Earpicks: Tightly Secure Two-Round Multi- and Threshold Signatures

Renas Bacho* Yanbo Chen†

March 22, 2026

Abstract

Multi-signatures are a fundamental cryptographic primitive in distributed systems, enabling a set of parties to jointly produce a compact signature on a common message. Of particular interest are constructions instantiated over pairing-free cyclic groups with a two-round signing protocol, as such schemes offer improved efficiency and deployability in practice. Support for key aggregation is an additional highly desirable property, allowing multiple public keys to be combined into a single succinct aggregate public key against which aggregate signatures can be verified. To improve concrete security guarantees, several works have proposed constructions with tight security reductions. However, existing tightly secure constructions have significant limitations. Notably, T-Spoon by Bacho and Wagner (Crypto 2025) is currently the only pairing-free two-round multi-signature scheme that simultaneously achieves tight security and supports key aggregation. Despite these advantages, T-Spoon incurs substantial efficiency overhead: its signatures comprise nine field elements and two group elements, resulting in prohibitively large signature sizes for many practical applications.

In this work, we introduce Earpick-MS, a tightly secure two-round multi-signature scheme over pairing-free cyclic groups that supports key aggregation while achieving compact signatures. Concretely, signatures in Earpick-MS consist of only three field elements and a single bit, thereby reducing the signature size by a factor of approximately 3.5 compared to the state-of-the-art T-Spoon construction. We further present Earpick-TS, a threshold signature variant of our scheme. Earpick-TS retains the same compact signature size and constitutes the first pairing-free two-round threshold signature scheme with a tight security proof. Prior to our work, achieving tight security in pairing-free threshold signatures required at least three rounds of interaction (Chen, PKC 2025; Bacho and Wagner, CiC 2026). Finally, we propose Earpick-muMS, an additional variant that achieves tight security in the multi-user setting while retaining the same compact signature size.

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

1 Introduction

A multi-signature scheme [Ita83] allows a set of n signers, each signer i holding an independent key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$, to collectively sign a common message μ. The generated signature σ (on message μ) can then be verified against the list of individual public keys $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ of signers. In recent years, multi-signatures have attracted considerable attention, largely due to their applications to cryptocurrencies. Of particular interest are constructions over *pairing-free cyclic groups* with a signing protocol consisting of *two rounds* of communication. Further, support for *key aggregation* is highly desired [MPSW19], as it allows the aggregation of multiple public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$ into a

*CISPA Helmholtz Center for Information Security, Saarland University. Email: renas.bacho@cispa.de
†University of Ottawa. Email: ychen918@uottawa.ca.

single short aggregate public key apk that can then be used for verification of σ, enhancing efficiency significantly. In this work, we therefore focus on *two-round multi-signatures with key aggregation over pairing-free cyclic groups*. We further stress that we work in the widely established plain public key model [BN06], where honest signers generate their keys independently, and malicious signers may choose their keys dependent on other keys. In particular, no additional assumptions such as knowledge of secret keys [Bol03] or proofs of possession [BCJ08] are required.

Concrete Security and Tightness. Security of cryptographic schemes is typically established via reductionist proofs. Specifically, to show the security of a scheme Π, one constructs a transformation that converts any adversary A with success probability ε_{A} against Π into an algorithm R that solves an underlying computational problem Σ with success probability ε_{R}. This then yields a relation of the form $\varepsilon_{\mathsf{A}} \leq L \cdot \varepsilon_{\mathsf{R}}$, where L is called the *security loss*. Minimizing the security loss L is critical, as it has direct impact on the concrete security guarantees of the scheme relative to the hardness of Σ. Concretely, if we assume the underlying problem Σ to be κ-bit secure (i.e., $\varepsilon_{\mathsf{R}} \leq 2^{-\kappa}$), then the proof guarantees $(\kappa - \log L)$ bits of security for the scheme Π. In particular, if L is large, then the concrete parameters of Π, such as the size of the underlying cyclic group, may have to be chosen prohibitively large, limiting the practical efficiency. Interestingly, Kales and Zaverucha [KZ20] give an efficient attack on the MQDSS signature scheme [SSH11], which leverages the observation that the concrete parameters of MQDSS were chosen without accounting for L. Consider for example a scheme with a guessing-based security proof yielding the bound $\varepsilon_{\mathsf{A}} \leq \Theta(q_{\mathrm{h}}) \cdot \varepsilon_{\mathsf{R}}$, where q_{h} is the number of random oracle queries made by the adversary. Then, if the scheme is instantiated with a group providing 128-bit security (i.e., $\kappa = 128$), the resulting security level would be only 68 bits for a practical reasonable $q_{\mathrm{h}} \approx 2^{60}$ [BR96, Nat16].[1] This motivates the study of schemes with a *tight* security proof, i.e., where the security loss L is a small constant.

Recent Progress in Multi-Signatures. Early designs of two-round multi-signatures [NRSW20, NRS21, BD21] rely on the (double-)forking lemma [PS96, BN06] for their security proof, which results in highly non-tight security bounds of the form $\varepsilon_{\mathsf{A}}^4 \leq \Theta({q_{\mathrm{h}}}^3) \cdot \varepsilon_{\mathsf{R}}$. To address this unsatisfactory state of affairs, Pan and Wagner [PW23] recently constructed the first tightly secure, two-round multi-signature, Chopsticks II. However, their scheme does not support key aggregation. Even more recently, Bacho and Wagner [BW25] proposed the first tightly secure, two-round multi-signature with key aggregation, T-Spoon. Unfortunately, this comes with a considerable penalty: Signatures are about 4 times larger than most non-tight, two-round multi-signatures. Concretely, signatures in T-Spoon consist of nine field elements and two group elements, while those in, e.g., HBMS [BD21] consist of only three field elements. This creates a considerable efficiency gap between tight and non-tight two-round multi-signatures with key aggregation. This raises the central question:

Can we design a tightly secure two-round multi-signature with key aggregation and short signatures, with security based on non-interactive and well-studied pairing-free assumptions?

1.1 Our Contributions

We resolve this question by presenting Earpick-MS, a novel tightly secure two-round multi-signature scheme. Earpick-MS is the first two-round multi-signature scheme in the setting of pairing-free cyclic groups with the following desirable characteristics:

- *Tightness.* Our scheme is tightly secure under the decisional Diffie-Hellman (DDH) assumption.
- *Compactness.* Our scheme has signatures consisting of three field elements and a single bit.
- *Key Aggregation.* Our scheme supports key aggregation.

[1] We note that rewinding-based proofs yield even much worse security bounds, making them essentially useless in terms of concrete security. For example, the proof of the MuSig2 [NRS21] multi-signature scheme provides a security level of only 9 bits for a generously chosen $q_{\mathrm{h}} \approx 2^{30}$ (see Table 1).

For a comparison with existing two-round schemes, see Table 1. We also emphasize that our scheme does not make use of any heavy cryptographic tools and is very practical.

As a secondary contribution, we construct a *threshold* variant of Earpick-MS, called Earpick-TS. A (t, n)-threshold signature [Des88] allows any subset of $(t+1)$-out-of-n signers to collectively sign a common message μ, where the generated signature σ can then be verified against a single common public key pk. Existing constructions of tightly secure threshold signatures [Che25, BW24b] have a signing protocol of three rounds of communication, which limits their practicality. Our tightly secure two-round threshold signature Earpick-TS addresses this drawback. Concretely, as for Earpick-MS, its security relies on the DDH assumption and it has compact signatures consisting of only 3 field elements and a single bit. In particular, our work resolves the open question raised by Chen [Che25] on the existence of a tightly secure two-round threshold signature.[2]

Concurrent and Independent Work. In a concurrent work, Lehmann and Özbay [LO25] initiate the study of tight multi-user security for multi-signatures. Standard notions of unforgeability for multi-signatures are typically formulated with respect to a single honest signer. In the corresponding security reductions, the challenger must guess this honest party in advance, resulting in a multiplicative security loss of $1/n$, where n denotes the number of participating signers.

To address this issue, Lehmann and Özbay propose Skewer-PF, a pairing-free two-round multi-signature scheme, achieving tight multi-user security. The scheme is proven tightly secure under the DDH assumption in the random oracle model and supports key aggregation. In Skewer-PF, signatures consist of three field elements and one group element, while the public key comprises one group element augmented with proofs of possession (two field elements and one group element). We emphasize that public keys augmented with proofs of possession do not fall within the widely adopted plain public-key model in which we work. Indeed, requiring proofs of possession corresponds to a strictly stronger model. For a detailed discussion of proofs of possession and their implications, we refer to the seminal work of Bellare and Neven [BN06].

After examining their manuscript, we observed that our scheme Earpick-MS can be adapted to achieve tight multi-user security by a minor modification of the signing protocol, without altering the final signature or the public keys. The resulting variant, Earpick-muMS, achieves tight security in the multi-user setting under the DDH assumption in the random oracle model and established plain public-key model. Its signatures consist of three field elements and a single bit, and its public key comprises two group elements.

2 Technical Overview

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

2.1 Takemure et al.'s Scheme

Our scheme is based on Takemure et al.'s two-round multi-signature scheme [TSS+24]. The scheme works over a cyclic group $\mathbb{G}$ of prime order p with generator g, and the public parameter additionally contains a random group element h. Each user has its secret key $\mathsf{sk} := x$ uniformly chosen from $\mathbb{Z}_p$, and public key $\mathsf{pk} := (X, Y) = (g, h)^x$. The scheme supports public-key aggregation: a list of public key $\mathcal{L} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ is aggregated into $\mathsf{apk} := (\tilde{X}, \tilde{Y}) = \prod_{i\in[n]}(X_i, Y_i)^{a_i}$, where the coefficient $a_i := \mathsf{H}_a(\mathcal{L}, (X_i, Y_i))$ is derived by a hash function.

[2] We note, however, that the existing tightly secure threshold signatures (with three rounds) [Che25, BW24b] achieve the strong notion of *adaptive security*, while ours only achieves static security (also see Section 2.4). As such, it remains an interesting open problem to construct a tightly secure two-round threshold signature with adaptive security.

Scheme	Key Agg	Assumption	Loss	PKey	Communication	Signature
MuSig-DN [NRSW20]	✓	DDH, DL	$\Theta(q_{\mathrm{h}}{}^3/\varepsilon^3)$	$1\mathbb{G}$	$2\mathbb{G}+1\mathbb{Z}_p+\lvert\pi\rvert$	$2\mathbb{Z}_p$
MuSig2 [NRS21]	✓	AOMDL	$\Theta(q_{\mathrm{h}}{}^3/\varepsilon^3)$	$1\mathbb{G}$	$4\mathbb{G}+1\mathbb{Z}_p$	$2\mathbb{Z}_p$
HBMS [BD21]	✓	DL	$\Theta(q_{\mathrm{s}}{}^4 q_{\mathrm{h}}{}^3/\varepsilon^3)$	$1\mathbb{G}$	$1\mathbb{G}+2\mathbb{Z}_p$	$3\mathbb{Z}_p$
TZ [TZ23]	✓	DL	$\Theta(q_{\mathrm{h}}{}^3/\varepsilon^3)$	$1\mathbb{G}$	$1\mathbb{G}+2\mathbb{Z}_p$	$3\mathbb{Z}_p$
Chopsticks I [PW23]	✓	DDH	$\Theta(q_{\mathrm{s}})$	$2\mathbb{G}$	$3\mathbb{G}+1\mathbb{Z}_p+\lambda$	$3\mathbb{G}+4\mathbb{Z}_p$
Toothpicks I [PW24]	✓	DDH	$\Theta(q_{\mathrm{s}})$	$2\mathbb{G}$	$2\mathbb{G}+1\mathbb{Z}_p+\lambda$	$3\mathbb{Z}_p+2\lambda$
TSSHO [TSS+24]	✓	DDH	$\Theta(q_{\mathrm{s}})$	$2\mathbb{G}$	$2\mathbb{G}+2\mathbb{Z}_p$	$3\mathbb{Z}_p$
Chopsticks II [PW23]	✗	DDH	$\Theta(1)$	$4\mathbb{G}$	$6\mathbb{G}+2\mathbb{Z}_p+\lambda+1$	$6\mathbb{G}+8\mathbb{Z}_p+n$
Toothpicks II [PW24]	✗	DDH	$\Theta(1)$	$4\mathbb{G}$	$2\mathbb{G}+1\mathbb{Z}_p+\lambda+1$	$3\mathbb{Z}_p+2\lambda+n$
T-Spoon [BW25]	✓	DDH	$\Theta(1)$	$2\mathbb{G}$	$3\mathbb{G}+1\mathbb{Z}_p+\lambda+1$	$2\mathbb{G}+9\mathbb{Z}_p$
Earpick-MS (ours)	✓	DDH	$\Theta(1)$	$2\mathbb{G}$	$5\mathbb{G}+4\mathbb{Z}_p$	$3\mathbb{Z}_p+1$

Table 1: Comparison of two-round multi-signature schemes over pairing-free cyclic groups in the plain public key model. We consider support for key aggregation, underlying computational assumption, security loss of the reduction (for an adversary with advantage ε against the scheme that makes at most q_{h} random oracle queries and q_{s} signing queries), size of public keys, size of signatures, and communication cost per signer. For the security loss, we do not consider proofs in the algebraic group model (AGM) [FKL18]. For communication and signature sizes (relevant only for some of the schemes), λ is the security parameter and n is the number of signers. In MuSig-DN, $|\pi|$ is the size of a zkSNARK and DDH is assumed in a distinct cyclic group $\mathbb{E}$ (used for proving correct computation of the deterministic nonces).

For a group of signers with public-key list $\mathcal{L}$ to sign message μ, they run a signing protocol with two rounds of message exchange. Each signer (without loss of generality indexed by 1) does the following.

- In the first round, the signer hashes the aggregated public key and the message to obtain $(u,v) := \mathsf{H}_{uv}(\mathsf{apk},\mu)$. It samples $r_1, s_1 \leftarrow \mathbb{Z}_p$ and broadcast a commitment $(R_1, S_1) := (g,h)^{r_1}(u,v)^{s_1}$ to other signers.
- In the second round, the signer receives $\{(R_i,S_i)\}_{i\in[n]}$ and computes the aggregated commitment $(\tilde{R},\tilde{S}) := \prod_{i\in[n]}(R_i,S_i)$. Then it derives the challenge $c := \mathsf{H}_c(\mathsf{apk},\mu,\tilde{R},\tilde{S})$ from a hash function H_c. It broadcasts its response $(z_1,y_1) := (r_1 + a_1 c x_1, s_1)$.
- Finally, the responses are aggregated into $(\tilde{z},\tilde{y}) := \sum_{i\in[n]}(z_i,y_i)$, and the multi-signature is output to be $(c,\tilde{z},\tilde{y})$.

To verify the multi-signature, the verifier recovers the commitment $(\tilde{R},\tilde{S}) := (g,h)^{\tilde{z}}(u,v)^{\tilde{y}}(\tilde{X},\tilde{Y})^{-c}$ and checks if c is the correct hash.

Security. In the security game, there is an honest signer with public key $(X,Y) = (g,h)^x$. The adversary can execute the signing protocol with the honest signer, arbitrarily choosing other signers' keys. The goal is to forge a multi-signature with the honest signer included in the signer set.

To understand the security proof, it is helpful to note that in the signing protocol, each signer essentially generates a Schnorr-Okamoto-like proof of knowledge [Sch90, Oka93] of (α,β) such that $(g,h)^\alpha(u,v)^\beta = X_1$, using $(\alpha,\beta) = (x_1,0)$ as the witness.

To prove the security of the scheme, we switch to a hybrid game in an indistinguishable way based on DDH, where forging a multi-signature is unconditionally hard. First, we reach an intermediate hybrid game. In this game, the space of (apk,μ) is partitioned into two parts: a "dual-key" part and a "lossy" part. The game programs $(u,v) = \mathsf{H}_{uv}(\mathsf{apk},\mu)$ differently for the two parts:

- The "dual-key" part: The game programs $(u, v) := (X, Y)^{1/w}$ for $w \leftarrow \mathbb{Z}_p^*$. Here, w is called the dual key, which allows the game to simulate the honest signer without secret key x. Namely, the game simply generates the Schnorr-Okamoto-like proof of knowledge using another witness $(\alpha, \beta) = (0, w)$. The simulation is perfect because of the witness indistinguishability of such proofs.
- The "lossy" part: The game programs $(u, v) := (g, h)^w$

We hope that the honest signer only needs to sign on (apk, μ) in the dual-key part, while the adversary forges a signature on (apk, μ) in the lossy part. With Coron's coin-tossing partitioning technique [Cor00], this is true with probability $\Theta(1/q)$.

Next, we reach the final hybrid game by sample (X, Y) to be independently random elements. With overwhelming probability, (g, h, X, Y) is not a DH-tuple, i.e., there does not exist a real secret key. The game can still simulate signing with the dual keys. However, signing with (apk, μ) in the lossy part becomes statistically hard. Since $(u, v) = (g, h)^w$ for some w, there does not exist a witness (α, β) satisfying $(g, h)^\alpha (u, v)^\beta = (X, Y)$ for the Schnorr-Okamoto-like proof. Signing becomes statistically hard because of the soundness of such proofs.

2.2 Existing Constructions with Tight Security

Pseudorandom Bit for Tight Security. The above proof does not invoke the forking lemma [PS96, BN06] so avoids a (double-)quadratic loss. However, the security is still not fully tight. The only problem is the loss of $\Theta(q)$ owing to the partitioning of the space of (apk, μ).

Katz and Wang's pseudorandom bit technique [KW03] can avoid such a loss by replace "partitioning" with "branching". A straightforward (but not working) application to Takemure et al.'s scheme is as follows. Now each (apk, μ) corresponds to two hashes $\mathsf{H}_{uv}(0, \mathsf{apk}, \mu)$ and $\mathsf{H}_{uv}(1, \mathsf{apk}, \mu)$. In the signing protocol, each signer additionally outputs a pseudorandom bit $b_1 := \mathsf{F}(\mathsf{apk}, \mu)$ as a branch selection, and it signs with $(u, v) = \mathsf{H}_{uv}(b_1, \mathsf{apk}, \mu)$. The multi-signature additional contains each signer's pseudorandom bit.

In the security proof, for every (apk, μ), the hybrid game randomly picks a branch b' to be the "dual-key" branch, programming $\mathsf{H}_{uv}(b', \mathsf{apk}, \mu)$ with a dual key, and the other branch to be the "lossy" branch, programming $\mathsf{H}_{uv}(1 - b', \mathsf{apk}, \mu)$ in the lossy way. It always selects the dual-key branch by setting $b_1 := b'$ when it signs. On the other hand, the adversary has about $1/2$ chance to pick the lossy branch, as the two branches are indistinguishable. The security loss is thus improved from $\Theta(q)$ to $\Theta(1)$.

However, Katz and Wang's technique, originally designed for standard (one-party) signature schemes, is not directly applicable to multi-signatures. In the signing protocol, the commitments $\{(R_i, S_i)\}_{i \in [n]}$ and responses $\{(z_i, y_i)\}_{i \in [n]}$ are aggregated to produce a compact multi-signature. The aggregatability relies on the fact that the signers use a common (u, v). Since each signer chooses one of the two possible (u, v), the aggregated signature is no longer valid.

Pseudorandom Bit for Multi-Signatures. Previous works have developed variants of Katz and Wang's technique for multi-signatures.[3] Pan and Wagner [PW24] proposed to branch the public key instead of the hash H_{uv}. Namely, the signers always use the same (u, v), but now each signer processes two key pairs, and one of them is pseudorandomly selected to sign every time. However, we then lose public-key aggregation, as now for the same set of signers, the "effective" public-key set can be different every time.

Alternatively, Bacho and Wagner [BW25] proposed to aggregate the commitments and responses separately in each of the two branches. Signers who choose the same branch use the same (u, v), so they can proceed correct aggregation. The final multi-signature contains two rather than one aggregated responses and some additional information. Their scheme can support key aggregation. Moreover, the pseudorandom bits are not included in the signatures, giving asymptotically smaller

[3] Their base schemes are not Takemure et al.'s scheme, but the techniques can be adopted.

signatures that contain constant number of elements. However, the signature size may actually become concretely worse when the number of signers is not too large, due to the double responses and extra information.

2.3 Our Solution

We propose a solution that achieves the best of both worlds. Our scheme supports key aggregation. At the same time, the signature size is concretely (not just asymptotically with growing number of signers) better than Pan and Wagner's key-branching solution.

Single Pseudorandom Bit? We observe that all the problems stem from the fact that each signer generates its own pseudorandom bit. Imagine that instead, a single bit b is given by a trusted party. Then they can sign based on the same $(u, v) = \mathsf{H}_{uv}(b, \mathsf{apk}, \mu)$ as in the original Takemure et al.'s scheme. Each signer only needs one key pair, and key aggregation is perfectly supported. The multi-signature contains only one extra bit compared with Takemure et al.'s scheme, resulting in a very compact size.

