

Tightly and Adaptively Secure Two-Round Threshold Signatures from DDH

Renas Bacho¹ and Yanbo Chen²

¹Ruhr University Bochum, renas.bacho@rub.de

²University of Ottawa, ychen918@uottawa.ca

Abstract

Threshold signatures enable a set of n parties to jointly generate signatures under a single public key such that any subset of at least $t + 1$ parties can produce a valid signature, whereas any coalition of at most t parties cannot. They constitute a fundamental primitive for distributed trust and are widely deployed in systems such as certification authorities, blockchains, and multiparty wallets. Modern applications require strong security guarantees under realistic adversarial models, including resistance to adaptive corruptions, tight security reductions, and low interaction complexity—most notably, a minimal number of communication rounds. Recent progress has produced threshold signature schemes in pairing-free groups that achieve tight and adaptive security with three rounds of interaction. This leaves open the central question of whether one can simultaneously achieve tight security, adaptive security, and two-round signing under a standard, non-interactive hardness assumption.

In this work, we resolve this question by presenting **TZAR**, the first two-round threshold signature scheme that is tightly and adaptively secure under the standard decisional Diffie-Hellman (DDH) assumption in pairing-free cyclic groups. Our construction achieves full adaptive security against up to $t < n$ corruptions and admits a tight security reduction with only constant-factor loss. Consequently, **TZAR** provides strong provable security guarantees and minimizes interaction by requiring only two signing rounds, an important advantage for latency-sensitive distributed environments.

Keywords. Threshold Signatures, Tight Security, Adaptive Security, Pairing-Free Groups

1 Introduction

Threshold signatures [Des88, DF89] allow a set of n parties to jointly generate signatures under a single public key such that any subset of at least $t + 1$ parties can produce a valid signature, whereas any coalition of at most t parties cannot. Threshold signatures constitute a fundamental primitive for distributed trust. By eliminating single points of failure and compromise, they play a central role in the security of distributed systems, including certification authorities, secure key management services, blockchains, consensus protocols, and multiparty wallets. In these settings, threshold signatures enhance security by preventing key exposure even in the presence of corrupted parties and enable signing without reliance on a single trusted entity. Beyond correctness, modern applications require strong security guarantees under realistic adversarial models. In particular, adaptive security, where the adversary may corrupt parties dynamically during protocol execution, and tight security reductions, in which the security loss is small and independent of the number of participants, are both highly desirable. At the same time, minimizing interaction—most notably, the number of communication rounds—is crucial for efficiency and practical deployability in latency-sensitive distributed environments.

Recent progress has led to threshold signature schemes with strong security guarantees. In particular, recent works [Che25a, BW26] present threshold signature schemes over pairing-free cyclic groups that achieve tight and adaptive security; however, this comes at the cost of a three-round signing protocol. On the other hand, a more recent work by Bacho and Chen [BC26] presents tightly secure two-round threshold signatures with only static security.¹ This leaves open the following central question, raised by Bacho and Chen [BC26]:

Is it possible to construct a tightly and adaptively secure two-round threshold signature scheme based on a non-interactive and well-studied assumption?

1.1 Our Contributions

We resolve this question by presenting **TZAR**, a novel two-round threshold signature scheme. **TZAR** is the first threshold signature scheme to simultaneously achieve the following properties:

- *Tight Reduction.* We prove security via a tight reduction to the standard decisional Diffie-Hellman (DDH) assumption in pairing-free cyclic groups.
- *Adaptive Security.* The scheme achieves full adaptive security against up to $t < n$ corruptions.
- *Two-Round Signing.* The signing protocol requires only two rounds of communication.

For a comparison with existing threshold signature schemes, see Table 1. We emphasize that our construction does not rely on secure erasure or heavy cryptographic tools. Finally, we note that **TZAR** is the first tightly and adaptively secure threshold signature scheme based solely on *standard, non-interactive assumptions*. Additional related work is discussed in Appendix A.

2 Technical Overview

In this section, we give an overview of our techniques.

2.1 A Recap on Dazzle

We recall **Dazzle** [Che25a], a recent two-round threshold signature scheme that achieves full adaptive security, albeit with a non-tight reduction. Its base construction is a threshold variant of the Chaum–Pedersen signature scheme [CP93]. Let $\mathbb{G}$ be a cyclic group of prime order p with generator g . The Chaum–Pedersen signature scheme has public key $X = g^x$, where the secret key is $x \leftarrow \mathbb{Z}_p$. A Chaum–Pedersen signature on a message m consists of the BLS-style signature element $U = u^x$ [BLS01], where $u = H_u(m)$, together with a Chaum–Pedersen proof that U is well formed with respect to X , i.e., that $\log_g X = \log_u U$. The proof is a pair (c, z) satisfying $H_c(X, R_1, R_2, U, m) = c$, $R_1 = g^z X^{-c}$, and $R_2 = u^z U^{-c}$.

To turn the Chaum–Pedersen signature scheme into a threshold scheme for n users with threshold t , the secret key x is Shamir-shared [Sha79] as $\{x_i\}_{i \in [n]}$. For any authorized set $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| = t + 1$, the secret key can be reconstructed as $x = \sum_{i \in \mathcal{S}} \lambda_{\mathcal{S}, i} x_i$, where $\lambda_{\mathcal{S}, i}$ denotes the corresponding Lagrange coefficient. A signer set $\mathcal{S}$ then generates a Chaum–Pedersen threshold signature via the following two-round protocol:

- In the first round, each signer $i \in \mathcal{S}$ samples $r_i \leftarrow \mathbb{Z}_p$, sets $u := H_u(m)$, and computes $U_i := u^{\lambda_{\mathcal{S}, i} x_i}$, $R_{1, i} := g^{r_i}$, and $R_{2, i} := u^{r_i}$. It then multicasts (i.e., sends to all parties) $(U_i, R_{1, i}, R_{2, i})$ as its first-round message.
- In the second round, each signer $i \in \mathcal{S}$ computes $U := \prod_{j \in \mathcal{S}} U_j$, $R_1 := \prod_{j \in \mathcal{S}} R_{1, j}$, and $R_2 := \prod_{j \in \mathcal{S}} R_{2, j}$. It then derives the challenge $c := H_c(X, R_1, R_2, U, m)$ and computes its response share $z_i := r_i + c \lambda_{\mathcal{S}, i} x_i$. It then multicasts z_i as its second-round message.

¹We note that their techniques crucially rely on corruptions being static and do not extend to the adaptive setting. For a discussion of this point, we refer to Section 2.4 (“Open Questions”) of their paper.

Scheme	Rounds	Assumption	Security loss	Adaptive	AGM	Signature
Crackle [KRT24]	5	DLOG	$\Theta(q^3/\varepsilon)$	✓	✗	$2\langle\mathbb{Z}_p\rangle$
Glacius [BDLR25b]	5	DDH	$\Theta(q/\varepsilon)$	✓	✗	$2\langle\mathbb{Z}_p\rangle$
Classic-S [Mak22]	3	DLOG	$\Theta(q/\varepsilon)$	✗	✗	$2\langle\mathbb{Z}_p\rangle$
Zero-S [Mak22]	3	DLOG	$\Theta(q/\varepsilon)$	✓	✗	$2\langle\mathbb{Z}_p\rangle$
Sparkle [CKM23]	3	DLOG	$\Theta(q/\varepsilon)$	✗	✗	$2\langle\mathbb{Z}_p\rangle$
Sparkle [CKM23]	3	AOMDL	$\Theta(q/\varepsilon)$	(✓)	✗	$2\langle\mathbb{Z}_p\rangle$
Sparkle [CKM23]	3	AOMDL, LDVR	$\Theta(1)$	✓	✓	$2\langle\mathbb{Z}_p\rangle$
Gargos [BDLR25a]	3	DDH	$\Theta(q/\varepsilon)$	✓	✗	$2\langle\mathbb{Z}_p\rangle$
FROST-Mask [Che25b]	3	AOMDL	$\Theta(q/\varepsilon)$	✓	✗	$2\langle\mathbb{Z}_p\rangle$
Rackle [NRT25]	3	DLOG	$\Theta(q^3/\varepsilon)$	✓	✗	$2\langle\mathbb{Z}_p\rangle$
Twinkle [BLT+24]	3	DDH	$\Theta(q)$	✓	✗	$2\langle\mathbb{G}\rangle + 3\langle\mathbb{Z}_p\rangle$
FROST [KG20]	2	AOMDL	$\Theta(q/\varepsilon)$	✗	✗	$2\langle\mathbb{Z}_p\rangle$
FROST [CKK+25]	2	AOMDL	$\Theta(q/\varepsilon)$	(✓)	✗	$2\langle\mathbb{Z}_p\rangle$
FROST [CKK+25]	2	AOMDL, LDVR	$\Theta(1)$	✓	✓	$2\langle\mathbb{Z}_p\rangle$
ms-FROST [BCL+25]	2	AOMDL	$\Theta(1)$	✓	✓	$2\langle\mathbb{Z}_p\rangle$
FaFROST [BGC+25]	2	AOMDL	$\Theta(1)$	✓	✓	$2\langle\mathbb{Z}_p\rangle$
GCRS [GCRS25]	2	DDH	$\Theta(q/\varepsilon)$	✓	✗	$2\langle\mathbb{Z}_p\rangle$
HBTS-Mask [Che25b]	2	DLOG	$\Theta(q^3/\varepsilon)$	✓	✗	$1\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$
TZ [TZ23]	2	DLOG	$\Theta(q/\varepsilon)$	✗	✗	$3\langle\mathbb{Z}_p\rangle$
Dazzle [Che25a]	2	DDH	$\Theta(q)$	✓	✗	$1\langle\mathbb{G}\rangle + 3\langle\mathbb{Z}_p\rangle$
Dazzle-T [Che25a]	3	DDH	$\Theta(1)$	✓	✗	$1\langle\mathbb{G}\rangle + 3\langle\mathbb{Z}_p\rangle$
Twinkle-T [BW26]	3	DDH	$\Theta(1)$	✓	✗	$2\langle\mathbb{G}\rangle + 4\langle\mathbb{Z}_p\rangle$
Earpicks-TS [BC26]	2	DDH	$\Theta(1)$	✗	✗	$3\langle\mathbb{Z}_p\rangle + 1$
TZAR (ours)	2	DDH	$\Theta(1)$	✓	✗	$1\langle\mathbb{G}\rangle + 7\langle\mathbb{Z}_p\rangle$

Table 1: Comparison of threshold signature schemes over pairing-free cyclic groups. We consider number of signing rounds, computational hardness assumption, security loss of the reduction (for an adversary with advantage ε against the scheme, making at most q random oracle and signing queries), adaptive security proof, reliance on the algebraic group model (AGM) [FKL18], and size of signatures. We do not consider schemes executing DKG-type protocols over broadcast channels during signing. We denote by $\langle\mathbb{G}\rangle$ and $\langle\mathbb{Z}_p\rangle$ the sizes of a group element and a scalar field element, respectively, and by λ the security parameter. *Additional notes:* (✓) in column “Adaptive” means the security proof assumes $t/2$ adaptive corruptions. Furthermore, Zero-S [Mak22] relies on erasures for adaptive security, and GCRS [GCRS25] uses Bulletproof-type NIZK proofs.

- Finally, the response shares are aggregated as $z := \sum_{j \in \mathcal{S}} z_j$. The signature is output as (U, c, z) .

Security. We first recall the security proof of the Chaum–Pedersen signature scheme under the CDH assumption. Given group parameters $(\mathbb{G}, p, g)$, an element $h \leftarrow \mathbb{G}$, and $A = g^a$ for $a \leftarrow \mathbb{Z}_p$, the CDH problem is to compute h^a . The security reduction sets the public key as $X := A$ and must simulate the security game without knowing the corresponding secret key $x = a$. To do so, it partitions the message space into two sets:

- For messages m in the “signing partition”, the reduction programs $u = H_u(m)$ as $u = g^b$

for $b \leftarrow \mathbb{Z}_p$. The reduction expects the adversary to request signatures only on messages in this partition. It can then simulate a signature by setting $U = X^b$ and using the standard honest-verifier zero-knowledge (HVZK) simulator to generate the proof (c, z) .

- For messages m in the “forgery partition”, the reduction programs $u = H_u(m)$ as $u = hg^b$ for $b \leftarrow \mathbb{Z}_p$. The reduction expects the adversary to output a valid forgery (U_*, c_*, z_*) on some message m_* in this partition. By soundness of the non-interactive proof, U_* is a valid BLS signature under X , namely $U_* = u_*^x$, where $u_* = H_u(m_*) = hg^{b_*}$ for known b_* . Since $X = g^x$, the reduction outputs $U_*/X^{b_*} = (hg^{b_*})^x / (g^x)^{b_*} = h^x = h^a$, which solves the given CDH instance.

The adaptive security of **Dazzle** relies on the algebraic one-more computational Diffie–Hellman (AOMCDH) assumption. Given group parameters $(\mathbb{G}, p, g)$, an element $h \leftarrow \mathbb{G}$, and challenge elements $A_1 = g^{a_1}, \dots, A_q = g^{a_q}$, the AOMCDH problem is to compute $h^{a_1}, \dots, h^{a_q}$ using at most $q - 1$ queries to a discrete-logarithm oracle DLOG. On input an algebraic representation $(\beta_1, \dots, \beta_q) \in \mathbb{Z}_p^q$ of an element $A = \prod_{i \in [q]} A_i^{\beta_i}$, the oracle returns the discrete logarithm of the input element $\text{DLOG}(A) = \sum_{i \in [q]} \beta_i a_i$. The DLOG oracle is used in two crucial steps of the adaptive-security reduction:

- *Signing queries.* The HVZK simulator cannot be applied directly in the two-round signing protocol. Instead, the reduction simulates signing with the help of DLOG. For each signing query, it embeds a challenge element in each commitment $R_{1,i}$ and uses the signing trapdoor b to compute $R_{2,i} := R_{1,i}^b$. In the second round, after the challenge c is fixed, the reduction queries DLOG to obtain $z_i := \text{DLOG}(R_{1,i} X_i^c)$.
- *Corruption queries.* The reduction embeds challenge elements in the public key shares and uses DLOG to answer adaptive corruption queries, which require revealing the secret key shares of an arbitrary set of up to t signers. Specifically, it embeds $t + 1$ challenge elements in the public key and its shares. Upon a corruption query, it queries DLOG to recover the requested secret key share.

DDH-Based Schemes. The authors further transform the above AOMCDH-based variant of **Dazzle** into a DDH-based one using the technique of **Twinkle** [BLT⁺24].² At a high level, this transformation proceeds as follows.

- First, the scheme is abstracted in terms of tagged linear functions. Exponentiation with base g is viewed as the function $F_g(x) = g^x$, while exponentiation with base u is viewed as the function $F'_u(x) = u^x$.
- Security is then based on “*Algebraic Translation Resistance*”, which can be viewed as an analogue of AOMCDH for tagged linear functions. Given a tag g for F , a tag u for F' , and challenge images $F_g(a_1), \dots, F_g(a_q)$, the task is to compute the corresponding images $F'_u(a_1), \dots, F'_u(a_q)$ using at most $q - 1$ queries to an F_g -preimage oracle.
- Finally, the linear functions are replaced by more complex ones. Namely, F is replaced by $F_{(g, \hat{g}, h, \hat{h})}(x, \hat{x}) = (g^x \hat{g}^{\hat{x}}, h^x \hat{h}^{\hat{x}})$, and F' is replaced by $F'_{(u, \hat{u})}(x, \hat{x}) = u^x \hat{u}^{\hat{x}}$. Algebraic Translation Resistance for these functions tightly reduces to the DDH assumption.

In this overview, we focus on the AOMCDH-based schemes. In the main body, we directly present our DDH-based construction.

Looseness of Security Reductions. **Dazzle** incurs a security loss of $\Theta(q_s)$, where q_s denotes the number of signing queries. This loss stems from the partitioning proof technique [Cor00], inherited from the CDH-based proof for Chaum–Pedersen signatures.

Dazzle-T [Che25a], a three-round threshold signature scheme with a tight security reduction proposed in the same work, avoids this loss by replacing the underlying one-party scheme with the

²Both **Dazzle** and **Twinkle** have a more efficient variant based on AOMCDH and a slightly less efficient variant based on DDH.

Chevallier–Mames signature scheme [Che05], a variant of Chaum–Pedersen in which u is computed as $H_u(R_1, m)$ rather than as $H_u(m)$. Here, the generation of R_1 is independent of u . However, thresholdizing the Chevallier–Mames scheme requires three rounds: the signers must first exchange the commitments $R_{1,i}$ in order to compute R_1 and hence derive u , after which they can compute U_i and $R_{2,i}$.

2.2 Our Construction

We begin with the observation that the Chaum–Pedersen signature scheme admits a *tight* security reduction to the AOMCDH assumption. The reduction sets the public key to $X := h$ and embeds the challenge elements in the responses to H_u queries. Specifically, for each distinct query $H_u(m)$, the reduction programs the random oracle to return the next unused challenge element A_i . When asked to sign such a message m , it queries $\text{DLOG}(A_i)$ to obtain the corresponding signing trapdoor a_i . It then computes the BLS element $U := X^{a_i} = A_i^{a_i}$ and uses the HVZK simulator to generate the accompanying Chaum–Pedersen proof.

The reduction is tight because it avoids the standard guessing or partitioning step at the time of the $H_u(m)$ query. Instead, access to the DLOG oracle enables *partitioning on the fly*: the reduction need not decide in advance whether a queried message will later be signed or forged.

However, proving adaptive security for the threshold variant introduces additional complications. In the above reduction, the embedding positions of the second base h and the challenge elements are swapped. Recall that in the proof for **Dazzle**, the AOMCDH challenge elements $g^{a_1}, \dots, g^{a_q}$ are embedded in the public key shares and commitments $R_{1,i}$, while h is embedded in the H_u outputs. The DLOG oracle can handle signing and corruption queries, as they only require the discrete logarithms of linear combinations of public key shares and $R_{1,i}$. On the contrary, to use DLOG oracle to obtain signing trapdoors, here the AOMCDH challenge elements have to be embedded to H_u outputs. Consequently, h is embedded to the main public key to make U constitute a CDH solution. This makes it unclear how to program the public key shares and commitments $R_{1,i}$ such that signing and corruption queries can be handled by DLOG, which only accepts linear combinations of the challenge elements. Thus, to obtain a *tight reduction under adaptive corruptions*, we must modify the scheme so that the reduction can operate within these restrictions.

Step 1: Homomorphic Dual-Mode Commitments. We first disregard adaptive corruption queries and focus on static security. Recall that, in the proof of **Dazzle**, the reduction relies on the DLOG oracle to simulate the two-round signing protocol. In the absence of this oracle, we instead employ homomorphic dual-mode commitments [BCJ08, PW23] to recover the HVZK simulation strategy.

Such commitments admit two modes for generating the commitment key ck . In the *hiding mode*, ck is generated together with a weak equivocation trapdoor $\mathbf{D}$. In the *binding mode*, commitments are statistically binding. In pairing-free groups, known dual-mode commitments provide only weak equivocation: a commitment to an element $R' \in \mathbb{G}$ can be opened, using the trapdoor, to an arbitrary shift $R = R' \cdot X^\beta$, where X is embedded in ck and the shift $\beta \in \mathbb{Z}_p$ may be determined only after the commitment is generated, for example from the challenge c . As a first step toward the final construction, we use such a homomorphic dual-mode commitment to modify the threshold signature scheme as follows.

- *Setup.* The public parameters additionally include a commitment key ck for the homomorphic dual-mode commitment scheme.
- *Signing protocol.*
 - In the first round, instead of sending $(R_{1,i}, R_{2,i})$ in the clear, each signer multicasts a commitment com_i to $(R_{1,i}, R_{2,i})$ under ck .
 - In the second round, the signers aggregate the commitments as $\text{com} := \bigotimes_{j \in \mathcal{S}} \text{com}_j$. By homomorphism, com is a commitment to $(R_1, R_2) := \prod_{j \in \mathcal{S}} (R_{1,j}, R_{2,j})$. The challenge is

then computed as $c := H_c(\text{pk}, \text{com}, U, m)$. In addition to z_i , each signer multicasts the randomness ρ_i used to generate com_i , i.e., the opening information for its commitment.

- In the final signature, the proof of validity of the BLS signature additionally includes the aggregated commitment randomness $\rho := \sum_{i \in \mathcal{S}} \rho_i$. By homomorphism, ρ is a valid opening of the aggregated commitment com .

- *Signature verification.* To verify a signature (U, c, z, ρ) , the verifier first recomputes R_1 and R_2 from (c, z) . It then recomputes the commitment com from (R_1, R_2) and ρ , and checks whether $c = H_c(\text{pk}, \text{com}, U, m)$.

We note that homomorphic dual-mode commitments have been used in several two-round multi-signature schemes [PW23, BW25, BC26], including all known tightly secure constructions. However, these constructions derive the commitment key from the message m to be signed, e.g., as $\text{ck} := H(m)$ for a random oracle H . Crucially, their security proofs still rely on partitioning the message space into signing and forgery regions, possibly after applying the Katz–Wang technique [KW03, GJKW07] to obtain the extended message spaces $0\|m$ and $1\|m$. The commitment key is then generated in hiding mode for the signing region and in binding mode for the forgery region. In Supp. Mat. A.1, we explain in detail why these approaches fail to extend to the adaptive threshold setting. In particular, using homomorphic dual-mode commitments in our setting requires a different proof strategy. For the scheme described above, our proof strategy proceeds as follows.

Proving Security. The security proof proceeds via a sequence of hybrid games. First, we switch the commitment key ck to binding mode. In this mode, every commitment com has a unique opening to some (R_1, R_2) . Moreover, the non-interactive proof remains statistically sound even though (R_1, R_2) are committed rather than sent in the clear. Hence, any valid signature must satisfy $U = u^x = X^a$, where $u = g^a$. We can therefore move to a game that, at the end, checks the forged signature by testing whether $U_* = X^{a_*}$, where the signing trapdoor a is known for every message m , including a_* for the forgery message m_* . This replaces explicit verification of the forged signature.

Then, we switch ck to hiding/equivocable mode. In this mode, the game knows a trapdoor $\mathbf{D}$ that allows it to (weakly) equivocate commitments. Hence, in the signing oracle, the game can defer the choice of $(R_{1,i}, R_{2,i})$ until the second round, thereby enabling HVZK simulation. Specifically, once the challenge c is fixed, the game samples $z_i \leftarrow \mathbb{Z}_p$ and sets $R_{1,i} := g^{z_i} X_i^{-c}$, $R_{2,i} := u^{z_i} U_i^{-c}$. It then uses $\mathbf{D}$ to equivocate com_i to $(R_{1,i}, R_{2,i})$. Thus, signing queries can be answered using only the signing trapdoor a such that $u = g^a$, together with the commitment trapdoor, and without knowing the secret key share x_i . Note that this strategy is compatible with weak equivocation: the game samples z_i and initially commits to (g^{z_i}, u^{z_i}) in the first round. In the second round, once c is known, it equivocates the commitment by the shift $(X_i^{-c}, U_i^{-c}) = (X_i^{-c}, X_i^{-ac})$, where X_i is embedded in ck .

There remains a gap in the above proof strategy: only one public key share X_i is embedded in ck , which allows the reduction to simulate signing for signer i ; however, in the threshold setting, there may be multiple honest signers, each with a different public key share. A natural idea would be to use separate commitment keys $\text{ck}_1, \dots, \text{ck}_n$, with X_i embedded in ck_i . This does not work, since the known dual-mode commitments are homomorphic only under a common commitment key ck ; moreover, we want the final threshold signature size to remain independent of n . Embedding the joint public key $X = \text{pk}$ into ck is more promising, since it avoids committing in advance to which signer will be simulated. Indeed, in the static-corruption setting, this already suffices: since the set of honest signers is fixed in advance, the reduction can derive consistent signature shares from the final threshold signature using interpolation.

Under adaptive corruptions, however, the above approach breaks down, since the adversary may later request the secret key shares and internal randomness underlying the simulated shares. Of course, we also have to consider the central challenge in the adaptive setting: how the reduction answers corruption queries.

Step 2: One-Time Masking to the Rescue. We now show how the above obstacles in the adaptive-corruption setting can be addressed simultaneously using the masking technique [KRT24], a tool used in many recent works on adaptively secure threshold signatures. In the original masking technique, parties use one-time shares $\{\Delta_i\}_i$ of zero to mask their signature shares. Consequently, the signing oracle leaks no information to the adversary until a signing session is completed consistently by all honest signers. At that point, the adversary receives only the final threshold signature, but never any valid individual signature shares. Thus, each secret key share x_i remains *perfectly* hidden until it is revealed by a corruption query. In particular, the adversary never observes the public key shares $\{X_i = g^{x_i}\}_{i \in [n]}$; i.e., there are no individual public key shares X_i anymore. The reduction can therefore answer each corruption query by sampling the corresponding secret key share uniformly as $x_i \leftarrow \mathbb{Z}_p$.

We use masking in two places. First, we mask the parties' signature shares U_i and z_i . Second, we mask the individual opening randomness values ρ_i . The latter prevents the adversary from seeing openings of the individual commitments before an adaptive corruption occurs; instead, it only sees an opening of the aggregated commitment $\text{com} = \bigotimes_{i \in \mathcal{S}} \text{com}_i$. Since the joint public key pk is embedded in ck , this aggregated commitment can be equivocated using the known trapdoor.

Specifically, the key generation algorithm additionally generates symmetric keys $\{\text{seed}_{i,j}\}_{i,j \in [n]}$. For each user $i \in [n]$, the secret key share x_i is augmented with $\{(\text{seed}_{i,j}, \text{seed}_{j,i})\}_{j \in [n]}$. In the first round, signer i masks its signature share $u^{\lambda_{\mathcal{S},i}x_i}$ using a mask $\Delta_{i,1} \in \mathbb{G}$ derived from $(\text{seed}_i, \mathcal{S}, m, 1)$, and computes $U_i := u^{\lambda_{\mathcal{S},i}x_i} \Delta_{i,1}$.³ In the second round, signer i masks its response share $r_i + c\lambda_{\mathcal{S},i}x_i$ using a mask $\Delta_{i,2} \in \mathbb{Z}_p$ derived from $(\text{seed}_i, \mathcal{S}, m, \mathcal{M}_1, 2)$, where $\mathcal{M}_1$ denotes the set of first-round protocol messages. It also masks its commitment randomness ρ_i using a mask $\Delta_{i,3} \in \mathbb{Z}_p$ derived analogously from $(\text{seed}_i, \mathcal{S}, m, \mathcal{M}_1, 3)$. If all masks are derived from consistent values of $(\mathcal{S}, m, \mathcal{M}_1)$, then $\prod_{i \in \mathcal{S}} \Delta_{i,1} = 1_{\mathbb{G}}$ and $\sum_{i \in \mathcal{S}} \Delta_{i,2} = \sum_{i \in \mathcal{S}} \Delta_{i,3} = 0$. We stress that the masks must be one-time: reusing the same mask for two different values may leak information, such as their difference. Since the first-round transcript $\mathcal{M}_1$ has high entropy, the second-round masks $\Delta_{i,2}, \Delta_{i,3}$ derived from $(\mathcal{S}, m, \mathcal{M}_1)$ are one-time. For the first round, the crucial observation is that $u^{\lambda_{\mathcal{S},i}x_i}$ is uniquely determined by $(\mathcal{S}, m)$. Thus, masking it with $\Delta_{i,1}$, which is itself derived only from $(\mathcal{S}, m)$, is sufficient. In contrast, the commitment com_i is randomized and therefore should not be masked using a mask determined only by $(\mathcal{S}, m)$. Nevertheless, the reduction can handle corruptions even when com_i is sent in the clear.

