Dazzle: Improved Adaptive Threshold Signatures from DDH

Yanbo Chen*

February 17, 2025

Abstract

The adaptive security of threshold signatures considers an adversary that adaptively corrupts users to learn their secret key shares and states. Crites, Komlo, and Maller (Crypto 2023) proposed Sparkle, the first threshold signature scheme in the pairing-free discrete-log setting to be proved adaptively secure. However, its proof of full adaptive security requires the algebraic group model (AGM) and is based on an interactive assumption. Bacho, Loss, Tessaro, Wagner, and Zhu (Eurocrypt 2024) proposed Twinkle, whose full adaptive security can be based on the standard DDH assumption only.

We propose Dazzle and Dazzle-T, adaptively secure threshold signature schemes based on DDH without the AGM, the same assumption and model as Twinkle. Our schemes improve upon Twinkle in signature size, round complexity, and/or security tightness. In particular, Dazzle and Dazzle-T both have signatures that are shorter than Twinkle by one group element. Regarding the round complexity and tightness, Twinkle is three-round and non-tight. Our Dazzle is two-round and has the same security loss as Twinkle, while Dazzle-T is three-round and fully tight.

We achieve our improvements by optimizing the underlying single-party signature scheme and showing that the single-party scheme can be transformed to a threshold scheme by a simpler transformation than that of Twinkle.

Keywords. Threshold Signatures, Adaptive Security, Pairing-Free, Tightness

1 Introduction

A threshold signature scheme [Des90, DF90] distributes a secret key to a group of signers, such that any t signers can jointly sign a message under the common public key, while any $t-1$ signers cannot. Threshold signature schemes have attracted considerable interest in recent years, mainly for their applications in cryptocurrency. Such new attention has led NIST to put forward the standardization of threshold signature schemes [BP23].

In this work, we follow the line of research on threshold signatures in the pairing-free discrete-log setting, including threshold Schnorr signature schemes [KG20, BCK+22, CGRS23, CKM23, Lin24, KRT24, BLSW24] and other signature schemes [TZ23, BLT+24].[1]

In the security model of threshold signatures, the adversary is allowed to corrupt signers to learn their secret key shares and states. Most schemes have been proved secure against static adversaries who decide which signers to corrupt before any communication. Recent works [CKM23, BLT+24, KRT24, BLSW24] constructed schemes with adaptive security. An adaptive adversary can dynamically choose which parties to corrupt based on previous communication and corruptions.

*University of Ottawa. Email: ychen918@uottawa.ca.

[1] Here we do not include *robust* threshold signature schemes, where the generation of valid signatures is guaranteed even in the presence of misbehaving parties. Achieving this property typically introduces inefficiencies. For further discussion, see Section 1.3.

Existing Adaptive Schemes. Crites, Komlo, and Maller [CKM23] proposed Sparkle, the first threshold signature scheme in the pairing-free discrete-log setting with proofs of adaptive security, but only in the algebraic group model (AGM) [FKL18] from an interactive assumption. Recently, Katsumata, Reichle, and Takemure [KRT24] proposed an adaptively secure scheme from the standard discrete logarithm (DL) assumption without the AGM. The main drawback of their scheme is the five-round signing protocol, compared to the three-round protocol of Sparkle. Both schemes are threshold Schnorr signature schemes, which have a small signature size but suffer from loose security reductions owing to the rewinding technique.

Bacho, Loss, Tessaro, Wagner, and Zhu [BLT+24] proposed Twinkle, a threshold signature scheme that produces non-Schnorr signatures. The security is based on the standard DDH assumption without the AGM, and it has three rounds just like Sparkle. Moreover, the non-Schnorr signature avoids rewinding and thus allows a tighter reduction than Schnorr signature schemes.

We believe that a proof based on DDH without the AGM is satisfactory. However, there is scope for improving Twinkle in the other dimensions that we have mentioned—namely, signature size, round complexity, and tightness of the reduction—even though Twinkle already performs as well as or better than the other two schemes in round complexity and tightness. In more detail:

- Signature size: Twinkle has relatively large signatures compared to Schnorr signatures, consisting of two group elements and three scalars.
- Round complexity: Twinkle has a three-round signing protocol like Sparkle, but the state-of-the-art statically secure schemes have only two rounds [KG20, TZ23].
- Tightness: Twinkle avoids rewinding-based reductions but still suffers from a security loss that is linear in the number of signing queries.

1.1 Our Contribution

We propose Dazzle and Dazzle-T, two threshold signature schemes that are adaptively secure based on DDH without the AGM, the same assumption and model as Twinkle. Our schemes improve upon the drawbacks of Twinkle as follows.

- Signature size: Dazzle and Dazzle-T both produce signatures that are shorter than Twinkle by one group element.
- Round complexity: Dazzle has only two rounds. Dazzle-T has three rounds like Twinkle.
- Tightness: Dazzle-T has a tight security reduction. Dazzle has the same security loss as Twinkle.

We can view the construction of Twinkle as comprising two components: a single-party signature scheme and a transformation from single-party to threshold schemes. We achieve the improvements by the following modifications to the construction:

- Signature size: We optimize the single-party signature scheme underlying Twinkle.
- Round complexity: In the single-party to threshold transformation, Twinkle introduces an additional hashing round at the beginning of the signing protocol, similar to Sparkle. We show that this hashing round is unnecessary for Twinkle and our schemes. A "naive" transformation to a two-round scheme is already secure.
- Tightness: We transform a tight variant of the single-party scheme into a three-round threshold scheme, also without the hashing round. To preserve the tightness, we introduce a novel modification to the naive transformation.