Delay Branch Selection. We clearly want to avoid assuming a trusted outsider. On the other hand, requiring signers to jointly compute the pseudorandom bit before the signing protocol would increase the round complexity. Our next observation is that, the secret key is not used in the first round of the signing protocol. In the security proof, even if the honest signer goes to the lossy branch, it will be stuck only in the second round. From this observation, we can modify the protocol to let each signer go with both branches in the first round, generating two commitments

$$(R_{1,0}, S_{1,0}) := (g, h)^{r_{1,0}}(u_0, v_0)^{s_{1,0}}, \quad (R_{1,1}, S_{1,1}) := (g, h)^{r_{1,1}}(u_1, v_1)^{s_{1,1}}$$

for $(u_0, v_0) = \mathsf{H}_{uv}(0, \mathsf{apk}, \mu)$ and $(u_1, v_1) = \mathsf{H}_{uv}(1, \mathsf{apk}, \mu)$. At the same time, the signers also run a protocol to jointly decide a bit b. In the second round, they continue with the b-branch and drop the other. This buys the signers one round of time to agree on a bit.

One-Round Bit Protocol. What remains for us is to find a one-round protocol for the signers to jointly evaluate the pseudorandom bit $b = \mathsf{F}(\mathsf{apk}, \mu)$. We want the hybrid game to fully control $\mathsf{F}(\mathsf{apk}, \mu)$ in the following sense. When (apk, μ) is queried to H_{uv}, the game uniformly samples $\mathsf{F}(\mathsf{apk}, \mu)$ and programs the two branches into H_{uv} based on the bit. Later when the adversary makes a signing query with (apk, μ), the game can ensure that the protocol gives $\mathsf{F}(\mathsf{apk}, \mu)$, despite the participation of adversarial signers.

Here is the protocol. Each user outputs $Z_1 := D^{x_1}$ for $D = \mathsf{H}_D(\mathsf{apk}, \mu)$ and a Schnorr-like proof of equality of discrete logarithms [CP93] for (g, D, X_1, Z_1). The protocol outputs $\mathsf{H}_b(\tilde{Z})$ for $\tilde{Z} := \prod_{i\in[n]} Z_i$. The soundness of the proof ensures that the signers honestly output their Z_i and thus guarantees a deterministic bit.

In the security proof, however, the hybrid game can output uniformly random Z_1 and simulate a proof. Consequently, $\tilde{Z}$ is unconditionally unpredictable until the adversary sees Z_1. Moreover, the game generates $D = g^d = \mathsf{H}_D(\mathsf{apk}, \mu)$ of known discrete logarithm, so it can computes $D^{x_i} = X_i^d$ without knowing the secret key x_i of the adversarial signer i. The game knows $\tilde{Z}$ earlier than the adversary and programs $\mathsf{H}_b(\tilde{Z}) := \mathsf{F}(\mathsf{apk}, \mu)$. This ensures that the protocol selects the dual-key branch as expected.

We note that $Z_1 = \mathsf{H}_D(\mathsf{apk}, \mu)^{x_1}$ is a verifiable random function [MRV99] in the ROM. We use the additional property that the controller of the random oracle can evaluate the function under an arbitrary public key without knowing the secret key. This property is also used in related context in [NRSW20]. Moreover, the function can be made threshold based on key homomorphism, which is important for our threshold signature scheme.

2.4 Open Questions

It is straightforward to extend our multi-signature construction to the threshold setting. However, the respective security proof crucially relies on the corruptions being static and *not* adaptive. We identify two points at which the reduction breaks down in the presence of adaptive corruptions.

First, on behalf of an honest party $i \in [n]$, the reduction outputs a uniformly random group element Z_i, together with a simulated proof, instead of the real value $Z_i = \mathsf{H}_D(\mathsf{apk}, \mu)^{x_i}$, where $\mathsf{apk} = \mathsf{pk}$ denotes the group public key and x_i is party i's secret key share. In the adaptive-corruption setting, the adversary may subsequently corrupt an honest party i, at which point the reduction must reveal the corresponding secret key x_i. Once x_i is revealed, the adversary can recompute the correct value of Z_i and detect that it differs from the uniformly random element previously output by the reduction. Hence, the simulation becomes inconsistent, and the reduction fails. More generally, the pseudorandom-bit (or controllable branch-selection) technique employed in the proof does not appear to extend to the adaptive setting.

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

Addressing both obstacles simultaneously appears to be highly non-trivial. In particular, it seems that novel techniques are required to achieve two-round signing together with tight adaptive security for threshold signatures. Furthermore, constructing a tightly secure, partially non-interactive multi-signature scheme remains an interesting open problem.

3 Preliminaries

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

3.1 Proof of Equality of Discrete Logarithms

In Fig. 1, we present a proof system DLEQ for the equality of discrete logarithms, which we will use in our schemes. It is an extension of the well-known Schnorr proof introduced in [CP93]. Our proof system works over a cyclic group $\mathbb{G}$ of prime order p and proves the following statement-witness relation $\mathcal{R}_{\text{dleq}} \subset \mathbb{G}^4 \times \mathbb{Z}_p$:

$$\mathcal{R}_{\text{dleq}} = \{((g, h, X, Y), \mathsf{wit}) \ : \ (g, h)^{\mathsf{wit}} = (X, Y)\}.$$

In Fig. 1, we first define a three-move public-coin proof system $(\mathsf{P}_1, \mathsf{P}_2, \mathsf{Vf})$. From the interactive proof system, we then define the non-interactive version $(\mathsf{NIP}^{\mathsf{H}}, \mathsf{NIV}^{\mathsf{H}})$ with respect to a hash function H, obtained via the well-known Fiat-Shamir transformation [FS87], where the challenge chl is derived non-interactively by hashing stm and cmt. The algorithm Sim (called *simulator*) will be used to define the zero-knowledge property of the proof system.

$\underline{\mathsf{P}_1(\mathsf{stm};\rho)}$
1: $(g,h,X,Y) := \mathsf{stm}$
2: $(A,B) := (g,h)^{\rho}$
3: **return** (A,B)

$\underline{\mathsf{P}_2(\mathsf{wit},\rho,\mathsf{chl})}$
4: **return** $\rho + \mathsf{chl}\cdot\mathsf{wit}$

$\underline{\mathsf{Vf}(\mathsf{stm},\mathsf{cmt},\mathsf{chl},\mathsf{rsp})}$
5: $(g,h,X,Y) := \mathsf{stm}$
6: $(A,B) := \mathsf{cmt}$
7: **return** $[\![(g,h)^{\mathsf{rsp}} = (A,B)(X,Y)^{\mathsf{chl}}]\!]$

$\underline{\mathsf{NIP}^{\mathsf{H}}(\mathsf{stm},\mathsf{wit})}$
8: $\rho \leftarrow \mathbb{Z}_p$
9: $\mathsf{cmt} := \mathsf{P}_1(\mathsf{stm};\rho)$
10: $\mathsf{chl} := \mathsf{H}(\mathsf{stm},\mathsf{cmt})$
11: $\mathsf{rsp} := \mathsf{P}_2(\mathsf{wit},\rho,\mathsf{chl})$
12: **return** $(\mathsf{cmt},\mathsf{rsp})$

$\underline{\mathsf{NIV}^{\mathsf{H}}(\mathsf{stm},\pi)}$
13: $(\mathsf{cmt},\mathsf{rsp}) := \pi$
14: $\mathsf{chl} := \mathsf{H}(\mathsf{stm},\mathsf{cmt})$
15: $\mathsf{Vf}(\mathsf{stm},\mathsf{cmt},\mathsf{chl},\mathsf{rsp})$

$\underline{\mathsf{Sim}(\mathsf{stm},\mathsf{chl})}$
16: $(g,h,X,Y) := \mathsf{stm}$
17: $\mathsf{rsp} \leftarrow \mathbb{Z}_p$
18: $(A,B) := (g,h)^{\mathsf{rsp}}/(X,Y)^{\mathsf{chl}}$
19: **return** $((A,B),\mathsf{rsp})$

Figure 1: Proof of discrete logarithm equality DLEQ.

In our constructions, we will use the non-interactive proof system. However, we will use the properties of the interactive system in the security analysis of our constructions, which are stated below.

Proposition 1. *The three-move public-coin proof system* $(\mathsf{P}_1,\mathsf{P}_2,\mathsf{Vf})$ *defined in Fig. 1 satisfies the following properties:*

- *Completeness: For every statement* $\mathsf{stm} = (g,h,X,Y) \in \mathbb{G}^4$ *and witness* $\mathsf{wit} \in \mathbb{Z}_p$ *such that* $(g,h)^{\mathsf{wit}} = (X,Y)$*, and every challenge* $\mathsf{chl} \in \mathbb{Z}_p$*, we have*

$$\Pr[\mathsf{Vf}(\mathsf{stm},\mathsf{cmt},\mathsf{chl},\mathsf{rsp}) \ : \ \mathsf{cmt} \leftarrow \mathsf{P}_1(\mathsf{stm};\rho),\ \mathsf{rsp} := \mathsf{P}_2(\mathsf{wit},\rho,\mathsf{chl})] = 1.$$

- *Statistical Soundness: For every statement* $\mathsf{stm} = (g,h,X,Y) \in \mathbb{G}^4$ *such that* $\mathsf{dlog}_g(X) \neq \mathsf{dlog}_h(Y)$*, for every commitment* $\mathsf{cmt} \in \mathbb{G}^2$*, there exists at most one challenge* $\mathsf{chl} \in \mathbb{Z}_p$ *such that there exists a response* rsp *satisfying* $\mathsf{Vf}(\mathsf{stm},\mathsf{cmt},\mathsf{chl},\mathsf{rsp}) = 1$*.*
- *Special Honest-Verifier Zero-Knowledge: For every statement-witness pair* $(\mathsf{stm},\mathsf{wit}) \in \mathcal{R}_{\mathrm{dleq}}$ *and every challenge* $\mathsf{chl} \in \mathbb{Z}_p$*, the commitment-response pairs* $(\mathsf{cmt},\mathsf{rsp})$ *output by the following two procedures are identically distributed:*
  1. $\mathsf{cmt} \leftarrow \mathsf{P}_1(\mathsf{stm};\rho)$*,* $\mathsf{rsp} := \mathsf{P}_2(\mathsf{wit},\rho,\mathsf{chl})$*;*
  2. $(\mathsf{cmt},\mathsf{rsp}) \leftarrow \mathsf{Sim}(\mathsf{stm},\mathsf{chl})$*.*

3.2 Computational Assumptions

Let GrGen be a group generation algorithm that takes the security parameter 1^λ as input and outputs group parameters $(\mathbb{G},p,g)$, where $\mathbb{G}$ is a cyclic group of prime order p and g is a generator.

Definition 1 (DDH Assumption)**.** The DDH problem is (τ,ε)-hard for group generation algorithm GrGen if for all λ, all τ-time algorithms A,

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

Definition 2 (CDH Assumption)**.** The CDH problem is (τ,ε)-hard for group generation algorithm GrGen if for all λ, all τ-time algorithms A,

$$\Pr[\mathsf{A}(\mathbb{G},p,g,h,g^a) = h^a \ : \ (\mathbb{G},p,g) \leftarrow \mathsf{GrGen}(1^\lambda),\ h \leftarrow \mathbb{G},\ a \leftarrow \mathbb{Z}_p] \leq \varepsilon.$$

$\underline{\mathsf{MSJointSign}(\{\mathsf{pk}_i\}_{i\in[n]}, \mu, \{\mathsf{sk}_i\}_{i\in n})}$

for $k \in [n]$ **do**

  $(\mathsf{st}_k, \mathsf{msg}_{1,k}) \leftarrow \mathsf{Sign}_1(\{\mathsf{pk}_i\}_{i\in[n]}, \mu, \mathsf{sk}_k)$

for $k \in [n]$ **do**

  $\mathsf{msg}_{2,k} \leftarrow \mathsf{Sign}_2(\{\mathsf{pk}_i\}_{i\in[n]}, \mu, \mathsf{sk}_k, \mathsf{st}_{1,k}, \{\mathsf{msg}_{1,i}\}_{i\in[n]})$

return $\sigma \leftarrow \mathsf{Combine}(\mathcal{L}, \mu, \{\mathsf{msg}_{1,i}\}_{i\in[n]}, \{\mathsf{msg}_{2,i}\}_{i\in[n]})$

Figure 2: Procedure MSJointSign.

4 Two-Round Multi-Signatures

In this section, we present our two-round multi-signature scheme with key aggregation Earpick-MS and then give a tight security proof for it under the DDH assumption. We start by giving the syntax and security definition of two-round multi-signature schemes with key aggregation following [BW25].

4.1 Formal Definitions

Syntax. In the following, we define two-round multi-signature schemes with key aggregation. Each signer's key pair is individually generated by the key generation algorithm KGen. A deterministic, non-interactive key aggregation algorithm KAgg can aggregate a list of individual public keys into a single aggregated key, such that the multi-signature verification algorithm Vf can take only the aggregated public key rather than the whole public key list as input. The two-round signing protocol consists of three algorithms: each signer runs Sign_1 and Sign_2 to generate its first and second-round protocol messages to multicast (i.e., send to all signers), respectively. Finally, the algorithm Combine generates the final multi-signature from the whole protocol transcript, which is deterministic and can be executed by any designated user without any secret key and state. And we use the procedure MSJointSign in Fig. 2 to describe an honest execution of the signing protocol and also to define the completeness property.

We remark that the protocol is independent of the indexing of signers, public keys, and protocol messages. That is, the signers do not need to agree on an order among them, as any order will yield the same output. Without loss of generality, we assume that each user indexes itself by 1.

Definition 3 (Two-Round Multi-Signatures with Key Aggregation)**.** A two-round multi-signature scheme MS with key aggregation consists of algorithms with syntax defined as follows:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: The setup algorithm takes the security parameter 1^λ as input and outputs the public parameter pp. All algorithms related to MS implicitly take pp as input.
- $\mathsf{KGen}() \to (\mathsf{pk}, \mathsf{sk})$: The key generation algorithm outputs a secret key sk and a public key pk.
- $\mathsf{KAgg}(\mathcal{L}) \to \mathsf{apk}$: The deterministic key aggregation algorithm takes a set of public keys $\mathcal{L} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ as input and outputs an aggregated public key apk.
- The signing protocol consists of three algorithms:
  - $\mathsf{Sign}_1(\mathcal{L}, \mu, \mathsf{sk}_1) \to (\mathsf{st}_1, \mathsf{msg}_{1,1})$: The first-round signing algorithm takes a set of public keys $\mathcal{L}$, a message μ, and a secret key sk_1 as inputs and outputs a state st_1 and a first-round protocol message $\mathsf{msg}_{1,1}$.
  - $\mathsf{Sign}_2(\mathcal{L}, \mu, \mathsf{sk}_1, \mathsf{st}_1, \mathcal{M}_1) \to \mathsf{msg}_{2,1}$: The second-round signing algorithm takes a set of public keys $\mathcal{L}$, a message μ, and a secret key sk_1, a state st_1, and a set of first-round protocol messages $\mathcal{M}_1$ as inputs and outputs a second-round protocol message $\mathsf{msg}_{2,1}$.
  - $\mathsf{Combine}(\mathcal{L}, \mu, \mathcal{M}_1, \mathcal{M}_2) \to \sigma$: The combining algorithm takes a set of public key $\mathcal{L}$, a message μ, a set of first-round protocol messages $\mathcal{M}_1$ and a set of second-round protocol messages $\mathcal{M}_2$ as inputs and outputs a multi-signature σ.

$\underline{\mathsf{MSUF}^{\mathsf{A}}_{\mathsf{MS}}(1^\lambda)}$
$\mathcal{Q} := \emptyset$
$\mathsf{ctr} := 0$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{sk}_*, \mathsf{pk}_*) \leftarrow \mathsf{KGen}()$
$(\mathcal{L}_*, \mu_*, \sigma_*) \leftarrow \mathsf{A}^{\textsc{Sign}_1, \textsc{Sign}_2}(\mathsf{pk}_*)$
return $[\![((\mathcal{L}_*, \mu_*) \notin \mathcal{Q}) \wedge (\mathsf{pk}_* \in \mathcal{L}_*) \wedge (\mathsf{Vf}(\mathsf{KAgg}(\mathcal{L}_*), \mu_*, \sigma_*) = 1)]\!]$

$\underline{\textsc{Sign}_1(\mathcal{L}, \mu)}$
if $\mathcal{L}[1] \neq \mathsf{pk}_*$ **then**
  return $\perp$
$\mathsf{ctr} := \mathsf{ctr} + 1$
$\mathsf{sid} := \mathsf{ctr}$
$\mathcal{Q} := \mathcal{Q} \cup (\mathcal{L}, \mu)$
$(\mathcal{L}_{\mathsf{sid}}, \mu_{\mathsf{sid}}) := (\mathcal{L}, \mu)$
$(\mathsf{st}_{\mathsf{sid}}, \mathsf{msg}_{1,\mathsf{sid}}) \leftarrow \mathsf{Sign}_1(\mathcal{L}, \mu, \mathsf{sk}_*)$
$\mathsf{round}_{\mathsf{sid}} := 1$
return msg_1

$\underline{\textsc{Sign}_2(\mathsf{sid}, \mathcal{M}_1)}$
if $(\mathsf{round}_{\mathsf{sid}} \neq 1) \vee (\mathcal{M}_1[1] \neq \mathsf{msg}_{1,\mathsf{sid}})$ **then**
  return $\perp$
$\mathsf{msg}_2 \leftarrow \mathsf{Sign}_2(\mathcal{L}_{\mathsf{sid}}, \mu_{\mathsf{sid}}, \mathsf{sk}_*, \mathsf{st}_{\mathsf{sid}}, \mathcal{M}_1)$
$\mathsf{round}_{\mathsf{sid}} := 2$
return msg_2

Figure 3: The MSUF security game.

- $\mathsf{Vf}(\mathsf{apk}, \mu, \sigma) \to 0/1$: The deterministic verification algorithm takes an aggregated public key apk, a message μ, and a multi-signature σ as inputs and outputs 0 or 1.

For completeness, we require that for every λ, every $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, every n, every $(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_n, \mathsf{sk}_n) \in \mathsf{KGen}()$, and every message μ, we have

$$\Pr[\mathsf{Vf}(\mathsf{apk}, \mu, \sigma) = 1 : \sigma \leftarrow \mathsf{MSJointSign}(\{\mathsf{pk}_i\}_{i\in[n]}, \mu, \{\mathsf{sk}_i\}_{i\in[n]})] = 1,$$

where the procedure MSJointSign, modeling an honest execution of MS, is defined in Fig. 2.

Security. We define the security of two-round multi-signature schemes with key aggregation via the security game MSUF described in Fig. 3. We now give an informal description of the security game. The adversary A is given the public parameters pp and a target public key pk_*. Its goal is to forge a multi-signature on a message μ_* under a public key list $\mathcal{L}_*$ that contains pk_*, both of its own choice. A can concurrently open multiple signing sessions with an honest signer possessing pk_*. In each session, A chooses the message μ to sign, and it plays the part of all other signers, with public keys list $\mathcal{L}$ of its choices. This is formalized by two signing oracles $\textsc{Sign}_1$, $\textsc{Sign}_2$, where we use a session identifier sid to distinguish the concurrent sessions. For the forgery $(\mathcal{L}_*, \mu_*, \sigma_*)$ to be non-trivial, it is required that no signing session with $(\mathcal{L}_*, \mu_*)$ has been started. Notably, there can be finished session under $\mathcal{L}_*$ or on μ_*, but they should not appear together. Finally, we note that there is no oracle for Combine, which would compute the multi-signature from individual signatures. This is without loss of generality, since the algorithm does not make use of any secret state and can be publicly run using the transcript output from Sign_1 and Sign_2.

Definition 4 (Security of Two-Round Multi-Signature Schemes with Key Aggregation)**.** A two-round multi-signature scheme MS with key aggregation is $(q_{\mathrm{s}}, \tau, \varepsilon)$-secure if for every λ, every adversary A that makes at most $q_{\mathrm{s}} = q_{\mathrm{s}}(\lambda)$ queries to $\textsc{Sign}_1$ and is of running time at most $\tau = \tau(\lambda)$, $\Pr[\mathsf{MSUF}^{\mathsf{A}}_{\mathsf{MS}}(1^\lambda) = 1] \leq \varepsilon = \varepsilon(\lambda)$, where the security game MSUF is defined in Fig. 3.

4.2 Our Scheme

We formally define our two-round multi-signature scheme Earpick-MS as pseudocode in Fig. 4 and give a verbal description next.

Setup and Key Generation. The global parameters of the scheme consist of group parameters $(\mathbb{G}, p, g)$ generated by GrGen and a uniformly random group element $h \in \mathbb{G}$. The key generation algorithm KGen uniformly samples $x \leftarrow \mathbb{Z}_p$. The public key is computed as $(X, Y) := (g, h)^x \in \mathbb{G}^2$, and the secret key is $x \in \mathbb{Z}_p$.

