Fully Adaptive FROST with Identifiable Aborts from AOMDL*

Ruben Baecker[1], Paul Gerhart[2], Davide Li Calsi[2], Luigi Russo[2], Dominique Schröder[2], and Arkady Yerukhimovich[3]

[1] *Friedrich-Alexander-Universität Erlangen–Nürnberg, Erlangen, Germany*
[2] *TU Wien, Vienna, Austria*
[3] *George Washington University, Washington, D.C., United States*

Abstract

We present FaFROST+, the first two-round Schnorr threshold signature scheme achieving full adaptive security under the Algebraic One-More Discrete Logarithm (AOMDL) assumption with identifiable aborts. Building on the FROST framework, FaFROST+ preserves its communication efficiency while dispensing with the LDVR assumption introduced at CRYPTO'25. Our result demonstrates that round-optimal adaptive security, practical efficiency, and identifiable aborts are simultaneously achievable under the well-studied AOMDL assumption.

*This is the full version of the paper appearing at Asiacrypt 2026. A previous version of this work was first published as a preprint under the title *Fully Adaptive FROST in the Algebraic Group Model From Falsifiable Assumptions* [BGC+25]. This full version is a major revision of that preprint and supersedes it.

1 Introduction

A *threshold signature scheme* allows any subset of t out of n parties to jointly compute a signature on a message m that verifies under a single verification key vk [Des88; DF90]. The resulting signature is indistinguishable from one produced by a single signer holding the full secret key, yet no individual party ever sees that key in the clear. This makes threshold signatures a powerful tool wherever a single point of trust is unacceptable, with applications ranging from the signing keys of *certificate authorities*, whose compromise would allow attackers to issue fraudulent certificates at internet scale [BSP+08], to *custody solutions* for digital assets, where no individual employee or device should authorize a signature alone [Kav; Lin], to *consensus protocols*, where committees of validators jointly attest to blocks or protocol decisions [Eth; Lea], among many others. In all these settings, threshold signatures provide distributed trust without requiring any change to how downstream systems verify signatures.

The cryptographic community has invested significant effort over the last years into building practical threshold signatures, and the field has matured to the point of standardization. NIST has issued a call for the *standardization of threshold signatures* [BD22; Nat26], and the leading candidate FROST [KG20] has been published as RFC 9591 by the IRTF [CKG+24]. FROST is a two-round Schnorr threshold signature scheme that is simple, efficient, and widely deployed in practice. Building on this progress, one important question remains open, namely whether full adaptive security can be achieved under well-established assumptions alone, without sacrificing other desirable properties of FROST. NIST frames this question explicitly [BD22]:

> *Given the possibility of adaptive corruptions in the real world, it is important to consider [. . .] whether the major safety properties of interest (such as unforgeability) are safeguarded against such an adversary.*

Adaptive security captures the realistic threat that an attacker decides whom to corrupt during protocol execution, based on what they observe, rather than fixing the corrupted set in advance. Recent works [CCK+25; CKK+25; CS25] show that full adaptive security for FROST cannot be established under standard assumptions alone, which has prompted several proposals.

Two approaches achieve full adaptive security for FROST-like schemes, each at a different cost. The first, due to Crites et al. [CKK+25], proves security under the AOMDL assumption but additionally requires the non-standard LDVR assumption, whose hardness must be conjectured for signer-set sizes above $n = 128$ with no known reduction to well-established problems. The second approach, introduced by Katsumata, Reichle, and Takemure [KRT24] and subsequently used in [Che25; NRT26], avoids LDVR entirely through a pairwise blinding technique, but pays a different price. It requires an additional round, sacrificing FROST's two-round structure, and prevents the detection of misbehaving parties.

In any practical deployment of a threshold signature, some signers will eventually fail to follow the protocol, whether due to bugs, network issues, or active malice. *Identifiable aborts* are the property that, whenever a signing attempt fails, the honest parties can identify at least one party responsible for the failure. That party can then be excluded from subsequent rounds, so that the system continues to make progress. Without identifiable aborts, a single malicious signer can cause every signing attempt to fail *indefinitely and undetectably*, while the honest parties have no way to tell whom to exclude. This makes identifiable aborts a deployment-critical property [PKN+25], and one that NIST has explicitly considered in its consultation process on threshold schemes [Nat26]. The concern is even more acute at the signer-set sizes NIST envisions: with up to 1024 signers, the

probability that at least one party misbehaves in any given round becomes non-negligible, and a protocol that cannot identify the culprit cannot scale.

These two approaches towards proving FROST fully adaptively secure thus present a seemingly inherent trade-off between relying on AOMDL alone and preserving both round-optimality and identifiable aborts. We therefore ask:

> *Can one construct a round-optimal Schnorr threshold signature scheme that achieves full adaptive security under well-established assumptions, for arbitrarily large signing groups, with identifiable aborts?*

Our contributions. We answer this question affirmatively.

- We introduce as an intermediate tool FaFROST, a two-round threshold Schnorr signature scheme achieving full adaptive security in the AGM under the AOMDL assumption alone, without relying on the LDVR assumption. Our scheme preserves the round structure and communication efficiency of FROST2 [CKM21], and our security guarantees impose no restriction on the signing set.
- We refine FaFROST by endowing it with identifiable aborts, yielding our main scheme FaFROST+. This upgrade comes at *no cost to honest executions*: the identification phase is invoked only upon signing failure, leaving honest signing rounds unchanged. We formally prove its security under full adaptive corruption.
- We implement a full prototype of FaFROST+ in Rust [Ger26], covering the full protocol stack including two-round threshold signing and identifiable aborts. The computational overhead compared to FROST is negligible, as the blinding step accounts for less than 5% of total signing time across all evaluated configurations. The resulting signatures verify under deployed single-signer verifiers, namely EdDSA of FIPS 186-5 [Nat23] and BIP-340 [WNR20]. We additionally demonstrate Bitcoin compatibility with a confirmed transaction on the Bitcoin testnet [Faf].

2 Technical Overview

We begin by revisiting the FROST2 protocol. The signing protocol of FROST2 has two rounds and a local combination algorithm that converts partial signatures into a Schnorr signature with respect to a keypair (vk, sk). Each signer i holds a secret share sk_i of the group signing key sk. In the first signing round, each signer i samples random nonces $(d_i, e_i) \leftarrow\$ \mathbb{Z}_p^2$ and broadcasts the corresponding commitments $(D_i, E_i) = (g^{d_i}, g^{e_i})$. These values are independent of the signing key, the message, and the set of signers. In the second round, executed amongst a set of signers S, each signer aggregates all commitments via $D = \prod_{j \in S} D_j, E = \prod_{j \in S} E_j$, computes a hash challenge $b = \mathsf{H}_{\text{non}}(vk, S, m, \{D_j, E_j\}_{j \in S})$, and defines the combined nonce as $R = D \cdot E^b$. Let $c = \mathsf{H}_{\text{sig}}(vk, R, m)$ be the Schnorr challenge associated with the commitment R. Each signer then computes its partial signature $s_i = c \cdot \Lambda_i sk_i + d_i + b \cdot e_i$, where Λ_i is the Lagrange coefficient for index i. Let $s = \sum_i s_i$ be the sum of partial signatures. The final signature is (R, s), which is valid as it satisfies the standard Schnorr verification equation $g^s = R \cdot vk^c$. While the signature scheme is conceptually simple, proving its security in the adaptive setting turns out to be challenging. To motivate our approach, we first revisit the techniques used in prior work and explain why they are not sufficient to achieve a round-optimal solution from established assumptions.

Static security for FROST. Typically, FROST[1] is proven secure under the *Algebraic One-More Discrete Logarithm (AOMDL)* assumption [CGR+23]. At a high level, the AOMDL assumption asserts that no efficient adversary can compute the discrete logarithms of $q+1$ challenge group elements while making at most q queries to a discrete logarithm oracle (each accompanied by a valid *algebraic representation* of the queried group element).

Under the AOMDL assumption, the reduction proceeds as follows. The adversary first commits to the set of parties it intends to corrupt in the security experiment. The reduction then embeds the DL challenge into the verification key vk and generates $t-1$ random signing keys, which are provided to the adversary as the secret keys of the corrupted parties. For the remaining honest parties, the reduction computes consistent partial verification keys using polynomial interpolation *in the exponent*. During the signing protocol, the reduction embeds DL challenges into the elements D_i and E_i of the honest signing parties. It then computes the corresponding partial signatures by querying the DL oracle on $vk_i^{c\Lambda_i} \cdot D_i \cdot E_i^b$, where b and c are derived as in the standard signing algorithm.

The adversary eventually outputs a forgery (R^*, s^*). Using either the algebraic group model (AGM) or rewinding-based extraction, the reduction can extract the signing key sk from this forgery. With sk, the reduction can solve all embedded DL challenges. Specifically, if the adversary produces a signature forgery after q signing queries, the reduction solves $q+1$ DL instances using only q oracle queries, which breaks the AOMDL assumption. While this proof strategy establishes the security of FROST against *static* adversaries, it does not extend to the *adaptive* setting.

Fully adaptive security from LDVR. Recall that in the static case, the reduction embeds the DL challenge into the global verification key and derives the honest parties' partial verification keys by interpolation in the exponent; this is sound because the reduction never needs to reveal the honest parties' secret keys. In the fully adaptive setting, however, the adversary may corrupt parties at any time, and the reduction must hand out their secret keys on demand. This is precisely where the static embedding fails: once the DL challenge is tied to the derived partial verification keys, producing a corrupted party's secret key would require computing the discrete logarithm of that party's individual verification key. The reduction can generate at most $t-1$ verification keys with known secret keys, but since it does not know in advance which parties will be corrupted, it cannot ensure that these correspond to the corrupted ones. Querying the DL oracle to recover additional secret keys would also reveal (and thereby solve) the global challenge embedded in the verification key, leaving nothing to break once the adversary outputs a forgery.

To overcome this limitation, Crites et al. [CKK+25] propose a reduction based on the *LDVR* and *AOMDL* assumptions in the AGM. Their idea is to embed t challenge values into the n partial verification keys as *linear combinations* constructed so that every subset of t verification keys is linearly independent. This structure guarantees that, after the adversary corrupts up to $t-1$ parties, the reduction can compute the corresponding secret keys using $t-1$ queries to the DL oracle, while at least one independent challenge remains embedded in the verification keys. Eventually, when the adversary outputs a forgery (R^*, s^*) together with an algebraic representation of R^*, the reduction rewrites this representation so that it depends only on the partial verification keys. If this expression constitutes a new linearly independent combination of the embedded challenges, the reduction, together with the $t-1$ independent combinations revealed through corruptions, obtains a system of t linear equations, which it can solve to recover all challenge values and thereby break

[1] This holds for FROST, FROST2, and FROST3.

the *AOMDL* assumption.

A gap remains when R^* lies *within the span* of the partial verification keys of corrupted parties, meaning that the forgery does not yield a linearly independent combination. In this case, the reduction cannot extract the signing key sk, and the argument fails. To address this, Crites et al. [CKK+25] introduced the *LDVR assumption*, which, informally, states that producing such a linearly dependent combination is computationally hard. This completes the proof idea: either the forgery yields a linearly independent combination and breaks AOMDL, or it yields a linearly dependent one and breaks LDVR. The drawback of this approach is that LDVR is largely a proof artifact and remains a relatively new and not yet well-understood assumption. Since we aim to avoid relying on LDVR, we next explore an approach that achieves full adaptive security without it, but at the cost of an additional round.

Fully adaptive security from blinding. The main challenge in the fully adaptive setting is that the reduction must provide valid signing keys only after the adversary has already seen the corresponding verification keys, such that the verification keys already commit to the secret keys. To overcome this challenge, Katsumata, Reichle, and Takemure [KRT24], followed by subsequent work [Che25], introduced a *blinding* technique in the random oracle model. The high-level idea is to blind each protocol message using the random oracle evaluated on a pairwise key shared between each pair of signers and a unique session identifier. As long as at least one pairwise key remains unknown, the resulting blinded messages are distributed uniformly at random, appear independent of any individual verification key, and do not commit to any private key. When the messages of all signers are aggregated, the blindings cancel out, revealing the correct combined value.

In the reduction, this mechanism allows the simulator to send uniformly random values instead of honestly computed protocol messages, which would otherwise require knowledge of the honest parties' secret keys. To ensure that the aggregated messages remain consistent, the reduction employs the *last-consistent-signer* technique: it samples random messages for all but the last honest signer and defines the final honest message as the difference between the intended aggregate value and the sum of the others. Each individual message, therefore, appears random, while their aggregate is consistent with the protocol specification. This approach never requires the reduction to commit to any partial secret key beyond those already revealed through corruptions.

Upon corruption, the reduction samples a fresh signing key for the newly corrupted party. To preserve consistency with previously sent messages, it then programs the random oracle so that the existing blinded messages equal the honestly computed messages combined with the corresponding blinding values. As long as at least one pairwise key remains hidden and session identifiers are unique, such oracle programming can be performed without conflict.

This approach achieves full adaptive security, but it crucially relies on unique session identifiers. If a session identifier is ever reused, the random-oracle programming would become inconsistent. Ensuring uniqueness thus appears to require either maintaining a secure state across sessions or performing an additional round to agree on a fresh identifier, both of which we want to avoid.

Our idea. We propose a refined blinding strategy that achieves fully adaptive security for FROST in two rounds without relying on the *LDVR* assumption. In contrast to the approach of [KRT24], we apply blinding only to those protocol messages that depend on the partial signing keys. In FROST, this separation is natural: the first round messages, which consist of exchanging (D_i, E_i), are independent of any secret key material. These messages are also uniformly random values, which

we repurpose as a session identifier. We can, therefore, leave the first round unchanged.

The second round messages s_i are key-dependent, and we blind them in the same spirit as [KRT24], using the first round messages as a session identifier. As described above, the blinding ensures that these key-dependent messages individually appear uniformly random from the adversary's view. Using our refined blinding strategy, we require no additional rounds, since the session identifier is derived directly from the first-round messages. As we show next, this minor modification enables us to prove full adaptive security in only two rounds, without relying on the LDVR assumption, by employing a new proof strategy. We highlight in Figure 5 the modifications to the protocol compared to FROST (beyond key generation).

2.1 Proving Security

At a high level, our refined blinding strategy should suffice to prove our scheme fully adaptively secure. However, we cannot apply existing proof techniques directly. We now first explain why the last-consistent-signer technique fails in our setting and then introduce a new challenge-embedding strategy. Using this new strategy, we show how to reduce unforgeability against a fully adaptive adversary to the hardness of the AOMDL assumption.

Last-consistent-signer. In the last-consistent-signer technique, the reduction uses a single DL challenge element and a single oracle query to simulate an entire signing session, regardless of the number of honest signers in that session. As explained above, this suffices when all protocol messages are blinded and the random oracle is programmed carefully. In our setting, however, we must embed the DL challenge into both first-round messages (D_i, E_i), which commits us to a signer's randomness. If the signer chosen as the last-consistent one is later corrupted after the first round, the reduction must query the DL oracle twice to solve these challenges. More critically, since only two challenges were embedded, no unresolved challenge remains within that signing query, yet the reduction still requires an additional oracle call to produce partial signatures, as it does not know the corresponding signing keys. This results in the reduction making more DL oracle calls than challenges embedded, preventing it from breaking the AOMDL assumption upon forgery. We therefore need a different strategy for embedding DL challenges.

New challenge embedding strategy. To overcome the issue of too many DL oracle calls for insufficient challenges, we modify our embedding strategy by introducing two DL challenges per honest signer and signing session, rather than embedding challenges for only a single signer throughout the entire session. This ensures that when a signer is corrupted after the first round, we can still open the two corresponding DL challenges and maintain consistency with previously published messages. Crucially, even in this case, at least one honest signer per session remains, since the adversary can corrupt at most $t-1$ parties while each signing session requires t signers to produce a valid signature. Therefore, when executing the second round, at least two honest challenges remain embedded in a random commitment, allowing us to use one more DL oracle query to compute a single partial signature. As in the last-consistent-signer technique, we can use the blinding of the second protocol message to simulate uniformly random outputs that combine into the expected values for the adversary.

Implications of first-round commitments. Our new embedding approach resolves the challenge of simulating signing and handling corruptions that occur after the first round. What remains

is to address corruptions that happen after the second round. This setting differs from prior techniques, where all protocol messages could be made consistent in the event of corruption. In our case, each signer commits to its randomness in the first round. Once the adversary learns a signer's secret key and all corresponding blinding keys, the reduction must ensure that the second-round message remains consistent with the first-round commitments in a retroactive manner.

To achieve this, we again rely on the DL oracle. When a corruption of signer i occurs, the reduction samples a fresh random signing key $sk_i \leftarrow\!\!\$\ \mathbb{Z}_p$ and queries the DL oracle to obtain the logarithm of each D_k and each E_k, which were handed out on behalf of signer i in the k-th first-round oracle query. This yields the randomness for this session, which is $d_k + b \cdot e_k$. Following the same blinding-programming strategy as before, the reduction makes the already-sent protocol message consistent by solving for the blinding term $B^s_{i,k}$ in $s_i = sk_i \cdot \Lambda_i \cdot c + d_k + b \cdot e_k + B^s_{i,k}$ and programming the random oracle accordingly. This strategy is effective as long as at least two honest signers remain in the signing set before corruption occurs. If only one honest signer is left, no further corruption can occur, and two unsolved challenge elements remain. Consequently, in every signing session, at least two honest challenges are preserved, and each session requires exactly one DL oracle query to produce the partial signature. Thus, when corruptions occur after either round, the total number of oracle queries is less than or equal to the number of embedded challenges for each signing round.

Why a rewinding-based proof fails. We argue that a rewinding-based proof does not work for our challenge-embedding strategy. Depending on the adversary's strategy, our reduction must extract the signing key *sk without* querying the DL oracle in order to solve one more challenge than the number of oracle queries. Extracting the signing key is typically achieved through rewinding [CGR+23].

In a rewinding-based proof, the reduction runs the adversary twice using the same randomness. At a certain point, it alters the output of the random oracle in the second run, causing the rewound adversary to deviate from its original behavior. This technique, commonly known as *forking*, enables the reduction to leverage the *special soundness* property of the Schnorr protocol by extracting the signing key once the adversary produces two valid forgeries on the same randomness and message. Based on the forking lemma [BN06], this event occurs with sufficiently high probability, allowing the extractor to rewind the adversary and obtain two transcripts with distinct challenges (but for the same nonce R) necessary for key extraction.

Although the forking technique enables the extraction of the signing key thus solving the discrete logarithm of vk, it is incompatible with our challenge-embedding approach. The core issue is that forking-based proofs effectively allow the adversary to corrupt at least t signers across the two forks, even though each individual run involves corrupting at most $t-1$ parties. In contrast, our embedding strategy requires that no more than $t-1$ signers are corrupted *per signing session*.

To understand why our approach fails under forking, consider a session that exists before the fork where no signer is corrupted. After forking, the adversary corrupts one subset of signers in the first branch and the complementary subset in the second. Consequently, the reduction must resolve the challenges corresponding to each subset separately in the respective forks. Additionally, the reduction already relies on DL oracle calls to simulate signatures.

Since the adversary's corruptions are split across forks, the total number of corrupted signers across the two executions exceeds the per-session corruption bound. This means that after these corruptions, no embedded challenge remains unextracted for the session in question. As a result,

the reduction issues more oracle queries than the number of embedded challenges, preventing the AOMDL assumption from being violated upon forgery and causing the reduction to fail. In other words, while the corruption threshold $t-1$ acts as a *local bound* within each individual execution, the forking strategy increases the *global bound* on corrupted signers across forks beyond what the embedding can handle, effectively nullifying the reduction strategy. The concurrent work [BCL+25] formalizes our intuition with an impossibility result.

Extracting the signing key. We resolve this tension by working in the *Algebraic Group Model (AGM)*. In the AGM, the adversary is assumed to output group elements along with an explicit algebraic representation that describes how they were computed from previously obtained elements. For our security proof, this representation may involve the generator g, the verification key vk, and all previously output first-round messages D_k and E_k, and takes the form $R^* = g^{\alpha_0} \cdot vk^{\alpha_1} \cdot \prod_{1\le k\le q_s} D_k^{\beta_k} \cdot \prod_{1\le k\le q_s} E_k^{\gamma_k}$, for known coefficients provided by the adversary to the reduction. Our reduction will use this representation to extract the signing key. The open question is how to do this, given that the representation may include DL challenges that the reduction has not yet solved. In addition, the reduction has no remaining oracle queries at this stage, and we cannot directly follow prior extraction methods [CKK+25; CKM23], since our embedding structure differs from those. To successfully extract sk, we first observe that each term involving D_k and E_k can be rewritten using only the verification key vk, since the reduction knows (unblinded) partial signatures s_{HON}, which satisfy $g^{s_{\text{HON}}} = vk_{\text{HON}}^c \cdot D_k \cdot E_k^b$, for some vk_{HON} dependent on vk. We defer the technical details to Section 6, but this rewriting yields a simplified representation of the form $R^* = g^\alpha \cdot vk^\beta$, with α and β known to the reduction. Moreover, from the validity of the forgery (R^*, s^*), the reduction also knows that $R^* \cdot vk^{c^*} = g^{s^*}$. Combining both equations allows the reduction to solve for the secret key as $sk = (s^* - \alpha)/(\beta + c^*)$, which yields a solution if $\beta + c^*$ does not vanish. As we will demonstrate in our security analysis, this is the case with overwhelming probability.

Breaking the AOMDL assumption. Once we have extracted sk, it remains to show that our reduction breaks the AOMDL assumption. The first DL challenge corresponds to the combined verification key vk. When the adversary outputs a valid forgery, the reduction extracts the signing key sk, thereby solving one challenge. Once the signing key is known to the reduction, it can recover all other embedded DL challenges with the remaining DL oracle queries. Crucially, as discussed above, our reduction is constructed such that, for each signing session, the number of embedded challenges is at least as large as the number of DL oracle queries made throughout the security experiment. Consequently, the total number of DL oracle queries is strictly smaller than the number of solved challenges, and the reduction successfully breaks the AOMDL assumption whenever the adversary outputs a valid forgery.

