Adaptively Secure Three-Round Threshold Schnorr Signatures from DDH

Renas Bacho[1,2], Sourav Das[3], Julian Loss[1,2], and Ling Ren[3]

[1] CISPA Helmholtz Center for Information Security
[2] Saarland University, Saarbrücken, Germany
[3] University of Illinois at Urbana-Champaign
{renas.bacho,loss}@cispa.de, {souravd2,renling}@illinois.edu

Abstract. Threshold signatures are one of the most important cryptographic primitives in distributed systems. Of particular interest is the threshold Schnorr signature, a pairing-free signature with efficient verification, compatible with standardized EdDSA (non-threshold) signature. However, most threshold Schnorr signatures have only been proven secure against a static adversary, which has to declare its corruptions before the protocol execution. Many existing adaptively secure constructions require either secure erasures or non-standard assumptions, such as the algebraic group model or hardness of the algebraic one-more discrete logarithm problem. The latest adaptively secure threshold Schnorr signature schemes under standard assumptions require five rounds of communication to create a single signature, limiting its practicality.

In this work, we present Gargos, a three-round, adaptively secure threshold Schnorr signature scheme based on the hardness of the decisional Diffie-Hellman (DDH) problem in the random oracle model (ROM). Our protocol supports full corruption threshold $t < n$, where t is the signing threshold and n is the total number of signers. We achieve our result with an enhanced proof technique that enables us to eliminate two rounds of communication from the recent Glacius scheme (Eurocrypt 2025). We believe our techniques are of independent interest to further research in improving the round complexity of multi-party signing protocols.

1 Introduction

Threshold signatures [Des88, DF89] are a type of interactive digital signature in which the signing key is distributed among a set of n signers. Importantly, any subset of at least $t+1$ signers can jointly generate a valid signature, while any t or less signers cannot. Threshold signatures have gained considerable interest in recent years, mainly for their applications in blockchain systems. A popular choice for a threshold signature are threshold variants of the Schnorr signature, a pairing-free signature with efficient verification. The Schnorr signature scheme has been standardized by NIST as the EdDSA signature and is also a focus of NIST's recent call for threshold cryptography [BP23]. Further, mainstream cryptocurrencies like Bitcoin [Nak08] adopt Schnorr signatures. In this work, we focus on threshold Schnorr signatures.

Static vs. adaptive security. The security model of threshold signatures generally considers two types of adversaries: *static* and *adaptive*. A static adversary has to decide which signers to corrupt (thereby learning their secret key shares and states) before the protocol execution. In contrast, an adaptive adversary can decide which signers to corrupt dynamically, based on previous signatures and corruptions. Adaptive security is a safer and more realistic model for the decentralized setting and is highly demanded.

Unfortunately, despite its popularity, all existing adaptively secure threshold Schnorr signature schemes have significant drawbacks. These include having signers to erase their internal states [CGJ+99, Mak22], requiring many rounds of communication [BDLR24, KRT24], relying on strong and non-standard assumptions such as algebraic one-more discrete logarithm (AOMDL) and the algebraic group model (AGM) [CKM23, BLSW24], or having suboptimal corruption thresholds [CKM23].

Our contribution. Motivated by this state of affairs, we present Gargos, the *first* threshold Schnorr signature scheme that combines all of the following desirable characteristics:

Table 1: Comparison of adaptively secure threshold Schnorr signature schemes. We do not consider schemes that assume a broadcast channel or the algebraic group model. We compare the number of signing rounds, the corruption threshold, the size of signing keys (as number of field elements), reliance on secure erasures, communication cost per signer (as number of field and group elements), and computational assumption.

Scheme	Rounds	Corruption threshold	Signing key size (in $\#\mathbb{Z}_p$)	No erasures?	Communication	Computational assumption
ZeroS [Mak22]‡	3	t	1	✗	$3\|\mathbb{Z}_p\|+\|\mathbb{G}\|$	DL
Sparkle [CKM23]	3	$t/2$	1	✓	$2\|\mathbb{Z}_p\|+\|\mathbb{G}\|$	AOMDL
KRT [KRT24]	5	t	$n+2$	✓	$4\|\mathbb{Z}_p\|+2\|\mathbb{G}\|$	DL
Glacius [BDLR24]	5	t	3	✓	$4\|\mathbb{Z}_p\|+\|\mathbb{G}\|$	DDH
Gargos (ours)	3	t	3	✓	$7\|\mathbb{Z}_p\|+2\|\mathbb{G}\|$	DDH

‡ The scheme additionally assumes private channels.

- *Adaptive security.* Gargos is provably secure under $t < n$ adaptive corruptions.
- *Standard assumptions.* Security is proven under the well-studied decisional Diffie-Hellman (DDH) assumption.
- *No erasures.* Gargos does not rely on secure erasures for its adaptive security proof.
- *Three-round signing.* Gargos has a signing protocol consisting of three communication rounds. In contrast, the recent adaptively secure schemes by Katsumata et al. [KRT24] and Bacho et al. [BDLR24] require five rounds of communication to generate a single signature.

We present detailed comparison of our scheme with existing adaptively secure threshold Schnorr signatures in Table 1.

In terms of techniques, we modify the five-round Glacius [BDLR24] to have three rounds of communication in signing. Our modification crucially relies on an enhanced proof technique and simple Σ-protocol based non-interactive zero-knowledge (NIZK) proofs. More precisely, we achieve our result as follows:

1. We combine the first and second rounds of Glacius, and add a simple NIZK proof to preserve security.
2. We remove the round of communication in Glacius where signers check the consistency of messages sent by others, and use a proof technique that defines equivalence classes on the views of signers.

We elaborate on this in the technical overview, see Section 2.

1.1 Related Work

We discuss more related work, also other adaptively secure threshold signatures.

Threshold Schnorr signatures. Early works [GJKR07, AF04, SS01, CGJ+99] primarily focused on robust threshold Schnorr schemes, ensuring that a signing session always produces a valid signature even in the presence of malicious signers. To this end, each signing session involves running an interactive protocol similar to a distributed key generation (DKG), which requires a broadcast channel and introduces high inefficiencies in round and communication cost. Notably, the schemes in [GJKR07, CGJ+99] achieve adaptive security but heavily rely on secure erasures. In contrast, the scheme by Abe and Fehr [AF04] eliminates the need for secure erasures but it requires suboptimal signing threshold $t < n/2$ and still a broadcast channel.

In recent years, the interest in more efficient threshold Schnorr signatures has been revived, mainly due to their applications in cryptocurrencies. This has sparked an extensive amount of works on efficient threshold Schnorr signatures with fewer rounds recently [KG21, BCK+22, CKM21, CGRS23, RRJ+22, BTZ22, Lin22, Mak22, CKM23, KRT24, BDLR24]. Some of these schemes [Mak22, CKM23, KRT24, BDLR24]

achieve adaptive security; however, each has its own drawbacks, such as requiring many rounds of communication, relying on secure erasures, or relying on strong and non-standard assumptions (see Table 1 for a comparison). Recently, several works have focused on achieving robustness in asynchronous networks [RRJ+22, GS24, BHK+24, BLSW24], where only [BLSW24] achieves adaptive security but has suboptimal corruption threshold $t/2$ and relies on the AGM, OMDL, secure erasures, and broadcast channels.

Adaptively secure threshold signatures. Canetti et al. [CGJ+99] and Frankel et al. [FMY99a, FMY99b] independently proposed the first adaptively secure threshold signatures using the *single-inconsistent player (SIP)* technique and secure erasures. Subsequently, Jarecki and Lysyanskaya [JL00] enhanced the SIP technique to avoid the use of secure erasures. Similar techniques were used in [LP01, AF04] to obtain adaptively secure threshold signatures without secure erasures. Remarkably, the scheme by Abe and Fehr [AF04] is a threshold Schnorr signature. Further works [ADN06, WQL09] also use the SIP technique to obtain threshold RSA signatures [Rab98] and threshold Waters signatures [Wat05].

In the pairing-based setting, Libert et al. [LJY14] presented a non-interactive, adaptively secure threshold signature scheme from DDH. Also, Bacho and Loss [BL22] proved adaptive security of Bolyreva's threshold BLS signature scheme [Bol03] from OMDL in the AGM. Notably, Das and Ren [DR24] proposed an adaptively secure threshold BLS signature from DDH and CDH in asymmetric pairing groups. Recently, Mitrokotsa et al. [MMS+24] also proposed a scheme with adaptive security from the bilinear DDH assumption in asymmetric groups. In the pairing-free setting, two recent works [BLT+24, BW24] give adaptively secure threshold signatures from DDH, where the latter achieves tight security.

2 Technical Overview

We start with a brief recap of the Schnorr signature scheme [Sch90]. For this, let $(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda)$ be a group generation algorithm, where $\mathbb{G}$ is a cyclic group of prime order p with generator $g \in \mathbb{G}$, and λ is the security parameter. Let $\mathsf{H}_{\mathsf{sig}} : \mathbb{G}^2 \times \mathcal{M} \to \mathbb{Z}_p$ be a hash function (modeled as a random oracle), where $\mathcal{M}$ is a message space. The signing key $\mathsf{sk} := s \in \mathbb{Z}_p$ is a random field element, and $\mathsf{pk} := g^s \in \mathbb{G}$ is the corresponding public verification key. The signature σ on a message m has the form $(\hat{A}, z) \in \mathbb{G} \times \mathbb{Z}_p$. To verify a signature $\sigma = (\hat{A}, z)$ on a message m, one first computes $c := \mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m)$ and checks that $g^z = \hat{A} \cdot \mathsf{pk}^c$.

2.1 Glacius: Five-Round Threshold Schnorr Signature [BDLR24]

Our starting point is Glacius [BDLR24], a recent five-round threshold Schnorr signature scheme with adaptive security. In the following, we review the main ideas behind its design and security proof.

Glacius adopts the commit-and-reveal approach for designing secure interactive signing protocols. In this approach, signers first commit to a random nonce using one round of communication. Next, signers reveal their committed value using another round of communication. Finally, signers compute and send their partial signatures using another round of communication. The commit-and-reveal approach naturally adopts three rounds of communication. The two additional rounds needed in Glacius are an artifact of their security proof against an adaptive adversary, which we describe next.

In Glacius, the secret signing key of signer i is $\mathsf{sk}_i := (s(i), r(i), u(i)) \in \mathbb{Z}_p^3$, where $s(x), r(x), u(x) \leftarrow\!\!\$\ \mathbb{Z}_p[x]$ are three uniformly random degree-t polynomials such that $r(0) = u(0) = 0$. The public key of the system is then $\mathsf{pk} := g^{s(0)} h^{r(0)} v^{u(0)} = g^{s(0)}$, where $g, h, v \in \mathbb{G}$ are three uniformly random and independent generators of $\mathbb{G}$. Further, each signer i also has its own public verification key $\mathsf{pk}_i = g^{s(i)} h^{r(i)} v^{u(i)}$ (where any $t+1$ out of those n verification keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$ interpolate to the joint public key pk).

We now describe the five-round signing protocol of Glacius, which is run among a set of signers $\mathsf{SS} \subseteq [n]$ of size at least $t+1$ and message m to be signed. Concretely, the signers in SS engage in the following protocol to jointly compute a signature on m.

1. *Randomness sampling.* Each signer i samples a uniformly random string $\rho_i \leftarrow\!\!\$\ \{0,1\}^\lambda$, and sends it to all other signers.

2. *Commitment phase.* Upon receiving all strings $\vec{\boldsymbol{\rho}} := (\rho_j)_{j \in \mathsf{SS}}$ from signers, each signer i does the following. It samples a random $a_i \leftarrow\!\!\$\ \mathbb{Z}_p$ and computes the nonce

$$A_i := \left(g^{a_i} \cdot \mathsf{H}_0(\vec{\boldsymbol{\rho}})^{r(i)} \cdot \mathsf{H}_1(\vec{\boldsymbol{\rho}})^{u(i)}\right)^{L_{i,\mathsf{SS}}},$$

where $\mathsf{H}_0, \mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}$ are two independent random oracles and $L_{i,\mathsf{SS}}$ is the i-th Lagrange coefficient for the set SS. Signer i then sends a commitment $\mu_i := \mathsf{H}_{\mathsf{com}}(i, A_i)$ to all other signers.

3. *View exchange.* Upon receiving all commitments $\vec{\boldsymbol{\mu}} := (\mu_j)_{j \in \mathsf{SS}}$ from signers, each signer i does the following. It takes the protocol messages it received in the previous two rounds, $\vec{\boldsymbol{\rho}}$ and $\vec{\boldsymbol{\mu}}$, and computes their hash $y_i := \mathsf{H}_{\mathsf{view}}(\vec{\boldsymbol{\rho}}, \vec{\boldsymbol{\mu}})$. Then, it sends the hash value y_i to all other signers.

4. *Opening phase.* Upon receiving all hash values $(y_j)_{j \in \mathsf{SS}}$ from signers, each signer i does the following. For all $j \in \mathsf{SS}$, it checks whether $y_j = y_i$ holds[4]. If any check fails, the signer aborts. Otherwise, it sends the nonce A_i to all other signers.

5. *Signing phase.* Upon receiving all openings $(A_j)_{j \in \mathsf{SS}}$ from signers, each signer i does the following. First, for all $j \in \mathsf{SS}$, it checks whether $\mu_j = \mathsf{H}_{\mathsf{com}}(j, A_j)$ holds. If any check fails, the signer aborts. Otherwise, it computes the combined nonce $\hat{A}$, challenge c, and its signature share z_i as

$$\hat{A} := \prod_{j \in \mathsf{SS}} A_j, \quad c := \mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m), \quad z_i := L_{i,\mathsf{SS}} \cdot (a_i + c \cdot s(i)).$$

Then, it sends its signature share z_i to all other signers. The final signature is then computed as $\sigma := (\hat{A}, z)$ where $\hat{A} := \prod_{j \in \mathsf{SS}} A_j$ and $z := \sum_{j \in \mathsf{SS}} z_j$. This is a standard Schnorr signature which verifies as usual.

To prove adaptive security, [BDLR24] employs the following high-level strategy. First, the reduction algorithm $\mathcal{B}$ uses the additional public parameter $h \in \mathbb{G}$ to embed the given DL instance. This crucial change allows $\mathcal{B}$ to locally sample the secret key polynomials. Second, $\mathcal{B}$ uses rigged keys during its interaction with adversary $\mathcal{A}$. Specifically, $\mathcal{B}$ samples the polynomials $r(x)$ and $u(x)$ with non-zero $r(0)$ and uniformly random $u(0)$. Using correlated programming of additional random oracles (which is undetectable under the DDH assumption), the reduction can simulate signatures under rigged keys. These changes allows them to prove adaptive security of their threshold Schnorr signature scheme from standard assumptions. In the following, we will explain the reasoning behind their design choice and proof ideas.

Proof intuition. In their security proof, the reduction switches from honestly generated keys to rigged keys where $r := r(0) = 1$ is no longer zero, and $u := u(0) \leftarrow\!\!\$\ \mathbb{Z}_p$ is uniformly random. In particular, the rigged public key becomes $\mathsf{pk} = g^s h v^u \in \mathbb{G}$, and the central issue here is that honestly generated signatures no longer verify against the rigged public key. Specifically, let $\alpha := \alpha_h + \alpha_v u \in \mathbb{Z}_p$, where α_h is the discrete logarithm value of h and α_v is the discrete logarithm value of v, so that the rigged public key is $\mathsf{pk} = g^s h v^u = g^{s+\alpha}$. Then, the signature $\sigma := (\hat{A}, z) = (g^a, a + c \cdot s)$ does not satisfy the Schnorr verification equation with respect to pk, because

$$g^z = g^{a+s\cdot c} \neq g^a \cdot (g^{s+\alpha})^c = \hat{A} \cdot \mathsf{pk}^c, \quad \text{where } c := \mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m).$$

To resolve this issue, Glacius modifies how signers compute the nonce $\hat{A}$. Specifically, signers jointly compute it as $\hat{A} := g^a \cdot \mathsf{H}_0(x)^{r(0)} \cdot \mathsf{H}_1(x)^{u(0)}$, where $\mathsf{H}_0, \mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}$ are two independent random oracles, and x is some common input. This modification ensures that when the keys are rigged (i.e., $r(0) = 1$ and $u(0) = u$), then we have that $\hat{A} = g^a \cdot \mathsf{H}_0(x) \cdot \mathsf{H}_1(x)^u = g^a \cdot g^\gamma \cdot g^{u\beta}$ where $\mathsf{H}_0(x) = g^\gamma$ and $\mathsf{H}_1(x) = g^\beta$ for some $\beta, \gamma \in \mathbb{Z}_p$. Now, during simulation, if the reduction $\mathcal{B}$ chooses $c \in \mathbb{Z}_p$ such that

$$\gamma + \beta \cdot u + c \cdot \alpha = 0, \tag{1}$$

and programs the hash function $\mathsf{H}_{\mathsf{sig}}$ on input $\hat{A}$ to output c, i.e., $\mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m) := c$, then we get

$$\hat{A} \cdot \mathsf{pk}^c = g^a \cdot \mathsf{H}_0(x) \cdot \mathsf{H}_1(x)^u \cdot (g^{s+\alpha})^c = g^a \cdot g^\gamma \cdot g^{\beta u} \cdot g^{(s+\alpha)c} = g^a \cdot g^{sc} = g^z. \tag{2}$$

[4] We note that Glacius relies on authenticated channels for this step.

Therefore, by appropriately sampling (β, γ, c) and correlating the programming of the random oracles H_0, H_1, and $\mathsf{H}_{\mathsf{sig}}$, $\mathcal{B}$ can ensure that the final signature satisfies the Schnorr signature verification equation, even after the public key becomes rigged.

For the security proof to go through, it is critical that $\mathcal{A}$ must not detect these changes. Glacius relies on the hardness of DDH in $\mathbb{G}$ to ensure this. More precisely, in each signing session, the reduction algorithm samples fresh $c, \beta \leftarrow\$ \mathbb{Z}_p$, and then programs the random oracles as

$$\mathsf{H}_0(x) := g^{-(\beta u + c\alpha)}, \quad \mathsf{H}_1(x) := g^{\beta}.$$

In Glacius, the input x needs to be unique for every signing session. Note that, for each signing session equation (1) must hold. Now, since $\hat{A}$ and hence $c := \mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m)$ change with each signing session, the tuple (β, γ) must also change. This is because, for any fixed (β, γ), only one value of c can satisfy equation (1). Now, since (β, γ) depends on x, x must change in every signing session as well. Using a different x in every session allows $\mathcal{B}$ to sample a new (β, γ, c) tuple to satisfy equation (1) for that session.

Glacius ensures a unique x in each signing session by requiring each signer i to send a fresh random value $\rho_i \leftarrow\$ \mathbb{Z}_p$ to the other signers as the first message of the signing protocol. The value of x is then chosen as vector $\vec{\boldsymbol{\rho}} := (\rho_i)_{i \in \mathsf{SS}}$, where $\mathsf{SS} \subseteq [n]$ is the set of (at least $t+1$) signers participating in that signing session. Note that $\vec{\boldsymbol{\rho}}$ is unique with high probability, as each session consists of at least one honest signer.

Glacius has another subtle issue due to which it needs one additional round. More precisely, in any given signing session, let (β, c) denote the tuple such that $\mathsf{H}_0(x) = g^{-(\beta u + c\alpha)}$ and $\mathsf{H}_1(x) = g^{\beta}$. According to equation (2), for the simulation to succeed, we need to program $\mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m) := c$ for the combined nonce $\hat{A}$ (and public key pk and message m). The problem is that the combined nonce $\hat{A}$ depends on the second-round messages that the adversary $\mathcal{A}$ sends to the honest signers. Concretely, by sending different second-round messages to different honest signers, the adversary $\mathcal{A}$ can enforce multiple combined nonces for the same one signing session. In such a case, $\mathcal{B}$ would need to program $\mathsf{H}_{\mathsf{sig}}$ to return the same c for all these different combined nonces. Glacius addresses this issue by requiring the signers to exchange the cryptographic hash of their view during the third round of the protocol to ensure that that protocol aborts in case $\mathcal{A}$ sends different messages to different honest signers. As such, this extra round ensures consistent views among the honest signers.

Summary. We briefly summarize the need for five signing rounds in their approach:

- The commitment, opening, and signing phase are standard and follow the usual commit-and-reveal approach for designing secure interactive signing protocols. This gives three rounds of communication.
- In order to have an adaptive security proof, the public key is rigged. To make the Schnorr verification equation hold under rigged keys, the nonces are computed differently with additional random oracles that require an unpredictable common input x. This x is sampled in an additional first round.
- Their security proof cannot directly handle the case in which an adversary sends different nonces to different honest signers within one signing session. As such, their scheme introduces another round of consistency checks among honest signers.

Thus, the natural question that arises is: *Is it possible to improve upon* Glacius *and restore the three-round signing procedure while preserving security against an adaptive adversary?*

2.2 Our Approach

Our protocol starts from Glacius and uses only three rounds of communication for signing.

From five to four rounds. Our first idea is to combine the randomness sampling phase with the commitment phase. Our insight here is that a hash commitment to a randomly sampled nonce already introduces enough entropy (in the random oracle model) to be used as source of unpredictability. As such, consider the following potential four-round scheme design.

In the first round, each signer i computes a hash commitment $\mu_i := \mathsf{H}_{\mathsf{com}}(i, g^{a_i})$ to its nonce and sends it to all. It then uses the vector of received commitments $\vec{\boldsymbol{\mu}} := (\mu_i)_{i \in S}$ as input to the random oracles $\mathsf{H}_0, \mathsf{H}_1$

as described in Glacius. In the opening phase later, each signer i then computes and publishes its (masked) nonce $A_i := g^{a_i} \cdot \mathsf{H}_0(\vec{\boldsymbol{\mu}})^{r(i)} \cdot \mathsf{H}_1(\vec{\boldsymbol{\mu}})^{u(i)}$ as in Glacius. Clearly, to preserve security, signer i also has to provide a proof that the same a_i was used for computing both μ_i and A_i. Otherwise[5], an adversary could simply make its A_j's depending on honest A_i's and mount Wagner's attack [Wag02] or the ROS attack [BLL+22]. Theoretically speaking, this approach could potentially work. However, it is clear that the proof of correctness that connects A_i to the preimage of $\mu_i = \mathsf{H}_{\mathsf{com}}(i, g^{a_i})$ would be very expensive and require the use of heavy SNARKs. On the other hand, we also never can reveal the g^{a_i} itself, since the a_i values of honest signers need to be hidden to make the adaptive security proof of rigged keys go through (otherwise, the adversary could directly detect rigged keys).