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

$(\mathbb{G}, p, g) \leftarrow \mathsf{GrGen}(1^\lambda)$

$h \leftarrow \mathbb{G}$

return $(\mathbb{G}, p, g, h)$

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

$x \leftarrow \mathbb{Z}_p$

$(X, Y) := (g, h)^x$

return $((X, Y), x)$

$\underline{\mathsf{KAgg}(\mathcal{L})}$

$\{(X_i, Y_i)\}_{i\in[n]} := \mathcal{L}$

for $i \in [n]$ **do**

  $a_i := \mathsf{H}_a(\mathcal{L}, X_i, Y_i)$

$(\tilde{X}, \tilde{Y}) := \prod_{i=1}^n (X_i, Y_i)^{a_i}$

return $(\tilde{X}, \tilde{Y})$

$\underline{\mathsf{Sign}_1(\mathcal{L}, \mu, \mathsf{sk}_1)}$

$\{(X_i, Y_i)\}_{i\in[n]} := \mathcal{L}$

$x_1 := \mathsf{sk}_1$

$\mathsf{apk} := \mathsf{KAgg}(\mathcal{L})$

$D := \mathsf{H}_D(\mathcal{L}, \mu)$

$Z_1 := D^{x_1}$

$\pi_1 := \mathsf{DLEQ.NIP}^{\mathsf{H}}((g, D, X_1, Z_1), x_1)$

$(u_0, v_0) := \mathsf{H}_{uv}(0, \mathsf{apk}, \mu)$

$(u_1, v_1) := \mathsf{H}_{uv}(1, \mathsf{apk}, \mu)$

$r_{1,0},\ s_{1,0},\ r_{1,1},\ s_{1,1} \leftarrow \mathbb{Z}_p$

$(R_{1,0}, S_{1,0}) := (g, h)^{r_{1,0}}(u_0, v_0)^{s_{1,0}}$

$(R_{1,1}, S_{1,1}) := (g, h)^{r_{1,1}}(u_1, v_1)^{s_{1,1}}$

$\mathsf{st}_1 := (r_{1,0}, s_{1,0}, r_{1,1}, s_{1,1})$

$\mathsf{msg}_1 := (Z_1, \pi_1, R_{1,0}, S_{1,0}, R_{1,1}, S_{1,1})$

$\underline{\mathsf{Sign}_2(\mathcal{L}, \mu, \mathsf{sk}_1, \mathsf{st}_1, \mathcal{M})}$

$\{(X_i, Y_i)\}_{i\in[n]} := \mathcal{L}$

$x_1 := \mathsf{sk}_1$

$(r_{1,0}, s_{1,0}, r_{1,1}, s_{1,1}) := \mathsf{st}_1$

$\{\mathsf{msg}_i\}_{i\in[n]} := \mathcal{M}$

$\mathsf{apk} := \mathsf{KAgg}(\mathcal{L})$

$D := \mathsf{H}_D(\mathcal{L}, \mu)$

for $i \in [n]$ **do**

  $(Z_i, \pi_i, R_{i,0}, S_{i,0}, R_{i,1}, S_{i,1}) := \mathsf{msg}_i$

  if $\mathsf{DLEQ.NIV}^{\mathsf{H}}((g, D, X_i, Z_i), \pi_i) = 0$ **then**

    return $\perp$

$\tilde{Z} := \prod_{i=1}^n Z_i$

$b := \mathsf{H}_b(\tilde{Z})$

$(\tilde{R}, \tilde{S}) := \prod_{i=1}^n (R_{i,b}, S_{i,b})$

$c := \mathsf{H}_c(\mathsf{apk}, \mu, \tilde{R}, \tilde{S})$

$a_1 := \mathsf{H}_a(\mathcal{L}, X_1, Y_1)$

$z_1 := r_{1,b} + a_1 c x_1$

$y_1 := s_{1,b}$

return (z_1, y_1)

$\underline{\mathsf{Combine}(\mathcal{L}, \mu, \mathcal{M}, \mathcal{M}')}$

$\{\mathsf{msg}_i\}_{i\in[n]} := \mathcal{M}$

$\{\mathsf{msg}'_i\}_{i\in[n]} := \mathcal{M}'$

for $i \in [n]$ **do**

  $(Z_i, \pi_i, R_{i,0}, S_{i,0}, R_{i,1}, S_{i,1}) := \mathsf{msg}_i$

  $(z_i, y_i) := \mathsf{msg}'_i$

$\tilde{Z} := \prod_{i=1}^n Z_i$

$b := \mathsf{H}_b(\tilde{Z})$

$(\tilde{R}, \tilde{S}) := \prod_{i=1}^n (R_{i,b}, S_{i,b})$

$c := \mathsf{H}_c(\mathsf{apk}, \mu, \tilde{R}, \tilde{S})$

$(\tilde{z}, \tilde{y}) := \sum_{i=1}^n (z_i, y_i)$

return $(b, c, \tilde{z}, \tilde{y})$

$\underline{\mathsf{Vf}(\mathsf{apk}, \mu, \sigma)}$

$(\tilde{X}, \tilde{Y}) := \mathsf{apk}$

$(b, c, \tilde{z}, \tilde{y}) := \sigma$

$(u, v) := \mathsf{H}_{uv}(b, \mathsf{apk}, \mu)$

$(\tilde{R}, \tilde{S}) := (g, h)^{\tilde{z}}(u, v)^{\tilde{y}}(\tilde{X}, \tilde{Y})^{-c}$

return $[\![c = \mathsf{H}_c(\mathsf{apk}, \mu, \tilde{R}, \tilde{S})]\!]$

Figure 4: Our two-round multi-signature scheme with key aggregation Earpick-MS.

Key Aggregation. The key aggregation algorithm KAgg aggregates a list of public keys $\mathcal{L} := \{(X_i, Y_i)\}_{i\in[n]}$ into a single aggregated public key. It makes use of a hash function H_a to compute $a_i := \mathsf{H}_a(\mathcal{L}, X_i, Y_i) \in \mathbb{Z}_p$ for all $i \in [n]$. The aggregated public key is then $(\tilde{X}, \tilde{Y}) := \prod_{i=1}^n (X_i, Y_i)^{a_i}$.

Signing Protocol. Suppose n signers with public keys $\mathcal{L} := \{(X_i, Y_i)\}_{i\in[n]}$ want to sign a message μ. We describe the signing protocol from the perspective of the first signer with secret key $\mathsf{sk}_1 := x_1$ and public key $\mathsf{pk}_1 := (X_1, Y_1)$. Let $\mathsf{apk} := \mathsf{KAgg}(\mathcal{L})$ be the aggregated public key as defined above. Further, signers make use of a hash function H_{uv} to get pairs of group elements, H_D to get single group elements, H_b to get single bits, and H_c to get single field elements.

1. *Commitment Phase.* The signer derives a group element $D := \mathsf{H}_D(\mathcal{L}, \mu) \in \mathbb{G}$, and computes $Z_1 := D^{x_1}$ along with a proof of discrete logarithm equality π_1 for statement (g, D, X_1, Z_1).

Next, it derives commitment keys

$$(u_0, v_0) := \mathsf{H}_{uv}(0, \mathsf{apk}, \mu) \in \mathbb{G}^2, \quad (u_1, v_1) := \mathsf{H}_{uv}(1, \mathsf{apk}, \mu) \in \mathbb{G}^2.$$

Then, it samples elements $r_{1,0}, s_{1,0}, r_{1,1}, s_{1,1} \leftarrow \mathbb{Z}_p$ and computes the commitments

$$(R_{1,0}, S_{1,0}) := (g, h)^{r_{1,0}}(u_0, v_0)^{s_{1,0}}, \quad (R_{1,1}, S_{1,1}) := (g, h)^{r_{1,1}}(u_1, v_1)^{s_{1,1}}.$$

It sends $\mathsf{msg}_1 := (Z_1, \pi_1, R_{1,0}, S_{1,0}, R_{1,1}, S_{1,1})$ as its protocol message to the other signers.

2. *Response Phase.* Upon receiving the protocol messages $\{(Z_i, \pi_i, R_{i,0}, S_{i,0}, R_{i,1}, S_{i,1})\}_{i\in[n]}$ from all signers, the signer checks that the proofs of discrete logarithm equality π_i verify with respect to statement (g, D, X_i, Z_i) for all $i \in [n]$. If one of these proofs does not verify, the signer aborts. Otherwise, it aggregates all elements $\{Z_i\}_{i\in[n]}$ into $\tilde{Z} := \prod_{i=1}^{n} Z_i$ and derives a bit $b := \mathsf{H}_b(\tilde{Z})$. Next, it aggregates all b-side commitments into $(\tilde{R}, \tilde{S}) := \prod_{i=1}^{n}(R_{i,b}, S_{i,b})$, and computes a hash challenge as $c := \mathsf{H}_c(\mathsf{apk}, \mu, \tilde{R}, \tilde{S}) \in \mathbb{Z}_p$. It then computes $z_1 := r_{1,b} + a_1 c x_1 \in \mathbb{Z}_p$ where $a_1 := \mathsf{H}_a(\mathcal{L}, X_1, Y_1) \in \mathbb{Z}_p$, and sends $\mathsf{msg}_2 := (z_1, y_1) := (z_1, s_{1,b})$ as its protocol message to the other signers.

3. *Combine Phase.* The signer combines the responses as $(\tilde{z}, \tilde{y}) := \sum_{i=1}^{n}(z_i, y_i)$. The final signature is then $(b, c, \tilde{z}, \tilde{y})$, where b and c are defined as in the response phase above.

Verification. The verifier gets the (aggregated) public key $(\tilde{X}, \tilde{Y})$, the signature $(b, c, \tilde{z}, \tilde{y})$, and the message μ. It computes $(u, v) := \mathsf{H}_{uv}(b, \mathsf{apk}, \mu)$ and recovers the commitments as $(\tilde{R}, \tilde{S}) := (g, h)^{\tilde{z}}(u, v)^{\tilde{y}}(\tilde{X}, \tilde{Y})^{-c}$. It accepts the signature if and only if $c = \mathsf{H}_c(\mathsf{apk}, \mu, \tilde{R}, \tilde{S})$.

4.3 Security Analysis

We give a tight security proof for Earpick-MS under the DDH assumption. In our proof, we also make use of the CDH assumption, which is tightly implied by the DDH assumption. Further, we note that correctness of our scheme is straightforward.

Theorem 2. *If the* DDH *problem is* $(\tau_{\mathrm{ddh}}, \varepsilon_{\mathrm{ddh}})$*-hard and the* CDH *problem is* $(\tau_{\mathrm{cdh}}, \varepsilon_{\mathrm{cdh}})$*-hard for the group generation algorithm* GrGen*, then Earpick-MS is a* $(q_{\mathrm{s}}, \tau_{\mathrm{uf}}, \varepsilon_{\mathrm{uf}})$*-secure multi-signature scheme in the random oracle model (ROM) against any adversary that makes at most* q_{h} *total hash queries, where essentially* $\tau_{\mathrm{ddh}} \approx \tau_{\mathrm{uf}}$*,* $\tau_{\mathrm{cdh}} \approx \tau_{\mathrm{uf}}$*, and*

$$\varepsilon_{\mathrm{uf}} \leq 4\varepsilon_{\mathrm{ddh}} + \varepsilon_{\mathrm{cdh}} + \frac{2q_h^2 + 2q_s q_h + 7q_h + 7}{p}.$$

Proof. We provide the proof as a sequence of hybrid games, where any two subsequent games are indistinguishable. Further, we will argue that winning the final game is a statistically hard problem, which concludes the proof. For this, we assume an adversary A that breaks the $(q_{\mathrm{s}}, \tau_{\mathrm{uf}}, \varepsilon_{\mathrm{uf}})$-security of Earpick-MS.

Game $\mathbf{G}_0$ (Real Game): This is the real security game $\mathsf{MSUF}^{\mathsf{A}}_{\mathsf{MS}}$ for multi-signatures as defined in Fig. 3. We briefly recall the game to fix some notation. First, the game samples public parameters $\mathsf{pp} := (\mathbb{G}, p, g, h) \leftarrow \mathsf{Setup}(1^\lambda)$, where $h \leftarrow \mathbb{G}$ is uniformly random. The game also samples $x \leftarrow \mathbb{Z}_p$ for a secret key $\mathsf{sk} := x$ and public key $\mathsf{pk} := (X, Y) := (g, h)^x$. Then, the game runs the adversary A on input $(\mathsf{pp}, \mathsf{pk})$ with access to signing oracles and random oracles defined as follows.

- *Signing Oracles* $\textsc{Sign}_1, \textsc{Sign}_2$: These oracles simulate the signing algorithms $\mathsf{Sign}_1, \mathsf{Sign}_2$ for the secret key sk, respectively. Whenever A calls the oracle $\textsc{Sign}_1(\mathcal{L}, \mu))$ to start a new session for message μ and list of public keys $\mathcal{L} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$, the game adds $(\mathcal{L}, \mu)$ to a list $\mathcal{Q}$. Without loss of generality, we always assume that $\mathsf{pk} = \mathsf{pk}_1$. Further, the game keeps track of the sessions using an identifier sid that starts from 1.

- *Random Oracles* $\mathsf{H}, \mathsf{H}_a, \mathsf{H}_b, \mathsf{H}_c, \mathsf{H}_D, \mathsf{H}_{uv}$: The random oracles are simulated honestly via standard lazy sampling. To this end, the game holds internal maps $\mathsf{H}[\cdot], \ldots, \mathsf{H}_{uv}[\cdot]$ that assign the inputs of the respective random oracles to their outputs.

At the end of the game, the adversary outputs a forgery $(\mathcal{L}_*, \mu_*, \sigma_*)$ and the game outputs 1 if and only if: the pair of public key list and message was never queried before, $(\mathcal{L}_*, \mu_*) \notin \mathcal{Q}$, the honest public key is included in the public key list, $\mathsf{pk} \in \mathcal{L}_*$, and the signature verifies for the message and public key list, $\mathsf{Vf}(\mathsf{KAgg}(\mathcal{L}_*), \mu_*, \sigma_*) = 1$. Henceforth, we write $\mathsf{apk}_* := \mathsf{KAgg}(\mathcal{L}_*)$. Clearly, we have

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

Game $\mathbf{G}_1$ (Simulate NIZK Proofs): In this game, we change how the non-interactive zero-knowledge proofs of discrete logarithm equality π_1 are computed by the first-round signing oracle $\textsc{Sign}_1$. Concretely, we switch from honestly generated proofs π_1 to simulated proofs $\tilde{\pi}_1$ using the special honest-verifier zero-knowledge (SHVZK) simulator Sim. Clearly, the games $\mathbf{G}_1$ and $\mathbf{G}_0$ are indistinguishable until Sim generates a commitment $\mathsf{cmt} \in \mathbb{G}^2$ (for the simulated proof $\tilde{\pi}_1$) that has already been queried to the random oracle H (underlying the non-interactive proof system DLEQ). However, since at least one component of the commitment is uniformly random over the group $\mathbb{G}$ of size p and the commitment is hidden from A (before we output the NIZK proof), the probability that this bad event happens is at most q_{h}/p per simulated proof. By a union bound over all signing queries, we thus get

$$\left|\Pr\left[\mathbf{G}_1 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_0 \Rightarrow 1\right]\right| \leq q_{\mathrm{s}} q_{\mathrm{h}}/p.$$

Game $\mathbf{G}_2$ (Program H_{uv} Correlated): In this game, we change how we program the random oracle H_{uv}. Recall that H_{uv} is the random oracle that signers use in the first round to derive the commitment keys (u_b, v_b) for both sides $b \in \{0,1\}$. Namely, for every new query $\mathsf{H}_{uv}(b, \mathsf{apk}, \mu)$ with $b \in \{0,1\}$, we sample a uniformly random $\alpha \leftarrow \mathbb{Z}_p$ and then program the random oracle as

$$\mathsf{H}_{uv}(b, \mathsf{apk}, \mu) := (u, v) := (X, Y)^{\alpha}.$$

We bound the distinguishing advantage between games $\mathbf{G}_2$ and $\mathbf{G}_1$ via a straightforward reduction B to DDH. To this end, B gets a tuple $\mathcal{D} := (g, h, \hat{g}, \hat{h}) \in \mathbb{G}^4$ as input and then simulates game $\mathbf{G}_1$ for A with the following change. For every new query $\mathsf{H}_{uv}(b, \mathsf{apk}, \mu)$ with $b \in \{0,1\}$, it samples uniformly random $e, f \leftarrow \mathbb{Z}_p$ and then programs the random oracle as

$$\mathsf{H}_{uv}(b, \mathsf{apk}, \mu) := (g, h)^e \cdot (\hat{g}, \hat{h})^f.$$

In the end, B outputs whatever the game outputs. Clearly, B perfectly simulates game $\mathbf{G}_1$ for A if $\mathcal{D}$ is *not* a Diffie-Hellman (DH) tuple. On the other hand, if $\mathcal{D}$ is indeed a DH tuple, then B perfectly simulates game $\mathbf{G}_2$ for A, unless $\mathsf{sk} = x = 0$ is trivial. For the second claim, note that the honest public key (X, Y) itself forms a DH tuple with base (g, h), and therefore, conditioned on $x \neq 0$,

$$(X, Y)^{\alpha} = (g, h)^{\alpha x} \quad \text{and} \quad (g, h)^e \cdot (\hat{g}, \hat{h})^f = (g, h)^{e + \hat{x} f}$$

are identically distributed, where we defined $\hat{x}$ via $(\hat{g}, \hat{h}) = (g, h)^{\hat{x}}$. Finally, since the secret key x is sampled uniformly at random from $\mathbb{Z}_p$, the probability that $x = 0$ is at most $1/p$. Thus, we get

$$\left|\Pr\left[\mathbf{G}_2 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1 \Rightarrow 1\right]\right| \leq \varepsilon_{\mathrm{ddh}} + 1/p.$$

Game $\mathbf{G}_3$ (Program H_{uv} via Dual Key): In this game, we change how we sample α to program the random oracle H_{uv}. Recall that until now, for every new query $\mathsf{H}_{uv}(b, \mathsf{apk}, \mu)$ with $b \in \{0,1\}$, we sample a uniformly random $\alpha \leftarrow \mathbb{Z}_p$ and then program the random oracle as $\mathsf{H}_{uv}(b, \mathsf{apk}, \mu) := (X, Y)^{\alpha}$. From now on, we sample a uniformly random $w \leftarrow \mathbb{Z}_p^*$ and then program the random oracle as

$$\mathsf{H}_{uv}(b, \mathsf{apk}, \mu) := (X, Y)^{1/w}.$$

We also store w in an (initially empty) list DK as $\mathsf{DK}[b, \mathsf{apk}, \mu] := w$. From now on, we will refer to w as the *dual key*. Clearly, the games $\mathbf{G}_3$ and $\mathbf{G}_2$ are indistinguishable until we sample an $\alpha \leftarrow 0$ in game $\mathbf{G}_2$. However, since each α is sampled uniformly random from $\mathbb{Z}_p$, the probability that this bad event happens is at most $1/p$ per sampling. Thus, by a union bound over all random oracle queries, we get

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq q_{\mathrm{h}}/p.$$

Game $\mathbf{G}_4$ (Signing with Dual Key): In this game, we change how to simulate the second-round signing oracle $\textsc{Sign}_2$ (i.e., the response phase). Recall that until now, we simulate the second-round response message (z_1, y_1) by honestly computing $z_1 := r_{1,b} + a_1 cx$ and $y_1 := s_{1,b}$ *using* the secret key x. We now change this as follows. Upon a signing request $\textsc{Sign}_2$ with session identifier sid, we first recover the corresponding tuple $(b, \mathcal{L}, \mu)$ (note that these values are already defined from the first round). Then, we retrieve the respective dual key $w = \mathsf{DK}(b, \mathsf{apk}, \mu)$ and compute $\tilde{y}_1 := s_{1,b} + a_1 cw$. Finally, we return the tuple $(\tilde{z}_1, \tilde{y}_1) := (r_{1,b}, \tilde{y}_1)$ as second-round response message. In the following, we will show that the adversary A's view is identically distributed in both games $\mathbf{G}_4$ and $\mathbf{G}_3$.

To this end, first observe that the tuple $(\tilde{z}_1, \tilde{y}_1)$ indeed satisfies the relation the adversary expects, namely:

$$\begin{aligned}(R_{1,b}, S_{1,b}) &= (g,h)^{\tilde{z}_1 - a_1 cx}(u_b, v_b)^{\tilde{y}_1} = (g,h)^{r_{1,b} - a_1 cx}(u_b, v_b)^{s_{1,b} + a_1 cw} \\ &= (g,h)^{r_{1,b} - a_1 cx}(g,h)^{x/w(s_{1,b} + a_1 cw)} = (g,h)^{r_{1,b} - a_1 cx}(g,h)^{x/w s_{1,b} + a_1 cx} \\ &= (g,h)^{r_{1,b}}(g,h)^{x/w s_{1,b}} = (g,h)^{r_{1,b}}(u_b, v_b)^{s_{1,b}},\end{aligned} \tag{1}$$

where we used $(u_b, v_b) = (X, Y)^{1/w} = (g,h)^{x/w}$ by the changes introduced in game $\mathbf{G}_3$. In particular, the above identity tells us that an unbounded adversary does not learn any additional information by the change in $\textsc{Sign}_2$. To further argue identical distributions of the second-round outputs, consider the map $F : \mathbb{Z}_p \times \mathbb{Z}_p \to \mathbb{Z}_p \times \mathbb{Z}_p$ defined by

$$(\zeta_0, \zeta_1) \mapsto (\zeta_0 - a_1 cx, \zeta_1 + a_1 cw).$$

Then, this map is bijective (for fixed values c, x, w) and the randomness $(r_{1,b}, s_{1,b})$ used in game $\mathbf{G}_3$ is mapped to the randomness $(\tilde{r}_{1,b}, \tilde{s}_{1,b}) := (\tilde{z}_1 - a_1 cx, \tilde{y}_1)$ used in game $\mathbf{G}_4$. The former is true because $w \in \mathbb{Z}_p$ is sampled uniformly random and independent of the secret key x. Since the values c and w are sampled uniformly random and independent across different signing sessions, the above observations (bijection of the function F and bidirectional mapping between randomnesses) imply that the joint distribution of the second-round outputs is identical in both games $\mathbf{G}_4$ and $\mathbf{G}_3$.

