# Adaptively Secure BLS Threshold Signatures from DDH and co-CDH

Sourav Das Ling Ren

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

**Abstract.** Threshold signature is 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 Boldyreva's threshold signature, 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. Moreover, to achieve static security, our scheme only needs the hardness of CDH in the ROM, which is the same as the standard non-threshold BLS signature. These properties make our protocol a suitable candidate for practical adoption with the added benefit of provable adaptive security. We also present an efficient distributed key generation (DKG) protocol to set up the signing keys for our signature scheme. We implement our scheme in Go and evaluate its signing and aggregation costs.

## 1 Introduction

Threshold signatures schemes [Des88, 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 Byzantine Fault Tolerant (BFT) applications has resulted in large-scale adoptions of threshold signature schemes. Many state-of-the-art BFT 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, and the signatures are unique and small, i.e., a signature consists of a single elliptic curve group element. The Boldyreva scheme is also very efficient in terms of both computation and communication. These properties have resulted in several 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 strong assumption such as OMDL is necessary.

**Our results.** We present an adaptively secure threshold signature scheme for BLS signatures. 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*. In other words, our scheme only relies on DDH besides what standard non-threshold BLS signature already relies on. In addition, our security reduction only incurs a very small security loss over the original BLS signature. Moreover, if one is content with proving our scheme statically secure, then we only need CDH in the ROM, similar to 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 consists 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 Go and compare its performance with Boldyreva's scheme. Our evaluation confirms that our scheme adds very small overheads. Hence, we believe our scheme provides a worthwhile trade-off for the added benefit of provable adaptive security at modest performance cost.

We also describe a distributed key generation (DKG) protocol to secret share the signing key of our signature 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.

**Paper organization.** We discuss the related work in §2 and present a technical overview of our scheme in §3. In §4, we describe 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, GG20, KG21, CKM21, BCK+22, RRJ+22, CGRS23, TZ23, Sho23, BHK+23, GS23]. 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, where we classify them into *interactive* and *non-interactive* schemes.

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

**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 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.

**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.

## 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.

Before we describe our techniques, we briefly recall the non-threshold BLS signature scheme.

### 3.1 Boneh-Lynn-Sacham (BLS) signature scheme [BLS01]

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 $\sigma$ on a message $m$ is then $\mathsf{H}(m)^{\mathsf{sk}} \in \hat{\mathbb{G}}$. Any verifier

validates a signature $\sigma'$ 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].

### 3.2 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. §5 and §6 describes our final protocol and proof of adaptive security.

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 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 $r(0) = 0$, upon interpolating the partial signatures, the aggregator will obtain 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. This is indeed the case. Instead, signers in our protocol use a "$\Sigma$"-protocol to prove the correctness of their partial signatures.

Naturally, the important question is how this modified BLS threshold signature scheme 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 $\mathsf{pk} = g^s h^r \in \mathbb{G}$. $\mathcal{A}_{\text{co-CDH}}$ 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 signature on some message $m$, which is a standard BLS signature, 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 $\sigma = \hat{g}^{(s+ar)b}$. This implies that if $s, r \in \mathbb{F}$ are known, then we can efficiently compute $\hat{g}^{ab}$ given $\sigma$.

Let $s(x), r(x)$ be degree $t$ polynomials for Shamir secret sharing of $s = s(0)$ and $r = r(0)$. We will discuss in §6 how $\mathcal{A}_{\text{co-CDH}}$ interacts with $\mathcal{A}$ while ensuring that it learns $s(x)$ and $r(x)$, and rigging the public key using just a single inconsistent party. Since $\mathcal{A}_{\text{co-CDH}}$ knows both $s(x), r(x)$, it can reveal the internal state of any party except the inconsistent party to $\mathcal{A}$. Unless $\mathcal{A}$ corrupts the inconsistent party, $\mathcal{A}$'s view is identically distributed in a real protocol instance and in an instance rigged by $\mathcal{A}_{\text{co-CDH}}$.

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 $r \neq 0$, 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 $\mathsf{H}_1$ and have the partial signatures as $\sigma_i = \mathsf{H}_0(m)^{s(i)} \mathsf{H}_1(m)^{r(i)}$. With this new partial signature, the final aggregated signature is $\sigma = \mathsf{H}_0(m)^s \mathsf{H}_1(m)^r$. 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 show 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 $\lambda$ 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 $\lambda$. 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*

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

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 co-computational Diffie-Hellman (co-CDH) holds for $(\mathbb{G}, \hat{\mathbb{G}})$.

**Assumption 1 (DDH)** *Let $\hat{\mathbb{G}}$ be a cyclic group with scalar field $\mathbb{F}$. For security parameter $\lambda$, any PPT adversary $\mathcal{A}$ and uniformly random $a, b, z \leftarrow\$ \mathbb{F}$, let $\varepsilon_{\mathrm{DDH}}$ be the advantage of $\mathcal{A}$ as defined below:*

$$\varepsilon_{\mathrm{DDH}} := \left|\Pr[\mathcal{A}(\hat{g}, \hat{g}^a, \hat{g}^b, \hat{g}^z) = 1] - \Pr[\mathcal{A}(\hat{g}, \hat{g}^a, \hat{g}^b, \hat{g}^{ab}) = 1]\right|. \quad (1)$$

*Here, the probability is over the choice $a, b, z$ and $\mathcal{A}$'s randomness. The group $\hat{\mathbb{G}}$ satisfies the Decisional Diffie-Hellman (DDH) assumption if, it holds that $\varepsilon_{\mathrm{DDH}} = \mathsf{negl}(\lambda)$.*

**Assumption 2 (co-CDH)** *Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ be a pairing group with scalar field $\mathbb{F}$, generators $g \in \mathbb{G}, \hat{g} \in \hat{\mathbb{G}}$, and a bilinear pairing $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$. For security parameter $\lambda$, any PPT adversary $\mathcal{A}$ and uniformly random $a, b \leftarrow\$ \mathbb{F}$, let $\varepsilon_{\mathrm{CDH}}$ be the advantage of $\mathcal{A}$ as defined below:*

$$\varepsilon_{\mathrm{CDH}} := \Pr[\mathcal{A}(g, \hat{g}, g^a, \hat{g}^a, \hat{g}^b) = \hat{g}^{ab}]. \tag{2}$$

*Here, the probability is over the choice $a, b$ and $\mathcal{A}$'s randomness. The paring group $(\hat{\mathbb{G}}, \hat{\mathbb{G}}, \mathbb{G}_T)$ satisfies the the co-Computational Diffie-Hellman (co-DDH) assumption if, it holds that $\varepsilon_{\mathrm{CDH}} = \mathsf{negl}(\lambda)$.*

**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).** *A non-interactive $(n, t)$-threshold signature scheme with a finite message space $\mathcal{M}$ is a tuple of polynomial time computable algorithms $\Sigma = (\mathsf{KGen}, \mathsf{PSign}, \mathsf{PVer}, \mathsf{Comb}, \mathsf{Ver})$ with the following properties. (All algorithms implicitly take the public parameters $\mathsf{pp}$ as input.)*

1. $\mathsf{KGen}(n, t) \to \mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in[n]}$. *The key generation algorithm takes as input the total number of signers $n$, and the threshold $t$. The algorithm then outputs a public key $\mathsf{pk}$, a vector of threshold public keys $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$, and a vector of secret signing keys $\{\mathsf{sk}_1, \ldots, \mathsf{sk}_n\}$. The $j$-th signer receives its own secret key and all the public keys, i.e., $(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{sk}_j)$.*
2. $\mathsf{PSign}(m, \mathsf{sk}_i) \to \sigma_i$. *The partial signing algorithm is a possibly randomized algorithm that takes as input a message $m \in \mathcal{M}$ and a secret key share $\mathsf{sk}_i$. It outputs a signature share $\sigma_i$.*
3. $\mathsf{PVer}(m, \sigma_i, \mathsf{pk}_i) \to 0/1$. *The signature share verification algorithm takes as input a message $m$, a threshold public key share $\mathsf{pk}_i$, and a signature share $\sigma_i$. It outputs 1 (accept) or 0 (reject).*
4. $\mathsf{Comb}(S, \{(\sigma_i, i)\}_{i\in S}) \to \sigma/\perp$. *The signature share combining algorithm takes as input a public key $\mathsf{pk}$, a vector of public key shares $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, a message $m$, and a set $S$ of $t + 1$ signature shares $(\sigma_i, i)$ (with corresponding indices). It outputs either a signature $\sigma$ or $\perp$.*
5. $\mathsf{Ver}(m, \mathsf{pk}, \sigma)$. *The signature verification algorithm is a deterministic algorithm that takes as input a public key $\mathsf{pk}$, a message $m$, and a signature $\sigma$. It outputs 1 (accept) or 0 (reject).*

We require any non-interactive threshold signature scheme to satisfy *Correctness* and *Unforgeability* as defined below.

Intuitively, *correctness* ensures that the protocol should behave as expected for honest parties. More precisely, it says that: (i) $\mathsf{PVer}$ should always accept honestly generated partial signatures; and (ii) if we combine $t + 1$ valid partial signatures (accepted by $\mathsf{PVer}$) using the $\mathsf{Comb}$ algorithm, the output of $\mathsf{Comb}$ should be always accepted by $\mathsf{Ver}$, 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 we allow $\mathcal{A}$ to generate partial signatures in an arbitrary manner. Also, we can achieve correctness even if $\mathcal{A}$ learns all signing keys.

Game $\mathbf{UF\text{-}CMA}_{\Sigma}^{\mathcal{A}}$:

// Setup and initialization

1: $\mathsf{pp}, \mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_{\mathbb{F}} \leftarrow \mathsf{Setup}(1^\lambda)$ // pp is the public parameters of the scheme

2: $n, t, \mathcal{C}, \mathsf{st}_{\mathcal{A}} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{H}}, \mathcal{O}^{\mathsf{H}_{\mathbb{F}}}}(\mathsf{pp})$. // $\mathcal{C}$ is the growing set of corrupt signers, and $\mathsf{st}_{\mathcal{A}}$ is some auxiliary information output by $\mathcal{A}$

3: $\mathcal{H} := [n] \setminus \mathcal{C}$ // set of honest signers

4: Let $\mathsf{st}_i := \{\}, \forall\, i \in \mathcal{H}$ be signer $i$'s internal state.

// Key generation

5: $\mathsf{pk}, \{\mathsf{pk}_i, \mathsf{sk}_i\} \leftarrow \mathsf{KGen}(n, t)$

6: Send $(\mathsf{pk}, \{\mathsf{pk}_j\}_{j\in[n]}, \mathsf{sk}_i)$ to each signer $i \in [n]$

7: Send $(\mathsf{pk}, \{\mathsf{pk}_j\}_{j\in[n]}, \{\mathsf{sk}_j\}_{j\in\mathcal{C}})$ to $\mathcal{A}$

// Forgery and output determination

8: Let $\mathsf{input} := (\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in\mathcal{C}}, \mathsf{st}_{\mathcal{A}})$

9: $(m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Corrupt}}, \mathcal{O}^{\mathsf{PSign}}, \mathcal{O}^{\mathsf{H}}, \mathcal{O}^{\mathsf{H}_{\mathbb{F}}}}(\mathsf{input})$

// $Q[m]$, initially $\{\}$, denotes the set of signers $\mathcal{A}$ queries for the partial signatures on $m$

10: **if** $|Q[m] \cup \mathcal{C}| \le t \wedge \mathsf{Ver}(m, \mathsf{pk}, \sigma) = 1$ **then**

11: **return** 1

12: **return** 0

$\mathcal{O}^{\mathsf{Corrupt}}(i)$:

9: **if** $i \in \mathcal{H}$ **then**

10: $\mathcal{H} := \mathcal{H} \setminus \{i\}$; $\mathcal{C} := \mathcal{C} \cup \{i\}$

11: **return** $(\mathsf{sk}_i, \mathsf{st}_i)$

12: **return** $\perp$

$\mathcal{O}^{\mathsf{PSign}}(i, m)$:

// Only honest signer query allowed

13: **if** $i \in \mathcal{H}$ **then**

14: $Q[m] := Q[m] \cup \{i\}$

15: Let $\sigma_i \leftarrow \mathsf{PSign}(m, \mathsf{sk}_i)$

16: Update $\mathsf{st}_i$ with internal states used by signer $i$ during $\mathsf{PSign}(m, \mathsf{sk}_i)$

17: **return** $\sigma_i$

18: **return** $\perp$

$\mathcal{O}^{\mathsf{H}}(b, m)$:

17: **return** $\mathsf{H}_b(m)$

$\mathcal{O}^{\mathsf{H}_{\mathbb{F}}}(x)$:

17: **return** $\mathsf{H}_{\mathbb{F}}(x)$

Fig. 1: $\mathbf{UF\text{-}CMA}_{\Sigma}^{\mathcal{A}}$ game for non-interactive threshold signature with an adaptive adversary $\mathcal{A}$.