Our solution. Our idea is the following. Since we cannot reveal g^{a_i} itself, we instead commit to a masked version of g^{a_i} which we can then output later along with the A_i as computed above. Concretely, each signer i commits to a_i as $\mu_i := \mathsf{H}_{\mathsf{com}}(i, \rho_i, B_i)$ where $\rho_i \leftarrow\!\$\ \{0,1\}^\lambda$ is a uniformly random string and $B_i := g^{a_i} \cdot \mathsf{F}_0(\rho_i)^{r(i)} \cdot \mathsf{F}_1(\rho_i)^{u(i)}$ for two (additional) independent random oracles $\mathsf{F}_0, \mathsf{F}_1 : \{0,1\}^* \to \mathbb{G}$. The reason why we use the random oracles $\mathsf{F}_0, \mathsf{F}_1$ to compute the base elements $h_0 := \mathsf{F}_0(\rho_i)$ and $h_1 := \mathsf{F}_1(\rho_i)$ is to use correlated programming to make simulation for honest signers with rigged keys work, similar to the technique used in Glacius. The signing protocol of our potential four-round design would look as follows:

1. *Commitment phase.* Each signer i samples a random string $\rho_i \leftarrow\!\$\ \{0,1\}^\lambda$ and a random element $a_i \leftarrow\!\$\ \mathbb{Z}_p$. Then, it computes the ephemeral nonce
$$B_i := g^{a_i} \cdot \mathsf{F}_0(\rho_i)^{r(i)} \cdot \mathsf{F}_1(\rho_i)^{u(i)},$$
where $\mathsf{F}_0, \mathsf{F}_1 : \{0,1\}^* \to \mathbb{G}$ are two independent random oracles. Signer i then sends a hash commitment $\mu_i := \mathsf{H}_{\mathsf{com}}(i, \rho_i, B_i)$ to all other signers.
2. *View exchange.* Essentially, as in Glacius to have consistent views among honest signers for $\vec{\boldsymbol{\mu}}$ by sending the hash $y_i := \mathsf{H}_{\mathsf{view}}(\vec{\boldsymbol{\mu}})$ to all other signers.
3. *Opening phase.* As usual, signer i does the consistency checks $y_j = y_i$ for all $j \in \mathsf{SS}$. Then, in the case of consistency, it computes the nonce
$$A_i := \left(g^{a_i} \cdot \mathsf{G}_0(\vec{\boldsymbol{\mu}})^{r(i)} \cdot \mathsf{G}_1(\vec{\boldsymbol{\mu}})^{u(i)}\right)^{L_{i,\mathsf{SS}}},$$
where $\mathsf{G}_0, \mathsf{G}_1 : \{0,1\}^* \to \mathbb{G}$ are two independent random oracles[6]. Then, signer i retrieves its ephemeral nonce B_i and computes a non-interactive zero-knowledge (NIZK) proof[7] π_i for correctness of A_i, given B_i, ρ_i (that determine the base elements $(g, h_0, h_1) := (g, \mathsf{F}_0(\rho_i), \mathsf{F}_1(\rho_i))$ for B_i), and pk_i. The witness for the NIZK proof is $(a_i, s(i), r(i), u(i))$. Signer i then sends $(A_i, \rho_i, B_i, \pi_i)$ to all other signers.
4. *Signing phase.* As usual, signer i does all correctness checks for other signers $j \in \mathsf{SS}$: the hash commitment $\mu_j = \mathsf{H}_{\mathsf{com}}(j, \rho_j, B_j)$ using ρ_j and B_j; the NIZK proof π_j using B_j, ρ_j, pk_j, and its own $\vec{\boldsymbol{\mu}}$ vector. In case of successful checks, it proceeds as in Glacius by computing the combined nonce $\hat{A}$, challenge c, and its signature share z_i. Finally, it sends z_i to all other signers.

Proof intuition. Our security proof would work as follows. After the first round (commitment phase), the reduction would extract (ρ_j, B_j) from the hash commitments $\mu_j = \mathsf{H}_{\mathsf{com}}(j, \cdot)$ of corrupted signers $j \in \mathcal{C}$ using observability of random oracle $\mathsf{H}_{\mathsf{com}}$. Then, it would pre-compute the nonce A_j using the corrupted signer j its secret key $\mathsf{sk}_j = (s(j), r(j), u(j))$ as follows:
$$A_j := \left(B_j \cdot \mathsf{F}_0(\rho_j)^{-r(j)} \cdot \mathsf{F}_1(\rho_j)^{-u(j)} \cdot \mathsf{G}_0(\vec{\boldsymbol{\mu}})^{r(j)} \cdot \mathsf{G}_1(\vec{\boldsymbol{\mu}})^{u(j)}\right)^{L_{j,\mathsf{SS}}}.$$

[5] An obvious solution is to introduce another hash commitment round to the masked nonce A_i itself. But this would again lead to five rounds of communication.

[6] In Glacius, these are denoted as $\mathsf{H}_0, \mathsf{H}_1$. Further, in our actual scheme Gargos, we will use $(m, \vec{\boldsymbol{\mu}})$ as input to the random oracles $\mathsf{G}_0, \mathsf{G}_1$, where m is the message. But this is just to make the security proof simpler.

[7] This NIZK proof can be instantiated very efficiently using a Σ-protocol. We do this in Figure 4.

Here, we emphasize that the reduction knows the secret key shares sk_j of adversarial parties at any point in time during the simulation. This is because $\mathcal{B}$ itself reveals these to the adversary upon corruption (we also assume an initial trusted key generation setup). Having done this, the reduction can proceed as in Glacius by programming the hash function $\mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m) := c$ on the combined nonce later. At the same time, it uses correlated programming of random oracles as (where $\mathsf{F}_0, \mathsf{F}_1$ are only programmed for honest signers $i \in \mathcal{H}$):

$$\mathsf{G}_0(\vec{\boldsymbol{\mu}}) := g^{-(\alpha\cdot\beta+\alpha_h\cdot\beta)\cdot\alpha_v^{-1}-\alpha\cdot c}, \quad \mathsf{G}_1(\vec{\boldsymbol{\mu}}) := g^{\beta},$$
$$\mathsf{F}_0(\rho_i) := g^{-(\alpha\cdot x_i+\alpha_h\cdot x_i)\cdot\alpha_v^{-1}-\alpha\cdot c}, \quad \mathsf{F}_1(\rho_i) := g^{x_i},$$

where the $x_i \leftarrow\!\!\$\ \mathbb{Z}_p$ are sampled per honest signer $i \in \mathcal{H}$ with randomness ρ_i, and $(c, \beta) \leftarrow\!\!\$\ \mathbb{Z}_p^2$ are sampled as in Glacius per signing session. The correlations of G_0 and G_1 are again used to make the Schnorr verification equation hold under rigged keys, while the correlations of F_0 and F_1 are used to preserve the relations needed to hide the nonce elements $a_i \in \mathbb{Z}_p$ even after switching to rigged keys. Note that the relation between $\mathsf{G}_0(\vec{\boldsymbol{\mu}})$ and $\mathsf{G}_1(\vec{\boldsymbol{\mu}})$ is essentially the same as the one between $\mathsf{F}_0(\rho_i)$ and $\mathsf{F}_1(\rho_i)$ (just with the β replaced by x_i). Later, in the opening phase, when the adversary outputs a nonce A'_j (on behalf of corrupt signer $j \in \mathcal{C}$) with a verifying NIZK proof π_j that is different from the nonce A_j pre-computed by the reduction, then this implies a (non-trivial) relation for the public key share pk_j as

$$g^{s_j} h^{r_j} v^{u_j} = \mathsf{pk}_j = g^{s(j)} h^{r(j)} v^{u(j)},$$

where $(s_j, r_j, u_j) \in \mathbb{Z}_p^3$ is part of the witness for generating the proof π_j. Crucially, we have the guarantee that $(s_j, r_j, u_j) \neq (s(j), r(j), u(j))$ since $A'_j \neq A_j$, which implies a non-trivial discrete logarithm relation between g, h, v which is infeasible assuming hardness of the discrete logarithm problem. In our security proof, we rule out the event that $\mathcal{A}$ outputs a different $A'_j \neq A_j$ in an initial hybrid, before programming the random oracle $\mathsf{H}_{\mathsf{sig}}$. Importantly, we can do that, since the reduction can always efficiently check this bad event, even after introducing rigged keys (which happens at the end).

From four to three rounds. Having eliminated a separate round of randomness sampling, we now revisit the security proof of Glacius to understand why the authors [BDLR24] needed the view exchange round. For this, recall that the $c \in \mathbb{Z}_p$ is sampled per signing session for each unique $\vec{\boldsymbol{\rho}}$ vector. However, an adversary could later enforce different combined nonces among different honest signers, which would force the reduction $\mathcal{B}$ to program $\mathsf{H}_{\mathsf{sig}}$ to return the same c for all these different combined nonces - clearly failing the simulation. As such, Glacius introduces the view exchange round. Looking at a deeper level, we see that the reason for this issue is that the randomness vector $\vec{\boldsymbol{\rho}}$ is decoupled from the nonces sampled by signers later. That is, the same $\vec{\boldsymbol{\rho}}$ can lead to different combined nonces. Importantly, we observe that for our modified scheme above this is not the case anymore. Concretely, given a first-round message tuple $\vec{\boldsymbol{\mu}} = (\mu_j)_{j\in\mathsf{SS}}$, then this allows the reduction to extract the adversarial nonces A_j by what we have explained before. As such, the nonces A_j are directly bound to the vector $\vec{\boldsymbol{\mu}} = (\mu_j)_{j\in\mathsf{SS}}$. On the other hand, when the adversary sends different μ_j to different honest signers, then we have the guarantee that they do not enter the third round. Concretely, consider two honest signers $i, j \in \mathcal{H}$, and let $\vec{\boldsymbol{\mu}}_{(i)}$ and $\vec{\boldsymbol{\mu}}_{(j)}$ be their received first-round messages, respectively. Further, let $(\rho_j, B_j, A_j, \pi_j)$ be the opening phase message signer j sends to signer i later. Then, assuming that $\vec{\boldsymbol{\mu}}_{(i)} \neq \vec{\boldsymbol{\mu}}_{(j)}$, we know that the proof π_j will not verify at signer i (except with negligible probability), as signer i uses its vector $\vec{\boldsymbol{\mu}}_{(i)} \neq \vec{\boldsymbol{\mu}}_{(j)}$ of first-round messages as part of the statement to verify the proof π_j, but j uses its own vector $\vec{\boldsymbol{\mu}}_{(j)}$ to generate the proof π_j (the same is also true in the other direction: the proof π_i will likely not verify at signer j). More precisely, the statement signer j proves relates to a representation of A_j in bases (g, g_0, g_1), while the statement signer i will expect relates to a representation of A_j in bases (g, g'_0, g'_1). Crucially, the elements (g_0, g_1) and (g'_0, g'_1) are independent, since they are computed via the random oracles $\mathsf{G}_0, \mathsf{G}_1$ on different inputs $\vec{\boldsymbol{\mu}}_{(j)}$ and $\vec{\boldsymbol{\mu}}_{(i)}$.

However, we need to be cautious. There is yet another subtle, more serious issue when it comes to adversarial behavior. Namely, it could be that different vectors $\vec{\boldsymbol{\mu}}_1 \neq \vec{\boldsymbol{\mu}}_2$ lead to the same combined nonce. For instance, the adversary could send a_i and a_j on behalf of corrupt signers $i, j \in \mathcal{C}$ to one honest signer, but $a_i + \nu$ and $a_j - \nu$ on behalf of corrupt signers $i, j \in \mathcal{C}$ to another honest signer for any sampled $\nu \leftarrow\!\!\$\ \mathbb{Z}_p$.

This would lead to the same combined (partial) nonce $a_i + a_j$, but obviously different $\vec{\boldsymbol{\mu}}$ vectors. Therefore, if the reduction would sample the $c \in \mathbb{Z}_p$ per tuple $\vec{\boldsymbol{\mu}}$ of first-round messages, the simulation would not work. Concretely, if it would sample $c_1 \leftarrow\$ \mathbb{Z}_p$ for $\vec{\boldsymbol{\mu}}_1$ and $c_2 \leftarrow\$ \mathbb{Z}_p$ for $\vec{\boldsymbol{\mu}}_2$, then the same combined nonce would lead to different hash programming $c_1 = \mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m) = c_2$, which an adversary could detect. Our idea to resolve this issue is to sample the c per equivalence class of $\vec{\boldsymbol{\mu}}$ vectors. Concretely, if different vectors $\vec{\boldsymbol{\mu}}_1 \neq \vec{\boldsymbol{\mu}}_2$ lead to the same combined nonce, then they define one equivalence class. This idea is inspired by a recent work [BLT+24], where the authors introduce the idea of random oracle programming on equivalence classes to handle the issue that the adversary sends different commitments to different honest signers. However, in our case, the situation seems more subtle, as we use NIZK proofs and a fundamentally different proof strategy for adaptive security. Still, we are able to succeed using the technique of equivalence classes, which allows us to eliminate the view exchange round from Glacius and obtain a three-round signing protocol with adaptive security.

3 Preliminaries

Notation. Let λ denote the security parameter. We assume that all algorithms get λ in unary as input. For a finite set S, we write $s \leftarrow\$ S$ to denote that s is sampled uniformly at random from S, and we write $|S|$ to denote the size of S. For an integer $a \in \mathbb{N}$, we use $[a]$ to denote the ordered set $\{1, \ldots, a\}$. Further, we write "$\leftarrow$" for probabilistic assignment and ":=" for deterministic assignment. We use the terms *party* (resp. *parties*) and *signer* (resp. *signers*) interchangeably. We use bold fonts with arrows such as $\vec{\boldsymbol{\rho}}$ to denote vectors. For any $i \in [n]$ and $\mathsf{SS} \subseteq [n]$, we use $L_{i,\mathsf{SS}} := \prod_{k \in \mathsf{SS} \setminus \{i\}} k/(k-i)$ for the i-th Lagrange coefficient. Throughout, when we write $\vec{\boldsymbol{\rho}} = (\rho_i)_{i \in \mathsf{SS}}$, then we really mean $(i, \rho_i)_{i \in \mathsf{SS}}$.

Threat model. We consider a system of n signers denoted by $\{1, 2, \ldots, n\}$, and a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ who can corrupt up to $t < n$ signers adaptively. Corrupted signers can deviate arbitrarily from the protocol specification. We assume an asynchronous network with authenticated channels. Importantly, we do not assume broadcast channels or secure erasures.

3.1 Shamir Secret Sharing, and Computational Assumptions

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

Computational assumptions. Our protocols assume the hardness of the discrete logarithm and decisional Diffie-Hellman problems. Let GGen be a group generation algorithm that on input 1^λ outputs the description of a prime order group $\mathbb{G}$. The description contains the prime order p, a generator $g \in \mathbb{G}$, and a description of the group operation. In our protocol, we assume the standard discrete logarithm (DL) and the decisional Diffie-Hellman (DDH) assumptions in the group $\mathbb{G}$, which we formally state in Appendix A.

3.2 Threshold Signatures

In this section, we define R-round threshold signatures following [KRT24, BDLR24].

Let $t < n$ and R be natural numbers. An R-round threshold signature scheme is a tuple of PPT algorithms $\mathsf{TS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Ver})$ defined as follows. The Setup algorithm outputs public parameters that all other algorithms take as input. The KGen algorithm generates the signing and public keys of all signers. The Sig algorithm describes what a signer does in each round of the R-round signing protocol to sign any given message. And the Ver algorithm does the final signature verification.

$\underline{\mathsf{Game}^{\mathsf{cor}}_{\mathsf{TS}}(1^\lambda, n, t, \mathsf{SS}, m):}$

1: **for** $i \in \mathsf{SS} : \mathsf{st}_i := \varnothing$
2: $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda, n, t)$
3: $(\mathsf{pk}, \{\mathsf{pk}_i, \mathsf{sk}_i\}_{i\in[n]}) \leftarrow \mathsf{KGen}(\mathsf{par})$
4: **for** $i \in \mathsf{SS} : \mathsf{pm}_{0,i} := \bot$
5: **for** $k \in [R]$:
6: **for** $i \in \mathsf{SS}$:
7: $(\mathsf{pm}_{k,i}, \mathsf{st}_i) \leftarrow \mathsf{Sig}_k(\mathsf{SS}, m, i, (\mathsf{pm}_{k-1,j})_{j\in\mathsf{SS}}, \mathsf{sk}_i, \mathsf{st}_i)$
8: $\sigma := \mathsf{Comb}(\mathsf{SS}, m, (\mathsf{pm}_{k,i})_{k\in[R],i\in\mathsf{SS}})$
9: **return** $\mathsf{Ver}(\mathsf{pk}, m, \sigma)$

Fig. 1: The game $\mathsf{Game}^{\mathsf{cor}}_{\mathsf{TS}}$ for an R-round threshold signature scheme TS.

Definition 1 (Threshold Signatures). *Let n be the total number of signers and $t < n$ be the threshold. Also, let $\mathsf{SS} \subseteq [n]$ be a set of signers with $|\mathsf{SS}| \geq t+1$. Each signer i maintains a state st_i to retain short-lived session-specific information. An R-round threshold signature scheme TS for message space $\mathcal{M}$ is a tuple of PPT algorithms $\mathsf{TS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Ver})$ defined as follows:*

- $\mathsf{Setup}(1^\lambda, n, t) \to \mathsf{par}$: *The setup algorithm takes as input the security parameter 1^λ, the number n of total signers, and a threshold $t < n$, and outputs public parameters par. We assume that all other algorithms implicitly take par as input.*
- $\mathsf{KGen}(\mathsf{par}) \to (\mathsf{pk}, \{\mathsf{pk}_i, \mathsf{sk}_i\}_{i\in[n]})$: *The key generation algorithm takes as input the public parameters par and outputs a public key pk, an ordered set of threshold public keys $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$, and an ordered set of secret signing keys $\{\mathsf{sk}_1, \ldots, \mathsf{sk}_n\}$. Each signer $j \in [n]$ receives the tuple $(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{sk}_j)$.*
- $\mathsf{Sig} = (\mathsf{Sig}_1, \ldots, \mathsf{Sig}_R, \mathsf{Comb})$: *The signing protocol is split into $R+1$ algorithms:*
  - $\mathsf{Sig}_k(\mathsf{SS}, m, i, (\mathsf{pm}_{k-1,j})_{j\in\mathsf{SS}}, \mathsf{sk}_i, \mathsf{st}_i) \to (\mathsf{pm}_{k,i}, \mathsf{st}_i)$: *The k-th round signing algorithm for $k \in [R]$ takes as input a signer set SS, a message m, an index $i \in [n]$, a tuple of protocol messages of the $(k-1)$-th round $(\mathsf{pm}_{k-1,j})_{j\in\mathsf{SS}}$, a secret signing key sk_i, and a state st_i. It outputs a protocol message $\mathsf{pm}_{k,i}$ for the k-th round and the updated state st_i. Here, we define $\mathsf{pm}_{0,j} := \bot$ for all $j \in \mathsf{SS}$.*
  - $\mathsf{Comb}(\mathsf{SS}, m, (\mathsf{pm}_{k,i})_{k\in[R],i\in\mathsf{SS}}) \to \sigma$: *The deterministic combine algorithm takes as input a signer set SS, a message m, and a tuple of protocol messages $(\mathsf{pm}_{k,i})_{k\in[R],i\in\mathsf{SS}}$, and outputs a signature σ.*
- $\mathsf{Ver}(\mathsf{pk}, m, \sigma) \to b$: *The deterministic verification algorithm takes as input a public key pk, a message m, and a signature σ, and outputs a bit $b \in \{0, 1\}$.*

We require TS to satisfy the *correctness* and *unforgeability* properties. Informally, correctness ensures that the protocol behaves as expected when everyone is honest. Unforgeability ensures that the adversary cannot forge signatures, even after engaging in previous signing sessions and corrupting up to t signers adaptively.

Definition 2 (Correctness). *Consider the game $\mathsf{Game}^{\mathsf{cor}}_{\mathsf{TS}}$ defined in Figure 1. Then, an R-round threshold signature scheme TS is correct, if for all $\lambda \in \mathbb{N}$, $n, t \in \mathsf{poly}(\lambda)$ with $t < n$, messages $m \in \mathcal{M}$, and $\mathsf{SS} \subseteq [n]$ with $|\mathsf{SS}| \geq t+1$, the following holds:*

$$\Pr\left[\mathsf{Game}^{\mathsf{cor}}_{\mathsf{TS}}(1^\lambda, n, t, \mathsf{SS}, m) \Rightarrow 1\right] \geq 1 - \mathsf{negl}(\lambda).$$

Unforgeability. Our unforgeability requirement is standard[8], and we formalize it using the game $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ in Figure 2. We give an informal description next.

Let $\mathcal{A}$ be the adversary in this game. At the beginning of the game, $\mathcal{A}$ is given the public parameters par, a joint public key pk, and threshold public keys $\{\mathsf{pk}_i\}_{i\in[n]}$ of all signers. At any time, $\mathcal{A}$ can initiate a new signing session by querying the oracle $\textsc{Next}(\mathsf{sid}, \mathsf{SS}, m)$ with session identifier sid, signer set SS, and message m. Further, $\mathcal{A}$ can corrupt up to t signers during the protocol using the oracle $\textsc{Corr}$. When signer

[8] Essentially, our notion is an interactive version of the TS-UF-0 unforgeability notion (for non-interactive threshold signatures) defined by Bellare et al. [BCK+22, BTZ22], which is similar to recent works [BDLR24, KRT24].

$i \in [n]$ gets corrupted, $\mathcal{A}$ learns its secret signing key sk_i and internal state st_i across all signing sessions. Further, $\mathcal{A}$ can interact with honest signers using the signing oracles $\textsc{Sig}_k$ for any $k \in [R]$. Specifically, $\mathcal{A}$ can query these oracles for any honest signer i and session identifier sid. For such a query to $\textsc{Sig}_k$, $\mathcal{A}$ can freely choose the protocol messages pm_{k-1} of the previous $(k-1)$-th round. Crucially, we do not assume broadcast channels in the signing protocol, and the adversary could send different messages to different honest signers. However, we do assume authenticated channels, and our unforgeability game captures this using the $\mathsf{Allowed}$ algorithm. Finally, when $\mathcal{A}$ outputs a forgery (m^*, σ^*), then $\mathcal{A}$ wins the game if the signature is valid and $\mathcal{A}$ has not initiated a signing session for the message m^* previously.