2.2 Identifiable Aborts

We have seen that our modified blinding technique enables proving unforgeability against fully adaptive adversaries without introducing additional rounds. However, a drawback of the blinding techniques proposed by Chen [Che25] and Katsumata, Reichle, and Takemure [KRT24] is that they only allow one to detect whether *someone* misbehaved during the signing protocol, namely by checking whether the final signature verifies, but not to determine *who* misbehaved. Identifying the misbehaving party is highly relevant in practice [RRJ+22], since even a *single* malicious participant can continuously disrupt the protocol if misbehaving parties cannot be identified and removed. This

property is known in the literature as *identifiable aborts* [PKN+25] and is especially crucial for large signing sets, as without it any signing protocol is likely to fail. In the following, we show that our updated blinding technique not only achieves round optimality but also suffices to realize efficient identifiable aborts.

In FROST, misbehavior by an individual signer can be detected by verifying each partial signature (D_i, E_i, s_i) with the equation $g^{s_i} \stackrel{?}{=} D_i \cdot E_i^b \cdot vk_i^{c\Lambda_i}$, where b is the hash-output as above. We cannot follow this verification approach, since (i) our protocol cannot publish binding partial verification keys that commit to individual signing keys, and (ii) each s_i is blinded with the pairwise blinding values. Facing a similar challenge, Del Pino et al. [PKN+25] proposed an approach for a lattice-based threshold signature scheme that achieves identifiable aborts while relying on pairwise blinding values in a static security setting. Their technique, however, is not directly applicable to our setting, as it *does not support adaptive security*. This is the case since using their protocol, one cannot maintain consistency between past protocol messages and newly sampled secret keys after corruption. Yet, we show how to modify their core idea to obtain identifiable aborts for our blinded round-optimal signature scheme with fully adaptive security. Following prior work [CKK+25], we assume a public key infrastructure (PKI) and authenticated point-to-point channels between all participants.

High-level idea. To enable identifiable aborts, each signer must be able to verify that the others' outputs are well-formed. Since the first-round messages are unblinded, it suffices to prove the well-formedness of the second-round messages. In particular, each signer should demonstrate that its second-round value is consistent with the publicly known first-round messages. If this can be done efficiently, the protocol can pinpoint the misbehaving participant whenever the final signature fails to verify. Importantly, this mechanism does not need to be embedded directly into the signing protocol. Instead, the identifiable-abort phase can be invoked only *after* a verification failure. This way, honest executions incur no extra cost, while dishonest behavior can be detected and attributed, disincentivizing misbehavior.

As a first step towards identifying aborts, we consider the unblinded values s_i, whose well-formedness we aim to demonstrate. To prove their well-formedness, we enhance our protocol to include Pedersen commitments that commit to the individual secret keys during setup. This modification does not interfere with our proving strategy: Pedersen commitments are perfectly hiding, and since the reduction knows a trapdoor, it can continue to sample signing keys retroactively. However, having the Pedersen-committed partial signing keys and the unblinded first-round messages, the signers can use Chaum–Pedersen proofs [CP93] to prove the well-formedness of s_i. Such proofs also do not interfere with our fully adaptive proving strategy, since Chaum–Pedersen proofs are simulation extractable [DDO+01], and thus the reduction can simulate these proofs even when a partial signing key is not fixed. It remains to incorporate the blinding values in order to prove well-formedness for the blinded shares s'_i. Recall that the shares s'_i have the form $s'_i = s_i + \sum_{j \in S \setminus \{i\}} B^s_{i,j} \Delta_{i,j}$, where $B^s_{i,j} \in \mathbb{Z}_p$ is the pairwise identical blinding for the signers i and j. To incorporate the blinding values into our proof of well-formedness, we let each signer commit to each blinding value $B^s_{i,j}$, again using Pedersen commitments. Then, we can enhance the Chaum–Pedersen proofs for proving just the correctness of s_i to now prove the correctness of s'_i w.r.t. these commitments. This proof and these commitments guarantee the correct behavior of signer i, if the commitments to the blinding values commit to the correct values. This does not hold in general, since an adversary could just use a malformed $B^s_{i,j}$ during signing, faithfully commit to it, and prove

that its s_i contains $B^s_{i,j}$. This way, the proof would be valid, but the partial blindings would not cancel out during signature aggregation. To avoid this, we observe the following about the blinding commitments, which will allow us to argue for their correctness.

- The correctness of s'_i is preserved as long as the pairwise blinding values of two parties cancel out, even if the individual values were computed incorrectly.
- To commit to the blinding values, we can use unique commitments, which ensure that, based on the same pre-shared key, a given blinding value will result in a unique commitment.
- Using these commitments, if the commitments to the blinding between the parties (i, j) match each other, the corresponding blindings will cancel out.
- If two commitments for the same blinding do not match, at least one of the two parties must be malicious. In that case, we can request that the two involved parties reveal their shared pairwise key to identify which participant used an incorrect blinding.

Requesting that the parties reveal their pairwise keys is acceptable, as if such a mismatch occurs, at least one of the two parties is already malicious, and therefore the shared key is already known to the adversary. Moreover, the approach of deterministically committing to the pairwise blindings is consistent with our equivocal approach, as we can use random oracle programming for the deterministically derived commitment randomness, as explained next.

Our identification protocol. We now provide a more detailed explanation of our protocol. During setup, each signer receives a Pedersen-committed partial verification key $vk_i = \mathsf{C}(sk_i, \omega_i)$, and for each pairwise shared key $k_{i,j}$ a commitment $K_{i,j} = \mathsf{H}_{vk}(k_{i,j})$. As explained above, this does not interfere with our fully adaptive strategy since our reduction will know the trapdoor for the Pedersen commitment, and then the public commitments can be equivocated. To initiate the identification protocol, each signer broadcasts its local transcript trans_i, which contains all first-round and second-round messages it has sent or received during the protocol session in question. Each signer i then creates deterministic commitments to its pairwise blinding values $B^s_{i,j}$ as $\mathsf{C}_{i,j} = \mathsf{C}(B^s_{i,j}, \mathsf{H}_{\mathsf{IA}}(k_{i,j}, \{(\mathsf{pm}_j, s_j)\}_{j\in S}))$, where the randomizer is derived from the pairwise key $k_{i,j}$ and the broadcasted view via the random oracle H_{IA}. This construction guarantees that honest parties produce matching commitments for the same blinding value. Once these commitments are computed, each signer proves in zero knowledge that its partial signature $s_i = sk_i \cdot \Lambda_i \cdot c + d_i + b \cdot e_i + B^s_i$ is well formed with respect to the partial verification key commitment vk_i and first-round values D_i and E_i. The relation proven is:

$$g^{s_i} D_i^{-1} E_i^{-b} = g^{sk_i \Lambda_i c} \prod_{j \neq i} g^{B^s_{i,j}} \ \wedge\ vk_i = \mathsf{C}(sk_i, \omega_i) \ \wedge\ \forall j \neq i : \mathsf{C}_{i,j} = \mathsf{C}(B^s_{i,j}, \omega_{i,j}).$$

This zero-knowledge proof integrates seamlessly with our overall proof strategy, as Chaum–Pedersen proofs are simulation extractable. Therefore, honest users can simulate this proof efficiently *without* needing to determine the blinding values at the time of proving. Moreover, these proofs do not prove over correct random oracle evaluations, since only the outputs of the random oracle are relevant, but not that these outputs are faithfully derived. Verification proceeds as follows:

1. If two honest signers received different first-round messages from the same party, that party is immediately identified as malicious. Since communication takes place over authenticated channels, this check can be performed reliably.
2. If a proof π_i does not verify, party i is identified as malicious.
3. If no misbehavior is detected in the steps above, there must exist at least one pair of signers (i, j) whose commitments $\mathsf{C}_{i,j}$ and $\mathsf{C}_{j,i}$ do not match, as otherwise all blindings would cancel and the

signature would verify. In this case, each pair (i, j) with mismatching commitments reveals their pairwise key $k_{i,j}$, whose consistency can be checked with the public commitment $K_{i,j}$ created during setup, and their commitments $\mathsf{C}_{i,j}$ and $\mathsf{C}_{j,i}$ can be publicly verified. Revealing this key is harmless, since at least one party in each pair must be malicious anyway.

3 Related Work

Threshold signature schemes have been extensively studied across a wide range of settings, beginning with foundational works [DDF+94; GRJ+00; Sho00]. Subsequent research has explored constructions for different signature families, including BLS-based schemes [Bol03], (EC)DSA-based protocols [ANO+22; CCL+20; CGG+20; DJN+20; DKL+19; DOK+20; GG18; GGN16; GKS+20; LN18; YCX21], and Schnorr-based variants [GJK+03; GJK+07; GJK+96; KG20]. Among the latter, statically secure designs include FROST [BCK+22; CGR+23; KG20] and Sparkle [CKM23], while recent works have also established strong impossibility results for achieving full adaptive security [CCK+25; CKK+25; CS25]. As already mentioned in the introduction, several fully adaptive schemes have been proposed recently. For completeness, we briefly cite them here as well. Crites et al. [CKK+25] propose an LDVR-based construction, Chen [Che25] achieves full adaptive security in three rounds from AOMDL, and Glacius [BDL+25b] and Gargos [BDL+25a] by Bacho et al. achieve it in five and three rounds, respectively, from DDH. Additionally, Gerhart et al. [GCR+25b] present a two-round deterministic scheme, also based on DDH, which has been further generalized in [GCR+25a]. A brief comparison of these works is provided in Table 1.

Concurrent work. Bacho et al. [BCL+26] independently and concurrently propose the same two-round threshold Schnorr signature scheme, which we therefore consider a shared contribution of both works. They prove security under the AOMDL and the weighted threshold ROS (wtROS) assumptions, and additionally show that wtROS is unconditionally hard, making the scheme effectively secure under AOMDL alone in the AGM+ROM. Our work differs in two respects. Most significantly, we provide a protocol for identifiable aborts, which Bacho et al. [BCL+26] explicitly leave as an open problem. As mentioned in the introduction, identifiable aborts are essential for practical deployment: without them, a single malicious participant can indefinitely stall the protocol undetected. Therefore, the lack of identifiable aborts is widely considered a hard barrier for real-world adoption. In addition, our proof is more direct and easier to follow. Instead of first reducing to an idealized forgery game, then to the wtROS assumption, and finally showing the unconditional hardness of wtROS, we keep a single security proof and argue the adversary's capabilities mainly using linear algebra. We thank the authors of [BCL+26] for pointing out an error in an earlier version of our proof, in which we did not account for the additional adversarial winning probability of $\frac{q_{\text{sig}} \cdot t}{p(t-t_c)}$, which boils down to the wtROS problem in their proof. Our current proof corrects this using a combinatorial argument inspired by wtROS, which we integrate into our direct proof without introducing wtROS as a separate assumption.

4 Preliminaries

Notation. We denote by $x \leftarrow y$ the assignment of value y to variable x, and we denote by $x \leftarrow\!\!\$\ X$ the uniform sampling of x from the set X. We utilize the symbol $\mathbb{G} := (\mathbb{G}, p, g)$ to denote a group description, where $\mathbb{G}$ is a cyclic group of order p, where p is a λ-bit prime, and g is a generator

	Rounds	Model	Assumption	IA
Bacho et al. [BDL+25b]	5	ROM	DDH	✓
Katsumata et al. [KRT24]	5	ROM	DL	✗
Bacho et al. [BDL+25a]	3	ROM	DDH	✓
Chen [Che25]	3	ROM	AOMDL	✗
Niot, Reichle, and Takemure [NRT26]	3	ROM	DL	✗
Gerhart et al. [GCR+25b]	2*	ROM	DDH	✓
Crites et al. [CKK+25]	2	AGM+ROM	AOMDL+LDVR	✓
Bacho et al. [BCL+26]†	2	AGM+ROM	AOMDL	✗
Our work	**2**	**AGM+ROM**	**AOMDL**	✓

Table 1: Comparison of threshold Schnorr signature schemes that achieve full adaptive security. * The work of Gerhart et al. [GCR+25b] needs a periodic update mechanism to handle adversaries making polynomially many signing queries. † Bacho et al. [BCL+26] proposed the same signature scheme concurrently with our work, but without identifiable aborts (IA).

of $\mathbb{G}$. Given an element $X \in \mathbb{G}$, we let $\log_g(X)$ denote the discrete logarithm (DL) of X with base g, i.e., $\log_g(X)$ is the value $x \in \mathbb{Z}_p$, such that $X = g^x$. We introduce the function $\mathsf{CheckView}$ that, on input so far exchanged protocol messages by honest parties verifies, that all parties have a consistent view for a given session. This is relevant to security experiments that consider identifiable aborts, in which we prohibit an adversary from winning by exploiting inconsistent views amongst parties. Introducing $\mathsf{CheckView}$ is reasonable, since in the setting of identifiable aborts, we rely on authenticated channels, so providing inconsistent views is infeasible for any efficient adversary.

Algebraic one-more discrete logarithm. Let $\mathbb{G}$ be a cyclic group of prime order p with generator g. The Algebraic One-More Discrete Logarithm (AOMDL) assumption [NRS21] is the falsifiable version of the One-More Discrete Logarithm assumption (OMDL). In the OMDL game, an adversary $\mathcal{A}$ interacts with two oracles OCH and OdLog. OCH is challenge oracle, and issues a DL challenge $Y \in \mathbb{G}$ upon receiving a request from $\mathcal{A}$. The OdLog oracle reads an element $Y \in \mathbb{G}$ and returns its discrete logarithm with respect to a generator g. To win the OMDL game, $\mathcal{A}$ must receive C challenges from OCH and compute all of their discrete logarithms in base g by using at most $C-1$ calls to OdLog. The associated assumption states that no PPT $\mathcal{A}$ wins the aforementioned game with non-negligible advantage. To make it falsifiable (in the sense of [GW11]), we can slightly modify OdLog by forcing $\mathcal{A}$ to provide an algebraic representation of the input Y. That is, if $Y_1, \ldots, Y_d$ denote the challenges issued by OCH so far, $\mathcal{A}$ must provide $\alpha, \beta_1, \ldots, \beta_d \in \mathbb{Z}_p$ such that:

$$Y = g^{\alpha} \cdot \prod_{i \in [d]} Y_i^{\beta_i}.$$

Figure 1 provides a detailed description of the associated security games. Similar to OMDL, AOMDL assumption states all PPT $\mathcal{A}$ win the security game with at best $negl(\lambda)$ advantage, which we define as

$$\mathsf{Adv}^{\mathsf{AOMDL}}_{\mathcal{A},(\mathbb{G},g,p)}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{AOMDL}}_{\mathcal{A},(\mathbb{G},g,p)} \to 1].$$

Both assumptions imply the hardness of dLog, and AOMDL implies the hardness of OMDL too.

The algebraic group model. The Algebraic Group Model (AGM) [FKL18] is an extension of the well-established Generic Group Model (GGM) [Mau05; Sho97]. Both are idealized computational

Game $\mathsf{AOMDL}^{\mathcal{A}}_{(\mathbb{G},p,g)}$	Oracle $\mathsf{OCH}()$	Oracle $\mathsf{OdLog}(X, \alpha, \{\beta_i\}_{i\in[C]})$
1 : $(x_1, \ldots, x_C) \leftarrow \mathcal{A}^{\mathsf{OCH},\mathsf{OdLog}}()$	1 : $X_C \leftarrow\!\!\$\ \mathbb{Z}_p$	1 : $Q \leftarrow Q + 1$
2 : **return** $Q < C \wedge \bigwedge_{i\in[C]} x_i == X_i$	2 : $C \leftarrow C + 1$	2 : **return** $\alpha + \sum_{i\in C} x_i \cdot \beta_i$
	3 : **return** g^{X_C}	

Figure 1: Security experiment for AOMDL.

models where efficient adversaries performing computations over a group $(\mathbb{G}, \cdot)$ are able to only perform a subset of all the possible computations. In the GGM, the adversary is given abstract handles to group elements and is oblivious to their representation. Said handles can be used to combine elements through $\cdot$, and perform equality checks. This limited set of operations enables information-theoretic bounds on the success probability of a target computation. The AGM deviates from the GGM by forcing the adversary to output an algebraic representation of every output of theirs. Similar to AOMDL, the algebraic representation of output Z given previously received group elements $g, Y_1, \ldots, Y_d \in \mathbb{G}$ is a list of scalars $\alpha, \beta_1, \ldots, \beta_d \in \mathbb{Z}_p$ such that:

$$Z = g^{\alpha} \cdot \prod_{i\in[d]} Y_i^{\beta_i}.$$

An adversary of this type is often called *algebraic*. Unlike the GGM, the AGM does not allow for unconditional lower bounds, but it instead enables reductions between assumptions.

Threshold signatures. A threshold signature scheme $\mathsf{TS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}_1, \mathsf{Sign}_2, \mathsf{Combine}, \mathsf{Verify})$ allows any subset of t out of n parties to jointly produce a signature verifiable under a single key vk. We follow the two-round formalization of [CKK+25] and define the interfaces for two-round threshold signatures via:

- $\mathsf{Setup}(\lambda, t, n) \to par$: The setup algorithm reads the security parameter λ, the group size n, and the threshold $t \leq n$. It outputs public parameters par.
- $\mathsf{KeyGen}(par) \to \big(vk, \{(sk_i, vk_i)\}_{i\in[n]}\big)$: The key generation algorithm reads public parameters. It outputs a verification key vk, and for each signer $i \in [n]$ a partial signing key sk_i and a partial verification key vk_i. If a protocol does not output partial verification keys, we set $vk_i = \perp$.
- $\mathsf{Sign}_1(i, sk_i, vk, \{vk_i\}_{i\in[n]}) \to (st_i, \mathsf{pm}_i)$: reads a signer index i, a secret key sk_i, and all the public keys. It outputs a state st_i and first-round message pm_i for signer i.
- $\mathsf{Sign}_2(i, st_i, sk_i, vk, m, S, \{(vk_i, \mathsf{pm}_i)\}_{i\in S}) \to s_i$: reads all the inputs that are supplied to Sign_1, plus a message to be signed, a signer set, the first-round messages, and a state st_i. It outputs a partial signature s_i.
- $\mathsf{Combine}(m, S, \{s_i\}_{i\in S}) \to \sigma$: it reads a signer set, a message, and a set of partial signatures. It outputs a combined full signature σ.
- $\mathsf{Verify}(vk, m, \sigma) \to b$: it reads a message m, a full signature σ, and a public key vk. It returns a bit $b \in \{0, 1\}$.

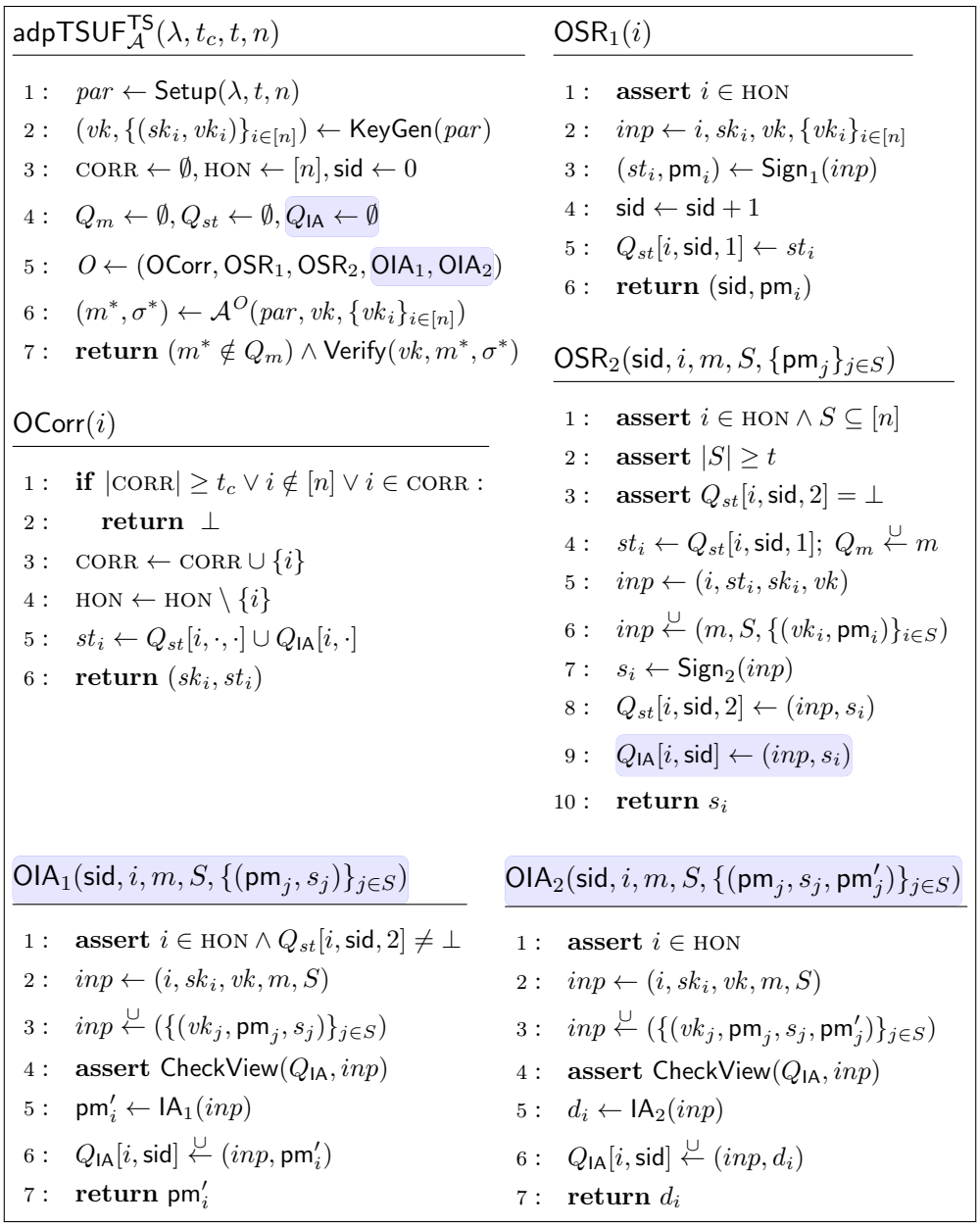

Figure 2: The adaptive unforgeability experiment with identifiable aborts. We mark the changes for adaptive unforgeability with identifiable aborts. Without the highlighted parts, we obtain the adaptive unforgeability experiment for ordinary threshold signature schemes.