**Definition 3 (Correctness).** *A non-interactive threshold signature scheme $\Sigma$ for a finite message space $\mathcal{M}$ is correct, if for all security parameters $\lambda$, all allowable $1 \le t+1 \le n$, all subset $S \subseteq [n]$ with $t+1 \le |S|$, and all messages $m \in \mathcal{M}$, and for all PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ the following holds:*

*(1)* $$\Pr\left[\mathsf{PVer}(m, \sigma_i, \mathsf{pk}_i) = 1 \;\middle|\; \mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in[n]} \leftarrow \mathsf{KGen}(n, t); \sigma_i = \mathsf{PSign}(m, \mathsf{sk}_i)\ \forall i \in S\right] = 1$$

*(2)* $$\Pr\left[\mathsf{Ver}(m, \mathsf{Comb}(\{(\sigma_i, i)\}_{i\in S}), \mathsf{pk}) = 1 \;\middle|\; \begin{array}{r} (n, t) \leftarrow \mathcal{A}_0(1^\lambda) \text{ where } t+1 \le n \\ \mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in[n]} \leftarrow \mathsf{KGen}(n, t) \\ m, S, \{\sigma_i\}_{i\in S} \leftarrow \mathcal{A}_1\left(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in[n]}\right) \\ |S| \ge t+1 \wedge \mathsf{PVer}(m, \sigma_i, \mathsf{pk}_i) = 1, \forall i \in S \end{array}\right] \ge 1 - \mathsf{negl}(\lambda)$$

*Here, the probability is over the choice of randomness of the* KGen *and $\mathcal{A}$'s internal randomness.*

We next define the Unforgeability requirement for our non-interactive threshold signature scheme with adaptive corruption. Let $\mathsf{H}_b : \mathcal{M} \to \hat{\mathbb{G}}$ $(b \in \{0, 1\})$ and $\mathsf{H}_{\mathbb{F}} : \{0,1\}^* \to \mathbb{F}$ be three hash functions (modeled as random oracles). We use $q_{\mathsf{H}}$ to denote the sum of the number of queries made to $\mathsf{H}_0$ and $\mathsf{H}_1$, $q_{\mathsf{H}_{\mathbb{F}}}$ to denote the upper-bound on the number of queries to $\mathsf{H}_{\mathbb{F}}$. We formally define the unforgeability attack game in Figure 1, and summarize it below.

**Game 3 ($\mathbf{UF\text{-}CMA}_{\Sigma}^{\mathcal{A}}$)** *Let $\Sigma = (\mathsf{KGen}, \mathsf{PSign}, \mathsf{PVer}, \mathsf{Comb}, \mathsf{Ver})$ be a non-interactive $(n, t)$-threshold signature scheme for a finite message space $\mathcal{M}$. For an algorithm $\mathcal{A}$, define game $\mathbf{UF\text{-}CMA}_{\Sigma}^{\mathcal{A}}$ as:*

– **Setup and initialization.** *The setup and initialization consists of the of the following steps:*

1. *Run $\mathsf{Setup}(1^\lambda)$ to get the pubic parameters pp and the descriptions of hash functions $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_{\mathbb{F}}$.*
2. *For any message $m \in \mathcal{M}$, let $Q[m]$ be the set of honest signers $\mathcal{A}$ queries for partial signatures on message $m$. $Q[m]$ is initialized to the empty set $\{\}$ for each $m \in M$.*
3. *Run $\mathcal{A}$ on the public parameters while giving $\mathcal{A}$ oracle access to $\mathcal{O}^{\mathsf{H}}$. Let $(n, t, \mathcal{C}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{H}}}(\mathsf{pp})$ be the output of $\mathcal{A}$. Here $\mathcal{C}$ denotes the (growing) set of signers $\mathcal{A}$ corrupts, and $\mathsf{st}_{\mathcal{A}}$ is some auxiliary information output by $\mathcal{A}$.*

*4. Let $\mathcal{H} = [n] \setminus \mathcal{C}$ be the set of honest signers. For each $i \in \mathcal{H}$, let $\mathsf{st}_i$ be the internal state of signer $i$.*

- **Key Generation**. *Run $\mathsf{KGen}(n,t)$ to generate the keys. Each signer $i \in [n]$ outputs its signing key $\mathsf{sk}_i$, along with the public key $\mathsf{pk}$, and threshold public key $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$. Also, $\mathcal{A}$ learns $\mathsf{sk}_i$ for each $i \in \mathcal{C}$.*
- **Simulating corruption oracle $\mathcal{O}^{\mathsf{Corrupt}}$**. *At any point during the experiment, $\mathcal{A}$ may corrupt a signer $i$ using $\mathcal{O}^{\mathsf{Corrupt}}$. Upon corrupting signer $i$, return internal state $\mathsf{st}_i$ and update $\mathcal{H} := \mathcal{H} \setminus \{i\}, \mathcal{C} := \mathcal{C} \cup \{i\}$.*
- **Simulating signing oracle $\mathcal{O}^{\mathsf{PSign}}$.** *$\mathcal{A}$ queries $\mathcal{O}^{\mathsf{PSign}}$ on input $(i,m)$ to get partial signatures of signer $i \in \mathcal{H}$ on message $m \in \mathcal{M}$. Upon such a query, return $\mathsf{PSign}(\mathsf{sk}_i, m)$, update the internal state $\mathsf{st}_i$, and update $Q[m]$ as $Q[m] := Q[m] \cup \{i\}$.*
- **Simulating random oracle query $\mathcal{O}^{\mathsf{H}}$.** *When $\mathcal{A}$ submits a query $(b,m)$ to $\mathcal{O}^{\mathsf{H}}$ for $m \in \mathcal{M}$ and either $b \in \{0,1\}$, return $\mathsf{H}_b(m)$. Similarly, when $\mathcal{A}$ submits query $x \in \{0,1\}^*$ to $\mathcal{O}^{\mathsf{H}_{\mathbb{F}}}$, return $\mathsf{H}_{\mathbb{F}}(x)$.*
- **Output Determination.** *When $\mathcal{A}$ outputs a message $m^*$ and a signature $\sigma^*$, output 1 if $|Q[m^*] \cup \mathcal{C}| \leq t$ and $\mathsf{Ver}(\mathsf{pk}, m^*, \sigma^*) = 1$. Otherwise, output 0.*

With the $\mathbf{UF\text{-}CMA}^{\mathcal{A}}_{\Sigma}$ game as defined in Figure 1, we define the unforgeability under chosen message attack property of a non-interactive threshold signature scheme as follows.

**Definition 4 (Unforgeability Under Chosen Message Attack).** *We say that $\Sigma$ is a $(\varepsilon, T, q_{\mathsf{H}}, q_{\mathsf{H}_{\mathbb{F}}}, q_s)$-unforgeable under chosen message attacks (UF-CMA) if for all PPT adversaries $\mathcal{A}$ running in time at most $T$, making at most $q_{\mathsf{H}}$ queries to $\mathcal{O}^{\mathsf{H}}$, at most $q_{\mathsf{H}_{\mathbb{F}}}$ queries to $\mathcal{O}^{\mathsf{H}_{\mathbb{F}}}$, and at most $q_s$ queries to $\mathcal{O}^{\mathsf{PSign}}$, $\Pr[\mathbf{UF\text{-}CMA}^{\mathcal{A}}_{\Sigma} = 1] \leq \varepsilon$ for the $\mathbf{UF\text{-}CMA}^{\mathcal{A}}_{\Sigma}$ game.*

### 4.4 Boldyreva's BLS threshold signature scheme

We now describe Boldyreva's BLS-based threshold signature scheme. The public parameters of Boldyerva's BLS signature scheme is a prime order bilinear pairing group $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ with generator $g \in \mathbb{G}$. For any $(n,t)$, the signature scheme works as follows where we use $\mathsf{H} = \mathsf{H}_0$.

- $\mathsf{KGen}$ : On input $(n,t)$, sample a uniformly random polynomial $s(x) \in \mathbb{F}[X]$ of degree $t$. Then, 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 key $\{\mathsf{pk}_i := g^{\mathsf{sk}_i}\}_{i \in [n]}$.
- $\mathsf{PSign}$: On input the message $m \in \mathcal{M}$, signer $i$ with signing key $\mathsf{sk}_i \in \mathbb{F}$ computes its partial signature as $\sigma_i := \mathsf{H}(m)^{\mathsf{sk}_i} \in \hat{\mathbb{G}}$.
- $\mathsf{PVer}$: On input the public key $\mathsf{pk}_i \in \mathbb{G}$, a signature share $\sigma_i$, and a message $m$, return 1 if $e(\mathsf{pk}_i, \mathsf{H}(m)) = e(g, \sigma_i)$, and 0 otherwise.
- $\mathsf{Comb}$: On input a vector of threshold public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, a set $\mathcal{S}$ of $t+1$ signature shares (and corresponding indices) $(\sigma_i, i)$, and a message $m$, run $\mathsf{PVer}(\sigma_i, \mathsf{pk}_i)$ for all $i \in \mathcal{S}_0 := \{i \in [n] \| (\sigma_i, i) \in \mathcal{S}\}$. If any of these calls return 0, return $\perp$. Otherwise, return $\sigma := \prod_{i \in \mathcal{S}_0 \setminus \{i\}} \sigma_i^{L_i}$, where $L_i := \prod_{j \in \mathcal{S}_0 \setminus \{i\}} \left(\frac{j}{j-i}\right)$ denotes the $i$-th Lagrange coefficient for the set $\mathcal{S}_0$.
- $\mathsf{Ver}$: On input a public key $\mathsf{pk}$, a signature $\sigma$, and a message $m$, return 1 if $e(\mathsf{pk}, \mathsf{H}(m)) = e(g, \sigma)$, and 0 otherwise.

## 5 Adaptively Secure BLS Threshold Signature

In this section, we will describe our adaptively secure threshold signature scheme assuming that $\mathsf{KGen}$ is run by a trusted party. Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, g, h, v)$ and the descriptions of $\mathsf{H}_0, \mathsf{H}_1$ and $\mathsf{H}_{\mathbb{F}}$ be the public parameters of our scheme. Here $g, h, v \in \mathbb{G}$ are three uniformly random independent generators of $\mathbb{G}$. $\mathsf{H}_0, \mathsf{H}_1 : \mathcal{M} \to \hat{\mathbb{G}}$ and $\mathsf{H}_{\mathbb{F}} : \{0,1\}^* \to \mathbb{F}$ are three distinct hash functions modelled as random oracles. We summarize our threshold signature scheme in Figure 2, and describe it next.

$\mathsf{KGen}(n,t)$: Given $(n,t)$ and the public parameters, the $\mathsf{KGen}$ algorithm samples three uniformly random polynomials $s(x), r(x)$ and $u(x)$ of degree $t$ 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 $\mathsf{pk} := g^{s(0)}$ 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.

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

1: Let $pp := (\mathbb{F}, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, g, h, v)$
2: Let $\mathsf{H}_0, \mathsf{H}_1 : \mathcal{M} \to \hat{\mathbb{G}}$ and $\mathsf{H}_\mathbb{F} : \{0,1\}^* \to \mathbb{F}$ be three hash functions modeled as random oracle.
3: **return** $(pp, \mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_\mathbb{F})$

// We assume that all algorithms implicitly take the output of the Setup algorithms as input. We use $\mathsf{H}_\mathbb{F}$ in SigmaProve and SigmaVer.

$\underline{\mathsf{KGen}}(n, t)$:

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

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

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** $\sigma_i, \pi_i$

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

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

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

20: Let $T := \{\}$
21: **for** each $i \in S$ **do**
22: **if** $\mathsf{PVer}(m, \mathsf{pk}_i, (\sigma_i, \pi_i))$ **then**
23: $T := T \cup \{(\sigma_i, i)\}$
24: **assert** $|T| \geq t + 1$
25: Let $\ell_i$ be the $i$-th Lagrange coefficients for $S$
26: **return** $\sigma := \prod_{i\in S} \sigma_i^{\ell_i}$

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

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

Fig. 2: Adaptively secure BLS threshold signature scheme with trusted key generation

**Input:** $(g, h, v, \mathsf{pk}) \in \mathbb{G}^4$, $(g_0, g_1) = (\mathsf{H}_0(m), \mathsf{H}_1(m))$ for some fixed $m \in \mathcal{M}$, and $\sigma \in \hat{\mathbb{G}}$
**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 $\sigma = g_0^s g_1^r$.

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

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

1: Let $g_0 := \mathsf{H}_0(m)$ and $g_1 := \mathsf{H}_1(m)$
2: Sample $\hat{s}, \hat{r}, \hat{u} \leftarrow\$ \mathbb{F}$. Let $x := g^{\hat{s}} h^{\hat{r}} v^{\hat{u}}$, and $y := \mathsf{H}_0(m)^{\hat{s}} \mathsf{H}_1(m)^{\hat{r}}$.
3: Let $c := \mathsf{H}_\mathbb{F}(x, y, \mathsf{pk}, \sigma, g_0, g_1)$, where $\mathsf{H}$ is a hash function modeled as a random oracle.
4: Let $z_s := \hat{s} + s \cdot c$, $z_r := \hat{r} + r \cdot c$ and $z_u := \hat{u} + u \cdot c$.
5: **Output** $\pi := (x, y, z_s, z_r, z_u)$.