Game $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}(1^\lambda, n, t)$:

$\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda, n, t)$
$(\mathsf{pk}, \{\mathsf{pk}_i, \mathsf{sk}_i\}_{i\in[n]}) \leftarrow \mathsf{KGen}(\mathsf{par})$
$\mathcal{C} := \varnothing,\ \mathcal{H} := [n]$
$\mathsf{Queried} := \varnothing,\ \mathsf{pmsg} := \varnothing$
$\textsc{Sig} := (\textsc{Next}, (\textsc{Sig}_k)_{k\in[R]})$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Corr},\textsc{Sig}}(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]})$
if $m^* \in \mathsf{Queried}$: **return** 0
return $\mathsf{Ver}(\mathsf{pk}, m^*, \sigma^*)$

// Oracle to corrupt signers
Oracle $\textsc{Corr}(i)$:

if $(|\mathcal{C}| \geq t) \vee (i \in \mathcal{C})$: **return** $\perp$
$\mathcal{C} := \mathcal{C} \cup \{i\},\ \mathcal{H} := \mathcal{H} \setminus \{i\}$
return $(\mathsf{sk}_i, \mathsf{st}_i)$

// Oracle to start a new signing session
Oracle $\textsc{Next}(\mathsf{sid}, \mathsf{SS}, m)$:

if $(|\mathsf{SS}| < t+1) \vee (\mathsf{SS} \not\subseteq [n])$:
  return $\perp$
if $\mathsf{sid} \in \mathsf{Sessions}$: **return** $\perp$
$\mathsf{Sessions} := \mathsf{Sessions} \cup \{\mathsf{sid}\}$
$\mathsf{Queried} := \mathsf{Queried} \cup \{m\}$
$\mathsf{message}[\mathsf{sid}] := m$
$\mathsf{signers}[\mathsf{sid}] := \mathsf{SS}$
for $i \in \mathsf{SS}$: $\mathsf{round}[\mathsf{sid}, i] := 1$

// Check if the input to $\textsc{Sig}_k$ *is valid*
$\mathsf{Allowed}(\mathsf{sid}, k, \mathsf{SS}, m, i, (\mathsf{pm}_{k-1,j})_{j\in\mathsf{SS}})$:

// assert returns 0 if the check fails
assert $\mathsf{sid} \in \mathsf{Sessions}$
assert $\mathsf{SS} = \mathsf{signers}[\mathsf{sid}]$
assert $i \in (\mathsf{SS} \cap \mathcal{H})$
assert $k = \mathsf{round}[\mathsf{sid}, i]$
assert $m = \mathsf{message}[\mathsf{sid}]$
if $k = 0$: **return** 1
for $j \in (\mathsf{SS} \cap \mathcal{H})$:
  if $\mathsf{pm}_{k-1,j} \neq \mathsf{pmsg}[\mathsf{sid}, k-1, j]$:
    return 0
return 1

// Oracle for the k-th signing round
Oracle $\textsc{Sig}_k(\mathsf{sid}, \mathsf{SS}, m, i, (\mathsf{pm}_{k-1,j})_{j\in\mathsf{SS}})$:

$\mathsf{inp} := (\mathsf{SS}, m, i, (\mathsf{pm}_{k-1,j})_{j\in\mathsf{SS}})$
if $\mathsf{Allowed}(\mathsf{sid}, k, \mathsf{inp}) = 0$:
  return $\perp$
$(\mathsf{pm}_{k,i}, \mathsf{st}_i) \leftarrow \mathsf{Sig}_k(\mathsf{inp}, \mathsf{sk}_i, \mathsf{st}_i)$
$\mathsf{pmsg}[\mathsf{sid}, k, i] := \mathsf{pm}_{k,i}$
$\mathsf{round}[\mathsf{sid}, i] := k+1$
return $\mathsf{pm}_{k,i}$

Fig. 2: The $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game for an R-round threshold signature scheme TS.

Definition 3 (Unforgeability Under Chosen-Message Attacks). *Let* TS *be an* R*-round threshold signature scheme, and consider the game* $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ *defined in Figure 2. Then, we say that* TS *is* $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ *secure, if for all* $\lambda \in \mathbb{N}$, $n, t \in \mathsf{poly}(\lambda)$ *with* $t < n$, *and PPT adversaries* $\mathcal{A}$, *the following advantage is negligible:*

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

Remark. In this work, we focus on unforgeability of our scheme. However, we are confident that our scheme can be easily extended to achieve the identifiable abort property as formalized in [BDLR24]. Concretely, this can be done via a public key infrastructure (PKI) and by requiring each signer to sign each protocol message in the signing protocol before sending it to the other signers.

4 Our Design

In this section, we present our three-round threshold Schnorr signature scheme Gargos, assuming a trusted key generation. For our construction, we use $\mathcal{M}$ to denote the message space. We formally define our scheme as pseudocode in Figure 3 and give a verbal description next.

$\underline{\mathsf{Setup}(1^\lambda, n, t)}$: Let $g, h, v \leftarrow\!\!\$\ \mathbb{G}$ be three uniformly random and independent generators of $\mathbb{G}$. Further, let $\mathsf{H}_{\mathsf{com}} : \{0,1\}^\lambda \times \mathbb{G} \to \mathcal{R}$, $\mathsf{F}_0, \mathsf{F}_1 : \{0,1\}^\lambda \to \mathbb{G}$, $\mathsf{G}_0, \mathsf{G}_1 : \mathcal{M} \times \mathcal{R}^* \to \mathbb{G}$, and $\mathsf{H}_{\mathsf{sig}} : \mathbb{G}^2 \times \mathcal{M} \to \mathbb{Z}_p$ be distinct hash functions modeled as random oracles, where $\mathcal{R} \subseteq \{0,1\}^*$ is a set such that $|\mathcal{R}| \geq 2^\lambda$. The public parameters of the scheme are then $\mathsf{par} := (n, t, \mathbb{G}, g, h, v, p, \mathsf{H}_{\mathsf{com}}, \mathsf{F}_0, \mathsf{F}_1, \mathsf{G}_0, \mathsf{G}_1, \mathsf{H}_{\mathsf{sig}})$, which are known to all signers. We assume that all the algorithms below implicitly take par as input.

$\underline{\mathsf{KGen}(\mathsf{par})}$: Sample three uniformly random polynomials $s(x), r(x), u(x) \leftarrow\!\!\$\ \mathbb{Z}_p[x]$ of degree t each with the constraint that $r(0) = u(0) = 0$. The secret signing key of signer i is then $\mathsf{sk}_i := (s(i), r(i), u(i))$, and its public key share is $\mathsf{pk}_i := g^{s(i)} h^{r(i)} v^{u(i)}$. Further, the public key is $\mathsf{pk} := g^{s(0)}$.

Throughout the paper, we assume that an external mechanism chooses the message $m \in \mathcal{M}$ and the signer set $\mathsf{SS} \subseteq [n]$ with $|\mathsf{SS}| \geq t+1$ for each signing session. We also assume that all signers agree on the input message m and the set SS. If the adversary $\mathcal{A}$ inputs different (m, SS) to different honest signers, then we treat them as different signing sessions. We describe the signing protocol that signers in SS run to jointly compute a signature on m.

$\underline{\mathsf{Sig}_1(\mathsf{SS}, m, i, \mathsf{sk}_i)}$: Each signer i samples uniformly random values $a_i \leftarrow\!\!\$\ \mathbb{Z}_p$, $\rho_i \leftarrow\!\!\$\ \{0,1\}^\lambda$, and computes the ephemeral nonce $B_i \in \mathbb{G}$ as

$$B_i := g^{a_i} \cdot \mathsf{F}_0(\rho_i)^{r(i)} \cdot \mathsf{F}_1(\rho_i)^{u(i)}.$$

Signer i then sends the commitment $\mu_i := \mathsf{H}_{\mathsf{com}}(i, \rho_i, B_i)$ to all other signers. Its state after the first signing round is $\mathsf{st}_i := (i, a_i, \rho_i, B_i)$.

$\underline{\mathsf{Sig}_2(\mathsf{SS}, m, i, (\mathsf{pm}_{1,j} = \mu_j)_{j \in \mathsf{SS}}, \mathsf{sk}_i, \mathsf{st}_i)}$: Each signer i, upon receiving all commitments $\vec{\boldsymbol{\mu}} := (\mu_j)_{j \in \mathsf{SS}}$, computes the nonce $A_i \in \mathbb{G}$ as

$$A_i := g^{a_i} \cdot \mathsf{G}_0(m, \vec{\boldsymbol{\mu}})^{r(i)} \cdot \mathsf{G}_1(m, \vec{\boldsymbol{\mu}})^{u(i)}.$$

Further, it computes a NIZK proof of correctness $\pi_i := \mathsf{SigProve}((\mathsf{pk}_i, A_i, B_i, g_0, g_1, \rho_i); (a_i, \mathsf{sk}_i))$ for A_i with respect to (ρ_i, B_i) and pk_i as specified in Figure 4, where $(g_0, g_1) := (\mathsf{G}_0(m, \vec{\boldsymbol{\mu}}), \mathsf{G}_1(m, \vec{\boldsymbol{\mu}}))$. Signer i then sends $(A_i, \rho_i, B_i, \pi_i)$ to all other signers. Its state after the second signing round is $\mathsf{st}_i := (i, a_i, A_i, \vec{\boldsymbol{\mu}}, g_0, g_1)$.

$\underline{\mathsf{Sig}_3(\mathsf{SS}, m, i, (\mathsf{pm}_{2,j} = (A_j, \rho_j, B_j, \pi_j))_{j \in \mathsf{SS}}, \mathsf{sk}_i, \mathsf{st}_i)}$: Each signer i, upon receiving all second round messages $((A_j, \rho_j, B_j, \pi_j))_{j \in \mathsf{SS}}$, performs for all $j \in \mathsf{SS}$ the following two checks: (i) the opening to commitment μ_j is correct, i.e., $\mu_j = \mathsf{H}_{\mathsf{com}}(j, \rho_j, B_j)$; and (ii) π_j is a valid NIZK proof, i.e., $\mathsf{SigVer}((\mathsf{pk}_j, A_j, B_j, g_0, g_1, \rho_j); \pi_j) = 1$. If any of these checks fails, signer i outputs $\perp$ and aborts. Otherwise, it computes the combined nonce $\hat{A}$, challenge c, and its signature share z_i as follows:

$$\hat{A} := \prod_{j \in \mathsf{SS}} A_j^{L_{j,\mathsf{SS}}}, \quad c := \mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m), \quad z_i := L_{i,\mathsf{SS}} \cdot (a_i + c \cdot s(i)),$$

where $L_{j,\mathsf{SS}}$ denotes the j-th Lagrange coefficient for the set SS. Finally, signer i sends its signature share z_i to all other signers.

$\underline{\mathsf{Comb}(\mathsf{SS}, m, ((A_j, z_j))_{j \in \mathsf{SS}})}$: Upon receiving all tuples $((A_j, z_j))_{j \in \mathsf{SS}}$, the combine algorithm computes $(\hat{A}, z)$ as the final signature, where

$$\hat{A} := \prod_{j \in \mathsf{SS}} A_j^{L_{j,\mathsf{SS}}}, \quad z := \sum_{j \in \mathsf{SS}} z_j.$$

$\underline{\mathsf{Ver}(\mathsf{pk}, m, \sigma = (\hat{A}, z))}$: The verification algorithm receives a public key pk, a message $m \in \mathcal{M}$, and a signature $\sigma = (\hat{A}, z)$. Then, it computes $c := \mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m)$ and accepts the signature if and only if $g^z = \hat{A} \cdot \mathsf{pk}^c$.

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

$(\mathbb{G}, g, p) \leftarrow \mathsf{GGen}(1^\lambda),\ h, v \leftarrow\$\ \mathbb{G}$
$\mathcal{R} \subseteq \{0,1\}^*$ where $|\mathcal{R}| \geq 2^\lambda$
select the following hash functions:

- $\mathsf{H}_{\mathsf{com}} : \{0,1\}^\lambda \times \mathbb{G} \to \mathcal{R}$
- $\mathsf{F}_0, \mathsf{F}_1 : \{0,1\}^\lambda \to \mathbb{G}$
- $\mathsf{G}_0, \mathsf{G}_1 : \mathcal{M} \times \mathcal{R}^* \to \mathbb{G}$
- $\mathsf{H}_{\mathsf{sig}} : \mathbb{G}^2 \times \mathcal{M} \to \mathbb{Z}_p$

$\mathsf{H}_{\mathsf{all}} := (\mathsf{H}_{\mathsf{com}}, \mathsf{F}_0, \mathsf{F}_1, \mathsf{G}_0, \mathsf{G}_1, \mathsf{H}_{\mathsf{sig}})$
return $\mathsf{par} := (n, t, \mathbb{G}, g, h, v, p, \mathsf{H}_{\mathsf{all}})$

// All algorithms implicitly take par *as input*

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

sample $s(x), r(x), u(x) \leftarrow\$\ \mathbb{Z}_p[x]$ three random polynomials of degree t each with $r(0) = u(0) = 0$.
for each $i \in [n]$:
  let $\mathsf{sk}_i := (s(i), r(i), u(i))$
  let $\mathsf{pk}_i := g^{s(i)} h^{r(i)} v^{u(i)}$
let $\mathsf{pk} := g^{s(0)} h^{r(0)} v^{u(0)} = g^{s(0)}$
return $(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{sk}_j)$ to each signer $j \in [n]$

// Signing protocol for $\mathsf{SS} \subseteq [n]$ *and* m

$\underline{\mathsf{Sig}_1(\mathsf{SS}, m, i, \mathsf{sk}_i)}$:

sample $a_i \leftarrow\$\ \mathbb{Z}_p,\ \rho_i \leftarrow\$\ \{0,1\}^\lambda$
let $B_i := g^{a_i} \cdot \mathsf{F}_0(\rho_i)^{r(i)} \cdot \mathsf{F}_1(\rho_i)^{u(i)}$
let $\mathsf{st}_i := (i, a_i, \rho_i, B_i)$
send $\mu_i := \mathsf{H}_{\mathsf{com}}(i, \rho_i, B_i)$ to all in SS

$\underline{\mathsf{Sig}_2(\mathsf{SS}, m, i, \vec{\boldsymbol{\mu}} = (\mu_j)_{j\in\mathsf{SS}}, \mathsf{sk}_i, \mathsf{st}_i)}$:

let $g_0 := \mathsf{G}_0(m, \vec{\boldsymbol{\mu}}),\ g_1 := \mathsf{G}_1(m, \vec{\boldsymbol{\mu}})$
let $A_i := g^{a_i} \cdot g_0^{r(i)} \cdot g_1^{u(i)}$
let $X_i := (\mathsf{pk}_i, A_i, B_i, g_0, g_1, \rho_i)$
let $\pi_i := \mathsf{SigProve}(X_i; (a_i, \mathsf{sk}_i))$
update $\mathsf{st}_i := (i, a_i, A_i, \vec{\boldsymbol{\mu}}, g_0, g_1)$
send $(A_i, \rho_i, B_i, \pi_i)$ to all in SS

$\underline{\mathsf{Sig}_3(\mathsf{SS}, m, i, (A_j, \rho_j, B_j, \pi_j)_{j\in\mathsf{SS}}, \mathsf{sk}_i, \mathsf{st}_i)}$:

for all $j \in \mathsf{SS}$:
  let $X_j := (\mathsf{pk}_j, A_j, B_j, g_0, g_1, \rho_j)$
  assert $\mu_j = \mathsf{H}_{\mathsf{com}}(j, \rho_j, B_j)$
  assert $\mathsf{SigVer}(X_j; \pi_j) = 1$
let $\hat{A} := \prod_{j\in\mathsf{SS}} A_j^{L_{j,\mathsf{SS}}}$
let $c := \mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m)$
send $z_i := L_{i,\mathsf{SS}} \cdot (a_i + c \cdot s(i))$ to all in SS

// Combine algorithm

$\underline{\mathsf{Comb}(\mathsf{SS}, m, (A_j, z_j)_{j\in\mathsf{SS}})}$:

let $\hat{A} := \prod_{j\in\mathsf{SS}} A_j^{L_{j,\mathsf{SS}}}$
let $z := \sum_{j\in\mathsf{SS}} z_j$
return $\sigma := (\hat{A}, z)$

// Verification algorithm

$\underline{\mathsf{Ver}(\mathsf{pk}, m, \sigma = (\hat{A}, z))}$:

let $c := \mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m)$
if $g^z = \hat{A} \cdot \mathsf{pk}^c$:
  return 1
return 0

Fig. 3: Our three-round threshold Schnorr signature scheme Gargos.

5 Security Analysis

5.1 Properties of the Σ-protocol

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

The completeness of our Σ-protocol is straightforward. The honest-verifier zero-knowledge (HVZK) property also follow from standard Σ-protocol analyses, which we briefly discuss next.

Honest-verifier zero-knowledge (HVZK). Let $\mathcal{S}$ be the simulator. $\mathcal{S}$ samples $e, z_s, z_a, z_r, z_u \leftarrow\$\ \mathbb{Z}_p$ and computes the elements $X_A, X_B, X_{\mathsf{pk}} \in \mathbb{G}$ with $(h_0, h_1) := (\mathsf{F}_0(\rho), \mathsf{F}_1(\rho))$ as follows:

$$X_A := g^{z_a} g_0^{z_r} g_1^{z_u} \cdot A^{-e}, \quad X_B := g^{z_a} h_0^{z_r} h_1^{z_u} \cdot B^{-e}, \quad X_{\mathsf{pk}} := g^{z_s} h^{z_r} v^{z_u} \cdot \mathsf{pk}^{-e}.$$

Then, $\mathcal{S}$ programs the random oracle $\mathsf{H}_{\mathsf{FS}}(X_A, X_B, X_{\mathsf{pk}}, A, B, \mathsf{pk}, g_0, g_1, \rho) := c$ and returns $\pi := (e, z_a, z_s, z_r, z_u)$ as the proof. It is easy to see that the simulated transcript is identically distributed as an honestly generated transcript.

We will analyze the knowledge-soundness of our above Σ-protocol in §5.4 while analyzing the unforgeability of Gargos.

Statement: $(g, h, v, \mathsf{pk}, A, B, g_0, g_1, \rho) \in \mathbb{G}^8 \times \{0,1\}^\lambda$

Witness: $(a, \mathsf{sk} = (s, r, u)) \in \mathbb{Z}_p \times \mathbb{Z}_p^3$

Let $(h_0, h_1) := (\mathsf{F}_0(\rho), \mathsf{F}_1(\rho)) \in \mathbb{G}^2$. The prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$ that there exist $a, s, r, u \in \mathbb{Z}_p$ such that the following holds:

$$A = g^a \cdot g_0^r \cdot g_1^u, \quad B = g^a \cdot h_0^r \cdot h_1^u, \quad \mathsf{pk} = g^s \cdot h^r \cdot v^u$$

// We assume both algorithms implicitly take $(g, h, v, \mathsf{F}_0, \mathsf{F}_1)$ and H_{FS} as input

$\underline{\mathsf{SigProve}((\mathsf{pk}, A, B, g_0, g_1, \rho); (a, \mathsf{sk})):}$

1: **let** $(h_0, h_1) := (\mathsf{F}_0(\rho), \mathsf{F}_1(\rho))$
2: **sample** $\hat{a}, \hat{s}, \hat{r}, \hat{u} \leftarrow\$ \mathbb{Z}_p$
3: **let** $X_A := g^{\hat{a}} \cdot g_0^{\hat{r}} \cdot g_1^{\hat{u}}$, $X_B := g^{\hat{a}} \cdot h_0^{\hat{r}} \cdot h_1^{\hat{u}}$, and $X_{\mathsf{pk}} := g^{\hat{s}} \cdot h^{\hat{r}} \cdot v^{\hat{u}}$
4: **let** $e := \mathsf{H}_{\mathsf{FS}}(X_A, X_B, X_{\mathsf{pk}}, A, B, \mathsf{pk}, g_0, g_1, \rho)$
5: **let** $z_a := \hat{a} + a \cdot e$, $z_s := \hat{a} + s \cdot e$, $z_r := \hat{r} + r \cdot e$, and $z_u := \hat{u} + u \cdot e$
6: **return** $\pi := (X_A, X_B, X_{\mathsf{pk}}, z_a, z_s, z_r, z_u)$ *// $\pi := (e, z_a, z_s, z_r, z_u)$ is also sufficient for the proof*

$\underline{\mathsf{SigVer}((\mathsf{pk}, A, B, g_0, g_1, \rho); \pi):}$

6: **parse** $(X_A, X_B, X_{\mathsf{pk}}, z_a, z_s, z_r, z_u) := \pi$
7: **let** $(h_0, h_1) := (\mathsf{F}_0(\rho), \mathsf{F}_1(\rho))$
8: **let** $e := \mathsf{H}_{\mathsf{FS}}(X_A, X_B, X_{\mathsf{pk}}, A, B, \mathsf{pk}, g_0, g_1, \rho)$
9: **if** $(g^{z_a} g_0^{z_r} g_1^{z_u} = X_A \cdot A^e) \wedge (g^{z_a} h_0^{z_r} h_1^{z_u} = X_B \cdot B^e) \wedge (g^{z_s} h^{z_r} v^{z_u} = X_{\mathsf{pk}} \cdot \mathsf{pk}^e)$:
10: **return** 1
11: **return** 0

Fig. 4: Σ-protocol for proving and verifying correctness of commitments.

5.2 Correctness

The correctness of our scheme is straightforward. For this, consider a signing session among (at least) $t+1$ signers $\mathsf{SS} \subseteq [n]$ with the message $m \in \mathcal{M}$ to be signed. In the first round, each signer i receives a common commitment vector $\vec{\boldsymbol{\mu}} = (\mu_j)_{j \in \mathsf{SS}}$. In the second round, each signer i computes its nonce

$$A_i := g^{a_i} \cdot \mathsf{G}_0(m, \vec{\boldsymbol{\mu}})^{r(i)} \cdot \mathsf{G}_1(m, \vec{\boldsymbol{\mu}})^{u(i)},$$

and sends $(A_i, \rho_i, B_i, \pi_i)$ to the other signers where (i, ρ_i, B_i) is the opening for μ_i and π_i is a NIZK proof of correctness of A_i with respect to $(\rho_i, B_i, \mathsf{pk}_i)$. Since all signers behave honestly, they will all obtain the same vector $\vec{\boldsymbol{\mu}}$. Therefore, all signers i will use the same $\vec{\boldsymbol{\mu}}$ to prove correctness of its nonce A_i. Thus, by completeness of the Σ-protocol as defined in Figure 4, the proof π_i will verify at all signers and they will all compute the same combined nonce:

$$\hat{A} = \prod_{i \in \mathsf{SS}} A_i^{L_{i,\mathsf{SS}}} = g^{\sum_{i \in \mathsf{SS}} a_i \cdot L_{i,\mathsf{SS}}} \cdot \mathsf{G}_0(m, \vec{\boldsymbol{\mu}})^{r(0)} \cdot \mathsf{G}_1(m, \vec{\boldsymbol{\mu}})^{u(0)} = g^{\sum_{i \in \mathsf{SS}} a_i \cdot L_{i,\mathsf{SS}}},$$

where we use $r(0) = u(0) = 0$. Then, the challenge is derived $c := \mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m)$, and each signer i computes its signature share $z_i := L_{i,\mathsf{SS}} \cdot (a_i + c \cdot s(i))$. Therefore, the combined signature is $z := \sum_{i \in \mathsf{SS}} z_i = c \cdot s(0) + \sum_{i \in \mathsf{SS}} L_{i,\mathsf{SS}} \cdot a_i$. As a result, the standard Schnorr verification equation $g^z = \hat{A} \cdot \mathsf{pk}^c$ holds, as $\mathsf{pk} = g^{s(0)}$.

5.3 Helper Lemmas

Our unforgeability proof relies on the following two lemmas.