Following [PKN+25], we say a threshold signature scheme has identifiable aborts if it additionally provides the algorithms[2] $(\mathsf{IA}_1, \mathsf{IA}_2, \mathsf{Decide})$ with the interfaces:

- $\mathsf{IA}_1(i, sk_i, vk, m, S, \{(vk_j, \mathsf{pm}_j, s_j)\}_{j\in S}) \to \mathsf{pm}'_i$: reads a signer index i, a secret key sk_i, the ver-

[2]We focus on the two-round case, but this generalizes to multiple rounds canonically.

ification key vk, a message m, a set of signers S, and the partial verification key, first-round signing message, and partial signature of each signer in S. It outputs a first-round IA message pm'_i.

- $\underline{\mathsf{IA}_2(i, sk_i, vk, m, S, \{(vk_j, \mathsf{pm}_j, s_j, \mathsf{pm}'_j)\}_{j\in S}) \to d_i}$: reads the same inputs as the first IA round, and additionally the first-round IA protocol messages of all signers in S. It outputs a partial IA decision d_i.
- $\underline{\mathsf{Decide}(vk, m, S, \{(vk_j, \mathsf{pm}_j, s_j, \mathsf{pm}'_j, d_j)\}_{j\in S}) \to \mathcal{C}}$: reads as inputs the verification key, the message, the signer set, and for each signer in the signer set the partial verification key, and the signing and IA protocol message sent by this signer. It outputs a set of identified misbehaving signers $\mathcal{C}$.

Adaptive unforgeability. In our experiment, we consider a fully adaptive adversary $\mathcal{A}$ with the ability to trigger signing sessions and corrupt users. We enable this adversarial power by adding corruption and signing oracles. To further support identifiable abort, we also provide oracles to IA. The signing oracles allow the adversary to receive protocol messages from honest signers on adversarially chosen messages and signing sets. The corruption oracle OCorr allows $\mathcal{A}$ to corrupt a user of its choice. Upon corruption of user i, $\mathcal{A}$ learns its partial secret key sk_i as well as its current state st_i. Furthermore, from that point on i is under $\mathcal{A}$'s full control, meaning that $\mathcal{A}$ dictates the messages issued by i. To avoid trivial wins, OCorr satisfies up to t_c many requests. The identifiable abort oracles are similar to the signing oracles and can be queried to receive protocol messages for the identifiable abort protocol from honest parties. To win the unforgeability experiment, $\mathcal{A}$ must output a non-trivial forgery (m^*, σ^*) such that $\mathsf{Verify}(vk, m^*, \sigma^*) = 1$. A forgery is considered trivial if m^* is a message for which $\mathcal{A}$ already queried the signing oracle. We formalize trivial messages in our experiment by storing all queried messages of the last signing round ($r = 2$) in a list Q_m.

This criterion corresponds to TS-UF-1 in the hierarchy of unforgeability notions for threshold signatures provided in [BCK+22]. We restrict our attention to TS-UF-1 security, which is justified by an argument of Katsumata, Reichle, and Takemure [KRT24]. As observed in the full version of their work (page 15), threshold signature schemes in which all message-dependent rounds are blinded with their techniques also achieve TS-UF-4 security whenever they achieve TS-UF-1 security. Both their scheme and FaFROST fall into this category. The reason is that their blinding technique allows message-dependent protocol messages to be sampled lazily until the last honest participant completes the signing protocol. Hence, prior to the contribution of the last honest participant, which is precisely the setting of TS-UF-4, no information is leaked to the adversary $\mathcal{A}$, and the reduction is in the same setting as for TS-UF-1.

We define fully-adaptive unforgeability for threshold signatures with identifiable abort as follows. Again, this definition subsumes full adaptive security for threshold signatures without identifiable abort, by omitting the respective oracles.

Definition 1 (Full Adaptive Security) For a threshold signature scheme with identifiable abort TS and an adversary $\mathcal{A}$ define

$$\mathsf{Adv}^{\mathsf{adpTSUF}}_{\mathcal{A},\mathsf{TS}}(t_c, t, n) = \Pr[\mathsf{Exp}^{\mathsf{adpTSUF}}_{\mathcal{A},\mathsf{TS}}(\lambda, t_c, t, n) \to 1],$$

where the probability is taken over the random choice of all randomized algorithms. TS is (t_c, t, n)-adaptively unforgeable if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $negl(\lambda)$, such

that $\mathsf{Adv}^{\mathsf{adpTSUF}}_{\mathcal{A},\mathsf{TS}}(\lambda, t_c, t, n) \leq negl(\lambda)$. TS is fully-adaptively secure if it is $(t, t-1, n)$-adaptively secure for all $t \leq n \in \mathbb{N}$. We present the security experiment in Figure 2.

Identifiable Abort Our notion of secure identifiable aborts follows [PKN+25], but our definition differs in one aspect. Honest signers are not required to explicitly agree on the set of misbehaving parties. This follows from the fact that our formalization includes a dedicated algorithm Decide that takes all messages from the signing and IA protocols from one session as input and determines which signers misbehaved. Therefore, all signers with the same protocol messages will decide identically. This simplification is meaningful, since we assume pairwise authenticated channels for the IA mechanism. Under this assumption, trivial forms of misbehavior, for example, refusing to send messages or sending inconsistent messages, can be attributed directly to the responsible signer. Our definition, therefore, focuses on misbehavior that arises during protocol execution and that cannot be detected by simple channel guarantees alone.

As in [PKN+25], the IA security experiment mirrors the adaptive unforgeability experiment until the point where the adversary produces its output. In the unforgeability experiment, the adversary must output a fresh message together with a valid signature. In the IA experiment, the adversary instead outputs a session identifier sid and all signing and IA protocol messages associated with that session. The adversary wins if the signing transcripts do not combine into a valid signature and the Decide algorithm either fails to identify any malicious signer or incorrectly labels at least one honest signer as malicious.

Definition 2 (Identifiable Abort) For a threshold signature scheme with identifiable abort TS and an adversary $\mathcal{A}$ define

$$\mathsf{Adv}^{\mathsf{adp-IA}}_{\mathcal{A},\mathsf{TS}}(t_c, t, n) = \Pr[\mathsf{Exp}^{\mathsf{adp-IA}}_{\mathcal{A},\mathsf{TS}}(\lambda, t_c, t, n) \to 1],$$

where the probability is taken over the random choice of all randomized algorithms. TS is (t_c, t, n)-adaptively identifiably abortable if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{adp-IA}}_{\mathcal{A},\mathsf{TS}}(\lambda, t_c, t, n) = negl(\lambda)$. TS is fully-adaptively identifiable abortable if it is $(t, t-1, n)$-adaptively identifiable abortable for all $t \leq n \in \mathbb{N}$. We present the security experiment in Figure 3.

We defer further preliminaries including commitment schemes, and NIZKs to Appendix A.

5 Our Scheme

In this section, we describe our two-round threshold Schnorr signature scheme $\mathsf{FaFROST+}$. For a high-level overview, see Section 2. We provide pseudocode in Figures 4 and 5.

Setup and key generation. To set up $\mathsf{FaFROST}$ for parameters (λ, t, n), we select $(\mathbb{G}, p, g)$ and distinct hash functions $\mathsf{H}_{\mathrm{sig}}$, $\mathsf{H}_{\mathrm{non}}$, H_{s}, H_{vk}, H_{IA} that map bitstrings to $\mathbb{Z}_p$ scalars. The parties jointly compute a Schnorr verification key $vk = g^{sk}$ and distribute Shamir secret shares of sk amongst the signers. Each pair of signers $i \neq j$ also receives a uniformly random λ-bit string $k_{i,j} = k_{j,i} \leftarrow\!\!\$\ \{0,1\}^\lambda$. For the variant with identifiable aborts, we additionally compute a partial verification key consisting of $\mathsf{H}_{vk}(k_{i,j})$ for all $i \neq j$ and Pedersen commitments $\mathsf{Com}(sk_i, \omega_i)$. Otherwise, we set $vk_i = \perp$.

$\mathsf{adp} - \mathsf{IA}^{\mathsf{TS}}_{\mathcal{A}}(\lambda, t_c, t, n)$

$par \leftarrow \mathsf{Setup}(\lambda, t, n);\ (vk, \{(sk_i, vk_i)\}_{i\in[n]}) \leftarrow \mathsf{KeyGen}(par)$

$\textsc{corr} \leftarrow \emptyset, \textsc{hon} \leftarrow [n];\ Q_m \leftarrow \emptyset, Q_{st} \leftarrow \emptyset, Q_{\mathsf{IA}} \leftarrow \emptyset; \mathsf{sid} \leftarrow 0$

$O \leftarrow (\mathsf{OCorr}, \mathsf{OSR}_1, \mathsf{Sign}_2, \mathsf{OIA}_1, \mathsf{OIA}_2)$

$(\mathsf{sid}^*, S^*, m^*, \{(\mathsf{pm}_j, s_j, \mathsf{pm}'_j, d_j)\}_{j\in S}) \leftarrow \mathcal{A}^{O}(par, vk, \{vk_i\}_{i\in[n]})$

assert $\mathsf{CheckView}(Q_{\mathsf{IA}}, \mathsf{sid}^*, S^*, m^*, \{(\mathsf{pm}_j, s_j, \mathsf{pm}'_j, d_j)\}_{j\in S})$

$\sigma^* = \mathsf{Combine}(m^*, S^*, \{s_j\}_{j\in S})$

assert $\mathsf{Verify}(vk, m^*, \sigma^*) = 0$

$\mathcal{C} \leftarrow \mathsf{Decide}(vk, m^*, S^*, \{(vk_j, \mathsf{pm}_j, s_j, \mathsf{pm}'_j, d_j)\}_{j\in S})$

return $\mathcal{C} = \emptyset \vee \mathcal{C} \nsubseteq \textsc{corr}$

Figure 3: The adaptive identifiable abort security experiment. The function $\mathsf{CheckView}$ verifies that the observed views of each party are consistent, and these views match the adversarial outputs.

Signing rounds. Each participant samples two uniformly random nonces $(d_i, e_i) \leftarrow\!\!\$\ \mathbb{Z}_p^2$ and publishes the corresponding commitments $D_i \leftarrow g^{d_i}$ and $E_i \leftarrow g^{e_i}$. The nonces are stored locally and used in the second round. To run the second signing round, each signer reads a message m, a signing set S, first-round commitments $\{(D_j, E_j)\}_{j\in S}$, and a local state. It aggregates $D \leftarrow \prod_{j\in S} D_j$ and $E \leftarrow \prod_{j\in S} E_j$. It then derives $b \leftarrow \mathsf{H}_{\text{non}}(vk, S, m, \{\mathsf{pm}_j\}_{j\in S})$ and sets the Schnorr nonce $R \leftarrow D \cdot E^b$. Signer i computes the pairwise blinding values $B^s_{i,j} \leftarrow \mathsf{H}_{\text{s}}(k_{i,j}, \{\mathsf{pm}_j\}_{j\in S}, m, S)$ and combines them as $B^s_i = \sum_j B^s_{i,j} \cdot \Delta_{i,j}$, where $\Delta_{i,j} = 1$ if $i > j$ and -1 otherwise. Finally, i computes the partial signature $s_i \leftarrow c \cdot \Lambda_i sk_i + b \cdot e_i + d_i$ and outputs the blinded value $(R, s'_i \leftarrow s_i + B^s_i)$.

Signature aggregation and verification. Once a signer receives all partial signatures, the final signature consists of a pair $\sigma \leftarrow (R, s)$, with R defined as above, and s being the sum of all partial signatures $s \leftarrow \sum_{i\in S} s'_i$. The verification procedure is identical to a standard Schnorr signature verification; that is, one computes $c = \mathsf{H}_{\text{sig}}(vk, R, m)$ and checks that $vk^c \cdot R = g^s$.

Identifiable abort. To identify misbehavior during the signing protocol, each signer i commits deterministically to the blinding values $B^s_{i,j}$ into $\mathsf{C}_{i,j}$, and proves that the blinded partial signature s'_i is well-formed w.r.t. these commitments, the first-round protocol message (D_i, E_i), and the partial verification key vk_i. Upon receiving all these commitments and proofs, each signer i reveals all blinding keys $k_{i,j}$, for which the commitment $\mathsf{C}_{i,j}$ differs from $\mathsf{C}_{j,i}$. In the decision algorithm, a signer is marked malicious if either the revealed key $k_{i,j}$ is not consistent w.r.t. the partial verification key. If the revealed key is correct, the commitment $\mathsf{C}_{i,j}$ can be honestly recomputed, and signer j is marked malicious if $\mathsf{C}_{i,j} \neq \mathsf{C}_{j,i}$. For an intuitive description, we refer to Section 2.2. We provide pseudocode of the identifiable abort protocol in Figure 6.

Prototype. We empirically demonstrate the practicality of FaFROST+ through a prototype implementation and evaluation. In Section 7, we present the results of our experiments, which show that FaFROST+ incurs only minor overhead compared to FROST.

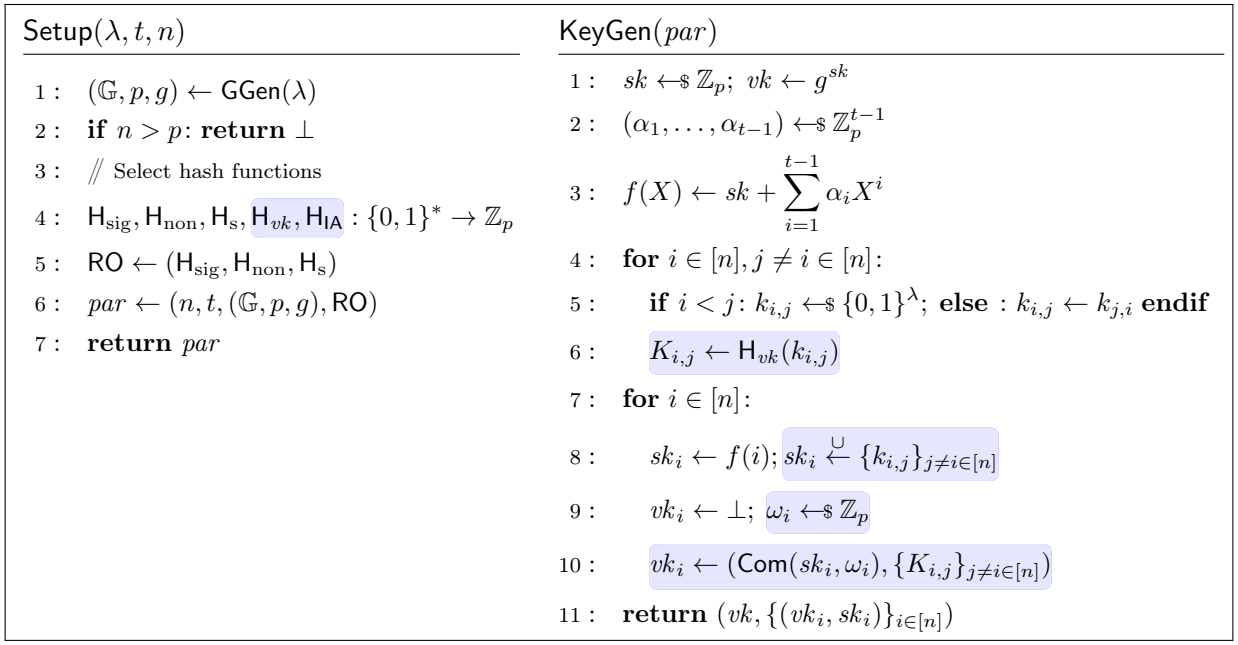

Figure 4: Setup and key generation for our scheme FaFROST. The group $\mathbb{G}$ has generator g and prime order p. We mark parts that appear only in the version with identifiable aborts.

6 Security Analysis

We prove the adaptive security of FaFROST+. We do this in two steps. First, in Theorem 1, we show that FaFROST, our intermediate scheme without identifiable aborts, achieves fully adaptive unforgeability under the AOMDL assumption in the AGM. Then, we show in Theorem 2 that FaFROST+ with identifiable aborts also achieves fully adaptive unforgeability, reducing to the result of Theorem 1. Finally, we show in Theorem 3 that our scheme securely realizes identifiable aborts (Definition 2).

Theorem 1 *Let* $\mathsf{H}_{\mathrm{sig}}, \mathsf{H}_{\mathrm{non}}$, *and* H_{s} *be hash functions modeled as random oracles. Let* $\mathbb{G}$ *be a cyclic group of prime order* p *with generator* g *and let* λ *be the security parameter. For all* $t_c < t \leq n \in \mathbb{N}$, *any algebraic PPT adversary* $\mathcal{A}$ *making at most* q_s *queries to* Sign_1 *and at most* $q_{\mathrm{sig}} + q_{\mathrm{non}} + q_{\mathsf{H}_{\mathrm{s}}}$ *random oracle queries, there is a PPT adversary* $\mathcal{B}$ *against AOMDL making at most* $2 \cdot q_s$ *queries to its oracle, such that* $\mathsf{Adv}^{\mathsf{adpTSUF}}_{\mathcal{A},\mathsf{FaFROST}}(t_c, t, n)$ *is upper-bounded by*

$$\mathsf{Adv}^{\mathsf{AOMDL}}_{\mathcal{B},(\mathbb{G},g,p)}(\lambda) + \frac{4 \cdot q_s^2}{p} + \frac{q_{\mathsf{H}_{\mathrm{s}}} \cdot n^2}{2^\lambda} + \frac{2 \cdot q_{\mathrm{sig}}}{p} + \frac{q_{\mathrm{sig}} \cdot t}{p(t - t_c)}.$$

Proof Let $\mathcal{A}$ be an adversary in the adaptive unforgeability experiment making at most q_s queries to the first-round signing oracle Sign_1 and up to t_c corruption queries. We construct an adversary $\mathcal{B}$ against the *algebraic one-more discrete logarithm (AOMDL)* assumption that simulates the view of $\mathcal{A}$. The adversary $\mathcal{B}$ receives $2q_s + 1$ challenge elements $X_0, \ldots, X_{2q_s}$ from its AOMDL challenger and is allowed at most $2q_s$ queries to the DL oracle.

If two of the received challenge elements are equal, then $\mathcal{B}$ can break the AOMDL assumption directly using $2q_s$ oracle calls. By the birthday bound, this happens with probability at most $4q_s^2/p$,

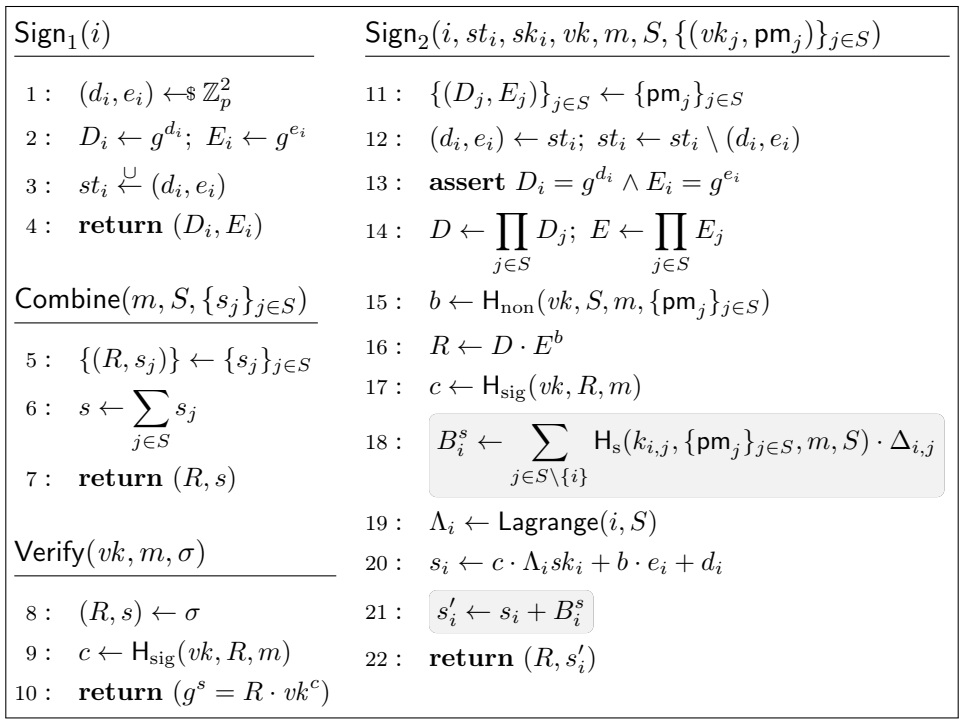

Figure 5: Pseudocode of FaFROST. Changes compared to FROST are highlighted. We set $\Delta_{i,j}$ as 1 if $i > j$, -1 otherwise. $\mathsf{Lagrange}(S, i)$ returns $\prod_{j\in S\setminus\{i\}} j/(j-i)$.

and hence is negligible. Looking ahead, the reduction embeds the first challenge into the verification key vk, while at the j-th call to Sign_1 it embeds the j-th and the $(q_s + j)$-th challenges, respectively, into the commitments (D_i, E_i) of an honest signer i.