Conclusion. Finally, extending our techniques to the DDH-based scheme introduces several additional technical challenges, which we do not discuss here. While the preceding discussion gives an intuitive overview of our proof, the full security proof (cf. Section 4.3) is considerably more delicate: it proceeds through a careful sequence of 24 hybrid games, organized into three lemmas for modularity. We also note that dual-mode commitments in pairing-free groups have previously been used only for linear functions of the form $x \mapsto g^x$ or $x \mapsto (g, h)^x$. In contrast, our setting requires handling linear functions of the form $(x, y) \mapsto g^x h^y$.

Although our signatures are twice as large as those of **Dazzle**, the best known fully adaptive two-round threshold signature scheme, **Dazzle** does not achieve a tight security reduction. Our work shows that tight adaptive security can be achieved in two rounds, thereby reducing the round complexity of prior tightly secure adaptively secure threshold signatures by one round. Designing tight two-round multi-party signing protocols is highly challenging, especially against adaptive corruptions. Indeed, **Chopsticks** [PW23] and **T-Spoon** [BW25] were the first tight two-round multi-signature schemes of their kind, without and with key aggregation, respectively: **Chopsticks** has signatures of 14 field/group elements, while **T-Spoon** has signatures of 11 field/group elements. Nevertheless, these works initiated a line of improvements, eventually yielding signatures with only

³For technical reasons, the masking terms applied to the shares $u^{\lambda_{\mathcal{S},i}x_i}$ must themselves be group elements in $\mathbb{G}$. By abuse of terminology, we also call them zero-sharing masks, although they aggregate multiplicatively to the identity element $1_{\mathbb{G}} \in \mathbb{G}$ (or simply 1) rather than additively to the field element $0 \in \mathbb{Z}_p$.

3–4 elements [BC26]. We believe that our work will similarly serve as a starting point for further improvements.

3 Preliminaries

Let λ denote the security parameter. For a finite set S , we write $s \leftarrow S$ to denote that s is sampled uniformly at random from S , and we use $|S|$ to denote the size of S . For a positive integer $n \in \mathbb{N}$, we write $[n]$ to denote the ordered set $\{1, \dots, n\}$. Moreover, “ $:=$ ” denotes deterministic assignment. Unless explicitly stated otherwise, all algorithms are assumed to be probabilistic and are written in uppercase serif font.

Shamir Secret Sharing. Shamir’s secret sharing scheme [Sha79] embeds a secret $s \in \mathbb{Z}_p$ as the constant term of a random polynomial $f(X) = s + a_1X + \dots + a_tX^t \in \mathbb{Z}_p[X]$, where the coefficients $a_1, \dots, a_t \leftarrow \mathbb{Z}_p$ are sampled uniformly at random. The i -th share of the secret is defined as $s_i := f(i)$. Any $t + 1$ distinct shares suffice to recover the polynomial (and hence the secret s) using Lagrange interpolation. For our construction, we denote by $\text{Shamir}(n, t, s)$ the algorithm that outputs the n Shamir shares $s_1, \dots, s_n$ of s using a degree- t polynomial. Further, for any index $i \in [n]$ and any subset $S \subseteq [n]$ of size $t + 1$ with $i \notin S$, the value $s_i = f(i)$ can be computed from the shares $\{s_j\}_{j \in S}$ as the linear combination $s_i = \sum_{j \in S} \lambda_{i,S,j} \cdot s_j$, where the Lagrange coefficient $\lambda_{i,S,j} \in \mathbb{Z}_p$ is given by $\lambda_{i,S,j} = \prod_{k \in S, k \neq j} (i - k)/(j - k)$. When $i = 0$, we write the Lagrange coefficient in short as $\lambda_{S,j} := \lambda_{0,S,j}$.

3.1 Computational Assumptions

Let GrGen be a group generation algorithm that, on input the security parameter 1^λ , outputs group parameters $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is a cyclic group of prime order p and $g \in \mathbb{G}$ is a generator of $\mathbb{G}$. Our construction relies on the DDH assumption. For technical convenience, we also consider the closely related uDDH4 and mDDH4 assumptions, both of which are tightly implied by DDH [EHK⁺13, PW23, BCP02]. We formally define these assumptions below.

Definition 1 (DDH Assumption). The DDH problem is said to be (τ, ε) -hard for the group generation algorithm GrGen if, for all $\lambda \in \mathbb{N}$ and all algorithms A running in time at most τ , we have

$$\begin{aligned} & \left| \Pr \left[A(\mathbb{G}, p, g, h, g^a, h^a) = 1 \mid (\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda), h \leftarrow \mathbb{G}, a \leftarrow \mathbb{Z}_p \right] \right. \\ & \quad \left. - \Pr \left[A(\mathbb{G}, p, g, h, \hat{g}, \hat{h}) = 1 \mid (\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda), h, \hat{g}, \hat{h} \leftarrow \mathbb{G} \right] \right| \leq \varepsilon. \end{aligned}$$

Definition 2 (uDDH4 Assumption). The uDDH4 problem is said to be (τ, ε) -hard for the group generation algorithm GrGen if, for all $\lambda \in \mathbb{N}$ and all algorithms A running in time at most τ , we have

$$\begin{aligned} & \left| \Pr \left[A(\mathbb{G}, p, g, (h_{i,j})_{i,j \in [4]}) = 1 \mid \begin{array}{l} (\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda), \\ a, b, c \leftarrow \mathbb{Z}_p, h_{1,1}, h_{2,1}, h_{3,1}, h_{4,1} \leftarrow \mathbb{G}, \\ \forall i \in [4] : h_{i,2} := h_{i,1}^a, h_{i,3} := h_{i,1}^b, h_{i,4} := h_{i,1}^c \end{array} \right] \right. \\ & \quad \left. - \Pr \left[A(\mathbb{G}, p, g, (h_{i,j})_{i,j \in [4]}) = 1 \mid \begin{array}{l} (\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda), \\ \forall (i,j) \in [4] \times [4] : h_{i,j} \leftarrow \mathbb{G} \end{array} \right] \right| \leq \varepsilon. \end{aligned}$$

Definition 3 (mDDH4 Assumption). The mDDH4 problem is (τ, ε) -hard for the group generation algorithm GrGen if, for all $\lambda \in \mathbb{N}$ and all algorithms A running in time at most τ , we have

$$\begin{aligned} & \left| \Pr \left[A(\mathbb{G}, p, g, h, (g^{a_i}, h^{a_i})_{i \in [4]}) = 1 \mid \begin{array}{l} (\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda), \\ h \leftarrow \mathbb{G}, a_1, \dots, a_4 \leftarrow \mathbb{Z}_p \end{array} \right] \right. \\ & \quad \left. - \Pr \left[A(\mathbb{G}, p, g, h, (\hat{g}_i, \hat{h}_i)_{i \in [4]}) = 1 \mid \begin{array}{l} (\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda), \\ h, \hat{g}_1, \hat{h}_1, \dots, \hat{g}_4, \hat{h}_4 \leftarrow \mathbb{G} \end{array} \right] \right| \leq \varepsilon. \end{aligned}$$

3.2 Two-Round Threshold Signatures

We define the syntax and security of (two-round) threshold signatures in the trusted key generation model, which is standard in the recent threshold signature literature [BLT⁺24, KRT24].

Syntax. A (t, n) -threshold signature scheme is a tuple of algorithms $\text{TS} = (\text{Setup}, \text{KGen}, \text{Sign}, \text{Ver})$ defined as follows. The setup algorithm Setup outputs system parameters par , and the key generation algorithm KGen outputs a public key pk together with secret key shares $\text{sk}_1, \dots, \text{sk}_n$. The signing protocol Sign is further decomposed into three algorithms $(\text{Sign}_1, \text{Sign}_2, \text{Comb})$. Informally, each algorithm Sign_j specifies how a signer locally computes its protocol message msg_{j+1} for the subsequent round and updates its internal state. The combine algorithm Comb then converts the transcript of all protocol messages into a single signature σ , which can be verified using the verification algorithm Ver under the public key pk .

Completeness of the scheme requires that, for all parameters, keys, and messages, any signing protocol execution among (at least) $t + 1$ honestly behaving parties outputs a signature σ for which $\text{Ver}(\text{pk}, m, \sigma) = 1$. We model completeness using the algorithm TSJointSign in Figure 1.

Definition 4 (Two-Round Threshold Signature Schemes). A two-round threshold signature scheme TS consists of the following algorithms:

- $\text{Setup}(1^\lambda) \rightarrow \text{par}$: On input the security parameter 1^λ , the setup algorithm Setup outputs global parameters par . All algorithms related to TS implicitly take par as input.
- $\text{KGen}(n, t) \rightarrow (\text{pk}, \{\text{sk}_i\}_{i \in [n]})$: On inputs a number of user n and a threshold t , the key generation algorithm KGen outputs a public key pk and n secret key shares $\{\text{sk}_i\}_{i \in [n]}$.
- The signing protocol consists of three algorithms:
 - $\text{Sign}_1(\text{pk}, \mathcal{S}, m, i, \text{sk}_i) \rightarrow (\text{st}_i, \text{msg}_{1,i})$: On inputs a public key pk , a set of signer indices $\mathcal{S} \subseteq [n]$, a message m , a signer index i , a secret key share sk_i , the first-round signing algorithm Sign_1 outputs a state st_i and a first-round protocol message $\text{msg}_{1,i}$.
 - $\text{Sign}_2(\text{st}_i, \text{sk}_i, \mathcal{M}_1) \rightarrow \text{msg}_{2,i}$: On inputs a state st_i , a secret key share sk_i , and a set of first-round protocol messages $\mathcal{M}_1$, the second-round signing algorithm Sign_2 outputs a second-round protocol message $\text{msg}_{2,i}$.
 - $\text{Comb}(\text{pk}, \mathcal{S}, m, \mathcal{M}_1, \mathcal{M}_2) \rightarrow \sigma$: On inputs a public key pk , a set of signer indices $\mathcal{S} \subseteq [n]$, a message m , a set of first-round protocol messages $\mathcal{M}_1$, and a set of second-round protocol messages $\mathcal{M}_2$, the combining algorithm outputs a signature σ .
- $\text{Ver}(\text{pk}, m, \sigma) \rightarrow 0/1$: On inputs a public key pk , a message m , and a signature σ , the deterministic verification algorithm Ver outputs 0 (reject) or 1 (accept).

For completeness, we require that for all $\lambda \in \mathbb{N}$, all $\text{par} \in \text{Setup}(1^\lambda)$, all $t < n \in \text{poly}(\lambda)$, all $(\text{pk}, \{\text{sk}_i\}_{i \in [n]}) \in \text{KGen}(n, t)$, all subsets $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| \geq t + 1$, and all messages m , we have

$$\Pr[\text{Ver}(\text{pk}, m, \sigma) = 1 : \sigma \leftarrow \text{TSJointSign}(\text{pk}, \mathcal{S}, m, \{\text{sk}_i\}_{i \in \mathcal{S}})] = 1,$$

where the algorithm TSJointSign , modeling an honest execution of TS , is defined in Figure 1.

Security. We define the security of two-round threshold signature schemes via the security game TSUF described in Figure 2. We now give an informal description of this game. The adversary $\mathbf{A}$ is given the system parameters par and the global public key pk as input. At any point during the game, $\mathbf{A}$ may corrupt an honest party i by querying the oracle $\text{CORR}(i)$, for up to t distinct parties in total. Upon corruption, the adversary obtains party i 's secret key sk_i as well as all the randomness that party i has used in signing sessions, including both ongoing and completed sessions. This allows the adversary to recover the entire computational history of every session. In particular, we stress that we do not assume secure erasures of secret state. The adversary may also concurrently initiate multiple signing sessions with any honest signer. In each session, $\mathbf{A}$ specifies an honest signer index

Alg TSJointSign(pk, $\mathcal{S}$, m , $\{\text{sk}_i\}_{i \in \mathcal{S}}$):

```

1: for  $i \in \mathcal{S}$  do
2:    $(\text{st}_i, \text{msg}_{1,i}) \leftarrow \text{Sign}_1(\text{pk}, \mathcal{S}, m, i, \text{sk}_i)$ 
3: for  $i \in \mathcal{S}$  do
4:    $\text{msg}_{2,i} \leftarrow \text{Sign}_2(\text{st}_i, \text{sk}_i, \{\text{msg}_{1,j}\}_{j \in \mathcal{S}})$ 
5: return  $\sigma \leftarrow \text{Comb}(\text{pk}, \mathcal{S}, m, \{\text{msg}_{1,j}\}_{j \in \mathcal{S}}, \{\text{msg}_{2,j}\}_{j \in \mathcal{S}})$ 

```

Figure 1: Algorithm TSJointSign for a (two-round) threshold signature scheme TS. The algorithm models an honest execution of the signing protocol.

Game TSUF_{TS}^A($1^\lambda, n, t$):

```

1:  $\mathcal{H} := [n]$ ,  $\mathcal{C} := \emptyset$ ,  $\mathcal{Q} := \emptyset$ 
2: for  $i \in [n]$  do
3:    $\text{ctr}_i := 0$ 
4:  $\text{par} \leftarrow \text{Setup}(1^\lambda)$ 
5:  $(\text{pk}, \{\text{sk}_i\}_{i \in [n]}) \leftarrow \text{KGen}(n, t)$ 
6:  $\text{SIGN} := (\text{SIGN}_1, \text{SIGN}_2)$ 
7:  $(m_*, \sigma_*) \leftarrow \text{A}^{\text{SIGN}, \text{CORR}}(\text{par}, \text{pk})$ 
8: if  $m_* \in \mathcal{Q}$  then
9:   return 0
10: return Ver(pk,  $m_*$ ,  $\sigma_*$ )

```

▷ *First-round signing oracle*

Oracle SIGN₁($\mathcal{S}, m, i$):

```

11: if  $(i \notin \mathcal{S} \cap \mathcal{H}) \vee (|\mathcal{S}| < t + 1)$  then
12:   return  $\perp$ 
13:  $\text{ctr}_i := \text{ctr}_i + 1$ 
14:  $\text{sid} := \text{ctr}_i$ 
15:  $(\mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}}) := (\mathcal{S}, m)$ 

```

```

16:  $(\text{st}_{i,\text{sid}}, \text{msg}_{1,i,\text{sid}}) \leftarrow \text{Sign}_1(\text{pk}, \mathcal{S}, m, i, \text{sk}_i)$ 
17:  $\text{round}_{i,\text{sid}} := 1$ 
18:  $\mathcal{Q} := \mathcal{Q} \cup \{m_{i,\text{sid}}\}$ 
19: return  $\text{msg}_{1,i,\text{sid}}$ 

```

▷ *Second-round signing oracle*

Oracle SIGN₂($i, \text{sid}, \mathcal{M}_1$):

```

20: if  $(i \notin \mathcal{H}) \vee (\text{round}_{i,\text{sid}} \neq 1) \vee (\text{msg}_{1,i,\text{sid}} \neq \mathcal{M}_1[i])$  then
21:   return  $\perp$ 
22:  $\text{msg}_{2,i,\text{sid}} \leftarrow \text{Sign}_2(\text{st}_{i,\text{sid}}, \text{sk}_i, \mathcal{M}_1)$ 
23:  $\text{round}_{i,\text{sid}} := 2$ 
24: return  $\text{msg}_{2,i,\text{sid}}$ 

```

▷ *Corruption oracle*

Oracle CORR(i):

```

25: if  $(i \notin \mathcal{H}) \vee (|\mathcal{C}| \geq t)$  then
26:   return  $\perp$ 
27:  $\mathcal{C} := \mathcal{C} \cup \{i\}$ ,  $\mathcal{H} := \mathcal{H} \setminus \{i\}$ 
28: return  $(\text{sk}_i, (\text{st}_{i,\text{sid}})_{\text{sid} \in [\text{ctr}_i]})$ 

```

Figure 2: The TSUF security game for a (two-round) threshold signature scheme TS with adversary A.

i , a signer set $\mathcal{S}$, and a message m . This interaction is formalized via two signing oracles, SIGN₁ and SIGN₂, and we use a session identifier sid to distinguish different sessions. At the end of the game, the adversary outputs a purported forgery (m_*, σ_*) and wins if it never initiated a signing session for message m_* and if σ_* is a valid signature on m_* under public key pk . Finally, we note that there is no oracle for Comb, which computes a threshold signature from individual signature shares. This is without loss of generality, since Comb uses no secret state and can be executed publicly using the transcript produced by Sign₁ and Sign₂.

Remark 1. An interesting open problem is to achieve tight and adaptive security under a stronger notion of unforgeability that allows the adversary to initiate signing sessions for the eventual forgery message m^* .

Definition 5 (Security of Two-Round Threshold Signature Schemes). Consider the security game TSUF defined in Figure 2. A two-round threshold signature scheme TS is said to be (q_s, τ, ε) -secure if, for all $\lambda \in \mathbb{N}$, all $t < n \in \text{poly}(\lambda)$, and all adversaries A running in time at most $\tau = \tau(\lambda)$ and making at most $q_s = q_s(\lambda)$ signing queries, we have $\Pr[\text{TSUF}_{\text{TS}}^{\text{A}}(1^\lambda, n, t) = 1] \leq \varepsilon = \varepsilon(\lambda)$.

4 Our Construction

In this section, we present our construction of a two-round threshold signature scheme and provide a security proof based on the DDH assumption.

4.1 Special Commitment

Let $\text{par} := (\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda)$ be a prime order group description. The commitment scheme has key space $\mathcal{K} := \mathbb{G}^{4 \times 4}$, message space $\mathcal{M} := \mathbb{G}^3$, randomness space $\mathcal{R} := \mathbb{Z}_p^4$, and commitment space $\mathcal{U} := \mathbb{G}^4$. The sets $\mathcal{R}$ and $\mathcal{U}$ are naturally equipped with component-wise group operations, denoted $+$ and $\otimes$, respectively. We now describe the algorithms of the commitment scheme.

We do not provide a standalone formal definition or security notion for the commitment scheme. Rather, we specify the concrete commitment scheme used in our construction and prove security of the resulting threshold signature scheme directly in the following sections.

- $\text{BGen}(\text{par}) \rightarrow \text{ck}$: Sample $v, \hat{v}, w, \hat{w} \leftarrow \mathbb{G}$ and $a, b, c \leftarrow \mathbb{Z}_p$. Then, set

$$\text{ck} := \mathbf{A} := \begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{pmatrix} := \begin{pmatrix} v & v^a & v^b & v^c \\ \hat{v} & \hat{v}^a & \hat{v}^b & \hat{v}^c \\ w & w^a & w^b & w^c \\ \hat{w} & \hat{w}^a & \hat{w}^b & \hat{w}^c \end{pmatrix} \in \mathbb{G}^{4 \times 4}.$$

- $\text{TGen}(\text{par}, (X, Y)) \rightarrow (\text{ck}, \text{td})$: Sample $d_{i,j} \leftarrow \mathbb{Z}_p$ for all $(i, j) \in [4] \times [4]$. Then, set

$$\text{ck} := \mathbf{A} := \begin{pmatrix} g^{d_{1,1}} & g^{d_{1,2}} & g^{d_{1,3}} & g^{d_{1,4}} \\ X^{d_{2,1}} & X^{d_{2,2}} & X^{d_{2,3}} & X^{d_{2,4}} \\ Y^{d_{2,1}} & Y^{d_{2,2}} & Y^{d_{2,3}} & Y^{d_{2,4}} \\ X^{d_{3,1}} Y^{d_{4,1}} & X^{d_{3,2}} Y^{d_{4,2}} & X^{d_{3,3}} Y^{d_{4,3}} & X^{d_{3,4}} Y^{d_{4,4}} \end{pmatrix} \in \mathbb{G}^{4 \times 4}.$$

Further, set $\text{td} := (\mathbf{D}, (X, Y))$ with

$$\mathbf{D} := \begin{pmatrix} d_{1,1} & d_{1,2} & d_{1,3} & d_{1,4} \\ d_{2,1} & d_{2,2} & d_{2,3} & d_{2,4} \\ d_{3,1} & d_{3,2} & d_{3,3} & d_{3,4} \\ d_{4,1} & d_{4,2} & d_{4,3} & d_{4,4} \end{pmatrix} \in \mathbb{Z}_p^{4 \times 4}.$$

- $\text{Com}(\text{ck}, \mathbf{R}; \boldsymbol{\rho}) \rightarrow \text{com}$: Parse $\boldsymbol{\rho} = (\alpha, \beta, \gamma, \delta) \in \mathbb{Z}_p^4$ and $\mathbf{R} = (R_1, R_2, R_3) \in \mathbb{G}^3$. Then, compute

$$\text{com} := \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{pmatrix} := \begin{pmatrix} A_{1,1}^\alpha \cdot A_{1,2}^\beta \cdot A_{1,3}^\gamma \cdot A_{1,4}^\delta \\ A_{2,1}^\alpha \cdot A_{2,2}^\beta \cdot A_{2,3}^\gamma \cdot A_{2,4}^\delta \cdot R_1 \\ A_{3,1}^\alpha \cdot A_{3,2}^\beta \cdot A_{3,3}^\gamma \cdot A_{3,4}^\delta \cdot R_2 \\ A_{4,1}^\alpha \cdot A_{4,2}^\beta \cdot A_{4,3}^\gamma \cdot A_{4,4}^\delta \cdot R_3 \end{pmatrix} = \mathbf{A}^\boldsymbol{\rho} \cdot \begin{pmatrix} 1 \\ R_1 \\ R_2 \\ R_3 \end{pmatrix} \in \mathbb{G}^4.$$

- $\text{TCom}(\text{ck}, \text{td}, \mathbf{m}; \boldsymbol{\zeta}) \rightarrow (\text{com}, \text{st})$: Parse $\mathbf{m} = (m_1, m_2, m_3) \in \mathbb{G}^3$ and $\boldsymbol{\zeta} = (\zeta_1, \zeta_2, \zeta_3, \zeta_4) \in \mathbb{Z}_p^4$. Then, set $\text{st} := (\text{td}, \boldsymbol{\zeta}, \mathbf{m})$ and compute⁴

$$\text{com} := (C_1, C_2, C_3, C_4)^T := (g^{\zeta_1}, m_1 \cdot X^{\zeta_2}, m_2 \cdot Y^{\zeta_2}, m_3 \cdot X^{\zeta_3} Y^{\zeta_4})^T \in \mathbb{G}^4.$$

- $\text{Equiv}(\text{st}, a, b, c) \rightarrow (\boldsymbol{\rho}, \mathbf{R})$: Parse $\text{st} = (\text{td}, \boldsymbol{\zeta}, \mathbf{m})$, where $\text{td} = (\mathbf{D}, (X, Y))$. Furthermore, set $R_1 := m_1 \cdot X^{-c}$, $R_2 := m_2 \cdot Y^{-c}$, and $R_3 := m_3 \cdot X^{-ac} Y^{-bc}$, and let $\mathbf{R} := (R_1, R_2, R_3)$. If the matrix $\mathbf{D}$ is not invertible, return $\perp$. Otherwise, compute

$$\boldsymbol{\rho} := (\alpha, \beta, \gamma, \delta)^T := \mathbf{D}^{-1} \cdot (\zeta_1, \zeta_2 + c, \zeta_3 + ac, \zeta_4 + bc)^T \in \mathbb{Z}_p^4.$$

⁴By convention, all vectors are column vectors, and x^T denotes the transpose of x .

Alg ZeroShare(seed_{*i*}, $\mathcal{S}$, m , [$\mathcal{M}_1$], ℓ):

```

1: (seedi,j, seedj,i)j ∈ [n]} := seedi
2: if  $\ell = 1$  then
3:    $\Delta_{i,1} := 1_{\mathbb{G}}$ 
4:   for  $j \in \mathcal{S}$  do
5:      $G_{i,j} := H_m(i, j, \text{seed}_{i,j}, \mathcal{S}, m, 1)$ 
6:      $G_{j,i} := H_m(j, i, \text{seed}_{j,i}, \mathcal{S}, m, 1)$ 
7:      $\Delta_{i,1} := \Delta_{i,1} \cdot G_{i,j} \cdot G_{j,i}^{-1}$ 
8:   return  $\Delta_{i,1}$ 
9:  $\Delta_{i,\ell} := 0$ 
10: for  $j \in \mathcal{S}$  do
11:    $\mathbf{X}_{i,j} := H_m(i, j, \text{seed}_{i,j}, \mathcal{S}, m, \mathcal{M}_1, \ell)$ 
12:    $\mathbf{X}_{j,i} := H_m(j, i, \text{seed}_{j,i}, \mathcal{S}, m, \mathcal{M}_1, \ell)$ 
13:    $\Delta_{i,\ell} := \Delta_{i,\ell} + \mathbf{X}_{i,j} - \mathbf{X}_{j,i}$ 
14: return  $\Delta_{i,\ell}$ 

```

Figure 3: The deterministic zero-sharing procedure ZeroShare using random oracle H_m . [$\mathcal{M}_1$] denotes an optional input that is omitted when $\ell = 1$.

4.2 Threshold Signature

We give a verbal description of our two-round threshold signature scheme TZAR next and present it formally as pseudocode in Figure 4.

Setup. The global parameters of the scheme are defined as follows. Let $(\mathbb{G}, p, g)$ be a cyclic group of prime order p generated by $\text{GrGen}(1^\lambda)$. Sample three additional independent and uniformly random group elements $\hat{g}, h, \hat{h} \leftarrow \mathbb{G}$. Further, sample a uniformly random commitment key $\text{ck} \leftarrow \mathbb{G}^{4 \times 4}$. We define the public parameters as

$$\text{par} := (\mathbb{G}, p, g, \hat{g}, h, \hat{h}, \text{ck}).$$

Key Generation. The key generation algorithm $\text{KGen}(n, t)$ samples two secret scalars $x, \hat{x} \leftarrow \mathbb{Z}_p$ and defines the public key as

$$\text{pk} := (g^x \hat{g}^{\hat{x}}, h^x \hat{h}^{\hat{x}}) \in \mathbb{G}^2.$$

It generates t -out-of- n Shamir secret shares $\{x_i\}_{i \in [n]}$ of x and $\{\hat{x}_i\}_{i \in [n]}$ of $\hat{x}$. For each pair $(i, j) \in [n] \times [n]$, a random seed $\text{seed}_{i,j} \leftarrow \{0, 1\}^\lambda$ is sampled. Each signer i stores the secret key share

$$\text{sk}_i := (x_i, \hat{x}_i, \text{seed}_i),$$

where seed_i consists of all pairwise seeds $(\text{seed}_{i,j}, \text{seed}_{j,i})_{j \in [n]}$.

