# 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 group 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 signature aggregation key is $n$ group elements, 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.** The rest of the paper is organized as follows. 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 §5 and analyze its security in §6. We discuss the implementation and evaluation details in §7, and conclude with a discussion in §8.

## 2 Related works

Threshold signature schemes are 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, RRJ+22, 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.

**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.

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

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-Peikart [LP01] and Abe et al., [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, 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 group with scalar field $\mathbb{F}$. Let $g \in \mathbb{G}$ be a uniformly random generator of $\mathbb{G}$ and $\mathsf{H} : \{0,1\}^* \to \hat{\mathbb{G}}$ 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 group with scalar field $\mathbb{F}$. Let $g, h \in \mathbb{G}$ be two uniformly random generators of $\mathbb{G}$. 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 $\mathsf{pk}_i = g^{s(i)}h^{s(i)} \in \mathbb{G}$.

With this initial setup, signers sign any message $m$ as follows. Let $\mathsf{H}_0, \mathsf{H}_1$ be two random oracles where $\mathsf{H}_b : \{0,1\}^* \to \hat{\mathbb{G}}$ 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 $\mathcal{A}_{\text{co-CDH}}$ can run a DKG protocol 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 node 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.** We use $\lambda$ to denote the security parameter. For any integer $a$, we use $[a]$ to denote the ordered set $\{1, 2, \ldots, a\}$. For any set $S$, we use $s \xleftarrow{\$} S$ to indicate that $s$ is sampled uniformly randomly from $S$. 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 $\lambda$. Also, 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 $t < n/2$ out of the $n$ signers. Corrupted signers can deviate arbitrarily from the protocol specification. Our signature scheme will run a Distributed Key Generation (DKG) protocol to set up the signing keys. The DKG step is why we require $t < n/2$. For simplicity, we assume lock-step synchrony for our DKG protocol, i.e., replicas execute the protocol in synchronized rounds. A message sent at the start of a round arrives by the end of that round. We also 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].

### 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 $d+1$ 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 the 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)** *A cyclic group $\mathbb{G}$ of prime order $q$, satisfies the the Decisional Diffie-Hellman (DDH) assumption if, for all PPT adversary $\mathcal{A}$ and $x, y, r \xleftarrow{\$} \mathbb{F}$, it holds that:*

$$\Pr[\mathcal{A}(g, g^x, g^y, g_b) \to b' : g_0 = g^{xy}, g_1 = g^r, b \xleftarrow{\$} \{0,1\}] = \mathsf{negl}(\kappa)$$

**Assumption 2 (co-CDH)** *A pairing group $(\mathbb{G}, \hat{\mathbb{G}})$ with generator $(g, \hat{g})$ and a bilinear pairing $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ satisfies the co-CDH assumption if, for all PPT adversary $\mathcal{A}$, it holds that*

$$\Pr[\mathcal{A}(g, \hat{g}, g^a, \hat{g}^b, \hat{g}^b) = \hat{g}^{ab} : a, b \xleftarrow{\$} \mathbb{F}] = \mathsf{negl}(\kappa)$$

### 4.3 Threshold Signature

In this section, we introduce the syntax and security definitions for threshold signature schemes. We focus on non-interactive schemes. Our security definitions are based on those of [BL22].

**Definition 2 (Non-Interactive Threshold Signature).** *A non-interactive $(n, t)$-threshold signature scheme is a tuple of polynomial time computable algorithms $\Sigma = (\mathsf{DKG}, \mathsf{PSign}, \mathsf{PVer}, \mathsf{Ver}, \mathsf{Comb})$ with the following properties:*

1. $\mathsf{DKG}$. *This is an interactive protocol among n signers, which all take as inputs common random string (CRS), a security parameter $\lambda \in \mathbb{N}$ as well as a pair of integers $n, t \in \mathsf{poly}(\lambda)$ with $2t < n$. 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 the secret key $\mathsf{sk}_i$.*
2. $\mathsf{PSign}$. *The partial signing algorithm is a possibly randomized algorithm that takes as input a message $m$ and a secret key share $\mathsf{sk}_i$. It outputs a signature share $\sigma_i$.*
3. $\mathsf{PVer}$. *The signature share verification algorithm is a deterministic algorithm that takes as input a message $m$, a public key share $\mathsf{pk}_i$, and a signature share $\sigma_i$. It outputs 1 (accept) or 0 (reject).*
4. $\mathsf{Comb}$. *The signature share combining algorithm takes as input the 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. $\mathsf{Ver}$. *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 next define the security of a non-interactive threshold signature scheme with adaptive corruption. Let $\mathsf{H}_b : \{0,1\}^* \to \hat{\mathbb{G}}$ for each $b \in \{0,1\}$ are two distinct cryptographic hash functions (modeled as random oracles). We use $q_h$ to denote the maximum number of allowed queries to $\mathsf{H}_0$.