Setup and key generation. The security experiment assumes a trusted key-generation phase during which no corruption occurs. The reduction $\mathcal{B}$ sets the global verification key to the first AOMDL challenge, namely $vk \leftarrow X_0$, and samples for every pair of distinct parties $1 \le i \ne j \le n$ a symmetric pairwise key $k_{i,j} = k_{j,i} \leftarrow\!\$\ \{0,1\}^\lambda$. $\mathcal{B}$ additionally maintains the following lists and variables:

- HON: initialized as $\{1, \ldots, n\}$, containing all honest parties.
- CORR: initialized as $\emptyset$; whenever $\mathcal{A}$ corrupts a party i, the reduction moves i from HON to CORR.
- sid: initialized as 0 is a counter, with which the adversary can choose which oracle interaction it refers to.
- Q_{st}: initialized as $\emptyset$; when $\mathcal{A}$ interacts with signing oracles, inputs and outputs are stored w.r.t. sid and the used signer index.
- Q_{RO}: records all random-oracle queries, distinguishing between the different oracles used in the protocol. $Q_{\mathsf{RO}}[\mathsf{id}]$ with $\mathsf{id} \in \{\mathrm{sig}, \mathrm{non}, \mathsf{s}\}$ stores a list of pairs (x, y), denoting that y was programmed in response to $\mathsf{H}_{\mathsf{id}}(x)$.

After this setup ends, $\mathcal{B}$ forwards vk to $\mathcal{A}$.

$\mathsf{IA}_1(i, sk_i, m, S, \{(\mathsf{pm}_j, s_j)\}_{j\in S})$

$b \leftarrow \mathsf{H}_{\mathrm{non}}(vk, S, m, \{\mathsf{pm}_j\}_{j\in S})$
for $j \neq i \in S$:
  $B^s_{i,j} = \mathsf{H}_{\mathrm{s}}(k_{i,j}, \{\mathsf{pm}_j\}_{j\in S}, m, S)$
  $\omega_{i,j} = \mathsf{H}_{\mathsf{IA}}(k_{i,j}, \{(\mathsf{pm}_j, s_j)\}_{j\in S})$
  $\mathsf{C}_{i,j} = \mathsf{Com}(B^s_{i,j}, \omega_{i,j})$
$\mathbb{x} \leftarrow (i, \mathsf{pm}_i, s_i, b, vk_i, \{\mathsf{C}_{i,j}\}_{j\neq i\in S})$
$\mathbb{w} \leftarrow (sk_i, \{(B^s_{i,j}, \omega_{i,j})\}_{j\neq i\in S})$
$\pi = \Pi.\mathcal{P}(\mathbb{x}, \mathbb{w})$
return $(\pi, \{\mathsf{C}_{i,j}\}_{j\neq i\in S})$

$\mathsf{IA}_2(i, sk_i, m, S, \{(\mathsf{pm}_j, s_j, \mathsf{pm}'_j)\}_{j\in S})$

parse sk_i **as** $(\cdot, \{k_{i,j}\}_{j\neq i\in[n]})$
parse pm'_i **as** $(\cdot, \{\mathsf{C}_{i,j}\}_{j\neq i\in S})$
$Q = \emptyset$; **for** $j \neq i \in S$:
  parse pm'_j **as** $(\cdot, \{\mathsf{C}'_{j,\ell}\}_{\ell\neq j\in S})$
  if $\mathsf{C}_{i,j} \neq \mathsf{C}'_{j,i} : Q \xleftarrow{\cup} (j, k_{i,j})$
return Q

$\mathsf{Decide}(m, S, \{(\mathsf{pm}_j, s_j, \mathsf{pm}'_j, d_j)\}_{j\in S})$

$\mathcal{C} = \emptyset$; $b \leftarrow \mathsf{H}_{\mathrm{non}}(vk, S, m, \{\mathsf{pm}_j\}_{j\in S})$
for $j \in S$: // Handle all accusations
  // Verify proofs of well-formedness
  $\mathbb{x}_j \leftarrow (j, \mathsf{pm}_j, s_j, b, vk_j, \{\mathsf{C}_{j,\ell}\}_{\ell\neq j\in S})$
  if $\Pi.\mathcal{V}(\mathbb{x}_j, \pi_j) = 0 : \mathcal{C} \xleftarrow{\cup} \{j\}$
  // Verify commitment correctness
  for $(\ell, k_{j,\ell}) \in d_j$:
    if $\mathsf{H}_{vk}(k_{j,\ell}) \neq K_{j,\ell} : \mathcal{C} \xleftarrow{\cup} \{j\}$
    else :
      $B^s_{j,\ell} = \mathsf{H}_{\mathrm{s}}(k_{j,\ell}, \{\mathsf{pm}_j\}_{j\in S}, m, S)$
      $\omega_{j,\ell} = \mathsf{H}_{\mathsf{IA}}(k_{j,\ell}, \{(\mathsf{pm}_\ell, s_\ell)\}_{\ell\in S})$
      if $\mathsf{C}_{\ell,j} \neq \mathsf{Com}(B^s_{j,\ell}, \omega_{j,\ell})$:
        $\mathcal{C} \xleftarrow{\cup} \{\ell\}$
// Detect adversarial pairs
for $j \neq \ell \in S$ **with** $\mathsf{C}_{j,\ell} \neq \mathsf{C}_{\ell,j}$:
  if $j \notin d_\ell : \mathcal{C} \xleftarrow{\cup} \{\ell\}$
  if $\ell \notin d_j : \mathcal{C} \xleftarrow{\cup} \{j\}$
return $\mathcal{C}$

Figure 6: Identifiable abort algorithms of FaFROST+. We remove (partial) verification keys from interface inputs for space reasons. To simplify readability, we assume that all $\mathsf{C}_{j,\ell}$, as well as π_j are parsed from pm'_j.

Simulating the random oracles. Whenever the adversary $\mathcal{A}$ queries one of the random oracles, say H_{id} on input x, $\mathcal{B}$ checks whether $Q_{\mathsf{RO}}[\mathsf{id}]$ contains a tuple of the form (x, y). If a match is found, $\mathcal{B}$ returns the previously stored output to ensure consistency. If no such pair exists, $\mathcal{B}$ samples a value uniformly at random from the oracle's range and verifies whether this new output is consistent with the simulation strategy. If any inconsistency arises, the reduction aborts. In Figure 7 we describe for each random oracle, what we would consider an inconsistency, and why such an abort only happens with negligible probability. If no inconsistency arises, $\mathcal{B}$ records the new (x, y) pair in Q_{RO}, and returns y.

Simulating Sign_1. When the adversary $\mathcal{A}$ queries the first-round signing oracle $\mathsf{OSR}_1(i)$ with an index i in the k-th first-round oracle query, the reduction $\mathcal{B}$ first checks whether i corresponds to an honest signer, i.e., whether $i \in \textsc{hon}$. If this is the case, $\mathcal{B}$ increases the session ID $\mathsf{sid} = \mathsf{sid} + 1$ and returns to $\mathcal{A}$ the k-th pair of unused DL challenges $(D_i, E_i) \leftarrow (X_k, X_{q_s+k})$, and sid. Finally, $\mathcal{B}$ records the index k in $Q_{st}[i, \mathsf{sid}]$ for future reference. This compact representation allows the reduction to retrieve the nonces $D_i \leftarrow X_k$ and $E_i \leftarrow X_{q_s+k}$ output in the k-th call to the oracle.

- Signature oracle H_{sig}. When $\mathcal{A}$ inputs a triple (vk, R, m) to H_{sig}, it must also provide its algebraic representation of R w.r.t. all group elements it has already learned, since $\mathcal{A}$ is an algebraic adversary. Let α_{vk} denote the coefficient associated with vk in this representation, i.e., $\mathcal{A}$ explains $R = vk^{\alpha_{vk}} \cdot \prod_i g_i^{\beta_i}$, where the g_i are all previously seen group elements. The reduction aborts if the first input is the challenge verification key vk, and $\mathsf{H}_{\text{sig}}(vk, R, m)$ equals $-\alpha_{vk}$. This happens with probability at most q_{sig}/p.[a]
- Nonce oracle H_{non}. When H_{non} is queried for the first time on input $(vk, S, m, \{\mathsf{pm}_j\}_{j\in S})$, the reduction computes the corresponding group element R according to the protocol specification. That is, it programs output $b \leftarrow\!\$\ \mathbb{Z}_p$, extracts the E and D terms from $\{\mathsf{pm}_j\}_{j\in S}$, and computes $R \leftarrow D \cdot E^b$. It then evaluates $\mathsf{H}_{\text{sig}}(vk, R, m)$ via lazy sampling.
- Blinding-key oracle H_{s}. The reduction aborts if $\mathcal{A}$ queries a blinding key $k_{i,j}$ to H_{s} that has not been previously leaked through corruption. Such an event occurs with probability at most $(q_{\mathsf{H}_{\text{s}}} \cdot n^2)/2^\lambda$, since there are at most n^2 possible keys of length λ and the adversary queries H_{s} at most $q_{\mathsf{H}_{\text{s}}}$ times.

[a] This is one of the two events covered by the term $\frac{2\cdot q_{\text{sig}}}{p}$ in Theorem 1. The other one is the adversary guessing the b_i values without querying H_{non}, which we bound in the full-rank analysis of the final matrix.

Figure 7: Simulation of the random oracles $\mathsf{H}_{\text{sig}}, \mathsf{H}_{\text{non}}$, and H_{s}.

Simulating Sign_2. When the adversary $\mathcal{A}$ queries the second-round signing oracle $\mathsf{OSR}_2(\mathsf{sid}, i, m, S, \{\mathsf{pm}_j\}_{j\in S})$, it provides sid, an index i indicating an honest signer, a message m, a set of participating signers S, and a set of first-round messages $\{\mathsf{pm}_j\}_{j\in S}$. The reduction $\mathcal{B}$ first verifies that i is an honest signer and that the second-round oracle was not yet queried for signer i on sid. It also checks, if $Q_{st}[i, \mathsf{sid}]$ is non-empty. If these checks fail, $\mathcal{B}$ returns $\perp$ as the oracle's output. This behavior is consistent with the real protocol, which rejects inputs unless the first-round messages satisfy $D_i = g^{d_i}$ and $E_i = g^{e_i}$, ensuring that only well-formed and consistent states are processed, and removes the values d_i, e_i after using them once (Line 12 in Figure 5).

After verifying the input consistency, the reduction $\mathcal{B}$ determines how many other honest signers in the current signing set S are expected to respond after this oracle call to the same set $\{\mathsf{pm}_j\}_{j\in S}$ on first-round messages. Let $i_1, \ldots, i_{|S\cap \text{HON}|}$ denote a permutation of the current honest signing set $S \cap \text{HON}$, such that for all $j \in S \cap \text{HON}$, i_j is the index of the j-th honest signer that $\mathcal{A}$ has caused to execute the second round on $\{\mathsf{pm}_j\}_{j\in S}$. We informally call $i_{|S\cap \text{HON}|} = i^*$ the *last* honest signer. For every honest signer $i \in S \cap \text{HON}$ except the last in this session, i.e., if $i \neq i^*$, the reduction samples a random value $s_i \leftarrow\!\$\ \mathbb{Z}_p$ and adds the appropriate blinding contributions from all corrupted signers $s'_i \leftarrow s_i + \sum_{j\in S\cap \text{CORR}} \mathsf{H}_{\text{s}}(k_{i,j}, \{\mathsf{pm}_j\}_{j\in S}, m, S) \cdot \Delta_{i,j}$. Computing the shared blinding values with the corrupted signers is feasible, since for all $i \in \text{HON}$ and $j \in \text{CORR}$, the keys $k_{i,j}$ are known to the reduction. The reduction returns s'_i to $\mathcal{A}$ and stores $(\mathsf{sid}, i, m, S, \{\mathsf{pm}_j\}_{j\in S}, s'_i)$ in $Q_{st}[i, \mathsf{sid}]$.

For the last honest signer, i.e., if $i = i^*$, the reduction $\mathcal{B}$ ensures that the resulting protocol messages remain consistent. Note that if $\mathcal{A}$ previously queried the second-round signing oracle on inconsistent first-round protocol messages for any honest signer, the consistency step is never triggered: a last honest signer only exists if all honest parties in S received the same first-round messages. If *inconsistent* first-round messages were sent when querying the second-round signing oracle earlier, then at least one honest signer in S responded with this inconsistent set, and our blinding strategy ensures that in an honest execution of the protocol, the resulting blinding values do not cancel out. If all honest signers receive the same first-round messages, the last honest signer needs to maintain consistency. To achieve consistency, the reduction $\mathcal{B}$ computes the aggregate

random commitment associated with all the honest signers in the current signing set S:

$$R_{\text{HON}} \leftarrow \prod_{j \in S \cap \text{HON}} D_j \cdot E_j^b,$$

where b is obtained from the random oracle H_{non} as specified in the protocol (Line 15 in Figure 5). It then simulates the computation of the challenge $c = \mathsf{H}_{\text{sig}}(vk, R, m)$ (cf. Figure 7), queries the DL oracle on the value $vk^c \cdot R_{\text{HON}}$, and obtains the response $\tilde{s}$. To make $\tilde{s}$ consistent with the effective partial signature s_{HON} on behalf of all the honest signers in S, it then computes $s_{\text{HON}} \leftarrow \tilde{s} - c \cdot \left(\sum_{j \in S \cap \text{CORR}} \Lambda_j \cdot sk_j\right)$. Next, $\mathcal{B}$ ensures that the value s_{HON} is consistently shared among all honest signers by setting $s_i \leftarrow s_{\text{HON}} - \sum_{j \in S \cap \text{HON} \setminus \{i\}} \left(s'_j - \sum_{k \in S \cap \text{CORR},\, k \neq j} B^s_{j,k} \cdot \Delta_{j,k}\right)$. This guarantees that the sum of all (unblinded) honest pre-signature shares equals the effective partial signature s_{HON}, as required by the protocol. Finally, the reduction $\mathcal{B}$ incorporates the blinding terms. The blinding terms between honest signers are handled implicitly: since the values s_i were chosen uniformly at random but in a way that preserves overall consistency, these already incorporate the effect of pairwise blindings among honest parties. Consequently, the reduction $\mathcal{B}$ does not add explicit blinding terms for honest pairs. However, blindings involving corrupted signers do not cancel, and must therefore be added explicitly to maintain consistency with the adversary's view (as also done above). For each honest signer i, the reduction $\mathcal{B}$ adds the due blinding factors to the pre-blinding share s_i, resulting in the final share $s'_i \leftarrow s_i + \sum_{j \in \text{CORR} \cap S,\, j \neq i} \mathsf{H}_{\text{s}}(k_{i,j}, \{\mathsf{pm}_j\}_{j \in S}, m, S) \cdot \Delta_{i,j}$. Our signing strategy yields a perfect simulation of the signing oracles: If at least two honest signers remain, there exists at least one pairwise key $k_{i,j}$ unknown to the adversary. If it would query the oracle H_{s} on this key, the reduction would abort. Moreover, the first-round protocol messages $\{\mathsf{pm}_j\}_{j \in S}$ contain at least a single honest group-element E, and no two honest E, E' are equal by our current abort strategy. Consequently, H_{s} was never queried before on $(k_{i,j}, \{\mathsf{pm}_j\}_{j \in S}, m, S)$, and the corresponding output of H_{s} appears to be uniformly random.

From the adversary's perspective, each honest signer's partial signature is therefore distributed uniformly at random and independent of any secret key. When combined with the corrupted parties' contributions, the aggregate of all partial signatures equals $sk \cdot c + d + b \cdot e$, producing a valid Schnorr signature as required by the protocol. If only a single honest signer remains, this signer necessarily acts as the last consistent signer, and our simulation strategy ensures that its blinded partial signature is both valid and indistinguishable from an honestly generated one. Furthermore, an adversary cannot trigger $\mathcal{B}$ to waste DL oracle calls by sending inconsistent inputs, as only the last honest consistent signer queries the DL oracle. For all signing oracle calls, $\mathcal{B}$ stores the inputs and outputs in $Q_{st}[i, \mathsf{sid}, 2]$. Additionally, it stores the values s_{HON}, R_{HON} for the last consistent signer. In total, the signing simulation includes $2 \cdot |S \cap \text{HON}| \leq 2|S|$ DL challenges and requires a single call to the DL oracle.

Corruptions. When the adversary queries the corruption oracle on an index i, the reduction $\mathcal{B}$ first verifies that i is still an honest signer, i.e., $i \in \text{HON}$. If i has already been corrupted, the oracle aborts. If there exist signing sessions, in which signer i is the last honest signer, but has not yet answered Sign_2, the reduction first runs the Sign_2 simulation for i as described above. This guarantees that the answers of all honest signers of this round are consistent. We note that by first simulating Sign_2, our reduction queries the DL oracle once. This is in accordance with our simulation strategy, in which each signing session uses a single DL oracle query (cf. Table 2). Once consistency

for all open signing sessions is ensured, the reduction $\mathcal{B}$ samples a fresh, uniformly random signing key $sk_i \leftarrow\!\!\$\ \mathbb{Z}_p$. Then, for all $k \in Q_{st}[i,\cdot,1]$, $\mathcal{B}$ queries the DL oracle on input X_k (resp. X_{q_s+k}) to obtain the corresponding value d_i (resp. e_i) and appends them to $Q_{st}[i,\cdot,1]$ accordingly. Next, $\mathcal{B}$ ensures that all previously generated second-round messages (i.e., the blinded partial signatures) remain consistent. For each record $(m, S, \{\mathsf{pm}_j\}_{j\in S}, s'_i)$ in $Q_{st}[i,\cdot,2]$, it reconstructs the honest partial signature $s''_i \leftarrow \Lambda_i \cdot sk_i \cdot c + d_i + b \cdot e_i$, that the adversary should now be able to compute after corruption, where c is the corresponding Fiat–Shamir challenge, b is the nonce hash (both can be recomputed from $vk, \{\mathsf{pm}_j\}_{j\in S}, m, S$), and d_i, e_i are the values recovered from the DL oracles on the same sid. The value s''_i represents the unblinded signature share that $\mathcal{A}$ would recompute upon receiving sk_i. Since after the corruption, the commitments (D_i, E_i) and sk_i are public, it is required that removing the blinding from the previously returned value s'_i yields exactly s''_i. Finally, $\mathcal{B}$ adjusts the blinding values so that this consistency holds: the previously output (blinded) signature share s'_i must equal the sum of the unblinded signature share s''_i and the combined blinding contributions from all other signers. We now describe the step that restores consistency for the blinded partial signatures. First, $\mathcal{B}$ computes the correction term δ_i required to make the previously output blinded signature share s'_i consistent with the unblinded s''_i:

$$\delta_i \leftarrow s'_i - s''_i, \tag{1}$$

which represents the sum of all pairwise blinding contributions expected for signer i. Next, $\mathcal{B}$ determines the portion of this blinding that is already known to the adversary, namely the contributions corresponding to corrupted signers (whose pairwise keys are known):

$$B_i^{s,\text{CORR}} \leftarrow \sum_{j\neq i \in \text{CORR}\cap S} \mathsf{H}_\text{s}(k_{i,j}, \{\mathsf{pm}_j\}_{j\in S}, m, S) \cdot \Delta_{i,j}.$$

For each remaining honest signer $j \in S \cap \text{HON}$, except for one designated last honest party i^*, $\mathcal{B}$ samples the corresponding blinding terms uniformly at random:

$$B^s_{i,j} = \mathsf{H}_\text{s}(k_{i,j}, \{\mathsf{pm}_j\}_{j\in S}, m, S) \leftarrow\!\!\$\ \mathbb{Z}_p.$$

If i itself is the last remaining honest signer, we can skip this programming step, since the adversary already knows all blinding factors; hence, we made the output consistent in a previous step. Otherwise, i.e., prior to corruption at least two honest signers remained in this signer set, $\mathcal{B}$ reprograms the random oracle for the remaining pairwise key k_{i,i^*} so that the combined blinding sums to δ_i:

$$B^s_{i,i^*} = \mathsf{H}_\text{s}(k_{i,i^*}, \{\mathsf{pm}_j\}_{j\in S}, m, S) = \Big(\delta_i - \sum_{\substack{j\in S\\ j\neq i, j\neq i^*}} B^s_{i,j} \cdot \Delta_{i,j}\Big) \cdot \Delta_{i,i^*}.$$

Note that for making the last honest signer's blinding consistent, we use all other remaining blindings for the signing set, and not only the corrupted ones. Combining the unblinded partial signature s''_i and all blindings, we now have $\delta_i = \sum_{j\in S, j\neq i} B^s_{i,j} \cdot \Delta_{i,j}$, as required by Equation (1). Hence, we made our uniformly random signing consistent with the previously output signature shares.

$$\begin{aligned}\sum_{j\in S, j\neq i} B^s_{i,j} \cdot \Delta_{i,j} &= \sum_{j\in S\setminus\{i,i^*\}} B^s_{i,j} \cdot \Delta_{i,j} + B^s_{i,i^*} \cdot \Delta_{i,i^*} \\ &= \sum_{j\in S\setminus\{i,i^*\}} B^s_{i,j} \cdot \Delta_{i,j} + \delta_i - \sum_{j\in S\setminus\{i,i^*\}} B^s_{i,j} \cdot \Delta_{i,j} = \delta_i.\end{aligned}$$

Since we only do this programming if at least one pairwise key k_{i,i^*} remains unknown to the adversary, and the first-round protocol messages are guaranteed to be unique for each signer ($\mathcal{B}$ would return $\perp$ if an honest signer's nonce is reused in $\{\mathsf{pm}_j\}_{j\in S}$), the corresponding input has never been queried to H_s before. Hence, this reprogramming step is consistent with the random-oracle model and indistinguishable from a freshly sampled response.
Eventually, the adversary $\mathcal{A}$ outputs a signature forgery $(R^*, s^*), m^*$, for which holds $c^* = \mathsf{H}_{\mathrm{sig}}(vk, R^*, m^*)$. In addition, $\mathcal{A}$ outputs a representation $(\alpha_{vk}, \alpha_g, \{(\beta_i, \gamma_i)\}_{1\le i\le q_s})$ of R^* w.r.t. the group elements vk, g, $\{(D_i, E_i)\}_{1\le i\le q_s}$, such that

$$R^* = vk^{\alpha_{vk}} \cdot g^{\alpha_g} \cdot \prod_{1\le i\le q_s} D_i^{\beta_i} \cdot \prod_{1\le i\le q_s} E_i^{\gamma_i}$$

for $\alpha_{vk}, \alpha_g, \beta_i, \gamma_i \in \mathbb{Z}_p$.