Deterministic Zero-Sharing. The procedure ZeroShare (see Fig. 3) deterministically derives a masking value for signer i in a given signing session from its pairwise seeds and the random oracle H_m . For each $\ell \in \{1, 2, 3\}$, the hash function

$$H_m : \{0, 1\}^* \rightarrow \begin{cases} \mathbb{G} & \text{if } \ell = 1, \\ \mathbb{Z}_p^4 & \text{if } \ell = 2, \\ \mathbb{Z}_p^2 & \text{if } \ell = 3, \end{cases}$$

is modeled as a random oracle. The parameter ℓ identifies the protocol component: $\ell = 1$ corresponds to first-round masks, $\ell = 2$ to second-round commitment randomness masks, and $\ell = 3$ to second-round response masks. We use the notation ZeroShare(seed_{*i*}, $\mathcal{S}$, m , 1) for the first-round branch and

$\text{ZeroShare}(\text{seed}_i, \mathcal{S}, m, \mathcal{M}_1, \ell)$ for $\ell \in \{2, 3\}$; hence the first-round branch has no transcript input. For any fixed session data $(\mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}}, \ell)$ for $\ell = 1$ or $(\mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}}, \mathcal{M}_1, \ell)$ for $\ell \in \{2, 3\}$, the masks computed by ZeroShare satisfy

$$\prod_{i \in \mathcal{S}_{i,\text{sid}}} \Delta_{i,1} = 1_{\mathbb{G}} \quad \text{and} \quad \sum_{i \in \mathcal{S}_{i,\text{sid}}} \Delta_{i,\ell} = \mathbf{0} \text{ for } \ell \in \{2, 3\}.$$

Signing Protocol. Suppose a set $\mathcal{S} \subseteq [n]$ of at least $t + 1$ signers want to sign a message m . We describe the protocol from the perspective of signer $i \in \mathcal{S}$ holding secret key share $\text{sk}_i = (x_i, \hat{x}_i, \text{seed}_i)$. The protocol makes use of a hash function H_u to derive pairs of group elements from the message, and a hash function H_c to derive a scalar challenge.

- *Round 1 (Commitment Phase).* The signer derives a pair of group elements $(u, \hat{u}) := H_u(m)$. It samples randomness $r_i, \hat{r}_i \leftarrow \mathbb{Z}_p$ and commitment randomness $\rho_i \leftarrow \mathbb{Z}_p^4$, and computes

$$R_{1,i} := g^{r_i} \hat{g}^{\hat{r}_i}, \quad R_{2,i} := h^{r_i} \hat{h}^{\hat{r}_i}, \quad R_{3,i} := u^{r_i} \hat{u}^{\hat{r}_i}.$$

It then commits to $\mathbf{R} := (R_{1,i}, R_{2,i}, R_{3,i}) \in \mathbb{G}^3$ using the commitment scheme under key ck , obtaining commitment $\text{com}_i := \text{com}(\text{ck}, \mathbf{R}; \rho_i)$.

Next, the signer derives a group-valued mask $\Delta_{i,1} \in \mathbb{G}$ using the deterministic zero-sharing procedure ZeroShare on inputs $(\text{seed}_i, \mathcal{S}, m, 1)$, and computes

$$U_i := u^{\lambda_{\mathcal{S},i} x_i} \hat{u}^{\lambda_{\mathcal{S},i} \hat{x}_i} \Delta_{i,1}.$$

It sends the first-round message $\text{msg}_{1,i} := (\text{com}_i, U_i)$ to all other signers in $\mathcal{S}$, and stores the local state consisting of all sampled randomness.

- *Round 2 (Response Phase).* Upon receiving all first-round messages $\{(\text{com}_j, U_j)\}_{j \in \mathcal{S}}$, the signer aggregates them into

$$\text{com} := \bigotimes_{j \in \mathcal{S}} \text{com}_j, \quad U := \prod_{j \in \mathcal{S}} U_j.$$

It derives a challenge

$$c := H_c(\text{pk}, \text{com}, U, m) \in \mathbb{Z}_p.$$

The signer then derives further masks $\Delta_{i,2} \in \mathbb{Z}_p^4$ and $(\Delta_{i,3}, \hat{\Delta}_{i,3}) \in \mathbb{Z}_p^2$ using ZeroShare on inputs $(\text{seed}_i, \mathcal{S}, m, \mathcal{M}_1, 2)$ and $(\text{seed}_i, \mathcal{S}, m, \mathcal{M}_1, 3)$, respectively, where $\mathcal{M}_1$ denotes the full list of first-round messages.

It masks its commitment randomness as $\bar{\rho}_i := \rho_i + \Delta_{i,2}$, and computes response values

$$z_i := r_i + c \lambda_{\mathcal{S},i} x_i + \Delta_{i,3}, \quad \hat{z}_i := \hat{r}_i + c \lambda_{\mathcal{S},i} \hat{x}_i + \hat{\Delta}_{i,3}.$$

It sends the second-round message $\text{msg}_{2,i} := (\bar{\rho}_i, z_i, \hat{z}_i)$ to all signers in $\mathcal{S}$.

- *Combine Phase.* Given all first-round messages $\{(\text{com}_j, U_j)\}_{j \in \mathcal{S}}$ and all second-round messages $\{(\bar{\rho}_j, z_j, \hat{z}_j)\}_{j \in \mathcal{S}}$, the signer aggregates

$$\text{com} := \bigotimes_{j \in \mathcal{S}} \text{com}_j, \quad U := \prod_{j \in \mathcal{S}} U_j, \quad \rho := \sum_{j \in \mathcal{S}} \bar{\rho}_j, \quad z := \sum_{j \in \mathcal{S}} z_j, \quad \hat{z} := \sum_{j \in \mathcal{S}} \hat{z}_j.$$

It recomputes $c = H_c(\text{pk}, \text{com}, U, m)$ and outputs the final signature

$$\sigma := (U, z, \hat{z}, c, \rho).$$

Verification. To verify a signature $\sigma = (U, z, \hat{z}, c, \rho)$ on message m under public key $\text{pk} = (\text{pk}_1, \text{pk}_2)$, the verifier derives $(u, \hat{u}) := H_u(m)$, and reconstructs

$$R_1 := g^z \hat{g}^{\hat{z}} \cdot \text{pk}_1^{-c}, \quad R_2 := h^z \hat{h}^{\hat{z}} \cdot \text{pk}_2^{-c}, \quad R_3 := u^z \hat{u}^{\hat{z}} \cdot U^{-c}.$$

It recomputes the commitment $\text{com} := \text{Com}(\text{ck}, \mathbf{R}; \rho)$, and accepts if and only if $c = H_c(\text{pk}, \text{com}, U, m)$.

<u>Setup(1^λ):</u> 1: $(\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda)$ 2: $\hat{g}, h, \hat{h} \leftarrow \mathbb{G}$ 3: $\text{ck} \leftarrow \mathbb{G}^{4 \times 4}$ 4: return $\text{par} := (\mathbb{G}, p, g, \hat{g}, h, \hat{h}, \text{ck})$ $\triangleright$ All algorithms take par as input $\triangleright$ Key generation <u>KGen(n, t):</u> 5: $x, \hat{x} \leftarrow \mathbb{Z}_p$ 6: $\text{pk} := (g^x \hat{g}^{\hat{x}}, h^x \hat{h}^{\hat{x}})$ 7: for $i, j \in [n]$ do 8: $\text{seed}_{i,j} \leftarrow \{0, 1\}^\lambda$ 9: $(x_1, \dots, x_n) \leftarrow \text{Shamir}(n, t, x)$ 10: $(\hat{x}_1, \dots, \hat{x}_n) \leftarrow \text{Shamir}(n, t, \hat{x})$ 11: for $i \in [n]$ do 12: $\text{seed}_i := (\text{seed}_{i,j}, \text{seed}_{j,i})_{j \in [n]}$ 13: $\text{sk}_i := (x_i, \hat{x}_i, \text{seed}_i)$ 14: return $(\text{pk}, \{\text{sk}_i\}_{i \in [n]})$ $\triangleright$ First signing round <u>Sign₁($\text{pk}, \mathcal{S}, m, i, \text{sk}_i$):</u> 15: $(x_i, \hat{x}_i, \text{seed}_i) := \text{sk}_i$ 16: $(u, \hat{u}) := H_u(m)$ 17: $r_i, \hat{r}_i \leftarrow \mathbb{Z}_p, \rho_i \leftarrow \mathbb{Z}_p^4$ 18: $R_{1,i} := g^{r_i} \hat{g}^{\hat{r}_i}, R_{2,i} := h^{r_i} \hat{h}^{\hat{r}_i}$ 19: $R_{3,i} := u^{r_i} \hat{u}^{\hat{r}_i}$ 20: $\mathbf{R} := (R_1, R_2, R_3)$ 21: $\text{com}_i := \text{Com}(\text{ck}, \mathbf{R}; \rho_i)$ 22: $\Delta_{i,1} := \text{ZeroShare}(\text{seed}_i, \mathcal{S}, m, 1)$ 23: $U_i := u^{\lambda_{\mathcal{S},i} x_i} \hat{u}^{\lambda_{\mathcal{S},i} \hat{x}_i} \Delta_{i,1}$ 24: $\text{st}_i := (\text{pk}, \mathcal{S}, m, i, r_i, \hat{r}_i, \rho_i)$ 25: return (com_i, U_i) $\triangleright$ Second signing round <u>Sign₂($\text{st}_i, \text{sk}_i, \mathcal{M}_1$):</u>	26: $(x_i, \hat{x}_i, \text{seed}_i) := \text{sk}_i$ 27: $(\text{pk}, \mathcal{S}, m, i, r_i, \hat{r}_i, \rho_i) := \text{st}_i$ 28: $(\text{com}_j, U_j)_{j \in \mathcal{S}} := \mathcal{M}_1$ 29: $\text{com} := \bigotimes_{j \in \mathcal{S}} \text{com}_j$ 30: $U := \prod_{j \in \mathcal{S}} U_j$ 31: $c := H_c(\text{pk}, \text{com}, U, m)$ 32: $\Delta_{i,2} := \text{ZeroShare}(\text{seed}_i, \mathcal{S}, m, \mathcal{M}_1, 2)$ 33: $(\Delta_{i,3}, \hat{\Delta}_{i,3}) :=$ $\text{ZeroShare}(\text{seed}_i, \mathcal{S}, m, \mathcal{M}_1, 3)$ 34: $\bar{\rho}_i := \rho_i + \Delta_{i,2}$ 35: $z_i := r_i + c \lambda_{\mathcal{S},i} x_i + \Delta_{i,3}$ 36: $\hat{z}_i := \hat{r}_i + c \lambda_{\mathcal{S},i} \hat{x}_i + \hat{\Delta}_{i,3}$ 37: return $(\bar{\rho}_i, z_i, \hat{z}_i)$ $\triangleright$ Combine algorithm <u>Comb($\text{pk}, \mathcal{S}, m, \mathcal{M}_1, \mathcal{M}_2$):</u> 38: $(\text{com}_j, U_j)_{j \in \mathcal{S}} := \mathcal{M}_1$ 39: $(\bar{\rho}_j, z_j, \hat{z}_j)_{j \in \mathcal{S}} := \mathcal{M}_2$ 40: $\text{com} := \bigotimes_{j \in \mathcal{S}} \text{com}_j$ 41: $U := \prod_{j \in \mathcal{S}} U_j$ 42: $\rho := \sum_{j \in \mathcal{S}} \bar{\rho}_j$ 43: $z := \sum_{j \in \mathcal{S}} z_j$ 44: $\hat{z} := \sum_{j \in \mathcal{S}} \hat{z}_j$ 45: $c := H_c(\text{pk}, \text{com}, U, m)$ 46: return $\sigma := (U, z, \hat{z}, c, \rho)$ $\triangleright$ Verification algorithm <u>Ver(pk, m, σ):</u> 47: $(\text{pk}_1, \text{pk}_2) := \text{pk}$ 48: $(U, z, \hat{z}, c, \rho) := \sigma$ 49: $(u, \hat{u}) := H_u(m)$ 50: $R_1 := g^z \hat{g}^{\hat{z}} \cdot \text{pk}_1^{-c}$ 51: $R_2 := h^z \hat{h}^{\hat{z}} \cdot \text{pk}_2^{-c}$ 52: $R_3 := u^z \hat{u}^{\hat{z}} \cdot U^{-c}$ 53: $\mathbf{R} := (R_1, R_2, R_3)$ 54: $\text{com} := \text{Com}(\text{ck}, \mathbf{R}; \rho)$ 55: return $[c = H_c(\text{pk}, \text{com}, U, m)]$
--	---

Figure 4: Our two-round threshold signature scheme TZAR.

4.3 Security Analysis

We prove tight security of our threshold signature scheme TZAR under the DDH assumption. The correctness of our scheme follows by inspection.

Theorem 1. *If the DDH problem is $(\tau_{\text{ddh}}, \varepsilon_{\text{ddh}})$ -hard for the group generation algorithm GrGen, then TZAR is a $(q_s, \tau_{\text{uf}}, \varepsilon_{\text{uf}})$ -secure threshold signature scheme in the random oracle model (ROM) against any adversary that makes at most q_h hash queries, where $\tau_{\text{ddh}} \approx \tau_{\text{uf}}$ and*

$$\varepsilon_{\text{uf}} \leq 3\varepsilon_{\text{ddh}} + 2\varepsilon_{\text{uddh4}} + \varepsilon_{\text{mddh4}} + \frac{q_h + 17 + q_s^2}{p} + \frac{3}{p^2} + \frac{2}{p^4} + \frac{n^2 q_h}{2^\lambda}.$$

Game 1MoreImg^A(1^λ): 1: $(\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda)$ 2: $\hat{g}, h, \hat{h} \leftarrow \mathbb{G}$ 3: $\text{par} := (\mathbb{G}, p, g, \hat{g}, h, \hat{h})$ $\triangleright$ All oracles take par implicitly as input 4: $X, Y \leftarrow \mathbb{G}$ 5: $\text{ctr} := 0, \mathcal{Q} := \emptyset$ 6: $(i, U) \leftarrow \text{A}^{\text{CHAL}, \text{PREIM}}(\text{par}, (X, Y))$ 7: if $(i \notin [\text{ctr}]) \vee (i \in \mathcal{Q})$ then 8: return 0 9: if $U \neq X^{a_i} Y^{b_i}$ then 10: return 0 11: return 1	$\triangleright$ Challenge oracle Oracle CHAL(): 12: $\text{ctr} := \text{ctr} + 1$ 13: $\text{sid} := \text{ctr}$ 14: $a_{\text{sid}}, b_{\text{sid}} \leftarrow \mathbb{Z}_p$ 15: $u_{\text{sid}} := g^{a_{\text{sid}}} h^{b_{\text{sid}}}$ 16: $\hat{u}_{\text{sid}} := \hat{g}^{a_{\text{sid}}} \hat{h}^{b_{\text{sid}}}$ 17: return $(u_{\text{sid}}, \hat{u}_{\text{sid}})$ $\triangleright$ Preimage oracle Oracle PREIM(sid): 18: if $\text{sid} \notin [\text{ctr}]$ then 19: return $\perp$ 20: $\mathcal{Q} := \mathcal{Q} \cup \{\text{sid}\}$ 21: return $(a_{\text{sid}}, b_{\text{sid}})$
---	--

Figure 5: The one-more image game with adversary A.

We prove the theorem via the following sequence of lemmas.

Lemma 2 (Effect of Masking). *The original security game can be simulated by a reduction that plays as an adversary in the following ideal game⁵:*

- *Setup:* Sample parameters $\text{par} := (\mathbb{G}, p, g, \hat{g}, h, \hat{h}, \text{ck})$ as in the real game. Sample $x, \hat{x} \leftarrow \mathbb{Z}_p$, and set $\text{pk} := (g^x \hat{g}^{\hat{x}}, h^x \hat{h}^{\hat{x}})$ as before. Note that there are no secret key shares $\{(x_i, \hat{x}_i)\}_{i \in [n]}$ and seeds $\{\text{seed}_i\}_{i \in [n]}$ anymore.
- $\text{SIGN}_1(\mathcal{S}, m, i)$: Compute com_i as in Sign_1 , using sampled randomness $r_i, \hat{r}_i$ and ρ_i . If this is the first query on message m for any signer in $\mathcal{H}$, also return $U := u^x \hat{u}^{\hat{x}}$.
- $\text{SIGN}_2(i, \text{sid}, \mathcal{M}_1)$: If $i \in \mathcal{H}$ is the last honest signer on which SIGN_2 is queried for the tuple $(\mathcal{S}_{i, \text{sid}}, m_{i, \text{sid}}, \mathcal{M}_1)$, define $H := \mathcal{H} \cap \mathcal{S}_{i, \text{sid}}$, and return

$$z := \sum_{j \in H} r_j + cx, \quad \hat{z} := \sum_{j \in H} \hat{r}_j + c\hat{x}, \quad \rho := \sum_{j \in H} \rho_j.$$

- $\text{CORR}(i)$: If $i \in \mathcal{H}$, return $(r_{i, \text{sid}}, \hat{r}_{i, \text{sid}}, \rho_{i, \text{sid}})$ for every session $\text{sid} \in [\text{ctr}_i]$.
- *Winning condition:* As in the real game, the adversary wins if and only if it outputs a valid forgery (m_*, σ_*) .

Concretely, for any group generation algorithm GrGen , if TZAR is $(q_s, \tau_{\text{ideal}}, \varepsilon_{\text{ideal}})$ -secure against an adversary that makes at most q_h random oracle queries, then TZAR is $(q_s, \tau_{\text{uf}}, \varepsilon_{\text{uf}})$ -secure against any adversary that makes at most q_h random oracle queries, where $\tau_{\text{uf}} \approx \tau_{\text{ideal}}$ and

$$\varepsilon_{\text{uf}} \leq \varepsilon_{\text{ideal}} + \frac{n^2 \cdot q_h}{2^\lambda} + \frac{q_s^2}{p}.$$

Lemma 3. *The above ideal game can be simulated by a reduction who plays as an adversary in the following one-more image game⁶:*

- *Setup:* Sample parameters $\text{par} := (\mathbb{G}, p, g, \hat{g}, h, \hat{h})$, where $\hat{g}, h, \hat{h} \leftarrow \mathbb{Z}_p$. Further, sample group elements $X, Y \leftarrow \mathbb{G}$ uniformly at random. Initialize a counter $\text{ctr} := 0$ and a set $\mathcal{Q} := \emptyset$.

⁵See Fig. 6 for a formal definition in pseudocode.

⁶See Fig. 5 for a formal definition in pseudocode.

- **CHAL()**: Update $\text{ctr} := \text{ctr} + 1$, and let $\text{sid} := \text{ctr}$. Sample $a_{\text{sid}}, b_{\text{sid}} \leftarrow \mathbb{Z}_p$ uniformly and independently, and let

$$(u_{\text{sid}}, \hat{u}_{\text{sid}}) := (g^{a_{\text{sid}}} h^{b_{\text{sid}}}, \hat{g}^{a_{\text{sid}}} \hat{h}^{b_{\text{sid}}}).$$

Return $(u_{\text{sid}}, \hat{u}_{\text{sid}})$.

- **PREIM(sid)**: Require $\text{sid} \in [\text{ctr}]$. Update $\mathcal{Q} := \mathcal{Q} \cup \{\text{sid}\}$, and return $(a_{\text{sid}}, b_{\text{sid}})$.
- **Winning condition**: Let $(i, U) \leftarrow \mathbf{A}^{\text{CHAL}, \text{PREIM}}(\text{par}, (X, Y))$ be the adversary's output. Check that $i \in [\text{ctr}]$, $i \notin \mathcal{Q}$, and $U = X^{a_i} Y^{b_i}$; otherwise, return 0.

Concretely, for any group generation algorithm **GrGen**, if the one-more image game is $(\tau_{\text{one-more}}, \varepsilon_{\text{one-more}})$ -secure against an adversary that makes at most q_h challenge oracle queries, then **TZAR** is $(q_s, \tau_{\text{ideal}}, \varepsilon_{\text{ideal}})$ -secure against any adversary that makes at most q_h random oracle queries, where $\tau_{\text{ideal}} \approx \tau_{\text{one-more}}$ and

$$\varepsilon_{\text{ideal}} \leq \varepsilon_{\text{one-more}} + 2\varepsilon_{\text{ddh}} + 2\varepsilon_{\text{uddh4}} + \varepsilon_{\text{mddh4}} + \frac{q_h + 15}{p} + \frac{3}{p^2} + \frac{2}{p^4}.$$

Lemma 4. If the DDH problem is $(\tau_{\text{ddh}}, \varepsilon_{\text{ddh}})$ -hard for the group generation algorithm **GrGen**, then the one-more image game is $(\tau_{\text{one-more}}, \varepsilon_{\text{one-more}})$ -hard, where $\tau_{\text{ddh}} \approx \tau_{\text{uf}}$ and

$$\varepsilon_{\text{one-more}} \leq \varepsilon_{\text{ddh}} + \frac{2}{p}.$$

Proof of Lemma 2. Since the proof follows from standard masking arguments that have appeared in several prior works [BCL⁺25, Che25b], we defer it to Section B. We note that [BCL⁺25] also defines a conceptually similar ideal game (and a corresponding lemma). $\square$

Proof of Lemma 3. We prove Lemma 3 via a sequence of games.

Game $\mathbf{G}_0$ (Ideal Game): This is the ideal security game as described above. We briefly recall the game to fix some notation. First, the game samples public parameters $\text{par} := (\mathbb{G}, p, g, \hat{g}, h, \hat{h}, \text{ck}) \leftarrow \text{Setup}(1^\lambda)$, where $(\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda)$, and $\hat{g}, h, \hat{h} \leftarrow \mathbb{G}$ and $\text{ck} \leftarrow \mathbb{G}^{4 \times 4}$ are sampled uniformly at random. The game also samples $x, \hat{x} \leftarrow \mathbb{Z}_p$ for a group secret key $\text{sk} := (x, \hat{x}) \in \mathbb{Z}_p^2$ and public key $\text{pk} := (X, Y) := (g^x \hat{g}^{\hat{x}}, h^x \hat{h}^{\hat{x}}) \in \mathbb{G}^2$. In particular, there are no secret key shares and seeds. Then, the game runs the adversary **A** on input (par, pk) with access to signing oracles, random oracles, and a corruption oracle defined as follows.

- **Signing Oracle SIGN_1** : Whenever **A** calls the oracle $\text{SIGN}_1(\mathcal{S}, m, i)$ to start a new signing session⁷ with user i for message m and signer set $\mathcal{S}$, the game adds m to a list $\mathcal{Q}$ and keeps track of the session using a signer-specific identifier (i, sid) , where sid starts from 1. Further, the game returns com_i honestly as computed in Sign_1 , using sampled nonce randomness $r_i, \hat{r}_i \leftarrow \mathbb{Z}_p$ and commitment randomness $\rho_i \leftarrow \mathbb{Z}_p^4$. If m is queried for the first time (for any honest user), the game also returns $U := u^x \hat{u}^{\hat{x}}$, where $(u, \hat{u}) := \text{H}_u(m)$.
- **Signing Oracle SIGN_2** : Whenever **A** calls the oracle $\text{SIGN}_2(i, \text{sid}, \mathcal{M}_1)$ to finish a signing session⁸ with user i for message $m_{i, \text{sid}}$; signer set $\mathcal{S}_{i, \text{sid}}$, and first-round message list $\mathcal{M}_1$, the game checks whether i is the last honest user for which SIGN_2 is queried under $(\mathcal{S}_{i, \text{sid}}, m_{i, \text{sid}}, \mathcal{M}_1)$. In that case, the game computes and returns $z := \sum_{j \in \mathcal{H}} r_j + cx$, $\hat{z} := \sum_{j \in \mathcal{H}} \hat{r}_j + c\hat{x}$, and $\rho := \sum_{j \in \mathcal{H}} \rho_j$. Here, $r_j, \hat{r}_j, \rho_j$ are the sampled randomness values from party j in the first signing round under $(\mathcal{S}_{i, \text{sid}}, m_{i, \text{sid}})$. (If i is the last honest user for which SIGN_2 is queried under $(\mathcal{S}_{i, \text{sid}}, m_{i, \text{sid}}, \mathcal{M}_1)$, then by definition all other honest users have already completed the first signing round under $(\mathcal{S}_{i, \text{sid}}, m_{i, \text{sid}})$, and the respective randomness values are already defined.)

⁷W.l.o.g., we assume that all first-round signing queries are valid in the sense that $i \in \mathcal{S} \cap \mathcal{H}$ and $|\mathcal{S}| \geq t + 1$.

⁸W.l.o.g., we assume that all second-round signing queries are valid for the ideal oracle: $i \in \mathcal{H}$, $\text{round}_{i, \text{sid}} = 1$, and the commitment component of $\mathcal{M}_1[i]$ equals the stored $\text{com}_{i, \text{sid}}$. The real-game wrapper additionally checks equality of the entire first-round message.

Further, c is the hash value computed as $H_c(\text{pk}, \text{com}, U, m_{i,\text{sid}})$ from the first-round messages $\mathcal{M}_1 =: (\text{com}_j, U_j)_{j \in \mathcal{S}_{i,\text{sid}}}, \text{pk}$, and $m_{i,\text{sid}}$.

- *Random Oracles:* The random oracles H_u and H_c are simulated honestly via standard lazy sampling. Concretely, for each new query $H_u(m)$, the game samples $u, \hat{u} \leftarrow \mathbb{G}$ uniformly at random and sets $(u, \hat{u}) := H_u(m)$. Likewise, for each new query $H_c(\text{pk}, \text{com}, U, m)$, the game samples $c \leftarrow \mathbb{Z}_p$ uniformly at random and sets $c := H_c(\text{pk}, \text{com}, U, m)$.
- *Corruption Oracle:* Whenever $\mathbf{A}$ calls the oracle $\text{CORR}(i)$ to corrupt a signer $i \in \mathcal{H}$, the game returns the randomness values $(r_{i,\text{sid}}, \hat{r}_{i,\text{sid}}, \rho_{i,\text{sid}})$, computed in the first round, for all existing sessions $\text{sid} \in [\text{ctr}_i]$ of that signer.

At the end of the game, the adversary outputs a forgery (m_*, σ_*) , and the game outputs 1 if and only if: the message m_* was never queried before, i.e., $m_* \notin \mathcal{Q}$, and the signature σ_* verifies for message m_* under public key pk i.e., $\text{Ver}(\text{pk}, m_*, \sigma_*) = 1$. From now on, we write $\sigma_* := (U_*, z_*, \hat{z}_*, c_*, \rho_*)$. By definition, we have

$$\Pr[\mathbf{G}_0 \Rightarrow 1] = \varepsilon_{\text{ideal}}.$$

Game $\mathbf{G}_1$ (Program H_u with Trapdoor): In this game, we change how we program the random oracle H_u that outputs the base elements $(u, \hat{u})$ for the computation of U . Namely, for each new query $H_u(m)$, we sample $a_m, b_m \leftarrow \mathbb{Z}_p$ uniformly and independently, and then program the random oracle as

$$H_u(m) := (u_m, \hat{u}_m) := (g^{a_m} h^{b_m}, \hat{g}^{a_m} \hat{h}^{b_m}).$$

It is clear that this does not change the distribution of random oracle H_u outputs, unless $(g, h, \hat{g}, \hat{h})$ is a DH tuple. But this happens only with probability $1/p$, as the elements $\hat{g}, h, \hat{h} \in \mathbb{G}$ are sampled uniformly at random at the beginning of the game. Therefore, the adversary's view is statistically close to before, and we get

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| \leq 1/p.$$

Game $\mathbf{G}_2$ (Compute U via Trapdoor): In this game, we change how we compute $U = u^x \hat{u}^{\hat{x}}$ in $\text{SIGN}_1(\cdot, m, i)$ when m is queried the first time (for any $i \in \mathcal{H}$). Recall that for a query $\text{SIGN}_1(\mathcal{S}, m, i)$ with $i \in \mathcal{H}$ and $|\mathcal{S}| \geq t + 1$, where m has not appeared in any previous (valid) SIGN_1 query, the game computes and returns $U := u^x \hat{u}^{\hat{x}}$ using $x, \hat{x}$. In this game, we instead compute this value as $U := U_m := X^{a_m} Y^{b_m}$, where $(a_m, b_m) \in \mathbb{Z}_p^2$ are the values introduced in the previous game (i.e., $H_u(m) := (u_m, \hat{u}_m) := (g^{a_m} h^{b_m}, \hat{g}^{a_m} \hat{h}^{b_m})$ and $\text{pk} = (X, Y)$). For this to be well defined, we may assume without loss of generality that the adversary always queries the hash value $H_u(m)$ prior to issuing any signing request on m (for any honest user). We claim that this change does not affect the adversary's view. But this is clear from the identity

$$X^{a_m} Y^{b_m} = (g^x \hat{g}^{\hat{x}})^{a_m} (h^x \hat{h}^{\hat{x}})^{b_m} = (g^{a_m} h^{b_m})^x (\hat{g}^{a_m} \hat{h}^{b_m})^{\hat{x}} = u_m^x \hat{u}_m^{\hat{x}},$$

which is the correct value. Thus, we get

$$\Pr[\mathbf{G}_2 \Rightarrow 1] = \Pr[\mathbf{G}_1 \Rightarrow 1].$$

Note that the changes so far have the effect that we do not use the signing keys $x, \hat{x}$ in the first round anymore. In particular, only the randomness values $r_i, \hat{r}_i, \rho_i$ are used in the first round.