Lemma 1 ([NR04]). *For any $q \in \mathsf{poly}(\lambda)$, assuming hardness of the* DDH *assumption in the group $\mathbb{G}$, the following two distributions are indistinguishable:*

$$\mathcal{D}_0 := \left\{ (g, g^\alpha, (g^{\beta_i}, g^{\gamma_i})_{i \in [q]}) \;\middle|\; \alpha \leftarrow\$ \mathbb{Z}_p,\ (\beta_i, \gamma_i) \leftarrow\$ \mathbb{Z}_p^2\ \forall i \in [q] \right\},$$
$$\mathcal{D}_1 := \left\{ (g, g^\alpha, (g^{\beta_i}, g^{\alpha \cdot \beta_i})_{i \in [q]}) \;\middle|\; \alpha \leftarrow\$ \mathbb{Z}_p,\ \beta_i \leftarrow\$ \mathbb{Z}_p\ \forall i \in [q] \right\}.$$

More precisely, if an adversary $\mathcal{A}$ can distinguish between a sample from $\mathcal{D}_0$ and $\mathcal{D}_1$ with probability ε, then we can break the DDH *assumption in the group $\mathbb{G}$ with probability at least $\varepsilon - 1/p$. This implies $\varepsilon \leq \varepsilon_{\mathsf{ddh}} + 1/p$ in time $\mathsf{poly}(q, \lambda)$ and running time of $\mathcal{A}$.*

Lemma 2 ([DR24], Lemma 4). *Let (X_0, Y_0) and (X_1, Y_1) be two tuples of discrete random variables, where X_0 is independent of Y_0 and X_1 is independent of Y_1. Then, for every function $f(X_\theta, Y_\theta)$ for either $\theta \in \{0, 1\}$, if $X_0 \equiv X_1$ and $Y_0 \equiv Y_1$, where $\equiv$ indicates that the two random variables are identically distributed, then $(X_0, Y_0, f(X_0, Y_0)) \equiv (X_1, Y_1, f(X_1, Y_1))$.*

5.4 Unforgeability Proof

We will show unforgeability assuming hardness of discrete logarithm (DL) and decisional Diffie-Hellman (DDH) in the group $\mathbb{G}$. Although hardness of DL is implied by the hardness of DDH, we will keep this separation in our discussion for the modularity of the proof.

We present our proof as a sequence of games $\mathbf{G}_0$-$\mathbf{G}_{14}$, where $\mathbf{G}_0$ is the real protocol execution and $\mathbf{G}_{14}$ is the interaction of a PPT adversary $\mathcal{A}$ with a reduction algorithm $\mathcal{A}_{\mathsf{dl}}$. Hereafter, for any game $\mathbf{G}_i$, we use the notation "$\mathbf{G}_i \Rightarrow 1$" for the event that the adversary $\mathcal{A}$ forges a signature in $\mathbf{G}_i$.

GAME $\mathbf{G}_0$: This is the security game $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ for our threshold signature scheme, where the game follows the honest protocol. Here, the game gives $\mathcal{A}$ access to any random oracle by standard lazy sampling. In the following, let $\alpha_h, \alpha_v \in \mathbb{Z}_p$ denote the exponents of $h, v \in \mathbb{G}$ to base $g \in \mathbb{G}$, i.e., $h = g^{\alpha_h}$ and $v = g^{\alpha_v}$. Recall that $h, v \in \mathbb{G}$ are uniformly random generators (along with $g \in \mathbb{G}$) output by the setup algorithm Setup.

Further, we make some conceptual changes to the game that are without loss of generality. Assuming the game outputs 1, we let $(m^*, \sigma^* = (\hat{A}, \hat{z}))$ denote $\mathcal{A}$'s output forgery. Concretely, we assume that $\mathcal{A}$ makes exactly t (distinct) corruption queries and that it always queries $\mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m^*)$ before outputting the forgery. These changes are without loss of generality and do not change $\mathcal{A}$'s advantage, since we can always construct a wrapper adversary that internally runs $\mathcal{A}$, but corrupts exactly t (distinct) signers and makes a query $\mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m^*)$ before terminating. Thus, we have:

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

GAME $\mathbf{G}_1$: Here, we rule out collisions for random oracles $\mathsf{F}_0, \mathsf{F}_1, \mathsf{G}_0, \mathsf{G}_1, \mathsf{H}_{\mathsf{FS}}$, and $\mathsf{H}_{\mathsf{com}}$. Concretely, the game aborts if one of the following events happens:

(i) There are inputs $\rho \neq \rho'$ such that either $\mathsf{F}_0(\rho) = \mathsf{F}_0(\rho')$ or $\mathsf{F}_1(\rho) = \mathsf{F}_1(\rho')$. Recall that $\mathsf{F}_0, \mathsf{F}_1 : \{0,1\}^* \to \mathbb{G}$ are the random oracles signers use to compute B_j in the first round.

(ii) There are inputs $(j, \rho_j, B_j) \neq (j', \rho'_j, B'_j)$ such that $\mathsf{H}_{\mathsf{com}}(j, \rho_j, B_j) = \mathsf{H}_{\mathsf{com}}(j', \rho'_j, B'_j)$. Recall that $\mathsf{H}_{\mathsf{com}} : [n] \times \{0,1\}^\lambda \times \mathbb{G} \to \mathcal{R}$ is the random oracle signers j use to compute their first round commitment μ_j.

(iii) There are inputs $(m, \vec{\boldsymbol{\mu}}) \neq (m', \vec{\boldsymbol{\mu}}')$ such that either $\mathsf{G}_0(m, \vec{\boldsymbol{\mu}}) = \mathsf{G}_0(m', \vec{\boldsymbol{\mu}}')$ or $\mathsf{G}_1(m, \vec{\boldsymbol{\mu}}) = \mathsf{G}_1(m', \vec{\boldsymbol{\mu}}')$. Recall that $\mathsf{G}_0, \mathsf{G}_1 : \{0,1\}^* \to \mathbb{G}$ are the random oracles signers j use to compute their nonces A_j in the second round.

(iv) There are inputs $\mathsf{inp} \neq \mathsf{inp}'$ such that $\mathsf{H}_{\mathsf{FS}}(\mathsf{inp}) = \mathsf{H}_{\mathsf{FS}}(\mathsf{inp}')$. Recall that $\mathsf{H}_{\mathsf{FS}} : \{0,1\}^* \to \mathbb{Z}_p$ is the random oracle signers use to compute the NIZK proof they attach to their second round messages (i.e., the A_j).

Via standard collision probability analysis, we have that (i) happens with probability at most q_{F}^2/p, (ii) happens with probability at most $q_{\mathsf{com}}^2/|\mathcal{R}|$, (iii) happens with probability at most q_{G}^2/p, and (iv) happens with probability at most q_{FS}^2/p. In this context, we use the following notations: q_{F} is an upper bound on the total number of queries $\mathcal{A}$ can make to F_0 and F_1 combined; q_{G} is an upper bound on the total number of queries $\mathcal{A}$ can make to G_0 and G_1 combined; q_{com} and q_{FS} are upper bounds on the total number of queries $\mathcal{A}$ can make to $\mathsf{H}_{\mathsf{com}}$ and H_{FS}, respectively. Adding up everything, we thus get:

$$|\Pr[\mathbf{G}_0 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \frac{q_{\mathsf{F}}^2}{p} + \frac{q_{\mathsf{com}}^2}{|\mathcal{R}|} + \frac{q_{\mathsf{G}}^2}{p} + \frac{q_{\mathsf{FS}}^2}{p}.$$

GAME $\mathbf{G}_2$: In this game, we introduce an abort condition. Concretely, the game aborts if the following event Neq happens: During any signing session, for any corrupt party $j \in \mathcal{C}$, and any honest party $i \in \mathcal{H}$, $\mathcal{A}$ outputs a second-round tuple $(A_j, \rho_j, B_j, \pi_j)$ to signer $i \in \mathcal{H}$ such that π_j is a verifying NIZK proof and the following inequality holds:

$$B_j \cdot \mathsf{F}_0(\rho_j)^{-r(j)} \cdot \mathsf{F}_1(\rho_j)^{-u(j)} \neq A_j \cdot g_0^{-r(j)} \cdot g_1^{-u(j)}. \tag{3}$$

Here, we define $g_0 := \mathsf{G}_0(m, \vec{\boldsymbol{\mu}})$ and $g_1 := \mathsf{G}_1(m, \vec{\boldsymbol{\mu}})$ where $\vec{\boldsymbol{\mu}}$ is the vector of Sig_1 messages the honest party i receives for that signing session. Looking ahead, in game $\mathbf{G}_7$, the game will extract A_j for each malicious signer $j \in \mathcal{C}$ from its first-round message. Intuitively, this game aborts when the extracted A_j does not match with the A_j signer j outputs in the Sig_2 phase of the protocol.

We will argue the indistinguishability between these games $\mathbf{G}_1$ and $\mathbf{G}_2$ assuming hardness of DL in $\mathbb{G}$. More formally, we have the following lemma which we prove in Appendix B.1.

Lemma 3. *Let $\varepsilon_{\mathsf{dl}}$ be the advantage of $\mathcal{A}$ in breaking the discrete logarithm assumption in $\mathbb{G}$. Then, we have:*

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \frac{q_{\mathsf{FS}}}{p} + \sqrt{2 \cdot q_{\mathsf{FS}} \cdot \varepsilon_{\mathsf{dl}}}. \tag{4}$$

Here, q_{FS} is an upper bound on the number of random oracle queries $\mathcal{A}$ makes to H_{FS} (i.e., the random oracle used for the NIZK proofs).

We briefly explain the high-level idea of the proof for this lemma. From the verifying NIZK proof π_j, we obtain a non-trivial discrete logarithm relation for the public key $\mathsf{pk}_j = g^{s(j)} h^{r(j)} v^{u(j)}$ of signer j of the form $g^{s_j} h^{r_j} v^{u_j} = g^{s(j)} h^{r(j)} v^{u(j)}$, where $(s_j, r_j, u_j) \in \mathbb{Z}_p^3$ is part of the witness for generating the proof π_j. Crucially, we have the guarantee that $(s_j, r_j, u_j) \neq (s(j), r(j), u(j))$ from the inequality in Equation (3). Then, after extracting the adversary's witness by rewinding, we can solve for the discrete logarithm value of either h or v via guessing with a success probability of $1/2$.

GAME $\mathbf{G}_3$: In this game, we rule out the event that an honest signer i will accept a second-round message from another honest signer j, when the first-round messages received by these two signers are distinct. More precisely, consider two honest signers $i, j \in \mathcal{H}$, and let $\vec{\boldsymbol{\mu}}_{(i)}$ and $\vec{\boldsymbol{\mu}}_{(j)}$ be their output first-round messages (i.e., those obtained from Sig_1), respectively. Further, let $(\rho_j, B_j, A_j, \pi_j)$ be the second-round message signer j sends to signer i. Then, the game aborts if $\vec{\boldsymbol{\mu}}_{(i)} \neq \vec{\boldsymbol{\mu}}_{(j)}$ but the proof π_j verifies at signer i (that uses its vector $\vec{\boldsymbol{\mu}}_{(i)}$ of first-round messages as part of the statement to verify the proof π_j). The idea behind this game is to guarantee that honest signers do not enter the third round of the signing protocol if there are distinct first-round messages among honest signers.

Intuitively, if an honest signer j used $\vec{\boldsymbol{\mu}}_{(j)}$ to generate its proof π_j, then the proof will not verify for a different statement $\vec{\boldsymbol{\mu}}_{(i)} \neq \vec{\boldsymbol{\mu}}_{(j)}$, except with negligible probability. In particular, note that if the game does not abort, then these two games $\mathbf{G}_2$ and $\mathbf{G}_3$ are identically distributed. As such, we will analyze the probability of abort next. In the following, define

$$(g_0, g_1) := \left(\mathsf{G}_0(m, \vec{\boldsymbol{\mu}}_{(i)}), \mathsf{G}_1(m, \vec{\boldsymbol{\mu}}_{(i)})\right), \quad (g_0', g_1') := \left(\mathsf{G}_0(m, \vec{\boldsymbol{\mu}}_{(j)}), \mathsf{G}_1(m, \vec{\boldsymbol{\mu}}_{(j)})\right).$$

By the changes in game $\mathbf{G}_1$ (ruling out collisions among random oracles), we know that $g_0 \neq g_0'$ and $g_1 \neq g_1'$. Additionally, by ruling out collisions among H_{FS} (the random oracle used to derive the challenge for π) in game $\mathbf{G}_1$, we know that the challenge e signer i computes for verifying π_j (step 7 in Figure 4) is different from the challenge e' signer j computes for generating π_j (step 4 in Figure 4), i.e., $e \neq e'$.

Then, for successful verification of the proof $\pi_j := (X_A, X_B, X_{\mathsf{pk}}, z_a, z_s, z_r, z_u)$, the following has to hold:

$$X_{\mathsf{pk}_j} \cdot \mathsf{pk}_j^{e'} = g^{z_s} h^{z_r} v^{z_u} = X_{\mathsf{pk}_j} \cdot \mathsf{pk}_j^{e} \implies \mathsf{pk}_j^{e'} = \mathsf{pk}_j^{e} \implies \mathsf{pk}_j^{e'-e} = 1_{\mathbb{G}}. \tag{5}$$

Since $e' - e \neq 0$, it follows that $\mathsf{pk}_j = 1_{\mathbb{G}}$ has to hold. Thus, the probability of abort in this game is bounded by the probability that for any $j \in [n]$, it is $\mathsf{pk}_j = 1_{\mathbb{G}}$. We now bound this event.

Note that for each $j \in [n]$, we have $\mathsf{pk}_j = g^{s(j) + \alpha_h r(j) + \alpha_v u(j)}$ for values $\alpha_h, \alpha_v \in \mathbb{Z}_p$ as specified in $\mathbf{G}_1$. Thus, to have the equality $\mathsf{pk}_j = 1_{\mathbf{G}}$, it must hold that $s(j) + \alpha_h r(j) + \alpha_v u(j) = 0$. However, since the

polynomial $s(x) \in \mathbb{Z}_p[x]$ is sampled uniformly at random (along with $r(x)$ and $u(x)$), the probability that $s(j) + \alpha_h r(j) + \alpha_v u(j) = 0$ is at most $1/p$. Therefore, by taking a union bound over all signer indices $j \in [n]$, we find that the game aborts with probability at most n/p. Thus, we have:

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \frac{n}{p}.$$

GAME $\mathbf{G}_4$: In this game, we change the signing oracle. Concretely, the commitment and opening phases of the signing protocol (i.e., Sig_1 and Sig_2). Recall that until now, during the commitment phase (i.e., Sig_1), an honest signer $i \in \mathsf{SS}$ computes $B_i := g^{a_i} \cdot \mathsf{F}_0(\rho_i)^{r(i)} \mathsf{F}_1(\rho_i)^{u(i)}$ for $a_i \leftarrow\$ \mathbb{Z}_p$, $\rho_i \leftarrow\$ \{0,1\}^\lambda$, and then sends $\mu_i := \mathsf{H}_{\mathsf{com}}(i, \rho_i, B_i)$ to the other signers.

In the following, let $\vec{\boldsymbol{\mu}} := ((j, \mu_j))_{j \in \mathsf{SS}}$ be the vector of commitments signer i receives in the first round, and let m be the message for the signing session. Further, let $g_0 := \mathsf{G}_0(m, \vec{\boldsymbol{\mu}})$ and $g_1 := \mathsf{G}_1(m, \vec{\boldsymbol{\mu}})$ as before. Then, in the opening phase (i.e., Sig_2), signer i reveals its nonce along with other data $(\rho_i, B_i, A_i, \pi_i)$, where the nonce is $A_i := g^{a_i} \cdot g_0^{r(i)} g_1^{u(i)}$ and π_i is a NIZK proof of correctness for A_i with respect to B_i, ρ_i, and pk_i.

In this game, we change this as follows. During the Sig_1 phase with session identifier sid, the game samples a random commitment $\mu_i \leftarrow\$ \mathcal{R}$ and sends it on behalf of honest signer i. The game also inserts an entry (sid, i, μ_i) into a list $\mathsf{Pending\text{-}H}_{\mathsf{com}}$. If there is already an entry $(\cdot, \cdot, \mu_i) \in \mathsf{Pending\text{-}H}_{\mathsf{com}}$, the game aborts.

There are two situations where we need to reveal the pre-image of the commitment μ_i to $\mathcal{A}$. Namely, (i) in the opening phase Sig_2, we need to reveal $(\rho_i, B_i, A_i, \pi_i)$, and (ii) when $\mathcal{A}$ corrupts signer i, we further need to reveal (ρ_i, a_i) (which also gives (B_i, A_i)). To handle this, we consider two cases:

1. Signer i gets corrupted before or during the opening phase. In that case, the game proceeds as follows:
   - It samples random elements $\rho_i \leftarrow\$ \{0,1\}^\lambda$ and $a_i \leftarrow\$ \mathbb{Z}_p$.
   - If either $\mathsf{F}_0(\rho_i) \neq \perp$ or $\mathsf{F}_1(\rho_i) \neq \perp$, then the game aborts. Otherwise, it programs these random oracle values via lazy sampling $\mathsf{F}_0(\rho_i) \leftarrow\$ \mathbb{G}$ and $\mathsf{F}_1(\rho_i) \leftarrow\$ \mathbb{G}$.
   - It computes $B_i := g^{a_i} \cdot \mathsf{F}_0(\rho_i)^{r(i)} \mathsf{F}_1(\rho_i)$. If $\mathsf{H}_{\mathsf{com}}(i, \rho_i, B_i) \neq \perp$, the game aborts. Otherwise, it programs $\mathsf{H}_{\mathsf{com}}(i, \rho_i, B_i) := \mu_i$, and then removes the entry (sid, i, μ_i) from $\mathsf{Pending}_{\mathsf{com}}$.
   - Next, it honestly computes $A_i := g^{a_i} \cdot g_0^{r(i)} g_1^{u(i)}$ where $g_0 := \mathsf{G}_0(m, \vec{\boldsymbol{\mu}})$ and $g_1 := \mathsf{G}_1(m, \vec{\boldsymbol{\mu}})$ for the first-round message $\vec{\boldsymbol{\mu}}$ signer i has received. Then, it honestly computes the NIZK proof π_i (proof of correctness for A_i with respect to B_i, ρ_i, and pk_i). Note that if signer i gets corrupted before it receives the first-round vector $\vec{\boldsymbol{\mu}}$, then the elements A_i, π_i are not defined.
   - After all of the steps above, the game can reveal $(\rho_i, B_i, A_i, \pi_i)$ in the opening phase, and in case the corruption happens before that, it can also reveal a_i along with $(\rho_i, B_i, A_i, \pi_i)$ for that session.
2. Signer i reaches the opening phase or gets corrupted after the opening phase. In that case, the game proceeds as above to compute $(a_i, \rho_i, B_i, A_i, \pi_i)$, output $(\rho_i, B_i, A_i, \pi_i)$ as the second-round (Sig_2) message, and additionally reveal a_i to $\mathcal{A}$ upon corruption.

With this change, we emphasize the following: For honest signers i that reach the opening phase without being corrupted (the second case above), we sample the elements ρ_i, a_i, and program the random oracles $\mathsf{F}_0, \mathsf{F}_1$ on ρ_i only after signer i receives all the first-round (Sig_1) messages.

Clearly, the view of $\mathcal{A}$ is only affected by this change if any of the following three events occurs. (i) First, the game samples the same commitment $\mu_i \in \mathcal{R}$ twice. (ii) Second, ρ_i matches a previous F_b query for either $b \in \{0,1\}$. (iii) Third, the tuple (i, ρ_i, B_i) matches a previous $\mathsf{H}_{\mathsf{com}}$ query.

We first analyze the probability of the game aborting in one signing session. For (i), note that the game samples each μ_i uniformly at random from $\mathcal{R}$; hence, in a single signing session, (i) happens with probability at most $q_s/|\mathcal{R}|$. Similarly, for (ii), ρ_i is uniformly random in $\{0,1\}^\lambda$. This implies that each ρ_i will match a previous F_b query for either $b \in \{0,1\}$ with probability at most $q_{\mathsf{F}}/2^\lambda$. Finally, for (iii), a_i is uniformly random in $\mathbb{Z}_p$, so B_i is uniformly random in $\mathbb{G}$. Therefore, each (i, ρ_i, B_i) will match a previous $\mathsf{H}_{\mathsf{com}}$ query with probability at most q_{com}/p. Therefore, taking a union bound over all signing sessions, we get:

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq \frac{q_s^2}{|\mathcal{R}|} + \frac{q_s \cdot q_{\mathsf{F}}}{2^\lambda} + \frac{q_s \cdot q_{\mathsf{com}}}{p}.$$

GAME $\mathbf{G}_5$: In this game, we rule out the event that upon receiving a first-round signing request from the adversary for honest signer i, we sample the same μ_i twice. To do so, we maintain a list PastMu to keep track of previously sampled first-round messages by honest signers. We populate PastMu upon each SIG_1 oracle query. Recall from game $\mathbf{G}_4$, upon each SIG_1 on input of the form $(\cdot, i, \cdot)$, the game samples $\mu_i \leftarrow\$ \{0,1\}^\lambda$. In this game, if $(i, \mu_i) \in \mathsf{PastMu}$, the game aborts. Otherwise, the updates $\mathsf{PastMu} := \mathsf{PastMu} \cup \{(i, \mu_i)\}$. We note that PastMu differs from $\mathsf{Pending}_{\mathsf{com}}$ (we introduce in game $\mathbf{G}_4$) in the following sense. PastMu stores all previously sample first-round messages (including the ones from the closed signing session). Contrary to this, $\mathsf{Pending}_{\mathsf{com}}$ only stores $\mathsf{H}_{\mathsf{com}}$ outputs for which the game is yet to sample a pre-image.

Looking ahead, this game, combined with our use of authenticated channels ensure that the first-round messages an honest signer receives in each signing session are unique. This is because the $\vec{\boldsymbol{\mu}}$ vector includes contributions from at least one honest signer. Since honest signers do not sample the same first-round message twice, the $\vec{\boldsymbol{\mu}}$ vector will be unique for every signing session.

We now bound the probability of aborting in this game. For each signing query, since μ_i is uniformly random, the game aborts with probability at most $q_s/2^\lambda$ (where q_s is an upper bound on the number of signing queries $\mathcal{A}$ can make). As a result, by a union bound over all signing queries, we get:

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq \frac{q_s^2}{2^\lambda}.$$

GAME $\mathbf{G}_6$: In this game, we abort if $\mathcal{A}$ breaks the observability of the random oracle $\mathsf{H}_{\mathsf{com}}$. More precisely, the game aborts if for any signing session and $j \in \mathcal{C}$, $\mathcal{A}$ outputs the tuple $(\rho_j, B_j, \cdot)$ in the second round for its first-round message μ_j such that $\mu_j = \mathsf{H}_{\mathsf{com}}(j, \rho_j, B_j)$, without having queried $\mathsf{H}_{\mathsf{com}}$ on (j, ρ_j, B_j).

Note that for each $j \in \mathcal{C}$, unless $\mathsf{H}_{\mathsf{com}}(j, \cdot) \neq \perp$, the game outputs $\mathsf{H}_{\mathsf{com}}(j, \cdot)$ with uniformly random values in $\mathcal{R}$. Moreover, after $\mathcal{A}$ corrupts the signer j, the game populates $\mathsf{H}_{\mathsf{com}}$ for inputs of the form $(j, \cdot)$ only when $\mathcal{A}$ explicitly queries $\mathsf{H}_{\mathsf{com}}$ on the input $(j, \cdot)$. Therefore, the probability of $\mathsf{H}_{\mathsf{com}}(j, B_j) = \mu_j$ for each (j, B_j) is $1/|\mathcal{R}|$. Hence, by a union bound over all signing sessions, we get:

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \frac{q_s}{|\mathcal{R}|}.$$

From here on, our security analysis critically deviates from the one of Glacius. While the security proof in Glacius assigns the challenge c to the randomness vector $\vec{\boldsymbol{\rho}}$, which is decoupled from the combined nonce, we observe that our modified scheme resolves this issue. Concretely, our randomness vector $\vec{\boldsymbol{\mu}}$ already encodes adversarial nonces (inside the hash commitment), and crucially, different $\vec{\boldsymbol{\mu}}$ vectors could lead to the same combined nonce. To handle this, we define the notion of equivalent $\vec{\boldsymbol{\mu}}$ vectors.