For a more formal calculation to argue that $\Pr[\mathbf{G}_4 \Rightarrow 1] = \Pr[\mathbf{G}_3 \Rightarrow 1]$, we refer the interested reader to Appendix B.1. To this end, we use the *H-coefficients technique* from Patarin [Pat09, HT16]. It is a foundational tool in the analysis of cryptographic systems, typically used in symmetric-key cryptography but has also found applications in multi-party signatures [BDLR25]. Importantly, it allows to upper-bound the adversary's distinguishing advantage between two probabilistic games by comparing the probabilities of realizing a specific protocol transcript after fixing the adversary's randomness. In particular, the argument on (information-theoretic) indistinguishability of distributions boils down to a straightforward calculation[4] of probabilities. We conclude with

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

Game $\mathbf{G}_5$ (No Collisions for apk): In this game, we add an abort condition. Namely, the game aborts if there exists a pair of public key lists $\mathcal{L} \neq \mathcal{L}'$ that have been submitted to H_a such that $\mathsf{pk} \in \mathcal{L}, \mathcal{L}'$ and $\mathsf{KAgg}(\mathcal{L}) = \mathsf{KAgg}(\mathcal{L}')$ (i.e., the corresponding aggregated public keys are identical). Intuitively, this says that an aggregated public key represents the set of signers in a unique way. We

[4] This is because all samples are uniformly random. But in general this calculation can indeed be quite involved.

bound the probability of abort in this game via a straightforward analysis. First, we condition on the event that $\mathsf{pk} \neq 1$ (i.e., $x \neq 0$). Now, consider a fixed pair of public key lists $\mathcal{L} \neq \mathcal{L}'$ with the above conditions. Then, for any given (even adversarially chosen) $a_1, (\mathsf{pk}_i, a_i)_{i=2}^{n}, (\mathsf{pk}'_i, a'_i)_{i=2}^{n'}$, there exists at most one value $a'_1 \in \mathbb{Z}_p$ to have a collision

$$a_1\mathsf{pk} + \sum_{i=2}^{n} a_i\mathsf{pk}_i = \mathsf{KAgg}(\mathcal{L}) \stackrel{!}{=} \mathsf{KAgg}(\mathcal{L}') = a'_1\mathsf{pk} + \sum_{i=2}^{n'} a'_i\mathsf{pk}'_i.$$

Thus, the probability to sample that unique $a'_1 \leftarrow \mathbb{Z}_p$ is bounded by $1/p$. Using this observation, the probability of having a collision for one pair of public key lists $\mathcal{L} \neq \mathcal{L}'$ is at most $1/p$. Therefore, by a union bound over all such pairs, the probability of abort in this game is at most ${q_\mathrm{h}}^2/p$. Further, the probability that $\mathsf{pk} = 1$ is $1/p$, since $x \leftarrow \mathbb{Z}_p$ is sampled uniformly random. Thus, we get

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq {q_\mathrm{h}}^2/p + 1/p.$$

Game $\mathbf{G}_6$ (Flip Coin for (apk, μ)): In this game, we introduce an initially empty list Coin and change the game as follows. For each query of the form $\mathsf{H}_{uv}(\cdot, \mathsf{apk}, \mu)$ with a new (apk, μ), we sample a uniformly random bit $b \leftarrow \{0,1\}$ (called *coin*) and set $\mathsf{Coin}[\mathsf{apk}, \mu] := b$. It is clear that A's view is unaffected by this change. Therefore, we have

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

Game $\mathbf{G}_7$ (No Collisions for H_D): In this game, we rule out collisions for the random oracle H_D. Specifically, the game aborts if there are tuples $(\mathcal{L}, \mu) \neq (\mathcal{L}', \mu')$ such that $\mathsf{H}_D(\mathcal{L}, \mu) = \mathsf{H}_D(\mathcal{L}', \mu')$. Using standard collision probability analysis, the probability of abort in this game can be bounded by ${q_\mathrm{h}}^2/p$. Thus, we get

$$|\Pr[\mathbf{G}_7 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq {q_\mathrm{h}}^2/p.$$

Game $\mathbf{G}_8$ (Output Random Z_1): In this game, we change the first-round signing oracle $\textsc{Sign}_1$ (i.e., commitment phase). Recall that until now, upon a signing request $\textsc{Sign}_1$ on input $(\mathcal{L}, \mu)$, we first derive a group element $D := \mathsf{H}_D(\mathcal{L}, \mu)$ and then compute $Z_1 := D^x$ (honestly using the secret key x) along with a (simulated) NIZK proof of discrete logarithm equality π_1 for the statement (g, D, X, Z_1) where $\mathsf{pk} = (X, Y)$. We change this as follows. First, we introduce an initially empty list Z that is updated whenever $\mathsf{H}_D[\cdot]$ is sampled. Concretely, whenever $\mathsf{H}_D[\mathcal{L}, \mu] \leftarrow \mathbb{G}$ is newly defined, we also define $\mathsf{Z}[\mathcal{L}, \mu] \leftarrow \mathbb{G}$ uniformly at random. Second, we let $\textsc{Sign}_1$ output (the uniformly random) $Z_1 := \mathsf{Z}[\mathcal{L}, \mu]$ instead of (the correct) D^x.

We can bound the distinguishing advantage between games $\mathbf{G}_8$ and $\mathbf{G}_7$ via a straightforward reduction B to DDH. To this end, B gets a tuple $\mathcal{D} := (g, \underline{h}, \hat{g}, \hat{h}) \in \mathbb{G}^4$ as input[5] and then simulates game $\mathbf{G}_7$ for A with the following changes. First, B generates the public parameter $h \in \mathbb{G}$ with known discrete logarithm by uniformly sampling $\alpha_h \leftarrow \mathbb{Z}_p$ and setting $h := g^{\alpha_h}$. Then, B generates the public key as $\mathsf{pk} := (\hat{g}, \hat{g}^{\alpha_h})$, which is easily seen to be correctly generated (with $x := \mathsf{dlog}(\hat{g})$, we have $\mathsf{pk} = (g^x, g^{x\alpha_h}) = (g^x, h^x)$). Further, for every new query $\mathsf{H}_D(\mathcal{L}, \mu)$, it samples uniformly random $e, f \leftarrow \mathbb{Z}_p$ and then programs the random oracle as

$$\mathsf{H}_D(\mathcal{L}, \mu) := g^e \cdot \underline{h}^f \in \mathbb{G}.$$

Moreover, upon a signing request $\textsc{Sign}_1$ on input $(\mathcal{L}, \mu)$, it first retrieves the elements $e, f \in \mathbb{Z}_p$ used to program $\mathsf{H}_D(\mathcal{L}, \mu)$, and then computes (and outputs) Z_1 as

$$Z_1 := \hat{g}^e \cdot \hat{h}^f \in \mathbb{G}.$$

[5] Observe the underscore in the second parameter $\underline{h}$ of the instance. This allows us to distinguish it from the public parameter h of the scheme that is given to the adversary A.

In the end, B outputs whatever the game outputs. It is not hard to see that B perfectly simulates game $\mathbf{G}_7$ for A if $\mathcal{D}$ is a DH tuple: With $x := \mathsf{dlog}(\hat{g})$ (also recall that $\mathsf{pk} = (\hat{g}, \hat{g}^{\alpha_h})$), we have

$$Z_1 = g^{xe} \cdot \underline{h}^{xf} = (g^e \cdot \underline{h}^f)^x = \mathsf{H}_D(\mathcal{L}, \mu)^x.$$

On the other hand, if $\mathcal{D}$ is *not* a DH tuple, then B perfectly simulates game $\mathbf{G}_8$ for A. This is clear, since the output Z_1 is then distributed as an uniformly sampled element from $\mathbb{G}$. Thus, we get

$$|\Pr[\mathbf{G}_8 \Rightarrow 1] - \Pr[\mathbf{G}_7 \Rightarrow 1]| \leq \varepsilon_{\mathrm{ddh}}.$$

By this change, observe that we *do not* use the secret key x anymore for signing. That is because we have already switched the NIZK proofs from honest to simulated, the response messages (z_1, y_1) are computed via the dual key w, and the element Z_1 is output as a uniformly random element. In particular, the only place where the secret key is referenced is during the key generation phase.

Game $\mathbf{G}_9$ (Adversary Copies pk): In this game, we add an abort condition. Namely, the game aborts if the adversary A copies the (honest) public key pk and submits an element $Z' \neq \mathsf{Z}[\mathcal{L}, \mu]$ along with a verifying NIZK proof π' as part of its first-round message. Note that the element Z' is then supposed to be computed as $\mathsf{H}_D(\mathcal{L}, \mu)^{\mathsf{sk}}$. In more formal terms: The game aborts if there exists a second-round signing request[6] $\textsc{Sign}_2(\mathcal{L}, \mu, \mathcal{M}_1)$, parsed as

$$\mathcal{L} := \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \quad \mathcal{M}_1 := \{(Z_i, \pi_i, R_{i,0}, S_{i,0}, R_{i,1}, S_{i,1})\}_{i \in [n]}, \quad D := \mathsf{H}_D(\mathcal{L}, \mu),$$

such that there is an $i \neq 1$ with $\mathsf{pk}_i = \mathsf{pk}$, $\mathsf{DLEQ.NIV}^{\mathsf{H}}((g, D, X_i, Z_i), \pi_i) = 1$, and $Z_i \neq Z_1$ (recall that Z_1 is output by the game as $\mathsf{Z}[\mathcal{L}, \mu]$).

We bound the probability of abort in this game via a reduction B to CDH that succeeds with overwhelming probability. To this end, B gets a tuple $(\hat{g}, \hat{h}) \in \mathbb{G}^2$ as input and then simulates game $\mathbf{G}_8$ for A with the following changes. First, B generates the public parameter $h \in \mathbb{G}$ with known discrete logarithm by uniformly sampling $\alpha_h \leftarrow \mathbb{Z}_p$ and setting $h := g^{\alpha_h}$. Then, B generates the public key as $\mathsf{pk} := (\hat{g}, \hat{g}^{\alpha_h})$, which is easily seen to be correctly generated (with $x := \mathsf{dlog}(\hat{g})$, we have $\mathsf{pk} = (g^x, g^{x\alpha_h}) = (g^x, h^x)$). Further, for every new query $\mathsf{H}_D(\mathcal{L}, \mu)$, it samples a uniformly random $e \leftarrow \mathbb{Z}_p$ and then programs the random oracle as $\mathsf{H}_D(\mathcal{L}, \mu) := \hat{h}^e \in \mathbb{G}$. Moreover, upon a (valid) second-round signing request $\textsc{Sign}_2(\mathcal{L}, \mu, \mathcal{M}_1)$, parsed as

$$\mathcal{L} := \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \quad \mathcal{M}_1 := \{(Z_i, \pi_i, R_{i,0}, S_{i,0}, R_{i,1}, S_{i,1})\}_{i \in [n]}, \quad D := \mathsf{H}_D(\mathcal{L}, \mu),$$

such that there is an $i \neq 1$ with $\mathsf{pk}_i = \mathsf{pk}$, $\mathsf{DLEQ.NIV}^{\mathsf{H}}((g, D, X_i, Z_i), \pi_i) = 1$, and $Z_i \neq \mathsf{Z}[\mathcal{L}, \mu]$, the reduction B terminates and outputs $\hat{z} := Z_i^{1/e}$ as its CDH solution to $(\hat{g}, \hat{h})$. We analyze its success probability. Intuitively, if the adversary's tuple[7] $(g, X_i, D, Z_i) = (g, X, D, Z_i)$ is *not* a DH tuple, then generating a verifying NIZK proof π_i is a statistically hard problem. In particular, this (statistically hard) case happens with probability at most q_{h}/p. Otherwise, the tuple (g, X, D, Z_i) is indeed a DH tuple and the reduction B succeeds.

More formally, we establish this as follows. First, by the statistical soundness of the proof system DLEQ, except with probability at most q_{h}/p, the tuple $(g, X_i, D, Z_i) = (g, X, D, Z_i)$ is a DH tuple. This can be easily observed as follows. If (g, X, D, Z_i) is not a DH tuple (i.e., $\mathsf{dlog}_g(X) \neq \mathsf{dlog}_D(Z_i)$), then for every random oracle input to H (underlying the non-interactive proof system DLEQ), there exists at most one challenge output $\mathsf{chl} \in \mathbb{Z}_p$ (with respective response rsp) that makes the proof π_i verify. Since the challenge chl is sampled uniformly at random from $\mathbb{Z}_p$, the probability to sample that unique challenge is at most $1/p$. Thus, by a union bound over all random oracle queries, the

[6] We always assume w.l.o.g. that the signing request is valid in the sense that (i) there is a corresponding first-round signing session and (ii) the adversary does not alter the honest first-round message. Otherwise, the game simply outputs $\perp$, as outlined in our security game for multi-signatures (cf. Fig. 3).

[7] Recall that $\mathsf{pk} = (X, Y)$, the adversary copies the key (i.e., $X_i = X$), and the NIZK proof π_i verifies by assumption.

claim follows: Except with probability at most q_{h}/p, the tuple (g, X, D, Z_i) is a DH tuple. On the other hand, if the reduction B ever samples an $e \leftarrow 0 \in \mathbb{Z}_p$ to program the random oracle H_D, then it might fail to compute its CDH solution as $\hat{z} := Z_i^{1/e}$. However, it is clear that this happens with probability at most q_{h}/p (the probability is $1/p$ per sample, and there are at most q_{h} samples). As a result, except with probability $2q_{\mathrm{h}}/p$, B outputs the correct CDH solution $\hat{z}$ to the instance $(\hat{g}, \hat{h})$. In summary, we therefore get

$$|\Pr[\mathbf{G}_9 \Rightarrow 1] - \Pr[\mathbf{G}_8 \Rightarrow 1]| \leq \varepsilon_{\mathrm{cdh}} + q_{\mathrm{h}}/p + q_{\mathrm{h}}/p.$$

Game $\mathbf{G}_{10}$ (Program H_D with Trapdoor): In this game, we change how we program the random oracle H_D. Namely, for every new query $\mathsf{H}_D(\mathcal{L}, \mu)$, we sample $d \leftarrow \mathbb{Z}_p$ uniformly at random and then program the random oracle as $\mathsf{H}_D(\mathcal{L}, \mu) := g^d$. Clearly, this change does not change the distribution of random oracle H_D outputs. Thus, the adversary A's view is identically distributed as before. Thus, we get

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

Game $\mathbf{G}_{11}$ (No Wrong Z_i for $\mathsf{pk}_i \neq \mathsf{pk}$): In this game, we add an abort condition. Namely, the game aborts if the adversary A submits an element $Z_i \neq \mathsf{pk}_i^d$ along with a verifying NIZK proof π_i such that $\mathsf{pk}_i \neq \mathsf{pk}$ as part of its first-round message. Note that the element Z_i is supposed to be computed as $\mathsf{H}_D(\mathcal{L}, \mu)^{\mathsf{sk}_i} = \mathsf{pk}_i^d$, where $d \in \mathbb{Z}_p$ is the known discrete logarithm of $\mathsf{H}_D(\mathcal{L}, \mu)$ introduced in the previous game. In more formal terms: The game aborts if there exists a second-round signing request $\textsc{Sign}_2(\mathcal{L}, \mu, \mathcal{M}_1)$, parsed as $\mathcal{L} := \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$,

$$\mathcal{M}_1 := \{(Z_i, \pi_i, R_{i,0}, S_{i,0}, R_{i,1}, S_{i,1})\}_{i \in [n]}, \quad D := \mathsf{H}_D(\mathcal{L}, \mu) = g^d,$$

such that there is an $i \neq 1$ with $\mathsf{pk}_i \neq \mathsf{pk}$, $\mathsf{DLEQ.NIV}^{\mathsf{H}}((g, D, X_i, Z_i), \pi_i) = 1$, and $Z_i \neq \mathsf{pk}_i^d$.

We can bound the probability of abort in this game via the statistical soundness of the proof system DLEQ. Namely, for every random oracle input to H (underlying the non-interactive proof system), there exists at most one challenge output $\mathsf{chl} \in \mathbb{Z}_p$ (with respective response rsp) that makes the proof π_i verify. Since the challenge chl is sampled uniformly at random from $\mathbb{Z}_p$, the probability to sample that unique challenge is at most $1/p$. Thus, by a union bound over all random oracle queries, we get

$$|\Pr[\mathbf{G}_{11} \Rightarrow 1] - \Pr[\mathbf{G}_{10} \Rightarrow 1]| \leq q_{\mathrm{h}}/p.$$

Game $\mathbf{G}_{12}$ (Program H_b via Coin): In this game, we change how we program the random oracle H_b. Recall that H_b is the random oracle signers use in the second round to derive the bit $b \in \{0,1\}^*$ that decides which signing-side to conclude. Further, by the changes introduced previously, recall that whenever $\mathsf{H}_D[\mathcal{L}, \mu] \leftarrow g^d$ is newly defined, we also define $\mathsf{Z}[\mathcal{L}, \mu] \leftarrow \mathbb{G}$ uniformly at random and later (upon a singing request $\textsc{Sign}_1(\mathcal{L}, \mu)$) output it as Z_1 message instead of (the correct) D^x. From now on, we additionally update H_b as follows (after $\mathsf{Z}[\mathcal{L}, \mu] \leftarrow \mathbb{G}$ is defined). For this purpose, let $m_{\mathcal{L}}(\mathsf{pk}) \in \mathbb{N}$ denote the multiplicity of pk in the public key list $\mathcal{L}$, and let $\mathcal{L}_0 := \mathcal{L} \setminus \{\mathsf{pk}\}$ (i.e., the remaining public keys $\mathsf{pk}_i \neq \mathsf{pk}$). First, the game pre-computes the element

$$\tilde{Z} := Z_1^{m_{\mathcal{L}}(\mathsf{pk})} \cdot \prod_{\mathsf{pk}_i \in \mathcal{L}_0} \mathsf{pk}_i^d \in \mathbb{G},$$

where $d \in \mathbb{Z}_p$ is the known discrete logarithm of $\mathsf{H}_D(\mathcal{L}, \mu)$ used to define it (which was introduced in game $\mathbf{G}_{10}$), and then programs the random oracle as $\mathsf{H}_b(\tilde{Z}) := \mathsf{Coin}(\mathsf{apk}, \mu)$. Clearly, the games $\mathbf{G}_{12}$ and $\mathbf{G}_{11}$ are indistinguishable until we rewrite H_b. However, since each $Z_1 = \mathsf{Z}[\mathcal{L}, \mu]$ is sampled uniformly random from $\mathbb{G}$, the probability that this bad event happens is at most q_{h}/p per sampling. Thus, by a union bound over all signing queries, we get

$$|\Pr[\mathbf{G}_{12} \Rightarrow 1] - \Pr[\mathbf{G}_{11} \Rightarrow 1]| \leq q_{\mathrm{s}} q_{\mathrm{h}}/p.$$

Before we proceed with our analysis, we take a brief moment to summarize what we have achieved so far. First, we do not use the secret key anymore, except in the key generation phase. In relation to this, we use the dual key w for signing (in the second round). And second, we are in full control over the bit b that decides which branch to use for signing in the second round. Specifically, that bit is defined via our internal coin flip Coin. From these points, we observe that we do not need the dual key for the undecided branch anymore and can program H_{uv} for that branch without embedding the public key pk. This is what we do next.