Game $\mathbf{G}_3$ (Switch ck to Binding): In this game, we change how we generate the commitment key ck . Recall that in the previous game, the commitment key was generated by sampling it uniformly at random from $\mathbb{G}^{4 \times 4}$. In this game, we generate the commitment key in binding mode as $\text{ck} \leftarrow \text{BGen}(\mathbb{G}, p, g)$. We can bound the distinguishing advantage between games $\mathbf{G}_3$ and $\mathbf{G}_2$ via a reduction $\mathbf{B}$ to uDDH4 . To this end, $\mathbf{B}$ gets a tuple $(h_{i,j})_{i,j \in [4]}$ of elements from $\mathbb{G}$ as input and then simulates game $\mathbf{G}_2$ for $\mathbf{A}$ with the following changes. It defines the commitment key $\text{ck} := \mathbf{A} \in \mathbb{G}^{4 \times 4}$ as follows:

- Let $\mathbf{H} \in \mathbb{Z}_p^{4 \times 4}$ be the matrix of scalars of the $(h_{i,j})_{i,j \in [4]}$ to base g . Further, let $\mathbf{H}_0 \in \mathbb{Z}_p^{4 \times 1}$ be its first column and let $\mathbf{H}_1 \in \mathbb{Z}_p^{4 \times 3}$ be the remaining submatrix, i.e., $\mathbf{H} = [\mathbf{H}_0 | \mathbf{H}_1]$.
- It samples $\mathbf{T} \leftarrow \mathbb{Z}_p^{4 \times 4}$ and $\mathbf{S} \leftarrow \mathbb{Z}_p^{1 \times 3}$. Now define $\mathbf{D} := \mathbf{T}\mathbf{H} + [0 | \mathbf{T}\mathbf{H}_0\mathbf{S}] \in \mathbb{Z}_p^{4 \times 4}$.
- Then, it computes the commitment key as $\mathbf{A} := (g)^{\mathbf{D}} \in \mathbb{G}^{4 \times 4}$ (component-wise). It is not hard to see that $\mathbf{B}$ can efficiently compute $\mathbf{A}$, given $\mathbf{T}$, $\mathbf{S}$, and $(h_{i,j})_{i,j \in [4]}$.

In the end, $\mathbf{B}$ outputs whatever the game outputs. We analyze $\mathbf{B}$'s success probability. We claim that if there are $a, b, c \in \mathbb{Z}_p$ such that $h_{i,2} = h_{i,1}^a$, $h_{i,3} = h_{i,1}^b$, and $h_{i,4} = h_{i,1}^c$ for all $i \in [4]$, then $\mathbf{B}$ provides a simulation statistically close to game $\mathbf{G}_3$. On the other hand, if the $(h_{i,j})_{i,j}$ are uniform and independent, then $\mathbf{B}$ provides a simulation statistically close to game $\mathbf{G}_2$. For the first claim, observe that $\mathbf{H}_1$ can then be written as $\mathbf{H}_1 = \mathbf{H}_0\mathbf{R}$ where $\mathbf{R} = [a, b, c] \in \mathbb{Z}_p^{1 \times 3}$. With that, we get the identity

$$\mathbf{D} = \mathbf{T}\mathbf{H} + [0 | \mathbf{T}\mathbf{H}_0\mathbf{S}] = [\mathbf{T}\mathbf{H}_0 | \mathbf{T}\mathbf{H}_0(\mathbf{R} + \mathbf{S})].$$

Assuming $\mathbf{H}_0 \in \mathbb{Z}_p^{4 \times 1}$ has full rank (i.e., at least one of $h_{1,1}, h_{2,1}, h_{3,1}, h_{4,1}$ is different from the neutral element $1 \in \mathbb{G}$), which happens with probability at least $1 - 1/p^4$, we see that the commitment key $\mathbf{A}$ is distributed exactly as a (binding-mode) commitment key in game $\mathbf{G}_3$. For the second claim, observe that a uniform $\mathbf{H} \in \mathbb{Z}_p^{4 \times 4}$ is invertible with probability at least $1 - 4/p$. Then, assuming $\mathbf{H}$ to be invertible, the matrix $\mathbf{T}\mathbf{H} \in \mathbb{Z}_p^{4 \times 4}$ is also uniform and $\mathbf{D}$ thus also. From this, we can see that the commitment key $\mathbf{A}$ is uniform and independent and thus distributed exactly as a (uniform) commitment key in $\mathbf{G}_2$. Overall, we get

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \varepsilon_{\text{uddh4}} + 4/p + 1/p^4.$$

Game $\mathbf{G}_4$ (Change Winning Condition): In this game, we change the winning condition. Recall that in the previous game, the winning condition was performed by checking that $m_* \notin \mathcal{Q}$ and $\text{Ver}(\text{pk}, m_*, \sigma_*) = 1$, where $\sigma_* = (U_*, z_*, \hat{z}_*, c_*, \rho_*)$. From now on, we replace the signature verification with checking whether $U_* = X^{a_*} Y^{b_*}$, where $(a_*, b_*) \in \mathbb{Z}_p^2$ is the trapdoor for $H_u(m_*)$ introduced in game $\mathbf{G}_2$, i.e., $H_u(m_*) = (g^{a_*} h^{b_*}, \hat{g}^{a_*} \hat{h}^{b_*})$. We claim that, except with probability q_h/p , the winning condition is equivalent to the winning condition in the previous game. For this, consider any hash query $H_c(\text{pk}, \text{com}, U, m)$. Since the commitment key ck is in binding mode due to the previous change, it can only be opened to a unique value $\mathbf{R}$. Now assume that $U \neq X^a Y^b$, where $(a, b) \in \mathbb{Z}_p^2$ is the trapdoor for $H_u(m) = (u, \hat{u})$.⁹ Then, we claim there exists only one possible answer $c \in \mathbb{Z}_p$ to this hash query such that

$$(R_1, R_2, R_3) \cdot (\text{pk}_1, \text{pk}_2, U)^c \in \text{Span}\{(g, h, u), (\hat{g}, \hat{h}, \hat{u})\}_{\mathbb{Z}_p},$$

i.e., there exists $(z, \hat{z}) \in \mathbb{Z}_p^2$ such that $(g, h, u)^z (\hat{g}, \hat{h}, \hat{u})^{\hat{z}} = (R_1, R_2, R_3) \cdot (\text{pk}_1, \text{pk}_2, U)^c$ (i.e., verification succeeds) with component-wise multiplication. Suppose, towards a contradiction, that there exist two distinct values $c \neq c' \in \mathbb{Z}_p$ and $(z, \hat{z}), (z', \hat{z}') \in \mathbb{Z}_p^2$ such that

$$\begin{aligned} (g, h, u)^z (\hat{g}, \hat{h}, \hat{u})^{\hat{z}} &= (R_1, R_2, R_3) \cdot (\text{pk}_1, \text{pk}_2, U)^c, \\ (g, h, u)^{z'} (\hat{g}, \hat{h}, \hat{u})^{\hat{z}'} &= (R_1, R_2, R_3) \cdot (\text{pk}_1, \text{pk}_2, U)^{c'}. \end{aligned}$$

Dividing the two equations yields

$$(g, h, u)^{z-z'} (\hat{g}, \hat{h}, \hat{u})^{\hat{z}-\hat{z}'} = (\text{pk}_1, \text{pk}_2, U)^{c-c'}.$$

Since $c - c' \neq 0$, we find that $(\text{pk}_1, \text{pk}_2, U) \in \text{Span}\{(g, h, u), (\hat{g}, \hat{h}, \hat{u})\}_{\mathbb{Z}_p}$. Thus, there exist $a, b \in \mathbb{Z}_p$ such that

$$(\text{pk}_1, \text{pk}_2, U) = (g, h, u)^a (\hat{g}, \hat{h}, \hat{u})^b.$$

⁹As before, we may assume without loss of generality that the adversary always queries the hash value $H_u(m)$ prior to $H_c(\cdot, m)$. Otherwise, the game itself can just make an internal query to $H_u(m)$.

Comparing the first two coordinates with $(\mathbf{pk}_1, \mathbf{pk}_2) = (g, h)^x(\hat{g}, \hat{h})^{\hat{x}}$ yields $a = x$ and $b = \hat{x}$, and thus $U = u^x \hat{u}^{\hat{x}} = X^a Y^b$ (see game $\mathbf{G}_2$ for the latter identity). This contradicts our initial assumption that $U \neq X^a Y^b$. Thus, for fixed $(\mathbf{pk}, \text{com}, U, m)$ with $U \neq X^a Y^b$, there exists at most one value c for which the verification relation can hold. Since the random oracle $\mathbf{H}_c$ samples c uniformly from $\mathbb{Z}_p$, the probability that this unique value is chosen is at most $1/p$. Thus, by a union bound over all random oracle queries, we get

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq q_h/p.$$

Game $\mathbf{G}_5$ (Switch $(g, \hat{g}, h, \hat{h})$ to DH): In this game, we change the distribution of the public parameters $(g, \hat{g}, h, \hat{h})$. Instead of sampling $\hat{g}, h, \hat{h} \leftarrow \mathbb{G}$ uniformly at random (as in the previous games), we sample $\alpha \leftarrow \mathbb{Z}_p$ and set $\hat{g} := g^\alpha$ and $\hat{h} := h^\alpha$, so that $(g, \hat{g}, h, \hat{h}) = (g, g^\alpha, h, h^\alpha)$ forms a DH tuple. We can bound the distinguishing advantage between games $\mathbf{G}_5$ and $\mathbf{G}_4$ via a straightforward reduction $\mathbf{B}$ to DDH. To this end, $\mathbf{B}$ gets a tuple $\mathcal{D} := (g, \hat{g}, h, T) \in \mathbb{G}^4$ as input, where T is either h^α (the DH case) or a uniformly random group element (the random case). It then simulates game $\mathbf{G}_4$ for $\mathbf{A}$ as follows: it sets the public parameters to $(g, \hat{g}, h, \hat{h})$ with $\hat{h} := T$, and samples all remaining public parameters exactly as in game $\mathbf{G}_4$. It answers all oracle queries exactly as in game $\mathbf{G}_4$. In the end, it outputs whatever the game outputs. If $T = h^\alpha$, then the simulated game is distributed identically to game $\mathbf{G}_5$. If T is uniform, then the simulated game is distributed identically to game $\mathbf{G}_4$. Therefore, we obtain

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq \varepsilon_{\text{ddh}}.$$

Game $\mathbf{G}_6$ (Switch ck Back to Uniform): In this game, we change how we generate the commitment key ck . Recall that in the previous game, the commitment key was generated in binding mode as $\text{ck} \leftarrow \text{BGen}(\mathbb{G}, p, g)$. In this game, we generate the commitment key by sampling it uniformly at random as $\text{ck} \leftarrow \mathbb{G}^{4 \times 4}$. As before, we can bound the distinguishing advantage between games $\mathbf{G}_6$ and $\mathbf{G}_5$ via a reduction to uDDH4. The reduction is exactly the same used in game $\mathbf{G}_3$ (switching from uniform to binding key), and we therefore omit the details here. We get

$$|\Pr[\mathbf{G}_6 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq \varepsilon_{\text{uddh4}} + 4/p + 1/p^4.$$

Game $\mathbf{G}_7$ (Switch ck to Rank-3): In this game, we change how we generate the commitment key ck . Instead of sampling it uniformly at random from $\mathbb{G}^{4 \times 4}$, we generate it as follows. Sample $d_{i,j} \leftarrow \mathbb{Z}_p$ for all $(i, j) \in [3] \times 3$ uniformly at random. Then set

$$\text{ck} := \mathbf{A} := \begin{pmatrix} g^{d_{1,1}} & g^{d_{1,2}} & g^{d_{1,3}} & g^{d_{1,4}} \\ g^{d_{2,1}} & g^{d_{2,2}} & g^{d_{2,3}} & g^{d_{2,4}} \\ h^{d_{2,1}} & h^{d_{2,2}} & h^{d_{2,3}} & h^{d_{2,4}} \\ g^{d_{3,1}} & g^{d_{3,2}} & g^{d_{3,3}} & g^{d_{3,4}} \end{pmatrix} \in \mathbb{G}^{4 \times 4}.$$

We can bound the distinguishing advantage between games $\mathbf{G}_7$ and $\mathbf{G}_6$ via a reduction $\mathbf{B}$ to mDDH4. To this end, $\mathbf{B}$ gets a tuple $\mathcal{D} := (g, h, (g_i, h_i)_{i \in [4]})$ of elements from $\mathbb{G}$ as input and then simulates game $\mathbf{G}_6$ for $\mathbf{A}$ with the following changes. It sets the public parameters to $(g, \hat{g}, h, \hat{h})$, where (g, h) is from $\mathcal{D}$. Further, it defines the commitment key as follows:

$$\text{ck} := \mathbf{A} := \begin{pmatrix} g^{a_1} & g^{a_2} & g^{a_3} & g^{a_4} \\ g_1 & g_2 & g_3 & g_4 \\ h_1 & h_2 & h_3 & g_4 \\ g^{b_1} & g^{b_2} & g^{b_3} & g^{b_4} \end{pmatrix} \in \mathbb{G}^{4 \times 4},$$

where $a_i, b_i \leftarrow \mathbb{Z}_p$ for all $i \in [4]$ are sampled uniformly random. Otherwise, it behaves exactly as in game $\mathbf{G}_6$. In the end, it outputs whatever the game outputs. Clearly, if $\mathcal{D}$ is a random tuple,

then the key ck is identically distributed as a uniformly sampled key. In that case, the reduction perfectly simulates game $\mathbf{G}_6$. On the other hand, if $\mathcal{D}$ is an mDH4 tuple, then the second and third row are linearly dependent over (g, h) and identically distributed as in game $\mathbf{G}_7$. In particular, the reduction then perfectly simulates game $\mathbf{G}_7$. Therefore, we obtain

$$|\Pr[\mathbf{G}_7 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \varepsilon_{\text{mddh4}}.$$

Game $\mathbf{G}_8$ (Switch ck to Equivocable): In this game, we again change how we generate the commitment key ck . Namely, we generate the commitment key in trapdoor/equivocable mode as $\text{ck} \leftarrow \text{TGen}((\mathbb{G}, p, g), (X, Y))$, where $\text{pk} = (X, Y)$ denotes the public key. We claim that the commitment keys generated in this game and the previous game $\mathbf{G}_7$ are identically distributed, conditioned on $\text{pk} = (X, Y) \neq (1, 1)$ (which fails with probability at most $1/p$). First, observe that

$$\text{pk} = (X, Y) = (g^x \hat{g}^{\hat{x}}, h^x \hat{h}^{\hat{x}}) = (g^{x+\alpha\hat{x}}, h^{x+\alpha\hat{x}}) \in \text{Span}\{(g, h)\}_{\mathbb{Z}_p},$$

as we have generated the parameters such that $(g, \hat{g}, h, \hat{h}) = (g, g^\alpha, h, h^\alpha)$, see game $\mathbf{G}_5$. Now recall the structure of a trapdoor commitment key: We sample $d_{i,j} \leftarrow \mathbb{Z}_p$ for all $(i, j) \in [4] \times [4]$, and set

$$\text{ck} := \mathbf{A} := \begin{pmatrix} g^{d_{1,1}} & g^{d_{1,2}} & g^{d_{1,3}} & g^{d_{1,4}} \\ X^{d_{2,1}} & X^{d_{2,2}} & X^{d_{2,3}} & X^{d_{2,4}} \\ Y^{d_{2,1}} & Y^{d_{2,2}} & Y^{d_{2,3}} & Y^{d_{2,4}} \\ X^{d_{3,1}} Y^{d_{4,1}} & X^{d_{3,2}} Y^{d_{4,2}} & X^{d_{3,3}} Y^{d_{4,3}} & X^{d_{3,4}} Y^{d_{4,4}} \end{pmatrix} \in \mathbb{G}^{4 \times 4}.$$

In game $\mathbf{G}_7$, for each column $j \in [4]$, the pair $(A_{2,j}, A_{3,j})$ is of the form (g^d, h^d) for uniform $d \leftarrow \mathbb{Z}_p$, and the columns are independent. In particular, the second and third rows are linearly dependent over (g, h) . In this game, for each column $j \in [4]$, the pair $(A_{2,j}, A_{3,j})$ is of the form (X^d, Y^d) for uniform $d \leftarrow \mathbb{Z}_p$, and the columns are independent. In particular, the second and third rows are linearly dependent over (X, Y) , and thus also over (g, h) . Therefore, conditioned on $\text{pk} \neq (1, 1)$, the joint distribution of the second and third rows in games $\mathbf{G}_7$ and $\mathbf{G}_8$ is identical. It remains to argue that the last row in game $\mathbf{G}_8$ consists of uniform and independent elements, just as in game $\mathbf{G}_7$. Fix $j \in [4]$. Since $\text{pk} \neq (1, 1)$, at least one of X or Y is non-trivial. W.l.o.g, we assume $X \neq 1$. Then the map $\mathbb{Z}_p \rightarrow \mathbb{G}$, $d \mapsto X^d$, is an automorphism of groups. Hence, for uniform $d_{3,j} \leftarrow \mathbb{Z}_p$, the element $X^{d_{3,j}}$ is uniformly distributed in $\mathbb{G}$. Moreover, $Y^{d_{4,j}}$ is independent of $X^{d_{3,j}}$ since $d_{4,j}$ is sampled independently. Therefore, the product $A_{4,j} = X^{d_{3,j}} Y^{d_{4,j}}$ is uniformly distributed in $\mathbb{G}$. Since the pairs $(d_{3,j}, d_{4,j})$ are sampled independently across $j \in [4]$, the four elements in the last row are uniform and mutually independent. In game $\mathbf{G}_7$, the last row consists of the elements $(g^{d_{3,1}}, \dots, g^{d_{3,4}})$ with independent uniform exponents, and thus also forms four independent uniform elements of $\mathbb{G}$. Finally, in both games the randomness used to generate the last row is independent of the randomness used to generate the first three rows. Thus, the full commitment keys generated in game $\mathbf{G}_7$ and $\mathbf{G}_8$ are identically distributed, and we conclude that

$$|\Pr[\mathbf{G}_8 \Rightarrow 1] - \Pr[\mathbf{G}_7 \Rightarrow 1]| \leq 1/p.$$

The purpose of the subsequent sequence of games is to gradually transition to a game in which the trapdoor of the commitment scheme enables the simulation of both signing and corruption queries. All these games are identically distributed; therefore, the simulations may be inefficient.

Game $\mathbf{G}_9$ (Sample Commitment Randomly): In this game, we change how we compute the commitment com_i whenever the adversary $\mathbf{A}$ queries the first-round signing oracle $\text{SIGN}_1(\mathcal{S}, m, i)$. Recall that in the previous game, the commitment is honestly computed by sampling nonce randomness $r_i, \hat{r}_i \leftarrow \mathbb{Z}_p$ and commitment randomness $\rho_i \leftarrow \mathbb{Z}_p^4$, and then returning $\text{com}_i := \text{com}(\text{ck}, \mathbf{R}; \rho_i)$, where

$$\mathbf{R} := (R_1, R_2, R_3) := (g^{r_i} \hat{g}^{\hat{r}_i}, h^{r_i} \hat{h}^{\hat{r}_i}, u^{r_i} \hat{u}^{\hat{r}_i}) \quad \text{and} \quad (u, \hat{u}) := H_u(m).$$

In this game, we instead sample com_i randomly from its distribution and return it in $\text{SIGN}_1(\mathcal{S}, m, i)$. Given that the second and third row of the commitment matrix are linearly dependent over (g, h) , we generate $\text{com}_i := (g^{e_1}, g^{e_2}, h^{e_2}, g^{e_3}) \in \mathbb{G}^4$ where $e_1, e_2, e_3 \leftarrow \mathbb{Z}_p$ are sampled uniformly at random. In particular, we defer the generation of $r_i, \hat{r}_i$ and ρ_i until $\text{SIGN}_2(i, \text{sid}, \mathcal{M}_1)$ or $\text{CORR}(i)$ is queried. Once any of these two events happens, we proceed as follows. First, sample $r_i, \hat{r}_i \leftarrow \mathbb{Z}_p$ uniformly at random and compute $\mathbf{R}$ as defined above. Next, consider the identity $\text{com}_i = \mathbf{A}^{\rho_i}(1, \mathbf{R})$, and let $x_0 := \log_g(X) = x + \alpha \hat{x}$, where $(\hat{g}, \hat{h}) = (g, h)^\alpha$ from the setup. Further, let $\beta := \log_g h$. Since $Y = X^\beta$, we have $Y = g^{\beta x_0}$. Let $R_1 = g^{r_i} \hat{g}^{\hat{r}_i} = g^{r_i + \alpha \hat{r}_i}$ and define $r := \log_g(R_1)$, $r_3 := \log_g(R_3)$. From $\text{com}_i = \mathbf{A}^{\rho_i}(1, R_1, R_2, R_3)$ and $\text{com}_i = (g^{e_1}, g^{e_2}, h^{e_2}, g^{e_3})$, we obtain

$$\langle d_1, \rho_i \rangle = e_1, \quad \langle d_2, \rho_i \rangle = (e_2 - r)/x_0, \quad \langle d_3 + \beta d_4, \rho_i \rangle = (e_3 - r_3)/x_0,$$

where for each $i \in [4]$ we write $d_i := (d_{i,1}, \dots, d_{i,4}) \in \mathbb{Z}_p^4$ for the i -th row of the trapdoor matrix $\mathbf{D} \in \mathbb{Z}_p^{4 \times 4}$ defined in the previous game $\mathbf{G}_8$. In matrix form, we thus have

$$\mathbf{M} \rho_i := \begin{pmatrix} d_1 \\ d_2 \\ d_3 + \beta d_4 \end{pmatrix} \rho_i = \begin{pmatrix} e_1 \\ (e_2 - r)/x_0 \\ (e_3 - r_3)/x_0 \end{pmatrix}.$$

Since the trapdoor matrix $\mathbf{D} \in \mathbb{Z}_p^{4 \times 4}$ is sampled uniformly at random, the vectors $d_1, d_2, d_3 + \beta d_4 \in \mathbb{Z}_p^4$ are linearly independent except with probability at most $3/p^2$. Conditioning on this event, the linear system has rank 3. Thus, for every commitment com in the image of the commitment map $F(\rho) := \mathbf{A}^\rho(1, \mathbf{R})$, the set of solutions

$$S(\text{com}) := \{\rho \in \mathbb{Z}_p^4 \mid F(\rho) = \text{com}\}$$

forms a one-dimensional affine subspace of $\mathbb{Z}_p^4$. As ρ is sampled uniformly in the previous game, its conditional distribution given com is uniform over $S(\text{com})$. In this game, sampling uniformly from $S(\text{com})$ therefore yields a value ρ that is identically distributed to the real ρ conditioned on com . We therefore get

$$|\Pr[\mathbf{G}_9 \Rightarrow 1] - \Pr[\mathbf{G}_8 \Rightarrow 1]| \leq 3/p^2.$$

Game $\mathbf{G}_{10}$ (Sample Combined Randomness): In this game, we no longer generate the values $r_i, \hat{r}_i$ and ρ_i inside $\text{SIGN}_2(i, \text{sid}, \mathcal{M}_1)$ for each $i \in \mathcal{H} \cap \mathcal{S}_{\text{sid}}$. Instead, once $\text{SIGN}_2(i, \text{sid}, \mathcal{M}_1)$ is queried for the last honest signer in $\mathcal{S}_{\text{sid}}$,¹⁰ we generate only the combined values

$$r = \sum_{i \in \mathcal{H} \cap \mathcal{S}_{\text{sid}}} r_i, \quad \hat{r} = \sum_{i \in \mathcal{H} \cap \mathcal{S}_{\text{sid}}} \hat{r}_i, \quad \rho = \sum_{i \in \mathcal{H} \cap \mathcal{S}_{\text{sid}}} \rho_i.$$

In more detail, we sample values $r, \hat{r} \leftarrow \mathbb{Z}_p$, and compute $\mathbf{R} := (R_1, R_2, R_3) := (g^r \hat{g}^{\hat{r}}, h^r \hat{h}^{\hat{r}}, u^r \hat{u}^{\hat{r}})$. Next, we compute $\text{com} := \otimes_{i \in \mathcal{H} \cap \mathcal{S}_{\text{sid}}} \text{com}_i$. Note that this value is well defined, as all honest signers already completed the first round. Then, from the identity $\text{com} = \mathbf{A}^\rho(1, \mathbf{R})$, we can generate ρ as in the previous game (i.e., by sampling ρ uniformly from the one-dimensional solution space of the linear system over $\mathbb{Z}_p$). We claim that this game is identically distributed to the previous game. Fix a session identifier sid and signer set $\mathcal{S}_{\text{sid}}$, and let $H := \mathcal{H} \cap \mathcal{S}_{\text{sid}}$ denote the set of honest signers in this session. Let $T \subseteq H$ be the subset of honest signers for which the adversary queries the corruption oracle $\text{CORR}(\cdot)$ before the final honest second-round query $\text{SIGN}_2(\cdot)$ for this session. By design of the signing protocol, we have $|T| \leq |H| - 1$, i.e., there always exists at least one honest signer in H that is never corrupted.