GAME $\mathbf{G}_7$: In this game, we introduce a list $\mathsf{Pending}$ and associated algorithms $\mathsf{UpdatePending}$ and $\mathsf{AddToPending}$ to manage this list. The list $\mathsf{Pending}$ keeps track of inputs to random oracles G_b for either $b \in \{0,1\}$ for which the game can not yet extract pre-images of all commitments included in the G_b input. More precisely, the list $\mathsf{Pending}$ contains a tuple $(m, \mathsf{SS}, \vec{\boldsymbol{\mu}})$ if and only if the following two invariants hold:

1. For each $(j, \mu_j) \in \vec{\boldsymbol{\mu}}$ with $(\cdot, j, \mu_j) \notin \mathsf{Pending}\text{-}\mathsf{H}_{\mathsf{com}}$, we have $\mathsf{H}_{\mathsf{com}}^{-1}(\mu_j) \neq \perp$.
2. There is a commitment $(j, \mu_j) \in \vec{\boldsymbol{\mu}}$ such that $\mathsf{H}_{\mathsf{com}}^{-1}(\mu_j) = \perp$.

We add a tuple $(m, \mathsf{SS}, \vec{\boldsymbol{\mu}})$ to $\mathsf{Pending}$ using the algorithm $\mathsf{AddToPending}$ immediately after the call to G_b for either $b \in \{0,1\}$ with the input $(m, \vec{\boldsymbol{\mu}} = ((j, \mu_j))_{j \in \mathsf{SS}}$. Furthermore, we invoke algorithm $\mathsf{UpdatePending}$ whenever we update $\mathsf{H}_{\mathsf{com}}$, either during queries to $\mathsf{H}_{\mathsf{com}}$ or during corruption and signing queries. On each invocation, the $\mathsf{UpdatePending}$ algorithm does the following:

1. Initialize an empty list $\mathsf{Visited}$.
2. Iterate through all entries $(m, \mathsf{SS}, \vec{\boldsymbol{\mu}})$ in $\mathsf{Pending}$ and do the following:
   (a) Check if the entry has to be removed because it is violating the invariant. That is, check if for all $j \in \mathsf{SS}$, we have $\mathsf{H}_{\mathsf{com}}^{-1}(\mu_j) \neq \perp$. If this is not the case, skip this entry and keep it in $\mathsf{Pending}$.

(b) Otherwise, we know that for all $j \in \mathsf{SS}$, the value $(j, \rho_j, B_j) := \mathsf{H}_{\mathsf{com}}^{-1}(\mu_j)$. Remove this entry from $\mathsf{Pending}$, and determine the commitment R_j as

$$R_j := B_j \cdot \mathsf{F}_0(\rho_j)^{-r(j)} \cdot \mathsf{F}_1(\rho_j)^{-u(j)}. \tag{6}$$

Next, compute the combined nonce $\hat{R} := \prod_{j \in \mathsf{SS}} R_j^{L_{j,\mathsf{SS}}}$.

(c) Let $g_0 := \mathsf{G}_0(m, \vec{\boldsymbol{\mu}})$ and $g_1 := \mathsf{G}_1(m, \vec{\boldsymbol{\mu}})$. Compute $\hat{A} := \hat{R} \cdot g_0^{r(0)} \cdot g_1^{u(0)}$.

(d) If $\mathsf{Chal}[\hat{R}] = \bot$: (i) Set $\mathsf{Nonce}[\hat{R}] := \hat{A}$, (ii) sample $\mathsf{Chal}[\hat{R}] := c \leftarrow\!\!\$\ \mathbb{Z}_p$.

(e) If $\hat{A} \neq \mathsf{Nonce}[\hat{R}]$, abort the execution of the entire game.

(f) If $(\hat{R}, m) \notin \mathsf{Visited}$: 1. If $\mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m) \neq \bot$, abort the execution of the entire game. 2. Otherwise, program $\mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m) := \mathsf{Chal}[\hat{R}]$, and insert the tuple $(\hat{R}, m)$ into $\mathsf{Visited}$.

In summary, the $\mathsf{UpdatePending}$ algorithm removes all entries violating the invariant from the list $\mathsf{Pending}$. For each such entry it removes, the algorithm computes $\hat{R}$ and the combined nonce $\hat{A}$. The game then tries to program $\mathsf{H}_{\mathsf{sig}}$ on input $(\hat{A}, \mathsf{pk}, m)$ with a uniformly random challenge $c := \mathsf{Chal}[\hat{R}]$ for some map Chal. During this process, the aborts in two situations: First, (shown in step (e) above), the game aborts, if there exists two triples $(m, \mathsf{SS}, \vec{\boldsymbol{\mu}})$ and $(m', \mathsf{SS}', \vec{\boldsymbol{\mu}}')$ with the same $\hat{R}$ but distinct $\hat{A}$. Second (shown in step (f).i. above), if the random oracle $\mathsf{H}_{\mathsf{sig}}$ on input $(\hat{A}, \mathsf{pk}, m)$ is already defined. List $\mathsf{Visited}$ ensures that the abort is not triggered if the algorithm itself programmed $\mathsf{H}_{\mathsf{sig}}$ in a previous iteration within the same invocation.

Note that if the game does not abort, then for each $\hat{R}$, there is an unique $\hat{A}$, and the game programs $\mathsf{H}_{\mathsf{sig}}$ on input $(\hat{A}, \mathsf{pk}, m)$ at most once with an uniformly random $c := \mathsf{Chal}[\hat{R}]$. Therefore, if the game does not abort, its view in game $\mathbf{G}_6$ and $\mathbf{G}_7$ are identically distributed.

We now analyze the probability that the game aborts. Note that since $r(0) = u(0) = 0$, we have $\hat{R} = \hat{A}$, and hence the abort condition (e) in never occurs. Now, for the condition (f.1), note that when we remove an entry from the list $\mathsf{Pending}$, and let $\hat{R} = \prod_{j \in \mathsf{SS}} (R_j)^{L_{j,\mathsf{SS}}}$ is as defined above, then there exists an $j \in \mathsf{SS}$ such that the game just sampled $a_j \in \mathbb{Z}_p$ such that $R_j := g^{a_j}$ before invoking the $\mathsf{UpdatePending}$ algorithm. Now, since R_j is uniformly random, so is $\hat{R}$ and hence $\hat{A}$. Moreover, $\hat{A}$ is hidden from $\mathcal{A}$ at the time of programming $\mathsf{H}_{\mathsf{sig}}$. This implies each input $(\hat{A}, \mathsf{pk}, m)$ will collide with a prior random oracle query with at most q_{sig}/p probability. Therefore, taking union bound over all signing queries, we get that the game will abort with probability at most $(q_s \cdot q_{\mathsf{sig}})/p$. Therefore, we get:

$$|\Pr[\mathbf{G}_6 \Rightarrow 1] - \Pr[\mathbf{G}_7 \Rightarrow 1]| \leq \frac{q_s \cdot q_{\mathsf{sig}}}{p}.$$

GAME $\mathbf{G}_8$: In this game, we introduce two more algorithms, $\mathsf{Equivalent}$, and $\mathsf{GetChallenge}$. Intuitively, these allow us to group triples of the form $(m, \mathsf{SS}, \vec{\boldsymbol{\mu}} = ((j, \mu_j))_{j \in \mathsf{SS}})$ that has been inserted into the list $\mathsf{Pending}$ into equivalence classes. We define this relation on all triples in $\mathsf{Pending}$ and all triples that already have been removed from $\mathsf{Pending}$, but not on any other entries. The intuition is that all such triples lead to the same combined nonce if and only if they are in the same equivalence class. The effect of this will be that we know the challenge just from the triple $(m, \mathsf{SS}, \vec{\boldsymbol{\mu}})$. Next, we describe the algorithm $\mathsf{Equivalent}$ that takes as input $(m, \mathsf{SS}, \vec{\boldsymbol{\mu}})$ and $(m', \mathsf{SS}', \vec{\boldsymbol{\mu}}')$, and decides whether they are equivalent. Let $\vec{\boldsymbol{\mu}} = [(j, \mu_j)]_{j \in \mathsf{SS}}$ and $\vec{\boldsymbol{\mu}}' = [(j, \mu'_j)]_{j \in \mathsf{SS}'}$. Then, the algorithm $\mathsf{Equivalent}$ works as follows:

1. If $m \neq m'$ or $\mathsf{SS} \neq \mathsf{SS}'$, then the triples are not equivalent.
2. Let $T \subseteq \mathsf{SS}$ (resp. $T' \subseteq \mathsf{SS}'$) be the set of indices $j \in \mathsf{SS}$ (resp. $j \in \mathsf{SS}'$) such that $\mathsf{H}_{\mathsf{com}}^{-1}(\mu_j) = \bot$ (resp. $\mathsf{H}_{\mathsf{com}}^{-1}(\mu'_j) = \bot$. Then, if either $T \neq T'$ or $[\mu_j]_{j \in T} \neq [\mu'_j]_{j \in T'}$ then the triples are not equivalent.
3. Let $\overline{T} = \mathsf{SS} \setminus T$ and $\overline{T}' = \mathsf{SS}' \setminus T$. For each $j \in \overline{T}$, we know that $(j, \rho_j, B_j) = \mathsf{H}_{\mathsf{com}}^{-1}(\mu_j)$ and let $(j, \rho'_j, B'_j) = \mathsf{H}_{\mathsf{com}}^{-1}(\mu'_j)$. Given these tuples, R_j and R'_j can be defined as:

$$R_j := B_j \cdot \mathsf{F}_0(\rho_j)^{-r(j)} \cdot \mathsf{F}_1(\rho_j)^{-u(j)}, \quad R'_j := B'_j \cdot \mathsf{F}_0(\rho'_j)^{-r(j)} \cdot \mathsf{F}_1(\rho'_j)^{-u(j)}. \tag{7}$$

Then, we can define partially combined nonce as:

$$\overline{R} = \prod_{j \in \overline{T}} (R_j)^{L_{j,\mathsf{SS}}} \text{ and } \overline{R}' = \prod_{j \in \overline{T}'} (R'_j)^{L_{j,\mathsf{SS}}}.$$

If $\overline{R} \neq \overline{R}'$, then the triples are not equivalent. Otherwise, they are equivalent.

In summary, two triples are equivalent if their signer sets, messages, partially combined nonces, and the remaining commitments match. Clearly, at any fixed point in time during the experiment, this is indeed an equivalence relation. Next, we will argue that the relation is preserved over time.

The equivalence relation can only change when we update $\mathsf{H}_{\mathsf{com}}$ oracle either during queries to Sig_2 phase or during corruption queries, i.e., whenever we remove an element from $\mathsf{Pending}\text{-}\mathsf{H}_{\mathsf{com}}$.

Claim (Equivalent triples remain equivalent). If two triples $(m, \mathsf{SS}, \vec{\boldsymbol{\mu}})$ and $(m', \mathsf{SS}', \vec{\boldsymbol{\mu}}')$ are equivalent at some point in time, then they stay equivalent for the rest of the game.

Proof. The signer set and the message do not change over time. For the rest of the conditions, let T and T' as defined above. Note that by definition of an equivalence class, we have $T = T'$ and $(\mu_j)_{j \in T} = (\mu'_j)_{j \in T}$.

Now, for every update of $\mathsf{H}_{\mathsf{com}}$, we have two cases: First, the update does not affect $\mu_k \in \vec{\boldsymbol{\mu}}$ for any $k \in T$. In this case, the triples stay equivalent. Second, the update samples pre-image of $\mu_k \in \vec{\boldsymbol{\mu}}$ for some $k \in T$. In this case, both T and T' get updated in the identical manner. Similarly, the combined partial nonce $\overline{R}$ and $\overline{R}'$ gets updated in an identical manner by multiplication by R_k where:

$$R_k = \left(B_k \cdot \mathsf{F}_0(\rho_k)^{-r(k)} \cdot \mathsf{F}_1(\rho_k)^{-u(k)}\right)^{L_{k,\mathsf{SS}}}. \tag{8}$$

Therefore, the triples remain equivalent. □

Claim (Non-equivalent remains non-equivalent, except with a negligible probability). If two triples $(m, \mathsf{SS}, \vec{\boldsymbol{\mu}})$ and $(m', \mathsf{SS}', \vec{\boldsymbol{\mu}}')$ are not equivalent at some point in time, then the probability that they become equivalent later is negligible. Concretely, if $\mathsf{Converge}$ is the event that any two non-equivalent triples become equivalence at some point tin time, then $\Pr[\mathsf{Converge}] \leq q_s^2/p$.

Proof. Clearly, if $m \neq m'$ and $\mathsf{SS} \neq \mathsf{SS}'$, then the triples remain non-equivalent. Now consider an update of $\mathsf{H}_{\mathsf{com}}$ that resulted in removing an entry μ from $\mathsf{Pending}\text{-}\mathsf{H}_{\mathsf{com}}$. Then, we consider the following cases.

1. First, $\mu = \mu_k = \mu'_k$ for some $k \in T \cap T'$. Then:
   (a) If $T \neq T'$, then the updated $T \setminus \{k\} \neq T' \setminus \{k\}$, hence they remain non-equivalent.
   (b) If $T = T'$ but $[\mu_j]_{j \in T} \neq [\mu'_j]_{j \in T}$ before sampling, then, after sampling pre-image of μ, we have that $[\mu_j]_{j \in T \setminus \{k\}} \neq [\mu'_j]_{j \in T \setminus \{k\}}$. Hence, they remain non-equivalent.
   (c) If $T = T'$ and $[\mu_j]_{j \in T} = [\mu'_j]_{j \in T}$, then by definition, $\overline{R} \neq \overline{R}'$. Now, for μ_k, let R_k be the value as per equation (7) of μ_k. Then, we have that: $\overline{R}/R_k \neq \overline{R}'/R_k$. Therefore, the triples remain non-equivalent.
2. Second, we sample a pre-image of $\mu_k \in \vec{\boldsymbol{\mu}}$ but $\mu_k \notin \vec{\boldsymbol{\mu}}'$. Let $\overline{R}$ and $\overline{R}'$ be the combined nonces of the two triples, respectively, before we define the pre-image of μ_k. Then, after we sample pre-image of μ_k, the tuple can become equivalent if and only if (a) $T \setminus \{k\} = T'$ and $[\mu_j]_{j \in T \setminus \{k\}} = [\mu'_j]_{j \in T'}$, or (b) the nonce $R_k = \overline{R}'/\overline{R}$.

   Since R_k is uniformly random, the probability of the event $R_k = \overline{R}'/\overline{R}$ is at most $1/p$. Note that in the entire game, we will do at most q_s pre-image sampling of inputs of $\mathsf{H}_{\mathsf{com}}$, and $\mathsf{Pending}$ has at most q_{com} tuples. Therefore, taking union bound across all pre-image sampling, we get that $\mathsf{Converge}$ happens with probability at most $(q_s \cdot q_{\mathsf{com}})/p$. □

With our equivalence relation at hand, we introduce an algorithm $\mathsf{GetChallenge}$ that behaves as a random oracle on equivalence classes. That is, it assigns each class a random oracle on equivalence classes. That is, it assigns each class a random challenge $c \leftarrow\$ \mathbb{Z}_p$ in a lazy manner. More precisely, it gets as input a triple $(m, \mathsf{SS}, \vec{\boldsymbol{\mu}})$ and checks if a triple in the same equivalence class is already assigned a challenge c. This is done using the algorithm $\mathsf{Equivalent}$. If so, it returns this challenge c. If not, it assigns a random challenge $c \leftarrow\$ \mathbb{Z}_p$ to the triple $(m, \mathsf{SS}, \vec{\boldsymbol{\mu}})$.

These two new algorithms are used in the following way. Recall, in the previous game, whenever the algorithm $\mathsf{UpdatePending}$ removes an triple $(m, \mathsf{SS}, \vec{\boldsymbol{\mu}})$ from $\mathsf{Pending}$ and no abort occurs, the algorithm programs $\mathsf{H}_{\mathsf{sig}}(\hat{A}, \mathsf{pk}, m) := c$ for an uniformly random $c := \mathsf{Chal}[\hat{R}]$ where where $\hat{R}$ is as defined in equation (7) and $\hat{A} := \hat{R} \cdot \mathsf{G}_0(m, \vec{\boldsymbol{\mu}})^{r(0)} \cdot \mathsf{G}_1(m, \vec{\boldsymbol{\mu}})^{u(0)}$. Now, instead of sampling c uniformly at random, the game sets $\mathsf{Chal}[\hat{R}] := \mathsf{GetChallenge}(m, \mathsf{SS}, \vec{\boldsymbol{\mu}})$.

It follows from the definition of a non-equivalence class that $\hat{R}$ values of two non-equivalent classes are different. Moreover, the value of $\mathsf{GetChallenge}$ for each equivalence class is uniformly at random. Therefore, if the game does not abort in this game, then we program $\mathsf{H}_{\mathsf{sig}}$ identically as in game $\mathbf{G}_7$. Hence, we get:

$$|\Pr[\mathbf{G}_7 \Rightarrow 1] - \Pr[\mathbf{G}_8 \Rightarrow 1]| \leq \Pr[\mathsf{Converge}] \leq \frac{q_s^2}{p}. \tag{9}$$

GAME $\mathbf{G}_9$: In this game, we change how we program the oracles $\mathsf{F}_0, \mathsf{F}_1, \mathsf{G}_0$, and G_1. We note that we still program these random oracles with uniformly random values, we simply change how we choose these values.

First sample a uniformly random $\alpha \leftarrow\$ \mathbb{Z}_p$, once for the entire game. Next, for every random oracle query to G_b on input $(m, \vec{\boldsymbol{\mu}}_\ell)$ for either $b \in \{0, 1\}$, if $\mathsf{G}_b(m, \vec{\boldsymbol{\mu}}_\ell) = \perp$, we sample $\beta_\ell, \gamma_\ell \leftarrow\$ \mathbb{Z}_p$, program the random oracles as:

$$\mathsf{G}_0(m, \vec{\boldsymbol{\mu}}_\ell) := g^{(-\gamma_\ell - \alpha_h \cdot \beta_\ell) \cdot \alpha_v^{-1} - \alpha \cdot c_\ell}, \quad \mathsf{G}_1(m, \vec{\boldsymbol{\mu}}_\ell) := g^{\beta_\ell}. \tag{10}$$

Here, $c_\ell := \mathsf{GetChallenge}(m, \mathsf{SS}, \vec{\boldsymbol{\mu}}_\ell)$ and $\mathsf{SS} \subseteq [n]$ is the set for $\vec{\boldsymbol{\mu}} = ((j, \mu_j))_{j \in \mathsf{SS}}$.

We will now describe how we change programming the random oracles F_0 and F_1. Recall from game $\mathbf{G}_2$, for each signing session and for each signer i, we sample ρ_i and program $\mathsf{F}_0(\rho_i)$ and $\mathsf{F}_1(\rho_i)$ with uniformly random values in $\mathbb{G}$, only after party i receives all its first round message $\vec{\boldsymbol{\mu}}_{(i)}$. In this game, for every signing session and every signer i, we sample $x_i, y_i \leftarrow\$ \mathbb{Z}_p$. Also, let $c_i := \mathsf{GetChallenge}(m, \mathsf{SS}, \vec{\boldsymbol{\mu}}_{(i)})$. Then, we program F_0 and F_1 as:

$$\mathsf{F}_0(\rho_i) := g^{(-y_i - \alpha_h \cdot x_i) \cdot \alpha_v^{-1} - \alpha \cdot c_i}, \quad \mathsf{F}_1(\rho_i) := g^{x_i}. \tag{11}$$

Note that since each γ_ℓ is uniformly random and independent of $(\beta_\ell, \alpha, c_\ell)$, and $\alpha_v \neq 0$, $(-\gamma_\ell - \alpha_h \cdot \beta_\ell) \cdot \alpha_v^{-1} - c_\ell \cdot \alpha$ is also uniformly random and independent. Similarly, each y_i is also uniformly random and independent of (α, x_i, c_i). Hence, $(-y_i - \alpha_h \cdot x_i) \cdot \alpha_v^{-1} - c_i \cdot \alpha$ is also uniformly random and independent. This implies that, in this game we program the random oracles $\mathsf{F}_0, \mathsf{F}_1, \mathsf{G}_0$ and G_1 with uniformly random values, and hence: $\Pr[\mathbf{G}_8 \Rightarrow 1] = \Pr[\mathbf{G}_9 \Rightarrow 1]$.

GAME $\mathbf{G}_{10}$: So far, we programmed the random oracles $\mathsf{F}_0, \mathsf{F}_1, \mathsf{G}_0$ and F_1 with uniformly random values from the appropriate range. In this game, we change that and instead program these random oracles with correlated values.

In this game, for each query to G_b for either $b \in \{0, 1\}$ on input $(m, \vec{\boldsymbol{\mu}}_\ell)$, we program the G_0 and G_1 as in game $\mathbf{G}_9$ except we compute $\gamma_\ell := -\alpha \cdot \beta_\ell$, instead of sampling it uniformly at random. More precisely, we sample $\beta_\ell \leftarrow\$ \mathbb{Z}_p$ and program:

$$\mathsf{G}_0(m, \vec{\boldsymbol{\mu}}_\ell) := g^{(-\alpha \cdot \beta_\ell - \alpha_h \cdot \beta_\ell) \cdot \alpha_v^{-1}} \cdot (g^{-\alpha})^{c_\ell}, \quad \mathsf{G}_1(m, \vec{\boldsymbol{\mu}}_\ell) := g^{\beta_\ell}, \tag{12}$$

where $c_\ell := \mathsf{GetChallenge}(m, \mathsf{SS}, \vec{\boldsymbol{\mu}}_\ell)$. We introduce a similar change for F_0 and F_1. For each signing session and for each signer i, we compute $y_i := -\alpha \cdot x_i$, instead of sampling y_i uniformly at random. More precisely, we sample $x_i \leftarrow\$ \mathbb{Z}_p$ and program:

$$\mathsf{F}_0(\rho_i) := g^{(-\alpha \cdot x_i - \alpha_h \cdot x_i) \cdot \alpha_v^{-1}} \cdot (g^{-\alpha})^{c_\ell}, \quad \mathsf{F}_1(\rho_i) := g^{x_i}, \tag{13}$$

where $c_i := \mathsf{GetChallenge}(m, \mathsf{SS}, \vec{\boldsymbol{\mu}}_{(i)})$ with $\vec{\boldsymbol{\mu}}_{(i)}$ being the first-round messages received by the signer i.

The indistinguishability between game $\mathbf{G}_9$ and $\mathbf{G}_{10}$ is another crucial step of our proof. We will prove this assuming hardness of DDH in $\mathbb{G}$. To this end, we rely on the following corollary of Lemma 1.