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

5: Let $g_0 := \mathsf{H}_0(m)$ and $g_1 := \mathsf{H}_1(m)$
6: Let $c := \mathsf{H}_\mathbb{F}(x, y, \mathsf{pk}, \sigma, g_0, g_1)$
7: **if** $g^{z_s} h^{z_r} v^{z_u} = x \mathsf{pk}^c$ and $g_0^{z_s} g_1^{z_r} = y \sigma^c$ **then**
8: **return** 1
9: **return** 0

Fig. 3: Protocol for computing and verifying the the correctness proof for partial signatures.

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

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

Comb: Upon receiving a set of the partial signatures $\{\sigma_i, \pi_i\}$, an aggregator validates them individually using $\mathsf{PVer}$. Let $T$ be the set of indices of parties with valid partial signatures. It then computes the threshold signature $\sigma$ as:

$$\sigma := \prod_{i \in T} \sigma_i^{L_i} \tag{3}$$

where $L_i$ is the $i$-th Lagrange coefficient with respect to the set $T$.

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

**Remark.** Note that $u(i)$ is not used in computing $\sigma_i$. It is in the public verification key (and hence used in computing $\pi_i$) 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 correctness and adaptive security of our threshold signature scheme.

### 6.1 Properties of the Σ-protocol

We require the Σ-protocol satisfy the standard *completeness, knowledge-soundness*, and *zero-knowledge* property [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 prove our Σ protocol to be zero-knowledge against honest verifiers, which 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, which we describe below.

**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 $\hat{c}$ on the same first message, to receive two pairs of valid responses $(z_s, z_r, z_u)$ and $(\hat{z}_s, \hat{z}_r, \hat{z}_u)$. Then, we have:

$$g^{z_s - \hat{z}_s} h^{z_r - \hat{z}_r} v^{z_u - \hat{z}_u} = \mathsf{pk}^{c - \hat{c}} \quad \text{and} \quad \mathsf{H}_0(m)^{z_s - \hat{z}_s} \mathsf{H}_1(m)^{z_r - \hat{z}_r} = \sigma^{c - \hat{c}}$$

$$\implies s = \frac{z_s - \hat{z}_s}{c - \hat{c}}; \quad r = \frac{z_r - \hat{z}_r}{c - \hat{c}}; \quad u = \frac{z_u - \hat{z}_u}{c - \hat{c}}$$

**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 $\mathsf{H}_{\mathbb{F}}(x, y, \mathsf{pk}, \sigma, m) = c$ 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 Correctness

Recall from §4.3 the two requirements for correctness: (1) honestly generated partial signatures are accepted by the $\mathsf{PVer}$ algorithm, and (2) an honest aggregator combines valid partial signatures into a valid threshold signature.

The first property follows from the completeness property of our Σ-protocol. To prove the second property, we will first argue that assuming hardness of discrete logarithm, if for any $i$, $\mathsf{PVer}(m, \sigma_i, \mathsf{pk}_i) = 1$, then $\sigma_i$ has been computed correctly.

**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 $\sigma_i$ along with a valid proof $\pi_i$, then assuming hardness of discrete logarithm in $\mathbb{G}$, $\sigma_i$ is well formed, i.e., $\sigma_i = \mathsf{H}_0(m)^{s(i)}\mathsf{H}_1(m)^{r(i)}$.*

*Proof.* For valid $\Sigma$-protocol proof $\pi_i$, let $\mathcal{E}$ be the extractor we describe in §6.1. Also, let $s', r', u'$ be the extracted witness where $\sigma_i = \mathsf{H}(m)^{s(i)}\mathsf{H}(m)^{r(i)}$. 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 $\theta$. $\mathcal{A}_{\mathrm{DL}}$ picks the other parameter as $g^\alpha$ for some known uniform random $\alpha \in \mathbb{F}$. $\mathcal{A}_{\mathrm{DL}}$ next faithfully emulates the trusted key generation with $\mathcal{A}$ with some chosen polynomials $s(\cdot), r(\cdot), v(\cdot)$.

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 $\mathbb{G}$.

Let $h = g^{\alpha_1}$ and $v = g^{\alpha_2}$, and let $\delta_s := s' - s(i)$, $\delta_r := r' - r(i)$, and $\delta_u := u' - u(i)$. Then, equation (5), implies that $\delta_s + \delta_r\alpha_1 + \delta_u\alpha_2 = 0$. Now, if either $\delta_r$ or $\delta_u$ is non-zero, then we can compute $\alpha_1$ or $\alpha_2$, respectively as:

$$\delta_r \neq 0 \implies \alpha_1 = (-\delta_s - \alpha_2\delta_u) \cdot \delta_r^{-1}; \qquad \delta_u \neq 0 \implies \alpha_2 = (-\delta_s - \alpha_1\delta_r) \cdot \delta_u^{-1} \tag{6}$$

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 $\mathcal{A}$ outputs $(s', r', u') \neq (s(i), r(i), u(i))$ with probability $\varepsilon$, then $\mathcal{A}_{\mathrm{DL}}$ outputs the discrete logarithm of $y$ with respect to $g$, with probability at least $\varepsilon/2$. □

**Theorem 4 (Correctness).** *Our threshold signature scheme is correct as per Definition 3.*

*Proof.* The completeness property of our $\Sigma$-protocol ensures that honestly generated partial signatures are always accepted by $\mathsf{PVer}$. Lemma 1 ensures that the aggregator only aggregates well-formed partial signatures. Thus, the final aggregated signature is:

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

### 6.3 Helper Lemmas for Unforgeability

Before we formally prove the unforgeability of our scheme, we prove the following two helper lemmas. We note that Lemma 2 appears as an exercise in the book [BS23, Exercise 10.8].

**Lemma 2.** *Let $\hat{\mathbb{G}}$ be a cyclic group of prime order $q$ with scalar field $\mathbb{F}$ and $\hat{g} \in \hat{\mathbb{G}}$ as its generator. Let* **DH** *be the set of all Diffie-Hellman triples in $\hat{\mathbb{G}}$, i.e.,*

$$\mathbf{DH} := \{(\hat{g}^a, \hat{g}^b, \hat{g}^{ab}) \in \hat{\mathbb{G}}^3 : a, b \in \mathbb{F}\}$$

*For fixed $a \in \mathbb{F}$, let $\mathbf{T}_a$ be the subset of $\hat{\mathbb{G}}^3$ whose first coordinate is $\hat{g}^a$. Consider the randomized mapping from $\hat{\mathbb{G}}^3$ to $\hat{\mathbb{G}}^3$ that sends $(\hat{g}^a, \hat{g}^b, \hat{g}^z)$ to $(\hat{g}^a, \hat{g}^{b'}, \hat{g}^{z'})$ where*

$$\gamma, \delta \leftarrow\$ \mathbb{F}, \quad \hat{g}^{b'} \leftarrow \hat{g}^\delta \hat{g}^{b\gamma}, \quad \hat{g}^{z'} \leftarrow \hat{g}^{a\delta}\hat{g}^{z\gamma}$$

*Then the following holds.*

*(1) if* $(\hat{g}^a, \hat{g}^b, \hat{g}^z) \in \mathbf{DH}$*, then* $(\hat{g}^a, \hat{g}^{b'}, \hat{g}^{z'})$ *is uniformly distributed over* $\mathbf{DH} \cap \mathbf{T}_a$*;*
*(2) if* $(\hat{g}^a, \hat{g}^b, \hat{g}^z) \notin \mathbf{DH}$*, then* $(\hat{g}^a, \hat{g}^{b'}, \hat{g}^{z'})$ *is uniformly distributed over* $\mathbf{T}_a$*.*

*Proof.* For (1), if $(\hat{g}^a, \hat{g}^b, \hat{g}^z) \in \mathbf{DH}$, then:

$$\hat{g}^{b'} = \hat{g}^{b\gamma+\delta}; \quad \text{and} \quad \hat{g}^{z'} = \hat{g}^{ab\gamma} \cdot \hat{g}^{a\delta} = \hat{g}^{a(b\gamma+\delta)}; \implies (\hat{g}^a, \hat{g}^{b'}, \hat{g}^{z'}) \in \mathbf{DH}. \tag{7}$$

Next, we will prove that $\hat{g}^{b'}$ is an uniformly random element in $\hat{\mathbb{G}}$, which implies that $(\hat{g}^a, \hat{g}^{b'}, \hat{g}^{z'})$ is an uniformly random element in $\mathbf{DH} \cap \mathbf{T}_a$. Note that $\hat{g}_{b'}$ being a uniformly random element in $\hat{\mathbb{G}}$ is equivalent to $b' = b\gamma + \delta$ being a uniformly random element in $\mathbb{F}$. This clearly holds since for any fixed $b$ and $x \in \mathbb{F}$:

$$\Pr_{\gamma,\delta \leftarrow \$ \mathbb{F}}[b\gamma + \delta = x] = \Pr_{\delta \leftarrow \$ \mathbb{F}}[\delta = x - b\gamma] = 1/q$$

For (2), for $(\hat{g}^a, \hat{g}^{b'}, \hat{g}^{z'})$ we have that $b' = b\gamma + \delta$ and $z' = z\gamma + a\delta$. Then, for any $(x, y) \in \mathbb{F}^2$, we will compute the probability that $b' = x$ and $z' = y$. We can re-write these constraints as:

$$\gamma = \frac{y - ax}{z - ab}; \quad \text{and} \quad \delta = \frac{zx - by}{z - ab} \tag{8}$$

Since $z - ab \neq 0$ in (2), eq. (8) is well defined and

$$\Pr_{\gamma,\delta \leftarrow \$ \mathbb{F}}[b\gamma + \delta = x \wedge z\gamma + a\delta = y] = \Pr_{\gamma,\delta \leftarrow \$ \mathbb{F}}[\gamma = \frac{y - ax}{z - ab} \wedge \delta = \frac{zx - by}{z - ab}] = \frac{1}{q^2} \qquad \square$$

**Lemma 3.** *Let* $\hat{\mathbb{G}}$ *be an elliptic curve group with scalar field* $\mathbb{F}$*,* $\hat{g} \in \hat{\mathbb{G}}$ *be a generator in* $\hat{\mathbb{G}}$*,* $\lambda$ *be the security parameter,* $q_{\mathsf{H}} := \mathsf{poly}(\lambda)$ *be an integer. For a fixed* $a, b \leftarrow \$ \mathbb{F}^2$*, let* $(g^a, g^b, g^z) \in \hat{\mathbb{G}}^3$ *for some* $z \in \mathbb{F}$*. Also, for the same* $a$*, let* $\mathbf{T}_a$ *be defined as in Lemma 2. For any fixed* $k \in [q_{\mathsf{H}}]$*, let the distributions* $\mathcal{D}_{1,k}$ *and* $\mathcal{D}_2$ *be defined as follows:*

$$\mathcal{D}_{1,k} := \{(\hat{g}^a, \hat{g}^{b_i}, \hat{g}^{z_i})\}_{i \in [q_{\mathsf{H}}]} \quad \textit{where} \quad \forall i \neq k,\ (\hat{g}^a, \hat{g}^{b_i}, \hat{g}^{z_i}) \leftarrow \$\ \mathbf{T}_a \cap \mathbf{DH};\ b_k, z_k \leftarrow \$\ \mathbb{F} \tag{9}$$

$$\mathcal{D}_2 := \{(\hat{g}^a, \hat{g}^{b_i}, \hat{g}^{z_i})\}_{i \in [q_{\mathsf{H}}]} \quad \textit{where} \quad \forall i \in [q_{\mathsf{H}}], \quad (\hat{g}^a, \hat{g}^{b_i}, \hat{g}^{z_i}) \leftarrow \$\ \mathbf{T}_a \tag{10}$$

*Given the tuple* $(\hat{g}^a, \hat{g}^b, \hat{g}^z)$ *and* $k \in [q_{\mathsf{H}}]$*, now consider the distribution* $\mathcal{D}$ *as defined belwo:*

$$\mathcal{D} := \{(\hat{g}^a, \hat{g}^{b_i}, \hat{g}^{z_i})\}_{i \in [q_{\mathsf{H}}]} \quad \textit{where} \quad \begin{cases} \forall i \neq k, & \hat{g}^{b_i} := \hat{g}^{\delta_i} \hat{g}^{b\gamma_i}, \hat{g}^{z_i} := \hat{g}^{a\delta_i} \hat{g}^{z\gamma_i} \quad \textit{for} \quad \gamma_i, \delta_i \leftarrow \$\ \mathbb{F} \\ i = k, & (\hat{g}^{b_k}, \hat{g}^{z_k}) \leftarrow \$\ \hat{\mathbb{G}}^2 \end{cases} \tag{11}$$

*With the distributions* $\mathcal{D}_{1,k}, \mathcal{D}_2$ *and* $\mathcal{D}$ *as defined above, we have:*

*(1) if* $z = ab$*, then* $\mathcal{D} = \mathcal{D}_{1,k}$
*(2) if* $z \leftarrow \$\ \mathbb{F} \setminus \{ab\}$*, then* $\mathcal{D} = \mathcal{D}_2$

*Proof.* For (1), when $z = ab$, i.e., $(\hat{g}^a, \hat{g}^b, \hat{g}^z) \in \mathbf{DH}$, then by Lemma 2, for all $i \in [q_{\mathsf{H}}]$ and $i \neq k$, $(\hat{g}^a, \hat{g}^{b_i}, \hat{g}^{z_i}) \in \mathcal{D}$ is uniformly distributed in $\mathbf{T}_a \cap \mathbf{DH}$, and hence are identically distributed as in $\mathcal{D}_{1,k}$. By definition, the $k$-th tuple $(\hat{g}^a, \hat{g}^{b_k}, \hat{g}^{z_k})$ in both $\mathcal{D}$ and $\mathcal{D}_{1,k}$ is a uniformly random element in $\mathbf{T}_a$, and hence are identically distributed. This implies that, when $z = ab$, $\mathcal{D}$ is identically distributed as $\mathcal{D}_{1,k}$.