Breaking the AOMDL assumption. It remains to show that we can solve the AOMDL problem whenever the adversary outputs a valid forgery, except with negligible probability. To make this connection precise, we frame the entire reduction as maintaining and solving a system of linear equations over unknowns from $\mathbb{Z}_p$. Whenever the reduction embeds an AOMDL challenge element in the simulation, it adds an unknown to the system of equations, because the DL of this element is unknown. Whenever the reduction calls the DL oracle, a new linear equation, and hence, a new constraint, is added to the system of equations. In this view,

- every first-round signing oracle call corresponds to adding two unknowns (d_i, e_i) to the system;
- every second-round oracle call, that relates to a last honest signer, corresponds to adding a new equation to the system;
- during corruption, each DL oracle query reveals the value of one unknown, and hence removes it from the set of unknowns.

The reduction succeeds if, at the time of the forgery, it has gathered enough linearly independent equations to solve for all remaining unknowns while having made at most $2 \cdot q_s$ many DL oracle calls. We show that this is the case with overwhelming probability.

Init. If $\mathcal{A}$ never queries the first-round signing oracle ($q_s = 0$), the only existing DL challenge is vk. Since there is also no second-round signing oracle call, the reduction never needs to query the DL oracle to attain consistency. When the adversary outputs a forgery (R^*, s^*), it also provides an algebraic representation (α_{vk}, α_g) of R^* that can only depend on vk and g, since the reduction never outputs any other group element. The signature forgery is valid, therefore $vk^{c^*} \cdot R^* = g^{s^*}$. Then, we have

$$\big(c^* + \alpha_{vk}, \alpha_g\big) \cdot \big(sk, 1\big)^T = s^*$$

for known α_{vk}, α_g, and c^*. This yields a unique solution for sk, because if $c^* = -\alpha_{vk}$, our reduction would have already aborted as described in Figure 7. $\mathcal{B}$ solves a single DL and never queries the DL oracle and, hence, breaks the AOMDL assumption.

First-round signing oracles. For each first-round oracle call, our system of linear equations obtains two unknowns. For the k-th oracle query, these are (d_k, e_k), which are the discrete logarithms to the challenges (D_k, E_k). If the adversary outputs its forgery after only first-round oracle calls,

our linear system has the form

$$(c^* + \alpha_{vk}, \beta_1, \cdots, \beta_{q_s}, \gamma_1, \cdots, \gamma_{q_s}, \alpha_g) \cdot (sk, d_1, \ldots, d_{q_s}, e_1, \ldots, e_{q_s}, 1)^T = s^*$$

for known $c^*, (\beta_k, \gamma_k), \alpha_{vk}, \alpha_g$. If the adversary does not query q_s many first-round oracle calls, we set the corresponding values β_k, γ_k to zero. Since the adversary did not call the second-round signing oracle, the reduction was not required to query the DL oracle to simulate the last consistent signer. Hence, it can solve the DL of $2 \cdot q_s$ elements using the DL oracle. Furthermore, $c^* + \alpha_{vk}$ is non-zero (Figure 7). Therefore, the reduction solves all d_k, e_k using the DL oracle, and sk using the resulting linear system.

Second-round signing oracles. Let $d_i^{(j)}$ and $e_i^{(j)}$ denote the discrete logarithms of the j-th pair of nonces received by the adversary, for the i-th session, where a session consists of all sign-oracle queries w.r.t. the same message m, signer set S, and first-round protocol messages $\{\mathsf{pm}_j\}_{j \in S}$.

When the adversary queries the second-round signing oracle for the last-honest signer, our reduction uses a DL oracle to compute the partial signature for that signer. At the same time, this partial signature of the last honest signer adds an equation to our system of linear equations. Recall that for the last-honest-signer to be consistent, our reduction computes the honest parties' combined share of the signature as

$$s_{\text{HON},i} = sk_{\text{HON},i} \cdot c_i + \sum_{j \in S \cap \text{HON}} d_i^{(j)} + e_i^{(j)} \cdot b_i.$$

When t honest parties are in the i-th session, this adds a constraint of the form

$$\big(c_i, 1, \ldots, 1, b_i, \ldots, b_i\big) \cdot (sk, d_i^{(1)}, \ldots, d_i^{(t)}, e_i^{(1)}, \ldots, e_i^{(t)})^T = z_i$$

to our system of linear equations. The challenges c_i and b_i are outputs of the random oracle as described in the simulation, and z_i is computed from the DL oracle call. The above constraint can be adapted when the number of honest parties is not t (it must be at least $t - t_c$). To ease readability, we will omit the vector containing the unknown values and solutions from this point forward, focusing solely on the coefficient matrix. Furthermore, we drop the last column, which refers to the known DL α_g, and therefore does not refer to an unknown. After m last-consistent-signer queries, we can combine the constraints into a block-diagonal matrix. We provide an example with $m = 2$, $t = 3$.

$$\begin{pmatrix} c_1 & 1 & 1 & 1 & 0 & 0 & 0 & b_1 & b_1 & b_1 & 0 & 0 & 0 \\ c_2 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & b_2 & b_2 & b_2 \end{pmatrix}.$$

Challenge random-oracle call. At some point, the adversary (or the challenger) queries the signing random oracle on input (vk, R^*, m^*) and receives the value c^* that will later appear in the forgery. Together with this query, the adversary (or, if needed, the challenger) provides a representation of R^* as a linear combination of the group elements known so far, namely vk, g, and all D_i and E_i. This representation fixes the coefficients of the corresponding linear equation, but it does not yet yield a complete constraint since the solution s^* remains unknown until the adversary outputs its forgery. If the adversary never makes this query, the verifier's check performed by the challenger when validating the forgery serves as the relevant random-oracle call, leading to the same

outcome for the reduction. At the moment of the challenge random-oracle query, we append to the coefficient matrix a row of the form (for our running example)

$$(c^* + \alpha_{vk}\, \beta_1^{(1)}\, \beta_1^{(2)}\, \beta_1^{(3)}\, \beta_2^{(1)}\, \beta_2^{(2)}\, \beta_2^{(3)}\, \gamma_1^{(1)}\, \gamma_1^{(2)}\, \gamma_1^{(3)}\, \gamma_2^{(1)}\, \gamma_2^{(2)}\, \gamma_2^{(3)}).$$

For readability, we omit the final column corresponding to the known discrete logarithm α_g, which is not an unknown. Let $\beta_i^{(j)}$ and $\gamma_i^{(j)}$ denote the coefficients of $d_i^{(j)}$ and $e_i^{(j)}$, respectively; these coefficients correspond to exponents in the algebraic representation of group elements. If, at the time of the query, the adversary has not yet learned certain challenge elements or omits them from its representation, the corresponding coefficients are zero. We refer to this matrix as the *challenge matrix*, which will play a central role in the subsequent analysis. In the actual simulation, there would be many additional rows, one for each query to the signing random oracle H_{sig} on inputs (vk, R, m) together with the corresponding representation of R. Only after the adversary outputs a forgery (R^*, s^*, m^*) does the reduction retrospectively identify the *challenge* random oracle call as the first query to $H_{\text{sig}}(vk, R^*, m^*)$ that returned c^*. For exposition, we display only the row corresponding to this challenge call and omit all other irrelevant oracle queries.[3]

Corrupting a signer. When the adversary corrupts a signer, we follow our simulation strategy and solve all $D_i^{(j)}$ and $E_i^{(j)}$ that this signer has output in any prior first-round signing query. Since the corresponding discrete logarithms $d_i^{(j)}$ and $e_i^{(j)}$ are now known, we can remove them from the set of unknowns in our system of equations. We rewrite the corresponding equations by setting the coefficients of the $d_i^{(j)}$s and $e_i^{(j)}$s to zero, and updating the solutions of these equations accordingly. This strategy also works if the adversary corrupts parties before making the challenge-oracle call, as we can perform the zeroing afterward.

The forgery matrix. Eventually, $\mathcal{A}$ outputs its forgery and cannot corrupt any party thereafter. In our matrix representation, this forgery corresponds to the reduction learning a solution for the challenge row (i.e., the row containing the challenge RO-query representation) of the challenge matrix. We refer to the matrix at the time of the forgery as the *forgery matrix*. If, in the second oracle simulation, we had to reply m times as the last consistent signer (we call this a last-consistent signer oracle call) before the adversary outputs the forgery, the resulting forgery matrix has $m + 1$ rows. This dimensionality arises from the m consistency conditions provided by the last-consistent-signer constraints, plus the one additional constraint yielded by the valid forgery itself. The matrix includes one column for every honest signer in every signing session, resulting in a potentially higher number of unknowns than the number of constraints. In the remainder of the proof, we transform the $(m + 1) \times x$ *forgery matrix* (with $x \geq m + 1$) into an $(m + 1) \times (m + 1)$ *final matrix* of full rank such that we can solve for all remaining unknowns using standard techniques like Gaussian elimination. To see that we have enough leftover DL oracle calls to remove the additional columns, note that for every challenge element except the verification key, each DL oracle call either (i) adds a row to the matrix, or (ii) removes an unknown and thus removes a column from the system of equations. As a result, the matrix ends up with the same number of rows and columns. The extra column for the verification key is balanced by the additional row obtained from the representation that the adversary provides with the forgery. We give an overview of the challenge embedding and DL oracle call progression in Table 2.

[3]This is w.l.o.g., as $\mathcal{B}$ only derives information from the challenge random oracle call after the forgery has occurred,

Event / Oracle Call	New Unk.	New Const.	DL Queries
Key generation	1 (sk)	0	0
First-round signing oracle	2 (d_i, e_i)	0	0
Second-round signing (non-last)	0	0	0
Second-round signing (last-hon-sig)	0	1	1
Corruption of a signer	$-2 \cdot \mathsf{OSR}_1(i)$	0	$2 \cdot \mathsf{OSR}_1(i)$
Forgery (final signature)	0	1	0

Table 2: Counting unknowns, constraints, and DL oracle queries in the reduction. The number $\mathsf{OSR}_1(i)$ denotes the number of first-round signing calls for signer i.

We only use DL queries during last-consistent signer oracle calls and during corruption. For last-consistent signer oracle calls, we use a DL call to add a row to the matrix that is always linearly independent from all other last-consistent-signer rows, because of their block-diagonal structure. During corruption, we use DL calls to remove columns. We never remove a row because there is always at least one honest signer left per signing session. In other words, when the adversary outputs its forgery, we can solve all but one embedded unknown in each last-consistent-signer session and all remaining challenge elements not used in a last-consistent-signer session.

Constraints on the forgery matrix. We will use the remaining DL oracle calls to remove columns by solving individual challenge elements, similar to how we handle corruptions. To ensure that we end up with a matrix of full rank, we distinguish between different scenarios to decide which columns to remove. We begin with two cases, where the full rank of the final matrix follows directly from the structure of the forgery matrix. For each case, we provide examples with $m = 2$ for better readability, but the results hold for arbitrary m. We defer the general matrices to Appendix B.

1. $\exists(i,j) : \gamma_i^{(j)} \neq b_i \cdot \beta_i^{(j)}$: For all sessions $\ell \neq i$, the reduction solves all $e_\ell^{(\cdot)}$ and all but one $d_\ell^{(\cdot)}$. For the session i, the reduction solves all $e_i^{(\cdot)}, d_i^{(\cdot)}$ except for $e_i^{(j)}, d_i^{(j)}$. The reduction uses the remaining DL oracle call of session i to solve for sk. We depict the resulting matrix M_{b_i} for $i = 1$ in Equation (2). M_{b_i} has full rank since $\gamma_1 \neq b_1 \cdot \beta_1$, and hence $(\beta_1, \gamma_1) \neq (0, 0)$.
2. $\exists i : \exists j \neq k : \beta_i^{(j)} \neq \beta_i^{(k)}$: For all sessions $\ell \neq i$, the reduction solves all $e_\ell^{(\cdot)}$ and all but one $d_\ell^{(\cdot)}$. For the session i, the reduction solves all $e_i^{(\cdot)}$ and all $d_i^{(\cdot)}$ except for $d_i^{(j)}$ and $d_i^{(k)}$. With the leftover DL oracle call for session i, the reduction solves for sk. We depict the resulting matrix M_β for $i = 1$ in Equation (2). It has full rank since $\beta_1^{(j)} \neq \beta_1^{(k)}$.

$$M_{\mathrm{b}_i} = \begin{pmatrix} 1 & 0 & b_1 \\ 0 & 1 & 0 \\ \beta_1 & \beta_2 & \gamma_1 \end{pmatrix}; \; M_\beta = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ \beta_1^{(j)} & \beta_1^{(k)} & \beta_2 \end{pmatrix}; \; M_{\text{final}} = \begin{pmatrix} c_1 & 1 & 0 \\ c_2 & 0 & 1 \\ c^* + \alpha_{vk} & \beta_1 & \beta_2 \end{pmatrix} \tag{2}$$

We know that within a session, all β_i's are the same, and for all γ_i's it must hold that $\gamma_i = b_i \cdot \beta_i$. Otherwise, we say that $\mathcal{A}$ provides a forgery matrix with trivial non-linearity. By this, we mean that our reduction can solve the AOMDL problem using Gaussian elimination, as the forgery matrix has

and the omitted rows do not influence our reasoning.

full rank. Since we have no trivial non-linearity, we can write the challenge matrix as

$$\begin{pmatrix} c_1 & 1 & 1 & 0 & 0 & 0 & 0 & b_1 & b_1 & 0 & 0 & 0 \\ c_2 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & b_2 & b_2 & b_2 \\ c^* + \alpha_{vk} & \beta_1 & \beta_1 & 0 & \beta_2 & \beta_2 & \beta_2 & b_1 \cdot \beta_1 & b_1 \cdot \beta_1 & b_2 \cdot \beta_2 & b_2 \cdot \beta_2 & b_2 \cdot \beta_2 \end{pmatrix}.$$

For all sessions ℓ[4], the reduction solves all $e_\ell^{(\cdot)}$ and all but one $d_\ell^{(\cdot)}$. We depict the resulting *final matrix* M_{final} in Equation (2).

It remains to show that the final matrix has full rank. Due to the $m \times m$ identity submatrix in the top-right corner, it suffices to ensure that the last row is not in the span of the first m ones. This holds if

$$c^* \neq \sum_{i=1}^{m} \beta_i \cdot c_i - \alpha_{vk}.$$

The values β_i, c_i, and α_{vk} are already fixed when c^* is sampled uniformly at random with overwhelming probability. The values c_i are fixed when the b_i values are fixed, due to our random-oracle simulation of H_{non} (Figure 7). Since we are not in Case 1, the adversary must know the b_i values used by honest signers. The only way to achieve this *without* querying H_{non} prior to the challenge RO call is by correctly guessing. Such a guess succeeds with probability at most q_{sig}/p, as each potential challenge RO call has probability of $1/p$ of guessing b_i correctly, and there are at most q_{sig} possible challenge RO calls. Furthermore, the algebraic representation of R^* determines all $\beta_i^{(j)}$ and α_{vk}. The last degree of freedom for the adversary is to pick one β_i out of the possible $\beta_i^{(j)}$ for each session i. In Section 6 we show that this transformation admits at most $t/(t-t_c)$ possible choices in total. Since there are at most q_{sig} different challenge matrices, this increases the adversary's success probability in failing the reduction by

$$\frac{q_{\text{sig}} \cdot t}{(t - t_c)\, p}.$$

This concludes our proof. We have demonstrated that if the reduction does not abort, it outputs $2 \cdot q_s + 1$ solutions to the AOMDL problem while querying the DL oracle at most $2 \cdot q_s$ times. Additionally, we have demonstrated that the probability of aborting the reduction is negligible. This demonstrates the unforgeability of our scheme against fully adaptive adversaries.

Possible matrix transformations. In the following, we show that in the proof of Theorem 1 a single challenge matrix results in at most $t/(t - t_c)$ possible forgery matrices without imposing a trivial non-linearity (i.e., violating Case 2).

We start with a warmup example with $t = 3$, $t_c = 2$, $n = 5$, and three signing sessions. For our argument, we focus on the challenge vector, i.e., the row of the challenge matrix that contains the adversarially chosen representation. For readability, we represent each signing session as a subvector of size n and set the value of non-participating, or corrupted, signers to 0. For our example, we assume the following:

- Session 1: $\mathcal{A}$ used the same β_1 for parties $(1, 2)$, and different for 3.
- Session 2: $\mathcal{A}$ used the same β_2 for parties $(3, 4)$, and different for 2.
- Session 3: $\mathcal{A}$ used the same β_3 for parties 3, 4, and 5.

[4]Note that a session consists of all sign-oracle queries w.r.t. the same message m, signer set S, and first-round protocol messages $\{\mathsf{pm}_j\}_{j \in S}$.

This results in the vector $\vec{v}$ defined as

$$\vec{v} = ((\beta_1^{(1)}, \beta_1^{(1)}, \beta_1^{(2)}, 0, 0), (0, \beta_2^{(1)}, \beta_2^{(2)}, \beta_2^{(2)}, 0), (0, 0, \beta_3^{(1)}, \beta_3^{(1)}, \beta_3^{(1)})).$$

As discussed above, the adversary has to select a single value β_i for each signing session. We denote the values $\beta_i^{(j)}$ as the candidate value for session i. We say a value candidate value is final, if by the provided corruption, this value cannot be changed for a session. We will examine all possible combinations of candidate values in the final matrix by iterating over all sessions and all choices of candidate values. For the first signing session $(\beta_1^{(1)}, \beta_1^{(1)}, \beta_1^{(2)}, 0, 0)$, there are two choices $\beta_1^{(1)}$ or $\beta_1^{(2)}$. If the adversary wants to pick $\beta_1^{(1)}$, it needs to corrupt party three (We discuss the choice of $\beta_1^{(2)}$ below). After corrupting party three, the updated vector $\vec{v}'$ has the form

$$\vec{v}' = ((\beta_1^{(1)}, \beta_1^{(1)}, 0, 0, 0), (0, \beta_2^{(1)}, 0, \beta_2^{(2)}, 0), (0, 0, 0, \beta_3^{(1)}, \beta_3^{(1)})),$$

and the adversary has one corruption left. The second signing session subvector has been modified due to the corruption introduced in the first choice and now has the form $(0, \beta_2^{(1)}, 0, \beta_2^{(2)}, 0)$. In this updated second session, the adversary can either choose $\beta_2^{(1)}$ by corrupting party four, or choose the value $\beta_2^{(2)}$ by corrupting party two. In either case, the adversary consumes the last corruption. Now, the adversary has no freedom of choice, so in all subsequent sessions, there can be only a single candidate left. Otherwise, the condition of the forgery matrix would be violated. This is the case in our example, so this choice of the adversary provides the possible solutions

$$\{(\beta_1^{(1)}, \beta_2^{(1)}, \beta_3^{(1)}), (\beta_1^{(1)}, \beta_2^{(2)}, \beta_3^{(1)})\}.$$

To select the second possible value in the first set, the adversary needs to corrupt parties one and two. The remaining vector of updated sessions then has the form

$$\vec{v}'' = ((0, 0, \beta_1^{(2)}, 0, 0), (0, 0, \beta_2^{(2)}, \beta_2^{(2)}, 0), (0, 0, \beta_3^{(1)}, \beta_3^{(1)}, \beta_3^{(1)}))$$

and the adversary has no corruptions left. Again, because of the condition on the forgery matrix, it is ensured that once the adversary has no corruptions left, the subsequent sessions can only have a single candidate value each. Therefore, the set of possible choices expands to

$$\{(\beta_1^{(1)}, \beta_2^{(1)}, \beta_3^{(1)}), (\beta_1^{(1)}, \beta_2^{(2)}, \beta_3^{(1)}), (\beta_1^{(2)}, \beta_2^{(2)}, \beta_3^{(1)})\}.$$

There are 3 possible choice combinations, which is less than or equal to $t/(t - t_c)$. We depict the adversary's choices in a tree diagram in Figure 8.

With this intuition in mind, we now prove that our claim holds. I.e., we show that if the adversary does not impose a trivial non-linearity, there are at most $t/(t - t_c)$ many different forgery matrices. We do this in two steps. First, we present a *local bound*: the maximum number of different candidate values within a (possibly updated) signing session depends only on the number of remaining honest signers t_i and the remaining corruption capability $c_{\mathsf{max},i}$, which initially equals t_c. Whenever a corruption occurs, $c_{\mathsf{max},i}$ is reduced by the number of consumed corruptions. Then, we use this local bound to derive a *global bound*. As the adversary progresses through the signing sessions, each choice of a candidate value consumes part of its remaining corruption capability, thereby reducing the freedom for subsequent sessions. Formally, we show that the sum of the maximum number of choices for all successor sessions never exceeds the maximum number of choices for the current

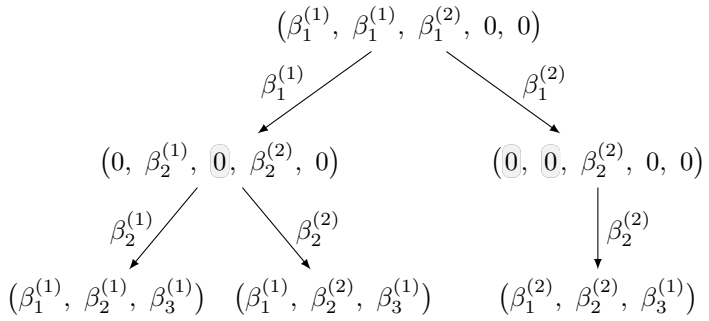

Figure 8: Adversarial branching by selecting candidate values. We omit the third level, as in our example, this level only exposes a single candidate value $\beta_3^{(1)}$.

session. We define the maximum number of choices independent of the actual values $\beta_i^{(j)}$, but only with respect to the remaining corruption capabilities. This invariant ensures that, although later sessions may present multiple branching options, the total potential is conserved; combined with the local bound, this yields the desired global limit on the number of distinct final vectors.