Corollary 1. *For any security parameter λ, let $(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^\lambda)$ be a prime order group of order p with generator $g \in \mathbb{G}$. For all $n, q \leq \mathsf{poly}(\lambda)$, any $\alpha_h, \alpha_v \in \mathbb{Z}_p^*$ and any vector $\boldsymbol{c} = [c_1, \ldots, c_q]$, assuming hardness of the DDH assumption in $\mathbb{G}$, the following two distributions are indistinguishable:*

$$\mathcal{D}'_0 := g, \alpha_h, \alpha_v, \{(g^{\beta_i}, g^{-\gamma_i}, c_i)\}_{i\in[q]} \text{ for } \begin{cases} \alpha, \beta_i, \tilde{\gamma}_i \leftarrow\$ \mathbb{Z}_p, \\ \gamma_i := (\tilde{\gamma}_i + \alpha_h \cdot \beta_i) \cdot \alpha_v^{-1} + c_i \cdot \alpha \end{cases}$$

$$\mathcal{D}'_1 := g, \alpha_h, \alpha_v, \{(g^{\beta_i}, g^{-\gamma_i}, c_i)\}_{i\in[q]} \text{ for } \begin{cases} \alpha, \beta_i \leftarrow\$ \mathbb{Z}_p, \\ \gamma_i := (\alpha \cdot \beta_i + \alpha_h \cdot \beta_i) \cdot \alpha_v^{-1} + c_i \cdot \alpha. \end{cases}$$

Proof. Fix $\alpha_h, \alpha_v \in \mathbb{Z}_p^*$ and $\boldsymbol{c} := [c_1, \ldots, c_q]$. Given a sample $(g, g^\alpha, \{(g^{\beta_i}, g^{\gamma_i})\}_{i\in[q]})$ from $\mathcal{D}_b$ for either $b \in \{0, 1\}$ as defined in Lemma 1, we can get a sample from $\mathcal{D}'_b$ as follows:

1. For all $i \in [q]$, define $g^{\beta'_i} := g^{\beta_i}$, and compute
$$g^{\gamma'_i} := (g^{\gamma_i})^{\alpha_v^{-1}} \cdot (g^{\beta_i})^{\alpha_h \cdot \alpha_v^{-1}} \cdot (g^\alpha)^{c_i}. \tag{14}$$
2. Output the tuple $(g, \alpha_h, \alpha_v, \{(g^{\beta'_i}, g^{-\gamma'_i}, c_i)\}_{i\in[q]})$.

We now analyze the distribution from which the tuple in step (2) above is sampled. When $b = 0$, then for all $i \in [q]$, $g^{-\gamma'_i}$ is uniformly random. Thus, the tuple in step (2) above is a sample from $\mathcal{D}'_0$. Similarly, when $b = 1$, then $g^{\gamma_i} = g^{\alpha \cdot \beta_i}$ for all $i \in [q]$. Thus, $g^{-\gamma'_i}$ in equation (14) is correctly distributed as in $\mathcal{D}'_1$. Consequently, if a PPT adversary $\mathcal{A}$ can distinguish between a sample from $\mathcal{D}'_0$ and $\mathcal{D}'_1$ with probability ε, then $\mathcal{A}$ can distinguish between $\mathcal{D}_1$ and $\mathcal{D}_2$ with probability ε. From Lemma 1, we then get $\varepsilon \leq \varepsilon_{\mathsf{ddh}} + 1/p$. □

It is easy to see that in game $\mathbf{G}_9$ we use a sample from the distribution $\mathcal{D}'_0$ to program the random oracles $\mathsf{G}_0, \mathsf{G}_1, \mathsf{F}_0$ and F_1, whereas in game $\mathbf{G}_{10}$ we use a sample from the distribution $\mathcal{D}'_0$. Therefore, the advantage of $\mathcal{A}$ in distinguishing between these two games is at most $\varepsilon_{\mathsf{ddh}} + 1/p$, and we get:

$$|\Pr[\mathbf{G}_9 \Rightarrow 1] - \Pr[\mathbf{G}_{10} \Rightarrow 1]| \leq \varepsilon_{\mathsf{ddh}} + \frac{1}{p}.$$

GAME $\mathbf{G}_{11}$: This game is identical to $\mathbf{G}_{11}$, except that for each honest signer we use simulated proofs for Sig_2 messages instead of actual NIZK proofs. Looking ahead, we switch to simulated NIZK proofs in this game to later argue in game $\mathbf{G}_{12}$ that the NIZK proofs do not reveal any information about the secret signing keys of honest signers. This is crucial to argue the indistinguishability between games $\mathbf{G}_{13}$ and $\mathbf{G}_{14}$. During NIZK simulation, we program the random oracle H_{FS} on $(X_A, X_B, X_{\mathsf{pk}}, A, B, \mathsf{pk}, \vec{\boldsymbol{\mu}}, \rho)$ at a choice of our challenge (see §5.1). Note that the NIZK protocol we use is perfect honest-verifier zero-knowledge (HVZK). Hence, conditioned on the successful programming of the random oracle H_{FS}, the view of the $\mathcal{A}$ in games $\mathbf{G}_{10}$ and $\mathbf{G}_{11}$ are identically distributed. Next, we will analyze the probability that we fail to program H_{FS} on desired inputs. We fail to program H_{FS} on desired inputs, if the adversary has already queried H_{FS} on this particular input. Next, we formally analyze this failure probability.

Let Coll be the event that at least one of our H_{FS} query collides with $\mathcal{A}$'s H_{FS} query. Since, $\mathcal{A}$'s view in games $\mathbf{G}_{10}$ and $\mathbf{G}_{11}$ only differ if Coll occurs (i.e, $\Pr[\mathbf{G}_{10} \Rightarrow 1 | \neg\mathsf{Coll}] = \Pr[\mathbf{G}_{11} = 1 | \neg\mathsf{Coll}]$), we trivially get:

$$|\Pr[\mathbf{G}_{10} \Rightarrow 1] - \Pr[\mathbf{G}_{11} \Rightarrow 1]| \leq \Pr[\mathsf{Coll}].$$

We now analyze the probability of event Coll. For each NIZK simulation, the game programs H_{FS} at a input $\mathsf{stm} := (X_A, X_B, X_{\mathsf{pk}}, A, B, \mathsf{pk}, \vec{\boldsymbol{\mu}}, \rho)$ to output a challenge of its choice and aborts if $\mathsf{H}_{\mathsf{FS}}(\mathsf{stm})$ is already defined. Since $X_A, X_B, X_{\mathsf{pk}} \leftarrow\$ \mathbb{G}$ are sampled uniformly at random and hidden from $\mathcal{A}$ (before we output

the NIZK proof), the probability that the game aborts is at most q_{FS}/p^3. By a union bound over all signing queries, we get $\Pr[\mathsf{Coll}] \leq q_s \cdot q_{\mathsf{FS}}/p^3 = \varepsilon_{\mathsf{coll}}$. Hence, we get: $|\Pr[\mathbf{G}_{10} \Rightarrow 1] - \Pr[\mathbf{G}_{11} \Rightarrow 1]| \leq \varepsilon_{\mathsf{coll}}$.

GAME $\mathbf{G}_{12}$: This game is identical to $\mathbf{G}_{11}$, except we change how we sample α. More precisely, in this game, we sample α by first sampling an uniform random $u \leftarrow\$ \mathbb{Z}_p$ and compute $\alpha := \alpha_h + \alpha_v u$. Since $\alpha_v \neq 0$ and u is uniformly random and independent of the rest of the values, it follows that α in game $\mathbf{G}_{12}$ is also uniformly random and independent. Thus: $\Pr[\mathbf{G}_{11} \Rightarrow 1] = \Pr[\mathbf{G}_{12} \Rightarrow 1]$.

GAME $\mathbf{G}_{13}$: In this game, we change how we sample the signing keys. To illustrate our modification, we will distinguish between the signing key polynomials of game $\mathbf{G}_{12}$ and $\mathbf{G}_{13}$. More precisely, let $s_{12}(x), r_{12}(x), u_{12}(x)$ and $s_{13}(x), r_{13}(x), u_{13}(x)$ be the signing key polynomials in game $\mathbf{G}_{12}$ and game $\mathbf{G}_{13}$, respectively. Then, in game $\mathbf{G}_{12}$ we sample the signing key polynomial $s_{13}(x) := s_{12}(x) + \alpha$ for α we describe in the previous game. The other two signing key polynomials remain unchanged, i.e., $r_{13}(x) := r_{12}(x)$ and $u_{13}(x) := u_{12}(x)$.

Observe that for any fixed α, since $s_{12}(x)$ is a random degree t polynomial, $s_{13}(x) := s_{12}(x) + \alpha$ is also a random degree t polynomial. Hence, $\mathcal{A}$'s view in game $\mathbf{G}_{12}$ is identically distributed to its view in game $\mathbf{G}_{13}$, and we get:

$$\Pr[\mathbf{G}_{12} \Rightarrow 1] = \Pr[\mathbf{G}_{13} \Rightarrow 1].$$

GAME $\mathbf{G}_{14}$: In this game, we change how we sample the signing keys again. More precisely, we sample signing key polynomials such that $s_{14}(x) := s_{12}(x)$, $r_{14}(x) := r_{12}(x) + 1$ and $u_{14}(x) := u_{12}(x) + u$, for uniformly random $u \in \mathbb{Z}_p$ we used to define $\alpha = \alpha_h + \alpha_v u$.

The indistinguishability between $\mathcal{A}$'s view in game $\mathbf{G}_{13}$ and $\mathbf{G}_{14}$ is another crucial step of our proof. To prove this, we will use Lemma 2 and the H-coefficient technique [Pat08, CS14]. We defer the proof of the lemma to Appendix B.2.

Lemma 4. $\Pr[\mathbf{G}_{13} \Rightarrow 1] = \Pr[\mathbf{G}_{14} \Rightarrow 1]$.

Combining everything, we get our main theorem.

Theorem 1 (Adaptively Secure Threshold Schnorr Signature). *For any $n, t \in \mathsf{poly}(\lambda)$ with $t < n$, assuming hardness of* DDH *and* DL *in the group $\mathbb{G}$, and assuming the random oracle model (ROM), any PPT adversary making at most $q_{\mathsf{F}}, q_{\mathsf{G}}, q_{\mathsf{com}}, q_{\mathsf{sig}}, q_{\mathsf{FS}}$ random oracle queries to $\mathsf{F}_0, \mathsf{F}_1, \mathsf{G}_0, \mathsf{G}_1, \mathsf{H}_{\mathsf{com}}, \mathsf{H}_{\mathsf{sig}}, \mathsf{H}_{\mathsf{FS}}$, respectively, and at most q_s signing queries, wins the $\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ game for our threshold Schnorr signature* Gargos *with probability at most ε_σ where:*

$$\varepsilon_\sigma \leq \frac{q_{\mathsf{com}}^2 + q_s^2 + q_s}{|\mathcal{R}|} + \frac{q_{\mathsf{F}}^2 + q_{\mathsf{G}}^2 + q_{\mathsf{FS}}^2 + q_s^2}{p} + \frac{q_s q_{\mathsf{F}} + q_s^2}{2^\lambda} + \sqrt{2 q_{\mathsf{FS}} \cdot \varepsilon_{\mathsf{dl}}}$$
$$+ \frac{q_s(q_{\mathsf{sig}} + q_{\mathsf{com}}) + q_{\mathsf{FS}}}{p} + \frac{q_s q_{\mathsf{FS}}}{p^3} + \frac{q_{\mathsf{sig}}}{p} + \frac{n+1}{p} + \varepsilon_{\mathsf{ddh}} + \sqrt{q_{\mathsf{sig}} \cdot \varepsilon_{\mathsf{dl}}}.$$

Here, $\varepsilon_{\mathsf{dl}}$ and $\varepsilon_{\mathsf{ddh}}$ are the advantages of an adversary running in $T \cdot \mathsf{poly}(\lambda, n)$ time in breaking the DL *and* DDH *assumption in $\mathbb{G}$, respectively.*

The proof of Theorem 1 follows the same approach as in the proof of [BDLR24, Theorem 1], and therefore we omit it here.

Acknowledgments

This work is funded by the National Science Foundation award #2240976, and by the European Union, ERC-2023-StG-101116713. Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

ADN06. Jesús F Almansa, Ivan Damgård, and Jesper Buus Nielsen. Simplified threshold rsa with adaptive and proactive security. In *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2006.

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

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

BDLR24. Renas Bacho, Sourav Das, Julian Loss, and Ling Ren. Glacius: Threshold schnorr signatures from ddh with full adaptive security. *Cryptology ePrint Archive*, 2024.

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

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

BLL+22. Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ros. *J. Cryptol.*, 35(4), October 2022.

BLSW24. Renas Bacho, Julian Loss, Gilad Stern, and Benedikt Wagner. Harts: High-threshold, adaptively secure, and robust threshold schnorr signatures. *Cryptology ePrint Archive*, 2024.

BLT+24. Renas Bacho, Julian Loss, Stefano Tessaro, Benedikt Wagner, and Chenzhi Zhu. Twinkle: Threshold signatures from ddh with full adaptive security. In *Advances in Cryptology-EUROCRYPT 2024: 43th Annual International Conference on the Theory and Applications of Cryptographic Techniques.*, 2024.

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

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

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

BTZ22. Mihir Bellare, Stefano Tessaro, and Chenzhi Zhu. Stronger security for non-interactive threshold signatures: Bls and frost. *Cryptology ePrint Archive*, 2022.

BW24. Renas Bacho and Benedikt Wagner. Tightly secure threshold signatures over pairing-free groups. Cryptology ePrint Archive, Paper 2024/1557, 2024.

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

CGRS23. Hien Chu, Paul Gerhart, Tim Ruffing, and Dominique Schröder. Practical schnorr threshold signatures without the algebraic group model. In *Annual International Cryptology Conference*, pages 743–773. Springer, 2023.

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

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

CS14. Shan Chen and John Steinberger. Tight security bounds for key-alternating ciphers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 327–350. Springer, 2014.

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

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

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.

DR24. Sourav Das and Ling Ren. Adaptively secure bls threshold signatures from ddh and co-cdh. In *Advances in Cryptology–CRYPTO 2024: 44nd Annual International Cryptology Conference, CRYPTO 2024, Santa Barbara, CA, USA*. Springer, 2024.

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

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

GJKR07. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 2007.

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

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

KG21. Chelsea Komlo and Ian Goldberg. Frost: flexible round-optimized schnorr threshold signatures. In *Selected Areas in Cryptography*. Springer, 2021.

KRT24. Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. Adaptively secure 5 round threshold signatures from mlwe/msis and dl with rewinding. In *Annual International Cryptology Conference*, pages 459–491. Springer, 2024.

Lin22. Yehuda Lindell. Simple three-round multiparty schnorr signing with full simulatability. *Cryptology ePrint Archive*, 2022.

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

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

Mak22. Nikolaos Makriyannis. On the classic protocol for MPC schnorr signatures. Cryptology ePrint Archive, Paper 2022/1332, 2022.

MMS+24. Aikaterini Mitrokotsa, Sayantan Mukherjee, Mahdi Sedaghat, Daniel Slamanig, and Jenit Tomy. Threshold structure-preserving signatures: Strong and adaptive security under standard assumptions. In *IACR International Conference on Public-Key Cryptography*, 2024.

Nak08. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

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

Pat08. Jacques Patarin. The "coefficients h" technique. In *International Workshop on Selected Areas in Cryptography*, pages 328–345. Springer, 2008.

PS96. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 387–398. Springer, 1996.

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

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

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

Sha79. Adi Shamir. How to share a secret. *Communications of the ACM*, 1979.

SS01. Douglas R. Stinson and Reto Strobl. Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In Vijay Varadharajan and Yi Mu, editors, *Information Security and Privacy*, pages 417–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

Wag02. David Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, pages 288–304, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

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

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

A Additional Preliminaries

We define the computational assumptions used in our security proof and the generalized forking lemma.

A.1 Computational Assumptions

Assumption 2 (DL) *We say that the discrete logarithm (DL) assumption holds, if for all PPT adversaries $\mathcal{A}$, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := \Pr\left[\mathcal{A}(g, g^{\alpha}) = \alpha \;\middle|\; (\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^{\lambda}),\ \alpha \leftarrow\!\$\ \mathbb{Z}_p\right] = \varepsilon_{\mathsf{dl}}.$$

Assumption 3 (DDH) *We say that the decisional Diffie-Hellman (DDH) assumption holds, if for all PPT adversaries $\mathcal{A}$, the following advantage is negligible:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := \Bigg|&\Pr\left[\mathcal{A}(g, g^{\alpha}, g^{\beta}, g^{\alpha\beta}) = 1 \;\middle|\; \begin{array}{r}(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^{\lambda}),\\ \alpha, \beta \leftarrow\!\$\ \mathbb{Z}_p\end{array}\right]\\ &- \Pr\left[\mathcal{A}(g, g^{\alpha}, g^{\beta}, g^{\gamma}) = 1 \;\middle|\; \begin{array}{r}(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(1^{\lambda}),\\ \alpha, \beta, \gamma \leftarrow\!\$\ \mathbb{Z}_p\end{array}\right]\Bigg| = \varepsilon_{\mathsf{ddh}}.\end{aligned}$$

A.2 Generalized Forking Lemma

We recall the generalized forking lemma [PS96, BN06].

Lemma 5. *Let $q \geq 1$ be an integer, and H be a set. Let $\mathcal{A}$ be a randomized algorithm that on input $x, h_1, h_2, \ldots, h_q$ outputs a pair (k, aux), where $k \in [0, q]$ and aux is a side output. Let IG be a randomized algorithm that generates x. The accepting probability of $\mathcal{A}$ is defined as:*

$$\mathsf{acc} = \Pr_{x \leftarrow \mathsf{IG}, h_1, h_2, \ldots, h_q \leftarrow\!\$ H}[(k, \mathsf{aux}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q) : k \neq 0].$$

Consider algorithm $\mathsf{Fork}^{\mathcal{A}}$ described in Figure 5. The accepting probability of $\mathsf{Fork}^{\mathcal{A}}$ is defined as:

$$\mathsf{frk} := \Pr_{x \leftarrow \mathsf{IG}}[\nu \leftarrow \mathsf{Fork}^{\mathcal{A}}(x) : \nu \neq \bot].$$

Then, we have:

$$\mathsf{frk} \geq \mathsf{acc}\left(\frac{\mathsf{acc}}{q} - \frac{1}{|H|}\right) \implies \mathsf{acc} \geq \frac{q}{|H|} + \sqrt{q \cdot \mathsf{frk}}.$$

Algorithm $\mathsf{Fork}^{\mathcal{A}}(x)$:

1: Pick the random coins ρ of $\mathcal{A}$ at random
2: $\{h_1, \ldots, h_q\} \leftarrow\!\$\ H^q$
3: $(k, \mathsf{aux}) \leftarrow \mathcal{A}(x, h_1, \ldots, h_q)$
4: **if** $k = 0$: **return** $\bot$
5: $\{h'_k, \ldots, h'_q\} \leftarrow\!\$\ H^q$
6: $(k', \mathsf{aux}') \leftarrow \mathcal{A}(x, h_1, \ldots, h_{k-1}, h'_k, \ldots, h'_q)$
7: **if** $k \neq k'$: **return** $\bot$
8: **return** $(k, \mathsf{aux}, \mathsf{aux}')$

Fig. 5: The generalized forking algorithm.

Input: Public parameters $(g, h, v) \in \mathbb{G}^3$, signing key polynomials $s(\cdot), r(\cdot), u(\cdot)$, randomness for random oracle programming $\{h_1, h_2, \ldots, h_q\} \leftarrow\$ \mathbb{Z}_p$.

KGen simulation

1. Use (g, h, v) as the public parameters and $s(\cdot), r(\cdot), u(\cdot)$ as the signing key polynomials.

Corruption simulation:

2. When $\mathcal{A}$ corrupts a signer $i \in \mathcal{H}$ if $|\mathcal{C}| < t$:
   (a) Update $\mathcal{H} := \mathcal{H} \setminus \{i\}$ and $\mathcal{C} := \mathcal{C} \cup \{i\}$.
   (b) Faithfully reveals the internal state of signer i to $\mathcal{A}$.

Simulating random oracle queries: For each random oracle query on some input x, use the next unused random value from the input $\{h_1, h_2, \ldots, h_q\}$ to program the random oracle.

Simulating signing protocol for any signing session:

3. Follow the honest protocol for all honest signers. Additionally, maintain the following state locally.
4. For any honest signer $i \in \mathcal{H}$, let $\vec{\boldsymbol{\mu}}_{(i)}$ be the Sig_1 message received by signer i. For readability, we will drop the subscript (i) and write $\vec{\boldsymbol{\mu}}_{(i)}$ simply as $\vec{\boldsymbol{\mu}}$.
5. Let $\vec{\boldsymbol{\mu}} = [\mu_j]_{j \in \mathsf{SS}}$. For each μ_j with $j \in \mathcal{C} \cap \mathsf{SS}$, extract (ρ_j, B_j) using the observability of the random oracle $\mathsf{H}_{\mathsf{com}}$ and compute A'_j as:

$$A'_j := \Big(B_j \cdot \mathsf{F}_0(\rho_j)^{-r(j)} \cdot \mathsf{F}_1(\rho_j)^{-u(j)} \cdot \mathsf{G}_0(\vec{\boldsymbol{\mu}})^{r(j)} \cdot \mathsf{G}_1(\vec{\boldsymbol{\mu}})^{u(j)}\Big)^{L_{j,\mathsf{SS}}}. \tag{15}$$

6. For signer j, let $(A_j, \rho_j, B_j, \pi_j)$ be the second-round message signer j sends to signer i in that session. Then, if $A_j \neq A'_j$ and π_j is a verifying NIZK proof, do the following:
   (a) Let $\pi_j := (X_A, X_B, X_{\mathsf{pk}}, z_a, z_s, z_r, z_u)$ be the NIZK proof.
   (b) Identify the random oracle query to H_{FS} with input $(X_A, X_B, X_{\mathsf{pk}}, A, B, \mathsf{pk}_j, \vec{\boldsymbol{\mu}}, \rho_j)$. Let idx be such that:

$$\mathsf{H}_{\mathsf{FS}}(X_A, X_B, X_{\mathsf{pk}}, A, B, \mathsf{pk}_j, \vec{\boldsymbol{\mu}}, \rho_j) := h_{\mathsf{idx}} \tag{16}$$

   (c) **return** $(\mathsf{idx}, w := (j, z_s, z_r, z_u))$.
7. If event Neq does not occur during the interaction with $\mathcal{A}$, then **return** $(0, \varepsilon)$.

Fig. 6: Description of Algorithm $\mathcal{B}$ that simulates game $\mathbf{G}_3$ to the adversary $\mathcal{A}$.

B Deferred Proofs

B.1 Proof of Lemma 3

Proof. To prove this lemma, we will rely on generalized forking lemma [PS96, BN06]. More specifically, given an adversary $\mathcal{A}$ that can cause the event Neq to happen, we will build a "wrapping" algorithm $\mathcal{B}$ (see Figure 6) which runs $\mathcal{A}$ and returns information regarding the bad event Neq. Algorithm $\mathcal{B}$ simulates all the random oracles with uniformly random outputs. We then use $\mathcal{B}$ to construct an algorithm $\mathcal{B}_{\mathsf{dl}}$ (see Figure 8) that first runs the forking algorithm $\mathsf{Fork}^{\mathcal{B}}$ (see Figure 7) which forks $\mathcal{B}$ with respect to H_{FS} query. Algorithm $\mathcal{B}_{\mathsf{dl}}$ then uses the output of the $\mathsf{Fork}^{\mathcal{B}}$ algorithm to solve for discrete logarithm in $\mathbb{G}$.