**Definition 3 (Unforgeability Under Chosen Message Attack).** *Let $\Sigma = (\mathsf{DKG}, \mathsf{PSign}, \mathsf{PVer}, \mathsf{Comb}, \mathsf{Ver})$ be a non-interactive $(n,t)$-threshold signature scheme. For an algorithm $\mathcal{A}$, define experiment $\mathbf{UF\text{-}CMA}_{\Sigma}^{\mathcal{A},t}$ as follows:*

- **Setup.** *Initialize the sets $\mathcal{H} := \{1, \ldots, n\}, \mathcal{C} = \emptyset$. Run $\mathcal{A}$ on public parameters.*
- **Corruption Queries**. *At any point during the experiment, $\mathcal{A}$ may corrupt a signer $i$. In this case, return the internal state of signer $i$ and set $\mathcal{H} = \mathcal{H} \setminus \{i\}$, $\mathcal{C} = \mathcal{C} \cup \{i\}$.*
- **Distributed Key Generation**. *Run $\mathsf{DKG}$ among parties $\{1, \ldots, n\}$. At the end of which each signer $i$ outputs its signing key $\mathsf{sk}_i$. Each signer additionally outputs $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$, i.e., threshold public keys of all parties. Also, $\mathcal{A}$ learns $\mathsf{sk}_i$ for each $i \in \mathcal{C}$.*
- **Online Phase.** *During this phase, $\mathcal{A}$ can make the following queries.*
  - **Signing Queries.** *$\mathcal{A}$ submits the tuple $(i, m)$ for some $i \in \mathcal{H}$, and receives $\sigma_i \leftarrow \mathsf{PSign}(\mathsf{sk}_i, m)$.*
  - **Random Oracle Queries.** *When $\mathcal{A}$ submits a query $m$ to random oracle $\mathsf{H}_b$ for either $b \in \{0,1\}$, check if $\mathsf{H}_b(m) = \perp$, if so, set $\mathsf{H}_b(m) \xleftarrow{\$} \hat{\mathbb{G}}$ and record. Return $\mathsf{H}_b(m)$.*
- **Output Determination.** *When $\mathcal{A}$ outputs a message $m^*$ and a signature $\sigma^*$, let $\mathcal{S}$ denote the subset of parties for which $\mathcal{A}$ made a signing query of the form $(i, m^*)$. Output 1 if $|\mathcal{C} \cup \mathcal{S}| \leq t$ and $\mathsf{Ver}(\mathsf{pk}, m^*, \sigma^*) = 1$. Otherwise, output 0.*

*We say that $\Sigma$ is a $(\varepsilon, T, q_h, q_s)$-unforgeable under chosen message attacks (UF-CMA) if for all adversaries $\mathcal{A}$ running in time at most $T$, making at most $q_h$ random oracle queries, and making at most $q_s$ signing queries, $\Pr[\mathbf{UF\text{-}CMA}_{\Sigma}^{\mathcal{A},t} = 1] \leq \varepsilon$. Conversely, we say that $\mathcal{A}$ $(\varepsilon, T, q_h, q_s)$-breaks unforgeability of $\Sigma$ under chosen message attacks if it runs in time at most $T$, makes at most $q_h$ queries to the random oracle $\mathsf{H}_0$, $q_s$ signing queries, and $\Pr[\mathbf{UF\text{-}CMA}_{\Sigma}^{\mathcal{A},t} = 1] > \varepsilon$.*

### 4.4 Boldyreva's BLS signature scheme

In this section, we describe Boldyreva's BLS-based threshold signature scheme. Let $\mathsf{DKG}$ be a distributed key generation protocol. Boldyreva's BLS threshold signature scheme for parameters $(n,t)$ works as follows where we use $\mathsf{H} = \mathsf{H}_0$.

- $\mathsf{PSign}$: On input the message $m \in \{0,1\}^*$, 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 public key shares $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, 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 $\sigma = \prod_{i \in \mathcal{S}_0} \sigma_i^{L_i}$, where $L_i = \prod_{j \in \mathcal{S}_0} \left(\frac{j}{j-i}\right)$ 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 Design

In this section, we will describe our adaptively secure threshold signature scheme along with an efficient distributed key generation (DKG) protocol to set up the signing keys. Recall $\mathsf{H}_0, \mathsf{H}_1$ are two distinct random oracles where $\mathsf{H}_b : \{0,1\}^* \to \hat{\mathbb{G}}$ for $b \in \{0,1\}$.