In game $\mathbf{G}_9$, for each $i \in H$ the values $r_i, \hat{r}_i \leftarrow \mathbb{Z}_p$ are sampled independently and uniformly, and the aggregate values are defined as $r := \sum_{i \in H} r_i$ and $\hat{r} := \sum_{i \in H} \hat{r}_i$. Since $|H \setminus T| \geq 1$, it follows

¹⁰We stress that queries to $\text{CORR}(i)$ for any honest signer $i \in \mathcal{H} \cap \mathcal{S}_{\text{sid}}$ are answered exactly as in the previous game.

that, conditioned on the values $\{r_i, \hat{r}_i\}_{i \in T}$ revealed via corruption queries, the sums r and $\hat{r}$ remain uniformly distributed over $\mathbb{Z}_p$. Equivalently, the joint distribution of $(\{r_i, \hat{r}_i\}_{i \in T}, r, \hat{r})$ in game $\mathbf{G}_9$ is identical to the distribution obtained by sampling $r, \hat{r} \leftarrow \mathbb{Z}_p$ uniformly and independently, and then defining the remaining shares $\{r_j, \hat{r}_j\}_{j \in H \setminus T}$ so that the sum constraints are satisfied. Next consider the commitment randomness. In game $\mathbf{G}_9$, for each $i \in H$, the value ρ_i is sampled uniformly from the one-dimensional affine solution space determined by com_i and the corresponding $\mathbf{R}_i$. Writing $\rho_i = \rho_{i,0} + t_i \mathbf{v}$ for a fixed nonzero direction $\mathbf{v} \in \ker(\mathbf{M})$ and a uniformly random scalar $t_i \leftarrow \mathbb{Z}_p$, we obtain

$$\rho := \sum_{i \in H} \rho_i = \rho_0 + \left(\sum_{i \in H} t_i \right) \mathbf{v}.$$

Since at least one coefficient t_i corresponding to an uncorrupted honest signer remains hidden from the adversary, the sum $\sum_{i \in H} t_i$ is uniformly distributed over $\mathbb{Z}_p$, even conditioned on the adversary's view. Consequently, the aggregate value ρ is uniformly distributed over the affine solution space $S(\text{com})$ of the combined commitment $\text{com} = \bigotimes_{i \in H} \text{com}_i$.

In game $\mathbf{G}_{10}$, we proceed analogously, but sample the randomness in a different order. Upon the final honest second-round query for the session, we sample aggregate values $r, \hat{r} \leftarrow \mathbb{Z}_p$ uniformly at random and compute the corresponding aggregate $\mathbf{R} = (g^r \hat{g}^{\hat{r}}, h^r \hat{h}^{\hat{r}}, u^r \hat{u}^{\hat{r}})$. We then sample ρ uniformly from the affine solution space $S(\text{com})$ of the combined commitment $\text{com} = \bigotimes_{i \in H} \text{com}_i$, ensuring that $\text{com} = A^\rho(1, \mathbf{R})$ holds. For any subsequent corruption query $\text{CORR}(i)$ with $i \in H$, we lazily sample values $(r_i, \hat{r}_i, \rho_i)$ that are consistent with the already fixed aggregates, i.e., such that

$$r = \sum_{i \in H} r_i, \quad \hat{r} = \sum_{i \in H} \hat{r}_i, \quad \rho = \sum_{i \in H} \rho_i.$$

Since at least one honest signer in H is never corrupted, the values revealed through corruption queries are distributed identically to those in game $\mathbf{G}_9$. Therefore, the adversary's view in the two games is identically distributed, and we conclude that

$$\Pr[\mathbf{G}_{10} \Rightarrow 1] = \Pr[\mathbf{G}_9 \Rightarrow 1].$$

Game $\mathbf{G}_{11}$ (Sample Response First): In this game, we change how we generate the aggregate nonce $\mathbf{R} = (R_1, R_2, R_3)$ in the second round. Recall that by the end of the first round the challenge value $c = H_c(\text{pk}, \text{com}, U, m) \in \mathbb{Z}_p$ is already fixed, and the public values $X = g^x \hat{g}^{\hat{x}}$, $Y = h^x \hat{h}^{\hat{x}}$, and $U = u^x \hat{u}^{\hat{x}}$ are already defined (and $(u, \hat{u}) := H_u(m)$ is fixed as well). In the previous game, we sample $r, \hat{r} \leftarrow \mathbb{Z}_p$ and set

$$\mathbf{R} := (R_1, R_2, R_3) := (g^r \hat{g}^{\hat{r}}, h^r \hat{h}^{\hat{r}}, u^r \hat{u}^{\hat{r}}).$$

Then we compute $z := r + cx$, $\hat{z} := \hat{r} + c\hat{x}$; in particular, the following relations hold:

$$g^z \hat{g}^{\hat{z}} = X^c \cdot R_1, \quad h^z \hat{h}^{\hat{z}} = Y^c \cdot R_2, \quad u^z \hat{u}^{\hat{z}} = U^c \cdot R_3. \quad (1)$$

In this game, we change the order of sampling as follows. Whenever the last honest second-round query for a session is made, we first sample $z, \hat{z} \leftarrow \mathbb{Z}_p$ uniformly at random, and then compute the values $\mathbf{R} = (R_1, R_2, R_3)$ by rearranging the above equation, namely:

$$R_1 := g^z \hat{g}^{\hat{z}} \cdot X^{-c}, \quad R_2 := h^z \hat{h}^{\hat{z}} \cdot Y^{-c}, \quad R_3 := u^z \hat{u}^{\hat{z}} \cdot U^{-c}.$$

Afterwards, we generate ρ exactly as in game $\mathbf{G}_{10}$ from the well-defined identity $\text{com} = A^\rho(1, \mathbf{R})$, i.e., we sample ρ uniformly from the (one-dimensional) affine solution space of the corresponding linear system over $\mathbb{Z}_p$. We claim that the distribution of both games is identical. To this end, fix $c \in \mathbb{Z}_p$ and the public values X, Y, U (which are already determined after round 1). In game $\mathbf{G}_{10}$ we have

$$g^z \hat{g}^{\hat{z}} = g^{r+cx} \hat{g}^{\hat{r}+c\hat{x}} \quad \text{with} \quad r, \hat{r} \leftarrow \mathbb{Z}_p \text{ uniform,}$$

hence $(z, \hat{z}) = (r + cx, \hat{r} + c\hat{x})$ is uniformly distributed over $\mathbb{Z}_p^2$ since it is obtained from $(r, \hat{r})$ by an invertible translation. Moreover, using $X = g^x \hat{g}^{\hat{x}}$, $Y = h^x \hat{h}^{\hat{x}}$, and $U = u^x \hat{u}^{\hat{x}}$, we can rewrite

$$R_1 = g^r \hat{g}^{\hat{r}} = g^z \hat{g}^{\hat{z}} \cdot X^{-c}, \quad R_2 = h^z \hat{h}^{\hat{z}} \cdot Y^{-c}, \quad R_3 = u^z \hat{u}^{\hat{z}} \cdot U^{-c}.$$

Therefore, the distribution of $\mathbf{R}$ induced in $\mathbf{G}_{10}$ by sampling $(r, \hat{r})$ and setting $(z, \hat{z}) = (r + cx, \hat{r} + c\hat{x})$ is exactly the same as the distribution in $\mathbf{G}_{11}$ obtained by sampling $(z, \hat{z}) \leftarrow \mathbb{Z}_p^2$ uniformly and defining $\mathbf{R}$ via Equation (1). Since the subsequent computation of ρ depends only on $(\text{com}, \mathbf{R})$ and is performed identically in both games, the adversary's view is identically distributed in $\mathbf{G}_{10}$ and $\mathbf{G}_{11}$, and we get $\Pr[\mathbf{G}_{11} \Rightarrow 1] = \Pr[\mathbf{G}_{10} \Rightarrow 1]$.

Game $\mathbf{G}_{12}$ (Sample Per-Signer Responses): In this game, we change how the values revealed by corruption queries are generated and how the aggregate response $(z, \hat{z})$ used to compute the aggregate nonce $\mathbf{R} = (R_1, R_2, R_3)$ in the second round is obtained. More precisely, upon each query $\text{SIGN}_1(\mathcal{S}, m, i)$ with $i \in \mathcal{H}$, we sample and store uniform and independent values $z_i, \hat{z}_i \leftarrow \mathbb{Z}_p$ as internal state for signer i associated with this invocation of SIGN_1 . Apart from storing $(z_i, \hat{z}_i)$, we answer SIGN_1 exactly as in the previous game. Next, in $\text{SIGN}_2(i, \text{sid}, \mathcal{M}_1)$ we proceed exactly as before, except that when $\text{SIGN}_2(\cdot)$ is invoked for the last honest signer in $\mathcal{S}_{\text{sid}}$, we first define the aggregate responses as

$$z := \sum_{j \in \mathcal{H} \cap \mathcal{S}_{\text{sid}}} z_j, \quad \hat{z} := \sum_{j \in \mathcal{H} \cap \mathcal{S}_{\text{sid}}} \hat{z}_j,$$

where the sums range over the values $(z_j, \hat{z}_j)$ stored at the corresponding $\text{SIGN}_1(\mathcal{S}, m, j)$ queries. We then compute the aggregate nonce $\mathbf{R} = (R_1, R_2, R_3)$ and the value ρ exactly as in the previous game. Finally, upon a corruption query $\text{CORR}(i)$ for an honest signer $i \in \mathcal{H}$, instead of sampling fresh values $r_i, \hat{r}_i \leftarrow \mathbb{Z}_p$ as in game $\mathbf{G}_{11}$, we set $r_i := z_i, \hat{r}_i := \hat{z}_i$, and output $(r_i, \hat{r}_i, \rho_i)$, where ρ_i is generated exactly as before.

We claim that the distributions of values in both games are identical. For this, observe that the only difference between games $\mathbf{G}_{11}$ and $\mathbf{G}_{12}$ is that in this game, we represent the aggregate responses $(z, \hat{z})$ as sums of independently sampled values $\{(z_j, \hat{z}_j)\}_{j \in \mathcal{H} \cap \mathcal{S}}$ that are drawn already in SIGN_1 . Since each $(z_j, \hat{z}_j)$ is uniform over $\mathbb{Z}_p^2$, their sum $(z, \hat{z}) = (\sum_j z_j, \sum_j \hat{z}_j)$ is uniform over $\mathbb{Z}_p^2$ as well, and therefore the induced distribution of the aggregate nonce $\mathbf{R}$ computed from $(z, \hat{z})$ is identical to $\mathbf{G}_{11}$, where $(z, \hat{z})$ is sampled directly as uniform. Moreover, answering $\text{CORR}(i)$ with $(r_i, \hat{r}_i) = (z_i, \hat{z}_i)$ does not change the distribution, because in game $\mathbf{G}_{11}$ the values $(r_i, \hat{r}_i)$ returned upon corruption are uniform in $\mathbb{Z}_p^2$, and in game $\mathbf{G}_{12}$ the stored values $(z_i, \hat{z}_i)$ are also uniform in $\mathbb{Z}_p^2$. Finally, ρ and each revealed ρ_i are generated in both games by sampling uniformly from the same affine solution spaces determined by the corresponding pairs $(\text{com}, \mathbf{R})$ and $(\text{com}_i, \mathbf{R}_i)$, respectively. Hence the adversary's view is identically distributed in both games, and therefore $\Pr[\mathbf{G}_{12} \Rightarrow 1] = \Pr[\mathbf{G}_{11} \Rightarrow 1]$.

We introduce a purely syntactical change as follows. In addition to sampling the values $z_i, \hat{z}_i$ upon each query $\text{SIGN}_1(\mathcal{S}, m, i)$ with $i \in \mathcal{H}$, we also store the corresponding group elements

$$\mathbf{m}_i := (m_{i,1}, m_{i,2}, m_{i,3}) := (g^{z_i} \hat{g}^{\hat{z}_i}, h^{z_i} \hat{h}^{\hat{z}_i}, u^{z_i} \hat{u}^{\hat{z}_i})$$

as internal state. All other parts of the game remain unchanged. It is immediate that this modification does not affect the adversary's view: the values $\mathbf{m}_i$ are deterministic functions of the already-sampled uniform randomness $(z_i, \hat{z}_i)$ and are never revealed independently. Consequently, this change is purely syntactical and preserves the distribution of all values visible to the adversary.

Game $\mathbf{G}_{13}$ (Change Commitment Generation): In this game, we change the way the commitment com_i is generated upon a first-round signing query $\text{SIGN}_1(\mathcal{S}, m, i)$ with $i \in \mathcal{H}$. In the previous game, we sample $e_{1,i}, e_{2,i}, e_{3,i} \leftarrow \mathbb{Z}_p$ and return $\text{com}_i := (g^{e_{1,i}}, g^{e_{2,i}}, h^{e_{2,i}}, g^{e_{3,i}})$. In this game, we instead sample $\zeta_i = (\zeta_{1,i}, \zeta_{2,i}, \zeta_{3,i}, \zeta_{4,i}) \leftarrow \mathbb{Z}_p^4$ and return

$$\text{com}_i := (g^{\zeta_{1,i}}, m_{i,1} X^{\zeta_{2,i}}, m_{i,2} Y^{\zeta_{2,i}}, m_{i,3} X^{\zeta_{3,i}} Y^{\zeta_{4,i}}) \in \mathbb{G}^4,$$

where $\mathbf{m}_i = (m_{i,1}, m_{i,2}, m_{i,3})$ is the internal state defined above. All other parts of the game remain unchanged. We claim that the distributions of both games are identical, conditioned on $X \neq 1$. Let $x_0 \in \mathbb{Z}_p$ such that $(X, Y) = (g, h)^{x_0}$. Moreover, for each honest signer i , there exists $s_i := z_i + \alpha \hat{z}_i$ such that $m_{i,1} = g^{s_i}$, $m_{i,2} = h^{s_i}$. Fix the internal state (s_i, t_i) , where $t_i := \log_g(m_{i,3})$. Define the map $\Phi_{s_i} : \mathbb{Z}_p^4 \rightarrow \mathbb{Z}_p^3$,

$$(\zeta_{1,i}, \zeta_{2,i}, \zeta_{3,i}, \zeta_{4,i}) \mapsto (e_{1,i}, e_{2,i}, e_{3,i}) := (\zeta_{1,i}, s_i + x_0 \zeta_{2,i}, t_i + x_0 \zeta_{3,i} + \beta x_0 \zeta_{4,i}).$$

A direct calculation shows that $\text{com}_i = (g^{e_{1,i}}, g^{e_{2,i}}, h^{e_{3,i}}, g^{e_{3,i}})$, with $(e_{1,i}, e_{2,i}, e_{3,i}) = \Phi_{s_i}(\zeta_i)$. Assuming $x_0 \neq 0$ (equivalently $X \neq 1$), the map $\zeta_{2,i} \mapsto s_i + x_0 \zeta_{2,i}$ is a bijection on $\mathbb{Z}_p$, and the map $(\zeta_{3,i}, \zeta_{4,i}) \mapsto x_0 \zeta_{3,i} + \beta x_0 \zeta_{4,i}$ is surjective with each value having exactly p preimages. Therefore, $(e_{1,i}, e_{2,i}, e_{3,i})$ is uniformly distributed over $\mathbb{Z}_p^3$ whenever ζ_i is uniform over $\mathbb{Z}_p^4$. It follows that com_i is distributed identically to the commitment sampled in $\mathbf{G}_{12}$. Thus, we get

$$|\Pr[\mathbf{G}_{13} \Rightarrow 1] - \Pr[\mathbf{G}_{12} \Rightarrow 1]| \leq 1/p.$$

Game $\mathbf{G}_{14}$ (Trapdoor Equivocation): In this game, we change the way the commitment randomness is computed upon the last honest second-round signing query and upon corruption queries. Namely, instead of computing these values inefficiently as in the previous game, we now compute them efficiently using the commitment trapdoor matrix $\mathbf{D} \in \mathbb{G}^{4 \times 4}$.

Efficient computation of the aggregate commitment randomness. Upon the last honest second-round signing query $\text{SIGN}_2(i, \text{sid}, \mathcal{M}_1)$, let $H := \mathcal{H} \cap \mathcal{S}_{\text{sid}}$ be the honest signers in that session. The game computes the combined commitment and nonce as before, namely:

$$\text{com} := \bigotimes_{i \in H} \text{com}_i = (g^{\zeta_1}, m_1 X^{\zeta_2}, m_2 Y^{\zeta_2}, m_3 X^{\zeta_3} Y^{\zeta_4}),$$

$$\mathbf{R} := (R_1, R_2, R_3) := (m_1 X^{-c}, m_2 Y^{-c}, m_3 U^{-c}),$$

where

$$(m_1, m_2, m_3) := \left(\prod_{i \in H} m_{i,1}, \prod_{i \in H} m_{i,2}, \prod_{i \in H} m_{i,3} \right), \quad \zeta := \sum_{i \in H} \zeta_i$$

denote the component-wise aggregation of the individual values $\mathbf{m}_i = (m_{i,1}, m_{i,2}, m_{i,3})$ and $\zeta_i = (\zeta_{1,i}, \zeta_{2,i}, \zeta_{3,i}, \zeta_{4,i})$ over all honest signers $i \in H$. Moreover, recall that $U = X^a Y^b$ for the trapdoor values (a, b) associated with $H_u(m_{\text{sid}})$. By construction, the commitment must satisfy Equation 1, i.e., $\text{com} = \mathbf{A}^\rho(1, \mathbf{R})$. Substituting the above expressions yields

$$\mathbf{A}^\rho = (g^{\zeta_1}, X^{\zeta_2+c}, Y^{\zeta_2+c}, X^{\zeta_3+ac} Y^{\zeta_4+bc}). \quad (2)$$

Recall that the trapdoor commitment key $\mathbf{A} \in \mathbb{G}^{4 \times 4}$ is generated together with a trapdoor matrix $\mathbf{D} = (d_{i,j}) \in \mathbb{Z}_p^{4 \times 4}$ such that for any $\rho \in \mathbb{Z}_p^4$,

$$\mathbf{A}^\rho = (g^{\langle d_1, \rho \rangle}, X^{\langle d_2, \rho \rangle}, Y^{\langle d_2, \rho \rangle}, X^{\langle d_3, \rho \rangle} Y^{\langle d_4, \rho \rangle}).$$

Here, for each $k \in \{1, 2, 3, 4\}$, we write $d_k := (d_{k,1}, \dots, d_{k,4}) \in \mathbb{Z}_p^4$ for the k -th row of the trapdoor matrix $\mathbf{D}$, and $\langle \cdot, \cdot \rangle$ denotes the standard inner product over $\mathbb{Z}_p$. Matching exponents yields the linear system

$$\mathbf{D} \rho = \begin{pmatrix} \zeta_1 \\ \zeta_2 + c \\ \zeta_3 + ac \\ \zeta_4 + bc \end{pmatrix}.$$

The game computes ρ efficiently using the trapdoor as $\rho := \mathbf{D}^{-1}(\zeta_1, \zeta_2 + c, \zeta_3 + ac, \zeta_4 + bc)^T$, aborting if $\mathbf{D}$ is not invertible (which happens with probability at most $4/p$).

Efficient computation of individual randomness upon corruption. We also compute the individual commitment randomness values ρ_i efficiently upon corruption queries. Namely, upon a corruption query $\text{CORR}(i)$ for an honest signer $i \in \mathcal{H}$, the game proceeds as follows. As in the previous game, it sets $(r_i, \hat{r}_i) := (z_i, \hat{z}_i)$ and defines the individual nonce

$$\mathbf{R}_i := (R_{1,i}, R_{2,i}, R_{3,i}) := (g^{r_i} \hat{g}^{\hat{r}_i}, h^{r_i} \hat{h}^{\hat{r}_i}, u^{r_i} \hat{u}^{\hat{r}_i}),$$

where $(u, \hat{u}) = H_u(m_{\text{sid}})$ is already fixed for the corresponding signing session. The commitment com_i must satisfy Equation 1, i.e., $\text{com}_i = \mathbf{A}^{\rho_i}(1, \mathbf{R}_i)$. Writing

$$\text{com}_i = (g^{\zeta_{1,i}}, m_{i,1} X^{\zeta_{2,i}}, m_{i,2} Y^{\zeta_{2,i}}, m_{i,3} X^{\zeta_{3,i}} Y^{\zeta_{4,i}}),$$

and recalling that

$$(m_{i,1}, m_{i,2}, m_{i,3}) = (g^{z_i} \hat{g}^{\hat{z}_i}, h^{z_i} \hat{h}^{\hat{z}_i}, u^{z_i} \hat{u}^{\hat{z}_i}) = (R_{1,i}, R_{2,i}, R_{3,i}),$$

we obtain

$$\mathbf{A}^{\rho_i} = \text{com}_i \cdot (1, \mathbf{R}_i)^{-1} = (g^{\zeta_{1,i}}, X^{\zeta_{2,i}}, Y^{\zeta_{2,i}}, X^{\zeta_{3,i}} Y^{\zeta_{4,i}}).$$

Using the trapdoor matrix $\mathbf{D}$, we can then compute ρ_i efficiently as $\rho_i := \mathbf{D}^{-1}(\zeta_{1,i}, \zeta_{2,i}, \zeta_{3,i}, \zeta_{4,i})^T$, aborting if $\mathbf{D}$ is not invertible. Conditioned on $\mathbf{D}$ being invertible, we claim that the distributions of the values in both games are identical.

Recall that in the previous game $\mathbf{G}_{13}$, after fixing $(\text{com}, \mathbf{R})$, the commitment randomness ρ was sampled uniformly from the one-dimensional affine solution space

$$S(\text{com}, \mathbf{R}) := \{\rho \in \mathbb{Z}_p^4 \mid \text{com} = \mathbf{A}^{\rho}(1, \mathbf{R})\}.$$

In this game, the value ρ is instead computed by solving the full-rank linear system $\mathbf{D} \rho = (\zeta_1, \zeta_2 + c, \zeta_3 + ac, \zeta_4 + bc)^T$. Although this system admits a unique solution for fixed $(\zeta_1, \zeta_2, \zeta_3, \zeta_4)$, this does not eliminate the randomness in ρ when conditioning on $(\text{com}, \mathbf{R})$. For this, observe that the commitment

$$\text{com} = (g^{\zeta_1}, m_1 X^{\zeta_2}, m_2 Y^{\zeta_2}, m_3 X^{\zeta_3} Y^{\zeta_4}) = (g^{\zeta_1}, m_1 X^{\zeta_2}, m_2 Y^{\zeta_2}, m_3 X^{\omega}),$$

where $\omega := \zeta_3 + \beta \zeta_4$, depends on the pair (ζ_3, ζ_4) only through the single scalar ω . Consequently, conditioned on $(\text{com}, \mathbf{R})$, the values ζ_1 , ζ_2 , and ω are fixed, while ζ_4 remains uniformly distributed over $\mathbb{Z}_p$, and ζ_3 is uniquely determined as $\zeta_3 = \omega - \beta \zeta_4$. Substituting this relation into the expression for ρ yields

$$\rho = \mathbf{D}^{-1} \begin{pmatrix} \zeta_1 \\ \zeta_2 + c \\ \omega - \beta \zeta_4 + ac \\ \zeta_4 + bc \end{pmatrix} = \rho_0 + \zeta_4 \cdot \mathbf{v}, \quad \rho_0 := \mathbf{D}^{-1} \begin{pmatrix} \zeta_1 \\ \zeta_2 + c \\ \omega + ac \\ bc \end{pmatrix}, \quad \mathbf{v} := \mathbf{D}^{-1} \begin{pmatrix} 0 \\ 0 \\ -\beta \\ 1 \end{pmatrix}.$$

Since ζ_4 is uniformly distributed over $\mathbb{Z}_p$, it follows that ρ is uniformly distributed over the affine line $\rho_0 + \text{Span}\{\mathbf{v}\}$ (note that $\mathbf{v} \neq 0$). It remains to observe that this affine line coincides exactly with the solution space $S(\text{com}, \mathbf{R})$. Indeed, by definition $\mathbf{D} \mathbf{v} = (0, 0, -\beta, 1)^T$, so $\langle d_1, \mathbf{v} \rangle = 0$, $\langle d_2, \mathbf{v} \rangle = 0$, $\langle d_3 + \beta d_4, \mathbf{v} \rangle = 0$, and therefore adding any multiple of $\mathbf{v}$ to a solution ρ does not change the value of $\mathbf{A}^{\rho}(1, \mathbf{R})$; recall that

$$\mathbf{A}^{\rho} = (g^{\langle d_1, \rho \rangle}, X^{\langle d_2, \rho \rangle}, Y^{\langle d_2, \rho \rangle}, X^{\langle d_3, \rho \rangle} Y^{\langle d_4, \rho \rangle}).$$

Hence, $\rho_0 + \text{Span}\{\mathbf{v}\} = S(\text{com}, \mathbf{R})$, and the value ρ computed in this game is distributed identically to the value sampled in the previous game. The same argument applies to each individual value ρ_i computed upon corruption queries.

Uniformity of the aggregate value ζ_4 . Finally, we argue why ζ_4 as a sum of individual (uniform and independent) $\zeta_{4,i}$ is uniformly distributed over $\mathbb{Z}_p$, even when the adversary chooses the signer set $\mathcal{S}_{\text{sid}}$ adaptively after observing many first-round commitments. Let τ denote the full first-round transcript available to the adversary, including all commitments $\{\text{com}_i\}$ revealed before the adversary decides which signing session to complete. Suppose that the signer set $\mathcal{S}_{\text{sid}}$ is chosen as an arbitrary (possibly randomized) function of τ , and let $H := \mathcal{H} \cap \mathcal{S}_{\text{sid}}$ be the resulting set of honest signers. Let $T \subseteq H$ denote the subset of honest signers corrupted by the adversary, and assume $|T| \leq |H| - 1$, so that there exists an uncorrupted honest signer $j \in H \setminus T$. For each signer i , we define

$$\omega_i := \zeta_{3,i} + \beta \zeta_{4,i}, \quad \text{and} \quad \omega := \sum_{i \in H} \omega_i = \sum_{i \in H} \zeta_{3,i} + \beta \sum_{i \in H} \zeta_{4,i} = \zeta_3 + \beta \zeta_4.$$

Recall that the commitment com_i reveals ω_i through its last coordinate, but does not reveal any additional information about $\zeta_{4,i}$ beyond ω_i . Now, since $(\zeta_{3,j}, \zeta_{4,j}) \leftarrow \mathbb{Z}_p^2$ is sampled uniformly and independently, and the map $(\zeta_{3,j}, \zeta_{4,j}) \mapsto (\omega_j, \zeta_{4,j})$ with $\omega_j = \zeta_{3,j} + \beta \zeta_{4,j}$ is a bijection on $\mathbb{Z}_p^2$, it follows that $(\omega_j, \zeta_{4,j})$ is uniformly distributed over $\mathbb{Z}_p^2$. In particular, conditioning on τ (and hence on ω_j) and on the event that $j \in \mathcal{S}_{\text{sid}}$, the value $\zeta_{4,j}$ remains uniform over $\mathbb{Z}_p$. Writing

$$\zeta_4 = \sum_{i \in H} \zeta_{4,i} = \left(\sum_{i \in H \setminus \{j\}} \zeta_{4,i} \right) + \zeta_{4,j},$$

we see that ζ_4 is the sum of a value fixed by the adversary's view and an independent uniform term $\zeta_{4,j}$. Hence, the aggregate ζ_4 is itself uniformly distributed over $\mathbb{Z}_p$, even under an adaptive choice of the signer set $\mathcal{S}_{\text{sid}}$. Since $\zeta_4 = \sum_{i \in T} \zeta_{4,i} + \sum_{i \in H \setminus T} \zeta_{4,i}$ and $\zeta_{4,j}$ is uniform and independent of the adversary's view, it follows that the aggregate ζ_4 remains uniformly distributed over $\mathbb{Z}_p$ even conditioned on all values revealed by corruption queries and on the value ω induced by the commitment.

Although in the current game we compute a unique representative ρ (and similarly each ρ_i) by solving a full-rank linear system, this merely fixes a canonical representative from the same one-dimensional affine solution space that was sampled from in the previous games. As shown above, these values remain uniformly distributed, and thus the joint distribution of all values visible to the adversary remains unchanged. Unconditioning on the event that the trapdoor matrix $\mathbf{D}$ is invertible, we conclude with $|\Pr[\mathbf{G}_{14} \Rightarrow 1] - \Pr[\mathbf{G}_{13} \Rightarrow 1]| \leq 4/p$.

