time. The verification time of Boldyreva-II is less than Boldyreva-I, since pairings operations are much slower than group exponentiations. As expected, our partial signature verification time is 2.84× longer than Boldyreva-II due to more expensive Σ-protocol verification. Compared to Boldyreva-I, our partial signature verification is 1.92× slower.

Table 1: Comparison of BLS threshold signatures using BLS12-381 elliptic curve. We assume that public keys are in $\mathbb{G}$ and signatures are in $\hat{\mathbb{G}}$.

Scheme	Partial signing time (in ms)	Partial signature verification time (in ms)	Partial Signature size (in bytes)	Aggregation time for $t = 64$ (in ms)
Boldyreva-I	0.81	1.12	96	74.01
Boldyreva-II	1.20	0.76	160	55.43
Ours scheme	3.92	2.16	224	149.52

Partial signature size. The partial signature size only depends on the underlying elliptic curve group we use. For the BLS12-381 elliptic curve, $\mathbb{F}, \mathbb{G}$ and $\hat{\mathbb{G}}$ elements are 32, 48, and 96 bytes, respectively. The partial signature in Boldyreva-I is a single $\hat{\mathbb{G}}$ element, which is 96 bytes. In Boldyreva-II, the partial signature also includes a Σ-protocol proof that is $(c, z) \in \mathbb{F}^2$, using the standard optimization of including the Fiat-Shamir challenge [CS97]. Hence, the partial signatures in Boldyreva-II are 64 bytes longer than Boldyreva-I. Finally, our partial signature includes a Σ-protocol proof $(c, z_s, z_r, z_u) \in \mathbb{F}^4$, and hence is 224-byte long in total. If we assume that parties are semi-honest, then partial signatures of all three schemes will be identical.

Total aggregation time. We measure the total signature aggregation time for $t = 64$. Recall during aggregation, the aggregator, apart from verifying the partial signatures, performs $O(t \log^2 t)$ field operations to compute all the Lagrange coefficients and a multi-exponentiation of width t [TCZ+20]. Since field operations are orders of magnitude faster than group exponentiations, for moderate values of t such as 64, the partial signature verification costs dominate the total aggregation time. Thus, the aggregation time of all three schemes we evaluate is approximately t times the single partial signature verification time.

Common case optimization of aggregation time. Note that it is possible to optimize the aggregation time of both the baselines and our scheme in the common case. More specifically, the aggregator can optimistically aggregate the partial signatures without verifying them individually and then verify the aggregated signature. If the final verification is successful, the aggregator outputs the aggregated signature. Otherwise, the aggregator validates the partial signature individually, identifies the invalid ones, discards them, and recomputes the aggregated signature. Moreover, the aggregator discards the partial signatures from the signers who sent invalid partial signatures in all future aggregations. We refer to the latter as the *fall-back* path.

Our evaluation illustrates that with this optimization, the aggregation in the optimistic case is 7.7 milliseconds (in AWS t3.2xlarge machine) for both the baselines and our scheme. Also, the robustness property implies that the aggregator will always identify at least one malicious party in case of the fall-back path and will never blame an honest party. This implies that the aggregator needs to run the fall-back path at most t times in total. Thus, we believe that in a long-running system, our added overhead is very minimal.

10 Discussion and Conclusion

In this paper, we presented a new adaptively secure threshold BLS signature scheme and a distributed key generation protocol for it. Our scheme is adaptively secure assuming the hardness of decisional Diffie Hellman (DDH) and co-computational Diffie Hellman assumption (co-CDH) in asymmetric pairing groups in the random oracle model (ROM). The security of our scheme gracefully degenerates: in the presence of a static adversary, our scheme relies only on the hardness of CDH in pairing groups in the ROM, which is the same assumption as in the standard non-threshold BLS signature scheme.