## 5.1 Distributed Key Generation

Existing DKG protocols, both in synchronous and asynchronous networks [Ped91, CGJ+99, CS04, FS01, GJKR07, Gro21, KHG12, KKMS20, DYX+22] has the following three phase consisting of:

– *Sharing phase.* Each party samples a random secret and secret shares it using a verifiable secret sharing (VSS). In almost all DKG protocols, parties broadcast commitments to their secret polynomials.
– *Agreement phase.* Parties then agree on a subset of $t+1$ or more set of qualified parties who correctly dealt their secrets using VSS.
– *Key Derivation phase.* Parties then deterministically compute the DKG secret key based on the VSS secrets of parties in the chosen subset. A common approach is to sum up the VSS secrets of the qualified parties.

To set up the signing keys of our signature scheme, we can use any existing DKG scheme that follows the abovementioned three-phase structure with only a minor modification to the sharing phase. We will illustrate this using Pedersen's DKG protocol [Ped91] as a concrete example. Also, we will only focus on the sharing phase as the agreement and key derivation phase remain unchanged.

**Sharing phase of our DKG protocol.** Let $g, h, v \in \mathbb{G}$ be three uniform random generators of $\mathbb{G}$. Each party $i$ samples three uniformly random degree-$t$ polynomials $s_i(x), r_i(x), u_i(x)$ with $r_i(0) = u_i(0) = 0$ such that

$$\begin{aligned} s_i(x) &= s_{i,0} + s_{i,1}x + \cdots + s_{i,t}x^t; \\ r_i(x) &= r_{i,1}x + \cdots + r_{i,t}x^t; \\ u_i(x) &= u_{i,1}x + \cdots + u_{i,t}x^t \end{aligned}$$

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

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

Party $i$ then publishes, using a broadcast channel, $\mathsf{cm}_i$ as the commitment to its polynomials. Along with the commitment, party $i$ additionally publishes a proof $\pi_i$ of knowledge of discrete logarithm of $\mathsf{cm}_i[0] = g^{s_{i,0}}$ with respect to the generator $g$ using the standard "$\Sigma$"-protocol [Sch90]. Intuitively, the proof $\pi_i$ ensures that the constant terms of $r_i(x)$ and $u_i(x)$ are zero. Also, party $i$ sends each party $j$, via private channel, the tuple $(s_i(j), r_i(j), u_i(j))$. Upon receiving the tuple $(s', r', u')$, party $j$ accepts them as valid shares if $\pi_i$ is a valid proof and the following holds.

$$g^{s'} h^{r'} v^{u'} = \prod_{k \in [0,t]} \mathsf{cm}_i[k]^{i^k} \tag{2}$$

**Input:** $(g, h, v, \mathsf{pk}) \in \mathbb{G}^4$; $m \in \{0,1\}^*$; $\mathsf{H}_b : \{0,1\}^* \to \hat{\mathbb{G}}$ for $b = 0, 1$; $\sigma \in \hat{\mathbb{G}}$
**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 $\sigma = \mathsf{H}_0(m)^s \mathsf{H}_1(m)^r$. The protocol works as follows.

1. $\mathcal{P}$ samples $\hat{s}, \hat{r}, \hat{u} \xleftarrow{\$} \mathbb{F}$ and computes $x = g^{\hat{s}} h^{\hat{r}} v^{\hat{u}}$, and $y = \mathsf{H}_0(m)^{\hat{s}} \mathsf{H}_1(m)^{\hat{r}}$. $\mathcal{P}$ sends the tuple $(x, y)$ to $\mathcal{V}$.
2. $\mathcal{V}$ responds with a random challenge $c \in \mathbb{F}$.
3. $\mathcal{P}$ responds with tuple $(z_s, z_r, z_u)$ where:

$$z_s = \hat{s} + s \cdot c; \quad z_r = \hat{r} + r \cdot c; \quad z_u = \hat{u} + u \cdot c;$$

$\mathcal{V}$ accepts the proof $(z_s, z_r, z_u)$ if the following checks are successful.

$$g^{z_s} h^{z_r} v^{z_u} = x \mathsf{pk}^c; \quad \mathsf{H}_0(m)^{z_s} \mathsf{H}_1(m)^{z_r} = y \sigma^c$$

**Fig. 1.** $\Sigma$-protocol to compute NIZK proof of correct partial signatures.

Upon receiving valid shares, parties continue participating in the rest of the DKG protocol to agree on a qualified set of parties. Finally, parties use the qualified parties' VSS shares to derive their signing keys per the key derivation protocol. More precisely, let $Q$ be the set of qualified parties with $|Q| \geq t + 1$ output by the agreement phase of the DKG protocol. Let $s(x), r(x), u(x)$ be the polynomials defined as:

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

Once the DKG protocol finishes, each party $i$ outputs its signing key $\mathsf{sk}_i = (s(i), r(i), u(i))$, the public key $g^{s(0)} = g^{s(0)} h^{r(0)} v^{u(0)}$, and the per signer public keys $\{\mathsf{pk}_i = g^{s(i)} h^{r(i)} v^{u(i)}\}_{i \in [n]}$.