Similarly, for (2), when $z \leftarrow \$\ \mathbb{F} \setminus \{ab\}$, $(\hat{g}^a, \hat{g}^b, \hat{g}^z) \notin \mathbf{DH}$. By Lemma 2, for all $i \in [q_{\mathsf{H}}]$ and $i \neq k$, $(\hat{g}^a, \hat{g}^{b_i}, \hat{g}^{z_i}) \in \mathcal{D}$ is uniformly distributed in $\mathbf{T}_a$. Also, by definition, $(\hat{g}^a, \hat{g}^{b_k}, \hat{g}^{z_k}) \in \mathcal{D}$ is an uniformly random element in $\mathbf{T}_a$. This implies that, when $z \leftarrow \$\ \mathbb{F} \setminus \{ab\}$, $\mathcal{D}$ is identically distributed as $\mathcal{D}_2$. $\square$

An immediate corollary of Lemma 3 is that assuming hardness of DDH in $\hat{\mathbb{G}}$ the distributions $\mathcal{D}_{1,k}$ and $\mathcal{D}_2$ as defined above are computationally indistinguishable.

**Corollary 1.** *Let $\hat{\mathbb{G}}$ be an elliptic curve group with scalar field $\mathbb{F}$ and generator $\hat{g} \in \hat{\mathbb{G}}$. Let $\lambda$ be the security parameter, $q_{\mathsf{H}} := \mathsf{poly}(\lambda)$ be an integer, and for $a \leftarrow\$ \mathbb{F}$, let $\mathbf{T}_a$ be defined as in Lemma 2. Then, for any fixed $k \in [q_{\mathsf{H}}]$, assuming the hardness of DDH in $\hat{\mathbb{G}}$, the distributions $\mathcal{D}_{1,k}$ and $\mathcal{D}_2$ as defined in Lemma 2 are computationally indistinguishable.*

*Proof.* Let $\mathcal{A}$ be the adversary that distinguishes between a sample from $\mathcal{D}_{1,k}$ and $\mathcal{D}_2$ with probability $\varepsilon$, then we can build a DDH adversary $\mathcal{A}_{\text{DDH}}$ with advantage $\varepsilon$ as follows. $\mathcal{A}_{\text{DDH}}$ on given a DDH input $(\hat{g}, \hat{g}^a, \hat{g}^b, \hat{g}^z)$, uses the randomization in equation (11) to sample a vector of tuples from distribution $\mathcal{D}$. $\mathcal{A}_{\text{DDH}}$ then runs $\mathcal{A}$ on the sampled vector and outputs what $\mathcal{A}$ outputs.

Now, from Lemma 3, depending upon whether $z = ab$ or $z \leftarrow\$ \mathbb{F} \setminus \{ab\}$, the sampled vector is either from distribution $\mathcal{D}_{1,k}$, or from distribution $\mathcal{D}_2$. This implies that for a DDH input $(g, g^a, g^b, g^z)$ with $z = ab$, $\mathcal{A}_{\text{DDH}}$ always outputs a sample from $\mathcal{D}_{1,k}$. Alternatively, when $z \leftarrow\$ \mathbb{F}$, except for the rare event where $z = ab$, $\mathcal{A}_{\text{DDH}}$ outputs a sample from $\mathcal{D}_2$. Since $\Pr_{z \leftarrow\$ \mathbb{F}}[z = ab] = 1/|\mathbb{F}|$, which is negligible, this implies that the distinguishing advantage of $\mathcal{A}_{\text{DDH}}$ is at least $\varepsilon - \varepsilon/|\mathbb{F}|$. Since we assume DDH is hard in $\hat{\mathbb{G}}$, it implies $\varepsilon - \varepsilon/|\mathbb{F}| \leq \varepsilon_{\text{DDH}}$. So $\varepsilon \leq \varepsilon_{\text{DDH}} + \varepsilon/|\mathbb{F}|$, and the two distributions $\mathcal{D}_{1,k}$ and $\mathcal{D}_2$ are indistinguishable.

Since $\varepsilon/|\mathbb{F}|$ is negligible, we will ignore this term for brevity, and simply say that the distinguishing advantage of $\mathcal{A}$ is at most $\varepsilon_{\text{DDH}}$.

### 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 $(g, \hat{g}, g^a, \hat{g}^a, \hat{g}^b)$, $\mathcal{A}_{\text{co-CDH}}$ simulates the trusted key generation functionality and the 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 $\hat{g}^{ab}$. We summarize $\mathcal{A}_{\text{co-CDH}}$'s interaction with $\mathcal{A}$ in Figure 4, and describe it next.

**Simulating the $\mathsf{KGen}$ functionality.** $\mathcal{A}_{\text{co-CDH}}$ samples $\alpha_2 \leftarrow\$ \mathbb{F}$ and sets $h := g^a$ and $v := g^{\alpha_2}$. $\mathcal{A}_{\text{co-CDH}}$ samples three uniformly random polynomials $s(\cdot), r(\cdot), u(\cdot) \in \mathbb{F}[x]$ of degree $t$ each, but crucially with the constraints that $r := r(0) \neq 0$. Let $\mathcal{H}$ and $\mathcal{C}$ be the set of honest and malicious parties, respectively. $\mathcal{A}_{\text{co-CDH}}$ then computes the public and threshold public keys as follows:

$$\mathsf{pk} := g^s h^r v^u; \quad \text{and} \quad \left\{ \mathsf{pk}_i := g^{s(i)} h^{r(i)} v^{u(i)} \right\}_{i \in [n]} \tag{14}$$

To each honest node $i \in \mathcal{H}$, $\mathcal{A}_{\text{co-CDH}}$ sends $\mathsf{sk}_i$ along with $\mathsf{pk}, \{\mathsf{pk}_j\}_{j \in [n]}$. $\mathcal{A}_{\text{co-CDH}}$ sends $\{\mathsf{sk}_i\}_{i \in \mathcal{C}}$ along with $\mathsf{pk}, \{\mathsf{pk}_j\}_{j \in [n]}$ to $\mathcal{A}$.

**Simulating threshold signature.** Anytime during the signing phase, if $\mathcal{A}$ corrupts node $i \in \mathcal{H}$, then $\mathcal{A}_{\text{co-CDH}}$ faithfully reveals the internal state of party $i$, including its secret signing key $\mathsf{sk}_i := (s(i), r(i), u(i))$, and updates $\mathcal{C} := \mathcal{C} \cup \{i\}$ and $\mathcal{H} := \mathcal{H} \setminus \{i\}$.

$\mathcal{A}_{\text{co-CDH}}$ simulates the signing queries by programming the random oracles as follows. At the start of the signing phase, $\mathcal{A}_{\text{co-CDH}}$ samples a random $\beta \in \mathbb{F}$. Let $\alpha = a + \alpha_2 \cdot (u/r)$. Note that $\mathsf{H}_0$ is always queried on the forged message, at least by $\mathcal{A}_{\text{co-CDH}}$ during the signature verification. Moreover, whenever $\mathcal{A}$ queries $\mathsf{H}_\theta$ 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 $q_{\mathsf{H}}$ be an upper bound on the number of queries by $\mathcal{A}$ to $\mathsf{H}_0$ and $\mathsf{H}_1$ 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.

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

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

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 $(g, g^a, \hat{g}, \hat{g}^a, \hat{g}^b) \in \mathbb{G}^3 \times \hat{\mathbb{G}}$.

**$\mathsf{KGen}$ simulation:**

1. Sample $\alpha_2 \leftarrow\$ \mathbb{F}$. Let $h := g^a$ and $v := g^{\alpha_2}$.
2. Sample three degree $t$ uniformly random polynomial $s(x), r(x), u(x) \in \mathbb{F}[x]$ with $r(0) \neq 0$.
3. Let $\mathcal{H}$ and $\mathcal{C}$ be the set of honest and malicious parties, respectively.
4. Let $s := s(0), r := r(0)$ and $u := u(0)$. Compute $\mathsf{pk} := g^s h^r v^u$, and for each $i \in [n]$, $\mathsf{pk}_i := g^{s(i)} h^{r(i)} v^{u(i)}$
5. For each $i \in [n]$, let $\mathsf{sk}_i := (s(i), r(i), u(i))$. Send to $\mathsf{sk}_i, \mathsf{pk}, \{\mathsf{pk}_j\}_{j\in[n]}$ to each honest signer $i \in \mathcal{H}$. Send $\{\mathsf{sk}_i\}_{i\in\mathcal{C}}$ along with $\mathsf{pk}, \{\mathsf{pk}_j\}_{j\in[n]}$ to $\mathcal{A}$.

**Corruption simulation:**

6. If $\mathcal{A}$ corrupts a signer $i \in \mathcal{H}$, send the internal state of signer $i$ including $\mathsf{sk}_i$ to $\mathcal{A}$. Update $\mathcal{H} := \mathcal{H} \setminus \{i\}$ and $\mathcal{C} := \mathcal{C} \cup \{i\}$.

**Threshold signature simulation:**

7. On a query to random oracle $\mathsf{H}_{\mathbb{F}}$ on input $x$, if $\mathsf{H}_{\mathbb{F}}(x) \neq \perp$, set $\mathsf{H}_{\mathbb{F}}(x) \leftarrow\$ \mathbb{F}$. Output $\mathsf{H}_{\mathbb{F}}(x)$.
8. Sample $\beta \leftarrow\$ \mathbb{F}$ and let $\alpha := a + \alpha_2 \cdot (u/r)$.
9. Let $q_{\mathsf{H}}$ be an upper bound on the total number of random oracle queries to $\mathsf{H}_0$ and $\mathsf{H}_1$,
10. Sample $\hat{k} \leftarrow\$ [q_{\mathsf{H}}]$.
11. On $k$-th random oracle query to $\mathsf{H}_\theta$ 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 $\mathsf{H}_\theta(m_k)$.

$$\mathsf{H}_0(m_k) := \hat{g}^{\beta\gamma_k + \delta_k};\ \mathsf{H}_1(m_k) := \hat{g}^{\alpha\cdot(\beta\gamma_k+\delta_k)} \text{ for } \gamma_k, \delta_k \leftarrow\$ \mathbb{F} \tag{12}$$

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

$$\mathsf{H}_0(m_k) := \hat{g}^b; \quad \mathsf{H}_1(m_k) := \hat{g}' \text{ for } \hat{g}' \leftarrow\$ \hat{\mathbb{G}} \tag{13}$$

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:**

13. Let $\sigma$ be the valid forgery on $m_{\hat{k}}$, then output $\sigma^{r^{-1}} \cdot \hat{g}^{-br^{-1}(s+\alpha_2 u)}$ as the co-CDH output.

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

**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 $g^s h^r v^u$ where $\mathcal{A}_{\text{co-CDH}}$ knows $(s, r, u)$. Then, $\mathcal{A}_{\text{co-CDH}}$ computes the co-CDH output $\hat{g}_{\text{cdh}}$ as

$$\hat{g}_{\text{cdh}} := \sigma^{r^{-1}} \cdot \hat{g}^{-br^{-1}(s+\alpha_2 u)} \tag{17}$$

**Lemma 4.** *If $\sigma$ is a valid forgery on message $m_{\hat{k}}$, then $\hat{g}_{\text{cdh}}$ is a valid co-CDH output.*

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

$$e(\mathsf{pk}, \mathsf{H}_0(m_{\hat{k}})) = e(g, \sigma) \implies e(g^s h^r v^u, g^b) = e(g, \sigma) \tag{18}$$

Let $\hat{h} = \hat{g}^a$ and $\hat{v} = \hat{g}^{\alpha_2}$. Then, from equation (18), we get that:

$$\begin{aligned} &\sigma = \left(\hat{g}^s \hat{h}^r \hat{v}^u\right)^b \\ \implies &\sigma \cdot \hat{g}^{-b(s+\alpha_2 u)} = \hat{h}^{br} \\ \implies &\sigma^{r^{-1}} \cdot \hat{g}^{-br^{-1}(s+\alpha_2 u)} = \hat{h}^b = \hat{g}^{ab} \end{aligned}$$ □

Next, we prove that assuming the hardness of DDH in $\hat{\mathbb{G}}$, if $\mathcal{A}$ forges a signature in the real protocol with probability $\varepsilon_\sigma$, then $\mathcal{A}$ also forges a signature in the simulated protocol with probability at least $\varepsilon_\sigma - \varepsilon_{\text{DDH}}$. Here, $\varepsilon_{\text{DDH}}$ is the advantage of any adversary in breaking DDH in $\hat{\mathbb{G}}$.