Conclusion so far. The above sequence of games establishes that the simulator never needs the signing secret key $(x, \hat{x})$. For each message m queried for the first time, it obtains/programs a trapdoor (a_m, b_m) for $H_u(m)$ and returns $U = U_m = X^{a_m} Y^{b_m}$. All first-round commitments com_i are generated in trapdoor mode from uniformly random $(z_i, \hat{z}_i, \zeta_i)$ using the commitment trapdoor matrix $\mathbf{D}$. The aggregate opening of $\text{com} = \bigotimes_i \text{com}_i$ is computed via trapdoor equivocation from $\zeta = \sum_i \zeta_i$, (a_m, b_m) , and the challenge c . Corruption queries are answered by setting $r_i := z_i$, $\hat{r}_i := \hat{z}_i$ and using individual equivocation to compute ρ_i . Finally, upon a forgery (m_*, σ_*) , the winning condition is checked using the trapdoor (a_*, b_*) for m_* . We now switch $(g, \hat{g}, h, \hat{h})$ back to the real distribution and give the reduction to the one-more image game.

Game $\mathbf{G}_{15}$ (Switch $(g, \hat{g}, h, \hat{h})$ to Random): In this game, we change the distribution of the public parameters $(g, \hat{g}, h, \hat{h})$ again. Instead of generating $(\hat{g}, \hat{h}) = (g, h)^\alpha$, we instead sample them uniformly random. Concretely, we generate the parameters as $\hat{g}, h, \hat{h} \leftarrow \mathbb{Z}_p$ sampled uniformly random and independent. We can bound the distinguishing advantage between games $\mathbf{G}_{15}$ and $\mathbf{G}_{14}$ via a reduction $\mathbf{B}$ to DDH. To this end, $\mathbf{B}$ gets a tuple $\mathcal{D} := (g, \hat{g}, h, T) \in \mathbb{G}^4$ as input. It then simulates game $\mathbf{G}_{14}$ for $\mathbf{A}$ with the following change: it sets the public parameters to $(g, \hat{g}, h, \hat{h})$ with $\hat{h} := T$. Otherwise, it behaves exactly as the game. In the end, it outputs whatever the game outputs. If $\mathcal{D}$ is a DH tuple, then $\mathbf{B}$ perfectly simulates game $\mathbf{G}_{14}$. On the other hand, if $\mathcal{D}$ is not a DH tuple, then $\mathbf{B}$ perfectly simulates game $\mathbf{G}_{15}$. Therefore, we obtain $|\Pr[\mathbf{G}_{15} \Rightarrow 1] - \Pr[\mathbf{G}_{14} \Rightarrow 1]| \leq \varepsilon_{\text{ddh}}$.

Reduction to the One-More Image Game. We construct a reduction $\mathbf{B}$ that simulates the final game for any adversary $\mathbf{A}$ and uses a successful forgery to win the one-more image game. The one-more image challenger provides $\mathbf{B}$ with the public parameters $\text{par}' = (\mathbb{G}, p, g, \hat{g}, h, \hat{h})$ and two group elements X, Y . The reduction samples the commitment key in trapdoor mode as in the final game, that is, $\text{ck} \leftarrow \text{TGen}((\mathbb{G}, p, g), (X, Y))$. It then forwards the public parameters and public key

$$\text{par} := (\mathbb{G}, p, g, \hat{g}, h, \hat{h}, \text{ck}), \quad \text{pk} := (X, Y),$$

respectively, to $\mathbf{A}$ as input. For each message m queried to $\mathbf{H}_u$ for the first time, $\mathbf{B}$ calls the oracle CHAL to obtain $(u_m, \hat{u}_m) = (g^{a_m} h^{b_m}, \hat{g}^{a_m} \hat{h}^{b_m})$, programs $\mathbf{H}_u(m)$ accordingly, and records the corresponding challenge index sid_m . When the simulation requires the value U upon the first first-round signing query $\text{SIGN}_1(\mathcal{S}, m, i)$, $\mathbf{B}$ queries the oracle PREIM on this challenge index sid_m to obtain (a_m, b_m) and returns $U = X^{a_m} Y^{b_m}$. All signing and corruption queries are answered exactly as in the final game, using only the programmed trapdoors and the commitment trapdoor, and without reference to the secret key.

If $\mathbf{A}$ outputs a forgery (m_*, σ_*) on a fresh message m_* , then $\mathbf{B}$ outputs the corresponding challenge index sid_{m_*} together with U_* . By the winning condition in the final game, we have $U_* = X^{a_*} Y^{b_*}$, and since $\mathbf{B}$ never queried the oracle PREIM for this challenge index sid_{m_*} , this constitutes a valid one-more image solution. We therefore conclude with $\Pr[\mathbf{G}_{15} \Rightarrow 1] \leq \varepsilon_{\text{one-more}}$. Combining this with the previous hybrids, we conclude the lemma. $\square$

Theorem 4. We prove Theorem 4 via a sequence of games. Let $\mathbf{G}_0$ be the original one-more image game.

Game $\mathbf{G}_1$ (Switch $(g, \hat{g}, h, \hat{h})$ to DH): In this game, we change the distribution of the public parameters $(g, \hat{g}, h, \hat{h})$. Instead of sampling $\hat{g}, \hat{h} \leftarrow \mathbb{G}$ uniformly at random (as in the previous games), we sample $w \leftarrow \mathbb{Z}_p$ and set $(h, \hat{h}) := (g^w, \hat{g}^w)$, so that $(g, \hat{g}, h, \hat{h})$ forms a DH tuple. We can bound the distinguishing advantage between games $\mathbf{G}_5$ and $\mathbf{G}_4$ via a straightforward reduction $\mathbf{B}$ to DDH, and therefore omit it. Thus, we have

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| \leq \varepsilon_{\text{ddh}}.$$

Game $\mathbf{G}_2$ (Sample $a_{\text{sid}} + wb_{\text{sid}}$ in Chal): Recall that in CHAL the challenger samples $a_{\text{sid}}, b_{\text{sid}} \leftarrow \mathbb{Z}_p$ uniformly and independently, and defines

$$u_{\text{sid}} := g^{a_{\text{sid}}} h^{b_{\text{sid}}} = g^{a_{\text{sid}} + wb_{\text{sid}}}, \quad \hat{u}_{\text{sid}} := \hat{g}^{a_{\text{sid}}} \hat{h}^{b_{\text{sid}}} = \hat{g}^{a_{\text{sid}} + wb_{\text{sid}}}.$$

Thus, the values returned to the adversary depend only on the linear combination $c_{\text{sid}} := a_{\text{sid}} + wb_{\text{sid}} \in \mathbb{Z}_p$. In this game, we instead sample $c_{\text{sid}} \leftarrow \mathbb{Z}_p$ uniformly at random and set $u_{\text{sid}} := g^{c_{\text{sid}}}$, $\hat{u}_{\text{sid}} := \hat{g}^{c_{\text{sid}}}$. Whenever $(a_{\text{sid}}, b_{\text{sid}})$ are later required (e.g., in PREIM or in the final check), the game samples $b_{\text{sid}} \leftarrow \mathbb{Z}_p$ and defines $a_{\text{sid}} := c_{\text{sid}} - wb_{\text{sid}}$. Since for uniform $(a_{\text{sid}}, b_{\text{sid}})$ the value $c_{\text{sid}} = a_{\text{sid}} + wb_{\text{sid}}$ is uniform in $\mathbb{Z}_p$, and conditioned on any fixed c_{sid} the reconstruction $(a_{\text{sid}}, b_{\text{sid}}) = (c_{\text{sid}} - wb_{\text{sid}}, b_{\text{sid}})$ with uniform b_{sid} is uniform over all pairs satisfying $a_{\text{sid}} + wb_{\text{sid}} = c_{\text{sid}}$, the joint distribution is unchanged. Therefore, we get $\Pr[\mathbf{G}_2 \Rightarrow 1] = \Pr[\mathbf{G}_1 \Rightarrow 1]$.

Unconditional bound for $\mathbf{G}_2$: Let x and y be the discrete logarithms of X and Y , respectively. Except with probability $1/p$, we have $y \neq wx$. When $\mathbf{A}$ outputs (i, U) , b_i is uniformly sampled, and the game checks whether $U = X^{a_i} Y^{b_i}$. We have

$$X^{a_i} Y^{b_i} = g^{x(c_i - wb_i) + y(b_i)} = g^{xc_i + (y - wx)b_i}.$$

Since $y \neq wx$, every possible value of b_i yields a distinct value of $X^{a_i} Y^{b_i}$. Therefore, $U = X^{a_i} Y^{b_i}$ with probability $1/p$. We have $\Pr[\mathbf{G}_2 \Rightarrow 1] \leq 2/p$, which concludes this proof. $\square$

References

- [AB21] Handan Kiliç Alper and Jeffrey Burdges. Two-round trip Schnorr multi-signatures via delinearized witnesses. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 157–188, Virtual Event, August 2021. Springer, Cham. 35
- [ABLS25] Ittai Abraham, Renas Bacho, Julian Loss, and Gilad Stern. Nearly quadratic asynchronous distributed key generation from recursive consensus. Cryptology ePrint Archive, Paper 2025/006, 2025. 34
- [AF04] Masayuki Abe and Serge Fehr. Adaptively secure feldman VSS and applications to universally-composable threshold cryptography. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 317–334. Springer, Berlin, Heidelberg, August 2004. 34
- [AFLT12] Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly-secure signatures from lossy identification schemes. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 572–590. Springer, Berlin, Heidelberg, April 2012. 35
- [BC26] Renas Bacho and Yanbo Chen. Earpicks: Tightly secure two-round multi and threshold signatures. In Joan Daemen and Emmanuel Thomé, editors, *Advances in Cryptology – EUROCRYPT 2026*, pages 154–183, Cham, 2026. Springer Nature Switzerland. 2, 3, 6, 8, 36
- [BCJ08] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 449–458. ACM Press, October 2008. 5, 34
- [BCL⁺25] Renas Bacho, Yanbo Chen, Julian Loss, Stefano Tessaro, and Chenzhi Zhu. Adaptively secure partially non-interactive threshold schnorr signatures in the AGM. Cryptology ePrint Archive, Paper 2025/1953, 2025. 3, 16, 34
- [BCP02] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Group Diffie-Hellman key exchange secure against dictionary attacks. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 497–514. Springer, Berlin, Heidelberg, December 2002. 8
- [BD21] Mihir Bellare and Wei Dai. Chain reductions for multi-signatures and the HBMS scheme. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 650–678. Springer, Cham, December 2021. 35
- [BDLR25a] Renas Bacho, Sourav Das, Julian Loss, and Ling Ren. Adaptively secure three-round threshold schnorr signatures from DDH. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 390–422. Springer, Cham, August 2025. 3, 34
- [BDLR25b] Renas Bacho, Sourav Das, Julian Loss, and Ling Ren. Glacius: Threshold schnorr signatures from DDH with full adaptive security. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part II*, volume 15602 of *LNCS*, pages 304–334. Springer, Cham, May 2025. 3, 34

- [BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 435–464. Springer, Cham, December 2018. [35](#)
- [BGC⁺25] Ruben Baecker, Paul Gerhart, Davide Li Calsi, Luigi Russo, Dominique Schröder, and Arkady Yerukhimovich. Fully adaptive FROST in the algebraic group model from falsifiable assumptions. Cryptology ePrint Archive, Paper 2025/1950, 2025. [3](#), [34](#)
- [BJ08] Ali Bagherzandi and Stanislaw Jarecki. Multisignatures using proofs of secret key possession, as secure as the Diffie-Hellman problem. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *SCN 08*, volume 5229 of *LNCS*, pages 218–235. Springer, Berlin, Heidelberg, September 2008. [35](#)
- [BJ10] Ali Bagherzandi and Stanislaw Jarecki. Identity-based aggregate and multi-signature schemes based on RSA. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 480–498. Springer, Berlin, Heidelberg, May 2010. [34](#)
- [BK25] Renas Bacho and Alireza Kavousi. SoK: Dlog-based distributed key generation. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *2025 IEEE Symposium on Security and Privacy*, pages 614–632. IEEE Computer Society Press, May 2025. [34](#)
- [BL22] Renas Bacho and Julian Loss. On the adaptive security of the threshold BLS signature scheme. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 193–207. ACM Press, November 2022. [34](#)
- [BLL⁺24] Renas Bacho, Christoph Lenzen, Julian Loss, Simon Ochsenreither, and Dimitrios Papachristoudis. Grandline: Adaptively secure dkg and randomness beacon with (log-)quadratic communication complexity. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS '24*, page 941–955, New York, NY, USA, 2024. Association for Computing Machinery. [34](#)
- [BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Berlin, Heidelberg, December 2001. [2](#)
- [BLSW24] Renas Bacho, Julian Loss, Gilad Stern, and Benedikt Wagner. HARTS: High-threshold, adaptively secure, and robust threshold Schnorr signatures. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 104–140. Springer, Singapore, December 2024. [34](#)
- [BLT⁺24] Renas Bacho, Julian Loss, Stefano Tessaro, Benedikt Wagner, and Chenzhi Zhu. Tinkle: Threshold signatures from DDH with full adaptive security. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 429–459. Springer, Cham, May 2024. [3](#), [4](#), [9](#), [34](#)
- [BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006. [34](#), [35](#)
- [BNN07] Mihir Bellare, Chanathip Namprempré, and Gregory Neven. Unrestricted aggregate signatures. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 411–422. Springer, Berlin, Heidelberg, July 2007. [35](#)

- [Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Berlin, Heidelberg, January 2003. [34](#)
- [BTT22] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 276–305. Springer, Cham, August 2022. [35](#)
- [BW24] Renas Bacho and Benedikt Wagner. Tightly secure non-interactive BLS multi-signatures. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 397–422. Springer, Singapore, December 2024. [35](#)
- [BW25] Renas Bacho and Benedikt Wagner. T-spoon: Tightly secure two-round multi-signatures with key aggregation. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 256–290. Springer, Cham, August 2025. [6](#), [7](#), [35](#)
- [BW26] Renas Bacho and Benedikt Wagner. Tightly secure threshold signatures over pairing-free groups. *IACR Communications in Cryptology*, 2(4), 2026. [2](#), [3](#), [34](#)
- [CGJ⁺99] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In Michael J. Wiener, editor, *CRYPTO’99*, volume 1666 of *LNCS*, pages 98–115. Springer, Berlin, Heidelberg, August 1999. [34](#)
- [Che05] Benoît Chevallier-Mames. An efficient CDH-based signature scheme with a tight security reduction. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 511–526. Springer, Berlin, Heidelberg, August 2005. [5](#)
- [Che25a] Yanbo Chen. Dazzle: Improved adaptive threshold signatures from DDH. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part III*, volume 15676 of *LNCS*, pages 233–261. Springer, Cham, May 2025. [2](#), [3](#), [4](#), [34](#)
- [Che25b] Yanbo Chen. Round-efficient adaptively secure threshold signatures with rewinding. *IACR Communications in Cryptology*, 2(2), 2025. [3](#), [16](#), [34](#), [35](#)
- [CKK⁺25] Elizabeth C. Crites, Jonathan Katz, Chelsea Komlo, Stefano Tessaro, and Chenzhi Zhu. On the adaptive security of FROST. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 480–511. Springer, Cham, August 2025. [3](#), [34](#)
- [CKM23] Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. Fully adaptive Schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 678–709. Springer, Cham, August 2023. [3](#), [34](#)
- [Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Berlin, Heidelberg, August 2000. [4](#), [36](#)
- [CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO’92*, volume 740 of *LNCS*, pages 89–105. Springer, Berlin, Heidelberg, August 1993. [2](#)

- [DEF⁺19] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1084–1101, 2019. 35
- [Des88] Yvo Desmedt. Society and group oriented cryptography: A new concept. In *Advances in Cryptology—CRYPTO’87: Proceedings 7*, pages 120–127. Springer, 1988. 1
- [DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *Advances in Cryptology - CRYPTO ’89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. Springer, 1989. 1
- [DR24] Sourav Das and Ling Ren. Adaptively secure BLS threshold signatures from DDH and co-CDH. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 251–284. Springer, Cham, August 2024. 34
- [EHK⁺13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Berlin, Heidelberg, August 2013. 8
- [FH21] Masayuki Fukumitsu and Shingo Hasegawa. A tightly secure ddh-based multisignature with public-key aggregation. *Int. J. Netw. Comput.*, 11(2):319–337, 2021. 35
- [FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018. 3
- [FMY99] Yair Frankel, Philip D. MacKenzie, and Moti Yung. Adaptively-secure distributed public-key systems. In *Proceedings of the 7th Annual European Symposium on Algorithms*, ESA ’99, page 4–27, Berlin, Heidelberg, 1999. Springer-Verlag. 34
- [FSZ22] Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang. Squirrel: Efficient synchronized multi-signatures from lattices. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1109–1123. ACM Press, November 2022. 35
- [GCRS25] Paul Gerhart, Davide Li Calsi, Luigi Russo, and Dominique Schröder. Fully-adaptive two-round threshold schnorr signatures from DDH. *Cryptology ePrint Archive*, Paper 2025/1478, 2025. 3, 34
- [GJKR99] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In Jacques Stern, editor, *EURO-CRYPT’99*, volume 1592 of *LNCS*, pages 295–310. Springer, Berlin, Heidelberg, May 1999. 34
- [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, January 2007. 34
- [GJKW07] Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the Diffie-Hellman problems. *Journal of Cryptology*, 20(4):493–514, October 2007. 6, 36
- [Ita83] Kazuharu Itakura. A public-key cryptosystem suitable for digital multisignature. *NEC research and development*, 71:1–8, 1983. 34

- [JL00] Stanislaw Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 221–242. Springer, Berlin, Heidelberg, May 2000. 34
- [KG20] Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson, Jr., and Colin O’Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 34–65. Springer, Cham, October 2020. 3, 35
- [KRT24] Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 459–491. Springer, Cham, August 2024. 3, 7, 9, 34
- [KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003*, pages 155–164. ACM Press, October 2003. 6, 35, 36
- [LJY14] Benoît Libert, Marc Joye, and Moti Yung. Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM PODC*, pages 303–312. ACM, July 2014. 34
- [LOS⁺06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 465–485. Springer, Berlin, Heidelberg, May / June 2006. 34
- [LP01] Anna Lysyanskaya and Chris Peikert. Adaptive security in the threshold setting: From cryptosystems to signature schemes. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 331–350. Springer, Berlin, Heidelberg, December 2001. 34
- [Mak22] Nikolaos Makriyannis. On the classic protocol for MPC schnorr signatures. Cryptology ePrint Archive, Paper 2022/1332, 2022. 3, 34
- [MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: Extended abstract. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 245–254. ACM Press, November 2001. 34
- [NRS21] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 189–221, Virtual Event, August 2021. Springer, Cham. 35
- [NRSW20] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. MuSig-DN: Schnorr multisignatures with verifiably deterministic nonces. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1717–1731. ACM Press, November 2020. 35
- [NRT25] Guilhem Niot, Michael Reichle, and Kaoru Takemure. Adaptively-secure three-round threshold schnorr from DL. Cryptology ePrint Archive, Paper 2025/1941, 2025. 3
- [PW23] Jiaxin Pan and Benedikt Wagner. Chopsticks: Fork-free two-round multi-signatures from non-interactive assumptions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 597–627. Springer, Cham, April 2023. 5, 6, 7, 8, 35

- [PW24] Jiaxin Pan and Benedikt Wagner. Toothpicks: More efficient fork-free two-round multi-signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 460–489. Springer, Cham, May 2024. [35](#)
- [RY07] Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multi-party signatures against rogue-key attacks. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 228–245. Springer, Berlin, Heidelberg, May 2007. [35](#)
- [Seu12] Yannick Seurin. On the exact security of Schnorr-type signatures in the random oracle model. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 554–571. Springer, Berlin, Heidelberg, April 2012. [34](#)
- [Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979. [2](#), [8](#)
- [TZ23] Stefano Tessaro and Chenzhi Zhu. Threshold and multi-signature schemes from linear hash functions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 628–658. Springer, Cham, April 2023. [3](#), [35](#)
- [WQL09] Zecheng Wang, Haifeng Qian, and Zhibin Li. Adaptively secure threshold signature scheme in the standard model. *Informatika, Lith. Acad. Sci.*, 20:591–612, 01 2009. [34](#)

A More Related Work

We discuss related work with an emphasis on adaptively secure threshold signatures and tightness in multi-party signature.

Adaptively Secure Threshold Signatures. The first adaptively secure threshold signature schemes were independently introduced in 1999 by Canetti et al. [CGJ⁺99] and Frankel et al. [FMY99]. Their constructions rely on the single-inconsistent player (SIP) technique in combination with secure erasures. Jarecki and Lysyanskaya [JL00] subsequently refined the SIP approach to eliminate the requirement for secure erasures, and related techniques were later employed in [LP01, AF04] to construct adaptively secure threshold signatures without erasures. Gennaro et al. [GJKR99, GJKR07] also proposed adaptively secure threshold signature schemes, albeit still relying on secure erasures. All of these early constructions require multiple rounds of interaction—often assuming the availability of a broadcast channel—are restricted to corruption thresholds $t < n/2$, and fail to achieve tight security reductions.

Threshold Schnorr signatures have recently received considerable attention, resulting in a number of adaptively secure constructions [Mak22, CKM23, KRT24, BLSW24, BDLR25b, BDLR25a, CKK⁺25, Che25b, GCRS25, BGC⁺25, BCL⁺25], including schemes tolerating a full corruption threshold $t < n$. Among these, [CKM23, BLSW24, CKK⁺25, BGC⁺25, BCL⁺25] provide tight security proofs, but rely on the AGM and the AOMDL assumption, with some additionally assuming LDVR. Other constructions achieve adaptive security under standard assumptions; however, they either require at least three rounds of communication or operate in non-standard models, additionally employing heavy zero-knowledge proofs to establish deterministic nonce derivation. Furthermore, all AGM-free proofs rely on the forking lemma, which inherently results in highly non-tight security reductions. Moreover, existing impossibility results for Schnorr-type signatures rule out tight security reductions in the random oracle model (ROM) from standard assumptions [Seu12].

More recent work constructs adaptively secure threshold signatures from standard assumptions [BLT⁺24, BW26, Che25a, Che25b]. Among these, [Che25a, Che25b] propose two-round signing protocols; however, [Che25a] incurs a security loss at least linear in the number of signing queries, while [Che25b] relies on the forking lemma and thus yields a highly non-tight reduction. In addition, [BW26, Che25a] present three-round constructions with tight security reductions. These schemes employ the adaptive partitioning technique in the distributed signing setting, which appears to inherently require an additional round of interaction. We discuss these constructions in greater detail in the technical overview.

In the pairing-based setting, Wang et al. [WQL09] proposed a non-interactive, adaptively secure threshold Waters signature inspired by [AF04]. Libert et al. [LJY14] subsequently introduced another non-interactive adaptively secure threshold signature producing BLS-like signatures. Bacho and Loss [BL22] established adaptive security for Boldyreva’s threshold BLS signature [Bol03], and more recently, Das and Ren [DR24] presented an adaptively secure threshold BLS signature based on standard assumptions. Among these works, only [BL22] achieves a tight security reduction, albeit under the AGM and the OMDL assumption. The remaining constructions incur a security loss linear in the number of hash or signing queries.

Other Multi-Party Signatures. Multi-signatures constitute a special case of threshold signatures with threshold $t = n - 1$, in which all n participants are required to engage in the signing protocol and each signer independently generates its own key pair. As a consequence, multi-signature schemes avoid the need for an interactive distributed key generation (DKG) protocol [BK25], which is a central component of general threshold signature constructions.¹¹ Since their introduction by Itakura and Nakamura [Ita83], multi-signatures have been studied extensively [MOR01, Bol03, BN06, LOS⁺06, BCJ08, BJ10], with renewed interest in recent years driven by

¹¹We note, however, that efficient DKG protocols with quadratic communication cost are known in both the synchronous [BLL⁺24] and asynchronous [ABLS25] network settings.

practical and theoretical advances [BDN18, DEF⁺19, NRSW20, NRS21, AB21, BD21, BTT22, FSZ22, TZ23]. Several works further focus on achieving tight security reductions in the multi-signature setting [BN06, RY07, BNN07, BJ08, FH21, PW23, PW24, BW24, BW25]. Among these, only the schemes in [PW23, PW24, BW25] admit two-round signing protocols in the pairing-free setting.

In contrast to threshold signatures, multi-signature schemes do not rely on a jointly generated public key representing all signers. Moreover, adaptive security in the multi-signature setting can be obtained from static security via a simple but non-tight¹² reduction. As a result, the design space and proof techniques for tightly and adaptively secure multi-signatures differ fundamentally from those required for threshold signatures. Existing techniques for obtaining tight security in multi-signature schemes include lossy identification-based approaches [AFLT12], constructions requiring multiple public keys per signer [PW23], and the Katz–Wang pseudorandom bit sampling technique [KW03]. Importantly, none of these methods appear to extend to the threshold setting in the presence of adaptive corruptions.

We briefly elaborate on the two-round constructions of [PW23, PW24, BW25]. The schemes in [PW23, PW24] are technically similar, with the latter providing an optimized instantiation. In these constructions, each signer possesses two public keys and pseudorandomly selects one of them on a per-message basis. This design crucially relies on the independence of the signers’ keys and induces an exponential number of possible public-key combinations—specifically, 2^n combinations for n signers. Consequently, it is unclear how such an approach could be adapted to threshold signatures, where all signers are represented by a single public key. Furthermore, multi-signature schemes typically assume the presence of a single honest signer and therefore avoid many of the difficulties inherent to adaptive corruptions. The most recent construction [BW25] addresses key aggregation by employing a different technique: for each message m , every signer pseudorandomly chooses whether to sign $(0, m)$ or $(1, m)$. This effectively partitions the set of signers into two subsets that independently produce aggregate signatures, which are then combined to form the final signature. Nevertheless, it remains unclear how this approach could be extended to threshold signatures, particularly in the presence of adaptive adversaries. We elaborate on this below in Section A.1. Furthermore, all three schemes critically rely on the lossy key technique, in which the public key is programmed to a lossy mode for which no corresponding secret key exists. This approach fundamentally fails in the presence of adaptive corruptions: upon corrupting a signer, the adversary would immediately detect the absence of a valid secret key. In summary, there are several obstacles along different dimensions that prevent extending these tight two-round multi-signatures to the tight, adaptive threshold setting.

A.1 Tight Two-Round Multi-Signatures

Tightly secure two-round schemes have been proposed in a highly related area, namely multi-signatures [PW23, PW24, BW25]. These are n -out-of- n signatures with dynamically formed groups of signers. Many multi-signature schemes have threshold counterparts with very similar construction (e.g., [NRS21] vs. [KG20], [BD21] vs. [Che25b]). Therefore, transforming a multi-signature scheme into a threshold scheme is naturally considered a promising approach. However, we explain why the techniques from [PW23, PW24, BW25] do not apply in the threshold setting.

In fact, [PW23, PW24] require the verifier to take a list of individual public keys for verification (i.e., they do not support key aggregation). This already prevents us from transforming them into threshold signature schemes, where a signature is verified under a single compact key. On the other hand, the scheme T-Spoon [BW25] indeed allows the verifier to take only a compact aggregated public key. We explain why it cannot be directly transformed into a threshold scheme.