### 5.2 Signature Scheme

DKG: For any $(n, t)$ threshold signature, parties run the DKG protocol as per §5.1. Let $\mathsf{sk}_i = (s(i), r(i), u(i))$ be the signing key of party $i$ for polynomials as defined in equation (3). Note that these polynomials are chosen such that $r(0) = u(0) = 0$. Let $\mathsf{pk} = g^{s(0)}$ be the public verification key, $\mathsf{pk}_i = g^{s(i)} h^{r(i)} v^{u(i)}$ for all $i \in [n]$ be the per party public verification key.

PSign: The partial signature of party $i$ 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 correctness of $\sigma_i$ with respect to $\mathsf{pk}_i$. Node $i$ computes $\pi_i$ using the $\Sigma$-protocol in Figure 1. To make the signing phase non-interactive, signers use the Fiat-Shamir heuristic to non-interactively generate the proof, i.e., singer $i$ computes its $\Sigma$-protocol challenge $c_i = \mathsf{H}(x, y, \mathsf{pk}_i, \sigma_i, m)$.

PVer: On input the partial 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 only if $\mathcal{V}$ accepts.

Comb: Upon receiving a set of the partial signatures $\{\sigma_i, \pi_i\}$, an aggregator validates them individually using 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. More precisely, on input the public key $\mathsf{pk}$, 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 essentially not used in partial signatures; it appears only in the public verification key as an artifact to make the proof go through.

## 6 Security Analysis

We first analyze the properties of the Σ-protocol in Figure 1, which we then use to prove the correctness and adaptive security of our threshold signature scheme.

### 6.1 Analysis of Σ-protocol

The correctness of Σ-protocol is straightforward. Also, the knowledge soundness and honest-verifier zero-knowledge property follow using the 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:

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

**Honest verifier zero-knowledge.** Let $\mathcal{S}$ be the simulator. $\mathcal{S}$ then 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 as $c := \mathsf{H}(x, y, \mathsf{pk}, \sigma, m)$ 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

To prove correctness of our scheme, we will first argue that assuming hardness of discrete logarithm, partial signatures $\{\sigma_i\}$ are correctly formed.

**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 Σ-protocol proof $\pi_i$, let $\mathcal{E}$ be the extractor we describe in §6.1. Also, let $s', r', u'$ be the extracted witness. Then, we prove that $(s', r', u') = (s(i), r(i), u(i))$.

For the sake of contradiction, assume this is not the case. Let $\mathcal{A}_{\mathrm{DL}}$ be a discrete logarithm adversary, which on input $(g, y) \in \mathbb{G}^2$ uses $\mathcal{A}$ to compute the discrete logarithm of $y$ with respect to $g$. $\mathcal{A}_{\mathrm{DL}}$ samples $\theta \in \{0, 1\}$, and sets either $h = y$ or $v = y$, depending upon 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 runs the DKG and threshold signature protocol with $\mathcal{A}$ on behalf of honest parties. Since $t < n/2$, at the end of the DKG protocol, $\mathcal{A}_{\mathrm{DL}}$ knows the polynomials $s(\cdot), r(\cdot), v(\cdot)$ in its entirety.

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 $\mathbb{G}$. Then, by using eq. (5) and by simple case analysis, it is easy to show that $\mathcal{A}_{\mathrm{DL}}$ will output the discrete logarithm of $y$ with respect to $g$ with probability $\varepsilon/2$. Here $\varepsilon$ is the probability that $\mathcal{A}$ outputs a malformed partial signature for signer $i$. Hence we get a contradiction. Therefore, $(s', r', u') = (s(i), r(i), u(i))$, which implies that $\sigma_i$ is well formed. □

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 Unforgeability with 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. 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 $\hat{g}^{ab}$. 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 the $\mathcal{A}$.

**Simulating the DKG.** $\mathcal{A}_{\text{co-CDH}}$ samples $\alpha_2 \xleftarrow{\$} \mathbb{F}$ and sets $h = g^a$ and $v = g^{\alpha_2}$. Let $\mathcal{H}$ and $\mathcal{C}$ be the set of honest and malicious parties, respectively. $\mathcal{A}_{\text{co-CDH}}$ samples an honest party uniformly at random. Let $\hat{i}$ be the chosen party and $\mathcal{H}_{-\hat{i}} = \mathcal{H} \setminus \{\hat{i}\}$ be the rest of the honest parties. $\mathcal{A}_{\text{co-CDH}}$ then does the following.