**Lemma 5.** *Given pairing groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$, if a PPT adversary $\mathcal{A}$ forges a signature in the real protocol with probability $\varepsilon_\sigma$, then $\mathcal{A}$ also forges a signature in the simulated protocol with probability at least $\varepsilon_\sigma - \varepsilon_{\text{DDH}}$. Here $\varepsilon_{\text{DDH}}$ is the advantage of $\mathcal{A}$ in breaking the DDH assumption in $\hat{\mathbb{G}}$.*

We will prove this lemma via a sequence of games. Game $\mathbf{G}_0$ is the real protocol execution, and game $\mathbf{G}_8$ is the interaction of $\mathcal{A}$ with $\mathcal{A}_{\text{co-CDH}}$.

GAME $\mathbf{G}_0$: This is the real execution of the protocol, as described in Figure 2. Then by definition, $\mathcal{A}$ outputs a forgery in $\mathbf{G}_0$ with probability $\Pr[\mathbf{G}_0 = 1] = \varepsilon_\sigma$.

GAME $\mathbf{G}_1$: This game is identical to $\mathbf{G}_0$, except that we sample $\alpha_1, \alpha_2 \leftarrow\$ \mathbb{F}$ and set $h = g^{\alpha_1}$ and $v = g^{\alpha_2}$. Clearly, the view of $\mathcal{A}$ in $\mathbf{G}_0$ is identical to its view in $\mathbf{G}_1$, hence $\Pr[\mathbf{G}_0 = 1] = \Pr[\mathbf{G}_1 = 1]$.

GAME $\mathbf{G}_2$: This game is identical to $\mathbf{G}_1$, except that we program the random oracles $\mathsf{H}_0$ and $\mathsf{H}_1$ as in steps 7 to 10 in Figure 4. In particular, we sample $\beta, \phi \leftarrow\$ \mathbb{F}^2$ and compute $\alpha := \alpha_1 + \alpha_2\phi$. Let $q_{\mathsf{H}}$ be the upper-bound on the total number of random oracle queries to $\mathsf{H}_0$ and $\mathsf{H}_1$. We then sample $\hat{k} \leftarrow\$ [q_{\mathsf{H}}]$ and program the random oracles on the $k$-th query as in equation (16) or equation (15) with $a := \alpha_1$, depending on whether $k = \hat{k}$. We next bound the probability $\Pr[\mathbf{G}_2 = 1]$ assuming the hardness of DDH in $\hat{\mathbb{G}}$.

**Lemma 6.** *Let $\varepsilon_{\text{DDH}}$ be the advantage of breaking DDH in $\hat{\mathbb{G}}$ as defined in Assumption 1, then*

$$|\Pr[\mathbf{G}_1 = 1] - \Pr[\mathbf{G}_2 = 1]| \leq \varepsilon_{\text{DDH}} \tag{19}$$

*Proof.* Observe that in the game $\mathbf{G}_2$, we program the random oracles $\mathsf{H}_0$ and $\mathsf{H}_1$ exactly with a sample from the distribution $\mathcal{D}_{1,\hat{k}}$ as we define in Lemma 3. Similarly, in $\mathbf{G}_2$, we program the random random oracles $\mathsf{H}_0$ and $\mathsf{H}_1$ with a sample from $\mathcal{D}_2$, we define in Lemma 3. Apart from the output of the random oracles $\mathsf{H}_0$ and $\mathsf{H}_1$, the rest of the view is identically distributed in $\mathbf{G}_1$ and $\mathbf{G}_2$.

We will prove this lemma by showing that if any adversary $\mathcal{A}$ can distinguish between its views in $\mathbf{G}_1$ and $\mathbf{G}_2$ with probability $\varepsilon$, then we can build a DDH adversary $\mathcal{A}_{\text{DDH}}$ who uses $\mathcal{A}$ to break the DDH assumption with advantage $\varepsilon$. This implies, by assumption 1, that $\varepsilon \leq \varepsilon_{\text{DDH}}$.

Let $\mathcal{A}_{\text{DDH}}$ be the DDH adversary. $\mathcal{A}_{\text{DDH}}$ on input $(\hat{g}, \hat{g}^a, \hat{g}^b, \hat{g}^z)$ interacts with $\mathcal{A}$ as follows. $\mathcal{A}_{\text{DDH}}$ samples $g \leftarrow\$ \mathbb{G}$, $\alpha_1, \alpha_2 \leftarrow\$ \mathbb{F}$ and sets $h = g^{\alpha_1}$ and $v = g^{\alpha_2}$. $\mathcal{A}_{\text{DDH}}$ then generates the signing and public keys by running the $\mathsf{KGen}$ functionality as in Figure 2. $\mathcal{A}_{\text{DDH}}$ then faithfully follows the rest of the protocol, except programming the random oracle as follows.

Let $a = \alpha_1 + \alpha_2\phi$ for some $\phi \in \mathbb{F}$ unknown to $\mathcal{A}_{\text{DDH}}$. $\mathcal{A}_{\text{DDH}}$ samples $\hat{k} \leftarrow\$ [q_{\mathsf{H}}]$, and programs the random oracle as follows. On the $k$-th query to $\mathsf{H}_\theta$ for $\theta \in \{0, 1\}$ on message $m_k$, if $\mathsf{H}_\theta(m_k) \neq \perp$, then output $\mathsf{H}_\theta(m_k)$. Otherwise, depending on whether $k = \hat{k}$ or not, set:

$$k \neq \hat{k} \implies \mathsf{H}_0(m_k) := \hat{g}^{b\gamma_k + \delta_k};\ \mathsf{H}_1(m_k) := \hat{g}^{z\gamma_k + a\delta_k} \text{ for } \gamma_k, \delta_k \leftarrow\$ \mathbb{F} \tag{20}$$

$$k = \hat{k} \implies \mathsf{H}_0(m_k) := \hat{g}';\quad \mathsf{H}_1(m_k) := \hat{g}'' \text{ for } \hat{g}', \hat{g}'' \leftarrow\$ \hat{\mathbb{G}} \tag{21}$$

$\mathcal{A}_{\text{DDH}}$ then outputs 1 whenever $\mathcal{A}$ forges a signature, otherwise outputs 0.

It follows from Lemma 3, for DDH input $(\hat{g}, \hat{g}^a, \hat{g}^b, \hat{g}^z)$, depending upon whether $z = ab$ or $z \leftarrow\$ \mathbb{F} \setminus \{ab\}$, $\mathcal{A}_{\text{DDH}}$ either programs the random oracles with a sample from $\mathcal{D}_{1,\hat{k}}$ or a sample from $\mathcal{D}_2$, respectively. Thus, when $z = ab$, $\mathcal{A}$'s view in interaction with $\mathcal{A}_{\text{DDH}}$ is identically distributed as in $\mathbf{G}_2$. Alternatively, when $z \leftarrow\$ \mathbb{F} \setminus \{ab\}$, $\mathcal{A}$'s view is identically distributed as in $\mathbf{G}_1$. Recall from Corollary 1, assuming hardness of DDH in $\hat{\mathbb{G}}$, samples from distributions $\mathcal{D}_{1,\hat{k}}$ and $\mathcal{D}_2$ are computationally indistinguishable. Since games $\mathbf{G}_1$ and $\mathbf{G}_2$ only differ in whether we program the random oracles $\mathsf{H}_0$ and $\mathsf{H}_1$, using a sample from $\mathcal{D}_{1,k}$ or $\mathcal{D}_2$, we get that:

$$|\Pr[\mathbf{G}_1 = 1] - \Pr[\mathbf{G}_2 = 1]| = \varepsilon \leq \varepsilon_{\text{DDH}} \tag{22}$$

GAME $\mathbf{G}_3$: This game is identical to $\mathbf{G}_2$, except that we sample $r \leftarrow\$ \mathbb{F}\setminus\{0\}$ and $u \leftarrow\$ \mathbb{F}$. Compute $\phi := u/r$, instead of sampling it directly from $\mathbb{F}$, and hence $\Pr[\mathbf{G}_2 = 1] = \Pr[\mathbf{G}_3 = 1]$.

GAME $\mathbf{G}_4$: This game is identical to $\mathbf{G}_3$, except that for each honest signer we use simulated proofs for correctness of partial signatures instead of actual NIZK proofs. During NIZK simulation, we program the random oracle $\mathsf{H}_{\mathbb{F}}$ on $(x, y, \mathsf{pk}, \sigma, m)$ at a choice of our challenge. This step will fail if the adversary has already queried $\mathsf{H}_{\mathbb{F}}$ on this particular input. Let $E$ be the event that at least one of our simulation queries collides with $\mathcal{A}$'s random oracle query. If event $E$ occurs, then we abort. Otherwise, if no such collision occurs, then $\mathcal{A}$'s view in $\mathbf{G}_3$ is identically distributed as its view in $\mathbf{G}_4$. This implies that:

$$|\Pr[\mathbf{G}_3 = 1] - \Pr[\mathbf{G}_4 = 1]| = |\Pr[\mathbf{G}_3 = 1|E] - \Pr[\mathbf{G}_4 = 1|E]| \cdot \Pr[E] \leq \Pr[E] \tag{23}$$

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

We now analyze the probability of event $E$. For each simulated proof, we need to program the random oracle at a tuple $(x, y, \mathsf{pk}, \sigma, g_0, g_1)$. Since $x$ and $y$ are chosen uniformly at random from $(\mathbb{G} \times \mathbb{G})$, and $\mathcal{A}$ makes at most $q_{\mathsf{H}_{\mathbb{F}}}$ queries to $\mathsf{H}_{\mathbb{F}}$, the probability of colliding with an adversarial query during simulation of a single partial signature is $q_{\mathsf{H}_{\mathbb{F}}}/|\mathbb{F}|^2$. 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

$$\Pr[E] \leq \frac{q_{\mathsf{H}_{\mathbb{F}}} \cdot q_s \cdot n}{|\mathbb{F}|^2} = \varepsilon_{\text{nizk-fail}}. \tag{24}$$

Hence, we get:

$$|\Pr[\mathbf{G}_3 = 1] - \Pr[\mathbf{G}_4 = 1]| \leq \frac{q_{\mathsf{H}_{\mathbb{F}}} \cdot q_s \cdot n}{|\mathbb{F}|^2} = \varepsilon_{\text{nizk-fail}}. \tag{25}$$

GAME $\mathbf{G}_5$: This game is identical to $\mathbf{G}_4$, except that during $\mathsf{KGen}$, we use a polynomial $s(x)$ such that

$$s(0) = s_0 + r\alpha_1 + u\alpha_2, \quad \text{for} \quad s_0 \leftarrow\$ \mathbb{F} \tag{26}$$

Observe that for any fixed $(\alpha_1, \alpha_2, r, u)$, since $s_0$ is chosen uniformly at random, $s_0 + r\alpha_1 + u\alpha_2$ is also uniformly random. Hence, $\mathcal{A}$'s view in game $\mathbf{G}_5$ is identical to its view in game $\mathbf{G}_4$, and $\Pr[\mathbf{G}_4 = 1] = \Pr[\mathbf{G}_5 = 1]$.