Game $\mathbf{G}_{13}$ (Program H_{uv} for $b \neq \mathsf{Coin}$): In this game, we change how we program the random oracle H_{uv}. Namely, for every new query $\mathsf{H}_{uv}(b, \mathsf{apk}, \mu)$ where $b \neq \mathsf{Coin}(\mathsf{apk}, \mu)$, we sample a uniformly random $\alpha \leftarrow \mathbb{Z}_p$ and then program the random oracle as $\mathsf{H}_{uv}(b, \mathsf{apk}, \mu) := (g, h)^{\alpha}$. Further, α is not stored into the list DK. In particular, the *dual key* w only exists for the $\mathsf{Coin}(\mathsf{apk}, \mu)$-sides of the signing protocol. We claim that this change does not affect the view of A much. In more detail: Since (g, h, X, Y) is a DH tuple, the distributions of random oracle outputs

$$\{(g,h)^{\alpha} \ : \ \alpha \leftarrow \mathbb{Z}_p\}_{\mathbf{G}_{13}} \ \text{ and } \ \{(X,Y)^{1/w} = (g,h)^{x/w} \ : \ w \leftarrow \mathbb{Z}_p^*\}_{\mathbf{G}_{12}}$$

in games $\mathbf{G}_{13}$ and $\mathbf{G}_{12}$ are statistically close, with distance bounded by $q_{\mathrm{h}}/p + 1/p$. This can be seen as follows. Conditioned on neither sample set having the trivial element $1 = (1,1) \in \mathbb{G}^2$, they are indistinguishable. The probability for the first sample set to have the trivial element is at most q_{h}/p (the probability is $1/p$ per sample, and there are at most q_{h} samples), while it is at most $1/p$ for the second sample set (the probability that $x = 0$ is at most $1/p$), and our claim follows. Finally, observe that the dual key is not needed anymore for the undecided branch $b \neq \mathsf{Coin}(\mathsf{apk}, \mu)$. Thus, we get

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

Game $\mathbf{G}_{14}$ (Forgery for $b_* = \mathsf{Coin}$): In this game, we add an abort condition. Namely, the game aborts if $b_* \neq \mathsf{Coin}(\mathsf{apk}_*, \mu_*)$ in the forgery $(\mathcal{L}_*, \mu_*, \sigma_*)$ output by the adversary A at the end of the game, where $\sigma_* := (b_*, c_*, \tilde{z}_*, \tilde{y}_*)$. To bound the probability of abort in this game, we first argue that the bit $\mathsf{Coin}(\mathsf{apk}_*, \mu_*)$ is independent of A's view. For this, observe that the bit $\mathsf{Coin}(\mathsf{apk}, \mu)$ is information-theoretically hidden from A until the game programs $\mathsf{H}_b(\tilde{Z}) := \mathsf{Coin}(\mathsf{apk}, \mu)$ (upon a signing request $\textsc{Sign}_1$). Further, by the collision-resistance of aggregated public keys (see changes in game $\mathbf{G}_5$), we know that $\mathsf{Coin}(\mathsf{apk}, \mu)$ is defined at most once for any tuple $(\mathcal{L}, \mu)$. Therefore, as long as $(\mathcal{L}, \mu)$ has not been submitted as query to $\textsc{Sign}_1$, the bit $\mathsf{Coin}(\mathsf{apk}, \mu)$ remains independent of A's view. In particular, this is also true for the forgery tuple $(\mathcal{L}_*, \mu_*)$. Finally, given that $\mathsf{Coin}(\mathsf{apk}_*, \mu_*)$ is independent of A's view, it is clear that $b_* \neq \mathsf{Coin}(\mathsf{apk}_*, \mu_*)$ happens with probability $1/2$. Therefore, we get

$$\Pr[\mathbf{G}_{14} \Rightarrow 1] \geq (1/2) \cdot \Pr[\mathbf{G}_{13} \Rightarrow 1].$$

In the final game, we switch from an honestly generated key $\mathsf{pk} = (X, Y) = (g^x, h^x)$ to a uniformly random tuple $(X, Y) \leftarrow \mathbb{G}^2$. With overwhelming probability, the tuple (g, h, X, Y) is then *not* a DH tuple, and signing with the aggregated key apk_* for the b_*-side becomes a statistically hard problem.

Game $\mathbf{G}_{15}$ (Switch pk to Random): In this game, we change how the public key $\mathsf{pk} = (X, Y)$ is generated. Recall that until now, we generate it by sampling $x \leftarrow \mathbb{Z}_p$ uniformly at random and then computing $\mathsf{pk} := (g^x, h^x)$. In this game, we sample it uniformly random as $\mathsf{pk} := (X, Y) \leftarrow \mathbb{G}^2$.

We can bound the distinguishing advantage between games $\mathbf{G}_{15}$ and $\mathbf{G}_{14}$ via a straightforward reduction B to DDH. To this end, B gets a tuple $\mathcal{D} := (g, h, \hat{g}, \hat{h}) \in \mathbb{G}^4$ as input and then simulates game $\mathbf{G}_{14}$ for A with the change that $\mathsf{pk} := (\hat{g}, \hat{h})$. In the end, B outputs whatever the game outputs. Clearly, B perfectly simulates game $\mathbf{G}_{14}$ for A if $\mathcal{D}$ is DH tuple. On the other hand, if $\mathcal{D}$ is not a DH tuple, then B perfectly simulates game $\mathbf{G}_{15}$ for A. Thus, we get

$$|\Pr[\mathbf{G}_{15} \Rightarrow 1] - \Pr[\mathbf{G}_{14} \Rightarrow 1]| \leq \varepsilon_{\mathrm{ddh}}.$$

We conclude by bounding the probability that the final game $\mathbf{G}_{15}$ outputs 1.

Lemma 3. *We have* $\Pr[\mathbf{G}_{15} \Rightarrow 1] \leq q_{\mathrm{h}}/p + 2/p$.

Proof. First, we show that the aggregated public key $\mathsf{apk}_* = (\tilde{X}_*, \tilde{Y}_*)$ in the forgery will also not be "correctly formed", with overwhelming probability. That means, if (g, h, X, Y) is not a DH tuple, then $(g, h, \tilde{X}_*, \tilde{Y}_*)$ is also not a DH tuple, with overwhelming probability. For this, consider a public key list $\mathcal{L} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ with $\mathsf{pk}_1 = \mathsf{pk}$, and let $(\tilde{X}, \tilde{Y}) := \mathsf{KAgg}(\mathcal{L})$. In the following, we condition on the event that $(g, h, \mathsf{pk}) = (g, h, X, Y)$ is not a DH tuple. Then, for any given (even adversarially chosen) $(a_i)_{i \in I}$, where $i \in [n]$ ranges over all indices such that $\mathsf{pk}_i \neq \mathsf{pk} \in \mathcal{L}$, there exists at most one value $a_1 \in \mathbb{Z}_p$ such that $(g, h, \tilde{X}, \tilde{Y})$ is a DH tuple. Otherwise, by contrapositive, assume that there are two values $a_1 \neq a_1'$ such that $(g, h, \tilde{X}, \tilde{Y})$ and $(g, h, \tilde{X}', \tilde{Y}')$ are both DH tuples, where

$$(\tilde{X}, \tilde{Y}) = (X, Y)^{m_{\mathcal{L}}(\mathsf{pk}) a_1} \prod_{\mathsf{pk}_i \neq \mathsf{pk} \in \mathcal{L}} (X_i, Y_i)^{a_i},$$
$$(\tilde{X}', \tilde{Y}') = (X, Y)^{m_{\mathcal{L}}(\mathsf{pk}) a_1'} \prod_{\mathsf{pk}_i \neq \mathsf{pk} \in \mathcal{L}} (X_i, Y_i)^{a_i},$$

and $m_{\mathcal{L}}(\mathsf{pk}) \in \mathbb{N}$ denotes the multiplicity of pk in the public key list $\mathcal{L}$. However, taking the ratio $(\tilde{X}, \tilde{Y})/(\tilde{X}', \tilde{Y}')$ leads to $\mathsf{pk} = (X, Y)$ also being a DH tuple. Clearly, a contradiction. Further, the probability to sample that unique $a_1 \leftarrow \mathbb{Z}_p$ is exactly $1/p$. Thus, we find that $(g, h, \tilde{X}, \tilde{Y})$ is a DH tuple with probability at most $1/p$. By a union bound over all random oracle queries, we get that $(g, h, \tilde{X}_*, \tilde{Y}_*)$ is a DH tuple with probability at most q_{h}/p.

Next, we argue that producing a forgery is statistically hard, conditioned on apk_* not being a DH tuple. This essentially boils down to statistical soundness of the Schnorr-Okamoto-like proofs. For this, let $\mathsf{apk} = (\tilde{X}, \tilde{Y})$ be a non-DH tuple. Consider any input (R, S) to the random oracle H_c (along with apk and some message μ). Then, there exists at most one challenge c such that there exists a suitable response (z, y) that satisfies the verification equation

$$(R, S)(\tilde{X}, \tilde{Y})^c = (g, h)^z (u, v)^y = (g, h)^{z + \alpha y},$$

where we use $(u, v) = \mathsf{H}_{uv}(b, \mathsf{apk}, \mu) = (g, h)^\alpha$ for some $\alpha \in \mathbb{Z}_p$ which is guaranteed in the forgery. For contradiction, suppose that there are two different challenges $c \neq c'$ with suitable $(z, y) \neq (z', y')$ satisfying the stated equation above. Then, taking the ratio of the two equations yields that $(\tilde{X}, \tilde{Y})$ is a DH tuple, which contradicts our assumption. Thus, there is at most one valid challenge c, and the probability of sampling it is $1/p$. Finally, the probability that $\mathsf{pk} = (X, Y)$ is a DH tuple, is bounded by $1/p$, and we conclude our proof.

□

□

5 Two-Round Threshold Signatures

In this section, we present our two-round threshold signature scheme $\mathsf{Earpick\text{-}TS}$ and then give a tight security proof for it under the DDH assumption. We start by giving the syntax and security definition of two-round threshold signature schemes following Chen [Che25] with slight adaptions to the static security model.

5.1 Formal Definitions

Syntax. In the following, we define two-round threshold signature schemes. We now give an informal description. The centralized key generation algorithm KGen generates a public key, n public key shares, and n secret key shares. The two-round signing protocol consists of three algorithms Sign_1, Sign_2, and $\mathsf{Combine}$, as for multi-signature schemes. We use the procedure $\mathsf{TSJointSign}$ in Fig. 5 to describe an honest execution of the signing protocol and also to define the completeness property.

$\underline{\mathsf{TSJointSign}(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathcal{S}, \mu, \{\mathsf{sk}_i\}_{i\in\mathcal{S}})}$

1: **for** $k \in \mathcal{S}$ **do**
2: $\quad (\mathsf{st}_k, \mathsf{msg}_{1,k}) \leftarrow \mathsf{Sign}_1(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathcal{S}, \mu, k, \mathsf{sk}_k)$
3: **for** $k \in \mathcal{S}$ **do**
4: $\quad \mathsf{msg}_{2,k} \leftarrow \mathsf{Sign}_2(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathcal{S}, \mu, k, \mathsf{sk}_k, \mathsf{st}_k, \{\mathsf{msg}_{1,i}\}_{i\in\mathcal{S}})$
5: **return** $\sigma \leftarrow \mathsf{Combine}(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathcal{S}, \mu, \{\mathsf{msg}_{1,i}\}_{i\in\mathcal{S}}, \{\mathsf{msg}_{2,i}\}_{i\in\mathcal{S}})$

Figure 5: Procedure $\mathsf{TSJointSign}$.

Definition 5 (Two-Round Threshold Signature Schemes)**.** A two-round threshold signature scheme TS consists of the following algorithms:

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

For completeness, we require that for every λ, every $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, every n and t that TS supports, every $(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in[n]}) \in \mathsf{KGen}(n, t)$, every subset of signers $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| = t + 1$, and every message μ, we have

$$\Pr[\mathsf{Vf}(\mathsf{pk}, \mu, \sigma) = 1 : \sigma \leftarrow \mathsf{TSJointSign}(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathcal{S}, \mu, \{\mathsf{sk}_i\}_{i\in\mathcal{S}})] = 1,$$

where the procedure $\mathsf{TSJointSign}$, modeling an honest execution of TS, is defined in Fig. 5.

Security. The static security of two-round threshold signature schemes is defined by the security game TSUF described in Fig. 6. The adversary A is given the public parameter pp, the public key pk, the public key shares of all users $\{\mathsf{pk}_i\}_{i\in[n]}$, and the secret key shares of all users in $\mathcal{C}$, a corruption set of size at most t as an input to the game. Throughout the game, A also has access to the signing oracles, defined similarly to those for multi-signatures. Its goal is to forge a signature on some message that it has never queried to the signing oracle.

Definition 6 (Static Security of Two-Round Threshold Signature)**.** A two-round threshold signature scheme TS is $(q_\mathrm{s}, \tau, \varepsilon)$-statically secure if for every λ, every adversary A that makes at most $q_\mathrm{s} = q_\mathrm{s}(\lambda)$ signing queries and is of running time at most $\tau = \tau(\lambda)$, for every n and t that TS supports, every $\mathcal{C} \subseteq [n]$ such that $|\mathcal{C}| \leq t$, $\Pr[\mathsf{TSUF}^{\mathsf{A}}_{\mathsf{TS}}(1^\lambda, n, t, \mathcal{C}) = 1] \leq \varepsilon = \varepsilon(\lambda)$, where the security game TSUF is defined in Fig. 6.

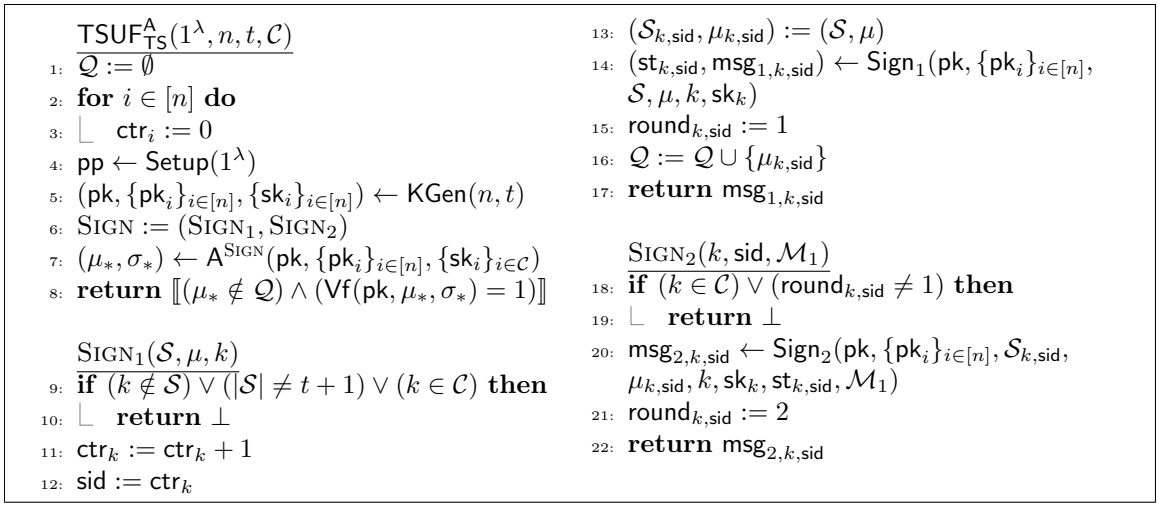

Figure 6: The TSUF security game.

5.2 Our Scheme

We formally define our two-round threshold signature scheme Earpick-TS as pseudocode in Fig. 7 and give a verbal description next.

Shamir's Secret Sharing. We first recap Shamir's secret sharing [Sha79]. To (n, t)-share a secret $x = x_0 \in \mathbb{Z}_p$, the sharing algorithm $\mathsf{Share}(n, t, x)$ sets $\alpha_0 := x$ and samples other t coefficients $\alpha_1, \ldots, \alpha_t \leftarrow \mathbb{Z}_p$ for a degree-t polynomial $\alpha(\cdot)$. It outputs the secret shares $\{x_i\}_{i\in[n]}$ where $x_i = \alpha(i)$. The secret is recovered by polynomial interpolation. Any x_i can be recovered with $\{x_j\}_{j\in\mathcal{S}}$ for a $(t+1)$-subset $\mathcal{S}$ of $[n]$ such that $i \notin \mathcal{S}$. In particular, x_i is given by the linear combination $x_i = \sum_{j\in\mathcal{S}} \lambda_{i,\mathcal{S},j} x_j$, where the Lagrange coefficient for $\mathcal{S}$ to recover x_i is $\lambda_{i,\mathcal{S},j} = \prod_{k\in\mathcal{S},k\neq j}(i-k)/(j-k)$. When $i = 0$, we write the Lagrange coefficient in short as $\lambda_{\mathcal{S},j}$. Shamir's secret sharing scheme is perfectly secure, in the sense that $x_0, x_1, \ldots, x_n$ are $(t+1)$-wise independent.

Setup and Key Generation. The global parameters of the scheme consist of group parameters $(\mathbb{G}, p, g)$ generated by GrGen and a uniformly random group element $h \in \mathbb{G}$. The key generation algorithm KGen uniformly samples $x \leftarrow \mathbb{Z}_p$ and generates t-out-of-n Shamir's secret shares $\{x_i\}_{i\in[n]}$ using algorithm Share. The public key is $\mathsf{pk} := (g, h)^x \in \mathbb{G}^2$, the public key shares are $\mathsf{pk}_i := (g, h)^{x_i}$ for $i \in [n]$, and the secret key shares are $\mathsf{sk}_i := x_i$ for $i \in [n]$.

Signing Protocol. Suppose a set of (at least) $t+1$ signers $\mathcal{S} \subseteq [n]$ want to sign a message μ. We assume all signers are aware of $(\mathcal{S}, \mu)$. We describe the signing protocol from the perspective of the k-th signer with secret key share $\mathsf{sk}_k := x_k$ and public key share $\mathsf{pk}_k := (X_k, Y_k)$. Signers make use of a hash function H_{uv} to get pairs of group elements, H_D to get single group elements, H_b to get single bits, and H_c to get single field elements.

1. *Commitment Phase.* The signer derives a group element $D := \mathsf{H}_D(\mathsf{pk}, \mu) \in \mathbb{G}$, and computes $Z_k := D^{x_k}$ along with a proof of discrete logarithm equality π_k for statement (g, D, X_k, Z_k). Next, it derives commitment keys

$$(u_0, v_0) := \mathsf{H}_{uv}(0, \mathsf{pk}, \mu) \in \mathbb{G}^2, \quad (u_1, v_1) := \mathsf{H}_{uv}(1, \mathsf{pk}, \mu) \in \mathbb{G}^2.$$

Then, it samples elements $r_{k,0}, s_{k,0}, r_{k,1}, s_{k,1} \leftarrow \mathbb{Z}_p$ and computes the commitments

$$(R_{k,0}, S_{k,0}) := (g, h)^{r_{k,0}}(u_0, v_0)^{s_{k,0}}, \quad (R_{k,1}, S_{k,1}) := (g, h)^{r_{k,1}}(u_1, v_1)^{s_{k,1}}.$$

It sends $\mathsf{msg}_{1,k} := (Z_k, \pi_k, R_{k,0}, S_{k,0}, R_{k,1}, S_{k,1})$ as its protocol message to all signers in $\mathcal{S}$.

2. *Response Phase.* Upon receiving the protocol messages $\{(Z_i, \pi_i, R_{i,0}, S_{i,0}, R_{i,1}, S_{i,1})\}_{i\in\mathcal{S}}$ from all signers in $\mathcal{S}$, the signer checks that the proofs of discrete logarithm equality π_i verify with respect to statement (g, D, X_i, Z_i) for all $i \in \mathcal{S}$. If one of these proofs does not verify, the signer aborts. Otherwise, it interpolates all elements $\{Z_i\}_{i\in\mathcal{S}}$ into $\tilde{Z} := \prod_{i\in\mathcal{S}} Z_i^{\lambda_{\mathcal{S},i}}$ and derives a bit $b := \mathsf{H}_b(\tilde{Z})$. Next, it aggregates all b-side commitments into $(\tilde{R}, \tilde{S}) := \prod_{i\in\mathcal{S}}(R_{i,b}, S_{i,b})$, and computes a hash challenge as $c := \mathsf{H}_c(\mathsf{pk}, \mu, \tilde{R}, \tilde{S}) \in \mathbb{Z}_p$. It then computes $z_k := r_{k,b} + c\lambda_{\mathcal{S},k}x_k \in \mathbb{Z}_p$, and sends $\mathsf{msg}_{2,k} := (z_k, y_k) := (z_k, s_{k,b})$ as its protocol message to all signers in $\mathcal{S}$.

3. *Combine Phase.* The signer combines the responses as $(\tilde{z}, \tilde{y}) := \sum_{i\in\mathcal{S}}(z_i, y_i)$. The final signature is then $(b, c, \tilde{z}, \tilde{y})$, where b and c are defined as in the response phase.

Verification. The verifier gets the public key (X, Y), the signature $(b, c, \tilde{z}, \tilde{y})$, and the message μ. It computes $(u, v) := \mathsf{H}_{uv}(b, \mathsf{pk}, \mu)$ and recovers the commitments as $(\tilde{R}, \tilde{S}) := (g, h)^{\tilde{z}}(u, v)^{\tilde{y}}(X, Y)^{-c}$. It accepts the signature if and only if $c = \mathsf{H}_c(\mathsf{pk}, \mu, \tilde{R}, \tilde{S})$.

5.3 Security Analysis

Earpick-TS is tightly secure under the DDH assumption. We note that correctness of our scheme is straightforward.