1. For each honest party in $\mathcal{H}_{-\hat{i}}$, $\mathcal{A}_{\text{co-CDH}}$ follows the honest protocol.
2. For honest party $\hat{i}$, $\mathcal{A}_{\text{co-CDH}}$ follows the honest protocol except that it samples $r, u \xleftarrow{\$} \mathbb{F} \setminus \{0\}$. In other words, it chooses the polynomial $r_{\hat{i}}(x)$ and $u_{\hat{i}}(x)$ such that $r_{\hat{i}}(0) = r \neq 0$ and $u_{\hat{i}}(x) = u \neq 0$.
3. $\mathcal{A}_{\text{co-CDH}}$ then uses the NIZK simulator to compute the Proof of knowledge of $\mathsf{cm}_{\hat{i}}[0]$ with respect to $g$.
4. If $\mathcal{A}$ corrupts any party $i \in \mathcal{H}_{-\hat{i}}$ anytime during the protocol (including 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.

Let $Q$ be the qualified set of parties chosen by $\mathcal{A}$ to derive the threshold signing key. If $\hat{i} \notin Q$, then $\mathcal{A}_{\text{co-CDH}}$ rewinds $\mathcal{A}$ to the start of the simulation and reruns the DKG protocol with new randomness. Otherwise, $\mathcal{A}_{\text{co-CDH}}$ continues to the signing phase.

**Simulating threshold signature.** As discussed above, anytime during the signing phase, if $\mathcal{A}$ corrupts node $i \in \mathcal{H}_{-\hat{i}}$, then $\mathcal{A}_{\text{co-CDH}}$ faithfully reveals the internal state of party $i$, 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 $\alpha = a + \alpha_2 \cdot (r/v)$. Note that $\mathsf{H}_0$ is always queried on the forged message, at least by $\mathcal{A}_{\text{co-CDH}}$ during the signature verification. Let $q_h$ be an upper bound on the number of random oracle queries to $\mathsf{H}_0$, including the query during the signature verification. $\mathcal{A}_{\text{co-CDH}}$ samples $\hat{k} \xleftarrow{\$} [q_h]$. 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 oracle as follows.

$$\begin{aligned}k \neq \hat{k} &\implies \mathsf{H}_0(m_k) = \hat{g}^{\beta\gamma_k+\delta_k};\ \mathsf{H}_1(m_k) = \hat{g}^{\alpha\cdot(\beta\gamma_k+\delta_k)} \text{ for } \gamma_k, \delta_k \xleftarrow{\$} \mathbb{F} \\ k = \hat{k} &\implies \mathsf{H}_0(m_k) = \hat{g}^b; \quad \mathsf{H}_1(m_k) = \hat{g}' \text{ for } \hat{g}' \xleftarrow{\$} \hat{\mathbb{G}}\end{aligned}$$

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

**Breaking the co-CDH assumption.** Let $(m_{\hat{k}}, \sigma)$ be a non-trivial 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

$$\hat{g}_{\text{cdh}} = \sigma^{r^{-1}} \cdot \hat{g}^{-br^{-1}(s+\alpha_2 u)} \tag{6}$$

**Lemma 2.** *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.

$$e(\mathsf{pk}, \mathsf{H}_0(m_{\hat{k}})) = e(g, \sigma) \tag{7}$$

This implies that $\mathsf{H}_0(m_{\hat{k}}) = \hat{g}^b$. Let $\hat{h} = \hat{g}^a$ and $\hat{v} = \hat{g}^{\alpha_2}$. Then, combining everything we get that:

$$\sigma = \left(\hat{g}^s \hat{h}^r \hat{v}^u\right)^b \implies \sigma^{r^{-1}} \cdot \hat{g}^{-br^{-1}(s+\alpha_2 u)} = \hat{h}^b = \hat{g}^{ab} \qquad \square$$

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

**Lemma 3.** *Given a pairing group $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$, assuming the hardness of decisional Diffie-Hellman (DDH) in $\hat{\mathbb{G}}$, $\mathcal{A}$'s view in its interaction with $\mathcal{A}_{\text{co-CDH}}$ is indistinguishable from its view in an instance of the real protocol.*

We will prove this via a sequence of hybrids. Hybrid 0 is the real protocol execution, and Hybrid 9 is the interaction of $\mathcal{A}$ with $\mathcal{A}_{\text{co-CDH}}$.

**Hybrid 0**. Same as the real execution of the protocol.

**Hybrid 1.** Same as Hybrid 0, except that we sample $\alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{F}$ and set $h = g^{\alpha_1}$ and $v = g^{\alpha_2}$. Trivially, hybrid 0 is identically distributed as hybrid 1.

**Hybrid 2.** Same as Hybrid 1, except that we program the signing random oracle as follows. Sample uniformly random $\beta, \phi \in \mathbb{F}$ at the start of the protocol. Let $\alpha = \alpha_1 + \alpha_2 \phi$. For the $k$-th random oracle query on message $m_k$, sample uniformly random $\gamma_k, \delta_k \in \mathbb{F}$. Set $\hat{g}^{\beta\gamma_k + \delta_k} := \mathsf{H}_0(m_k)$, and $\hat{g}^{\alpha(\beta\gamma_k+\delta_k)} := \mathsf{H}_1(m_k)$.