Our scheme maintains the non-interactive signing, compatible verification, and practical efficiency of Boldyreva's BLS threshold signatures. We implemented our scheme in Go, and our evaluation illustrates that it has a small overhead over the Boldyreva scheme.

Future research directions. Our scheme only works with type-II and type-III asymmetric pairing groups. This is because the security of our signature scheme assumes the hardness of DDH. Removing the reliance on the DDH assumption on a source group is a fascinating open problem. Another exciting research direction is

to extend our ideas to prove the adaptive security of other threshold signature or encryption schemes such as threshold Schnorr, ECDSA, and RSA.

Acknowledgments

We want to thank Dan Boneh for pointing us to the DDH rerandomization in their book and Leonid Reyzin for pointing us to the [NR04]. We would also like to thank Crypto 2024 and Eurocrypt 2024 reviewers for their helpful suggestions on how to improve the paper presentation. Finally, we thank Amit Agarwal, Renas Bacho, Julian Loss, Victor Shoup, Alin Tomescu, and Zhoulun Xiang for helpful discussions related to the paper. This work is funded in part by a Chainlink Labs Ph.D. fellowship and the National Science Foundation award #2240976.

References

ADN06. Jesús F Almansa, Ivan Damgård, and Jesper Buus Nielsen. Simplified threshold rsa with adaptive and proactive security. In *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25*, pages 593–611. Springer, 2006.

AF04. Masayuki Abe and Serge Fehr. Adaptively secure feldman vss and applications to universally-composable threshold cryptography. In *Annual International Cryptology Conference*, pages 317–334. Springer, 2004.

AMS19. Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. Asymptotically optimal validated asynchronous byzantine agreement. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 337–346, 2019.

arp23. Randcast-arpa network. `https://docs.arpanetwork.io/randcast`, 2023.

BCK+22. Mihir Bellare, Elizabeth Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In *Annual International Cryptology Conference*, pages 517–550. Springer, 2022.

BCLS22. Jeff Burdges, Oana Ciobotaru, Syed Lavasani, and Alistair Stewart. Efficient aggregatable bls signatures with chaum-pedersen proofs. *Cryptology ePrint Archive*, 2022.

BDLO12. Daniel J Bernstein, Jeroen Doumen, Tanja Lange, and Jan-Jaap Oosterwijk. Faster batch forgery identification. In *Progress in Cryptology-INDOCRYPT 2012: 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings 13*, pages 454–473. Springer, 2012.

BGP92. Piotr Berman, Juan A Garay, and Kenneth J Perry. Bit optimal distributed consensus. In *Computer science*, pages 313–321. Springer, 1992.

BHK+24. Fabrice Benhamouda, Shai Halevi, Hugo Krawczyk, Yiping Ma, and Tal Rabin. Sprint: High-throughput robust distributed schnorr signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 62–91. Springer, 2024.

BL22. Renas Bacho and Julian Loss. On the adaptive security of the threshold bls signature scheme. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 193–207, 2022.

BLS01. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*, pages 514–532. Springer, 2001.

BLT+24. Renas Bacho, Julian Loss, Stefano Tessaro, Benedikt Wagner, and Chenzhi Zhu. Twinkle: Threshold signatures from ddh with full adaptive security. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 429–459. Springer, 2024.

BN06. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 390–399, 2006.

Bol03. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography*, volume 2567, pages 31–46. Springer, 2003.

BP23. Luís T. A. N. Brandão and Rene Peralta. Nist ir 8214c: First call for multi-party threshold schemes. `https://csrc.nist.gov/pubs/ir/8214/c/ipd`, 2023.

BPH+23. Gautam Botrel, Thomas Piellard, Youssef El Housni, Arya Tabaie, Gus Gutoski, and Ivo Kubjas. Consensys/gnark-crypto: v0.9.0, January 2023.

BS23. Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Draft 0.6*, 2023.

CGG+20. Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. Uc non-interactive, proactive, threshold ecdsa with identifiable aborts. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1769–1787, 2020.