Lemma 1 (Local Bound: Maximum Candidates per Session) *Let i be a session containing t_i honest parties, and let $\vec{v}_i = (\beta_1^{(1)}, \beta_1^{(2)}, \ldots, \beta_i^{(t_i)})$ be the associated session vector. An adversary, that can corrupt up to $c_{\mathsf{max},i} < t_i$ parties, can choose up to $\nu_i \leq \left\lfloor \frac{t_i}{t_i - c_{\mathsf{max},i}} \right\rfloor$ different candidate values in a signing session i without imposing a trivial non-linearity.*

Proof Let ν_i be the number of final candidates. For a candidate value $\beta_i^{(j)}$ with a multiplicity $m_i^{(j)}$ to be a final candidate, the adversary must corrupt all honest parties holding a *different* value. With a budget of $c_{\mathsf{max},i}$, this is possible only if $\beta_i^{(j)}$ appears $m_i^{(j)} \geq t_i - c_{\mathsf{max},i}$ times in $\vec{v}_i$. Hence the ν_i candidate values together occupy at least $\nu_i \cdot (t_i - c_{\mathsf{max},i})$ positions among the t_i honest parties. Since we have at most t_i honest parties, it holds

$$\nu_i \cdot (t_i - c_{\mathsf{max},i}) \leq t_i,$$

and therefore,

$$\nu_i \leq \frac{t_i}{t_i - c_{\mathsf{max},i}}.$$

Since ν_i is an integer, $\nu_i \leq \left\lfloor t_i/(t_i - c_{\mathsf{max},i}) \right\rfloor$, as claimed.

Lemma 2 (Global Bound: Preserving the Number of Candidates) *Let signing session i contain t_i honest parties, and let the adversary have remaining corruption capacity $c_{\mathsf{max},i} < t_i$. Denote by ν_i the current number of candidate values in session i (c.f. Lemma 1), by $\hat{\nu}_i$ the maximal number of feasible candidate values for the given corruption capacity $c_{\mathsf{max},i}$, and by $\hat{\nu}_{i,j}$ the corresponding maximum for the successor session that follows after selecting candidate j. Then*

$$\sum_{j=1}^{\nu_i} \hat{\nu}_{i,j} \ \leq \ \hat{\nu}_i.$$

Proof From Lemma 1, the number of possible candidates in any session with parameters $(t_i, c_{\mathsf{max},i})$ satisfies

$$\nu_i \leq \left\lfloor \frac{t_i}{t_i - c_{\mathsf{max},i}} \right\rfloor. \tag{3}$$

We must assume $c_{\mathsf{max},i} < t_i$, and $t_c < t$. Therefore, ν_i is strictly increasing in $c_{\mathsf{max},i}$ and reaches its maximum if $c_{\mathsf{max},i} = t_i - (t - t_c)$, i.e., if the maximal possible corruption allows to corrupt all but $t - t_c$ out of the t_i honest parties. Substituting this in Equation (3) gives

$$\hat{\nu}_i = \left\lfloor \frac{c_{\mathsf{max},i} + t - t_c}{t - t_c} \right\rfloor \text{ and } \hat{\nu}_{i,j} = \left\lfloor \frac{c_{\mathsf{max},i,j} + t - t_c}{t - t_c} \right\rfloor. \tag{4}$$

We observe that the denominator is fixed for all i; therefore, we focus on the numerator first. For each feasible candidate $\beta_i^{(j)}$, let m_j be its multiplicity among the honest parties. Selecting $\beta_i^{(j)}$ requires corrupting all other honest parties, i.e., $\delta_j = t_i - m_j$, leaving $c_{\mathsf{max},i,j} = c_{\mathsf{max},i} - \delta_j \geq 0$. We first show that the total remaining corruption capacity across all successors never exceeds the current one:

$$\sum_{j=1}^{\nu_i} (c_{\mathsf{max},i,j} + (t - t_c)) = \sum_{j=1}^{\nu_i} (c_{\mathsf{max},i} - \delta_j + (t - t_c)) \tag{5}$$

$$= \nu_i \cdot (c_{\mathsf{max},i} + (t - t_c)) - \sum_{j=1}^{\nu_i} \delta_j. \tag{6}$$

Substituting $\delta_j = t_i - m_j$ and using $\sum_{j=1}^{\nu_i} m_j = t_i$ yields

$$\sum_{j=1}^{\nu_i} \delta_j = \sum_{j=1}^{\nu_i} (t_i - m_j) = \nu_i t_i - \sum_{j=1}^{\nu_i} m_j = \nu_i t_i - t_i = (\nu_i - 1) t_i, \tag{7}$$

and hence we can substitute Equation (7) in Equation (5), obtaining

$$\sum_{j=1}^{\nu_i} c_{\mathsf{max},i,j} + (t - t_c) = \nu_i \cdot (c_{\mathsf{max},i} + (t - t_c)) - (\nu_i - 1) t_i. \tag{8}$$

We now want to substitute t_i, and to do this, we use the observation above stating that $t_i \geq c_{\mathsf{max},i} + (t - t_c)$. We perform this substitution in Equation (8) resulting in

$$\sum_{j=1}^{\nu_i} c_{\mathsf{max},i,j} + (t - t_c) \leq \nu_i (c_{\mathsf{max},i} + (t - t_c)) \tag{9}$$

$$- (\nu_i - 1)(c_{\mathsf{max},i} + (t - t_c)) \tag{10}$$

$$= c_{\mathsf{max},i} + (t - t_c). \tag{11}$$

Applying the formula of the maximum number of candidate values given in Equation (4), using the subadditivity of the floor function, and finally applying the inequality of Equation (9) results in

$$\sum_{j=1}^{\nu_i} \hat{\nu}_{i,j} = \sum_{j=1}^{\nu_i} \left\lfloor \frac{c_{\mathsf{max},i,j} + (t - t_c)}{t - t_c} \right\rfloor \leq \left\lfloor \sum_{j=1}^{\nu_i} \frac{c_{\mathsf{max},i,j} + (t - t_c)}{t - t_c} \right\rfloor$$

$$\leq \left\lfloor \frac{c_{\mathsf{max},i} + (t - t_c)}{t - t_c} \right\rfloor = \hat{\nu}_i,$$

and our lemma holds.

Putting everything together, our claim holds, since $\hat{\nu}_0 \leq t/(t-t_c)$, therefore there also exist at most $t/(t-t_c)$ forgery matrices without imposing a trivial non-linearity.

Theorem 2 *Let* $\mathsf{H}_{\mathrm{sig}}, \mathsf{H}_{\mathrm{non}}$, H_{s}, H_{vk}, *and* H_{IA} *be hash functions modeled as a single random oracle* H_{RO} *using domain separation techniques. Let* $\mathbb{G}$ *be a cyclic group of prime order* p *with generator* g *and let* λ *be the security parameter. Let* CS *be an* ϵ*-equivocable trapdoor commitment scheme and* Π *be an NIZK. For all* $t_c < t < n \in \mathbb{N}$*, and any algebraic PPT adversary* $\mathcal{A}$ *querying* H_{RO} *at most* q_{RO} *times, there exist PPT adversaries* $\mathcal{B}_{\mathsf{ZK}}$ *against the zero-knowledge property of* Π *and* $\mathcal{B}_{\mathsf{adpTSUF}}$ *against the adaptive unforgeability of* $\mathsf{FaFROST}$*, such that*

$$\mathsf{Adv}^{\mathsf{adpTSUF}}_{\mathcal{A},\mathsf{FaFROST}+}(t_c,t,n) \leq \frac{q_{\mathsf{RO}} \cdot n^2}{2^\lambda} + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi,\mathcal{B}_{\mathsf{ZK}}}(\lambda) + \epsilon + \mathsf{Adv}^{\mathsf{adpTSUF}}_{\mathcal{B}_{\mathsf{adpTSUF}},\mathsf{FaFROST}}(t_c,t,n),$$

where the experiment $\mathsf{adpTSUF}$ *is defined in Figure 2 and the probabilities are taken over the random choices of all randomized algorithms.*

Proof We show Theorem 2 using a series of game hops from the experiment $\mathsf{adpTSUF}$ to an experiment, in which the advantage of the adversary is independent of the identifiable abort protocols. In this setting, we show that the advantage of the adversary in winning the $\mathsf{adpTSUF}$ experiment instantiated with $\mathsf{FaFROST}+$ equals the advantage of an adversary $\mathcal{B}_{\mathsf{adpTSUF}}$ against the adaptive unforgeability of $\mathsf{FaFROST}$ (namely, *without* identifiable aborts).
Experiment Exp_0**.** The initial experiment is the $\mathsf{adpTSUF}$ experiment instantiated with $\mathsf{FaFROST}+$ simulated honestly to $\mathcal{A}$.
Experiment Exp_1**.** In Exp_1, we replace honestly computed NIZK proofs with simulated ones.

Claim 2.1 *The adversary's view in* Exp_0 *and* Exp_1 *is computationally indistinguishable if* Π *is zero-knowledge.*

Proof The zero-knowledge property of the proof system ensures that the view remains computationally indistinguishable when replacing honestly computed proofs with simulated ones.

Experiment Exp_2**.** In Exp_2, we abort, if the adversary queries the oracle H_{IA} on input a key $k_{i,j}$ with $i,j \in \textsc{hon}$.

Claim 2.2 *The probability that* Exp_2 *aborts is bounded by* $q_{\mathsf{RO}} \cdot n^2/2^\lambda$.

Proof The key $k_{i,j}$ is sampled uniformly at random from the space $\{0,1\}^\lambda$. Guessing it correctly has probability $1/2^\lambda$. Since all hash functions are modeled as random oracles, the only way to guess $k_{i,j}$ correctly is by querying the random oracle. The adversary makes at most q_{RO} queries to the random oracle, and there exist at most n^2 unknown keys. The claim follows from union bound.

Experiment Exp_3**.** In Exp_3, we replace the way the challenger simulates queries to OIA_1. Instead of honestly computing commitments in Line 5 (Figure 6), the challenger answers with consistent trapdoor commitments. In more detail, the challenger replaces ck by $(\mathsf{ck},\mathsf{td}) \leftarrow \mathsf{tdKeyGen}(1^\lambda)$, and when a session sid starts, it pre-computes and stores for all honest signers $i \neq j \in \textsc{hon}$ the commitment $\mathsf{C}_{i,j} \leftarrow \mathsf{tdComm}(\mathsf{td})$. These commitments are used when OIA_1 is called, and both signer i and j are still honest. To make this consistent with corruptions, the challenger additionally does the following: When a signer is corrupted, it calls tdOpen on the message $B^s_{i,j}$ to receive the opening $\omega_{i,j}$, and programs $\omega_{i,j} \to \mathsf{H}_{\mathsf{IA}}(k_{i,j}, \{(\mathsf{pm}_j, s_j)\}_{j \in S})$.

Claim 2.3 *The adversary's view in* Exp_2 *and* Exp_3 *are* ϵ*-statistically close.*

Proof We show this claim based on the trapdoor property of CS. First, we observe that ck and $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{tdKeyGen}(1^\lambda)$, and computing $\mathsf{C}_{i,j} \leftarrow \mathsf{tdComm}(\mathsf{td})$ instead of honest computing has statistical distance of at most ϵ. Additionally, for two honest signers, the commitments are equal by our precompute-and-store method. Furthermore, Exp_2 ensured that $\mathcal{A}$ never queries $k_{i,j}$ for honest i and j to H_{IA}. Therefore, the programming provides no gap in the adversary's view. Trapdoor opening is furthermore possible, since the challenger knows the honest messages $B^s_{i,j}$, as it knows the key $k_{i,j}$. The claim follows.

We can now show that the remaining advantage of an adversary winning Exp_3 is bounded by the advantage of winning $\mathsf{adpTSUF}$. We do this using a reduction $\mathcal{B}_{\mathsf{adpTSUF}}$. As input, $\mathcal{B}_{\mathsf{adpTSUF}}$ receives a verification key vk, and partial verification keys vk_i. Our reduction samples values $K_{i,j} \leftarrow\!\!\$\ \mathbb{Z}_p$ for all $i \neq j \in [n]$ and sets $vk'_i \leftarrow (vk_i, \{K_{i,j}\}_{j \neq i \in [n]})$. It forwards $(vk, \{vk'_i\}_{i \in [n]})$ to $\mathcal{A}$. Furthermore, our reduction has access to a corruption oracle and two sign-round oracles. The reduction simulates the sign round oracles to $\mathcal{A}$ by simply forwarding messages. When the adversary queries the OIA_1 oracle, the reduction proceeds as in Exp_3. When the adversary queries the OIA_2 oracle, the reduction also proceeds as in Exp_3. Notice that the reduction never has to hand out a key $k_{i,j}$ in which either i or j is honest, since we utilize the $\mathsf{CheckView}$ function, and in Exp_3 the pairwise commitments of two honest signers are always equal.

When the adversary queries the corrupt oracle, the reduction forwards this request and receives (sk_i, st_i). It then recomputes all previously output blinding values via $B^s_{i,j} = \mathsf{H}_{\mathrm{s}}(k_{i,j}, \{\mathsf{pm}_j\}_{j \in S}, m, S)$. Notice that this is possible since $k_{i,j}$ is part of sk_i. In addition, it programs $\mathsf{H}_{vk}(k_{i,j}) \leftarrow K_{i,j}$. This programming can be done because, in the situation of Exp_3, the adversary never queries any random oracle on input $k_{i,j}$ (since it is not yet corrupted). Knowing the values $B^s_{i,j}$, the reduction proceeds as in Exp_3 to program the random oracle H_{IA}. Finally, it forwards (sk_i, st_i) to the adversary. Eventually, the adversary outputs a forgery which our reduction also outputs.

Our reduction simulates the experiment $\mathsf{adpTSUF}$ perfectly to the adversary, as in Exp_3, the additional identifiable abort oracles are simulated without knowledge of any honest secret key, and making the oracle outputs consistent after corruption is also guaranteed. In addition, our reduction is efficient since it simulates the efficient experiment Exp_3 to an efficient adversary. Therefore, Theorem 2 holds.

Theorem 3 *Let* $\mathsf{H}_{\mathrm{sig}}, \mathsf{H}_{\mathrm{non}}, \mathsf{H}_{\mathrm{s}}, \mathsf{H}_{vk}$, *and* H_{IA} *be hash functions modeled as a single random oracle* H_{RO} *using domain separation techniques. Let* $\mathbb{G}$ *be a cyclic group of prime order* p *with generator* g *and let* λ *be the security parameter. Let* CS *be a commitment scheme and* Π *be an NIZK. For all* $t_c < t \leq n \in \mathbb{N}$ *and any algebraic PPT adversary* $\mathcal{A}$*, there exist PPT adversaries* $\mathcal{B}_{\mathsf{bind}}$ *against the binding property of* CS*,* $\mathcal{B}_{\mathsf{ZK}}$ *against the zero-knowledge property of* Π*, and* $\mathcal{B}_{\mathsf{sfext}}$ *against the simulation extractability of* Π*, such that*

$$\mathsf{Adv}^{\mathsf{adp-IA}}_{\mathcal{A},\mathsf{FaFROST}+}(t_c, t, n) \leq \mathsf{Adv}^{\mathsf{bind}}_{\mathsf{CS},\mathcal{B}_{\mathsf{bind}}}(\lambda) + \mathsf{Adv}^{\mathsf{ZK}}_{\Pi,\mathcal{B}_{\mathsf{ZK}}}(\lambda) + t_c \cdot \mathsf{Adv}^{\mathsf{sfext}}_{\Pi,\mathcal{B}_{\mathsf{sfext}}}(\lambda).$$

Where the probability is taken over the randomness of all randomized algorithms and the experiment $\mathsf{adp-IA}$ is defined in Figure 3.

Proof To win the identifiable abort experiment, the adversary can have two strategies. Either the set $\mathcal{C}$ is empty, or an honest signer was identified as behaving maliciously. We show that either case is negligible.

We start with the case where an honest signer is marked as malicious. A party can be identified as malicious in four situations. We show that an honest signer cannot be identified in any of these cases, even with the correctness of the underlying primitives.

1. When the proof of correct blinding does not verify (Line 20, Figure 6). By perfect completeness of the NIZK scheme, the proof of an honest party will never fail, since all parties got consistent inputs, and the values $B^s_{i,j}, \omega_{i,j}, \mathsf{C}_{i,j}, s_i$ are all well-formed.
2. When a signer reveals a key $k_{j,\ell}$, but this key does not match the partial public key $K_{j,\ell}$ (Line 23, Figure 6). An honest signer reveals the key received during key generation that is well-formed since the random oracle is modeled deterministically.
3. A signer provided the correct key $k_{j,\ell}$, but the commitment $\mathsf{C}_{\ell,j}$ does not match the recomputation (Line 28, Figure 6). All commitments computed by honest signers are well-formed and all signers share a consistent view. Hence, the recomputation yields the same commitment $\mathsf{C}_{\ell,j}$.
4. A signer did not accuse another signer, even if the commitments do not match (Lines 31-32, Figure 6). An honest signer accuses other signers if their commitment does not match (Line 14, Figure 6).

Therefore, no honest signer is ever identified as behaving maliciously, as all primitives used in FaFROST+ have perfect correctness.

We now focus on the second case, in which the set $\mathcal{C}$ is empty, i.e., no party is identified as behaving maliciously. Our first observation in this case is that if there are any two commitments $\mathsf{C}_{i,j} \neq \mathsf{C}_{j,i}$, then at least one party is identified as malicious. This is true, since in Lines 31- 32, it is ensured that in this case, both parties i and j would provide their respective key $k_{i,j}$. In addition, by checking well-formedness of this key in Line 23, it is ensured that either $\mathsf{C}_{i,j}$ or $\mathsf{C}_{j,i}$ is correct. Therefore, if these commitments are unequal, at least one of them is malformed, and the party that computed this commitment is marked as malicious in Line 28. Similarly, the proofs of all signers verify in Line 20, otherwise such a signer would be marked as malicious. In addition, since combining the signatures failed, at least two values $B^s_{i,j} \neq B^s_{j,i}$. This is the case, since the Schnorr verification algorithm checks $vk^c \cdot R = g^s$, and it follows that

$$\sum_{j \in S^*} s_j \neq sk \cdot c + \sum_{j \in S^*} d_j + b \cdot \sum_{j \in S^*} e_j.$$

By construction, the sum of all partial signatures generated by honest signers equals

$$\sum_{j \in S^* \cap \text{HON}} s_j = \sum_{j \in S^* \cap \text{HON}} (sk_j c + d_j + b e_j) + \sum_{i \in S^* \cap \text{HON}, j \in S^* \cap \text{CORR}} B^s_{i,j} \cdot \Delta_{i,j}.$$

Furthermore, we observe that if all the corrupted active signers generate their nonces honestly, then by correctness of FaFROST, verification is trivially satisfied; therefore, there must exist at least one $i \in S^* \cap \text{CORR}$ such that for some $j \in S^* \setminus \{i\}$, $B^s_{i,j}$ was not honestly generated, i.e.

$$B^s_{i,j} \neq \mathsf{H}_{\text{s}}(k_{i,j}, \{\mathsf{pm}_j\}_{j \in S}, m, S).$$

We show that the event $\mathcal{C} = \emptyset$ while the combined signature does not verify occurs only with negligible probability using a series of game hops.

Experiment Exp_0. The initial experiment equals the $\mathsf{adp} - \mathsf{IA}_{\mathcal{A}}$ experiment in the case that $\mathcal{C} = \emptyset$.

Experiment Exp_1. In Exp_1, we replace honestly computed proofs with simulated proofs.

Claim 3.1 *The adversary's view in Exp_0 and Exp_1 is computationally indistinguishable if Π is zero-knowledge.*

Proof The zero-knowledge property of the proof system ensures that the view remains computationally indistinguishable when replacing honestly computed proofs with simulated ones.

Experiment Exp_2. In Exp_2, we use the extractor $\mathcal{E}$, whose existence is guaranteed by the simulation extractability of Π, to extract all adversarially used witnesses from the provided proofs of well-formedness and abort if extraction fails. These witnesses have the form $(sk'_i, \{(B^s_{i,j}{}', \omega'_{i,j})\}_{j \neq i \in S^*})$.

Claim 3.2 *The difference in the adversary's view in* Exp_1 *and* Exp_2 *is bounded by* $t_c \cdot \mathsf{Adv}^{\mathsf{sfext}}_{\Pi, \mathcal{B}_{\mathsf{sfext}}}(\lambda)$.

Proof The simulation extractability of Π guarantees that, for each adversarially provided proof, a valid witness can be extracted even in presence of simulated proofs, unless with negligible probability. Since $\mathcal{A}$ is an algebraic adversary, this extraction is also straight-line without rewinding. Formally, it is possible to build a reduction $\mathcal{B}_{\mathsf{sfext}}$ that breaks the simulation extractability of Π whenever an abort occurs in Exp_2 but not in Exp_1, thus upper-bounding the probability of the distinguishing event to the advantage of $\mathcal{B}_{\mathsf{sfext}}$ in the simulation extractability game for Π. Notice that we never invoke the extractor $\mathcal{E}$ on input a pair statement-proof that is simulated: this is because in Exp_2 we only feed the extractor with proofs crafted by the adversary (and thus are not output by the simulation oracle). Moreover, we observe that the statement must be *fresh*, namely one for which he has never seen a simulated proof: this is because the statament is bound to the index of the party, the message and the signer set (cf. Line 6 in Figure 6). The adversary extracts from all corrupted parties once, so we have the factor t_c. The proof holds.