The indistinguishability of hybrid 2 from hybrid 1 is a crucial step in our proof. Before we prove this indistinguishability, we first prove the following helper lemmas. We want to note that Lemma 4 appears as an exercise in the book [BS23, Exercise 10.8].

**Lemma 4.** *Let $\hat{\mathbb{G}}$ be a cyclic group of prime order $q$ with $\hat{g} \in \hat{\mathbb{G}}$ as its generator. Let* **DH** *be the set of all DH-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 $\hat{g}^a \in \hat{\mathbb{G}}$, 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*

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

*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), we will show that $(\hat{g}^a, \hat{g}^b, \hat{g}^z) \in \mathbf{DH}$ implies $(\hat{g}_a, \hat{g}_{b'}, \hat{g}_{z'}) \in \mathbf{DH}$, and $\hat{g}_{b'}$ is uniformly random element in $\hat{\mathbb{G}}$. Since $(\hat{g}^a, \hat{g}^b, \hat{g}^z) \in \mathbf{DH}$, this implies

$$\hat{g}_{b'} = \hat{g}^{b\gamma+\delta}; \quad \text{and} \quad \hat{g}_{z'} = \hat{g}^{ab\gamma} \cdot \hat{g}^{a\delta} = \hat{g}^{a(b\gamma+\delta)};$$

$\hat{g}_{b'}$ being a uniformly random element in $\hat{\mathbb{G}}$ is equivalent to $b\gamma + \delta$ being a uniformly random element in $\mathbb{F}$. This holds because for a fixed $b$ and $x \in \mathbb{F}$:

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

For (2), let $\hat{g}_a = \hat{g}^a, \hat{g}_b = \hat{g}^b, \hat{g}_z = \hat{g}^z$ for some $a, b, z \in \mathbb{F}^3$ with $z \neq ab$. Then, for any $(x, y) \in \mathbb{F}^2$, the constraints $b\gamma + \delta = x$ and $z\gamma + a\delta = y$ implies that

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

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

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

**Lemma 5.** *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, $q_h := \mathsf{poly}(\lambda)$ be an integer, and $\mathbf{T}_a$ be defined as in Lemma 4. Then, assuming the hardness of DDH in $\hat{\mathbb{G}}$, the following two distributions $\mathcal{D}_1, \mathcal{D}_2$ are computationally indistinguishable,*

$$\mathcal{D}_1 := \{(\hat{g}^a, \hat{g}^{b_i}, \hat{g}^{z_i})\}_{i \in [q_h]} \quad \textit{for} \quad (\hat{g}^a, \hat{g}^{b_i}, \hat{g}^{z_i}) \leftarrow \mathbf{T}_a \cap \mathbf{DH}$$
$$\mathcal{D}_2 := \{(\hat{g}^a, \hat{g}^{b_i}, \hat{g}^{z_i})\}_{i \in [q_h]} \quad \textit{for} \quad (\hat{g}^a, \hat{g}^{b_i}, \hat{g}^{z_i}) \leftarrow \mathbf{T}_a$$

*Proof.* Given a DDH tuple $(\hat{g}^a, \hat{g}^b, \hat{g}^z)$ where either $z = ab$ or $z \xleftarrow{\$} \mathbb{F}$, we generate $q_h$ tuples using the re-randomization technique we describe in Lemma 4. This implies that if $(\hat{g}^a, \hat{g}^b, \hat{g}^z) \in \mathbf{DH}$, then the generated vector is identically distributed as $\mathcal{D}_1$. Alternatively, when $(\hat{g}^a, \hat{g}^b, \hat{g}^z) \notin \mathbf{DH}$, then the generated vector is identically distributed as $\mathcal{D}_2$. Thus, if an adversary $\mathcal{A}$ can distinguish between the distribution $\mathcal{D}_1$ and $\mathcal{D}_2$, we can use it to break DDH. $\square$

*Claim.* Assuming the hardness of DDH in $\hat{\mathbb{G}}$, hybrid 2 is computationally indistinguishable from hybrid 1.

*Proof.* Let $\mathcal{A}_{\text{DDH}}$ be the DDH distinguisher. $\mathcal{A}_{\text{DDH}}$ on input $(\hat{g}, \hat{g}^a, \hat{g}^b, \hat{g}^z)$ simulates the DDH game as follows. $\mathcal{A}_{\text{DDH}}$ samples $g \xleftarrow{\$} \mathbb{G}$, $\alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{F}$ and sets $h = g^{\alpha_1}$ and $v = g^{\alpha_2}$. Let $a = \alpha_1 + \alpha_2\phi$ for some $\phi \in \mathbb{F}$. $\mathcal{A}_{\text{DDH}}$ then interacts with $\mathcal{A}$ to simulate the threshold signature protocol where $\mathcal{A}_{\text{DDH}}$ programs the random oracle as follows.