We present more detailed comparisons between our schemes and previous schemes in security and efficiency in Tables 1 and 2, respectively.[2] Notably, Dazzle is the first two-round scheme proved to be adaptively secure in the literature.

Concurrent Work. About two weeks before we submitted this work, Bacho and Wagner [BW24] posted a manuscript that presents Twinkle-T, a tightly secure variant of Twinkle. There is no

[2]Our comparison excludes robust schemes, as they are significantly less efficient. We also omit [Lin24], which focuses on UC security.

Scheme	Adaptive	Assumption	AGM	Loss
FROST [KG20]	✗	AOMDL	✗	$\Theta(q_{\mathrm{h}}/\varepsilon)$
TZ [TZ23]	✗	DLOG	✗	$\Theta(q_{\mathrm{h}}/\varepsilon)$
Sparkle [CKM23]	✓	AOMDL	✓	$\Theta(q_{\mathrm{h}}/\varepsilon)$[3]
KRT [KRT24]	✓	DLOG	✗	$\Theta(q_{\mathrm{h}}/\varepsilon)$[4]
Twinkle [BLT+24]	✓	DDH	✗	$\Theta(q_{\mathrm{s}})$
Dazzle	✓	DDH	✗	$\Theta(q_{\mathrm{s}})$
Dazzle-T	✓	DDH	✗	$\Theta(1)$

Table 1: Comparison between our schemes and existing schemes in security. We compare whether the security proofs of the schemes are against adaptive adversaries, the algebraic assumptions the schemes rely on, whether the proofs are in the AGM, and the security loss. Assuming the existence of an adversary that breaks the unforgeability with advantage ε, the security proof for the scheme constructs a reduction that breaks the algebraic assumption in roughly the same running time with advantage ε'. We say the security loss is L if ε and ε' are related via a bound of $\varepsilon \leq L \cdot \varepsilon'$. We let q_{h} and q_{s} denote the number of random oracle and signing queries, respectively, and ε denote the advantage of an adversary against the scheme.

Scheme	Round	PK	Sig. Size	Comm.
FROST [KG20]	2	$1\langle\mathbb{G}\rangle$	$2\langle\mathbb{Z}_p\rangle$	$2\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle$
TZ [TZ23]	2	$1\langle\mathbb{G}\rangle$	$3\langle\mathbb{Z}_p\rangle$	$2\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$
Sparkle [CKM23]	3	$1\langle\mathbb{G}\rangle$	$2\langle\mathbb{Z}_p\rangle$	$1\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + 2\lambda$
KRT [KRT24]	5	$1\langle\mathbb{G}\rangle$	$2\langle\mathbb{Z}_p\rangle$	$3\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + 4\lambda + \langle\sigma\rangle$
Twinkle [BLT+24]	3	$2\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle + 3\langle\mathbb{Z}_p\rangle$	$6\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle + 2\lambda$
Dazzle	2	$2\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle + 3\langle\mathbb{Z}_p\rangle$	$4\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$
Dazzle-T	3	$2\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle + 3\langle\mathbb{Z}_p\rangle$	$4\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$

Table 2: Comparison between our schemes and existing schemes in efficiency. We compare the number of rounds of the signing protocols, the size of public keys, the size of signatures, and the communication complexity per signer. We let $\langle\mathbb{G}\rangle$ and $\langle\mathbb{Z}_p\rangle$ denote the size of a group element and a scalar, respectively, and λ denote the security parameter. In Sparkle, KRT, and Twinkle, each signer broadcasts a hash commitment of size 2λ. In KRT, each signer additionally broadcasts a random string of length 2λ and a single-party signature, for which we use $\langle\sigma\rangle$ to denote the size.

essential overlap between their and our contribution. They did not improve signature size or round complexity. Their tight security reduction uses a different technique, resulting in larger signatures and higher communication. We construct Dazzle-T from Dazzle, increasing the number of round from 2 to 3. In contrast, they start from three-round Twinkle and achieve tightness without increasing the round complexity while at the cost of signature size and communication.

Specific vs. Generic. We present our results differently from Twinkle [BLT+24]. They proposed a generic construction and then provided two instantiations. The first instantiation is based on the algebraic one-more computational Diffie-Hellman (AOMCDH), a new interactive assumption, and the second is based on DDH. In contrast, we go directly to DDH-based specific schemes, as we view them as the main contribution. Moreover, we think it is easier to see the improved construction

[3] To be fair, we only consider the security loss without the AGM. The row for Sparkle combines two security results. Without the AGM, Sparkle is proved partially adaptively secure with a loose reduction. In the AGM, Sparkle is proved fully adaptively secure with a tight reduction.

[4] Katsumata et al. state their security theorem with a loss of factor $\Theta({q_{\mathrm{h}}}^3/\varepsilon)$ for a stronger security notion. They note that for the weaker notion considered in this work, the loss is only $\Theta(q_{\mathrm{h}}/\varepsilon)$.

when looking at the specific schemes. Nevertheless, our improvements also work for Bacho et al.'s generic construction and therefore the AOMCDH-based scheme. We discuss this in Section 5.