CGJ+99. Ran Canetti, Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*, pages 98–116. Springer, 1999.

CGRS23. Hien Chu, Paul Gerhart, Tim Ruffing, and Dominique Schröder. Practical schnorr threshold signatures without the algebraic group model. In *Annual International Cryptology Conference*. Springer, 2023.

CKM21. Elizabeth Crites, Chelsea Komlo, and Mary Maller. How to prove schnorr assuming schnorr: security of multi-and threshold signatures. *Cryptology ePrint Archive*, 2021.

CKM23. Elizabeth Crites, Chelsea Komlo, and Mary Maller. Fully adaptive schnorr threshold signatures. In *Annual International Cryptology Conference*. Springer, 2023.

CL24. Yi-Hsiu Chen and Yehuda Lindell. Feldman's verifiable secret sharing for a dishonest majority. *Cryptology ePrint Archive*, 2024.

CS97. Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. *Technical Report/ETH Zurich, Department of Computer Science*, 260, 1997.

CS04. John Canny and Stephen Sorkin. Practical large-scale distributed key generation. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 138–152. Springer, 2004.

Dam02. Ivan Damgård. On σ-protocols. *Lecture Notes, University of Aarhus, Department for Computer Science*, page 84, 2002.

DCX+23. Sourav Das, Philippe Camacho, Zhuolun Xiang, Javier Nieto, Benedikt Bünz, and Ling Ren. Threshold signatures from inner product argument: Succinct, weighted, and multi-threshold. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 356–370, 2023.

Des88. Yvo Desmedt. Society and group oriented cryptography: A new concept. In *Advances in Cryptology—CRYPTO'87: Proceedings 7*, pages 120–127. Springer, 1988.

DF89. Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, 1989.

dra23. Distributed randomness beacon: Verifiable, unpredictable and unbiased random numbers as a service. `https://drand.love/docs/overview/`, 2023.

DS83. Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

DYX+22. Sourav Das, Thomas Yurek, Zhuolun Xiang, Andrew Miller, Lefteris Kokoris-Kogias, and Ling Ren. Practical asynchronous distributed key generation. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2518–2534. IEEE, 2022.

FMY99a. Yair Frankel, Philip MacKenzie, and Moti Yung. Adaptively-secure distributed public-key systems. In *European Symposium on Algorithms*, pages 4–27. Springer, 1999.

FMY99b. Yair Frankel, Philip MacKenzie, and Moti Yung. Adaptively-secure optimal-resilience proactive rsa. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 180–194. Springer, 1999.

FS86. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

FS01. Pierre-Alain Fouque and Jacques Stern. One round threshold discrete-log key generation without private channels. In *International Workshop on Public Key Cryptography*, pages 300–316. Springer, 2001.

GHM+17. Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th symposium on operating systems principles*, pages 51–68, 2017.

GJKR96. Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold dss signatures. In *Advances in Cryptology—EUROCRYPT'96: International Conference on the Theory and Application of Cryptographic Techniques Saragossa, Spain, May 12–16, 1996 Proceedings 15*, pages 354–371. Springer, 1996.

GJKR07. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, 2007.

GJM+24. Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. hints: Threshold signatures with silent setup. In *2024 IEEE Symposium on Security and Privacy (SP)*, 2024.

GKKS+22. Rati Gelashvili, Lefteris Kokoris-Kogias, Alberto Sonnino, Alexander Spiegelman, and Zhuolun Xiang. Jolteon and ditto: Network-adaptive efficient consensus with asynchronous fallback. In *International conference on financial cryptography and data security*. Springer, 2022.

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

Gro21. Jens Groth. Non-interactive distributed key generation and key resharing. *IACR Cryptol. ePrint Arch.*, 2021:339, 2021.