GAME $\mathbf{G}_6$: This game is identical to $\mathbf{G}_5$, except that we choose polynomials $s(x), r(x)$ and $u(x)$ such that

$$s(0) = s_0, \quad r(0) = r, \quad u(0) = u \tag{27}$$

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 7.** $\Pr[\mathbf{G}_5 = 1] = \Pr[\mathbf{G}_6 = 1]$

*Proof.* It is easy to see that the public key $\mathsf{pk}$ is identically distributed in $\mathbf{G}_5$ and $\mathbf{G}_6$. For any signer $i$, let $\mathsf{pk}_{i,\mathbf{G}_5}$ and $\mathsf{pk}_{i,\mathbf{G}_6}$, be its threshold public key in $\mathbf{G}_5$ and $\mathbf{G}_6$, respectively. Then, since $h = g^{\alpha_1}$ and $v = g^{\alpha_2}$, we have:

$$\mathsf{pk}_{i,\mathbf{G}_5} = g^{s(i)+\alpha_1 r+\alpha_2 u} \cdot h^{r(i)} \cdot v^{u(i)} = g^{s(i)} \cdot h^{r(i)+r} \cdot v^{u(i)+u} = \mathsf{pk}_{i,\mathbf{G}_6} \tag{28}$$

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:

$$\mathsf{H}_0(m_k) = \hat{g}^{\beta_k}; \qquad \text{and} \qquad \mathsf{H}_1(m_k) = \hat{g}^{\alpha \cdot \beta_k}, \text{ where } \beta_k := (\beta\gamma_k + \delta_k) \tag{29}$$

This implies that,

$$\begin{aligned}
\sigma_{i,\mathbf{G}_5} = \mathsf{H}_0(m_k)^{s(i)+r\alpha_1+u\alpha_2}\mathsf{H}_1(m_k)^{r(i)} &= g^{\beta_k \cdot (s(i)+\alpha_1 r+\alpha_2 u)} \cdot g^{\alpha\beta_k r(i)} \\
&= g^{\beta_k s(i)} \cdot g^{\beta_k r \cdot (\alpha_1+\alpha_2 u/r)} \cdot g^{\alpha\beta_k r(i)} \\
&= g^{\beta_k s(i)} \cdot g^{\beta_k r\alpha} \cdot g^{\alpha\beta_k r(i)} \\
&= g^{\beta_k s(i)} \cdot g^{\alpha\beta_k \cdot (r(i)+r)} \\
&= \mathsf{H}_0(m)^{s(i)}\mathsf{H}_1(m)^{r+r(i)} = \sigma_{i,\mathbf{G}_6}
\end{aligned} \tag{30}$$

Equations (28) and (30) 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 parties, 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 parties 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 parties. Let $Q[m_{\hat{k}}] \subset \mathcal{H}$ be the subset of honest parties $\mathcal{A}$ 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 $\alpha_1, \alpha_2, r, u$, let $\mathcal{D}_5$ and $\mathcal{D}_6$ be the view of $\mathcal{A}$ in game $\mathbf{G}_5$ and $\mathbf{G}_6$, respectively, i.e.,

$$\mathcal{D}_5 = \left( \left\{ \hat{g}_0^{s(i)+r\alpha_1+u\alpha_2} \cdot \hat{g}_1^{r(i)} \right\}_{i \in Q[m_{\hat{k}}]}, \{ s(i) + r\alpha_1 + u\alpha_2, \quad r(i), \quad u(i) \}_{i \in \mathcal{C}} \right)$$

$$\mathcal{D}_6 = \left( \left\{ \hat{g}_0^{s(i)} \cdot \hat{g}_1^{r(i)+r} \right\}_{i \in Q[m_{\hat{k}}]}, \{ s(i), \quad r(i) + r, \quad u(i) + u \}_{k \in \mathcal{C}} \right)$$

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:

$$\mathcal{D}_{5,t} = \left( \{ s(i) + r\alpha_1 + u\alpha_2, \quad r(k), \quad u(k) \}_{k \in \mathcal{C} \cup Q[m_{\hat{k}}]} \right)$$

$$\mathcal{D}_{6,t} = \left( \{ s(k), \quad r(k) + r, \quad u(k) + u \}_{k \in \mathcal{C} \cup Q[m_{\hat{k}}]} \right)$$

Observe that the distributionss $\mathcal{D}_{5,t}$ 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 $\Pr[\mathbf{G}_5 = 1] = \Pr[\mathbf{G}_6 = 1]$. □

GAME $\mathbf{G}_7$: This game is identical to $\mathbf{G}_6$, except that we use $h = g^a$, i.e., $\alpha = a$ from the co-CDH input, and use $\hat{g}^a$ to compute the random oracle outputs in step 10(b) in Figure 4. Clearly, $\mathcal{A}$'s view in game $\mathbf{G}_7$ is identical to its view in game $\mathbf{G}_6$, and hence $\Pr[\mathbf{G}_6 = 1] = \Pr[\mathbf{G}_7 = 1]$.

GAME $\mathbf{G}_8$: This game is identical to $\mathbf{G}_7$, except that we use actual NIZK proofs for partial signatures. Thus, using an argument similar as in the advantage of $\mathcal{A}$ between $\mathbf{G}_3$ and $\mathbf{G}_4$, we get that:

$$|\Pr[\mathbf{G}_7 = 1] - \Pr[\mathbf{G}_6 = 8]| \leq \varepsilon_{\text{nizk-fail}}. \tag{31}$$

Observe that game $\mathbf{G}_8$ is exactly the $\mathcal{A}$'s interaction with $\mathcal{A}_{\text{co-CDH}}$. Hence, from the above sequence of games, we get that:

$$|\Pr[\mathbf{G}_0 = 1] - \Pr[\mathbf{G}_8 = 1]| \leq \varepsilon_{\text{DDH}} - 2\varepsilon_{\text{nizk-fail}} \implies \Pr[\mathbf{G}_8 = 1] \geq \varepsilon_\sigma - \varepsilon_{\text{DDH}} - 2\varepsilon_{\text{nizk-fail}}. \tag{32}$$

This implies that any adversary $\mathcal{A}$ outputs a forgery in the real execution of our signature scheme (i.e., $\mathbf{G}_0$) with probability $\varepsilon_\sigma$, then $\mathcal{A}$ outputs a forgery during its interaction with $\mathcal{A}_{\text{co-CDH}}$ (i.e., $\mathbf{G}_8$) with probability at least $\varepsilon_\sigma - \varepsilon_{\text{DDH}} - 2\varepsilon_{\text{nizk-fail}}$. This allows us to prove the following main theorem.

**Theorem 5 (Adaptively secure BLS threshold signature).** *Let $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ be a pairing group of prime order $q$. Let $\lambda$ be the security parameter. 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, the threshold signature scheme in §5 is $(\varepsilon_\sigma, T, q_{\mathsf{H}}, q_{\mathsf{H}_\mathbb{F}}, q_s)$ unforgeable under chosen message attack as per Definition 4 against an adaptive adversary $\mathcal{A}$ corrupting up to $t$ signers where:*

$$\varepsilon_\sigma \leq \varepsilon_{\text{DDH}} + 2\varepsilon_{\text{nizk-fail}} + q_{\mathsf{H}} \cdot \varepsilon_{\text{CDH}},$$

*$\varepsilon_{\text{nizk-fail}} = (q_{\mathsf{H}_\mathbb{F}} \cdot q_s \cdot n)/|\mathbb{F}|^2$, and $\varepsilon_{\text{DDH}}$ and $\varepsilon_{\text{CDH}}$ 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.*

*Proof.* $\mathcal{A}_{\text{co-CDH}}$ successfully computes the co-CDH output whenver $\mathcal{A}$ forges a signature in game $\mathbf{G}_8$ on a message $m$ such that $m = m_{\hat{k}}$. Let $E$ be the event where $m = m_{\hat{k}}$. Since $\mathcal{A}_{\text{co-CDH}}$ chooses $m_{\hat{k}}$ uniformly at random among all messages $\mathcal{A}$ queries the random oracles with, $\Pr[E]$ is at least $1/q_{\mathsf{H}}$. Moreover, $E$ is independent of the event $\mathbf{G}_8 = 1$. Combined with Lemma 5, we get

$$\begin{aligned}
\varepsilon_{\text{CDH}} &\geq \Pr[m_{\hat{k}} = m \wedge \mathbf{G}_8 = 1] \\
&= \Pr[m_{\hat{k}} = m | \mathbf{G}_8 = 1] \cdot \Pr[\mathbf{G}_8 = 1] \\
&\geq 1/q_{\mathsf{H}} \cdot (\varepsilon_{\sigma} - \varepsilon_{\text{DDH}} - 2\varepsilon_{\text{nizk-fail}}) \qquad (33) \\
\implies \varepsilon_{\sigma} &\leq \varepsilon_{\text{DDH}} + 2\varepsilon_{\text{nizk-fail}} + q_{\mathsf{H}} \cdot \varepsilon_{\text{CDH}} \qquad (34)
\end{aligned}$$

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

### 6.5 Unforgeability with static adversary

We now briefly argue that our signature scheme is statically secure, assuming 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 that $\mathbb{G} = \hat{\mathbb{G}}$ in this analysis, and hence the CDH assumption instead of co-CDH. Our security proof is similar to the static security proof of Boldyreva's scheme.

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. 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 $\mathcal{A}_{\text{CDH}}$. $\mathcal{A}_{\text{CDH}}$ on input a CDH input $(g, g^a, g^b) \in \mathbb{G}^3$ simulates the $\mathsf{KGen}$ functionality and the signature scheme with $\mathcal{A}_{\text{static}}$ as follows.

**Simulating the $\mathsf{KGen}$ functionality.** For simplicity let us assume that $|\mathcal{C} \cup \mathcal{S}| = t$ $\mathcal{A}_{\text{CDH}}$ samples $h, v \leftarrow\$ \mathbb{G}$. Next, $\mathcal{A}_{\text{CDH}}$ samples two random degree $t$ polynomials $r(x), u(x)$ with the constraint $r_i(0) = u_i(0) = 0$. To compute the polynomial $s(x)$, $\mathcal{A}_{\text{static}}$ samples $s(j) \leftarrow\$ \mathbb{F}$ for each $j \in \mathcal{C} \cup \mathcal{S}$. $\mathcal{A}_{\text{CDH}}$ then computes the public key as $\mathsf{pk} = g^a$ and threshold public keys $\{\mathsf{pk}_i\} = \{g^{s(i)}\}_{i \in [n]}$ using interpolation in the exponent. $\mathcal{A}_{\text{CDH}}$ 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 $\mathcal{A}_{\text{CDH}}$ always faithfully responds to queries to $\mathsf{H}_1$. Note that $\mathsf{H}_0$ is always queried on the forged message, at least by $\mathcal{A}_{\text{CDH}}$ during the signature verification. Let $q_{\mathsf{H}}$ be an upper bound on the number of random oracle queries to $\mathsf{H}_0$, including the query during the signature verification. For static security, the number of queries to $\mathsf{H}_1$ can be unbounded. $\mathcal{A}_{\text{CDH}}$ samples $\hat{k} \leftarrow\$ [q_{\mathsf{H}}]$. On the $k$-th random oracle query on message $m_k$, depending upon the value of $k$, $\mathcal{A}_{\text{CDH}}$ programs the random oracle as follows:

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

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 $k \neq \hat{k}$, $\mathcal{A}_{\text{CDH}}$ uses its knowledge of $\gamma_k$ 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, $\mathcal{A}_{\text{CDH}}$ 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 $\mathcal{C} \cup \mathcal{S}$, $\mathcal{A}_{\text{CDH}}$ aborts.

Now, whenever $\mathcal{A}_{\text{static}}$ forges a signature on $m_{\hat{k}}$, i.e., outputs a valid signature $\sigma^*$, $\mathcal{A}_{\text{CDH}}$ also outputs $\sigma^*$. It is easy to see that $\sigma^* = g^{ab}$.

## 7 Distributed Key Generation (DKG) Design and Analysis

In this section, we present a DKG protocol that signers can run to set up the signing keys of our threshold signature scheme, instead of relying on the trusted $\mathsf{KGen}$. It has the following interface.

$\underline{\mathsf{DKG}(n,t)}$: For any $(n,t)$, $\mathsf{DKG}$ is an interactive protocol among $n$ parties, which all take as inputs some public parameters, a security parameter $\lambda \in \mathbb{N}$, as well as a pair of integers $n, t \in \mathsf{poly}(\lambda)$ with $t < n/2$. At the end of the protocol, signers output a public key $\mathsf{pk}$, a vector of threshold public keys $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$. Each signer $i$ additionally outputs a secret key share $\mathsf{sk}_i$.

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

We require the DKG protocol to satisfy *correctness*. Intuitively, the correctness property 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.

**Definition 5 (Correctness).** *A DKG protocol* $\mathsf{DKG}$ *is correct, if for all security parameters* $\lambda$*, all allowable* $t < n/2$*, and all PPT adversary* $\mathcal{A}$ *that can adaptively corrupt up to* $t$ *parties during the DKG protocol, the following holds:*

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

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

Additionally, 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 non-zero $r(0)$. However, with DKG, we do not have a trusted entity to set up the rigged public key. Instead, we will rely on the *single inconsistent party* (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 real execution of the protocol where all parties are honest and the execution with a single inconsistent party. We capture this by requiring the DKG protocol to satisfy the SIP simulatability property we define below.

**Definition 6 (SIP Simulatability).** *For security parameter* $\lambda$*, 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 $\mathsf{pk} = g^s h^r v^u$ for some $r \leftarrow\!\!\$\ \mathbb{F} \setminus \{0\}$.

### 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 that we can use many other DKG protocols using a similar modification (see our discussion towards the end of this section). We summarize our protocol in Figure 5 and describe it next.

Let $g, h, v \in \mathbb{G}$ be three uniform random generators of $\mathbb{G}$ with 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

$$s_i(x) = s_{i,0} + s_{i,1}x + \cdots + s_{i,t}x^t; \quad r_i(x) = r_{i,1}x + \cdots + r_{i,t}x^t; \quad u_i(x) = u_{i,1}x + \cdots + u_{i,t}x^t \tag{38}$$

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

**Sharing phase:**

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

$$s_i(x) = s_{i,0} + s_{i,1}x + \cdots + s_{i,t}x^t; \quad r_i(x) = r_{i,1}x + \cdots + r_{i,t}x^t; \quad u_i(x) = u_{i,1}x + \cdots + u_{i,t}x^t \tag{35}$$

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 $\pi_i$, the NIZK proof of knowledge of $s_{i,0}$ with respect to $g^{s_{i,0}}$.
4. Party $i$ broadcasts $(\mathsf{cm}_i, \pi_i)$ 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 $i = 1, \ldots, n$:

$$g^{s_i(j)}h^{r_i(j)}v^{u_i(j)} = \prod_{k\in[0,t]} \mathsf{cm}_i[k]^{j^k} \tag{36}$$

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

$$\mathsf{pk}_j = \prod_{i\in\mathcal{Q}} \prod_{k\in[0,t]} \mathsf{cm}_i[k]^{j^k} \tag{37}$$

10. Each party $j$ sets its signing keys to be $\mathsf{sk}_j = (\sum_{i\in\mathcal{Q}} s_i(j), \sum_{i\in\mathcal{Q}} r_i(j), \sum_{i\in\mathcal{Q}} u_i(j))$.
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].