1.2 Technical Overview

The Underlying One-Party Signature Scheme. To construct Dazzle, we thresholdize a suitable one-party signature scheme using a simple transformation. At a high level, the transformation lets each signer produce a normal signature using its secret key share. The individual signatures are interactively aggregated into a single threshold signature under the main key. This transformation underlies the whole line of work on threshold signature and multi-signature schemes for Schnorr(-like) signatures.

To prove the adaptive security of threshold schemes, the security reduction needs to handle corruption queries, which require it to output secret key shares. Moreover, we aim for security from non-interactive assumptions, so the reduction cannot handle corruptions by querying oracles. Therefore, we want the security reduction to know the secret key shares from the beginning. This property should come from the single-party scheme, as the transformation does not provide it. That is, we want the reduction of the single-party scheme to know the secret key from the beginning. In many constructions, including Schnorr signatures, however, the security reduction embeds a hard problem in the public key and thus does not know the secret key. There have been constructions with such a property from DDH in the line of research on tightly secure signatures in the multi-user setting with adaptive corruptions [GJ18, DGJL21]. However, these signatures appear to be difficult to aggregate.

Our scheme is very similar to Twinkle while is shorter by one group element. We note that a similar idea also appeared in a ring signature scheme [LPQ18]. It works over a cyclic group $\mathbb{G}$ of prime order p with generator g. It further takes uniformly random elements h, $\hat{g}$, $\hat{h}$ as public parameters. The secret key consists of random exponents w, $x \leftarrow \mathbb{Z}_p$. The public key is $\left[\begin{smallmatrix} W \\ X \end{smallmatrix}\right] := \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]}$, which denotes $\left[\begin{smallmatrix} g^w \hat{g}^x \\ h^w \hat{h}^x \end{smallmatrix}\right]$. To sign a message μ, the signer hashes μ into group elements $(u, \hat{u}) := \mathsf{H}_1(\mu)$ and computes the ephemeral public key $Y = u^w \hat{u}^x$. Then it produces a Schnorr-like non-interactive proof of knowledge of (w, x) for relation $\left[\begin{smallmatrix} W \\ X \\ Y \end{smallmatrix}\right] := \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]}$. Specifically, the signer samples masking scalars r, $s \leftarrow \mathbb{Z}_p$ and computes the commitment $\left[\begin{smallmatrix} R \\ S \\ T \end{smallmatrix}\right] := \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} r \\ s \end{smallmatrix}\right]}$. It computes the challenge as $c = \mathsf{H}_2(\mu, Y, R, S, T)$. Then it computes the response $\left[\begin{smallmatrix} y \\ z \end{smallmatrix}\right] := \left[\begin{smallmatrix} r \\ s \end{smallmatrix}\right] + c \cdot \left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]$. Finally, the signature is output as $\sigma := (Y, c, y, z)$, where (c, y, z) is the Schnorr-like proof. The verifier recovers the commitment as $\left[\begin{smallmatrix} R \\ S \\ T \end{smallmatrix}\right] := \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} y \\ z \end{smallmatrix}\right]} \cdot \left[\begin{smallmatrix} W \\ X \\ Y \end{smallmatrix}\right]^{-c}$ and checks if c is the correct hash value.

Now we briefly sketch the security proof of this scheme. Suppose that the adversary forges a signature that contains ephemeral public key Y^* on message μ^* with $(u^*, \hat{u}^*) = \mathsf{H}_1(\mu^*)$. A successful forgery implies $Y^* = (u^*)^w (\hat{u}^*)^x$ with overwhelming probability for the following reason. Recall that a signature contains a proof of knowledge of (w, x) for relation $\left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \\ u^* & \hat{u}^* \end{smallmatrix}\right]^{\left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]} = \left[\begin{smallmatrix} W \\ X \\ Y \end{smallmatrix}\right]$. Since $\hat{g}$, h, $\hat{h}$ are uniformly random, $(g, \hat{g})$ and $(h, \hat{h})$ are likely to be non-parallel, i.e., there does not exist $\alpha \in \mathbb{Z}_p$ such that $(h, \hat{h})^\alpha$. In this case, the secret key (w, x) is the only pair of exponents satisfying $\left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]} = \left[\begin{smallmatrix} W \\ X \end{smallmatrix}\right]$. If $Y \neq (u^*)^w (\hat{u}^*)^x$, then there exists no (w, x) satisfying the relation to prove, and by the soundness of Schnorr-like proofs, the adversary is unlikely to forge a proof.