The basic framework of T-Spoon [BW25], which is a non-tight scheme proposed in [PW23],

¹²Specifically, one can guess the single uncorrupted signer, incurring a security loss of $1/n$ when reducing adaptive security to static security. For general threshold signatures, such an approach would instead incur an exponential loss in t and n .

is as follows. A public key consists of four group elements (g, h, X, Y) . A functional public key consists of a DH tuple, while one can indistinguishably generate a lossy public key as a non-DH tuple. In the signing protocol, the message m is hashed to generate a message-specific commitment key ck for a dual-mode commitment scheme, which has a hiding mode and a binding mode that are indistinguishable. Signing is possible if the key is functional or ck is hiding (as it restores the HVZK simulator like in our construction). In contrast, it is unconditionally hard under a lossy key with a binding ck . In the security proof, we switch the public key to be lossy and partition the message space into a hiding partition (where each message is hashed to a hiding ck) and a binding partition (where each message is hashed to a binding ck). For a reduction to efficiently execute the security game, we need the adversary to request signatures only on messages in the hiding partition, while making a forgery for a message in the binding partition. Using Coron’s coin tossing technique [Cor00], we can get the basic scheme with security loss $\Theta(q_s)$, where q_s is the number of signing queries.

To achieve fully tight security, the basic framework is combined with the pseudorandom bit technique by Katz and Wang [KW03, GJKW07]. Now, each message is hashed to two commitment keys ck_0, ck_1 . In the signing protocol, each signer independently chooses a pseudorandom bit (as a function of message m) to decide which commitment key to use. In the security proof, the public key is again switched to be lossy, while for each message, one of $\{ck_0, ck_1\}$ is hiding and the other is binding, in a random order. In the n -out-of- n setting, the reduction only simulates one honest signer. The reduction can set the honest signer’s pseudorandom bit to choose the hiding commitment key. On the other hand, the adversary has to make a forgery for a message that the honest signer never signed, for which it does not know which key is hiding. It has a $1/2$ chance to choose the binding key, which makes signing unconditionally hard.

In the t -out-of- n setting, however, the reduction needs to simulate multiple honest signers whose public key shares are all lossy. Thus, they have to always choose the same bit rather than choosing each bit independently, making the simulation distinguishable from the real game.

Finally, we note that a recent work [BC26] proposes Earpicks-MS, a tightly secure two-round multi-signature scheme with key aggregation. The scheme also admits a tightly secure two-round threshold variant, Earpicks-TS, but only in the static-corruption setting. As explained by the authors in Section 2.4 (“Open Questions”), their proof of tight security for Earpicks-TS crucially relies on corruptions being static and breaks down at two points in the adaptive setting. We quote their explanation verbatim: “First, on behalf of an honest party $i \in [n]$, the reduction outputs a uniformly random group element Z_i , together with a simulated proof, instead of the real value $Z_i = H_D(\text{apk}, \mu)^{x_i}$, where $\text{apk} = \text{pk}$ denotes the group public key and x_i is party i ’s secret key share. In the adaptive-corruption setting, the adversary may subsequently corrupt an honest party i , at which point the reduction must reveal the corresponding secret key x_i . Once x_i is revealed, the adversary can recompute the correct value of Z_i and detect that it differs from the uniformly random element previously output by the reduction. Hence, the simulation becomes inconsistent, and the reduction fails. More generally, the pseudorandom-bit (or controllable branch-selection) technique employed in the proof does not appear to extend to the adaptive setting.

Second, the proof involves switching the honest public key $\text{pk} = (g, h)^x$ to a lossy key, i.e., a public key for which no corresponding secret key exists. Concretely, pk is replaced by a uniformly random tuple $(X, Y) \in \mathbb{G}^2$. In the threshold setting, the joint public key is derived from the n public key shares of the parties. If more than t shares are honestly generated, then the resulting joint key cannot be lossy. Consequently, at most t shares may be generated consistently with secret keys. However, under adaptive corruptions, the reduction must be prepared to reveal a valid secret key share for any party that is corrupted. This would require the reduction to guess in advance the subset of at most t parties that will be corrupted and generate only their shares honestly. Such a guessing strategy incurs a security loss proportional to $\binom{n}{t}$, which is exponential in the number of involved parties.”

B Proof of Lemma 2

In this section, we provide the proof of Lemma 2 (Effect of Masking).

Proof. We prove the lemma via a sequence of games.

Game $\mathbf{G}_0$ (Real Game): This is the real security game TSUF for threshold signatures as defined in Fig. 2. By definition, we have

$$\Pr[\mathbf{G}_0 \Rightarrow 1] = \varepsilon_{\text{uf}}.$$

Game $\mathbf{G}_1$ (Guessing Honest Seed): In this game, we introduce an abort condition to handle the unlikely event that the adversary $\mathbf{A}$ guesses a secret seed shared between two honest parties. Concretely, the game now aborts if $\mathbf{A}$ makes a query to the random oracle $\mathbf{H}_m$ of the form $\mathbf{H}_m(i, j, \text{seed}_{i,j}, \cdot)$ such that $(i, j) \in \mathcal{H} \times \mathcal{H}$. The only difference between $\mathbf{G}_1$ and the previous game is this additional abort event. Hence, the difference in the adversary's success probability is bounded by the probability that this abort occurs.

All seeds $\text{seed}_{i,j}$ are chosen independently and uniformly at random from $\{0, 1\}^\lambda$ during key generation and remain perfectly hidden from the adversary as long as both i and j are honest. Therefore, for any fixed pair $(i, j) \in \mathcal{H} \times \mathcal{H}$ and any single random oracle query, the probability that $\mathbf{A}$ correctly guesses $\text{seed}_{i,j}$ is exactly $1/2^\lambda$. By a union bound over all random oracle queries and all pairs of honest signers, we therefore get

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| \leq n^2 \cdot q_h / 2^\lambda.$$

Game $\mathbf{G}_{1.5}$ (Ruling Out First-Round Message Collisions): The game aborts if two distinct successful queries to the first-round signing oracle for the same triple $(i, \mathcal{S}, m)$ return the same first-round message $\text{msg}_{1,i} = (\text{com}_i, U_i)$. Recall

$$U_i = u^{\lambda_{\mathcal{S}, i x_i}} \hat{u}^{\lambda_{\mathcal{S}, i \hat{x}_i}} \Delta_{i,1}$$

is determined by $(i, \mathcal{S}, m)$. Thus, such a collision occurs if and only if the two commitments com_i are equal.

Consider a newly generated commitment $\text{com}_i = \text{com}(\text{ck}, \mathbf{R}_i; \rho_i)$, where $R_{1,i} = g^{r_i} \hat{g}^{\hat{r}_i}$. Conditioned on the preceding view, $\hat{r}_i$, and ρ_i , the first coordinate of com_i is a fixed group element multiplied by g^{r_i} . It is therefore uniform in $\mathbb{G}$, since $r_i \leftarrow \mathbb{Z}_p$ is fresh, and matches the first coordinate of any fixed earlier commitment with probability $1/p$. Since equality of the complete commitments requires equality of their first coordinates, a union bound gives

$$|\Pr[\mathbf{G}_{1.5} \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \frac{q_s^2}{p}.$$

Conditioned on no abort, the own-message check in the second-round signing oracle ensures that, for every signer i , each second-round tuple $(\mathcal{S}, m, \mathcal{M}_1)$ identifies at most one local session. Indeed, otherwise the first-round messages of two distinct local sessions would both equal the entry for i in $\mathcal{M}_1$, contradicting the absence of a collision. Consequently, the parties answering under a given session tuple can be recorded by their identities, and the same masks $\Delta_{i,2} \in \mathbb{Z}_p^4$ and $\Delta_{i,3} \in \mathbb{Z}_p^2$ are reused whenever signer i is invoked under the same tuple.

Game $\mathbf{G}_2$ (Lazy Evaluation of ZeroShare): In this game, we change how the signing oracle evaluates the procedure `ZeroShare` for honest signers inside `SIGN1` and `SIGN2`. Recall that upon a query `SIGN1( $\mathcal{S}, m, i$ )`, the first-round group mask is defined as

$$\Delta_{i,1} = \prod_{j \in \mathcal{S}} \mathbf{H}_m(i, j, \text{seed}_{i,j}, \mathcal{S}, m, 1) \cdot \mathbf{H}_m(j, i, \text{seed}_{j,i}, \mathcal{S}, m, 1)^{-1} \in \mathbb{G},$$

and the second-round masks $\Delta_{i,2} \in \mathbb{Z}_p^4$ and $\Delta_{i,3} := (\Delta_{i,3}, \hat{\Delta}_{i,3}) \in \mathbb{Z}_p^2$ are defined by the corresponding additive sums using $(\mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}}, \mathcal{M}_1)$. In the previous game, whenever the signing oracle executed $\text{ZeroShare}(\text{seed}_i, \mathcal{S}, m, 1)$ (or the second-round analogues), all underlying H_m values were evaluated independently according to the random oracle. In this game, we modify this as follows. Namely, whenever the signing oracle receives a query $\text{SIGN}_1(\mathcal{S}, m, i)$ or $\text{SIGN}_2(i, \text{sid}, \mathcal{M}_1)$ for the first time for an honest signer $i \in \mathcal{H}$, the game proceeds as follows:

- Upon a signing query $\text{SIGN}_1(\mathcal{S}, m, i)$, it computes the first-round group mask $\Delta_{i,1} = \Delta_1(i, \mathcal{S}, m)$ by evaluating the defining product above, using lazy sampling of H_m .
- Upon a signing query $\text{SIGN}_2(i, \text{sid}, \mathcal{M}_1)$, it computes the second-round mask values

$$\Delta_{i,2} = \Delta_2(i, \mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}}, \mathcal{M}_1) \quad \text{and} \quad \Delta_{i,3} = \Delta_3(i, \mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}}, \mathcal{M}_1)$$

by evaluating the defining sums, using lazy sampling of H_m .

- That is, whenever a term of the form $H_m(i, j, \text{seed}_{i,j}, \cdot)$ or $H_m(j, i, \text{seed}_{j,i}, \cdot)$ is required and has not been queried before, the game samples its value uniformly at random from its prescribed range: $\mathbb{G}$ for $\ell = 1$, $\mathbb{Z}_p^4$ for $\ell = 2$, and $\mathbb{Z}_p^2$ for $\ell = 3$, and records it in the random oracle table.
- The entire group mask $\Delta_{i,1}$, or second-round vector mask $\Delta_{i,\ell}$ for $\ell \in \{2, 3\}$, is thus determined at once, and stored for future consistency.

Only honest–honest seeded entries are reordered in this way. Entries incident to a corrupted signer remain ordinary random oracle entries and may already have been queried by the adversary; the signing oracle simply reuses them.

We claim that this change does not affect the adversary’s view. Indeed, conditioned on no abort in $\mathbf{G}_1$, the adversary never queries H_m on inputs containing $\text{seed}_{i,j}$ for $(i, j) \in \mathcal{H} \times \mathcal{H}$. Therefore, all honest–honest oracle evaluations occur only inside executions of ZeroShare by the signing oracle or are deferred as described below. Sampling them lazily at that point, instead of conceptually fixing them earlier, preserves their joint distribution exactly. Therefore, we get

$$\Pr[\mathbf{G}_2 \Rightarrow 1] = \Pr[\mathbf{G}_{1.5} \Rightarrow 1].$$

Game $\mathbf{G}_3$ (Sample Masks First): In this game, we change how the masks are generated for honest signers in a signing session. Recall that the masks satisfy

$$\prod_{j \in \mathcal{S}} \Delta_{j,1} = 1_{\mathbb{G}}, \quad \sum_{j \in \mathcal{S}_{i,\text{sid}}} \Delta_{j,\ell} = \mathbf{0} \quad \text{for } \ell \in \{2, 3\}.$$

In this game, we reverse the generation order of the H_m values and the resulting masks $\Delta_{i,1}, \Delta_{i,2}, \Delta_{i,3}$. Namely, instead of first sampling the underlying H_m values and then deriving the masks, we proceed as follows whenever a new signing query $\text{SIGN}_1(\mathcal{S}, m, i)$ or $\text{SIGN}_2(\text{sid}, i, \mathcal{M}_1)$ (for an honest $i \in \mathcal{H}$) is made:

- Let $H := \mathcal{H} \cap \mathcal{S}$ be the current set of honest signers for the relevant signer set (with $\mathcal{S} = \mathcal{S}_{i,\text{sid}}$ in round two).
- For the first-round tag $(\mathcal{S}, m)$, sample each honest mask that becomes exposed uniformly as long as another honest mask remains hidden. When only one honest mask remains to be fixed, call its signer $i^* \in H$ and set

$$\Delta_{i^*,1} := \left(\prod_{j \in \mathcal{S} \setminus \{i^*\}} \Delta_{j,1} \right)^{-1}.$$

A value already fixed for the same triple $(i, \mathcal{S}, m)$ is reused; in particular, repeated first-round queries never resample $\Delta_{i,1}$.

- For each second-round tag $(\mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}}, \mathcal{M}_1, \ell)$ with $\ell \in \{2, 3\}$, use the additive analogue: sample masks uniformly as long as another honest mask remains hidden and, when only one honest mask remains to be fixed, set it to

$$\Delta_{i^*, \ell} := - \sum_{j \in \mathcal{S}_{i,\text{sid}} \setminus \{i^*\}} \Delta_{j, \ell}.$$

After fixing all masks in this way, the game programs the random oracle H_m lazily and consistently so that the defining equations of **ZeroShare** are satisfied. For $\ell = 1$, the masks determine the corresponding quotient of products of group-valued oracle outputs; for $\ell \in \{2, 3\}$, they determine the corresponding difference of sums. Since the individual oracle outputs are uniform in their respective ranges, it is always possible to sample the H_m values consistently.

We claim that this modification does not change the distribution of the adversary's view. Indeed, in the previous game the honest first-round group masks are uniform subject to the single product constraint after the already fixed corrupted masks are taken into account. Likewise, for $\ell \in \{2, 3\}$, the honest masks are uniform subject to the corresponding additive constraint. Sampling all but one honest mask uniformly and defining the remaining one deterministically produces exactly these conditional distributions. Moreover, since the H_m values are uniform and independent, programming them after fixing the masks preserves their marginal distribution. Therefore, we get

$$\Pr[\mathbf{G}_3 \Rightarrow 1] = \Pr[\mathbf{G}_2 \Rightarrow 1].$$

Game $\mathbf{G}_4$ (Sample H_m Upon Corruption): In this game, we further modify how the random oracle H_m is implemented. Intuitively, each pairwise seed $\text{seed}_{i,j}$ is information-theoretically hidden from the adversary until either i or j is corrupted. Hence, except with the negligible probability captured by the abort event in $\mathbf{G}_1$, no oracle query issued by A can ever contain $\text{seed}_{i,j}$ while both i and j remain honest. Therefore, we may postpone sampling such H_m values until the first moment at which one of the two parties becomes corrupt.

Formally, the game maintains the standard random oracle table for H_m , but with the following change for *seeded* inputs. For any session $(\mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}})$ and any $\ell \in \{1, 2, 3\}$, whenever the signing oracle needs a value of the form $\mathbf{X}_{i,j}^{(\ell)} := H_m(\text{tag}_{i,j}^{(\ell)})$, where

$$\text{tag}_{i,j}^{(\ell)} = \begin{cases} (i, j, \text{seed}_{i,j}, \mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}}, 1), & \ell = 1, \\ (i, j, \text{seed}_{i,j}, \mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}}, \mathcal{M}_1, \ell), & \ell \in \{2, 3\}, \end{cases} ; \quad \mathbf{X}_{i,j}^{(\ell)} \in \begin{cases} \mathbb{G}, & \ell = 1, \\ \mathbb{Z}_p^4, & \ell = 2, \\ \mathbb{Z}_p^2, & \ell = 3, \end{cases}$$

the game proceeds as follows:

- If neither i nor j is corrupted at that time, the game does *not* sample or store $\mathbf{X}_{i,j}^{(\ell)}$ yet. Instead, it records a symbolic placeholder for this oracle input, to be instantiated later.
- If at least one of i or j is corrupted, then $\text{seed}_{i,j}$ is considered revealed. At this moment, if $\mathbf{X}_{i,j}^{(\ell)}$ has not yet been defined, the game samples

$$\mathbf{X}_{i,j}^{(\ell)} \leftarrow \begin{cases} \mathbb{G}, & \ell = 1, \\ \mathbb{Z}_p^4, & \ell = 2, \\ \mathbb{Z}_p^2, & \ell = 3, \end{cases}$$

uniformly at random (or, if the oracle value has already been implicitly fixed by the conditional programming step of $\mathbf{G}_3$, the game returns that fixed value). The sampled value is then stored in the random oracle table and used consistently thereafter.

We claim that this modification does not change the adversary's view. Conditioned on the non-abort in $\mathbf{G}_1$, the adversary cannot query H_m on any input containing $\text{seed}_{i,j}$ while both i and j are honest. Hence, until one of i or j is corrupted, the exact value of $H_m(i, j, \text{seed}_{i,j}, \cdot)$ is never revealed to $\mathbf{A}$ and has no externally observable effect beyond its contribution to the masks, whose distribution was already fixed in $\mathbf{G}_3$. Therefore, delaying the sampling of these oracle values until seed reveal is a pure reordering of lazy sampling and preserves the joint distribution of the adversary's view. Thus, we have

$$\Pr[\mathbf{G}_4 \Rightarrow 1] = \Pr[\mathbf{G}_3 \Rightarrow 1].$$

Game $\mathbf{G}_5$ (Last Mask Algebraically): In this game, we modify how a dependent honest mask $i^* \in \mathcal{H} \cap \mathcal{S}_{i,\text{sid}}$ is computed when it is the last honest mask to be fixed for a session $(\mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}})$. Recall that the first-round masks satisfy a product-one condition, whereas the two second-round masks satisfy additive zero-sum conditions. In this game, we compute the dependent mask using the same condition, but we rewrite each honest mask of every other honest signer $j \in \mathcal{H} \cap \mathcal{S}_{i,\text{sid}}$, $j \neq i^*$, using its defining relation with the corresponding *masked* protocol value. In more detail:

- *First-round masks* ($\ell = 1$). For every honest signer $j \neq i^*$, we have

$$U_j = u^{\lambda_{\mathcal{S},j} x_j} \hat{u}^{\lambda_{\mathcal{S},j} \hat{x}_j} \Delta_{j,1},$$

and hence

$$\Delta_{j,1} = U_j (u^{\lambda_{\mathcal{S},j} x_j} \hat{u}^{\lambda_{\mathcal{S},j} \hat{x}_j})^{-1}.$$

This is a direct group computation and requires no representation of U_j with respect to $(u, \hat{u})$. Equivalently, the dependent mask is defined by

$$\Delta_{i^*,1} = \left(\prod_{j \in \mathcal{S}_{i,\text{sid}} \setminus \{i^*\}} \Delta_{j,1} \right)^{-1},$$

where for honest $j \neq i^*$ the terms $\Delta_{j,1}$ are expressed via U_j as above, and the masks of corrupted signers are retained explicitly.

- *Second-round commitment-randomness masks* ($\ell = 2$). For every honest signer $j \neq i^*$, the masked commitment randomness is

$$\bar{\rho}_j = \rho_j + \Delta_{j,2}.$$

Hence,

$$\Delta_{j,2} = \bar{\rho}_j - \rho_j.$$

Thus, in the zero-sum equation for $\Delta_{i^*,2}$, we replace every *honest* $\Delta_{j,2}$ (for $j \in \mathcal{H} \cap \mathcal{S}_{i,\text{sid}}$ and $j \neq i^*$) by the algebraically equivalent difference $\bar{\rho}_j - \rho_j$. We keep the masks $\Delta_{j,2}$ for corrupted signers $j \in \mathcal{C} \cap \mathcal{S}_{i,\text{sid}}$ as they are.

Equivalently, using the identity

$$\Delta_{i^*,2} = - \sum_{j \in \mathcal{S}_{i,\text{sid}} \setminus \{i^*\}} \Delta_{j,2},$$

we can rewrite

$$\Delta_{i^*,2} = - \sum_{j \in (\mathcal{H} \cap \mathcal{S}_{i,\text{sid}}) \setminus \{i^*\}} (\bar{\rho}_j - \rho_j) - \sum_{j \in \mathcal{C} \cap \mathcal{S}_{i,\text{sid}}} \Delta_{j,2}.$$

- *Second-round response masks* ($\ell = 3$). For every honest signer $j \neq i^*$, the masked responses satisfy

$$z_j = r_j + c\lambda_{\mathcal{S},j} x_j + \Delta_{j,3}, \quad \hat{z}_j = \hat{r}_j + c\lambda_{\mathcal{S},j} \hat{x}_j + \hat{\Delta}_{j,3}.$$

Hence,

$$\Delta_{j,3} = z_j - (r_j + c\lambda_{\mathcal{S},j}x_j), \quad \hat{\Delta}_{j,3} = \hat{z}_j - (\hat{r}_j + c\lambda_{\mathcal{S},j}\hat{x}_j).$$

Thus, in the zero-sum equation for $\Delta_{i^*,3}$, we replace every *honest* $\Delta_{j,3}$ (for $j \in \mathcal{H} \cap \mathcal{S}_{i,\text{sid}}$ and $j \neq i^*$) by the algebraically equivalent difference between the masked and the plain response. That is, for such honest j , we use

$$\Delta_{j,3} = z_j - (r_j + c\lambda_{\mathcal{S},j}x_j), \quad \hat{\Delta}_{j,3} = \hat{z}_j - (\hat{r}_j + c\lambda_{\mathcal{S},j}\hat{x}_j).$$

We keep $\Delta_{j,3}$ for corrupted signers $j \in \mathcal{C} \cap \mathcal{S}_{i,\text{sid}}$ as they are. Equivalently, using the identity

$$\Delta_{i^*,3} = - \sum_{j \in \mathcal{S}_{i,\text{sid}} \setminus \{i^*\}} \Delta_{j,3},$$

we can rewrite

$$\begin{aligned} \Delta_{i^*,3} = & - \sum_{j \in (\mathcal{H} \cap \mathcal{S}_{i,\text{sid}}) \setminus \{i^*\}} \left((z_j, \hat{z}_j) - (r_j + c\lambda_{\mathcal{S},j}x_j, \hat{r}_j + c\lambda_{\mathcal{S},j}\hat{x}_j) \right) \\ & - \sum_{j \in \mathcal{C} \cap \mathcal{S}_{i,\text{sid}}} \Delta_{j,3}. \end{aligned}$$

This modification is purely algebraic: we rewrite the product-one equation for the first-round masks and the zero-sum equations for the second-round masks by substituting each honest mask (with $j \neq i^*$) by its defining masked-versus-plain form. Therefore, the joint distribution of all protocol messages is unchanged, and

$$\Pr[\mathbf{G}_5 \Rightarrow 1] = \Pr[\mathbf{G}_4 \Rightarrow 1].$$

Game $\mathbf{G}_6$ (Reverse Generation of Masks and Masked Values): In this game, we reverse the order in which masks and masked protocol values are generated inside SIGN for honest signers.

First round ($\ell = 1$). Fix a session $(\mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}})$ arising from queries to SIGN_1 . Let

$$\mathcal{S}_H := \mathcal{H} \cap \mathcal{S}_{i,\text{sid}}, \quad \mathcal{S}_C := \mathcal{S}_{i,\text{sid}} \setminus \mathcal{S}_H.$$

For every $j \in \mathcal{S}_{i,\text{sid}}$, write

$$P_j := u^{\lambda_{\mathcal{S},j}x_j} \hat{u}^{\lambda_{\mathcal{S},j}\hat{x}_j}.$$

Let $i^* \in \mathcal{S}_H$ denote the signer whose contribution is fixed last under the first-round product constraint, whether because of a signing query or a later corruption. For every honest signer $j \in \mathcal{S}_H$ with $j \neq i^*$ whose value is exposed first, we sample the masked value U_j uniformly from $\mathbb{G}$ and define

$$\Delta_{j,1} := U_j P_j^{-1}.$$

The value U_j is sampled only once for each triple $(j, \mathcal{S}, m)$ and is reused on repeated first-round queries, exactly as in the real protocol.

For every corrupted signer $j \in \mathcal{S}_C$, define its *virtual honest-form contribution*

$$V_{j,1} := P_j \Delta_{j,1}.$$

This virtual value is determined by the corrupted signer's share, seeds, and the programmed $\mathbf{H}_m$ table; it is not the arbitrary value that a corrupted signer may place in a first-round message list. Let

$$T_m := u^x \hat{u}^{\hat{x}}.$$

Since $\prod_{j \in \mathcal{S}} P_j = T_m$ and $\prod_{j \in \mathcal{S}} \Delta_{j,1} = 1_{\mathbb{G}}$, the real honest first-round outputs satisfy

$$\prod_{j \in \mathcal{S}_H} U_j = T_m \left(\prod_{j \in \mathcal{S}_C} V_{j,1} \right)^{-1}.$$

Accordingly, for the dependent honest signer i^* , we set

$$U_{i^*} := T_m \left(\prod_{j \in \mathcal{S}_C} V_{j,1} \right)^{-1} \left(\prod_{j \in \mathcal{S}_H \setminus \{i^*\}} U_j \right)^{-1}.$$

Equivalently, $\Delta_{i^*,1} = U_{i^*} P_{i^*}^{-1}$ is exactly the unique group mask that completes the product-one constraint. This computation uses only group operations. In particular, it neither requires the simulator to find scalar exponents for a dependent U_{i^*} nor omits the corrupted parties' virtual contributions.