Party $i$ then computes the commitment $\mathsf{cm}_i \in \mathbb{G}^{t+1}$ to these polynomials as (step 2):

$$\mathsf{cm}_i = [g^{s_{i,0}}, g^{s_{i,1}}h^{r_{i,1}}v^{u_{i,1}}, \cdots, g^{s_{i,t}}h^{r_{i,t}}v^{u_{i,t}}] \tag{39}$$

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

Party $i$ then publishes (step 4), using a broadcast channel, $(\mathsf{cm}_i, \pi_i)$. Intuitively, the proof $\pi_i$ ensures (except with a negligible probability) that the constant terms of $r_i(x)$ and $u_i(x)$ are zero. 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 $(\mathsf{cm}_i, \pi_i)$ (via the broadcast channel), accepts them as valid shares if $\pi_i$ is a valid proof and the following holds:

$$g^{s'}h^{r'}v^{u'} = \prod_{k\in[0,t]} \mathsf{cm}_i[k]^{j^k} \tag{40}$$

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 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 $\mathsf{pk}_j$ of party $j$ is computed as in equation (37). The signing key $\mathsf{sk}_j$ of each party $j$ is the sum

**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 $\alpha_2 \leftarrow\$ \mathbb{F}$. Let $h = g^a$ and $v = g^{\alpha_2}$.
2. Let $\mathcal{C}$ and $\mathcal{H}$ be the set of honest and malicious parties, respectively. Sample $\hat{i} \leftarrow\$ \mathcal{H}$, and let $\mathcal{H}_{\_\hat{i}} = \mathcal{H} \setminus \{\hat{i}\}$.
3. On behalf of each party $i \in \mathcal{H}_{\_\hat{i}}$, run the DKG protocol as per the specification.
4. 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 $r_{\hat{i}}(0) \neq 0$. Compute the commitment $\mathsf{cm}_{\hat{i}}$ as:
$$\mathsf{cm}_{\hat{i}} = [g^{s_{\hat{i},0}} h^{r_{\hat{i},0}} v^{u_{\hat{i},0}}, g^{s_{\hat{i},1}} h^{r_{\hat{i},1}} v^{u_{\hat{i},1}}, \ldots, g^{s_{\hat{i},t}} h^{r_{\hat{i},t}} v^{u_{\hat{i},t}}] \tag{42}$$
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.
5. Compute $\pi_{\hat{i}}$, the proof of knowledge of exponent $\mathsf{cm}_{\hat{i}}[0]$ with respect to $g$, by running the NIZK simulator.
6. Run the rest of the DKG protocol, follow the honest protocol specification on behalf of every honest party.
7. Let $\mathcal{Q}$ be the qualified set of parties for the DKG. Note that, since we are working in an synchronous network, $\hat{i} \in \mathcal{Q}$. Let $r(x)$ be the polynomial where $r(x) = \sum_{i \in \mathcal{Q}} r_i(x)$. Abort, if $r(0) = 0$.

**Corruption simulation.**

6. If $\mathcal{A}$ corrupts a signer $i \in \mathcal{H}$.
   (a) If $i = \hat{i}$, rewind $\mathcal{A}$ to step 2 and rerun.
   (b) Otherwise, send the internal state of signer $i$ including $\mathsf{sk}_i$ to $\mathcal{A}$. 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.

of the $j$-th share of all dealers in $\mathcal{Q}$ as shown in step 11 of Figure 5. Let $s(x), r(x), u(x)$ be the polynomials defined as:
$$s(x) = \sum_{i \in Q} s_i(x); \quad r(x) = \sum_{i \in Q} r_i(x); \quad u(x) = \sum_{i \in Q} u_i(x); \tag{41}$$
Once the DKG protocol finishes, each party $i$ outputs its signing key $\mathsf{sk}_i = (s(i), r(i), u(i))$, the public key $\mathsf{pk} = g^{s(0)} = g^{s(0)} h^{r(0)} v^{u(0)}$, and the per signer public keys $\{\mathsf{pk}_i = g^{s(i)} h^{r(i)} v^{u(i)}\}_{i \in [n]}$.

*Using other DKG protocols.* In Figure 5, we illustrate how to augment the JF-DKG protocol for our threshold 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 to generate keys for our threshold signature scheme by each VSS dealer to: (i) share two additional zero-polynomials $r(\cdot)$ and $u(\cdot)$; and (ii) publish a NIZK proof $\pi$ 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 satisfy the *correctness* and *SIP simulatibility* property we define.

**Lemma 8 (Correctness).** *Assuming the hardness of discrete logarithm in* $\mathbf{G}$*, our DKG protocol in Figure 5, is correct 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 $\theta$. $\mathcal{A}_{\mathrm{DL}}$ picks the other parameter as $g^\alpha$ for some known $\alpha \leftarrow\$ \mathbb{F}$. Without loss of generality, let $h = g_1^\alpha$ and $v = g_2^\alpha$ for some $(\alpha_1, \alpha_2) \in \mathbb{F}^2$. $\mathcal{A}_{\mathrm{DL}}$ next faithfully runs the DKG and signing protocol 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 hardness of discrete logarithm that if $(r_i, u_i) \neq (0,0)$ with probability $\varepsilon$, then $\mathcal{A}_{\mathrm{DL}}$ will solve discrete logarithm with probability at least $\varepsilon/2$.

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 $\mathsf{cm}_i$ be the commitment vector output by the malicious party $i$, along with a NIZK proof-of-knowledge $\pi_i$. $\mathcal{A}_{\mathrm{DL}}$ also extracts using the NIZK proof-of-knowledge extractor the witness $w$ such that $g^w = \mathsf{cm}_i[0]$. It is also the case that $\mathsf{cm}_i[0] = g^{s_i} h^{r_i} v^{u_i}$. Then, we have that:

$$w = s_i + \alpha_1 r_i + \alpha_2 u_i \tag{43}$$

Now, depending upon whether $r_i \neq 0$ or $u_i \neq 0$, we can extract $\alpha_1$ or $\alpha_2$, respectively as:

$$r_i \neq 0 \implies \alpha_1 = (w - s_i - u_i\alpha_2) \cdot r_i^{-1}; \qquad \text{and} \qquad u_i \neq 0 \implies \alpha_2 = (w - s_i - r_i\alpha_1) \cdot u_i^{-1} \tag{44}$$

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 $\varepsilon$, then $\mathcal{A}_{\mathrm{DL}}$ outputs the discrete logarithm of $y$ with respect to $g$, with probability at least $\varepsilon/2$. Hence, assuming the hardness of discrete logarithm $r(0) = u(0) = 0$.

**Lemma 9 (SIP Simulatability).** *The DKG protocol in Figure 5 is SIP simulatable as per Definition 6, except with a negligible probability.*

*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 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 $\Sigma$-protocols are perfect HVZK, we get that $\mathcal{A}$'s view in game $\mathbf{G}_0$ is identical to its view in game $\mathbf{G}_1$.

GAME $\mathbf{G}_2$: This game is identical to $\mathbf{G}_1$, except that we use a polynomial $s_{\hat{i}}(x)$ such that

$$s_{\hat{i}}(0) = s_{\hat{i},0} + r\alpha_1 + u\alpha_2, \quad \text{for} \;\; s_0 \leftarrow\$ \; \mathbb{F} \tag{45}$$

Observe that for any fixed $(\alpha_1, \alpha_2)$, $r \leftarrow\$ \; \mathbb{F} \setminus \{0\}$ and $u \leftarrow\$ \; \mathbb{F} \setminus \{0\}$. Since $s_0$ is chosen uniformly at random, $s_0 + r\alpha_1 + u\alpha_2$ is also uniformly random. Hence, $\mathcal{A}$'s view in game $\mathbf{G}_2$ is identical to its view in game $\mathbf{G}_1$.

GAME $\mathbf{G}_3$: This game is identical to $\mathbf{G}_2$, except that we choose polynomials $s_{\hat{i}}(x), r_{\hat{i}}(x)$ and $u_{\hat{i}}(x)$ such that

$$s_{\hat{i}}(0) = s_{\hat{i},0}, \quad r_{\hat{i}}(0) = r, \quad u_{\hat{i}}(0) = u, \quad \text{for } s_{\hat{i},0} \leftarrow\$ \; \mathbb{F} \tag{46}$$

Based on a similar argument as Lemma 7, $\mathcal{A}$'s view in game $\mathbf{G}_3$ is identically distributed as its view in $\mathbf{G}_2$.

GAME $\mathbf{G}_4$: This game is identical to $\mathbf{G}_4$, except abort if $r(0) = 0$ where $r(x) = \sum_{i \in Q} r_i(x)$ for the qualified set $\mathcal{Q}$. We now argue that in game $\mathbf{G}_4$, we will abort with probability at most $\varepsilon_{\mathbb{F}} := 1/(|\mathbb{F}|-1)$. Note that with $u_{\hat{i}}(0)$ being uniformly random elements in $\mathbb{F}$, the commitment $\mathsf{cm}_{\hat{i}}[0]$, perfectly hides $s, r$ and $u$. This implies that $\mathcal{A}$'s choice of $r'$ and $u'$ is independent of $r$ and $u$. Hence, the probability that $r' + r$ being 0 is exactly:

$$\Pr[r + r' = 0] = \frac{1}{|\mathbb{F}|-1} \tag{47}$$

Combining all of these, we get that:

$$|\Pr[\mathbf{G}_4 = 1] - \Pr[\mathbf{G}_3 = 1]| \leq \frac{1}{|\mathbb{F}|-1} = \varepsilon_{\mathbb{F}} \tag{48}$$

Note that game $\mathbf{G}_4$ is exactly as the simulated protocol, and hence $\mathcal{A}$'s view in $\mathbf{G}_3$ is $\mathsf{View}^{\mathcal{A}}_{\text{sim}}$. This implies that $\mathsf{View}^{\mathcal{A}}_{\text{real}}$ and $\mathsf{View}^{\mathcal{A}}_{\text{sim}}$ are statistically indistinguishable, except with at most $\varepsilon_{\mathbb{F}}$ probability. □

**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 $\lambda$ 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. Then, the probability that $\mathcal{A}$ does not corrupt the SIP, i.e., $\hat{i} \notin \mathcal{C}$ is at least:

$$\Pr[\hat{i} \notin \mathcal{C}] \geq \frac{\binom{n-1}{t}}{\binom{n}{t}} = \frac{n-t}{n} \geq 1/2 \tag{49}$$

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

Equation (49) implies that the simulator fails to successfully simulate the DKG protocol with probability at most 1/2. Hence, $\mathcal{S}_{\mathsf{DKG}}$ will fail to simulate the DKG protocol after $\lambda$ iterations with probability at most $2^{-\lambda}$, which is negligible.

### 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 correctness of our threshold signature scheme with DKG follows from the correctness property of the DKG protocol. Hence, we will 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 mention 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.

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 $\mathcal{S}_{\mathsf{DKG}}$. Lemma 9 ensures that the simulated public is rigged, and at the same time $\mathcal{A}$'s view of the simulated protocol is statistically indistinguishable from its view of the real protocol. 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.** $\mathcal{A}_{\text{co-CDH}}$ 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, $\mathcal{H} \subseteq \mathcal{Q}$, it is also the case that $\hat{i} \in \mathcal{Q}$. Let $s(\cdot), r(\cdot), u(\cdot)$ be the degree $t$ polynomials defined as in step 2 of Figure 7, where:

$$s(x) = \sum_{i \in \mathcal{Q}} s_i(x); \quad r(x) = \sum_{i \in \mathcal{Q}} r_i(x); \quad u(x) = \sum_{i \in \mathcal{Q}} u_i(x) \tag{52}$$

Note that 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. Let $s = s(0)$ and let $r(0) = r + r'$ and $u(0) = u + u'$, where $r'$ and $u'$ are the constant terms of polynomials chosen by signers in $\mathcal{C} \cap \mathcal{Q}$. From Lemma 8, assuming hardness of discrete logarithm in $\mathbb{G}$, both $r' = u' = 0$, except with a negligible probability. Thus, we can safely ignore them from here on without affecting our simulation much. Nevertheless, as we illustrate later, even non-zero $r'$ and $u'$ do not

**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 $r = r_{\hat{i}}(0)$ and $u = u_{\hat{i}}(0)$.
2. Let $Q$ 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:
$$s(x) = \sum_{i \in Q} s_i(x); \quad r(x) = \sum_{i \in Q} r_i(x); \quad u(x) = \sum_{i \in Q} u_i(x); \tag{50}$$
Then the resulting secret key $\mathsf{sk}_i$ of party $i$ is $\mathsf{sk}_i = (s(i), r(i), u(i))$. The public key $\mathsf{pk} = g^{s(0)} h^{r(0)}$, and the threshold public key $\mathsf{pk}_i$ for each party $i$ is $\mathsf{pk}_i = g^{s(i)} h^{r(i)} v^{u(i)}$.
3. Let $s = s(0)$, $r + r' = r(0)$ and $u + u' = u(0)$. Here $r', u$ are defined as:
$$r' = \sum_{i \in Q \cap \mathcal{C}} r_i(0); \quad u' = \sum_{i \in Q \cap \mathcal{C}} u_i(0); \tag{51}$$

**Corruption simulation:**

6. If $\mathcal{A}$ corrupts a signer $i \in \mathcal{H}$.
   (a) If $i = \hat{i}$, rewind $\mathcal{A}$ to step 2 and rerun.
   (b) Otherwise, send the internal state of signer $i$ including $\mathsf{sk}_i$ to $\mathcal{A}$. 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. Note that $\mathcal{A}_{\text{co-CDH}}$ does not use $r'$ and $u'$ while simulating threshold signatures, and programs the random oracles only based on $r$ and $u$.