On the other hand, assuming DDH, we can switch to the following indistinguishable hybrid game. Now h, $\hat{g}$, $\hat{h}$ are not uniformly random. Instead, $(h, \hat{h})$ is parallel to $(g, \hat{g})$, which means

$(h, \hat{h}) = (g, \hat{g})^{\alpha}$ for some $\alpha \in \mathbb{Z}_p$. For every signing query that the game needs to answer, $(u, \hat{u})$ is also parallel to $(g, \hat{g})$, but the adversary forges a signature with $(u^*, \hat{u}^*)$ that is non-parallel to $(g, \hat{g})$. As a result, every possible public key (W, X) corresponds to p possible secret keys (w', x') satisfying $\left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right] \left[\begin{smallmatrix} w' \\ x' \end{smallmatrix}\right] = \left[\begin{smallmatrix} W \\ X \end{smallmatrix}\right]$. Moreover, for each signing query, all possible secret keys yield the same $Y = u^{w'} \hat{u}^{x'}$. The actual secret key (w, x) is perfectly hidden among the p possible keys by the witness indistinguishability of Schnorr-like proofs. For the forged signature with $(u^*, \hat{u}^*)$ non-parallel to $(g, \hat{g})$, each possible secret key (w', x') results in a distinct value of $(u^*)^{w'} (\hat{u}^*)^{x'}$. The probability that $Y^* = (u^*)^w (\hat{u}^*)^x$ for the perfectly hidden secret key (w, x) is only $1/p$. Recall that however, in the original security game, a valid forgery must contain $Y^* = (u^*)^w (\hat{u}^*)^x$. Consequently, the adversary is unlikely to win the original game.

The security analysis above relies on DDH to transfer between different distributions of g, $\hat{g}$, h, $\hat{h}$, u, $\hat{u}$, u^*, $\hat{u}^*$, but the DDH instances are never embedded in the public key (W, X). The reduction always knows the secret key (w, x).

From One-Party to Threshold Scheme. Now we show how to transform our one-party scheme to a two-round threshold scheme, resulting in Dazzle. This simple transformation can be viewed as the common starting point of most works in threshold signatures and multi-signatures in the pairing-free discrete-log setting. In the interactive signing protocol, each signer essentially produces a normal signature with its own secret key share. In the first round, they exchange their commitments. Then they calculate an aggregated commitment and hash it to a common challenge. The signers produce and exchange their responses in the second round, and finally an aggregated response is contained in the signature.

Let us specifically describe our resulting scheme Dazzle. The secret key (w, x) is t-out-of-n shared. It can be recovered as a linear combination $\left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right] = \sum_{i \in \mathcal{S}} \lambda_{\mathcal{S},i} \left[\begin{smallmatrix} w_i \\ x_i \end{smallmatrix}\right]$ for every set $\mathcal{S} \subseteq [n]$ of size t. Suppose that a set of signers $\mathcal{S}$ want to jointly sign a message μ. In the first round of the signing protocol, each signer (indexed by k) samples r_k, $s_k \leftarrow \mathbb{Z}_p$, computes $(u, \hat{u}) := \mathsf{H}_1(\mu)$ and then its individual ephemeral public key $Y_k := u^{w_k} \hat{u}^{x_k}$ and commitment $\left[\begin{smallmatrix} R_k \\ S_k \\ T_k \end{smallmatrix}\right] := \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{smallmatrix}\right] \left[\begin{smallmatrix} r_k \\ s_k \end{smallmatrix}\right]$, and broadcasts (Y_k, R_k, S_k, T_k). Then the aggregated ephemeral key and commitment are calculated as $\tilde{Y} := \prod_{i \in \mathcal{S}} Y_i$ and $\left[\begin{smallmatrix} \tilde{R} \\ \tilde{S} \\ \tilde{T} \end{smallmatrix}\right] := \prod_{i \in \mathcal{S}} \left[\begin{smallmatrix} R_i \\ S_i \\ T_i \end{smallmatrix}\right]$. They are hashed to a challenge c. In the second round, each signer computes its individual response $\left[\begin{smallmatrix} y_k \\ z_k \end{smallmatrix}\right] := \left[\begin{smallmatrix} r_k \\ s_k \end{smallmatrix}\right] + c \cdot \lambda_{\mathcal{S},i} \cdot \left[\begin{smallmatrix} w_k \\ x_k \end{smallmatrix}\right]$. The aggregated response is calculated as $\left[\begin{smallmatrix} \tilde{y} \\ \tilde{z} \end{smallmatrix}\right] = \sum_{i \in \mathcal{S}} \left[\begin{smallmatrix} y_i \\ z_i \end{smallmatrix}\right]$. The threshold signature is output as $(\tilde{Y}, c, \tilde{y}, \tilde{z})$.

In many previous works, the above transformation results in insecure schemes that are vulnerable to concurrent attacks [DEF+19, BLL+21], and various techniques have been developed to resolve this issue. However, in our case, the simple transformation, without any modification, already yields a secure threshold scheme. Below, we will explain why the security proofs for previous schemes fail when applying the above transformation and what makes our scheme different.

In the security reduction for many signature schemes in the pairing-free discrete-log setting, including Schnorr signatures, a hard problem is embedded in the public key. The reduction does not know the secret key and applies the honest-verifier zero-knowledge (HVZK) simulator for Schnorr(-like) proofs to answer signing queries. The HVZK simulator must know the challenge before producing the commitment, for which the reduction exploits its ability to program the random oracle. However, the HVZK simulator fails in the two-round interactive schemes given by the above basic transformation. The reduction has to output the commitment in the first round. At this point, the HVZK simulator needs to know the challenge, but the challenge depends on other signers' commitments, which can be chosen by the adversary and cannot be known in advance. Sparkle and Twinkle apply a technique from [BN06] to make the HVZK simulator work, resulting in three-round schemes.

Our key observation is that our scheme (and Twinkle) do not require the HVZK simulator.