Let $m_k$ be the $k$-th random oracle query. $\mathcal{A}_{\text{DDH}}$ samples $\gamma_k, \delta_k \xleftarrow{\$} \mathbb{F}$, and programs the random oracle queries as:

$$\hat{g}^{b\gamma_k + \delta_k} := \mathsf{H}_0(m_k) \quad \text{and} \quad \hat{g}^{z \cdot \gamma_k} \cdot \hat{g}^{a \cdot \delta_k} := \mathsf{H}_1(m_k) \tag{9}$$

$\mathcal{A}_{\text{DDH}}$ then follows the rest of the protocol as per the specification. Observe that, if $z = ab$, then we program the random oracle exactly as hybrid 2. Alternatively, when $z \xleftarrow{\$} \mathbb{F} \setminus \{ab\}$, then by Lemma 4, the random oracle outputs are independent and uniformly random as in hybrid 1. Thus, by Lemma 5, assuming the hardness of DDH in $\hat{\mathbb{G}}$, hybrid 2 is indistinguishable from hybrid 1. $\square$

**Hybrid 3.** Same as Hybrid 2, except that we sample $r, u \xleftarrow{\$} \mathbb{F} \setminus \{0\}$ and let $\phi = r/u$. Trivially, hybrid 3 is identically distributed as hybrid 2.

**Hybrid 4.** Same as Hybrid 3, except that during the DKG, we use simulated proof of knowledge for $\mathsf{cm}_{\hat{i}}[0]$. Hybrid 4 is identically distributed as hybrid 3 due to the perfect honest verifier zero-knowledge property of the $\Sigma$-protocol used for proof of knowledge.

**Hybrid 5.** Same as Hybrid 4, except that we use simulated proof for correctness of partial signatures. Again, hybrid 5 is identically distributed as hybrid 4 due to the perfect honest verifier zero-knowledge property of the $\Sigma$-protocol used for proving the correctness of the partial signatures.

**Hybrid 6.** Same as Hybrid 5, except that for party $\hat{i}$ we choose the polynomial $s_{\hat{i}}(x)$ such that

$$s_{\hat{i}}(0) = s_0 + r\alpha_1 + u\alpha_2; \quad \text{for} \quad s_0 \xleftarrow{\$} \mathbb{F} \tag{10}$$

Note that for any fixed $\alpha, r$ and $u$, since $s_0$ is chosen uniformly at random, $s_0 + r\alpha_1 + u\alpha_2$ is also uniformly random. This implies that hybrid 6 is identically distributed as hybrid 5.

**Hybrid 7.** Same as Hybrid 5, except that for party $\hat{i}$ we choose polynomials $s_{\hat{i}}(x), r_{\hat{i}}(x)$ and $u_{\hat{i}}(x)$ such that

$$s_{\hat{i}}(0) = s_0; \quad r_{\hat{i}}(0) = r; \quad u_{\hat{i}}(0) = u; \quad \text{for} \quad s_0, r, u \xleftarrow{\$} \mathbb{F} \tag{11}$$

The indistinguishability between hybrid 6 and 7 is another crucial step of our security proof. Next, we prove the following claim.

*Claim.* $\mathcal{A}$'s view in hybrid 7 is identically distributed to its view in hybrid 6.

*Proof.* It is easy to see that the polynomial commitments revealed during the VSS step of DKG are identically distributed in hybrids 6 and 7. Hence, the threshold public keys of parties are identically distributed. Similarly, the partial signatures of signers are also identically distributed in hybrid 6 and 7. Moreover, the simulated proof-of-knowledge NIZK proof and partial signature correctness NIZK proof 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 parties corrupted by $\mathcal{A}$ in hybrid 6 and 7, are identically distributed. Let $\mathcal{C}$ be the set of corrupt parties. For any fixed $\alpha_1, \alpha_2, r, u$, let $\mathcal{D}_6$ and $\mathcal{D}_7$ be the view of $\mathcal{A}$ in hybrid 6 and 7, respectively, i.e.,

$$\begin{aligned} \mathcal{D}_6 &= \{s(k) + r\alpha_1 + u\alpha_2, \quad r(k), \quad u(k)\}_{k\in\mathcal{C}} \\ \mathcal{D}_7 &= \{s(k), \quad r(k) + r, \quad u(k) + u\}_{k\in\mathcal{C}} \end{aligned} \tag{12}$$

Observe that $\mathcal{A}$'s view in both hybrid 6 and 7 are Shamir's secret shares of three secrets using independent random polynomials. Since $|\mathcal{C}| \leq t$, the perfect secrecy of Shamir's secret sharing implies that $\mathcal{D}_6$ and $\mathcal{D}_7$ are identically distributed. Hence, hybrid 6 is identically distributed as hybrid 7. □

**Hybrid 8.** Same as Hybrid 7, except that we use $h = g^a$ from the co-CDH input, and use $\hat{g}^a$ to compute the random oracle outputs. Trivially, hybrid 8 is identically distributed as hybrid 7.