**Compute co-CDH output:**

13. Let $\sigma$ be the valid forgery on $m_{\hat{k}}$, then output $\sigma^{(r+r')^{-1}} \cdot \hat{g}^{-b(r+r')^{-1}(s+\alpha_2(u+u'))}$ as the co-CDH output. Note that in Figure 6, $\mathcal{S}_{\mathsf{DKG}}$ ensures that $(r + r') \neq 0$, and hence $(r + r')^{-1}$ is well defined.

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

affect our unforgeability guarantees. Stated differently, this also implies that the *soundness* of the proof-of-knowledge scheme we use in the DKG protocol is needed only for our threshold signature's correctness and not for its unforgeability.

**Simulating threshold signature.** $\mathcal{A}_{\text{co-CDH}}$ simulates the threshold signing phase exactly as in §6, except how it responds to additional corruption queries. More precisely, if $\mathcal{A}$ corrupts any party $i \in \mathcal{H}_{-\hat{i}} \setminus \{\hat{i}\}$ anytime during the signing phase, $\mathcal{A}_{\text{co-CDH}}$ reveals the internal state of $i$ to $\mathcal{A}$. $\mathcal{A}_{\text{co-CDH}}$ 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.

We want to note that while simulating the threshold signatures, $\mathcal{A}_{\text{co-CDH}}$ will program only based on $r$ and $u$ and will not use $r'$ and $u'$. Nevertheless, when either $r' \neq 0$ and $u' \neq 0$, the final threshold signature will not be accepted (except with a negligible probability) by the $\mathsf{Ver}$ algorithm in both the simulated protocol and the real execution of the signature scheme. But this is an acceptable behavior while proving unforgeability, as we only care about the indistinguishability of $\mathcal{A}$'s view in the real execution and the simulated protocol.

**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 $g^s h^{r+r'} v^{u+u'}$ where $\mathcal{A}_{\text{co-CDH}}$ knows $(s, r + r', u + u')$. Moreover, in the simulated protocol we have that $r + r' \neq 0$. $\mathcal{A}_{\text{co-CDH}}$ computes the co-CDH output $\hat{g}_{\text{cdh}}$ as

$$\hat{g}_{\text{cdh}} = \sigma^{(r+r')^{-1}} \cdot \hat{g}^{-b(r+r')^{-1}(s+\alpha_2(u+u'))} \tag{53}$$

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

We now prove that assuming the hardness of DDH in $\hat{\mathbb{G}}$, if $\mathcal{A}$ forges a signature in the real protocol with probability $\varepsilon_\sigma$, then $\mathcal{A}$ also forges a signature in the simulated protocol with probability at least

$\varepsilon_\sigma - \varepsilon_{\text{DDH}} - 2\varepsilon_{\text{nizk-fail}} - \varepsilon_{\mathbb{F}}$. Here $\varepsilon_{\text{DDH}}$ is the advantage of any adversary in breaking the DDH in $\hat{\mathbb{G}}$, $\varepsilon_{\text{nizk-fail}} = (q_{\mathsf{H}_{\mathbb{F}}} \cdot q_s \cdot n)/|\mathbb{F}|^2$, and $\varepsilon_{\mathbb{F}} = 1/(|\mathbb{F}|-1)$.

**Lemma 10.** *Given pairing groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$, if a PPT adversary $\mathcal{A}$ forges a signature in the real protocol with probability $\varepsilon_\sigma$, then $\mathcal{A}$ also forges a signature in the simulated protocol with probability at least at least $\varepsilon_\sigma - \varepsilon_{\text{DDH}} - 2\varepsilon_{\text{nizk-fail}} - \varepsilon_{\mathbb{F}}$. Here $\varepsilon_{\text{DDH}}$ is the advantage of any adversary in breaking the DDH in $\hat{\mathbb{G}}$, $\varepsilon_{\text{nizk-fail}} = (q_{\mathsf{H}_{\mathbb{F}}} \cdot q_s \cdot n)/|\mathbb{F}|^2$, and $\varepsilon_{\mathbb{F}} = 1/(|\mathbb{F}|-1)$.*

We will prove this via a sequence of games. Game $\mathbf{G}_0$ being the real execution is the real protocol execution, and game $\mathbf{G}_8$ being the interaction of $\mathcal{A}$ with $\mathcal{A}_{\text{co-CDH}}$.

GAME $\mathbf{G}_0$ **to** GAME $\mathbf{G}_4$: Similar to game $\mathbf{G}_0$ to $\mathbf{G}_4$ in §6.4, except $\mathcal{A}_{\text{co-CDH}}$ runs the $\mathsf{DKG}$ protocol in Figure 5 with $\mathcal{A}$ instead of the $\mathsf{KGen}$ functionality. Hence, by a similar argument as in §6.4, $|\Pr[\mathbf{G}_0 = 1] - \Pr[\mathbf{G}_4 = 1]| \leq \varepsilon_{\text{DDH}} + \varepsilon_{\text{nizk-fail}}$.

GAME $\mathbf{G}_5$: This game is identical to $\mathbf{G}_5$ except that we run the DKG simulator $\mathcal{S}_{\mathsf{DKG}}$ to set up the signing keys. Lemma 9 ensures that $\mathcal{A}$'s interaction with $\mathcal{S}_{\mathsf{DKG}}$ is statistically indistinguishable from its view in the real protocol execution, except with a negligible probability of $\varepsilon_{\mathbb{F}}$. Moreover, an argument similar to Lemma 7 implies that $\mathcal{A}$'s view of the rest of the signing phase is identically distributed in the game $\mathbf{G}_4$ and $\mathbf{G}_5$. Thus, combining these, we get:

$$|\Pr[\mathbf{G}_4 = 1] - \Pr[\mathbf{G}_5 = 1]| \leq \varepsilon_{\mathbb{F}} \tag{54}$$

GAME $\mathbf{G}_6$: This game is identical to $\mathbf{G}_6$, except that we use $h = g^a$, i.e., $\alpha = a$, from the co-CDH input, and use $\hat{g}^a$ to compute the random oracle outputs in step 10.(b) in Figure 4. Clearly, $\mathcal{A}$'s view in game $\mathbf{G}_6$ is identical to its view in game $\mathbf{G}_7$, and hence $\Pr[\mathbf{G}_5 = 1] = \Pr[\mathbf{G}_6 = 1]$.

GAME $\mathbf{G}_7$: This game is identical to $\mathbf{G}_6$, except that we use actual NIZK proofs for partial signatures. Thus, using an argument similar as in the advantage of $\mathcal{A}$ between $\mathbf{G}_3$ and $\mathbf{G}_4$ in §6.4, we get that:

$$|\Pr[\mathbf{G}_6 = 1] - \Pr[\mathbf{G}_7 = 1]| \leq \varepsilon_{\text{nizk-fail}} \tag{55}$$

Observe that game $\mathbf{G}_7$ is exactly the $\mathcal{A}$'s interaction with $\mathcal{A}_{\text{co-CDH}}$. Hence, from the above sequence of games, we get that:

$$|\Pr[\mathbf{G}_0 = 1] - \Pr[\mathbf{G}_7 = 1]| \leq \varepsilon_{\text{DDH}} - 2\varepsilon_{\text{nizk-fail}} - \varepsilon_{\mathbb{F}} \implies \Pr[\mathbf{G}_7 = 1] \geq \varepsilon_\sigma - \varepsilon_{\text{DDH}} - 2\varepsilon_{\text{nizk-fail}} - \varepsilon_{\mathbb{F}} \tag{56}$$

This implies that any adversary $\mathcal{A}$ outputs a forgery in the real execution of our signature scheme (i.e., $\mathbf{G}_0$) with probability $\varepsilon_\sigma$, then $\mathcal{A}$ outputs a forgery during its interaction with $\mathcal{A}_{\text{co-CDH}}$ (i.e., $\mathbf{G}_7$) with probability at least $\varepsilon_\sigma - \varepsilon_{\text{DDH}} - 2\varepsilon_{\text{nizk-fail}} - \varepsilon_{\mathbb{F}}$. We use this to prove the following main theorem.

**Theorem 6 (Adaptively secure BLS threshold signature with DKG).** *Let $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ be a pairing group of prime order $q$. Let $\lambda$ be the security parameter. For any $n, t$ for $n = \mathsf{poly}(\lambda)$ and $t < n/2$, let $\mathsf{DKG}$ be a DKG protocol that satisfies the properties we describe in §7.1. Then, assuming hardness of decisional diffie-hellman (DDH) in $\hat{\mathbb{G}}$, and co-computational diffie-hellman (co-CDH) in $\mathbb{G}, \hat{\mathbb{G}}$ in the random oracle model, the threshold signature scheme in §5 where signers use $\mathsf{DKG}$ to generate the signing keys is $(\varepsilon_\sigma, T, q_{\mathsf{H}}, q_{\mathsf{H}_{\mathbb{F}}}, q_s)$ unforgeable under chosen message attack as per Definition 4 against an adaptive adversary $\mathcal{A}$ corrupting up to $t$ signers, where*

$$\varepsilon_\sigma \leq \varepsilon_{\text{DDH}} + 2\varepsilon_{\text{nizk-fail}} + \varepsilon_{\mathbb{F}} + q_{\mathsf{H}} \cdot \varepsilon_{\text{CDH}},$$

*$\varepsilon_{\text{nizk-fail}} = (q_{\mathsf{H}_{\mathbb{F}}} \cdot q_s \cdot n)/|\mathbb{F}|^2, \varepsilon_{\mathbb{F}} = 1/(|\mathbb{F}|-1)$, and $\varepsilon_{\text{DDH}}$ and $\varepsilon_{\text{CDH}}$ are the advantages of an adversary running in $T \cdot \mathsf{poly}(\lambda, n)$ time in breaking the DDH in $\hat{\mathbb{G}}$ and co-CDH assumption in $\mathbb{G}, \hat{\mathbb{G}}$, respectively.*

*Proof.* Same as the proof of Theorem 5, except we now also have the additional additive security loss of $\varepsilon_{\mathbb{F}}$.

### 8.1 Unforgeability with static adversary

We now briefly argue that our signature scheme is statically secure, assuming 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 $i$. $\mathcal{A}_{\text{co-CDH}}$ 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, §4] 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, as 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, as 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.

**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

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) | Parital 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 in Boldyreva-I is a single $\hat{\mathbb{G}}$ element, which is 96 bytes. In Boldyreva-II, the partial signature also consists of a $\Sigma$-protocol proof, that, using the standard optimization of including the fiat-shamir challenge [CS97] is $(c, z) \in \mathbb{F}^2$. Hence, the partial signatures in Boldyreva-II are 64 bytes longer compared to Boldyreva-I. Finally, our partial signature includes a $\Sigma$-protocol proof $(c, z_s, z_r, z_u) \in \mathbb{F}^4$, and hence in total are 224 bytes long. 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.

## 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 Helmann (DDH) and co-computational Diffie Hellmann 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

The authors would like to thank Dan Boneh for pointing us to the DDH rerandomization exercise in their book. We thank Amit Agarwal, Renas Bacho, Julian Loss, Victor Shoup, and 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+23. Fabrice Benhamouda, Shai Halevi, Hugo Krawczyk, Yiping Ma, and Tal Rabin. Sprint: High-throughput robust distributed schnorr signatures. *Cryptology ePrint Archive*, 2023.

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.

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.

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. *Cryptology ePrint Archive*, 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 $\sigma$-protocols. *Lecture Notes, University of Aarhus, Department for Computer Science*, page 84, 2002.

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

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.

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.

GG20. Rosario Gennaro and Steven Goldfeder. One round threshold ecdsa with identifiable abort. *Cryptology ePrint Archive*, 2020.

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.

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.

GS23. Jens Groth and Victor Shoup. Fast batched asynchronous distributed key generation. *Cryptology ePrint Archive*, 2023.

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.

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.