Recall that, in order to achieve adaptive security, our reduction knows the secret key shares from the beginning. Instead of relying on the HVZK simulator, our reduction can just sign messages honestly to answer signing queries. We leverage the witness indistinguishability (WI) of Schnorr-like proofs to show that the secret key is perfectly hidden among p possible ones in the hybrid game. WI guarantees that different witnesses yield identically distributed proofs, even for adversarially chosen challenges. Unlike the HVZK simulator, WI can be applied directly in the two-round scheme.

Three-Round Scheme with Tight Security. With a reduction that knows all the secret key shares from the beginning, the security of Dazzle does not degrade with the number of users and corruptions, but it suffers from a security loss of factor q_{s}, the number of signing queries. Recall that in the security analysis for the one-party scheme, we expect the hybrid to answer signing queries with $(u, \hat{u})$ in the same direction of $(g, \hat{g})$ but the adversary to forge a signature with $(u^*, \hat{u}^*)$ in another direction. We use a standard coin tossing argument [Cor00], which introduces a loss of factor q_{s}.

A fully tight scheme can be obtained by adopting the idea of the Chevallier-Mames signature scheme [Che05, KLP17]. We let the signer compute the commitment (R, S) at first. Then it hashes not only the message μ but also (R, S) to obtain $(u, \hat{u}) := \mathsf{H}_2(\mu, R, S)$. It computes the ephemeral public key Y and the remaining commitment T as in the original scheme. Since (R, S) has high entropy, the internal random oracle query $\mathsf{H}_2(\mu, R, S)$ that the reduction makes to answer a signing query is likely to be fresh, i.e., the adversary is very unlikely to have made the same hash query before. Thus, the reduction can answer internal queries $\mathsf{H}_2(\mu, R, S)$ with $(u, \hat{u})$ parallel to $(g, \hat{g})$ and answer external queries from the adversary with $(u^*, \hat{u}^*)$ not parallel to $(g, \hat{g})$. We thus avoid the coin tossing and the resulting security loss.

Next we thresholdize this tightly secure one-party scheme, also with a simple transformation without the additional hashing round. Now the signers need to exchange (R_k, S_k) before calculating $(u, \hat{u})$, so the resulting threshold scheme becomes three-round. In the first round, the signers exchange (R_k, S_k) and hash the aggregated commitment $(\tilde{R}, \tilde{S})$ to obtain $(u, \hat{u})$. They exchange ephemeral keys Y_k and the remaining commitment T_k in the second round and obtain the challenge c. In the third round, they exchange their responses.

However, this direct construction does not preserve tightness. Recall that the single-party scheme is tightly secure because the adversary cannot predict (R, S) that the hybrid uses for signing. If we naively let $\left[\begin{smallmatrix}\tilde{R}\\ \tilde{S}\end{smallmatrix}\right] := \prod_{i\in\mathcal{S}} \left[\begin{smallmatrix}R_i\\ S_i\end{smallmatrix}\right]$ in the interactive setting, the adversary, who decides every (R_i, S_i) except the one output by the hybrid, can fully control $(\tilde{R}, \tilde{S})$. It can first query $\mathsf{H}_2(\mu, \tilde{R}, \tilde{S})$, then make a signing query on μ, and force the aggregated commitment to be $(\tilde{R}, \tilde{S})$. Consequently, our reduction has to sign with $(u^*, \hat{u}^*) := \mathsf{H}_2(\mu, \tilde{R}, \tilde{S})$ that is not parallel to $(g, \hat{g})$, because $\mathsf{H}_2(\mu, \tilde{R}, \tilde{S})$ was externally made first.

Our solution is to shift $(\tilde{R}, \tilde{S})$ by (g^b, h^b) where $b := \mathsf{H}_1(\mathcal{S}, \mu, \{(R_i, S_i)\}_{i\in\mathcal{S}})$. Now the adversary cannot predict $(\tilde{R}, \tilde{S})$ that will be used in the signing query before making a corresponding $\mathsf{H}_1(\mathcal{S}, \mu, \{(R_i, S_i)\}_{i\in\mathcal{S}})$ query. The H_1 query is also unlikely to be made before a corresponding signing query on μ, because $\{(R_i, S_i)\}_{i\in\mathcal{S}}$ should contain (R_k, S_k) output by the signing oracle. As a result, for each opened signing session, the reduction can recognize those H_2 queries that the signing oracle will potentially make in the second round, and it answers them with $(u, \hat{u})$ parallel to $(g, \hat{g})$.

1.3 Other Related Work

Multi-Signatures. Roughly speaking, multi-signature schemes can be viewed as n-out-of-n threshold signature schemes with dynamic groups, where each signer has its own key pair, and any set of signers can jointly sign a message. Multi-signatures in the pairing-free discrete-log setting also attract enormous interest for applications in cryptocurrency [BN06, BCJ08, MPSW19, DEF+19, NRSW20, AB21, NRS21, BD21, TZ23, PW23, PW24]. The technique of an additional hashing

round used by Sparkle and Twinkle is from the Schnorr multi-signature scheme by Bellare and Neven [BN06].