In the situation of Exp_2, our challenger compares the extracted witnesses with their honest counterparts. This is possible, since the challenger knows all honestly computed keys, and the view of each signer is consistent in Exp_2. We now show that winning with $\mathcal{C} = \emptyset$ implies breaking the binding property of the commitment scheme CS. We show this by constructing a reduction $\mathcal{B}_{\mathsf{bind}}$ against the binding property of CS. Our reduction simulates Exp_2 to the adversary. As input, our reduction receives a commitment key ck and uses this in the setup protocol. The rest of Exp_2 is simulated honestly to the adversary.

When the challenger recovers the adversarially used commitment openings $\{(B^s_{i,j}{}', \omega'_{i,j})\}_{j \neq i \in S^*}$, it selects one pair $B^s_{i,j}{}', \omega'_{i,j}$, for which $B^s_{i,j}{}' \neq B^s_{i,j}$ and $\omega'_{i,j} \neq \omega_{i,j}$. The reduction outputs $(B^s_{i,j}{}', \omega'_{i,j})$, $(B^s_{i,j}, \omega_{i,j})$. When committing to each of these pairs, they result in the same commitment, as we are in the situation of Exp_2. In addition, they do not equal. Hence, our reduction breaks the binding property. Our reduction is efficient, as Exp_2 is efficient. It simulates the view of the adversary perfectly, as ck is identically distributed in this reduction and Exp_2. This shows that the probability that $\mathcal{C} = \emptyset$ is bounded by $\mathsf{Adv}^{\mathsf{bind}}_{\mathsf{CS}, \mathcal{B}_{\mathsf{bind}}}(\lambda)$.

7 Prototype

We implemented a prototype of FaFROST+ in Rust [Ger26]. Our implementation includes dealer-based key generation, two-round signing with pairwise blinding, signature aggregation and verification, as well as the identifiable-abort protocol. The prototype runs on two ciphersuites, which share the same protocol code. First, edwards25519 with SHA-512 via `curve25519-dalek` produces EdDSA signatures [JL17], as standardized in FIPS 186-5 [Nat23]. Second, secp256k1 with BIP-340 tagged hashes via `k256` targets the Bitcoin Schnorr variant, so its aggregate signatures work as Taproot key spends [WNR20]. In both cases, the verifier of the corresponding single-signer scheme

accepts our aggregate signatures (for edwards25519, the independent `ed25519-dalek` implementation on its strict path).

Performance. The only additional computation introduced by FaFROST+ compared to FROST2 is the derivation of the blinding factors. FaFROST+ leaves the communication pattern unchanged, and signing still takes two rounds and has the same message complexity. Each signer derives $t-1$ blinding factors locally from the first-round commitments, requiring $t-1$ hash-to-scalar evaluations. We do not benchmark FROST2 separately, since its signing cost corresponds to the *sign* column excluding the *blind* column. Blinding accounts for less than 5% of the signing time on edwards25519 and less than 1.6% on secp256k1, across all evaluated configurations. That share grows with the signer set, however, since blinding grows faster in t than signing does. In absolute terms, blinding takes between 0.3 and 16 µs, while signing takes between 79 and 391 µs. Commitment generation and signing run faster on edwards25519, since `curve25519-dalek` multiplies the base point against a precomputed table (an option `k256` does not provide), so the absolute runtimes of operations dominated by base-point multiplication are not directly comparable across the two curves.

We also benchmark the three phases of the identifiable-abort protocol, which are executed only in the pessimistic path after a failed signing session. IA1 scales linearly with the number of signers, ranging from 595 µs for $t = 4$ to 9.43 ms for $t = 64$ on edwards25519. The *decide* phase is more expensive, since it performs pairwise consistency checks using the opened pairwise keys and therefore grows quadratically (521.8 ms on edwards25519 and 554.6 ms on secp256k1 for $t = 64$). IA2 remains below 9 µs in all measurements, so IA1 and *decide* dominate the cost of identification. We report all runtimes in Table 3.

(t, n)	**Commit**	**Sign**	**Blind**	**Agg**	**IA1**	**IA2**	**Decide**
edwards25519, SHA-512							
$(4, 256)$	17 µs	79 µs	810 ns	374 ns	595 µs	354 ns	2.81 ms
$(8, 256)$	17 µs	105 µs	2.33 µs	887 ns	1.25 ms	794 ns	9.32 ms
$(16, 256)$	17 µs	145 µs	4.92 µs	1.75 µs	2.33 ms	1.93 µs	35.5 ms
$(32, 256)$	17 µs	229 µs	7.81 µs	3.66 µs	4.96 ms	4.33 µs	131.0 ms
$(48, 256)$	17 µs	313 µs	15 µs	5.53 µs	7.07 ms	6.82 µs	296.9 ms
$(64, 256)$	17 µs	391 µs	16 µs	7.20 µs	9.43 ms	8.77 µs	521.8 ms
secp256k1, BIP-340							
$(4, 256)$	58 µs	112 µs	305 ns	165 ns	687 µs	197 ns	2.90 ms
$(8, 256)$	58 µs	132 µs	728 ns	381 ns	1.35 ms	433 ns	10.1 ms
$(16, 256)$	58 µs	163 µs	1.50 µs	807 ns	2.81 ms	988 ns	37.0 ms
$(32, 256)$	58 µs	234 µs	2.91 µs	1.65 µs	5.57 ms	2.48 µs	144.6 ms
$(48, 256)$	58 µs	307 µs	4.50 µs	2.52 µs	8.44 ms	3.54 µs	320.1 ms
$(64, 256)$	58 µs	378 µs	5.75 µs	3.38 µs	11.0 ms	5.16 µs	554.6 ms

Table 3: Median runtimes per signer for both ciphersuites, measured with Criterion on an Apple M3 Pro with 36 GB RAM. Signing differs from FROST2 only in the blinding, so the signing cost of FROST2 is the *sign* column minus the *blind* column. The identifiable-abort phases (IA, IA2, Decide) are executed only if aggregation fails.

On-chain validation. To validate the BIP-340 ciphersuite against the live Bitcoin network, we generated a two-of-three FaFROST key, derived the corresponding Taproot (P2TR) address, and spent from it using a key-spend transaction signed by two participants. The transaction was confirmed on the Bitcoin testnet on April 26, 2026 in block 4,947,515 and has a weight of 134 vB [Faf]. Its input passes standard BIP-340 verification, and the P2TR output is indistinguishable from a regular single-signer spend. Existing Bitcoin verification infrastructure therefore accepts FaFROST+ signatures without changes to on-chain validation.

Acknowledgements

We thank Phillip Nazarian and Stanislaw Jarecki for helpful discussions about the adaptively secure identifiable abort mechanism at an early stage of the project. We thank Kaoru Takemure and Nadav Kohen for their helpful comments on an earlier version of our proof. They pointed out that our analysis omitted a step needed to fully account for ROS attacks in our simulation and that our probability argument did not capture the adversary's multiple options for transforming the challenge matrix into a forgery matrix. We addressed these issues, which increases the reduction's failure probability by a factor of $\frac{t}{t-t_c} \leq t$. We thank the authors of [BCL+26] for independently identifying the latter observation and bringing it to our attention. We also thank Omkant Pandey for his helpful advice and suggestions regarding this work. This work is also partially supported by SBA Research (SBA-K1 NGC), which is a COMET Center within the COMET (Competence Centers for Excellent Technologies) Programme and funded by BMIMI, BMWET, and the federal state of Vienna. The COMET Programme is managed by FFG. This work is further partially supported by the Vienna Science and Technology Fund (WWTF) under grant ICT25-081. Paul Gerhart's research has been supported by the Google PhD Fellowship in Privacy, Safety, and Security. Arkady Yerukhimovich is supported in part by NSF grants CNS-1955620 and CNS-2144798 (CAREER).

References

[ANO+22] Damiano Abram, Ariel Nof, Claudio Orlandi, Peter Scholl, and Omer Shlomovits. "Low-Bandwidth Threshold ECDSA via Pseudorandom Correlation Generators". In: *2022 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2022, pp. 2554–2572. DOI: 10.1109/SP46214.2022.9833559.

[BCC+25] Christian Badertscher, Matteo Campanelli, Michele Ciampi, Luigi Russo, and Luisa Siniscalchi. "Universally Composable SNARKs with Transparent Setup without Programmable Random Oracle". In: *CRYPTO 2025, Part VII*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16006. LNCS. Springer, Cham, Aug. 2025, pp. 225–258. DOI: 10.1007/978-3-032-01907-3_8.

[BCK+22] Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. "Better than Advertised Security for Non-interactive Threshold Signatures". In: *CRYPTO 2022, Part IV*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13510. LNCS. Springer, Cham, Aug. 2022, pp. 517–550. DOI: 10.1007/978-3-031-15985-5_18.

[BCL+25] Renas Bacho, Yanbo Chen, Julian Loss, Stefano Tessaro, and Chenzhi Zhu. *Adaptively Secure Partially Non-Interactive Threshold Schnorr Signatures in the AGM*. Cryptology ePrint Archive, Report 2025/1953. 2025. URL: https://eprint.iacr.org/2025/1953.

[BCL+26] Renas Bacho, Yanbo Chen, Julian Loss, Stefano Tessaro, and Chenzhi Zhu. "Adaptively Secure Partially Non-interactive Threshold Schnorr Signatures in the AGM". In: *EUROCRYPT 2026, Part I*. Ed. by Joan Daemen and Emmanuel Thomé. Vol. 16541. LNCS. Springer, Cham, May 2026, pp. 33–62. DOI: 10.1007/978-3-032-25291-3_2.

[BD22] Luís T. A. N. Brandão and Michael Davidson. *Notes on Threshold EdDSA/Schnorr Signatures*. Tech. rep. NIST IR 8214B (Draft). Initial Public Draft. National Institute of Standards and Technology (NIST), Aug. 2022. DOI: 10.6028/NIST.IR.8214B.ipd. URL: https://doi.org/10.6028/NIST.IR.8214B.ipd.

[BDL+25a] Renas Bacho, Sourav Das, Julian Loss, and Ling Ren. "Adaptively Secure Three-Round Threshold Schnorr Signatures from DDH". In: *CRYPTO 2025, Part VI*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16005. LNCS. Springer, Cham, Aug. 2025, pp. 390–422. DOI: 10.1007/978-3-032-01887-8_13.

[BDL+25b] Renas Bacho, Sourav Das, Julian Loss, and Ling Ren. "Glacius: Threshold Schnorr Signatures from DDH with Full Adaptive Security". In: *EUROCRYPT 2025, Part II*. Ed. by Serge Fehr and Pierre-Alain Fouque. Vol. 15602. LNCS. Springer, Cham, May 2025, pp. 304–334. DOI: 10.1007/978-3-031-91124-8_11.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. "Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)". In: *20th ACM STOC*. ACM Press, May 1988, pp. 103–112. DOI: 10.1145/62212.62222.

[BGC+25] Ruben Baecker, Paul Gerhart, Davide Li Calsi, Luigi Russo, Dominique Schröder, and Arkady Yerukhimovich. *Fully Adaptive FROST in the Algebraic Group Model From Falsifiable Assumptions*. Cryptology ePrint Archive, Paper 2025/1950. https://eprint.iacr.org/2025/1950. 2025.

[BN06] Mihir Bellare and Gregory Neven. "Multi-signatures in the plain public-Key model and a general forking lemma". In: *ACM CCS 2006*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. ACM Press, 2006, pp. 390–399. DOI: 10.1145/1180405.1180453.

[Bol03] Alexandra Boldyreva. "Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme". In: *PKC 2003*. Ed. by Yvo Desmedt. Vol. 2567. LNCS. Springer, Berlin, Heidelberg, Jan. 2003, pp. 31–46. DOI: 10.1007/3-540-36288-6_3.

[BSP+08] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. May 2008. DOI: 10.17487/RFC5280. URL: https://www.rfc-editor.org/info/rfc5280.

[CCK+25] Michele Ciampi, Elizabeth Crites, Chelsea Komlo, and Mary Maller. *On the Adaptive Security of Key-Unique Threshold Signatures*. Cryptology ePrint Archive, Paper 2025/943. 2025. URL: https://eprint.iacr.org/2025/943.

[CCL+20] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. "Bandwidth-Efficient Threshold EC-DSA". In: *PKC 2020, Part II*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Vol. 12111. LNCS. Springer, Cham, May 2020, pp. 266–296. DOI: 10.1007/978-3-030-45388-6_10.

[CGG+20] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. "UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts". In: *ACM CCS 2020*. Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna. ACM Press, Nov. 2020, pp. 1769–1787. DOI: 10.1145/3372297.3423367.

[CGR+23] Hien Chu, Paul Gerhart, Tim Ruffing, and Dominique Schröder. "Practical Schnorr Threshold Signatures Without the Algebraic Group Model". In: *CRYPTO 2023, Part I*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14081. LNCS. Springer, Cham, Aug. 2023, pp. 743–773. DOI: 10.1007/978-3-031-38557-5_24.

[Che25] Yanbo Chen. "Round-Efficient Adaptively Secure Threshold Signatures with Rewinding". In: *CiC* 2.2 (2025), p. 16. DOI: 10.62056/ah893z10k.

[CKG+24] Deirdre Connolly, Chelsea Komlo, Ian Goldberg, and Christopher A. Wood. *The Flexible Round-Optimized Schnorr Threshold (FROST) Protocol for Two-Round Schnorr Signatures*. RFC 9591. June 2024. DOI: 10.17487/RFC9591. URL: https://www.rfc-editor.org/info/rfc9591.

[CKK+25] Elizabeth C. Crites, Jonathan Katz, Chelsea Komlo, Stefano Tessaro, and Chenzhi Zhu. "On the Adaptive Security of FROST". In: *CRYPTO 2025, Part VI*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16005. LNCS. Springer, Cham, Aug. 2025, pp. 480–511. DOI: 10.1007/978-3-032-01887-8_16.

[CKM21] Elizabeth Crites, Chelsea Komlo, and Mary Maller. *How to Prove Schnorr Assuming Schnorr: Security of Multi- and Threshold Signatures*. Cryptology ePrint Archive, Report 2021/1375. 2021. URL: https://eprint.iacr.org/2021/1375.

[CKM23] Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. "Fully Adaptive Schnorr Threshold Signatures". In: *CRYPTO 2023, Part I*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14081. LNCS. Springer, Cham, Aug. 2023, pp. 678–709. DOI: 10.1007/978-3-031-38557-5_22.

[CP93] David Chaum and Torben P. Pedersen. "Wallet Databases with Observers". In: *CRYPTO'92*. Ed. by Ernest F. Brickell. Vol. 740. LNCS. Springer, Berlin, Heidelberg, Aug. 1993, pp. 89–105. DOI: 10.1007/3-540-48071-4_7.

[CS25] Elizabeth C. Crites and Alistair Stewart. "A Plausible Attack on the Adaptive Security of Threshold Schnorr Signatures". In: *CRYPTO 2025, Part VI*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16005. LNCS. Springer, Cham, Aug. 2025, pp. 457–479. DOI: 10.1007/978-3-032-01887-8_15.

[DDF+94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. "How to Share a Function Securely". In: *26th ACM STOC*. ACM Press, May 1994, pp. 522–533. DOI: 10.1145/195058.195405.

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. "Robust Non-interactive Zero Knowledge". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Berlin, Heidelberg, Aug. 2001, pp. 566–598. DOI: 10.1007/3-540-44647-8_33.

[Des88] Yvo Desmedt. "Society and Group Oriented Cryptography: A New Concept". In: *CRYPTO'87*. Ed. by Carl Pomerance. Vol. 293. LNCS. Springer, Berlin, Heidelberg, Aug. 1988, pp. 120–127. DOI: 10.1007/3-540-48184-2_8.

[DF90] Yvo Desmedt and Yair Frankel. "Threshold Cryptosystems". In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, New York, Aug. 1990, pp. 307–315. DOI: 10.1007/0-387-34805-0_28.

[DJN+20] Ivan Damgård, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Jakob Illeborg Pagter, and Michael Bæksvang Østergaard. "Fast Threshold ECDSA with Honest Majority". In: *SCN 20*. Ed. by Clemente Galdi and Vladimir Kolesnikov. Vol. 12238. LNCS. Springer, Cham, Sept. 2020, pp. 382–400. DOI: 10.1007/978-3-030-57990-6_19.

[DKL+19] Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. "Threshold ECDSA from ECDSA Assumptions: The Multiparty Case". In: *2019 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2019, pp. 1051–1066. DOI: 10.1109/SP.2019.00024.

[DOK+20] Anders P. K. Dalskov, Claudio Orlandi, Marcel Keller, Kris Shrishak, and Haya Shulman. "Securing DNSSEC Keys via Threshold ECDSA from Generic MPC". In: *ESORICS 2020, Part II*. Ed. by Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider. Vol. 12309. LNCS. Springer, Cham, Sept. 2020, pp. 654–673. DOI: 10.1007/978-3-030-59013-0_32.

[Eth] Ethereum Foundation. *Distributed validator technology*. Accessed: 2026-04-25. URL: https://ethereum.org/staking/dvt/.

[Faf] *FaFROST Demo Transaction*. Bitcoin Testnet, block 4 947 515. https://mempool.space/testnet/tx/3b2646d16ee0bd32d765844cac9de7bb5c75c26e4928fc9cd32ab768c2f84a58. Accessed: April 26, 2026. 2026.

[FFR24] Antonio Faonio, Dario Fiore, and Luigi Russo. "Real-World Universal zkSNARKs are Non-Malleable". In: *ACM CCS 2024*. Ed. by Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie. ACM Press, Oct. 2024, pp. 3138–3151. DOI: 10.1145/3658644.3690351.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. "The Algebraic Group Model and its Applications". In: *CRYPTO 2018, Part II*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10992. LNCS. Springer, Cham, Aug. 2018, pp. 33–62. DOI: 10.1007/978-3-319-96881-0_2.

[FKM+12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. "On the Non-malleability of the Fiat-Shamir Transform". In: *INDOCRYPT 2012*. Ed. by Steven D. Galbraith and Mridul Nandi. Vol. 7668. LNCS. Springer, Berlin, Heidelberg, Dec. 2012, pp. 60–79. DOI: 10.1007/978-3-642-34931-7_5.

[GCR+25a] Paul Gerhart, Davide Li Calsi, Luigi Russo, and Dominique Schröder. *Bounded-Equivocable Pseudorandom Functions*. Cryptology ePrint Archive, Paper 2025/1894. 2025. URL: https://eprint.iacr.org/2025/1894.

[GCR+25b] Paul Gerhart, Davide Li Calsi, Luigi Russo, and Dominique Schröder. *Fully-Adaptive Two-Round Threshold Schnorr Signatures from DDH*. Cryptology ePrint Archive, Paper 2025/1478. 2025. URL: https://eprint.iacr.org/2025/1478.

[Ger26] Paul Gerhart. *FaFROST – Fully Adaptive FROST with Identifiable Aborts from AOMDL*. https://github.com/pgerhart/fafrost. 2026.

[GG18] Rosario Gennaro and Steven Goldfeder. "Fast Multiparty Threshold ECDSA with Fast Trustless Setup". In: *ACM CCS 2018*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM Press, Oct. 2018, pp. 1179–1194. DOI: 10.1145/3243734.3243859.

[GGN16] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. "Threshold-Optimal DSA/ECDSA Signatures and an Application to Bitcoin Wallet Security". In: *ACNS 2016*. Ed. by Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider. Vol. 9696. LNCS. Springer, Cham, June 2016, pp. 156–174. DOI: 10.1007/978-3-319-39555-5_9.

[GJK+03] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. "Secure Applications of Pedersen's Distributed Key Generation Protocol". In: *CT-RSA 2003*. Ed. by Marc Joye. Vol. 2612. LNCS. Springer, Berlin, Heidelberg, Apr. 2003, pp. 373–390. DOI: 10.1007/3-540-36563-X_26.

[GJK+07] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems". In: *Journal of Cryptology* 20.1 (Jan. 2007), pp. 51–83. DOI: 10.1007/s00145-006-0347-3.

[GJK+96] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. "Robust Threshold DSS Signatures". In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Berlin, Heidelberg, May 1996, pp. 354–371. DOI: 10.1007/3-540-68339-9_31.

[GKS+20] Adam Gągol, Jędrzej Kula, Damian Straszak, and Michał Świętek. *Threshold ECDSA for Decentralized Asset Custody*. Cryptology ePrint Archive, Report 2020/498. 2020. URL: https://eprint.iacr.org/2020/498.

[GKS24] Kamil Doruk Gür, Jonathan Katz, and Tjerand Silde. "Two-Round Threshold Lattice-Based Signatures from Threshold Homomorphic Encryption". In: *PQCrypto 2024, Part II*. Ed. by Markku-Juhani Saarinen and Daniel Smith-Tone. Springer, Cham, June 2024, pp. 266–300. DOI: 10.1007/978-3-031-62746-0_12.

[GM17] Jens Groth and Mary Maller. "Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs". In: *CRYPTO 2017, Part II*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. LNCS. Springer, Cham, Aug. 2017, pp. 581–612. DOI: 10.1007/978-3-319-63715-0_20.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)". In: *17th ACM STOC*. ACM Press, May 1985, pp. 291–304. DOI: 10.1145/22145.22178.

[GRJ+00] Rosario Gennaro, Tal Rabin, Stanislaw Jarecki, and Hugo Krawczyk. "Robust and Efficient Sharing of RSA Functions". In: *Journal of Cryptology* 13.2 (Mar. 2000), pp. 273–300. DOI: 10.1007/s001459910011.

[GW11] Craig Gentry and Daniel Wichs. "Separating succinct non-interactive arguments from all falsifiable assumptions". In: *43rd ACM STOC*. Ed. by Lance Fortnow and Salil P. Vadhan. ACM Press, June 2011, pp. 99–108. DOI: 10.1145/1993636.1993651.

[JL17] S. Josefsson and I. Liusvaara. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032. Jan. 2017. DOI: 10.17487/RFC8032. URL: https://www.rfc-editor.org/info/rfc8032/.