Second round ($\ell = 2, 3$). Fix a second-round session $(\mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}}, \mathcal{M}_1)$ arising from queries to SIGN_2 . Let

$$\mathcal{S}_H := \mathcal{H} \cap \mathcal{S}_{i,\text{sid}}, \quad \mathcal{S}_C := \mathcal{S}_{i,\text{sid}} \setminus \mathcal{S}_H,$$

and let $i^* \in \mathcal{S}_H$ denote the signer whose mask is fixed last under the zero-sum constraint for this exact triple. This can occur on the final honest second-round query or when a corruption removes the final unanswered honest signer. Let

$$\begin{aligned} (\text{com}_j, U_j)_{j \in \mathcal{S}_{i,\text{sid}}} &:= \mathcal{M}_1, \quad \text{com} := \bigotimes_{j \in \mathcal{S}_{i,\text{sid}}} \text{com}_j, \quad U := \prod_{j \in \mathcal{S}_{i,\text{sid}}} U_j, \\ c &:= H_c(\text{pk}, \text{com}, U, m_{i,\text{sid}}). \end{aligned}$$

For every honest signer $j \in \mathcal{S}_H$ with $j \neq i^*$, we first sample the *masked* second-round values from their distributions:

$$\bar{\rho}_j \leftarrow \mathbb{Z}_p^4, \quad (z_j, \hat{z}_j) \leftarrow \mathbb{Z}_p^2,$$

and then define the corresponding masks as the “masked minus plain” differences

$$\Delta_{j,2} := \bar{\rho}_j - \rho_j,$$

$$(\Delta_{j,3}, \hat{\Delta}_{j,3}) := (z_j - (r_j + c\lambda_{\mathcal{S},j}x_j), \hat{z}_j - (\hat{r}_j + c\lambda_{\mathcal{S},j}\hat{x}_j)).$$

For corrupted signers $j \in \mathcal{S}_C$, we do not modify their protocol messages. Since all seeds are sampled at key generation and revealed upon corruption, the game can compute the corresponding masks $\Delta_{j,2}$ and $\Delta_{j,3}$ deterministically according to the definition of **ZeroShare** (equivalently, using the programmed H_m table entries). For the dependent signer i^* , we define the masked protocol values $\bar{\rho}_{i^*}$ and $(z_{i^*}, \hat{z}_{i^*})$ so that the zero-sum constraints

$$\sum_{j \in \mathcal{S}_{i,\text{sid}}} \Delta_{j,2} = \mathbf{0} \in \mathbb{Z}_p^4, \quad \sum_{j \in \mathcal{S}_{i,\text{sid}}} (\Delta_{j,3}, \hat{\Delta}_{j,3}) = \mathbf{0} \in \mathbb{Z}_p^2$$

hold for this session. Equivalently, solving these constraints yields the explicit equations

$$\bar{\rho}_{i^*} = \sum_{j \in \mathcal{S}_H} \rho_j - \sum_{j \in \mathcal{S}_H \setminus \{i^*\}} \bar{\rho}_j - \sum_{j \in \mathcal{S}_C} \Delta_{j,2},$$

and

$$(z_{i^*}, \hat{z}_{i^*}) = \left(\sum_{j \in \mathcal{S}_H} r_j + cx_H, \sum_{j \in \mathcal{S}_H} \hat{r}_j + c\hat{x}_H \right) - \sum_{j \in \mathcal{S}_H \setminus \{i^*\}} (z_j, \hat{z}_j) - \sum_{j \in \mathcal{S}_C} (\Delta_{j,3}, \hat{\Delta}_{j,3}).$$

where

$$x_H := \sum_{j \in \mathcal{S}_H} \lambda_{\mathcal{S},j} x_j, \quad \hat{x}_H := \sum_{j \in \mathcal{S}_H} \lambda_{\mathcal{S},j} \hat{x}_j.$$

Indistinguishability. We claim that the adversary's view in $\mathbf{G}_6$ is identically distributed to its view in $\mathbf{G}_5$. Fix any session $(\mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}})$ (for $\ell = 1$) and any second-round session $(\mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}}, \mathcal{M}_1)$ (for $\ell \in \{2, 3\}$), and let $\mathcal{S}_H = \mathcal{H} \cap \mathcal{S}_{i,\text{sid}}$, $\mathcal{S}_C = \mathcal{S}_{i,\text{sid}} \setminus \mathcal{S}_H$. In $\mathbf{G}_5$, for each $\ell \in \{1, 2, 3\}$, the masks of honest signers are uniform subject only to the corresponding cancellation constraint

$$\prod_{j \in \mathcal{S}_{i,\text{sid}}} \Delta_{j,1} = 1_{\mathbb{G}}, \quad \sum_{j \in \mathcal{S}_{i,\text{sid}}} \Delta_{j,2} = \mathbf{0}, \quad \sum_{j \in \mathcal{S}_{i,\text{sid}}} \Delta_{j,3} = \mathbf{0}.$$

Moreover, conditioned on these masks, the protocol messages of honest parties are deterministic functions of the masks and the honest “plain” values; e.g.,

$$U_j = u^{\lambda_{\mathcal{S},j} x_j} \hat{u}^{\lambda_{\mathcal{S},j} \hat{x}_j} \Delta_{j,1}, \quad \bar{\rho}_j = \rho_j + \Delta_{j,2}, \\ (z_j, \hat{z}_j) = (r_j, \hat{r}_j) + c\lambda_{\mathcal{S},j}(x_j, \hat{x}_j) + (\Delta_{j,3}, \hat{\Delta}_{j,3}).$$

In $\mathbf{G}_6$, we instead sample the masked protocol messages of all but one honest signer uniformly from their marginal distributions and then define the remaining honest masked message deterministically so that the corresponding cancellation constraint holds. This produces the same joint distribution by the following elementary fact: Let G be a finite abelian group and fix a target $d \in G$. A tuple uniform subject to having product d can be sampled by choosing all but one entry uniformly and defining the last by the product constraint. The additive version is identical with products replaced by sums. We apply the multiplicative statement in $\mathbb{G}$ for $\ell = 1$, and the additive statement in $\mathbb{Z}_p^4$ and $\mathbb{Z}_p^2$ for $\ell = 2$ and $\ell = 3$, respectively. The target is determined by the already fixed masks of corrupted signers. For $\ell = 2$ and $\ell = 3$, the masked values are obtained from the masks by adding fixed “plain” terms in the additive groups $\mathbb{Z}_p^4$ and $\mathbb{Z}_p^2$, respectively:

$$\bar{\rho}_j = \rho_j + \Delta_{j,2}, \\ (z_j, \hat{z}_j) = (r_j, \hat{r}_j) + c\lambda_{\mathcal{S},j}(x_j, \hat{x}_j) + (\Delta_{j,3}, \hat{\Delta}_{j,3}).$$

Since addition by a fixed element is a permutation of a finite abelian group, sampling the masks uniformly (subject to the zero-sum constraint) and then computing the masked values yields the same distribution as sampling the masked values uniformly and defining the masks as “masked minus plain.” For $\ell = 1$, multiplication by the fixed plain contribution $P_j = u^{\lambda_{\mathcal{S},j} x_j} \hat{u}^{\lambda_{\mathcal{S},j} \hat{x}_j}$ is a permutation of $\mathbb{G}$. Hence a uniform $\Delta_{j,1} \leftarrow \mathbb{G}$ makes $U_j = P_j \Delta_{j,1}$ uniform in $\mathbb{G}$, exactly as in $\mathbf{G}_6$, while the dependent value enforces the same product constraint including the corrupted virtual contributions. Hence, in all three cases $\ell \in \{1, 2, 3\}$, sampling masks first and then computing masked values is distributionally identical to sampling masked values first and defining the first-round group mask by division and the second-round masks by subtraction.

Finally, corrupted parties' messages are never modified in either game, and the corresponding corrupted masks are well-defined deterministically from the revealed seeds and the programmed $\mathbf{H}_m$ table. Therefore, for each session the joint distribution of all messages seen by $\mathbf{A}$ is unchanged, and hence

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

Game G_7 (Deferred Masks and Secret Shares): In this game, the honest shares, masks, and the as-yet-unexposed pairwise oracle values are sampled only when needed. For a first-round tag $\tau_1 = (\mathcal{S}, m)$ and an exact second-round tag $\tau_2 = (\mathcal{S}, m, \mathcal{M}_1)$, the game keeps one honest contribution undefined whenever the corresponding constraint has not yet become observable. While at least two signers in $\mathcal{S}$ remain honest, one unrevealed pairwise oracle value shared by two honest signers can be left undefined and later chosen so that the masks satisfy the required constraint. If a corruption leaves a unique honest signer, the game fixes the last such oracle value before releasing the newly corrupted signer's seeds. Thereafter, no oracle value shared by two honest signers remains hidden, but the corruption budget is exhausted, so the unique honest signer's local share and nonce are never exposed and can be chosen to satisfy the remaining protocol-output constraint. For every tag created thereafter, the oracle values involving corrupted signers determine the unique honest mask, while its protocol output is chosen to satisfy the corresponding aggregate constraint. When a query or corruption exposes the currently undefined honest contribution, the game leaves another honest contribution undefined whenever possible; if none remains, the newly exposed value is chosen as the unique value completing the product-one or zero-sum constraint.

This online conditional sampling is possible because every valid signer set has $|\mathcal{S}| \geq t+1$, whereas at most t signers are corrupted. The pairwise oracle values induce a uniform distribution over all mask vectors satisfying the product-one constraint for $\ell = 1$ and over all mask vectors satisfying the zero-sum constraint for $\ell \in \{2, 3\}$. When at least two honest signers remain, programming one not-yet-exposed pairwise oracle value shared by two honest signers gives exactly the real conditional distribution. In the one-honest-signer case, the last such oracle value was fixed before the preceding corruption was answered; for tags first created afterward, the cancellation constraint uniquely determines the honest mask from the oracle values involving corrupted signers, while the permanently hidden local contribution is chosen to satisfy the remaining protocol-output constraint. Different values of $\mathcal{M}_1$ give disjoint second-round oracle inputs and are handled independently.

When a returned first-round value of signer i later becomes exposed, the game samples its Shamir shares and defines

$$P_i := u^{\lambda_{\mathcal{S}, i} x_i} \hat{u}^{\lambda_{\mathcal{S}, i} \hat{x}_i}, \quad \Delta_{i,1} := U_i P_i^{-1}.$$

This uses only group operations. In contrast, the former scalar-valued first-round-mask proof would have had to find a representation of the already fixed group element U_i with respect to $(u, \hat{u})$. For an already returned second-round value, the game defines

$$\Delta_{i,2} := \bar{\rho}_i - \rho_i,$$

$$(\Delta_{i,3}, \hat{\Delta}_{i,3}) := (z_i, \hat{z}_i) - (r_i, \hat{r}_i) - c\lambda_{\mathcal{S}, i}(x_i, \hat{x}_i).$$

It then programs the hidden pairwise oracle entries, before their seeds are released, so that **ZeroShare** returns these masks.

The Shamir shares may also be deferred: any set of at most t shares of a degree- t sharing is uniform independently of the shared secret. Hence newly corrupted shares can be sampled lazily with the same joint distribution. Conditioned on the non-abort event of G_1 , the adversary has not queried an honest-honest seeded oracle input before the relevant seed is released. The change therefore preserves the joint distribution of all protocol messages and all state revealed by adaptive corruptions. Therefore,

$$\Pr[G_7 \Rightarrow 1] = \Pr[G_6 \Rightarrow 1].$$

Simulation of G_7 via the Ideal Game. We describe a reduction B that simulates the oracles of G_7 while interacting as an adversary in the ideal game.

Setup. The reduction receives (par, pk) from the ideal game, where

$$\text{pk} = (g^x \hat{g}^{\hat{x}}, h^x \hat{h}^{\hat{x}}).$$

It samples the pairwise seeds exactly as in real key generation, initializes empty tables for the lazily sampled Shamir shares, masks, first-round values, answered sets, and completed aggregates, and implements H_m using the online conditional sampling of $\mathbf{G}_7$. It forwards (par, pk) to $\mathbf{A}$ and keeps its own copy of the real-game session counters and round flags. These outer round flags are set to 2 after every valid external second-round query, whether or not its exact tag is complete.

Simulation of $\text{SIGN}_1(\mathcal{S}, m, i)$. Upon a query $\text{SIGN}_1(\mathcal{S}, m, i)$ for $i \in \mathcal{H}$, it does:

1. Forward the query to the ideal oracle $\text{SIGN}_1(\mathcal{S}, m, i)$ and obtain com_i . If this is the first honest query on m , also obtain and store $T_m = U(m) = u^x \hat{u}^{\hat{x}}$.
2. If $U_i[\mathcal{S}, m]$ was already fixed, reuse it. Otherwise use the online conditional sampler of $\mathbf{G}_7$. While another honest virtual contribution for this tag remains hidden, sample

$$U_i[\mathcal{S}, m] \leftarrow \mathbb{G}$$

uniformly.

3. If this is the last honest first-round value to be fixed under the product constraint, let $H = \mathcal{H} \cap \mathcal{S}$, $C = \mathcal{S} \setminus H$, and define it dependently as

$$U_i[\mathcal{S}, m] := T_m \left(\prod_{j \in C} V_{j,1} \right)^{-1} \left(\prod_{j \in H \setminus \{i\}} U_j[\mathcal{S}, m] \right)^{-1},$$

where $V_{j,1} = u^{\lambda_{\mathcal{S},j} x_j} \hat{u}^{\lambda_{\mathcal{S},j} \hat{x}_j} \Delta_{j,1}$ is the corrupted signer's virtual honest-form contribution. Actual values supplied by corrupted signers in $\mathcal{M}_1$ are not used in this equation.

4. Return $(\text{com}_i, U_i[\mathcal{S}, m])$ to $\mathbf{A}$.

Simulation of $\text{SIGN}_2(i, \text{sid}, \mathcal{M}_1)$. The reduction first performs the complete real-game validity check, including equality of signer i 's entire stored first-round message with $\mathcal{M}_1[i]$, and immediately consumes the outer session. Fix the exact tag $\tau = (\mathcal{S}_{i,\text{sid}}, m_{i,\text{sid}}, \mathcal{M}_1)$, let $H = \mathcal{H} \cap \mathcal{S}_{i,\text{sid}}$ and $C = \mathcal{S}_{i,\text{sid}} \setminus H$, and compute c from the actual first-round list $\mathcal{M}_1$. The reduction adds i to the local set A_τ , records this external call and its session identifier, but does not yet forward it to the ideal second-round oracle.

If $H \not\subseteq A_\tau$, the simulator samples

$$\bar{\rho}_i \leftarrow \mathbb{Z}_p^4, \quad (z_i, \hat{z}_i) \leftarrow \mathbb{Z}_p^2$$

uniformly, stores them under the exact tag, and returns them to $\mathbf{A}$.

If instead $H \subseteq A_\tau$, this is the first completion of the tag. Without giving control back to $\mathbf{A}$, the reduction forwards one stored ideal query $\text{SIGN}_2(j, \text{sidOf}[\tau, j], \mathcal{M}_1)$ for every $j \in H$, ordering the completing caller i last. Since no call for an incomplete tag was forwarded, the first $|H| - 1$ ideal calls return $\perp$ and the last returns

$$\rho^I = \sum_{j \in H} \rho_j, \quad z^I = \sum_{j \in H} r_j + cx, \quad \hat{z}^I = \sum_{j \in H} \hat{r}_j + c\hat{x}.$$

Define the corrupted offsets

$$x_C := \sum_{j \in C} \lambda_{\mathcal{S},j} x_j, \quad \hat{x}_C := \sum_{j \in C} \lambda_{\mathcal{S},j} \hat{x}_j,$$

$$\Delta_{C,2} := \sum_{j \in C} \Delta_{j,2}, \quad (\Delta_{C,3}, \hat{\Delta}_{C,3}) := \sum_{j \in C} (\Delta_{j,3}, \hat{\Delta}_{j,3}).$$

The honest masked aggregates in the real game are therefore

$$\begin{aligned}\bar{\rho}_H &= \rho^I - \Delta_{C,2}, \\ z_H &= z^I - cx_C - \Delta_{C,3}, \quad \hat{z}_H = \hat{z}^I - c\hat{x}_C - \hat{\Delta}_{C,3}.\end{aligned}$$

If $i = i^*$ is the completing honest signer, set

$$\begin{aligned}\bar{\rho}_{i^*} &:= \bar{\rho}_H - \sum_{j \in H \setminus \{i^*\}} \bar{\rho}_j, \\ (z_{i^*}, \hat{z}_{i^*}) &:= (z_H, \hat{z}_H) - \sum_{j \in H \setminus \{i^*\}} (z_j, \hat{z}_j),\end{aligned}$$

and return $(\bar{\rho}_{i^*}, z_{i^*}, \hat{z}_{i^*})$. The aggregate and the current set H are cached under the exact tag τ , and $\text{Done}[\tau]$ is set. Thus the dependent response removes the corrupted parties' virtual mask and key contributions before splitting the ideal aggregate among the honest signers.

Simulation of CORR(i). If $i \in \mathcal{H}$, sample $(x_i, \hat{x}_i) \leftarrow \mathbb{Z}_p^2$ lazily and forward $\text{CORR}(i)$ to the ideal oracle, obtaining $(r_{i,\text{sid}}, \hat{r}_{i,\text{sid}}, \rho_{i,\text{sid}})_{\text{sid} \in [\text{ctr}_i]}$. The ideal corruption updates $\mathcal{H}$, but the reduction delays its answer to $\mathbf{A}$ while it programs every mask that is about to become visible.

For every active first-round tag $(\mathcal{S}, m)$ containing i , the reduction programs i 's virtual group mask. If i previously returned a first-round value, it defines

$$\Delta_{i,1} := U_i[\mathcal{S}, m] (u^{\lambda_{\mathcal{S},i} x_i} \hat{u}^{\lambda_{\mathcal{S},i} \hat{x}_i})^{-1}.$$

If i did not return a value for this tag, its virtual contribution is sampled according to the same conditional distribution subject to the product-one constraint. If this corruption leaves no other honest contribution undefined, so that all values U_j for $j \in H$ have already been fixed, let $C^- = (\mathcal{C} \cap \mathcal{S}) \setminus \{i\}$ be the signers in $\mathcal{S}$ that were already corrupt and set

$$V_{i,1} := T_m \left(\prod_{j \in H} U_j \right)^{-1} \left(\prod_{j \in C^-} V_{j,1} \right)^{-1}, \quad \Delta_{i,1} := V_{i,1} P_i^{-1},$$

where $H = \mathcal{H} \cap \mathcal{S}$ is the new honest set and $P_i = u^{\lambda_{\mathcal{S},i} x_i} \hat{u}^{\lambda_{\mathcal{S},i} \hat{x}_i}$.

For every second-round tag on which i already returned a response, define

$$\begin{aligned}\Delta_{i,2} &:= \bar{\rho}_i - \rho_i, \\ \Delta_{i,3} &:= z_i - r_i - c\lambda_{\mathcal{S},i} x_i, \quad \hat{\Delta}_{i,3} := \hat{z}_i - \hat{r}_i - c\lambda_{\mathcal{S},i} \hat{x}_i.\end{aligned}$$

If an aggregate for that tag was already cached while i was honest, update its current-honest form by subtracting $(\rho_i, r_i, \hat{r}_i)$ from the cached tuple.

It remains to handle a tuple that becomes complete because of this corruption. For each exact pending tag $\tau = (\mathcal{S}, m, \mathcal{M}_1)$ with $\text{Done}[\tau] = 0$ for which the new honest set $H = \mathcal{H} \cap \mathcal{S}$ satisfies $H \subseteq A_\tau$, the reduction atomically forwards one stored ideal second-round query for every $j \in H$. None of these calls was forwarded while the tag was incomplete; the first $|H| - 1$ calls therefore return $\perp$, and the last returns

$$\rho^I = \sum_{j \in H} \rho_j, \quad z^I = \sum_{j \in H} r_j + cx, \quad \hat{z}^I = \sum_{j \in H} \hat{r}_j + c\hat{x}.$$

Cache this aggregate under τ and set $\text{Done}[\tau] := 1$. Let

$$C^- := (\mathcal{C} \cap \mathcal{S}) \setminus \{i\}$$

be the parties in $\mathcal{S}$ that were corrupt immediately before the corruption of i , and let

$$\bar{\rho}_H = \sum_{j \in H} \bar{\rho}_j, \quad z_H = \sum_{j \in H} z_j, \quad \hat{z}_H = \sum_{j \in H} \hat{z}_j$$

be the already returned honest sums. Notice that $i \notin A_\tau$ for a tag newly completed by this corruption; otherwise the tag was already complete before the corruption. Thus no response of i for this tag has been returned. The newly exposed masks of i are programmed as

$$\begin{aligned} \Delta_{i,2} &:= \rho^I - \bar{\rho}_H - \sum_{j \in C^-} \Delta_{j,2}, \\ \Delta_{i,3} &:= z^I - z_H - c \left(\lambda_{\mathcal{S},i} x_i + \sum_{j \in C^-} \lambda_{\mathcal{S},j} x_j \right) - \sum_{j \in C^-} \Delta_{j,3}, \\ \hat{\Delta}_{i,3} &:= \hat{z}^I - \hat{z}_H - c \left(\lambda_{\mathcal{S},i} \hat{x}_i + \sum_{j \in C^-} \lambda_{\mathcal{S},j} \hat{x}_j \right) - \sum_{j \in C^-} \hat{\Delta}_{j,3}. \end{aligned}$$

Together with the first-round programming above, these equations enforce precisely the product-one and zero-sum constraints with the newly corrupted party as the dependent contribution. A single corruption may complete several different $\mathcal{M}_1$ tags; the reduction performs the corresponding programming separately for each such tag. For a tag that remains incomplete, it samples i 's newly exposed masks conditionally and leaves the protocol output of a still-unanswered honest signer undefined. When two or more honest signers remain, one not-yet-exposed pairwise oracle value shared by two honest signers also remains undefined and can later be programmed to satisfy the mask constraint.

Finally, the reduction programs all remaining incident H_m entries before releasing i 's seeds, and returns the complete real-game state

$$\text{sk}_i = (x_i, \hat{x}_i, \text{seed}_i) \quad \text{and} \quad (\text{st}_{i,\text{sid}})_{\text{sid} \in [\text{ctr}_i]}.$$

Correctness of the simulation. For every first-round tag, the simulator samples the exposed masked outputs uniformly subject to the single product constraint, including all corrupted virtual contributions. For every exact second-round tag, it samples outputs uniformly while keeping one unanswered honest output undefined. When the tag becomes complete, either through a second-round query or through a corruption, it completes the stored ideal-oracle calls and enforces the aggregate equations after subtracting all corrupted virtual mask and key contributions. On corruption, every returned first-round value is explained by group division and every returned second-round value by scalar differences. These are exactly the real conditional distributions described in $\mathbf{G}_7$. Therefore, the view of $\mathbf{A}$ under this simulation is identical to its view in $\mathbf{G}_7$, and

$$\Pr[\mathbf{G}_7 \Rightarrow 1] = \varepsilon_{\text{ideal}}.$$

□

Game Ideal^A(1^λ, n, t):

```

1:  $\mathcal{H} := [n], \mathcal{C} := \emptyset, \mathcal{Q} := \emptyset$ 
2:  $\text{flag}[\cdot] := 0, \text{LastHon}[\cdot] := \emptyset$ 
3: for  $i \in [n]$  do
4:    $\text{ctr}_i := 0$ 
5:    $(\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda)$ 
6:    $\hat{g}, h, \hat{h} \leftarrow \mathbb{G}$ 
7:    $\text{ck} \leftarrow \mathbb{G}^{4 \times 4}$ 
8:    $\text{par} := (\mathbb{G}, p, g, \hat{g}, h, \hat{h}, \text{ck})$ 
    $\triangleright$  All oracles take par as input
9:    $x, \hat{x} \leftarrow \mathbb{Z}_p$ 
10:   $\text{pk} := (g^x \hat{g}^{\hat{x}}, h^x \hat{h}^{\hat{x}})$ 
11:   $\text{SIGN} := (\text{SIGN}_1, \text{SIGN}_2)$ 
12:   $(m_*, \sigma_*) \leftarrow A^{\text{SIGN}, \text{CORR}}(\text{par}, \text{pk})$ 
13:  if  $m_* \in \mathcal{Q}$  then
14:    return 0
15:  return  $\text{Ver}(\text{pk}, m_*, \sigma_*)$ 

```

$\triangleright$ First-round signing oracle

Oracle SIGN₁($\mathcal{S}, m, i$):

```

16: if  $(i \notin \mathcal{S} \cap \mathcal{H}) \vee (|\mathcal{S}| < t + 1)$  then
17:   return  $\perp$ 
18:  $\text{ctr}_i := \text{ctr}_i + 1$ 
19:  $\text{sid} := \text{ctr}_i$ 
20:  $(\mathcal{S}_{i, \text{sid}}, m_{i, \text{sid}}) := (\mathcal{S}, m)$ 
21:  $(u, \hat{u}) := H_u(m)$ 
22:  $r_i, \hat{r}_i \leftarrow \mathbb{Z}_p, \rho_i \leftarrow \mathbb{Z}_p^4$ 
23:  $R_{1,i} := g^{r_i} \hat{g}^{\hat{r}_i}, R_{2,i} := h^{r_i} \hat{h}^{\hat{r}_i}$ 
24:  $R_{3,i} := u^{r_i} \hat{u}^{\hat{r}_i}$ 
25:  $\mathbf{R} := (R_1, R_2, R_3)$ 
26:  $\text{com}_i := \text{Com}(\text{ck}, \mathbf{R}; \rho_i)$ 
27:  $\text{com}_{i, \text{sid}} := \text{com}_i$ 
28:  $\text{st}_{i, \text{sid}} := (\text{pk}, \mathcal{S}, m, i, r_i, \hat{r}_i, \rho_i)$ 
29:  $\text{round}_{i, \text{sid}} := 1$ 
30:  $\mathcal{Q} := \mathcal{Q} \cup \{m_{i, \text{sid}}\}$ 
31: if  $\text{flag}[m] = 0$  then
32:    $\text{flag}[m] := 1$ 

```

```

33:    $U := u^x \hat{u}^{\hat{x}}$ 
34:   return  $(\text{com}_i, U)$ 
35: return  $\text{com}_i$ 

```

$\triangleright$ Second-round signing oracle

Oracle SIGN₂($i, \text{sid}, \mathcal{M}_1$):

```

36:  $(\text{com}'_i, U'_i) := \mathcal{M}_1[i]$ 
37: if  $(i \notin \mathcal{H}) \vee (\text{round}_{i, \text{sid}} \neq 1) \vee (\text{com}_{i, \text{sid}} \neq \text{com}'_i)$  then
38:   return  $\perp$ 
39:  $(\text{pk}, \mathcal{S}, m, i, r_i, \hat{r}_i, \rho_i) := \text{st}_{i, \text{sid}}$ 
40:  $(\mathcal{S}_{i, \text{sid}}, m_{i, \text{sid}}) := (\mathcal{S}, m)$ 
41:  $\mathcal{H}_{i, \text{sid}} := \mathcal{H} \cap \mathcal{S}_{i, \text{sid}}$ 
42:  $\text{sinf} := (\mathcal{S}_{i, \text{sid}}, m_{i, \text{sid}}, \mathcal{M}_1)$ 
43:  $\text{LastHon}[\text{sinf}] := \text{LastHon}[\text{sinf}] \cup \{i\}$ 
44:  $\text{sidOf}[\text{sinf}, i] := \text{sid}$ 
45: if  $\text{LastHon}[\text{sinf}] \neq \mathcal{H}_{i, \text{sid}}$  then
46:   return  $\perp$ 
47:  $\text{round}_{i, \text{sid}} := 2$ 
48:  $(\text{com}_j, U_j)_{j \in \mathcal{S}_{i, \text{sid}}} := \mathcal{M}_1$ 
49:  $\text{com} := \bigotimes_{j \in \mathcal{S}_{i, \text{sid}}} \text{com}_j$ 
50:  $U := \prod_{j \in \mathcal{S}_{i, \text{sid}}} U_j$ 
51:  $c := H_c(\text{pk}, \text{com}, U, m_{i, \text{sid}})$ 
52: for  $j \in \mathcal{H}_{i, \text{sid}}$  do
53:    $(\text{pk}, \mathcal{S}, m, j, r_j, \hat{r}_j, \rho_j) := \text{st}_{j, \text{sidOf}[\text{sinf}, j]}$ 
54:    $r := \sum_{j \in \mathcal{H}_{i, \text{sid}}} r_j$ 
55:    $\hat{r} := \sum_{j \in \mathcal{H}_{i, \text{sid}}} \hat{r}_j$ 
56:    $z := r + c\mathcal{X}$ 
57:    $\hat{z} := \hat{r} + c\hat{\mathcal{X}}$ 
58:    $\rho := \sum_{j \in \mathcal{H}_{i, \text{sid}}} \rho_j$ 
59: return  $(\rho, z, \hat{z})$ 

```

$\triangleright$ Corruption oracle

Oracle CORR(i):

```

60: if  $(i \notin \mathcal{H}) \vee (|\mathcal{C}| \geq t)$  then
61:   return  $\perp$ 
62:  $\mathcal{C} := \mathcal{C} \cup \{i\}, \mathcal{H} := \mathcal{H} \setminus \{i\}$ 
63: return  $(r_{i, \text{sid}}, \hat{r}_{i, \text{sid}}, \rho_{i, \text{sid}})_{\text{sid} \in [\text{ctr}_i]}$ 

```

Figure 6: The ideal security game for TZAR with adversary A (Section 4.3).