GS24. Jens Groth and Victor Shoup. Fast batched asynchronous distributed key generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 370–400. Springer, 2024.

ic23. Internet computer: Chain-key cryptography. `https://internetcomputer.org/how-it-works/chain-key-technology/`, 2023.

JL00. Stanisław Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In *Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19*, pages 221–242. Springer, 2000.

KG21. Chelsea Komlo and Ian Goldberg. Frost: flexible round-optimized schnorr threshold signatures. In *Selected Areas in Cryptography: 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers 27*, pages 34–65. Springer, 2021.

KHG12. Aniket Kate, Yizhou Huang, and Ian Goldberg. Distributed key generation in the wild. *IACR Cryptol. ePrint Arch.*, 2012:377, 2012.

KKMS20. Eleftherios Kokoris Kogias, Dahlia Malkhi, and Alexander Spiegelman. Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1751–1767, 2020.

KL07. Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols*. Chapman and hall/CRC, 2007.

KY02. Jonathan Katz and Moti Yung. Threshold cryptosystems based on factoring. In *Advances in Cryptology—ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security Queenstown, New Zealand, December 1–5, 2002 Proceedings 8*, pages 192–205. Springer, 2002.

LJY14. Benoît Libert, Marc Joye, and Moti Yung. Born and raised distributively: Fully distributed non-interactive adaptively-secure threshold signatures with short shares. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pages 303–312, 2014.

LLTW20. Yuan Lu, Zhenliang Lu, Qiang Tang, and Guiling Wang. Dumbo-mvba: Optimal multi-valued validated asynchronous byzantine agreement, revisited. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 129–138, 2020.

LP01. Anna Lysyanskaya and Chris Peikert. Adaptive security in the threshold setting: From cryptosystems to signature schemes. In *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*, pages 331–350. Springer, 2001.

LSP82. Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

LY13. Benoît Libert and Moti Yung. Adaptively secure non-interactive threshold cryptosystems. *Theoretical Computer Science*, 478:76–100, 2013.

MR21. Atsuki Momose and Ling Ren. Optimal communication complexity of authenticated byzantine agreement. In *35th International Symposium on Distributed Computing, DISC 2021*, page 32. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2021.

MXC+16. Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 31–42, 2016.

NR04. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM (JACM)*, 51(2):231–262, 2004.

Ped91. Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 522–526. Springer, 1991.

Rab98. Tal Rabin. A simplified approach to threshold and proactive rsa. In *Advances in Cryptology—CRYPTO'98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings 18*, pages 89–104. Springer, 1998.

RRJ+22. Tim Ruffing, Viktoria Ronge, Elliott Jin, Jonas Schneider-Bensch, and Dominique Schröder. Roast: Robust asynchronous schnorr threshold signatures. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2551–2564, 2022.

Sch90. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology—CRYPTO'89 Proceedings 9*, pages 239–252. Springer, 1990.

Sha79. Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

Sho00. Victor Shoup. Practical threshold signatures. In *Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19*, pages 207–220. Springer, 2000.

Sho23. Victor Shoup. The many faces of schnorr. *Cryptology ePrint Archive*, 2023.

ska23. Skale network documentation: Distributed key generation (dkg). `https://docs.skale.network/technology/dkg-bls`, 2023.

TCZ+20. Alin Tomescu, Robert Chen, Yiming Zheng, Ittai Abraham, Benny Pinkas, Guy Golan Gueta, and Srinivas Devadas. Towards scalable threshold cryptosystems. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 877–893. IEEE, 2020.

TZ23. Stefano Tessaro and Chenzhi Zhu. Threshold and multi-signature schemes from linear hash functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 628–658. Springer, 2023.

Wat05. Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24*, pages 114–127. Springer, 2005.

WQL09. Zecheng Wang, Haifeng Qian, and Zhibin Li. Adaptively secure threshold signature scheme in the standard model. *Informatica*, 20(4):591–612, 2009.

YMR+19. Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: Bft consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356. ACM, 2019.