[JP14] Abhishek Jain and Omkant Pandey. "Non-Malleable Zero Knowledge: Black-Box Constructions and Definitional Relationships". In: *SCN 14*. Ed. by Michel Abdalla and Roberto De Prisco. Vol. 8642. LNCS. Springer, Cham, Sept. 2014, pp. 435–454. DOI: 10.1007/978-3-319-10879-7_25.

[Kav] Darya Kaviani. *Fireblocks MPC Wallet*. Accessed: 2026-04-25. URL: https://mpc.cs.berkeley.edu/posts/Fireblocks-MPC-Wallet/.

[KG20] Chelsea Komlo and Ian Goldberg. "FROST: Flexible Round-Optimized Schnorr Threshold Signatures". In: *SAC 2020*. Ed. by Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn. Vol. 12804. LNCS. Springer, Cham, Oct. 2020, pp. 34–65. DOI: 10.1007/978-3-030-81652-0_2.

[KRT24] Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. "Adaptively Secure 5 Round Threshold Signatures from MLWE/MSIS and DL with Rewinding". In: *CRYPTO 2024, Part VII*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14926. LNCS. Springer, Cham, Aug. 2024, pp. 459–491. DOI: 10.1007/978-3-031-68394-7_15.

[Lea] League of Entropy. *Docs: Cryptography*. Accessed: 2026-04-25. URL: https://docs.drand.love/docs/cryptography/.

[Lin] Yehuda Lindell. *Digital Asset Management with MPC (Whitepaper)*. Accessed: 2026-04-25. URL: https://www.coinbase.com/en-gb/blog/digital-asset-management-with-mpc-whitepaper.

[LN18] Yehuda Lindell and Ariel Nof. "Fast Secure Multiparty ECDSA with Practical Distributed Key Generation and Applications to Cryptocurrency Custody". In: *ACM CCS 2018*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM Press, Oct. 2018, pp. 1837–1854. DOI: 10.1145/3243734.3243788.

[Mau05] Ueli M. Maurer. "Abstract Models of Computation in Cryptography (Invited Paper)". In: *10th IMA International Conference on Cryptography and Coding*. Ed. by Nigel P. Smart. Vol. 3796. LNCS. Springer, Berlin, Heidelberg, Dec. 2005, pp. 1–12. DOI: 10.1007/11586821_1.

[Nat23] National Institute of Standards and Technology. *Digital Signature Standard (DSS)*. (U.S. Department of Commerce) Federal Information Processing Standards (FIPS) 186-5. 2023. DOI: 10.6028/NIST.FIPS.186-5.

[Nat26] National Institute of Standards and Technology. *First Call for Multi-Party Threshold Schemes*. Tech. rep. 8214C. National Institute of Standards and Technology, 2026. DOI: 10.6028/NIST.IR.8214C. URL: https://csrc.nist.gov/pubs/ir/8214/c/final.

[NRS21] Jonas Nick, Tim Ruffing, and Yannick Seurin. "MuSig2: Simple Two-Round Schnorr Multi-signatures". In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Cham, Aug. 2021, pp. 189–221. DOI: 10.1007/978-3-030-84242-0_8.

[NRT26] Guilhem Niot, Michael Reichle, and Kaoru Takemure. "Adaptively-Secure Three-Round Threshold Schnorr from DL". In: *EUROCRYPT 2026, Part I*. Ed. by Joan Daemen and Emmanuel Thomé. Vol. 16541. LNCS. Springer, Cham, May 2026, pp. 3–32. DOI: 10.1007/978-3-032-25291-3_1.

[Ped92] Torben P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". In: *CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. LNCS. Springer, Berlin, Heidelberg, Aug. 1992, pp. 129–140. DOI: 10.1007/3-540-46766-1_9.

[PKN+25] Rafaël Del Pino, Shuichi Katsumata, Guilhem Niot, Michael Reichle, and Kaoru Takemure. "Unmasking TRaccoon: A Lattice-Based Threshold Signature with An Efficient Identifiable Abort Protocol". In: *CRYPTO 2025, Part VI*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16005. LNCS. Springer, Cham, Aug. 2025, pp. 423–456. DOI: 10.1007/978-3-032-01887-8_14.

[RRJ+22] Tim Ruffing, Viktoria Ronge, Elliott Jin, Jonas Schneider-Bensch, and Dominique Schröder. "ROAST: Robust Asynchronous Schnorr Threshold Signatures". In: *ACM CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM Press, Nov. 2022, pp. 2551–2564. DOI: 10.1145/3548606.3560583.

[Sah99] Amit Sahai. "Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security". In: *40th FOCS*. IEEE Computer Society Press, Oct. 1999, pp. 543–553. DOI: 10.1109/SFFCS.1999.814628.

[Sho00] Victor Shoup. "Practical Threshold Signatures". In: *EUROCRYPT 2000*. Ed. by Bart Preneel. Vol. 1807. LNCS. Springer, Berlin, Heidelberg, May 2000, pp. 207–220. DOI: 10.1007/3-540-45539-6_15.

[Sho97] Victor Shoup. "Lower Bounds for Discrete Logarithms and Related Problems". In: *EUROCRYPT'97*. Ed. by Walter Fumy. Vol. 1233. LNCS. Springer, Berlin, Heidelberg, May 1997, pp. 256–266. DOI: 10.1007/3-540-69053-0_18.

[WNR20] Pieter Wuille, Jonas Nick, and Tim Ruffing. *Schnorr Signatures for secp256k1*. Bitcoin Improvement Proposal 340. https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki. 2020.

[YCX21] Tsz Hon Yuen, Handong Cui, and Xiang Xie. "Compact Zero-Knowledge Proofs for Threshold ECDSA with Trustless Setup". In: *PKC 2021, Part I*. Ed. by Juan Garay. Vol. 12710. LNCS. Springer, Cham, May 2021, pp. 481–511. DOI: 10.1007/978-3-030-75245-3_18.

A Additional Preliminaries

We provide additional preliminaries.

A.1 Commitment Schemes

A commitment scheme with message space $\mathcal{M}$ is a tuple of algorithms $\mathsf{CS} := (\mathsf{KGen}, \mathsf{Com}, \mathsf{VerCom})$ with the following syntax:

- $\mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda)$: takes as input the security parameter (and possibly group parameters) and outputs a commitment key ck.
- $(\mathsf{C}, o) \leftarrow \mathsf{Com}(\mathsf{ck}, m; \omega)$: takes as input the commitment key ck, a message $m \in \mathcal{M}$, and randomness ω, and outputs a commitment C and an opening o.
- $b \leftarrow \mathsf{VerCom}(\mathsf{ck}, \mathsf{C}, m, o)$: takes as input the commitment key ck, a commitment C, a message $m \in \mathcal{M}$, an opening o, and it accepts ($b = 1$) or rejects ($b = 0$).

Besides correctness, a commitment scheme CS satisfies two more properties.

Definition 3 (Binding) A commitment scheme CS is (computationally) binding if no PPT adversary can find, except with negligible probability, a commitment C, two messages $m \neq m'$ and two openings o, o':

$$\mathsf{VerCom}(\mathsf{ck}, \mathsf{C}, m, o) = \mathsf{VerCom}(\mathsf{ck}, \mathsf{C}, m', o') = 1$$

Definition 4 (Hiding) A commitment scheme CS is (computationally) hiding if $\forall m, m'$, $\forall \mathsf{ck}$:

$$\{\mathsf{C} : (\mathsf{C}, o) \leftarrow \mathsf{Com}(\mathsf{ck}, m)\} \approx \{\mathsf{C}' : (\mathsf{C}', o') \leftarrow \mathsf{Com}(\mathsf{ck}, m')\}$$

In contrast, perfect hiding/binding strengthens this guarantee by requiring security even against unbounded adversaries.

Trapdoor commitment schemes. In some applications, one further requires the trapdoor property for commitment schemes [GKS24]. That is, we require the existence of three additional PPT algorithms $(\mathsf{tdKeyGen}, \mathsf{tdComm}, \mathsf{tdOpen})$. The first outputs $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{tdKeyGen}(1^\lambda)$, where ck is a commitment key indistinguishable from an honestly generated key, and td is a commitment trapdoor. The second algorithm outputs $(\mathsf{C}, st) \leftarrow \mathsf{tdComm}(\mathsf{td})$, where C is a commitment and st is a piece of side information. The third outputs a commitment opening $o \leftarrow \mathsf{tdOpen}(\mathsf{C}, m, st, \mathsf{td})$. The requirement is that o can be used to open C to value m without triggering a rejection, for any message m of our choice. A trapdoor commitment is said to be ϵ-equivocable if the statistical distance between a honestly generated commitment key, commitment, and opening, and an analogous triple generated by $(\mathsf{tdKeyGen}, \mathsf{tdComm}, \mathsf{tdOpen})$ is at most ϵ. We refer to [GKS24] for a formal definition.

Pedersen commitment scheme. We mostly rely on the Pedersen commitment scheme [Ped92] with message space $\mathbb{Z}_p^N$, for some $N \in \mathbb{N}$, over some group $\mathbb{G}$, that is computationally binding under the DL assumption and perfectly hiding, and works as follows:

- $\mathsf{KGen}(1^\lambda)$ outputs $N + 1$ random generators $g_1, \ldots, g_N, h$ of $\mathbb{G}$.

- $\mathsf{Com}(\mathsf{ck}, \vec{a}; \omega)$ parses ck as $(g_1, \ldots, g_N, h)$ and outputs the commitment $\mathsf{C} := \prod_{i \in [N]} g_i^{a_i} h^{\omega}$ and the opening ω.
- $\mathsf{VerCom}(\mathsf{ck}, \mathsf{C}, \vec{a}, o)$ outputs 1 if and only if $\mathsf{Com}(\mathsf{ck}, \vec{a}; o) = \mathsf{C}$.

The Pedersen commitment scheme is a perfectly equivocable (i.e., with factor $\epsilon = 0$) trapdoor commitment scheme if endowed with the following extra algorithms:

- $\mathsf{tdKeyGen}(1^\lambda)$: sample $e_1, \ldots, e_N, \rho \leftarrow\!\!\$\ \mathbb{Z}_p$. Set $g_i = g^{e_i}$ for all $i \in [N]$, and $h = g^\rho$. Output $\mathsf{ck} = g_1, \ldots, g_N, h$ and $\mathsf{td} = e_1, \ldots, e_N, \rho$.
- $\mathsf{tdComm}(\mathsf{td})$: sample $r \leftarrow\!\!\$\ \mathbb{Z}_p$. Output (g^r, r).
- $\mathsf{tdOpen}(\mathsf{C}, \vec{a}, \mathsf{td})$: solve the linear equation $\rho\omega + \sum_{i=1}^{N} e_i a_i = r$. Return ω.

A.2 Non-interactive Arguments

A *non-interactive argument system* (NARG) for relation $\mathcal{R}$ in the random oracle model, denoted by Π, consists of a tuple of algorithms $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ having black-box access to a random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$, with the following syntax:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: takes as input the security parameter 1^λ and outputs public parameters pp.
- $\pi \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \mathbb{w})$: takes as input parameters pp, a statement $\mathbb{x}$ and witness $\mathbb{w}$, and outputs a proof π if $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$.
- $b \leftarrow \mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \pi)$: takes as input parameters pp, a statement $\mathbb{x}$ and proof π, and it accepts $(b = 1)$ or rejects $(b = 0)$.

In this work, we consider NARGs with *transparent* setup, i.e. pp can be generated with a call to the random oracle. For this reason, we may omit pp in the description of the prover and verifier algorithms as it is implicit.

Definition 5 (Completeness) A NARG Π satisfies perfect *completeness* if for every $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, it holds that

$$\Pr\Big[b = 1 : \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); \pi \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \mathbb{w}); b \leftarrow \mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \pi)\Big] = 1.$$

Definition 6 (Soundness) A NARG Π satisfies *soundness* if for every PPT adversary $\mathcal{A}$, it holds that

$$\Pr\Big[b = 1 \wedge \mathbb{x} \notin \mathcal{L}_{\mathcal{R}} : \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); (\mathbb{x}, \pi) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathsf{pp}); b \leftarrow \mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \pi)\Big] \leq \mathit{negl}(\lambda)$$

where $\mathcal{L}_{\mathcal{R}} := \{\mathbb{x} \,|\, \exists \mathbb{w} : (\mathbb{x}, \mathbb{w}) \in \mathcal{R}\}$ is the set of true-statements.

If soundness holds with respect to a computationally unbounded adversary, we say that the argument has *statistical* (or *information-theoretic*) soundness, and we call the argument a *proof* system.

Zero-knowledge. Informally, an argument is zero-knowledge if a proof reveals no information about the witness [GMR85]. We formalize this property by following the syntax of Faust et al. [FKM+12]. A zero-knowledge simulator $\mathcal{S}$ is defined as a stateful algorithm, whose initial state $st = \mathsf{pp}$, that operates in two modes. The first mode, $(y, st') \leftarrow \mathcal{S}(1, st, a)$ takes care of handling calls to the oracle H on input a query a. The second mode, $(\pi, st') \leftarrow \mathcal{S}(2, st, \mathbb{x})$ simulates a proof for the input statement $\mathbb{x}$. We define the following "wrapper" oracles, that are stateful and share their internal state:

- $\mathcal{S}_1(a)$ returns the first output of $\mathcal{S}(1, st, a)$;
- $\mathcal{S}_2(\mathbb{x}, \mathbb{w})$ returns the first output of $\mathcal{S}(2, st, \mathbb{x})$ if $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and $\perp$ otherwise;
- $\mathcal{S}'_2(\mathbb{x})$ returns the first output of $\mathcal{S}(2, st, \mathbb{x})$.

Definition 7 (Zero-knowledge) A NARG Π is zero-knowledge if there exists a simulator $\mathcal{S}$, with wrapper oracles $\mathcal{S}_1, \mathcal{S}_2$, such that for all PPT adversaries $\mathcal{A}$ it holds that:

$$\Pr\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathcal{A}^{\mathcal{P},\mathsf{H}}(\mathsf{pp}) = 1\end{array}\right] \approx \Pr\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}(\mathsf{pp}) = 1\end{array}\right]$$

Notice that zero-knowledge is a security property that is only guaranteed for valid statements in the language, hence the above definition uses $\mathcal{S}_2$ as a proof simulation oracle. Also, a zero-knowledge NARG with statistical soundness is simply referred to as a non-interactive zero-knowledge proof (NIZK) [BFM88].

Simulation exctractability. A strong security property of NARGs is *simulation extractability* [GM17]. Informally, from a prover producing a valid proof after seeing polynomially many proofs provided by a simulator (even for *false* statements), it is possible to extract a valid witness. This property can be formalized in many different ways, depending on the model and the constraints of the adversary and the extractor (see [BCC+25; DDO+01; FFR24; JP14]).

Definition 8 (Simulation extractability) A NARG Π is simulation-extractable with respect to a zero-knowledge simulator $\mathcal{S}$, with wrapper oracles $\mathcal{S}_1, \mathcal{S}'_2$, if for all PPT adversaries $\mathcal{A}$ there exists a PPT extractor $\mathcal{E}$ such that:

$$\Pr\left[\begin{array}{l|l}\mathcal{V}^{\mathcal{S}_1}(\mathbb{x}, \pi) = 1 & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \wedge\ (\mathbb{x}, \pi) \notin \mathcal{Q} & (\mathbb{x}, \pi) \leftarrow \mathcal{A}^{\mathcal{S}_1,\mathcal{S}'_2}(\mathsf{pp}) \\ \wedge\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R} & \mathbb{w} \leftarrow \mathcal{E}(st, \mathbb{x}, \pi)\end{array}\right] \in negl(\lambda)$$

where st is the final internal state of $\mathcal{S}$, and $\mathcal{Q}$ is the set of statement-proof pairs obtained by $\mathcal{A}$ when interacting with $\mathcal{S}'_2$.

The previous definition implies the weaker notion of *simulation soundness* [Sah99]. Informally, a NARG is simulation-sound if it is infeasible to generate a valid proof π for a false statement $\mathbb{x}$, even after seeing polynomially many simulated proofs. The implication holds because Definition 8 imposes a negligible probability of extraction failure, and extraction is by definition impossible for a false statement.

$\mathsf{TS\text{-}Correct}(vk, m, S, \{(sk_i, vk_i)\}_{i\in[n]})$

1 : **if** $S \not\subseteq [n] \vee |S| \leq t$: **return** $\perp$
2 : **for** $i \in S$: $(st_i, \mathsf{pm}_i) \leftarrow \mathsf{Sign}_1(i, sk_i, vk, \{vk_i\}_{i\in[n]})$
3 : **for** $i \in S$: $s_i \leftarrow \mathsf{Sign}_2(i, st_i, sk_i, vk, m, S, \{(vk_i, \mathsf{pm}_i)\}_{i\in S})$
4 : $\sigma \leftarrow \mathsf{Combine}(m, S, \{s_i\}_{i\in S})$
5 : **return** $\mathsf{Verify}(vk, m, \sigma)$

Figure 9: The TS-Correct procedure for a two-round threshold signature scheme.

A.3 Threshold Signatures

Correctness. Intuitively, a threshold signature scheme is correct if a signature generated by honest users is accepted by the verification procedure. Formally, TS is perfectly correct if for all λ, n, t, all $par \leftarrow \mathsf{Setup}(\lambda, t, n)$ and $(vk, \{(sk_i, vk_i)\}_{i\in[n]}) \leftarrow \mathsf{KeyGen}(par)$, and all signing sets $S \subseteq [n]$ with $|S| \geq t$

$$\Pr[\mathsf{TS\text{-}Correct}(vk, m, S, \{(sk_i, vk_i)\}_{i\in[n]}) \to 1] = 1,$$

with $\mathsf{TS\text{-}Correct}(vk, m, S, \{(sk_i, vk_i)\}_{i\in[n]})$ defined as in Figure 9, and the probability taken over the random choices of all randomized algorithms.

B General Description and Analysis of Matrices

Forgery matrix. We can in general represent the matrix at the point of the forgery as

$$\begin{pmatrix} \vec{c} & M_1 & \dots M_m & \hat{M}_1 & \dots & \hat{M}_m \end{pmatrix},$$

with $\vec{c} = \begin{pmatrix} c_1 & \dots & c_m & c^* + \alpha_{vk} \end{pmatrix}^T$, and

$$M_i = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \dots & 0 \\ & \mathsf{OneHot(S_i)} & \\ 0 & \dots & 0 \\ \vdots & \vdots & \vdots \\ \beta_i^{(1)} & \dots & \beta_i^{(|S_i|)} \end{pmatrix},$$

and

$$\hat{M}_i = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \dots & 0 \\ & b_i \cdot \mathsf{OneHot(S_i)} & \\ 0 & \dots & 0 \\ \vdots & \vdots & \vdots \\ \gamma_i^{(1)} & \dots & \gamma_i^{(|S_i|)} \end{pmatrix}.$$

In the above, $\mathsf{OneHot}(S_i)$ denotes the standard one-hot encoding of S_i as a subset of $[n]$, and the first non-null row of both M_i and $\hat{M}_i$ is the i-th one. We mark it as one-hot encoding, since honest, participating signers are represented by 1 and dishonest, or non-participating signers are marked as 0.

Case 1. The matrix obtained at the end of the first case of trivial non-linearity (Case 1) has the form

$$\begin{pmatrix} L_1 & \dots L_m & \hat{L}_i \end{pmatrix},$$

with

$$L_i = \begin{pmatrix} \mathsf{OneHot(i)} & \beta_i \end{pmatrix}^T, \quad \hat{L}_i = \begin{pmatrix} b_i \cdot \mathsf{OneHot(i)} & \gamma_i \end{pmatrix}^T,$$

where $\mathsf{OneHot(i)}$ denotes the binary vector of length $m-1$ where only the i-th entry is 1, and everything else is set to 0. For the last row to be a linear combination of the previous ones, the structure of the L_i columns imposes that $\gamma_i = b_i\gamma_i$ for all i's. Therefore, since for the remaining γ_i, it holds, that $\gamma_i \beta_i \cdot b_i$, the full-rank is guaranteed.

Case 2. The general structure of the matrix obtained at the end of Case Item 2 is

$$\begin{pmatrix} M_1 & \dots M_m \end{pmatrix},$$

with M_i defined as above. We immediately note that the last row is a linear combination of the previous m ones iff there exists a constant $\kappa_i \in \mathbb{Z}_p$ such that $\begin{pmatrix} \beta_1^{(i)} & \dots & \beta_{|S_i|}^{(i)} \end{pmatrix} = \begin{pmatrix} \kappa_i & \dots & \kappa_i \end{pmatrix}$. Therefore, as soon as there exist two $j \neq k$ such that $\beta_k^{(i)} \neq \beta_j^{(i)}$, the above condition is contradicted, and the resulting matrix is full rank.

Final matrix. After excluding Cases 1 and 2, we can verify that the general matrix has structure

$$\begin{pmatrix} \vec{c} & N_1 & \dots N_m & \hat{N}_1 & \dots & \hat{N}_m \end{pmatrix},$$

with $\vec{c}$ defined as before and

$$N_i = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \dots & 0 \\ & \mathsf{OneHot(S_i)} & \\ 0 & \dots & 0 \\ \vdots & \vdots & \vdots \\ \beta_i & \dots & \beta_i \end{pmatrix}, \quad \hat{N}_i = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \dots & 0 \\ & b_i \cdot \mathsf{OneHot(S_i)} & \\ 0 & \dots & 0 \\ \vdots & \vdots & \vdots \\ b_i\beta_i & \dots & b_i\beta_i \end{pmatrix}.$$

From the latter, following the procedure outline in the main body, we obtain a *final matrix* having the following form

$$\begin{pmatrix} c_1 & 1 & 0 & \dots & 0 \\ c_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ c_m & 0 & 0 & \dots & 1 \\ c^* + \alpha_{vk} & \beta_1 & \beta_2 & \dots & \beta_m \end{pmatrix}.$$