Description of Algorithm $\mathcal{B}$ (Figure 6). $\mathcal{B}$ takes as input the public parameters (g, h, v), the signing keys $(s(\cdot), r(\cdot), u(\cdot))$, and a vector $\{h_1, \ldots, h_{q_{\mathsf{FS}}}\}$ of uniformly random field elements. $\mathcal{B}$ then interacts with $\mathcal{A}$ with these inputs, where $\mathcal{B}$ uses $\{h_1, \ldots, h_{q_{\mathsf{FS}}}\}$ to program the random oracle H_{FS}. Simultaneously, $\mathcal{B}$ also locally checks for the event Neq (step 5 in Figure 6). Let $i \in \mathcal{H}$ be the honest party at which the event Neq happens. Here on, in the rest of the analysis of event Neq we will drop the subscript (i).

When the event Neq occurs, $\mathcal{B}$ identifies the H_{FS} query associated with the event Neq. Let j be the malicious signer causing the event Neq. Also, let $\pi_j = (X_A, X_B, X_{\mathsf{pk}}, z_a, z_s, z_r, z_u)$ and $(\mathsf{pk}_j, A_j, B_j, \vec{\boldsymbol{\mu}}, \rho_j)$ be the NIZK proof and the statement associated with the event Neq. Then, $\mathcal{B}$ finds the index idx such that $\mathcal{B}$ programmed the H_{FS} query on input $(X_A, X_B, X_{\mathsf{pk}}, \mathsf{pk}, A, B, \vec{\boldsymbol{\mu}}, \rho_j)$ with h_{idx}, and returns the tuple $(\mathsf{idx}, (j, z_s, z_r, z_u))$ as its output.

Algorithm $\mathsf{Fork}^{\mathcal{B}}(x := (g, h, v, s(\cdot), r(\cdot), u(\cdot)))$:

1: **sample** randomness tape ζ for $\mathcal{B}$
2: **sample** $h_1, h_2, \ldots, h_q \leftarrow\$ \mathbb{Z}_p$
3: **let** $(\mathsf{idx}, w) := \mathcal{B}(x, \{h_1, h_2, \ldots, h_q\}; \zeta)$
4: **if** $\mathsf{idx} = 0$: **return** $(0, \perp, \perp)$
5: **sample** $h'_{\mathsf{idx}}, \ldots, h'_q \leftarrow\$ \mathbb{Z}_p$
6: **let** $(\mathsf{idx}', w') := \mathcal{B}(x, \{h_1, h_2, h_{\mathsf{idx}-1}, h'_{\mathsf{idx}}, \ldots, h'_q\}; \zeta)$
7: **if** $\mathsf{idx} = \mathsf{idx}' \vee h_{\mathsf{idx}} \neq h'_{\mathsf{idx}}$: **return** $\perp$
8: **let** $\mathsf{out} := (h_{\mathsf{idx}}, w),\ \mathsf{out}' := (h'_{\mathsf{idx}}, w')$
9: **return** $(1, \mathsf{out}, \mathsf{out}')$

Fig. 7: Description of Algorithm $\mathsf{Fork}^{\mathcal{B}}$.

Algorithm $\mathcal{B}_{\mathsf{dl}}(\mathbb{G}, p, g, y)$:

1: **sample** values $\alpha \leftarrow\$ \mathbb{Z}_p^*$ and $\theta \leftarrow\$ \{0, 1\}$
2: **if** $\theta = 0$: $(h, v) := (y, g^{\alpha})$; **otherwise**, $(h, v) := (g^{\alpha}, y)$
3: **sample** signing keys polynomials $s(\cdot), r(\cdot), u(\cdot)$ as per protocol specification
4: **let** $(\mathsf{val}, \mathsf{out}, \mathsf{out}') \leftarrow \mathsf{Fork}^{\mathcal{B}}(g, h, v, s(\cdot), r(\cdot), u(\cdot))$
5: **if** $\mathsf{val} = 0$: **return** $\perp$
6: **parse** $(e, (j, z_s, z_r, z_u)) := \mathsf{out}$ and $(e', (j', z'_s, z'_r, z'_u)) := \mathsf{out}'$
7: Compute s_j, r_j, u_j as

$$s_j := \frac{z_s - z'_s}{e - e'}, \qquad r_j := \frac{z_r - z'_r}{e - e'}, \qquad u_j := \frac{z_u - z'_u}{e - e'}. \tag{17}$$

8: Let $\delta_s := s(j) - s_j$, $\delta_r := r(j) - r_j$, and $\delta_u := u(j) - u_j$
// Let $\alpha_h, \alpha_v \in \mathbb{Z}_p$ be such that $h = g^{\alpha_h}$ and $v = g^{\alpha_v}$
9: **if** $\theta = 0 \wedge \delta_r \neq 0$:
10: **return** $(-\delta_s - \alpha_v \delta_u) \cdot \delta_r^{-1}$ as the DL solution
11: **else if** $\theta = 1 \wedge \delta_u \neq 0$:
12: **return** $(-\delta_s - \alpha_h \delta_r) \cdot \delta_u^{-1}$ as the DL solution
13: **return** $\perp$

Fig. 8: Description of Algorithm $\mathcal{B}_{\mathsf{dl}}$ solves discrete logarithm in $\mathbb{G}$.

Description of Algorithm $\mathsf{Fork}^{\mathcal{B}}$ (Figure 7). $\mathsf{Fork}^{\mathcal{B}}$ takes as input the public parameters (g, h, v) and the secret signing keys $(s(\cdot), r(\cdot), u(\cdot))$. $\mathsf{Fork}^{\mathcal{B}}$ samples the randomness tape ζ for $\mathcal{B}$ and the random oracle outputs $\{h_1, \ldots, h_{q_{\mathsf{FS}}}\}$. Next, $\mathsf{Fork}^{\mathcal{B}}$ runs $\mathcal{B}$ on these input. Let (idx, w) be the output of $\mathcal{B}$. If $\mathsf{idx} = 0$, then $\mathsf{Fork}^{\mathcal{B}}$ returns $(0, \perp, \perp)$. Otherwise, $\mathsf{Fork}^{\mathcal{B}}$ samples $h'_{\mathsf{idx}}, \ldots, h_{q_{\mathsf{FS}}}$ uniformly at random. $\mathsf{Fork}^{\mathcal{B}}$ then runs the second execution of $\mathcal{B}$ by changing the random oracle programming since the index idx with $\{h'_{\mathsf{idx}}, \ldots, h'_{q_{\mathsf{FS}}}\}$.

Let (idx', w') be the output of $\mathcal{B}$ from the second execution. $\mathsf{Fork}^{\mathcal{B}}$ then checks whether $\mathsf{idx} = \mathsf{idx}'$ and $h_{\mathsf{idx}} \neq h'_{\mathsf{idx}}$. If both of these conditions hold $\mathsf{Fork}^{\mathcal{B}}$ returns $(1, \mathsf{out}, \mathsf{out}')$ where $\mathsf{out} := (h_{\mathsf{idx}}, w)$ and $\mathsf{out}' := (h'_{\mathsf{idx}}, w')$. Otherwise, $\mathsf{Fork}^{\mathcal{B}}$ returns $(0, \perp, \perp)$.

Description of Algorithm $\mathcal{B}_{\mathsf{dl}}$ (Figure 8). $\mathcal{B}_{\mathsf{dl}}$ takes as input $(\mathbb{G}, p, g, y)$: the description of the group $\mathbb{G}$ of order p, a generator g and a uniformly random element $y \in \mathbb{G}$. $\mathcal{B}_{\mathsf{dl}}$ then samples uniformly random $\alpha \leftarrow\$ \mathbb{Z}_p$ and a bit $\theta \leftarrow \{0, 1\}$. Next, depending upon the value of θ, $\mathcal{B}_{\mathsf{dl}}$ sets the public parameters (g, h, v) in two different manner. More precisely, if $\theta = 0$, $\mathcal{B}_{\mathsf{dl}}$ sets $(g, h, v) := (g, y, g^{\alpha})$, otherwise $\mathcal{B}_{\mathsf{dl}}$ sets $(g, h, v) := (g, g^{\alpha}, y)$.

Next, $\mathcal{B}_{\mathsf{dl}}$ samples the signing key polynomials $s(\cdot), r(\cdot), u(\cdot)$ as per the honest protocol. $\mathcal{B}_{\mathsf{dl}}$ then runs $\mathsf{Fork}^{\mathcal{B}}$ with $(g, h, v, s(\cdot), r(\cdot), u(\cdot))$ as input. Let $(\mathsf{val}, \mathsf{out}, \mathsf{out}')$ be the output of $\mathsf{Fork}^{\mathcal{B}}$. If $\mathsf{val} = 0$, $\mathcal{B}_{\mathsf{dl}}$ returns

$\perp$. Otherwise, let $\mathsf{out} = (e, (j, z_s, z_r, z_u))$ and $\mathsf{out}' = (e', (j', z'_s, z'_r, z'_u))$. $\mathcal{B}_{\mathsf{dl}}$ computes (s_j, r_j, u_j) as:

$$s_j := \frac{z_s - z'_s}{e - e'}, \qquad r_j := \frac{z_r - z'_r}{e - e'}, \qquad u_j := \frac{z_u - z'_u}{e - e'}. \tag{18}$$

Let $\delta_s := s(j) - s_j$, $\delta_r := r(j) - r_j$, and $\delta_u := u(j) - u_j$. Also, for notational convenience, let $h = g^{\alpha_h}$ and $v = g^{\alpha_v}$. Then, if $(\theta = 0)$ and $\delta_r \neq 0$, $\mathcal{B}_{\mathsf{dl}}$ outputs $(-\delta_s - \alpha_v \delta_u) \cdot \delta_u^{-1}$ as the DL solution. Alternatively, if $\theta = 1$ and $\delta_u \neq 0$, $\mathcal{B}_{\mathsf{dl}}$ outputs $(-\delta_s - \alpha_h \delta_r) \cdot \delta_u^{-1}$ as the DL solution.

Analysis of $\mathcal{B}_{\mathsf{dl}}$. Let ε be the probability that $\mathsf{Fork}^{\mathcal{B}}$ outputs $(1, \mathsf{out}, \mathsf{out}')$. Also, let $\varepsilon_{\mathsf{dl}}$ be the probability that $\mathcal{B}_{\mathsf{dl}}$ outputs the discrete logarithm of y. Then, we will next argue that $\varepsilon_{\mathsf{dl}} \geq \varepsilon/2$. Also, let $\varepsilon_{\mathsf{neq}}$ be the probability of the event Neq in game $\mathbf{G}_2$. Then, from the local forking lemma, we get that:

$$\varepsilon \geq \frac{\varepsilon_{\mathsf{neq}}^2}{q_{\mathsf{FS}}} - \frac{\varepsilon_{\mathsf{neq}}}{p}.$$

Therefore, by combining the above, we will get:

$$\varepsilon_{\mathsf{dl}} \geq \frac{1}{2} \cdot \left(\frac{\varepsilon_{\mathsf{neq}}^2}{q_{\mathsf{FS}}} - \frac{\varepsilon_{\mathsf{neq}}}{p} \right) \implies \varepsilon_{\mathsf{neq}} \leq \frac{q_{\mathsf{FS}}}{p} + \sqrt{2 \cdot q_{\mathsf{FS}} \cdot \varepsilon_{\mathsf{dl}}}. \tag{19}$$

Note that $\mathsf{Fork}^{\mathcal{B}}$ outputting $(1, \mathsf{out}, \mathsf{out}')$ implies that the event Neq happens for the first time during $\mathcal{B}$'s interaction with $\mathcal{A}$ for the idx-th H_{FS} query for both execution of $\mathcal{B}$. Since, $\mathcal{A}$'s view in both execution is identical until the idx-th query, it implies that the input to the idx-th H_{FS} is identical in both execution. Moreover, $\mathcal{A}$ outputs valid NIZK proof π_j and π'_j for the same statement in both the execution.

Let $(X_A, X_B, X_{\mathsf{pk}}, A, B, \mathsf{pk}_j, \vec{\boldsymbol{\mu}}, \rho_j)$ be the idx-th H_{FS} input. This implies, the $\mathsf{Fork}^{\mathcal{B}}$ outputs $\mathsf{out} := (e, (j, z_s, z_r, z_u))$ and $\mathsf{out}' = (e', (j', z'_s, z'_r, z'_u))$ satisfy that:

$$g^{z_s} h^{z_r} v^{z_u} = X_{\mathsf{pk}} \cdot \mathsf{pk}_j^c, \qquad g^{z'_s} h^{z'_r} v^{z'_u} = X_{\mathsf{pk}} \cdot \mathsf{pk}_j^{c'}.$$

Therefore, we get that:

$$g^{s_j} h^{r_j} v^{u_j} = \mathsf{pk}_j = g^{s(j)} h^{r(j)} v^{u(j)},$$

where (s_j, r_j, u_j) are the values $\mathcal{B}_{\mathsf{dl}}$ computes in equation (18), and $(s(j), r(j), u(j))$ are the signing keys of party j as per the protocol specification. This implies that:

$$g^{s_j - s(j)} h^{r_j - r(j)} v^{u_j - u(j)} = 1_{\mathbb{G}} = g^{\delta_s} h^{\delta_r} v^{\delta_u}. \tag{20}$$

We will next argue that $(\delta_r, \delta_u) \neq (0, 0)$. For the sake of contradiction, assume that $(\delta_r, \delta_u) = (0, 0)$. Also, let $\pi_j = (X_A, X_B, X_{\mathsf{pk}}, , z_a, z_s, z_r, z_u)$ and $\pi'_j = (X_A, X_B, X_{\mathsf{pk}}, z'_a, z'_s, z'_r, z'_u)$ be the NIZK proofs, $\mathcal{A}$ outputs in its two execution, respectively. Then, for (z_a, z_r, z_u) and (z'_a, z'_r, z'_u) it holds that:

$$\begin{aligned} g^{z_a} \cdot \mathsf{H}_0(\vec{\boldsymbol{\mu}})^{z_r} \cdot \mathsf{F}_0(\vec{\boldsymbol{\mu}})^{z_r} &= X_B \cdot A^c, \qquad & g^{z_a} \cdot \mathsf{F}_0(\rho_j)^{z_r} \cdot \mathsf{F}_0(\rho_j)^{z_r} &= X_B \cdot B^c, \\ g^{z'_a} \cdot \mathsf{H}_0(\vec{\boldsymbol{\mu}})^{z'_r} \cdot \mathsf{H}_1(\vec{\boldsymbol{\mu}})^{z'_u} &= X_B \cdot B^{c'}, \qquad & g^{z'_a} \cdot \mathsf{F}_0(\rho_j)^{z'_r} \cdot \mathsf{F}_0(\rho_j)^{z'_u} &= X_B \cdot B^{c'}. \end{aligned}$$

Let $a' := (z_a - z'_a)/(e - e')$, for the corresponding H_{FS} output e and e', respectively.

$$g^{a'} \cdot \mathsf{F}_0(\rho_j)^{r_j} \cdot \mathsf{F}_1(\rho_j)^{u_j} = B, \qquad g^{a'} \cdot \mathsf{H}_0(\vec{\boldsymbol{\mu}})^{r_j} \cdot \mathsf{H}_1(\vec{\boldsymbol{\mu}})^{u_j} = A. \tag{21}$$

However, since by our assumption $(\delta_r, \delta_u) = 0$, we have $(r_j, u_j) = (r(j), u(j))$, and hence

$$g^{a'} \cdot \mathsf{F}_0(\rho_j)^{r(j)} \cdot \mathsf{F}_1(\rho_j)^{u(j)} = B, \qquad g^{a'} \cdot \mathsf{H}_0(\vec{\boldsymbol{\mu}})^{r(j)} \cdot \mathsf{H}_1(\vec{\boldsymbol{\mu}})^{u(j)} = A. \tag{22}$$

Thus, from equation (22), we get that:

$$B \cdot \mathsf{F}_0(\rho_j)^{-r(j)} \cdot \mathsf{F}_1(\rho_j)^{-u(j)} \cdot \mathsf{H}_0(\vec{\boldsymbol{\mu}})^{r(j)} \cdot \mathsf{H}_1(\vec{\boldsymbol{\mu}})^{u(j)} = A.$$

Equation (23) implies that the event Neq will not occur for the idx-th $\mathsf{H_{FS}}$ query. This is a contradiction to the fact that $\mathcal{B}$ outputs the idx-th index.

Say $h = g^{\alpha_h}$ and $v = g^{\alpha_v}$ for some $\alpha_h, \alpha_v \in \mathbb{Z}_p$. Then, equation (20), implies that $\delta_s + \delta_r\alpha_h + \delta_u\alpha_v = 0$. If either δ_r or δ_u is non-zero, then $\mathcal{B}_{\mathsf{dl}}$ computes α_h or α_v, respectively, as:

$$\delta_r \neq 0 \Rightarrow \alpha_h = (-\delta_s - \alpha_v\delta_u) \cdot \delta_r^{-1}, \qquad \delta_u \neq 0 \Rightarrow \alpha_v = (-\delta_s - \alpha_h\delta_r) \cdot \delta_u^{-1}.$$

Now, note that $\mathcal{B}_{\mathsf{dl}}$ will be able to compute the discrete logarithm of y if either of the following happens: (i) $\theta = 0$ and $\delta_r \neq 0$; and (ii) $\theta = 0$ and $\delta_u \neq 0$. This implies that:

$$\varepsilon_{\mathsf{dl}} \geq \Pr[\theta = 0 \wedge \delta_r \neq 0] + \Pr[\theta = 1 \wedge \delta_u \neq 0]. \tag{23}$$

Note that the view of $\mathsf{Fork}^{\mathcal{B}}$ is identically distributed for both $\theta = 0$ and $\theta = 1$, and hence the value of (δ_r, δ_u) is independent of θ. Therefore we get:

$$\begin{aligned}\varepsilon_{\mathsf{dl}} &\geq \Pr[\theta = 0] \cdot \Pr[\delta_r \neq 0] + \Pr[\theta = 1] \cdot \Pr[\delta_u \neq 0] \\ &= \frac{1}{2} \cdot (\Pr[\delta_r \neq 0] + \Pr[\delta_u \neq 0]) \geq \frac{1}{2} \cdot \Pr[\delta_r \neq 0 \vee \delta_u \neq 0] = \frac{1}{2} \cdot \varepsilon.\end{aligned} \tag{24}$$

Now, combining this with equation (19), we get that:

$$\varepsilon_{\mathsf{neq}} \leq \frac{q_{\mathsf{FS}}}{p} + \sqrt{2 \cdot q_{\mathsf{FS}} \cdot \varepsilon_{\mathsf{dl}}}.$$

Since, the game $\mathbf{G}_1$ and $\mathbf{G}_2$ only differ if the event Neq occurs, we get that:

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \varepsilon_{\mathsf{neq}} \leq \frac{q_{\mathsf{FS}}}{p} + \sqrt{2 \cdot q_{\mathsf{FS}} \cdot \varepsilon_{\mathsf{dl}}}.$$ □

B.2 Proof of Lemma 4

Proof. Let T_0 and T_1 be the random variables denoting the transcripts of $\mathcal{A}$'s interaction in games $\mathbf{G}_{13}$ and $\mathbf{G}_{14}$, respectively. Then, for any potential value τ of T_θ for $\theta \in \{0,1\}$, let $\mathsf{p}_0(\tau)$ and $\mathsf{p}_1(\tau)$ be the interpolation probabilities, i.e., the probabilities of choosing randomness in the respective game that would lead to transcript τ, if the corruption set, the signing queries, and the random oracle queries are fixed in advance. These probabilities depend solely on τ and the game's randomness, and are independent of $\mathcal{A}$. The H-coefficient technique [Pat08] now tells us that, to argue indistinguishability between games $\mathbf{G}_{13}$ and $\mathbf{G}_{14}$, it it sufficient to show that for all possible transcripts τ, $\mathsf{p}_0(\tau) = \mathsf{p}_1(\tau)$. We reiterate that we fix $\mathcal{A}$'s queries when computing the interpolation probabilities.

Since the interpolation probabilities are independent of $\mathcal{A}$, instead of working with the random variables T_0 and T_1, we can work with the marginal transcript random variables we get after fixing $\mathcal{A}$'s queries. Let $W_0 = (X_0, Y_0, f(X_0, Y_0))$ and $W_1 = (X_1, Y_1, f(X_1, Y_1))$ be the marginal transcript random variables of game $\mathbf{G}_{13}$ and $\mathbf{G}_{14}$, respectively, where:

$$\begin{aligned}X_0 :=&(\alpha_h, \alpha_v, s_{12}(0), u, \{s_{13}(i), r_{13}(i), u_{13}(i)\}_{i\in\mathcal{C}}, \{c_{i,j}, \beta_{i,j}, y_{i,j}, B_{i,j}, A_{i,j}, z_{i,j}\}_{i\in\mathsf{SS}_j\cap\mathcal{H}, j\in[q_s]}),\\ X_1 :=&(\alpha_h, \alpha_v, s_{12}(0), u, \{s_{14}(i), r_{14}(i), u_{14}(i)\}_{i\in\mathcal{C}}, \{c_{i,j}, \beta_{i,j}, y_{i,j}, B_{i,j}, A_{i,j}, z_{i,j}\}_{i\in\mathsf{SS}_j\cap\mathcal{H}, j\in[q_s]}).\end{aligned}$$

Y_0 (resp. Y_1) denotes the random variable for:

- The randomness $\{\rho_{i,j}\}_{j\in[q_s], i\in\mathcal{H}\cap\mathsf{SS}_j}$ and the outputs of F_b for both $b \in \{0,1\}$, on all inputs except inputs from $\{\rho_{i,j}\}_{j\in[q_s], i\in\mathcal{H}\cap\mathsf{SS}_j}$.
- The commitments $\{\mu_{i,j}\}_{j\in[q_s], i\in\mathcal{H}\cap\mathsf{SS}_j}$ and the outputs of the G_b for both $b \in \{0,1\}$ on all inputs except on all inputs that are the Sig_1 messages received by honest signers.
- The simulated NIZK proofs of all honest signers and outputs of $\mathsf{H_{FS}}$.

– The outputs of $\mathsf{H}_{\mathsf{sig}}$ on all inputs except for the inputs the game programs $\mathsf{H}_{\mathsf{sig}}$ on by extracting $\hat{A}$ as we show discuss in game $\mathbf{G}_7$.
– All $\{a_{i,j}\}_{j\in[q_s],i\in\mathsf{SS}_j\cap\mathcal{C}}$ values for that the game samples for the j-th signing session before $\mathcal{A}$ corrupts the signer i.

It is easy to see that Y_0 (resp. Y_1) is independent of X_0 (resp. X_1). Also, by design of games $\mathbf{G}_{13}$ and $\mathbf{G}_{14}$, Y_0 is identically distributed as Y_1.