Tightly Secure Multi-Signatures. Adopting the Bellare-Neven scheme [BN06] to the Katz-Wang [KW03] signature scheme gives three-round tightly secure multi-signature and threshold signature schemes based on DDH [FH21], but only against static adversaries. Recently, Pan and Wagner [PW23, PW24] proposed tightly secure two-round multi-signature schemes based on DDH using pseudorandom bit techniques. In their schemes, the verification needs every signer's public key instead of a single group key, so their techniques do not apply to threshold signatures. In summary, our scheme Dazzle-T fills the gap of tightly secure three-round threshold signatures against adaptive adversaries. For two-round schemes, tight security remains open, even against static adversaries.

Robustness. Robust threshold signature schemes guarantee the generation of valid signatures in the presence of a bounded number of misbehaving parties. This property was a primary focus in many early works on threshold signatures [CGJ+99, SS01, AF04, GJKR07]. More recently, there has also been significant interest in robust schemes, particularly in asynchronous networks [RRJ+22, Sho23, BHK+24, GS24, BLSW24]. Adaptive security was considered in [CGJ+99, AF04], and a recent work [BLSW24] presents an asynchronous threshold Schnorr scheme with a proof of adaptive security based on an interactive assumption in the AGM.

Non-robust yet more efficient threshold signature schemes have also gained considerable attention in recent years [KG20, BCK+22, CGRS23, CKM23, Lin24, KRT24, BLSW24]. Our work follows this line of research.

Distributed Key Generation. In this work, key generation is defined as a centralized process. In principle, one can use general multi-party computation protocols. Specifically designed distributed key generation (DKG) protocols [Ped92, CGJ+99, JL00, GJKR07, KMS20, DYX+22, KGS23], which enable more efficient key generation without a trusted dealer, are an important topic.

Readers are referred to [BLT+24] for a list of related work on threshold signatures from different algebraic structures.

2 Preliminaries

Notations. For a positive number n, $[n]$ denotes $\{1, \ldots, n\}$. If x is a variable, then $y := x$ denotes that we assign the value of x to y. If S is a set, then $y \leftarrow S$ denotes that we uniformly sample y from S. If A is a (randomized) algorithm, then $y \in \mathsf{A}(x)$ denotes that y is a possible output of A on input x.

If $g_1, \ldots, g_m$ are group elements, $x_1, \ldots, x_m$ are scalars, then we let the inner product $\begin{bmatrix} g_1 & \cdots & g_m \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$ denote multi-exponentiation $\prod_{i=1}^{m} g_i^{x_i}$. If $\vec{g}_1, \ldots, \vec{g}_n$ are row vectors of group elements, $\vec{x}$ is a column vector of scalars, then we let the matrix multiplication $\begin{bmatrix} \vec{g}_1 \\ \vdots \\ \vec{g}_n \end{bmatrix} \vec{x}$ denote $\begin{bmatrix} \vec{g}_1^{\vec{x}} \\ \vdots \\ \vec{g}_n^{\vec{x}} \end{bmatrix}$.

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

$\underline{\mathsf{Exec}(\mathcal{S}, \{\mathsf{sk}_i\}_{i\in\mathcal{S}}, \mu)}$
1: **for** $j \in \mathcal{S}$ **do**
2: $\quad(\mathsf{st}_j, \mathsf{msg}_j) \leftarrow \mathsf{Sign}(\mathcal{S}, j, \mathsf{sk}_j, \mu)$
3: **end for**
4: **for** $j \in \mathcal{S}$ **do**
5: $\quad\mathsf{msg}'_j \leftarrow \mathsf{Sign}'(\mathcal{S}, j, \mathsf{sk}_j, \mu, \mathsf{st}_j, \{\mathsf{msg}_i\}_{i\in\mathcal{S}})$
6: **end for**
7: **return** $\sigma \leftarrow \mathsf{Combine}(\mathcal{S}, \mu, \{\mathsf{msg}_i\}_{i\in\mathcal{S}}, \{\mathsf{msg}'_i\}_{i\in\mathcal{S}})$

Figure 1: Procedure Exec for defining the completeness of threshold signature schemes.

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

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

Shamir's Secret Sharing. Shamir's secret sharing scheme [Sha79] allows sharing a secret over $\mathbb{Z}_p$. On input the number of users $n < p$, the threshold $t \leq n$, and the secret $x \in \mathbb{Z}_p$, the sharing algorithm $\mathsf{Share}(n, t, x)$ samples t coefficients $\alpha_0, \ldots, \alpha_{t-1} \leftarrow \mathbb{Z}_p$ and returns the secret shares $\{x_i\}_{i\in[n]}$ where $x_i = \sum_{j=0}^{t-1} \alpha_j i^j$. On input $\{x_i\}_{i\in\mathcal{S}}$ where $|\mathcal{S}| = t$, the recovering algorithm $\mathsf{Rec}(\{x_i\}_{i\in\mathcal{S}})$ recovers x by polynomial interpolation. In particular, $x = \sum_{i\in\mathcal{S}} \lambda_{\mathcal{S},i} x_i$, where the Lagrange coefficient for $\mathcal{S}$ is defined as $\lambda_{\mathcal{S},i} = \prod_{j\in\mathcal{S}, j\neq i} j/(i-j)$. Shamir's secret sharing scheme has perfect security.