**Hybrid 9.** Same as Hybrid 8, except that we use actual proofs for partial signatures. Hybrid 9 is identically distributed as hybrid 8 due to the perfect honest verifier zero-knowledge property of the Σ-protocol used to prove the partial signatures' correctness.

Observe that hybrid 9 is exactly the simulated protocol. This implies that assuming the hardness of DDH in $\hat{\mathbb{G}}$, $\mathcal{A}$'s view in the real protocol is indistinguishable from its view when it interacts with $\mathcal{A}_{\text{co-CDH}}$. Combining all of the above, we get the following main theorem.

**Theorem 3 (Adaptively secure BLS threshold signature).** *Let $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ be an asymmetric pairing group of prime order $q$. Let $\lambda$ be the security parameter. For any $n, t$ for $n = \mathsf{poly}(\lambda)$ and $t < n/2$, 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 is $(\varepsilon, T, q_s, q_h)$ unforgeable under chosen message attack as per Definition 3 in the presence of an adaptive adversary $\mathcal{A}$ corrupting up to $t$ signers where:*

$$\varepsilon \leq \varepsilon_{\text{DDH}} + q_h \cdot \varepsilon_{\text{CDH}}$$

*where $\varepsilon_{\text{DDH}}$ 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.*

### 6.4 Unforgeability with static adversary

We now 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. 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 $\mathcal{A}_{\text{static}}$ will corrupt during 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. Then, by definition of a static adversary, we require that $|\mathcal{C} \cup \mathcal{S}| = t$ and $\mathcal{A}_{\text{static}}$ declares 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 DKG and signature scheme with $\mathcal{A}_{\text{static}}$ as follows.

**Simulating the DKG protocol.** $\mathcal{A}_{\text{CDH}}$ samples $h, v \xleftarrow{\$} \mathbb{G}$. Next, on behalf of each honest node $i \in \mathcal{H}$, $\mathcal{A}_{\text{CDH}}$ picks random degree $t$ polynomials $r_i(x), u_i(x)$ as per the protocol specification, i.e., $r_i(0) = u_i(0) = 0$. $\mathcal{A}_{\text{CDH}}$ then uses the standard static adversary VSS simulation technique where it picks polynomial $s_i(x)$ for honest party $i$ such that $s_i = s \cdot \alpha_i$ for some uniform random $\alpha_i \in \mathbb{F}$ and it knows $s_i(j)$ for every malicious node $j \in \mathcal{C} \cup \mathcal{S}$. $\mathcal{A}_{\text{CDH}}$ then continue the rest of the simulation as per the standard approach [GJKR07].

**Simulating the signing queries.** Throughout the simulation $\mathcal{A}_{\text{CDH}}$ always faithfully responds to queries to $\mathsf{H}_1$, i.e., for any $\mathsf{H}_1$ query on a new message $m_k$, $\mathcal{A}_{\text{CDH}}$ samples uniformly random $g_k \in \mathbb{G}$ and sets $\mathsf{H}_1(m_k) = g_k$. Note that $\mathsf{H}_0$ is always queried on the forged message, at least by $\mathcal{A}_{\text{CDH}}$ during the signature verification. Let $q_h$ be an upper bound on the number of random oracle queries to $\mathsf{H}_0$, including the query during the signature verification. $\mathcal{A}_{\text{CDH}}$ samples $\hat{k} \xleftarrow{\$} [q_h]$. 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:

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

Let $q_s$ be the maximum number of signing queries made by $\mathcal{A}_{\text{static}}$. We have $q_s \leq q_h$. 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, $\mathcal{A}_{\text{CDH}}$ always 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 Implementation and Evaluation

## 7.1 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 $\Sigma$-protocol proof attesting to the correctness of the partial signatures. Instead of pairings, the aggregator uses the $\Sigma$-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 $\Sigma$-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).

**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 |

### 7.2 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 $\Sigma$-protocol proof. Similarly, our per signer signing cost is $3.3\times$ higher than Boldyreva-II as our $\Sigma$-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 $2.84\times$ longer than Boldyreva-II due to more expensive $\Sigma$-protocol verification. Compared to Boldyreva-I, our partial signature verification is $1.92\times$ 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 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, which, 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.

## 8 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 computational Diffie Hellmann assumption (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 also the assumption of 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 asymmetric pairing groups with no efficiently computable isomorphism between the source 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 Amit Agarwal, Victor Shoup, and Alin Tomescu for helpful discussions related to the paper, Dan Boneh for pointing us to the DDH rerandomization exercise in their book, and Zhoulun Xiang for his feedback on the draft. 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.

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.

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.

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.

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.

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.

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.