We now argue that for any fixed queries of $\mathcal{A}$, given (X_θ, Y_θ) for either $\theta \in \{0,1\}$, the rest of the transcript is a deterministic function of (X_θ, Y_θ). We also argue that in both games $\mathbf{G}_{13}$ and $\mathbf{G}_{14}$, this (deterministic) function is the same, and we use $f(\cdot,\cdot)$ to denote it, as we describe below:

– Let $\alpha := \alpha_h + u\cdot\alpha_v$. Then, for each $j \in [q_s]$ and $i \in \mathcal{H}\cap\mathsf{SS}_j$, the F_b and G_b outputs for both $b\in\{0,1\}$ on $\rho_{i,j}$ and $\vec{\boldsymbol{\mu}}_{i,j}$, respectively, are deterministic function of $(\alpha_h, \alpha_v, \alpha, \beta_{i,j}, y_{i,j}, c_{i,j}, Y_\theta)$.
– The discrete logarithm of the public key in both games $\mathbf{G}_{13}$ and $\mathbf{G}_{14}$ is the same and is equal to $s_{12}(0)+\alpha$. More precisely, $\mathsf{pk}_{\mathbf{G}_{13}} = g^{s_{12}(0)+\alpha}$ by definition. Also, recall that we have $r_{14}(0) = 1$ and $u_{14}(0) = u$. Therefore,

$$\mathsf{pk}_{\mathbf{G}_{14}} = g^{s_{14}(0)}h^{r_{14}(0)}v^{u_{14}(0)} = g^{s_{12}(0)}hv^u = g^{s_{12}(0)+\alpha_h+\alpha_v u} = g^{s_{12}(0)+\alpha}.$$

  Since $|\mathcal{C}|= t$, given $s_{12}(0)+\alpha$ and the signing keys of signers in $\mathcal{C}$, the threshold public keys of all signers are fixed and a deterministic function of these values.
– For each signing session $j\in[q_s]$, the combined nonces and final signatures are deterministic function of $\{(c_{i,j}, A_{i,j}, z_{i,j})\}_{i\in\mathsf{SS}_j\cap\mathcal{H}}\}$, the signing keys of the corrupt signers, $\mathcal{A}$'s internal state, and Y_θ.

Given Lemma 2, to prove that games $\mathbf{G}_{13}$ and $\mathbf{G}_{14}$ are identically distributed, it remains to show that X_0 and X_1 are identically distributed, i.e., $X_0 \equiv X_1$. Concretely, for any potential value τ of X_θ for $\theta\in\{0,1\}$, that we denote as

$$\tau = \left(\underline{\alpha}_h, \underline{\alpha}_v, \underline{s}, \underline{u}, \{\underline{s}_i, \underline{r}_i, \underline{u}_i\}_{i\in\mathcal{C}}, \{c_{i,j}, \beta_{i,j}, y_{i,j}, \underline{A}_{i,j}, \underline{z}_{i,j}\}_{i\in\mathsf{SS}_j\cap\mathcal{H}, j\in[q_s]}\right),$$

let $\mathsf{p}_\theta(\tau) = \Pr[X_\theta = \tau]$ for either $\theta\in\{0,1\}$.

Analysis of $\mathsf{p}_0(\tau)$. For $\mathsf{p}_0(\tau)$, note that the randomness consists of $\alpha_h, \alpha_v \leftarrow\$\ \mathbb{Z}_p^*$ and $s_{12}(0), u \leftarrow\$\ \mathbb{Z}_p$. To generate a particular transcript τ, we therefore need the identities:

$$\alpha_h = \underline{\alpha}_h, \quad \alpha_v = \underline{\alpha}_v, \quad u = \underline{u}, \quad s_{12}(0) = \underline{s}.$$

Since $(\alpha_h, \alpha_v, s_{12}(0), u)$ are chosen independently, this is true with probability

$$\frac{1}{p-1}\cdot\frac{1}{p-1}\cdot\frac{1}{p}\cdot\frac{1}{p} = \frac{1}{p^2(p-1)^2}. \tag{25}$$

Further, we need to ensure that $\{(s_{12}(i), r_{12}(i), u_{12}(i))\}_{i\in\mathcal{C}} = \{(\underline{s}_i, \underline{r}_i, \underline{u}_i)\}_{i\in\mathcal{C}}$. Since $|\mathcal{C}|= t$, conditioned on $(\alpha_h, \alpha_v, s_{12}(0), u) = (\underline{\alpha}_h, \underline{\alpha}_v, \underline{s}, \underline{u})$, there exists a unique set of three polynomials of degree at most t each, with constant terms being equal to $(\underline{s}+\underline{\alpha}, 0, 0)$ for $\underline{\alpha} := \underline{\alpha}_h + \underline{u}\cdot\underline{\alpha}_v$, such that the above equality holds. Since in game $\mathbf{G}_{13}$, we sample the t additional coefficients of each of these polynomials uniformly at random the equality holds with probability $1/p^{3t}$.

Next, let $\mathsf{NumRhos}$ be the number of times honest singers sample $\rho_{i,j}$ values for some $j\in[q_s]$ and $i\in\mathsf{SS}_j\cap\mathcal{H}$. For each such $\rho_{i,j}$ for $j\in[q_s]$ and $i\in\mathsf{SS}_j\cap\mathcal{H}$, we sample $y_{i,j}$, uniformly at random. Therefore, the probability of obtaining a particular sequence of $\underline{y}_{i,j}$ for $j\in[q_s]$ and $i\in\mathsf{SS}_j\cap\mathcal{H}$ is exactly $1/p^{\mathsf{NumRhos}}$.

Similarly, let NumMsg_1 be the total number of unique first-round messages received by honest signers. Then, for each such distinct first round message $\vec{\boldsymbol{\mu}}_{(i)}$ that $\mathcal{A}$ sends to honest an honest signer $i\in\mathsf{SS}_j\cap\mathcal{H}$, we sample a uniformly random $\beta_{i,j}$. Therefore, the probability of obtaining a particular sequence of $\underline{\beta}_{i,j}$ for $j\in[q_s]$ and $i\in\mathsf{SS}_j\cap\mathcal{H}$ is exactly $1/p^{\mathsf{NumMsg}_1}$.

Finally, let $\mathsf{NumEquiv}$ be the distinct number of equivalence classes formed by the first round messages received by honest signers. Then, for each such equivalence class, we sample a uniformly random challenge $c_{i,j}$. Therefore, the probability of obtaining a particular sequence of $\underline{c}_{i,j}$ for $j \in [q_s]$ and $i \in \mathsf{SS}_j \cap \mathcal{H}$ is exactly $1/p^{\mathsf{NumEquiv}}$.

Combining all the above, we get that the following event

$$\{\rho_{i,j}, \beta_{i,j}, c_{i,j}\}_{j\in[q_s], i\in\mathsf{SS}_j\cap\mathcal{H}} = \{\underline{\rho}_{i,j}, \underline{\beta}_{i,j}, \underline{c}_{i,j}\}_{j\in[q_s], i\in\mathsf{SS}_j\cap\mathcal{H}} \tag{26}$$

happens with probability

$$\frac{1}{p^{\mathsf{NumRhos}}} \cdot \frac{1}{p^{\mathsf{NumMsg}_1}} \cdot \frac{1}{p^{\mathsf{NumEquiv}}}. \tag{27}$$

Next, consider the triples $\{(B_{i,j}, A_{i,j}, z_{i,j})\}_{i\in\mathcal{H}\cap\mathsf{SS}_j, j\in[q_s]}$. For each (i,j), we have:

$$\begin{aligned}
B_{i,j} &= g^{a_{i,j}} \cdot \mathsf{F}_0(\rho_{i,j})^{r_{13}(i)} \cdot \mathsf{F}_1(\rho_{i,j})^{u_{13}(i)} = g^{a_{i,j}} \cdot \mathsf{F}_0(\rho_{i,j})^{r_{12}(i)} \cdot \mathsf{F}_1(\rho_{i,j})^{u_{12}(i)},\\
A_{i,j} &= g^{a_{i,j}} \mathsf{H}_0(m, \vec{\boldsymbol{\mu}}_j)^{r_{13}(i)} \mathsf{H}_1(m, \vec{\boldsymbol{\mu}}_j)^{u_{13}(i)} = g^{a_{i,j}} \mathsf{H}_0(m, \vec{\boldsymbol{\mu}}_{i,j})^{r_{12}(i)} \mathsf{H}_1(m, \vec{\boldsymbol{\mu}}_{i,j})^{u_{12}(i)},\\
z_{i,j} &= a_{i,j} + c_{i,j} \cdot s_{13}(i) = a_{i,j} + c_{i,j} \cdot (s_{12}(i) + \alpha),
\end{aligned}$$

where $a_{i,j} \leftarrow\$ \mathbb{Z}_p$ and $\rho_{i,j}$ are the randomness of honest signer i in the j-th signing session, and $\vec{\boldsymbol{\mu}}_{i,j}$ is the first round message signer i receives during the j-th session.

Observe that, given that everything else is fixed, each triple $(A_{i,j}, B_{i,j}, z_{i,j})$ is determined by the choice of $a_{i,j}$. Thus, the probability that $(B_{i,j}, A_{i,j}, z_{i,j}) = (\underline{B}_{i,j}, \underline{A}_{i,j}, \underline{z}_{i,j})$ holds is equal to the probability that

$$a_{i,j} = \underline{a}_{i,j} \text{ for } \underline{a}_{i,j} \in \mathbb{Z}_p. \tag{28}$$

Since $a_{i,j}$ is chosen uniformly at random, we get any plausible triple with probability $1/p$. This implies that the following event

$$\{(B_{i,j}, A_{i,j}, z_{i,j})\}_{i\in\mathcal{H}\cap\mathsf{SS}_j, j\in[q_s]} = \{(\underline{B}_{i,j}, \underline{A}_{i,j}, \underline{z}_{i,j})\}_{i\in\mathcal{H}\cap\mathsf{SS}_j, j\in[q_s]} \tag{29}$$

happens with probability $\leq 1/p^{\mathsf{NumRhos}}$, where $\mathsf{NumRhos} := \sum_{j\in[q_s], i\in\mathsf{SS}_j} \mathsf{SS}_j \cap \mathcal{H}$ as defined before.

Combining all of the above, for all transcripts $\tau \in W_0$ with $\mathsf{p}_0(\tau) > 0$, we get the following interpolation probability:

$$\mathsf{p}_0(\tau) = \frac{1}{p^2(p-1)^2} \cdot \frac{1}{p^{3t}} \cdot \frac{1}{p^{2q_s}} \cdot \frac{1}{p^{\mathsf{NumRhos}}} \cdot \frac{1}{p^{\mathsf{NumMsg}_1}} \cdot \frac{1}{p^{\mathsf{NumEquiv}}} \cdot \frac{1}{p^{\mathsf{NumRhos}}}.$$

Analysis of $\mathsf{p}_1(\tau)$. We now turn to the analysis of $\mathsf{p}_1(\tau)$. First, by a similar argument as for the calculation of $\mathsf{p}_0(\tau)$, we need the identities.

$$\alpha_h = \underline{\alpha}_h, \quad \alpha_v = \underline{\alpha}_v, \quad u = \underline{u}, \quad s_{12}(0) = \underline{s},$$

which again are true with probability

$$\frac{1}{p-1} \cdot \frac{1}{p-1} \cdot \frac{1}{p} \cdot \frac{1}{p} = \frac{1}{p^2(p-1)^2}. \tag{30}$$

Again, for the required identity $\{(s_{14}(i), r_{14}(i), u_{14}(i))\}_{i\in\mathcal{C}} = \{(\underline{s}_i, \underline{r}_i, \underline{u}_i)\}_{i\in\mathcal{C}}$, conditioned on $(\alpha_h, \alpha_v, s_{12}(0), u) = (\underline{\alpha}_h, \underline{\alpha}_v, \underline{s}, \underline{u})$, there is a unique set of three degree-t polynomials with constant terms $(\underline{s}, 1, \underline{u})$ for $\underline{\alpha} := \underline{\alpha}_h + \underline{u} \cdot \underline{\alpha}_v$ such that this identity holds. Since in game $\mathbf{G}_{14}$, we sample the t additional coefficients of each of these polynomials uniformly at random, the equality holds with probability $1/p^{3t}$.

Furthermore, using an analysis similar to $\mathsf{p}_0(\tau)$, we get that even in $\mathsf{p}_1(\tau)$, we have that:

$$\{\rho_{i,j}, \beta_{i,j}, c_{i,j}\}_{j\in[q_s], i\in\mathsf{SS}_j\cap\mathcal{H}} = \{\underline{\rho}_{i,j}, \underline{\beta}_{i,j}, \underline{c}_{i,j}\}_{j\in[q_s], i\in\mathsf{SS}_j\cap\mathcal{H}} \tag{31}$$

happens with probability:

$$\frac{1}{p^{\mathsf{NumRhos}}} \cdot \frac{1}{p^{\mathsf{NumMsg}_1}} \cdot \frac{1}{p^{\mathsf{NumEquiv}}}. \tag{32}$$

Lastly, consider the tuples $\{(B_{i,j}, A_{i,j}, z_{i,j})\}_{i \in \mathcal{H} \cap \mathsf{SS}_j, j \in [q_s]}$. For each (i,j), for uniformly random $a_{i,j}$, we have $z_{i,j} = a_{i,j} + c_{i,j} \cdot s_{14}(i) = a_{i,j} + c_j \cdot s_{12}(i)$. Next, using the facts that

$$\mathsf{F}_0(\rho_{i,j}) := g^{-u \cdot y_{i,j} - c_{i,j} \cdot \alpha}, \qquad \mathsf{F}_1(\rho_{i,j}) := g^{y_{i,j}}, \tag{33}$$

we get the following:

$$\begin{aligned}
B_{i,j} &= g^{a_{i,j}} \cdot \mathsf{F}_0(\rho_{i,j})^{r_{14}(i)} \cdot \mathsf{F}_1(\rho_{i,j})^{u_{14}(i)} \\
&= g^{a_{i,j}} \cdot \mathsf{F}_0(\rho_{i,j})^{1+r_{12}(i)} \cdot \mathsf{F}_1(\rho_{i,j})^{u+u_{12}(i)} \\
&= g^{a_{i,j}} g^{-u \cdot y_{i,j} - c_{i,j}\alpha} g^{u \cdot y_{i,j}} \cdot \mathsf{F}_0(\rho_{i,j})^{r_{12}(i)} \cdot \mathsf{F}_1(\rho_{i,j})^{u_{12}(i)} \\
&= g^{a_{i,j} - c_{i,j} \cdot \alpha} \cdot \mathsf{F}_0(\rho_{i,j})^{r_{12}(i)} \cdot \mathsf{F}_1(\rho_{i,j})^{u_{12}(i)},
\end{aligned}$$

where $a_{i,j} \leftarrow\!\!\$\ \mathbb{Z}_p$ and $\rho_{i,j}$ are the randomness of honest signer i in the j-th signing session (with signer set SS_j), and $c_{i,j} := \mathsf{GetChallenge}(m, \mathsf{SS}, \vec{\boldsymbol{\mu}}_{i,j})$ where $\vec{\boldsymbol{\mu}}_{i,j}$ is the first round message signer i receives during the j-th signing session.

Using a similar calculation, we also get:

$$\begin{aligned}
A_{i,j} &= g^{a_{i,j}} \cdot \mathsf{G}_0(\vec{\boldsymbol{\mu}}_{i,j})^{r_{14}(i)} \cdot \mathsf{G}_1(\vec{\boldsymbol{\mu}}_j)^{u_{14}(i)} \\
&= g^{a_{i,j} - c_{i,j} \cdot \alpha} \cdot \mathsf{G}_0(\vec{\boldsymbol{\mu}}_{i,j})^{r_{12}(i)} \cdot \mathsf{G}_1(\vec{\boldsymbol{\mu}}_{i,j})^{u_{12}(i)}.
\end{aligned}$$

Let $\tilde{a}_{i,j} := a_{i,j} - c_{i,j} \cdot \alpha$, then

$$B_{i,j} = g_{i,j}^{\tilde{a}} \cdot \mathsf{F}_0(\rho_j)^{r_{12}(i)} \cdot \mathsf{F}_1(\rho_j)^{u_{12}(i)}, \tag{34}$$

$$A_{i,j} = g_{i,j}^{\tilde{a}} \cdot \mathsf{G}_0(\vec{\boldsymbol{\mu}}_{i,j})^{r_{12}(i)} \cdot \mathsf{G}_1(\vec{\boldsymbol{\mu}}_{i,j})^{u_{12}(i)} \tag{35}$$

$$z_{i,j} = \tilde{a}_{i,j} + c_{i,j} \cdot (s_{12}(i) + \alpha). \tag{36}$$

Recall from game $\mathbf{G}_3$, an honest signer i responds with its signature share $z_{i,j}$ in the j-th session, only if all honest signers in SS_j received the same first round message. Therefore, if for any signing session j, there exists an $i \in \mathsf{SS}_j \cap \mathcal{H}$ such that $z_{i,j} \neq \perp$, then for all $(i, i') \in \mathsf{SS}_j \cap \mathcal{H}$, we have that $\vec{\boldsymbol{\mu}}_{i,j} = \vec{\boldsymbol{\mu}}_{i',j} = \vec{\boldsymbol{\mu}}_j$. In that case, we have that for all $i \in \mathsf{SS}_j \cap \mathcal{H}$, we have that $\tilde{a}_{i,j} = a_{i,j} - c_j \cdot \alpha$, where $c_j = \mathsf{GetChallenge}(m, \mathsf{SS}, \vec{\boldsymbol{\mu}}_j)$. As a result, the signature share signer i outputs is either $z_{i,j} = \tilde{a}_{i,j} + c_j(s_{12}(i) + \alpha)$ or $z_{i,j} = \perp$. Moreover, let $\hat{A}_j$ be the combined nonce. Then, from game $\mathbf{G}_7$, we have that $c_j = \mathsf{H}_{\mathsf{sig}}(\hat{A}_j, \mathsf{pk}, m) = \mathsf{GetChallenge}(m, \mathsf{SS}, \vec{\boldsymbol{\mu}}_j)$.

From all of the above, we get that, for all signing sessions, where $\mathcal{A}$ sends the same first round message $\vec{\boldsymbol{\mu}}_j$ to all, given that everything else is fixed, the triple $(B_{i,j}, A_{i,j}, z_{i,j})$ are a function of $\tilde{a}_{i,j} := a_{i,j} - c_j \cdot \underline{\alpha}$. Therefore, we get $(B_{i,j}, A_{i,j}, z_{i,j}) = (\underline{B}_{i,j}, \underline{A}_{i,j}, \underline{z}_{i,j})$ only if $a_{i,j} - \underline{c}_j \cdot \alpha = \underline{a}_{i,j}$ for the same $\underline{a}_{i,j} \in \mathbb{Z}_p$ we used in equation (28). Since $a_{i,j}$ is chosen uniformly at random, the probability of $a_{i,j} - \underline{c}_j \cdot \underline{\alpha} = \underline{a}_{i,j}$ is also $1/p$. As each signer i selects its $a_{i,j}$ independently, the probability of $\{(B_{i,j}, A_{i,j}, z_{i,j}) = (\underline{B}_{i,j}, \underline{A}_{i,j}, \underline{z}_{i,j})\}_{j \in [q_s], i \in \mathsf{SS}_j \cap \mathcal{H}}$ is $1/p^k$, where k is the total number of honest signers across all such signing sessions, where $\mathcal{A}$ sends the same first round message.

Now consider all other signing sessions where $\mathcal{A}$ did not send identical first-round messages. For each such session j, we have:

$$\begin{aligned}
B_{i,j} &= g^{a_{i,j} - c_{i,j} \cdot \alpha} \cdot \mathsf{F}_0(\rho_{i,j})^{r_{12}(i)} \cdot \mathsf{F}_1(\rho_{i,j})^{u_{12}(i)}, \\
A_{i,j} &= g^{a_{i,j} - c_{i,j} \cdot \alpha} \cdot \mathsf{G}_0(\vec{\boldsymbol{\mu}}_{i,j})^{r_{12}(i)} \cdot \mathsf{G}_1(\vec{\boldsymbol{\mu}}_{i,j})^{u_{12}(i)}.
\end{aligned}$$

Here, $a_{i,j} \leftarrow\!\!\$\ \mathbb{Z}_p$ and $\rho_{i,j}$ are the randomness of honest signer i in the j-th signing session (with signer set SS_j), and $c_{i,j} := \mathsf{GetChallenge}(m, \mathsf{SS}, \vec{\boldsymbol{\mu}}_{i,j})$ where $\vec{\boldsymbol{\mu}}_{i,j}$ is the first round message signer i receives during the j-th signing session.

From all of the above, we get that given that everything else is fixed, the tuple $(B_{i,j}, A_{i,j})$ is a function of $\tilde{a}_{i,j} := a_{i,j} - c_j \cdot \underline{\alpha}$. Therefore, we get $(B_{i,j}, A_{i,j}) = (\underline{B}_{i,j}, \underline{A}_{i,j})$ only if $a_{i,j} - \underline{c}_j \cdot \alpha = \underline{a}_{i,j}$ for the same $\underline{a}_{i,j} \in \mathbb{Z}_p$ we used in equation (28). Since $a_{i,j}$ is chosen uniformly at random, the probability of $a_{i,j} - \underline{c}_j \cdot \underline{\alpha} = \underline{a}_{i,j}$ is also $1/p$. As each signer i selects its $a_{i,j}$ independently, the probability of $\{(B_{i,j}, A_{i,j}) = (\underline{B}_{i,j}, \underline{A}_{i,j})\}_{j \in [q_s], i \in \mathsf{SS}_j \cap \mathcal{H}}$ is $1/p^{k'}$, where k' is the total number of honest signers across all such signing sessions, where $\mathcal{A}$ sends different first-round messages to different honest signers.

Next, note that $k + k' = \mathsf{NumRhos}$, therefore, the event

$$\{(B_{i,j}, A_{i,j}, z_{i,j})\}_{i \in \mathcal{H} \cap \mathsf{SS}_j, j \in [q_s]} = \{(\underline{B}_{i,j}, \underline{A}_{i,j}, \underline{z}_{i,j})\}_{i \in \mathcal{H} \cap \mathsf{SS}_j, j \in [q_s]} \tag{37}$$

happens in game $\mathbf{G}_{14}$ with probability $1/p^{\mathsf{NumRhos}}$.

Combining all of the above, for all transcripts $\tau \in W_1$ with $\mathsf{p}_1(\tau) > 0$, we get the following interpolation probability:

$$\mathsf{p}_0(\tau) = \frac{1}{p^2(p-1)^2} \cdot \frac{1}{p^{3t}} \cdot \frac{1}{p^{2q_s}} \cdot \frac{1}{p^{\mathsf{NumRhos}}} \cdot \frac{1}{p^{\mathsf{NumMsg}_1}} \cdot \frac{1}{p^{\mathsf{NumEquiv}}} \cdot \frac{1}{p^{\mathsf{NumRhos}}}.$$

Since the above holds for any (possible) transcripts, we get $\mathsf{p}_0(\tau) = \mathsf{p}_1(\tau)$ for all transcripts. This implies that $X_0 \equiv X_1$. Finally, by Lemma 2, we get that the view of $\mathcal{A}$ in the games $\mathbf{G}_{13}$ and $\mathbf{G}_{14}$ is identically distributed. □