2.1 Threshold Signatures

We define two-round threshold signature schemes. The key generation in our syntax is assumed to be trusted but can be implemented by a distributed protocol in practice. The signing protocol is described by three algorithms that each signer runs locally. The first stage of signing returns a secret state and a protocol message. After exchanging the protocol messages with each other, each signer runs the second stage of signing to obtain the second protocol message. The last stage is a combining algorithm that takes all the previous protocol messages as inputs and outputs the final threshold signature. The combining algorithm does not take any secret state as input and can be executed by any designated party. An honest execution of the signing protocol is described as the procedure Exec in Fig. 1, which we use to define completeness. The syntax can be easily adapted to three-round schemes, so we omit it.

Definition 2 (Two-Round Threshold Signature Schemes)**.** A two-round threshold signature scheme TS consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{par}$: On input the security parameter 1^λ, the setup algorithm Setup outputs global parameters par. All algorithms related to TS implicitly take par as input.
- $\mathsf{KGen}(n, t) \to (\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]})$: On inputs a signer group size n and a threshold t, the key generation algorithm KGen outputs a public key pk and n secret key shares $\{\mathsf{sk}_i\}_{i\in[n]}$. Each secret key share implicitly indicates n and t.
- The signing protocol consists of three algorithms:
  - $\mathsf{Sign}(\mathcal{S}, k, \mathsf{sk}_k, \mu) \to (\mathsf{st}, \mathsf{msg})$: On inputs a set of signer indices $\mathcal{S}$, a signer index k, a secret key share sk_k, and a message μ, the first-round signing algorithm Sign outputs a state st and a protocol message msg.

  - $\mathsf{Sign}'(\mathcal{S}, k, \mathsf{sk}_k, \mu, \mathsf{st}, \mathcal{M}) \to \mathsf{msg}'$: On inputs a set of signer indices $\mathcal{S}$, a signer index k, a secret key share sk_k, a message μ, a state st, and a set of protocol messages $\mathcal{M}$, the second-round signing algorithm Sign' outputs a protocol message msg'.
  - $\mathsf{Combine}(\mathcal{S}, \mu, \mathcal{M}, \mathcal{M}') \to \sigma$: On inputs a set of signer indices $\mathcal{S}$, a message μ, two sets of protocol messages $\mathcal{M}$, $\mathcal{M}'$, 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 verification algorithm Vf outputs 0 or 1.

For completeness, we require that for all n and t that the scheme supports, all $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}) \in \mathsf{KGen}(n, t)$, all subsets of signers $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| = t$, and all messages μ, we have

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

where the procedure Exec, which models an honest execution of TS, is defined in Fig. 1.

Security Model. We define the (fully) adaptive security of two-round threshold signature schemes. The adversary is given the public parameters, a public key, and oracles that allow corrupting and querying signatures from the signers. It can concurrently open many signing sessions with each signer. When the adversary corrupts a signer, it learns not only the secret key share but also the secret state produced in the first round of each signing session. We set up a counter for each signer which is used to set up session identifiers to keep track of signing sessions. The goal of the adversary is to forge a signature on some message that it has never queried to the signing oracle.

Our security definition models an adversary with fully control over the communication channels. In particular, we do *not* assume *authenticated channels*. As the inputs to the second-stage signing oracle, the adversary decides every protocol message, even on behalf of the honest parties. In fact, honest signers will not realize the signing sessions of other signers. The adversary needs not ensure consistency of protocol messages sent to different signers.

We also do *not* assume *secure erasure*. When the adversary corrupts a signer, it learns all random coins generated by the signer in signing sessions. To account for this, we implicitly require that the state st output by Sign includes all random coins it used.[5]

Definition 3 (Adaptive Security of Threshold Signature Schemes.)**.** A threshold signature scheme TS is $(q_{\mathrm{s}}, \tau, \varepsilon)$-adaptively secure if for all adversaries 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 all n and t that the scheme supports, $\Pr[\mathsf{adp\text{-}UF}^{\mathsf{A}}_{\mathsf{TS}}(\lambda, n, t) = 1] \leq \varepsilon = \varepsilon(\lambda)$, where the security game $\mathsf{adp\text{-}UF}$ is defined in Fig. 2.

3 Two-Round Scheme from DDH

We describe our scheme Dazzle below. We give a formal pseudocode description in Fig. 3. The scheme is defined with respect to a group generation algorithm GrGen. It supports all practical (polynomially bounded) n and t.

Setup. The global parameters consist of group parameters $(\mathbb{G}, p, g)$ generated by GrGen and uniformly random group elements $\hat{g}$, h, $\hat{h}$.

Key Generation. On inputs n and t, the key generation algorithm uniformly samples w and x from $\mathbb{Z}_p$. The public key is computed as $\left[\begin{smallmatrix} W \\ X \end{smallmatrix}\right] := \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]}$. The secret key shares $\{(w_i, x_i)\}_{i\in[n]}$ are t-out-of-n Shamir's secret shares of w and x.

[5] If Sign' is randomized, then the random coins should also be output when the signer is corrupted. We omit this definitional complication, as the last-round signing algorithms of our schemes are deterministic.