Theorem 4. *If the* DDH *problem is* $(\tau_{\mathrm{ddh}}, \varepsilon_{\mathrm{ddh}})$*-hard for the group generation algorithm* GrGen*, then Earpick-TS is a* $(q_{\mathrm{s}}, \tau_{\mathrm{uf}}, \varepsilon_{\mathrm{uf}})$*-secure threshold signature scheme in the random oracle model (ROM) against any adversary that makes at most* q_{h} *hash queries, where essentially* $\tau_{\mathrm{ddh}} \approx \tau_{\mathrm{uf}}$*, and*

$$\varepsilon_{\mathrm{uf}} \leq 4\varepsilon_{\mathrm{ddh}} + \frac{{q_{\mathrm{h}}}^2 + 2q_{\mathrm{s}}q_{\mathrm{h}} + 5q_{\mathrm{h}} + 6}{p}.$$

The proof is very similar to the proof of Theorem 2. It follows the same roadmap. By a series of hybrid games, we switch to a game that: simulates the NIZK proofs; programs H_{uv} with dual keys and use them to sign; precomputes the element $\tilde{Z}$ to control the pseudorandom bit from H_b. The last step is to switch pk to random, that is to sample the main public key $(X, Y) \leftarrow \mathbb{G}^2$ instead of computing it as $(X, Y) := (g, h)^x$ for $x \leftarrow \mathbb{Z}_p$. As a result, forging a signature is unconditionally hard for the adversary that selects the wrong bit b_*.

Here we highlight the main differences from the proof of Theorem 2 that require extra care. First, to be prepared for the final step, we require the game to use the main secret key x only to generate (X, Y). In Earpick-TS, there is one more step that uses the main secret key x: to generate the secret key shares $\{x_i\}_{i\in[n]}$. At some point before the hybrid game that starts using dual keys to sign, we add a hybrid game that simulates the threshold key generation. The hybrid game uniformly samples the secret key shares $\{x_i\}_{i\in\mathcal{C}'}$ for an arbitrary t-subset $\mathcal{C}' \supseteq \mathcal{C}$. It generates the public key share (X_i, Y_i) as $(g, h)^{x_i}$ for every $i \in \mathcal{C}'$. For $i \in [n] \setminus \mathcal{C}'$, it generates (X_i, Y_i) without using x_i by polynomial interpolation on public key shares, namely

$$(X_i, Y_i) = (X, Y)^{\lambda_{i,\mathcal{C}'\cup\{0\},0}} \prod_{j\in\mathcal{C}'} (X_j, Y_j)^{\lambda_{i,\mathcal{C}'\cup\{0\},j}}.$$

It gives (X, Y), $\{(X_i, Y_i)\}_{i\in[n]}$, and $\{x_i\}_{i\in\mathcal{C}}$ to the adversary A as inputs. The game does not run $\mathsf{Share}(x, n, t)$ so x is not used.

Second, when we program H_{uv}, we still set $\mathsf{H}_{uv}(b, \mathsf{pk}, \mu) := (X, Y)^{1/w}$. Here w is a dual key under the main public key (X, Y). However, the hybrid game needs to sign under public key shares, so w cannot be directly used to sign. Above we have generated (X_k, Y_k) for $k \in [n] \setminus \mathcal{C}'$ as a linear combination of (X, Y) and $\{(X_j, Y_j)\}_{j\in\mathcal{C}'}$. Since the hybrid game knows $\{x_j\}_{j\in\mathcal{C}'}$, (X_k, Y_k) is in fact a linear combination $(g, h)^{\beta}(X, Y)^{\gamma}$ of (g, h) and (X, Y) with known (β, γ). Then from w, the

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

$(\mathbb{G}, p, g) \leftarrow \mathsf{GrGen}(1^\lambda)$

$h \leftarrow \mathbb{G}$

return $(\mathbb{G}, p, g, h)$

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

$x \leftarrow \mathbb{Z}_p$

$(X, Y) := (g, h)^x$

$\{x_i\}_{i\in[n]} \leftarrow \mathsf{Share}(n, t, x)$

$\mathsf{pk} := (X, Y)$

for $i \in [n]$ **do**

  $\mathsf{sk}_i := x_i$

  $\mathsf{pk}_i := (g, h)^{x_i}$

return $(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \{\mathsf{sk}_i\}_{i\in[n]})$

$\underline{\mathsf{Sign}_1(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathcal{S}, \mu, k, \mathsf{sk}_k)}$

$x_k := \mathsf{sk}_k$

$D := \mathsf{H}_D(\mathsf{pk}, \mu)$

$Z_k := D^{x_k}$

$\pi_k := \mathsf{DLEQ.NIP}^{\mathsf{H}}((g, D, X_k, Z_k), x_k)$

$(u_0, v_0) := \mathsf{H}_{uv}(0, \mathsf{pk}, \mu)$

$(u_1, v_1) := \mathsf{H}_{uv}(1, \mathsf{pk}, \mu)$

$r_{k,0},\ s_{k,0},\ r_{k,1},\ s_{k,1} \leftarrow \mathbb{Z}_p$

$(R_{k,0}, S_{k,0}) := (g, h)^{r_{k,0}}(u_0, v_0)^{s_{k,0}}$

$(R_{k,1}, S_{k,1}) := (g, h)^{r_{k,1}}(u_1, v_1)^{s_{k,1}}$

$\mathsf{st}_k := (r_{k,0}, s_{k,0}, r_{k,1}, s_{k,1})$

$\mathsf{msg}_{1,k} := (Z_k, \pi_k, R_{k,0}, S_{k,0}, R_{k,1}, S_{k,1})$

$\underline{\mathsf{Sign}_2(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathcal{S}, \mu, k, \mathsf{sk}_k, \mathsf{st}_k, \mathcal{M}_1)}$

$x_k := \mathsf{sk}_k$

$(r_{k,0}, s_{k,0}, r_{k,1}, s_{k,1}) := \mathsf{st}_k$

$\{\mathsf{msg}_{1,i}\}_{i\in\mathcal{S}} := \mathcal{M}_1$

$D := \mathsf{H}_D(\mathsf{pk}, \mu)$

for $i \in \mathcal{S}$ **do**

  $(X_i, Y_i) := \mathsf{pk}_i$

  $(Z_i, \pi_i, R_{i,0}, S_{i,0}, R_{i,1}, S_{i,1}) := \mathsf{msg}_{1,i}$

  if $\mathsf{DLEQ.NIV}^{\mathsf{H}}((g, D, X_i, Z_i), \pi_i) = 0$ **then**

    return $\perp$

$\tilde{Z} := \prod_{i\in\mathcal{S}} Z_i^{\lambda_{\mathcal{S},i}}$

$b := \mathsf{H}_b(\tilde{Z})$

$(\tilde{R}, \tilde{S}) := \prod_{i\in\mathcal{S}}(R_{i,b}, S_{i,b})$

$c := \mathsf{H}_c(\mathsf{pk}, \mu, \tilde{R}, \tilde{S})$

$z_k := r_{k,b} + c\lambda_{\mathcal{S},k}x_k$

$y_k := s_{k,b}$

return (z_k, y_k)

$\underline{\mathsf{Combine}(\mathsf{pk}, \{\mathsf{pk}_i\}_{i\in[n]}, \mathcal{S}, \mu, \mathcal{M}_1, \mathcal{M}_2)}$

$\{\mathsf{msg}_{1,i}\}_{i\in\mathcal{S}} := \mathcal{M}_1$

$\{\mathsf{msg}_{2,i}\}_{i\in\mathcal{S}} := \mathcal{M}_2$

for $i \in \mathcal{S}$ **do**

  $(Z_i, \pi_i, R_{i,0}, S_{i,0}, R_{i,1}, S_{i,1}) := \mathsf{msg}_{1,i}$

  $(z_i, y_i) := \mathsf{msg}_{2,i}$

$\tilde{Z} := \prod_{i\in\mathcal{S}} Z_i^{\lambda_{\mathcal{S},i}}$

$b := \mathsf{H}_b(\tilde{Z})$

$(\tilde{R}, \tilde{S}) := \prod_{i\in\mathcal{S}}(R_{i,b}, S_{i,b})$

$c := \mathsf{H}_c(\mathsf{pk}, \mu, \tilde{R}, \tilde{S})$

$(\tilde{z}, \tilde{y}) := \sum_{i\in\mathcal{S}}(z_i, y_i)$

return $(b, c, \tilde{z}, \tilde{y})$

$\underline{\mathsf{Vf}(\mathsf{pk}, \mu, \sigma)}$

$(X, Y) := \mathsf{pk}$

$(b, c, \tilde{z}, \tilde{y}) := \sigma$

$(u, v) := \mathsf{H}_{uv}(b, \mathsf{pk}, \mu)$

$(\tilde{R}, \tilde{S}) := (g, h)^{\tilde{z}}(u, v)^{\tilde{y}}(X, Y)^{-c}$

return $[\![c = \mathsf{H}_c(\mathsf{pk}, \mu, \tilde{R}, \tilde{S})]\!]$

Figure 7: Our two-round threshold signature scheme Earpick-TS.

hybrid game knows a dual key $(\beta, \gamma w)$ under (X_k, Y_k), satisfying $(g,h)^{\beta}(u,v)^{\gamma w} = (X_k, Y_k)$. The hybrid game signs on behalf of user k by setting $z_k := r_{k,b} + c\lambda_{\mathcal{S},k}\beta$ and $y_k := s_{k,b} + c\lambda_{\mathcal{S},k}\gamma w$.

Lastly, we need to be careful that $\tilde{Z} = \prod_{i\in\mathcal{S}} Z_i^{\lambda_{\mathcal{S},i}} = D^{\sum_{i\in\mathcal{S}} \lambda_{\mathcal{S},i} x_i} = D^x$ is invariant for fixed μ while different $\mathcal{S}$. We also need to ensure this in the hybrid games. We let the hybrid game samples a $\mathsf{Z}[\mu]$ for every message μ and forces $\tilde{Z} = \mathsf{Z}[\mu]$ in every signing session on μ. To do so, the hybrid game computes $Z_i := D^{x_i}$ for every $i \in \mathcal{C}'$, and then it computes Z_i for $i \in [n] \setminus \mathcal{C}'$ by interpolation. It always outputs Z_i when signing μ on behalf of user $i \in [n] \setminus \mathcal{C}'$. Since the adversary cannot choose public key (shares), we do not need the trapdoor d to compute Z_i for corrupt user i. Furthermore, we avoid the need to handle the case the adversary copying the honest public key, as was necessary in the proof of Theorem 2. We present the respective sequence of hybrid games in Appendix B.2.

6 Two-Round Multi-Signatures with Tight Multi-User Security

In this section, we present Earpick-muMS, a variant of Earpick-MS which achieves tight multi-user security.

6.1 Multi-User Security Model

The multi-user security of two-round multi-signature schemes with key aggregation is defined via the security game muMSUF described in Fig. 8. The adversary A is given the public parameters pp and N target public keys $\{\mathsf{pk}_{*,i}\}_{i\in[N]}$, each possessed by an honest signer. Its goal is to forge a multi-signature under a public key $\mathcal{L}_*$ that contains any target public key pk_{*,k_*} of its choice. It can open signing sessions with any honest users via the signing oracles. For the forgery to be non-trivial, it is required that no signing session on $(\mathcal{L}_*, \mu_*)$ has been started with the k_*-th honest user who possesses pk_{*,k_*}. Crucially, $\mathcal{L}_*$ may contain multiple honest public keys, and A is considered winning even if it has opened signing sessions on $(\mathcal{L}_*, \mu_*)$ with other honest users than the k_*-th.

Definition 7 (Multi-User Security of Two-Round Multi-Signature Schemes with Key Aggregation)**.** A two-round multi-signature scheme MS with key aggregation is $(N, q_\mathrm{s}, \tau, \varepsilon)$-secure in the multi-user model if for every λ, every adversary A that makes at most $q_\mathrm{s} = q_\mathrm{s}(\lambda)$ queries to $\textsc{Sign}_1$ and is of running time at most $\tau = \tau(\lambda)$, $\Pr[\mathsf{muMSUF}^{\mathsf{A}}_{\mathsf{MS}}(1^\lambda, N) = 1] \leq \varepsilon = \varepsilon(\lambda, N)$, where the security game muMSUF is defined in Fig. 8.

6.2 Our scheme

Earpick-muMS differs from Earpick-MS only in how to derive the pseudorandom bit in the signing protocol. Specifically, now each signer computes Z_i based on a per-public-key D_i instead of a common D, and the pseudorandom bit is derived as $b := \mathsf{H}_b(\{Z_i\}_{i\in[n]})$. We formally define its signing protocol as pseudocode in Fig. 9 and give a verbal description next.

Signing Protocol. Suppose n signers with public keys $\mathcal{L} := \{(X_i, Y_i)\}_{i\in[n]}$ want to sign a message μ. We describe the signing protocol from the perspective of the first signer with secret key $\mathsf{sk}_1 := x_1$ and public key $\mathsf{pk}_1 := (X_1, Y_1)$. Let $\mathsf{apk} := \mathsf{KAgg}(\mathcal{L})$ be the aggregated public key as defined above. Further, signers make use of a hash function H_{uv} to get pairs of group elements, H_D to get single group elements, H_b to get single bits, and H_c to get single field elements.

1. *Commitment Phase.* The signer derives a group element $D_1 := \mathsf{H}_D(\mathcal{L}, \mu, \mathsf{pk}_1) \in \mathbb{G}$, and computes $Z_1 := D_1^{x_1}$ along with a proof of discrete logarithm equality π_1 for statement (g, D_1, X_1, Z_1). Next, it derives commitment keys

$$(u_0, v_0) := \mathsf{H}_{uv}(0, \mathsf{apk}, \mu) \in \mathbb{G}^2, \quad (u_1, v_1) := \mathsf{H}_{uv}(1, \mathsf{apk}, \mu) \in \mathbb{G}^2.$$

$\underline{\mathsf{muMSUF}^{\mathsf{A}}_{\mathsf{MS}}(\lambda, N)}$
1: $\mathsf{ctr} := 0$
2: $\mathcal{Q} := \emptyset$
3: $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
4: **for** $i \in [N]$ **do**
5: $\quad (\mathsf{sk}_{*,i}, \mathsf{pk}_{*,i}) \leftarrow \mathsf{KGen}()$
6: $(k_*, \mathcal{L}_*, \mu_*, \sigma_*) \leftarrow \mathsf{A}^{\textsc{Sign}_1, \textsc{Sign}_2}(\{\mathsf{pk}_{*,i}\}_{i\in[N]})$
7: **return** $\Big[\!\Big[(k_*, \mathcal{L}_*, \mu_*) \notin \mathcal{Q} \wedge \mathsf{pk}_{*,k_*} \in \mathcal{L}_* \wedge \mathsf{Vf}(\mathsf{KAgg}(\mathcal{L}_*), \mu_*, \sigma_*) = 1\Big]\!\Big]$

$\underline{\textsc{Sign}_1(k, \mathcal{L}, \mu)}$
8: **if** $\mathcal{L}[1] \neq \mathsf{pk}_{*,k}$ **then**
9: $\quad$ **return** $\bot$
10: $\mathsf{ctr} := \mathsf{ctr} + 1$
11: $\mathsf{sid} := \mathsf{ctr}$
12: $\mathcal{Q} := \mathcal{Q} \cup (k, \mathcal{L}, \mu)$
13: $(k_{\mathsf{sid}}, \mathcal{L}_{\mathsf{sid}}, \mu_{\mathsf{sid}}) := (k, \mathcal{L}, \mu)$
14: $(\mathsf{st}_{\mathsf{sid}}, \mathsf{msg}_{1,\mathsf{sid}}) \leftarrow \mathsf{Sign}_1(\mathcal{L}, \mu, \mathsf{sk}_*)$
15: $\mathsf{round}_{\mathsf{sid}} := 1$
16: **return** msg_1

$\underline{\textsc{Sign}_2(\mathsf{sid}, \mathcal{M}_1)}$
17: **if** $(\mathsf{round}_{\mathsf{sid}} \neq 1) \vee (\mathcal{M}_1[1] \neq \mathsf{msg}_{1,\mathsf{sid}})$ **then**
18: $\quad$ **return** $\bot$
19: $\mathsf{msg}_2 \leftarrow \mathsf{Sign}_2(\mathcal{L}_{\mathsf{sid}}, \mu_{\mathsf{sid}}, \mathsf{sk}_{*,k_{\mathsf{sid}}}, \mathsf{st}_{\mathsf{sid}}, \mathcal{M}_1)$
20: $\mathsf{round}_{\mathsf{sid}} := 2$
21: **return** msg_2

Figure 8: The muMSUF security game.

Then, it samples elements $r_{1,0}, s_{1,0}, r_{1,1}, s_{1,1} \leftarrow \mathbb{Z}_p$ and computes the commitments

$$(R_{1,0}, S_{1,0}) := (g, h)^{r_{1,0}}(u_0, v_0)^{s_{1,0}}, \quad (R_{1,1}, S_{1,1}) := (g, h)^{r_{1,1}}(u_1, v_1)^{s_{1,1}}.$$

It sends $\mathsf{msg}_1 := (Z_1, \pi_1, R_{1,0}, S_{1,0}, R_{1,1}, S_{1,1})$ as its protocol message to the other signers.

2. *Response Phase.* Upon receiving the protocol messages $\{(Z_i, \pi_i, R_{i,0}, S_{i,0}, R_{i,1}, S_{i,1})\}_{i\in[n]}$ from all signers, the signer checks that the proofs of discrete logarithm equality π_i verify with respect to statement (g, D_i, X_i, Z_i) for all $i \in [n]$. If one of these proofs does not verify, the signer aborts. Otherwise, it derives a bit $b := \mathsf{H}_b(\{Z_i\}_{i\in[n]})$. Next, it aggregates all b-side commitments into $(\tilde{R}, \tilde{S}) := \prod_{i=1}^n (R_{i,b}, S_{i,b})$, and computes a hash challenge as $c := \mathsf{H}_c(\mathsf{apk}, \mu, \tilde{R}, \tilde{S}) \in \mathbb{Z}_p$. It then computes $z_1 := r_{1,b} + a_1 c x_1 \in \mathbb{Z}_p$ where $a_1 := \mathsf{H}_a(\mathcal{L}, X_1, Y_1) \in \mathbb{Z}_p$, and sends $\mathsf{msg}_2 := (z_1, y_1) := (z_1, s_{1,b})$ as its protocol message to the other signers.

3. *Combine Phase.* The signer combines the responses as $(\tilde{z}, \tilde{y}) := \sum_{i=1}^n (z_i, y_i)$. The final signature is then $(b, c, \tilde{z}, \tilde{y})$, where b and c are defined as in the response phase above.

6.3 Security Analysis

Earpick-muMS is tightly secure in the multi-user model under the DDH assumption.

Theorem 5. *If the* DDH *problem is* $(\tau_{\mathrm{ddh}}, \varepsilon_{\mathrm{ddh}})$*-hard and the* CDH *problem is* $(\tau_{\mathrm{cdh}}, \varepsilon_{\mathrm{cdh}})$*-hard for the group generation algorithm* GrGen*, then Earpick-muMS is an* $(N, q_{\mathrm{s}}, \tau_{\mathrm{uf}}, \varepsilon_{\mathrm{uf}})$*-secure multi-signature scheme in the random oracle model (ROM) against any adversary that makes at most* q_{h} *total hash queries, where essentially* $\tau_{\mathrm{ddh}} \approx \tau_{\mathrm{uf}}$*,* $\tau_{\mathrm{cdh}} \approx \tau_{\mathrm{uf}}$*, and*

$$\varepsilon_{\mathrm{uf}} \leq 4\varepsilon_{\mathrm{ddh}} + \varepsilon_{\mathrm{cdh}} + \frac{2q_h^2 + 2q_s q_h + 7q_h + N^2 + N + 6}{p}.$$

In the proof, we generate all the honest public keys $(X_i, Y_i) = (g, h)^{x_i}$ by rerandomizing a single "seed" public key (X, Y) with a corresponding secret key x. Consequently, all signing queries are essentially under the seed public key. Meanwhile, using per-public-key D_i in the signing protocol allows us to switch (by DDH) to a game where $\textsc{Sign}_1$ independently samples Z_i on behalf of each honest signer, even under the same $(\mathcal{L}, \mu)$. This ensures that the random bit $b := \mathsf{H}_b(\{Z_i\}_{i\in[n]})$ remains perfectly hidden to the adversary until it has queried $\textsc{Sign}_1(k, \mathcal{L}, \mu)$ for every honest public key $\mathsf{pk}_{*,k} \in \mathcal{L}$ and seen the corresponding Z_i.

$\underline{\mathsf{Sign}_1(\mathcal{L}, \mu, \mathsf{sk}_1)}$
$\{(X_i, Y_i)\}_{i\in[n]} := \mathcal{L}$
$x_1 := \mathsf{sk}_1$
$\mathsf{apk} := \mathsf{KAgg}(\mathcal{L})$
$D_1 := \mathsf{H}_D(\mathcal{L}, \mu, \mathsf{pk}_1)$
$Z_1 := D_1^{x_1}$
$\pi_1 := \mathsf{DLEQ.NIP}^{\mathsf{H}}((g, D_1, X_1, Z_1), x_1)$
$(u_0, v_0) := \mathsf{H}_{uv}(0, \mathsf{apk}, \mu)$
$(u_1, v_1) := \mathsf{H}_{uv}(1, \mathsf{apk}, \mu)$
$r_{1,0},\ s_{1,0},\ r_{1,1},\ s_{1,1} \leftarrow \mathbb{Z}_p$
$(R_{1,0}, S_{1,0}) := (g, h)^{r_{1,0}}(u_0, v_0)^{s_{1,0}}$
$(R_{1,1}, S_{1,1}) := (g, h)^{r_{1,1}}(u_1, v_1)^{s_{1,1}}$
$\mathsf{st}_1 := (r_{1,0}, s_{1,0}, r_{1,1}, s_{1,1})$
$\mathsf{msg}_1 := (Z_1, \pi_1, R_{1,0}, S_{1,0}, R_{1,1}, S_{1,1})$

$\underline{\mathsf{Sign}_2(\mathcal{L}, \mu, \mathsf{sk}_1, \mathsf{st}_1, \mathcal{M})}$
$\{(X_i, Y_i)\}_{i\in[n]} := \mathcal{L}$
$x_1 := \mathsf{sk}_1$
$(r_{1,0}, s_{1,0}, r_{1,1}, s_{1,1}) := \mathsf{st}_1$
$\{\mathsf{msg}_i\}_{i\in[n]} := \mathcal{M}$
$\mathsf{apk} := \mathsf{KAgg}(\mathcal{L})$
for $i \in [n]$ **do**
    $D_i := \mathsf{H}_D(\mathcal{L}, \mu, \mathsf{pk}_i)$
    $(Z_i, \pi_i, R_{i,0}, S_{i,0}, R_{i,1}, S_{i,1}) := \mathsf{msg}_i$
    if $\mathsf{DLEQ.NIV}^{\mathsf{H}}((g, D_i, X_i, Z_i), \pi_i) = 0$ **then**
        return $\bot$
$b := \mathsf{H}_b(\{Z_i\}_{i\in[n]})$
$(\tilde{R}, \tilde{S}) := \prod_{i=1}^n (R_{i,b}, S_{i,b})$
$c := \mathsf{H}_c(\mathsf{apk}, \mu, \tilde{R}, \tilde{S})$
$a_1 := \mathsf{H}_a(\mathcal{L}, X_1, Y_1)$
$z_1 := r_{1,b} + a_1 c x_1$
$y_1 := s_{1,b}$
return (z_1, y_1)

$\underline{\mathsf{Combine}(\mathcal{L}, \mu, \mathcal{M}, \mathcal{M}')}$
$\{\mathsf{msg}_i\}_{i\in[n]} := \mathcal{M}$
$\{\mathsf{msg}'_i\}_{i\in[n]} := \mathcal{M}'$
for $i \in [n]$ **do**
    $(Z_i, \pi_i, R_{i,0}, S_{i,0}, R_{i,1}, S_{i,1}) := \mathsf{msg}_i$
    $(z_i, y_i) := \mathsf{msg}'_i$
$b := \mathsf{H}_b(\{Z_i\}_{i\in[n]})$
$(\tilde{R}, \tilde{S}) := \prod_{i=1}^n (R_{i,b}, S_{i,b})$
$c := \mathsf{H}_c(\mathsf{apk}, \mu, \tilde{R}, \tilde{S})$
$(\tilde{z}, \tilde{y}) := \sum_{i=1}^n (z_i, y_i)$
return $(b, c, \tilde{z}, \tilde{y})$

Figure 9: Our two-round multi-signature scheme with key aggregation Earpick-muMS.

In comparison, for Earpick-MS with a common $D := \mathsf{H}_D(\mathcal{L}, \mu)$, $Z_i = D^{x_i}$ for every honest signer in $\mathcal{L}$ can be viewed as generated from a single seed $Z = D^x$. By DDH, we can switch to a game where $\textsc{Sign}_1$ samples Z uniformly, but these Z_i for honest signers are still correlated. Consequently, seeing one honest Z_i would leak information about others. We present the respective sequence of hybrid games in Appendix B.3.

References

[AB21] Handan Kilinç Alper and Jeffrey Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 157–188, Virtual Event, August 2021. Springer, Cham. 32

[BCJ08] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 449–458. ACM Press, October 2008. 2, 32

[BD21] Mihir Bellare and Wei Dai. Chain reductions for multi-signatures and the HBMS scheme. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 650–678. Springer, Cham, December 2021. 2, 4, 32

[BDLR25] Renas Bacho, Sourav Das, Julian Loss, and Ling Ren. Glacius: Threshold schnorr signatures from ddh with full adaptive security. In *Advances in Cryptology – EUROCRYPT 2025: 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4–8, 2025, Proceedings, Part II*, page 304–334, Berlin, Heidelberg, 2025. Springer-Verlag. 14

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 435–464. Springer, Cham, December 2018. 32

[BJ08] Ali Bagherzandi and Stanislaw Jarecki. Multisignatures using proofs of secret key possession, as secure as the Diffie-Hellman problem. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *SCN 08*, volume 5229 of *LNCS*, pages 218–235. Springer, Berlin, Heidelberg, September 2008. 32

[BL22] Renas Bacho and Julian Loss. On the adaptive security of the threshold BLS signature scheme. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 193–207. ACM Press, November 2022. 32

[BLSW24] Renas Bacho, Julian Loss, Gilad Stern, and Benedikt Wagner. Harts: High-threshold, adaptively secure, and robust threshold schnorr signatures. In *Advances in Cryptology – ASIACRYPT 2024: 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9–13, 2024, Proceedings, Part III*, page 104–140, Berlin, Heidelberg, 2024. Springer-Verlag. 32

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006. 2, 3, 5, 32

[BNN07] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 411–422. Springer, Berlin, Heidelberg, July 2007. 32

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Berlin, Heidelberg, January 2003. 2, 32

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, Berlin, Heidelberg, May 1996. 2

[BW24a] Renas Bacho and Benedikt Wagner. Tightly secure non-interactive bls multi-signatures. In *Advances in Cryptology – ASIACRYPT 2024: 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9–13, 2024, Proceedings, Part II*, page 397–422, Berlin, Heidelberg, 2024. Springer-Verlag. 32

[BW24b] Renas Bacho and Benedikt Wagner. Tightly secure threshold signatures over pairing-free groups. Cryptology ePrint Archive, Paper 2024/1557, 2024. 3, 32

[BW25] Renas Bacho and Benedikt Wagner. T-spoon: Tightly secure two-round multi-signatures with key aggregation. Cryptology ePrint Archive, Paper 2025/840, 2025. 2, 4, 5, 9, 32

[Che25] Yanbo Chen. Dazzle: Improved adaptive threshold signatures from ddh. In *Public-Key Cryptography – PKC 2025: 28th IACR International Conference on Practice and Theory of Public-Key Cryptography, Røros, Norway, May 12–15, 2025, Proceedings, Part III*, page 233–261, Berlin, Heidelberg, 2025. Springer-Verlag. 3, 19, 32

[CKM23] Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. Fully adaptive Schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 678–709. Springer, Cham, August 2023. 32

[Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Berlin, Heidelberg, August 2000. 5

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Berlin, Heidelberg, August 1993. 6, 7

[DEF+19] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy*, pages 1084–1101. IEEE Computer Society Press, May 2019. 32

[Des88] Yvo Desmedt. Society and group oriented cryptography: A new concept. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 120–127. Springer, Berlin, Heidelberg, August 1988. 3

[FH21] Masayuki Fukumitsu and Shingo Hasegawa. A tightly secure ddh-based multisignature with public-key aggregation. *Int. J. Netw. Comput.*, 11(2):319–337, 2021. 32

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018. 4

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987. 7

[Har94] Lein Harn. Group-oriented (mi) threshold signature and digital multisignature. *1EE Proceedings of Computers and Digital Techniques*, 141(5):307–313, 1994. 32

[HMP95] Patrick Horster, Markus Michels, and Holger Petersen. *Meta-Multisignature schemes based on the discrete logarithm problem*, pages 128–142. Springer US, Boston, MA, 1995. 32

[HT16] Viet Tung Hoang and Stefano Tessaro. Key-alternating ciphers and key-length extension: Exact bounds and multi-user security. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 3–32. Springer, Berlin, Heidelberg, August 2016. 14, 33

[Ita83] Kazuharu Itakura. A public-key cryptosystem suitable for digital multisignature. *NEC research and development*, 71:1–8, 1983. 1, 32

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003*, pages 155–164. ACM Press, October 2003. 5

[KZ20] Daniel Kales and Greg Zaverucha. An attack on some signature schemes constructed from five-pass identification schemes. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 3–22. Springer, Cham, December 2020. 2

[LHL95] Chuan-Ming Li, Tzonelih Hwang, and Narn-Yih Lee. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 194–204. Springer, Berlin, Heidelberg, May 1995. 32

[LK23] Kwangsu Lee and Hyoseung Kim. Two-round multi-signatures from okamoto signatures. *Mathematics*, 11:3223, 07 2023. 32

[LO25] Anja Lehmann and Cavit Özbay. Putting multi into multi-signatures: Tight security for multiple signers. Cryptology ePrint Archive, Paper 2025/2198, 2025. 3

[LOS+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 465–485. Springer, Berlin, Heidelberg, May / June 2006. 32

[MH96] Markus Michels and Patrick Horster. On the risk of disruption in several multiparty signature schemes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '96*, pages 334–345, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. 32

[MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: Extended abstract. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 245–254. ACM Press, November 2001. 32

[MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Des. Codes Cryptogr.*, 87(9):2139–2164, 2019. 1, 32

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, October 1999. 6

[Nat16] National Institute of Standards and Technology (NIST). Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf, December 2016. 2

[NRS21] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 189–221, Virtual Event, August 2021. Springer, Cham. 2, 4, 32

[NRSW20] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1717–1731. ACM Press, November 2020. 2, 4, 6, 32

[OK90] Kazuo Ohta and Kenji Koyama. Meet-in-the-middle attack on digital signature schemes. In Jennifer Seberry and Josef Pieprzyk, editors, *AUSCRYPT'90*, volume 453 of *LNCS*, pages 140–154. Springer, Berlin, Heidelberg, January 1990. 32

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Berlin, Heidelberg, August 1993. 4

[OO98] Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 354–369. Springer, Berlin, Heidelberg, August 1998. 32

[OO99] Kazuo Ohta and Tatsuaki Okamoto. Multi-signature schemes secure against active insider attacks (special section on cryptography and information security). *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 82:21–31, 1999. 32

[Pat09] Jacques Patarin. The "coefficients H" technique (invited talk). In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008*, volume 5381 of *LNCS*, pages 328–345. Springer, Berlin, Heidelberg, August 2009. 14, 33

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Berlin, Heidelberg, May 1996. 2, 5

[PW23] Jiaxin Pan and Benedikt Wagner. Chopsticks: Fork-free two-round multi-signatures from non-interactive assumptions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 597–627. Springer, Cham, April 2023. 2, 4, 32

[PW24] Jiaxin Pan and Benedikt Wagner. Toothpicks: More efficient fork-free two-round multi-signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 460–489. Springer, Cham, May 2024. 4, 5, 32

[QLH12] Haifeng Qian, Xiangxue Li, and Xinli Huang. Tightly secure non-interactive multisignatures in the plain public key model. *Informatica (Vilnius)*, 3, 01 2012. 32

[RY07] Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multi-party signatures against rogue-key attacks. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 228–245. Springer, Berlin, Heidelberg, May 2007. 32

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, New York, August 1990. 4

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. 21

[SSH11] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. Public-key identification schemes based on multivariate quadratic polynomials. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 706–723. Springer, Berlin, Heidelberg, August 2011. 2

[TJZ25] Shaolong Tang, Peng Jiang, and Liehuang Zhu. Glitter: A fully adaptive and tightly secure threshold signature. In Willy Susilo and Josef Pieprzyk, editors, *Information Security and Privacy - 30th Australasian Conference, ACISP 2025, Proceedings*, Lecture Notes in Computer Science, pages 184–204, Germany, 2025. Springer Science and Business Media Deutschland GmbH. Publisher Copyright: © The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2025.; 30th Australasian Conference on Information Security and Privacy, ACISP 2025 ; Conference date: 14-07-2025 Through 16-07-2025. 32

[TSS+24] Kaoru Takemure, Yusuke Sakai, Bagus Santoso, Goichiro Hanaoka, and Kazuo Ohta. More efficient two-round multi-signature scheme with provably secure parameters for standardized elliptic curves. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 107(7):966–988, 2024. 3, 4, 32

[TZ23] Stefano Tessaro and Chenzhi Zhu. Threshold and multi-signature schemes from linear hash functions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 628–658. Springer, Cham, April 2023. 4, 32

Appendix

A More Related Work

In this section, we discuss more related work on multi-signatures and threshold signatures.

Public Key Model in Multi-Signatures. In their seminal work, Itakura and Nakamura [Ita83] introduced the first construction of a multi-signature scheme. Following their work, numerous constructions have been proposed [OK90, Har94, OO98, OO99]. Most early schemes [HMP95, OO99] rely on a trusted dealer for key generation, while others [Har94, LHL95] were later shown to be vulnerable to rogue-key attacks [MH96]. In a rogue-key attack, In such an attack, an adversary can pick its public key as a function of honest users' keys to forge signatures. Subsequent work aimed to avoid trusted setup and rogue-key attacks either by using interactive key generation protocols [MOR01] or by adopting the "knowledge of secret key" (KOSK) assumption [Bol03, LOS+06], which obliges the adversary to reveal its secret key at public key registration. Several schemes [BJ08, DEF+19] instead opt for proofs of possession at public key registration. These models come with strong limitations and we refer the reader to [BN06, RY07] for further discussion on that. The most realistic and now widely accepted model for multi-signatures is the plain public key model introduced by Bellare and Neven [BN06]. In our work, we adopt this model.

Tight Security in Multi-Signatures. We focus on pairing-free multi-signatures with tight security proofs *without* the algebraic group model. However, we note that there are several pairing-free multi-signatures with tight security proofs in the algebraic group model [AB21, NRS21, BD21, LK23]. There are several three-round schemes with tight security proofs [BN06, BJ08, FH21], all of them being based on DDH (the authors in [BJ08] also provide a construction based on CDH), where only [FH21] supports key aggregation and [BJ08] relies on proofs of possession for key generation. There are only few two-round schemes with tight security proofs [PW23, PW24, BW25], all of them being based on DDH, where only [BW25] supports key aggregation. In the pairing-based setting, there are some non-interactive schemes [BNN07, QLH12, BW24a] with tight security proofs, all of them being based on CDH, but none of them support key aggregation.

Other Multi-Signatures. There are many pairing-free three-round schemes [BN06, BCJ08, BDN18, MPSW19, BJ08, FH21, MOR01], all of them being based on standard assumptions (specifically, DDH and DL), where [BDN18, MPSW19, FH21] support key aggregation. There are also many pairing-free two-round schemes [DEF+19, NRS21, NRSW20, TZ23, TSS+24], where most of them support key aggregation, some of them being based on standard assumptions, and few of them [NRS21, TZ23] are partially non-interactive (i.e., the first signing round is message-independent).

Tight Security in Threshold Signatures. We focus on threshold signatures with tight security proofs *without* the algebraic group model. However, we note that there are several threshold signatures with tight security proofs in the algebraic group model [BL22, BLSW24, CKM23]. Note that by definition, threshold signatures support key aggregation, since the public key is already aggregated. There are only few threshold signatures with tight security proofs [Che25, TJZ25, BW24b], all of them being based on DDH. Notably, all of them have adaptive security.

B Postponed Proofs

In this section, we provide the proofs that were omitted from the main body.

B.1 Proofs for Earpick-MS

Lemma 6. *We have* $\Pr[\mathbf{G}_4 \Rightarrow 1] = \Pr[\mathbf{G}_3 \Rightarrow 1]$.

Proof. We show the lemma using the *H-coefficients technique* from Patarin [Pat09, HT16]. For this, let T_0 and T_1 define the random variables for the transcripts[8] of A's interaction in the games $\mathbf{G}_3$ and $\mathbf{G}_4$, respectively. Further, for any possible transcript T, let $P_0(T)$ and $P_1(T)$ define the probabilities of choosing randomness in the respective game that would yield transcript T, after having fixed the signing queries and the random oracle queries in advance. Crucially, these probabilities depend only on the value of T and the game's randomness, but are independent of A. By the H-coefficients technique, indistinguishability between the two games then follows from the assertion $P_0(T) = P_1(T)$, which we will now show.

We can describe a possible transcript T by the tuple

$$\Big(g, h, \mathsf{pk}, \{\mathsf{H}, \mathsf{H}_a, \mathsf{H}_b, \mathsf{H}_c, \mathsf{H}_D, \mathsf{H}_{uv}\}_{[q_\mathrm{h}]},$$
$$\{(Z_1^{(i)}, \pi_1^{(i)}, R_{1,0}^{(i)}, S_{1,0}^{(i)}, R_{1,1}^{(i)}, S_{1,1}^{(i)})\}_{i\in[q_\mathrm{s}]}, \{(z_1^{(i)}, y_1^{(i)})\}_{i\in[q_\mathrm{s}]}\Big),$$

where (by abuse of notation) $\{\mathsf{H}, \ldots, \mathsf{H}_{uv}\}_{[q_\mathrm{h}]}$ denotes all random oracle outputs, $\{(Z_1^{(i)}, \ldots, S_{1,1}^{(i)})\}_{i\in[q_\mathrm{s}]}$ denotes all the first-round messages (output by the honest signer), and $\{(z_1^{(i)}, y_1^{(i)})\}_{i\in[q_\mathrm{s}]}$ denotes all the second-round messages (output by the honest signer). Here, we have ignored the outputs of the adversary, as these are fixed from the very beginning anyway by assumption. We rearrange the above transcript T as (V, W, U), where

$$V := \Big(g, h, \mathsf{pk}, \{\mathsf{H}_{uv}\}_{[q_\mathrm{s}]}\{(R_{1,b}^{(i)}, S_{1,b}^{(i)})\}_{i\in[q_\mathrm{s}]}, \{(z_1^{(i)}, y_1^{(i)})\}_{i\in[q_\mathrm{s}]}\Big),$$
$$W := \big(\{\mathsf{H}, \mathsf{H}_a, \mathsf{H}_b, \mathsf{H}_c, \mathsf{H}_D\}_{[q_\mathrm{h}]}\big), \quad U := \Big(\{(Z_1^{(i)}, \pi_1^{(i)}, R_{1,1-b}^{(i)}, S_{1,1-b}^{(i)})\}_{i\in[q_\mathrm{s}]}\Big).$$

We now take a closer look at the transcripts output in the games $\mathbf{G}_3$ and $\mathbf{G}_4$. For this, recall that the distinction between both games solely lies in the second-round message outputs. Also, it is clear that the random oracle outputs of $\mathsf{H}, \mathsf{H}_a, \mathsf{H}_b, \mathsf{H}_c, \mathsf{H}_D$ are identically distributed in both games and independent of the other outputs in the transcript. Further, for fixed pk, the outputs $(Z_1^{(i)}, \pi_1^{(i)})$ are also identically distributed in both games. This is because the value $Z_1^{(i)}$ is a deterministic function of the output from H_D and the key pk, while $\pi_1^{(i)}$ is simulated in zero-knowledge and thus identically distributed to an honestly generated proof (cf. Proposition 1). Also, it is clear that the elements $(R_{1,1-b}^{(i)}, S_{1,1-b}^{(i)})$ are identically distributed in both games, since these are a deterministic function in the outputs from H_{uv} and the uniformly and independently sampled randomness $(r_{1,1-b}^{(i)}, s_{1,1-b}^{(i)})$ is sampled uniformly at random. Having said that, it is sufficient to show that the marginal random variables defining the transcript V are identically distributed in both games $\mathbf{G}_3$ and $\mathbf{G}_4$. For this, we consider the information from that transcript (by computing the discrete logarithms of the elements to the base g), which is described by the tuple

$$\Big(\underline{\alpha}_h := \mathsf{dlog}(h), \underline{x} := \mathsf{sk}, \{\underline{w}^{(i)}\}_{i\in[q_\mathrm{h}]}, \{(\underline{r}_{1,b}^{(i)} + \underline{x}/\underline{w}^{(i)}\underline{s}_{1,b}^{(i)})\}_{i\in[q_\mathrm{s}]}\{(\underline{z}_1^{(i)}, \underline{y}_1^{(i)})\}_{i\in[q_\mathrm{s}]}\Big).$$

Note that the information $\underline{r}_{1,b}^{(i)} + \underline{x}/\underline{w}^{(i)}\underline{s}_{1,b}^{(i)}$ is already determined by the elements $\underline{x}, \underline{w}^{(i)}, (\underline{z}_1^{(i)}, \underline{y}_1^{(i)})$, and this is true[9] for both games $\mathbf{G}_3$ and $\mathbf{G}_4$. Therefore, we can ignore it for our probability analysis and can consider the simplified transcript

$$\Big(\underline{\alpha}_h, \underline{x}, \{\underline{w}^{(i)}\}_{i\in[q_\mathrm{h}]}, \{(\underline{z}_1^{(i)}, \underline{y}_1^{(i)})\}_{i\in[q_\mathrm{s}]}\Big).$$

[8] By *transcript*, we essentially mean all the information that an unbounded adversary can observe from the outputs of the protocol.

[9] Here we use the identity given by Eq. (1) shown in the main body.

Now consider game $\mathbf{G}_3$. We find that the probability to generate such a potential transcript in game $\mathbf{G}_3$ is given by

$$P_0(V) = \frac{1}{p} \cdot \frac{1}{p} \cdot \frac{1}{(p-1)^{q_\mathsf{h}}} \cdot \frac{1}{(p \cdot p)^{q_\mathsf{s}}}.$$

This can be seen as follows. In order to generate the transcript, we need the identities

$$\alpha_h = \underline{\alpha}_h, \quad x = \underline{x}, \quad w^{(i)} = \underline{w}^{(i)}, \quad (r_{1,b}^{(i)} + a_1 c^{(i)} x, s_{1,b}^{(i)}) = (\underline{z}_1^{(i)}, \underline{y}_1^{(i)}).$$

Since the values $\alpha_h, x \in \mathbb{Z}_p$ and $w^{(i)} \in \mathbb{Z}_p^*$ are sampled uniformly random, the former identities are true with probability inverse $p^2(p-1)^{q_\mathsf{h}}$. On the other hand, once the x is fixed, it is clear that there is a unique pair $(r_{1,b}^{(i)}, s_{1,b}^{(i)}) \in \mathbb{Z}_p^2$ to make the latter identity hold.[10] As such, the latter identity is true with probability inverse p^{2q_s}, yielding our claim on $P_0(V)$ above.

Now consider game $\mathbf{G}_4$ and recall that the tuple $(z_1^{(i)}, y_1^{(i)})$ is computed as $(r_{1,b}^{(i)}, s_{1,b}^{(i)} + a_1 c^{(i)} w^{(i)})$. Here, we find that the probability to generate such a potential transcript is also given by

$$P_1(V) = \frac{1}{p} \cdot \frac{1}{p} \cdot \frac{1}{(p-1)^{q_\mathsf{h}}} \cdot \frac{1}{(p \cdot p)^{q_\mathsf{s}}},$$

which can be seen as follows. Again, in order to generate the transcript, we need the identities

$$\alpha_h = \underline{\alpha}_h, \quad x = \underline{x}, \quad w^{(i)} = \underline{w}^{(i)}, \quad (r_{1,b}^{(i)}, s_{1,b}^{(i)} + a_1 c^{(i)} w^{(i)}) = (\underline{z}_1^{(i)}, \underline{y}_1^{(i)}).$$

From the considerations made before, the probability computation should be clear. Overall, we have shown that $P_0(V) = P_1(V)$, implying $P_0(T) = P_1(T)$ for any potential transcript T. This completes the proof of the lemma. □

B.2 Proofs for Earpick-TS

We present the hybrid games for the security proof of Earpick-TS.

Proof. The proof follows the same roadmap as the proof of Theorem 2.

Game $\mathbf{G}_0$ (Real Game): This is the real security game $\mathsf{TSUF}_{\mathsf{TS}}^{\mathsf{A}}$ for threshold signatures.

Game $\mathbf{G}_1$ (Simulate NIZK Proofs): In this game, we switch from honestly generated proofs π_k for honest $k \in \mathcal{H}$ to simulated proofs $\tilde{\pi}_k$ using the SHVZK simulator Sim.

Game $\mathbf{G}_2$ (Program H_{uv} Correlated): In this game, for every new query $\mathsf{H}_{uv}(b, \mathsf{pk}, \mu)$ with $b \in \{0,1\}$, we sample a uniformly random $\alpha \leftarrow \mathbb{Z}_p$ and then program the random oracle as

$$\mathsf{H}_{uv}(b, \mathsf{pk}, \mu) := (u, v) := (X, Y)^{\alpha}.$$

Note that $(X, Y) \in \mathbb{G}^2$ is the group public key.

Game $\mathbf{G}_3$ (Program H_{uv} via Dual Key): In this game, for every new query $\mathsf{H}_{uv}(b, \mathsf{pk}, \mu)$ with $b \in \{0,1\}$, we sample a uniformly random $w \leftarrow \mathbb{Z}_p^*$ and then program the random oracle as

$$\mathsf{H}_{uv}(b, \mathsf{pk}, \mu) := (X, Y)^{1/w}.$$

We also store the *dual key* w in an (initially empty) list DK as $\mathsf{DK}[b, \mathsf{pk}, \mu] := w$.

Game $\mathbf{G}_4$ (Change Threshold Key Generation): In this game, we change the threshold key generation. Recall that in the previous game, we use the secret key x and generate the secret key shares $\{x_i\}_{i \in [n]}$ using $\mathsf{Share}(x, n, t)$. In this game, we uniformly sample the secret key shares $\{x_i\}_{i \in \mathcal{C}'}$

[10] Note that $c^{(i)}$ is not part of the randomness of the marginal transcript random variable V that we consider here.

for an arbitrary subset $\mathcal{C}' \supseteq \mathcal{C}$ of size t. We then generate the public key share (X_i, Y_i) as $(g,h)^{x_i}$ for every $i \in \mathcal{C}'$. For $i \in [n] \setminus \mathcal{C}'$, we generate (X_i, Y_i) without using x_i by polynomial interpolation on public key shares, namely

$$(X_i, Y_i) = (X, Y)^{\lambda_{i,\mathcal{C}'\cup\{0\},0}} \prod_{j\in\mathcal{C}'} (X_j, Y_j)^{\lambda_{i,\mathcal{C}'\cup\{0\},j}}.$$

The adversary A receives (X,Y), $\{(X_i,Y_i)\}_{i\in[n]}$, and $\{x_i\}_{i\in\mathcal{C}}$ as inputs. In particular, we do not execute $\mathsf{Share}(x,n,t)$ anymore, so x is not used.

Game $\mathbf{G}_5$ (Signing with Dual Key Shares): Recall we programmed $\mathsf{H}_{uv}(b,\mathsf{pk},\mu) := (X,Y)^{1/w}$, where w is a dual key under the main public key (X,Y). However, we need to sign under public key shares, so w cannot be directly used to sign. In the previous game, we have generated (X_k,Y_k) for $k \in [n]\setminus\mathcal{C}'$ as a linear combination of (X,Y) and $\{(X_j,Y_j)\}_{j\in\mathcal{C}'}$. Since we know $\{x_j\}_{j\in\mathcal{C}'}$, (X_k,Y_k) is in fact a linear combination $(g,h)^{\beta}(X,Y)^{\gamma}$ of (g,h) and (X,Y) with known (β,γ). Then from w, we know a dual key $(\beta,\gamma w)$ under (X_k,Y_k), satisfying $(g,h)^{\beta}(u,v)^{\gamma w} = (X_k,Y_k)$. In this game, we sign on behalf of user k by setting $z_k := r_{k,b} + c\lambda_{\mathcal{S},k}\beta$ and $y_k := s_{k,b} + c\lambda_{\mathcal{S},k}\gamma w$.

Game $\mathbf{G}_6$ (Flip Coin for (pk,μ)): In this game, for each query of the form $\mathsf{H}_{uv}(\cdot,\mathsf{pk},\mu)$ with a new μ, we sample a uniformly random bit $b \leftarrow \{0,1\}$ (called *coin*) and set $\mathsf{Coin}[\mathsf{pk},\mu] := b$.

Game $\mathbf{G}_7$ (No Collisions for H_D): In this game, we rule out collisions for the random oracle H_D. Specifically, the game aborts if there are messages $\mu \neq \mu'$ such that $\mathsf{H}_D(\mathsf{pk},\mu) = \mathsf{H}_D(\mathsf{pk},\mu')$.

Game $\mathbf{G}_8$ (Output Random $\tilde{Z}$): In this game, we sample a $\mathsf{Z}[\mathsf{pk},\mu] \leftarrow \mathbb{G}$ uniformly at random for every message μ and force $\tilde{Z} = \mathsf{Z}[\mathsf{pk},\mu]$ in every signing session on μ. To do so, we compute $Z_i := D^{x_i}$ for every $i \in \mathcal{C}'$, and then compute Z_k for $k \in [n]\setminus\mathcal{C}'$ by interpolation. We always output Z_k when signing μ on behalf of user $k \in [n]\setminus\mathcal{C}'$. Note that since the adversary cannot choose public key shares, we do not need the trapdoor d to compute Z_i for corrupt users i. Furthermore, we avoid the need to handle the case where the adversary copies the honest public key, as was necessary in the proof of Earpick-MS.

Game $\mathbf{G}_9$ (No Wrong Z_i for Corrupt Users): In this game, we abort if the adversary A submits an element $Z_i \neq \mathsf{H}_D(\mathsf{pk},\mu)^{x_i}$ along with a verifying NIZK proof π_i as part of its first-round message.

Game $\mathbf{G}_{10}$ (Program H_b via Coin): In this game, we program $\mathsf{H}_b(\tilde{Z}) := \mathsf{Coin}(\mathsf{pk},\mu)$. Note that this is done after $\mathsf{Z}[\mathsf{pk},\mu] \leftarrow \mathbb{G}$ is already defined.

Game $\mathbf{G}_{11}$ (Program H_{uv} for $b \neq \mathsf{Coin}$): In this game, for every new query $\mathsf{H}_{uv}(b,\mathsf{pk},\mu)$ where $b \neq \mathsf{Coin}(\mathsf{pk},\mu)$, we sample a uniformly random $\alpha \leftarrow \mathbb{Z}_p$ and then program the random oracle as $\mathsf{H}_{uv}(b,\mathsf{pk},\mu) := (g,h)^{\alpha}$. Further, α is not stored into the list DK. In particular, the dual key w only exists for the $\mathsf{Coin}(\mathsf{pk},\mu)$-sides of the signing protocol.

Game $\mathbf{G}_{12}$ (Forgery for $b_* = \mathsf{Coin}$): In this game, we abort if $b_* \neq \mathsf{Coin}(\mathsf{pk},\mu_*)$ in the forgery (μ_*,σ_*) output by the adversary A at the end of the game, where $\sigma_* := (b_*, c_*, \tilde{z}_*, \tilde{y}_*)$.

Game $\mathbf{G}_{13}$ (Switch pk to Random): In this game, we change how the public key $\mathsf{pk} = (X,Y)$ is generated. Recall that until now, we generate it by sampling $x \leftarrow \mathbb{Z}_p$ uniformly at random and then computing $\mathsf{pk} := (g^x, h^x)$. In this game, we sample it uniformly random as $\mathsf{pk} := (X,Y) \leftarrow \mathbb{G}^2$.

We conclude by having $\Pr[\mathbf{G}_{13} \Rightarrow 1] \leq \mathsf{negl}(\lambda)$.

□

B.3 Proofs for Earpick-muMS

We present the hybrid games for the security proof of Earpick-muMS.

Proof. The proof follows the same roadmap as the proof of Theorem 2. We only highlight the main differences in the game hop sequence.

Game $\mathbf{G}_0$ (Real Game): This is the real security game $\mathsf{muMSUF}^{\mathsf{A}}_{\mathsf{MS}}$.

Game $\mathbf{G}_{0.1}$ (Generate Public Keys by Rerandomization): We insert a hybrid game into the sequence here. In the key generation phase, this game generates $(X, Y) := (g, h)^x$ with $x \leftarrow \mathbb{Z}_p$. Then it computes the target public keys as $\mathsf{pk}_{*,i} = (X_i, Y_i) := (g, h)_i^{\beta}(X, Y)$ for $i \in [N]$. The corresponding secret key is computed as $\mathsf{sk}_{*,i} = x_i := \beta_i + x$. This does not change the distribution of the public keys.

Game $\mathbf{G}_{0.2}$ (No Collisions for Public Keys): We insert a hybrid game into the sequence here. This game aborts if there exist $i \neq i'$ such that $\mathsf{pk}_{*,i} = \mathsf{pk}_{*,i'}$. This introduces an additive loss of N^2/p to the winning probability.

Game $\mathbf{G}_1$ (Simulate NIZK Proofs): Same as in the proof of Theorem 2, this game simulates the NIZK proof in $\textsc{Sign}_1$ using the SHVZK simulator. This introduces an additive loss of $q_s q_h/p$ to the winning probability.

Game $\mathbf{G}_2$ (Program H_{uv} Correlated): Same as in the proof of Theorem 2, this game programs H_{uv} using $(u, v) := (g, h)^{\alpha}$ with $\alpha \leftarrow \mathbb{Z}_p$. This introduces an additive loss of $\varepsilon_{\mathrm{ddh}} + 1/p$ to the winning probability.

Game $\mathbf{G}_3$ (Program H_{uv} via Dual Key): This game programs the random oracle as $\mathsf{H}_{uv}(b, \mathsf{apk}, \mu) := (X, Y)^{1/w}$ for $w \leftarrow \mathbb{Z}_p^*$. This introduces an additive loss of q_h/p to the winning probability.

Game $\mathbf{G}_4$ (Signing with Dual Key): In this game, $\textsc{Sign}_2$ computes the second-round response message using (β_k, w) instead of $(x_k, 0)$ as the Okamoto signing key. Specifically, let k be the index of the honest signer for this session, and w be retrieved from $\mathsf{DK}(b, \mathsf{apk}, \mu)$ which satisfies $\mathsf{H}_{uv}(b, \mathsf{apk}, \mu) = (X, Y)^{1/w}$. $\textsc{Sign}_2$ computes the response as $\tilde{z}_1 := r_{1,b} + a_1 c \beta_k$ and $\tilde{y}_1 := s_{1,b} + a_1 c w$. This change does not affect the view of A.

Game $\mathbf{G}_5$ (No Collisions for apk): This game aborts if there exists a pair of public key lists $\mathcal{L} \neq \mathcal{L}'$ that have been submitted to H_a such that $\mathcal{L}$ and $\mathcal{L}'$ both contain at least one honest public key, and $\mathsf{KAgg}(\mathcal{L}) = \mathsf{KAgg}(\mathcal{L}')$. This introduces an additive loss of $q_h^2/p + N/p$ to the winning probability, where N/p bounds the probability that any honest public key equals 1.

Game $\mathbf{G}_6$ (Flip Coin for (apk, μ)): Same as in the proof of Theorem 2, this game samples a coin $\mathsf{Coin}(\mathsf{apk}, \mu) \leftarrow \{0, 1\}$ for every query of the form $\mathsf{H}_{uv}(\cdot, \mathsf{apk}, \mu)$ with a new (apk, μ).

Game $\mathbf{G}_7$ (No Collisions for H_D): This game aborts if there are tuples $(\mathcal{L}, \mu, \mathsf{pk}) \neq (\mathcal{L}', \mu', \mathsf{pk}')$ such that $\mathsf{H}_D(\mathcal{L}, \mu, \mathsf{pk}) = \mathsf{H}_D(\mathcal{L}', \mu', \mathsf{pk}')$. This introduces an additive loss of q_h^2/p to the winning probability.

Game $\mathbf{G}_8$ (Output Random Z_1): We introduce an initially empty list Z. Whenever $\mathsf{H}_D[\mathcal{L}, \mu, \mathsf{pk}]$ is newly defined, we also define $\mathsf{Z}[\mathcal{L}, \mu, \mathsf{pk}] \leftarrow \mathbb{G}$ uniformly at random. We let $\textsc{Sign}_1$ output $Z_1 := \mathsf{Z}[\mathcal{L}, \mu, \mathsf{pk}_{*,k}]$ instead of $D_1^{x_k} = D_1^{\beta_k + x}$, where k is the honest-signer index queried to $\textsc{Sign}_1$. This introduces an additive loss of $\varepsilon_{\mathrm{ddh}}$ to the winning probability.

We can build a reduction to DDH similar to that in the proof of Theorem 2. The reduction embeds the DDH instance in g, X and (by rerandomization) every pair of $\mathsf{H}_D[\mathsf{apk}, \mu, \mathsf{pk}]$ and $\mathsf{Z}[\mathsf{apk}, \mu, \mathsf{pk}]$. Here we crucially rely on the fact that in $\mathsf{Earpick\text{-}muMS}$, each signer uses a per-public-key D_i in the signing protocol. Suppose that $\mathcal{L}$ contains multiple honest public keys, say $\mathsf{pk}_{*,k}$ and $\mathsf{pk}_{*,k'}$ where $k \neq k'$. When A queries $\textsc{Sign}_1(k, \mathcal{L}, \mu)$ and $\textsc{Sign}_1(k', \mathcal{L}, \mu)$, using per-public-key D_i allows us to set Z_1 from these two sessions independently. By contrast, for $\mathsf{Earpick\text{-}MS}$, $\textsc{Sign}_1(k, \mathcal{L}, \mu)$ and $\textsc{Sign}_1(k', \mathcal{L}, \mu)$ need to compute Z_1 based on the same D, and we must set these Z_1 to be correlated.

Game $\mathbf{G}_9$ (Adversary Copies Honest Public Keys): This game aborts if A represents an honest public key $\mathsf{pk}_{*,k}$ to submit an element $Z' \neq \mathsf{Z}[\mathcal{L}, \mu, \mathsf{pk}_{*,k}]$ along with a verifying NIZK proof as part of the first-round message. Same as in the proof of Theorem 2, this introduces an additive loss of $\varepsilon_{\mathrm{cdh}} + q_h/p + q_h/p$.

Game $\mathbf{G}_{10}$ (Program H_D with Trapdoor): In this game, each new query $\mathsf{H}_D(\mathcal{L}, \mu, \mathsf{pk})$ is answered with g^d with a trapdoor $d \leftarrow \mathbb{Z}_p$. This does not affect the view of A.

Game $\mathbf{G}_{11}$ (No Wrong Z_i for Dishonest Public Keys): This game aborts if A submits an element $Z_i \neq \mathsf{pk}_i^d$ along with a verifying NIZK proof π_i such that $\mathsf{pk}_i \neq \mathsf{pk}_{*,k}$ for every $k \in [N]$. As in the proof of Theorem 2, this introduces an additive loss of q_h/p.

Game $\mathbf{G}_{12}$ (Program H_b via Coin): This game programs H_b by Coin upon the $\textsc{Sign}_1$ query to the last honest signer in $\mathcal{L}$. Precisely, upon any $\textsc{Sign}_1(k, \mathcal{L}, \mu)$ query, if for every other honest public key $\mathsf{pk}_{*,k'} \in \mathcal{L}$, A has queried $\textsc{Sign}_1(k', \mathcal{L}, \mu)$, then the game programs H_b as follows. Parse $\mathcal{L}$ into $\{\mathsf{pk}_i\}_{i\in[n]}$. For every $i \in [n]$, if pk_i is an honest public key (i.e., $\mathsf{pk}_i \in \{\mathsf{pk}_{*,k}\}_{k\in[N]}$), then let $Z_i := \mathsf{Z}[\mathcal{L}, \mu, \mathsf{pk}_i]$. If pk_i is not an honest public key (i.e., $\mathsf{pk}_i \notin \{\mathsf{pk}_{*,k}\}_{k\in[N]}$), then let $Z_i := X_i^d$, where pk_i is parsed into (X_i, Y_i), and d is the trapdoor embedded in H_D such that $\mathsf{H}_D(\mathcal{L}, \mu, \mathsf{pk}_i) = g^d$. The game programs $\mathsf{H}_b(\{Z_i\}_{i\in[n]}) := \mathsf{Coin}(\mathsf{apk}, \mu)$. As in the proof of Theorem 2, this introduces an additive loss of $q_s q_h/p$.

Game $\mathbf{G}_{13}$ (Program H_{uv} for $b \neq \mathsf{Coin}$): Same as in the proof of Theorem 2, this game answers each new query $\mathsf{H}_{uv}(b, \mathsf{apk}, \mu)$ where $b \neq \mathsf{Coin}(\mathsf{apk}, \mu)$ using $(g, h)^\alpha$, with $\alpha \leftarrow \mathbb{Z}_p$. This introduces an additive loss of $q_h/p + 1/p$.

Game $\mathbf{G}_{14}$ (Forgery for $b_* = \mathsf{Coin}$): Same as in the proof of Theorem 2, the game aborts if $b_* \neq \mathsf{Coin}(\mathsf{apk}_*, \mu_*)$. As it is required that $\mathsf{pk}_{*,k_*} \in \mathcal{L}_*$ and $\textsc{Sign}_1(k_*, \mathcal{L}_*, \mu_*)$ has not been queried, the bit $\mathsf{Coin}(\mathsf{apk}_*, \mu_*)$ has not been programmed into H_b and remains independent of A's view. Thus, this introduces a multiplicative loss of $1/2$.

Game $\mathbf{G}_{15}$ (Switch (X, Y) to Random): This game samples (X, Y) uniformly from $\mathbb{G}^2$ instead of generating them as $(X, Y) := (g, h)^x$ with $x \leftarrow \mathbb{Z}_p$. This introduces an additive loss of $\varepsilon_{\mathrm{ddh}}$. As in the proof of Theorem 2, we have $\Pr[\mathbf{G}_{15} \Rightarrow 1] \leq q_h/p + 2/p$. □