$\underline{\mathsf{adp\text{-}UF}^{\mathsf{A}}_{\mathsf{TS}}(\lambda, n, t)}$
$\mathcal{Q} := \emptyset$, $\mathcal{C} := \emptyset$
for $i \in [n]$ **do**
  $\mathsf{ctr}_i := 0$
end for
$\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow \mathsf{KGen}(n, t)$
$(\mu^*, \sigma^*) \leftarrow \mathsf{A}^{\textsc{Sign},\textsc{Sign}',\textsc{Corr}}(\mathsf{pk})$
return $[\![\mu^* \notin \mathcal{Q} \wedge \mathsf{Vf}(\mathsf{pk}, \mu^*, \sigma^*) = 1]\!]$

$\underline{\textsc{Corr}(k)}$
if $|\mathcal{C}| \geq t$ **then**
  return $\perp$
end if
$\mathcal{C} := \mathcal{C} \cup \{k\}$
return $(\mathsf{sk}_k, \{\mathsf{st}_{k,i}\}_{i\in[\mathsf{ctr}_k]})$

$\underline{\textsc{Sign}(\mathcal{S}, k, \mu)}$
if $k \in \mathcal{C}$ **then**
  return $\perp$
end if
$\mathcal{Q} := \mathcal{Q} \cup \{\mu\}$
$\mathsf{ctr}_k := \mathsf{ctr}_k + 1$
$\mathsf{sid} := \mathsf{ctr}_k$
$(\mathsf{st}, \mathsf{msg}) \leftarrow \mathsf{Sign}(\mathcal{S}, k, \mathsf{sk}_k, \mu)$
$\mathsf{st}_{k,\mathsf{sid}} := \mathsf{st}$
$\mathsf{st}'_{k,\mathsf{sid}} := (\mathcal{S}, \mu)$
$\mathsf{round}_{k,\mathsf{sid}} := 1$
return msg

$\underline{\textsc{Sign}'(k, \mathsf{sid}, \mathcal{M})}$
if $k \in \mathcal{C} \vee \mathsf{round}_{k,\mathsf{sid}} \neq 1$ **then**
  return $\perp$
end if
$(\mathcal{S}, \mu) := \mathsf{st}'_{k,\mathsf{sid}}$
$\mathsf{msg}' \leftarrow \mathsf{Sign}'(\mathcal{S}, k, \mathsf{sk}_k, \mu, \mathsf{st}_{k,\mathsf{sid}}, \mathcal{M})$
$\mathsf{round}_{k,\mathsf{sid}} := 2$
return msg'

Figure 2: The adp-UF security for defining the adaptive security of threshold signature schemes.

Signing Protocol. Suppose a group $\mathcal{S}$ of signers want to jointly sign a message μ. In the first stage, each participant (whose index is denoted by k) hashes the message to obtain $(u, \hat{u}) := \mathsf{H}_1(\mu)$. It computes its ephemeral public key $Y_k := u^{w_k}\hat{u}^{x_k}$ and commitment $\begin{bmatrix} R_k \\ S_k \\ T_k \end{bmatrix} := \begin{bmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{bmatrix}^{\begin{bmatrix} r_k \\ s_k \end{bmatrix}}$ where r_k, $s_k \leftarrow \mathbb{Z}_p$. It sends (Y_k, R_k, S_k, T_k) as its protocol message.

In the second stage, the signer aggregates all protocol messages it receives into the aggregated ephemeral public key $\tilde{Y} := \prod_{i\in\mathcal{S}} Y_i$ and the aggregated commitment $\begin{bmatrix} \tilde{R} \\ \tilde{S} \\ \tilde{T} \end{bmatrix} := \prod_{i\in\mathcal{S}} \begin{bmatrix} R_i \\ S_i \\ T_i \end{bmatrix}$. It hashes them and the message μ to get the challenge $c := \mathsf{H}_2(\mu, \tilde{Y}, \tilde{R}, \tilde{S}, \tilde{T})$. It computes its response $\begin{bmatrix} y_k \\ z_k \end{bmatrix} := \begin{bmatrix} r_k \\ s_k \end{bmatrix} + c \cdot \lambda_{\mathcal{S},k} \cdot \begin{bmatrix} w_k \\ x_k \end{bmatrix}$, where $\lambda_{\mathcal{S},k}$ is the corresponding Lagrange coefficient for $\mathcal{S}$. It sends the response as its protocol message.

The last stage can be run by a designated combiner without any secret key and state. The combiner aggregates the first-round protocol messages to recompute the ephemeral public key $\tilde{Y}$ and the challenge c. It also aggregates the second-round protocol messages into the aggregated response $\begin{bmatrix} \tilde{y} \\ \tilde{z} \end{bmatrix} := \sum_{i\in\mathcal{S}} \begin{bmatrix} y_i \\ z_i \end{bmatrix}$. It outputs the threshold signature $\sigma := (\tilde{Y}, c, \tilde{y}, \tilde{z})$.

Verification. The verifier recovers the commitment as $\begin{bmatrix} \tilde{R} \\ \tilde{S} \\ \tilde{T} \end{bmatrix} := \begin{bmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{bmatrix}^{\begin{bmatrix} \tilde{y} \\ \tilde{z} \end{bmatrix}} \cdot \begin{bmatrix} W \\ X \\ \tilde{Y} \end{bmatrix}^{-c}$ and checks if $c = \mathsf{H}_2(\mu, \tilde{Y}, \tilde{R}, \tilde{S}, \tilde{T})$.

Theorem 1. *If* DDH *is* $(\tau_{\mathrm{ddh}}, \varepsilon_{\mathrm{ddh}})$*-hard for* GrGen*, then* Dazzle *is* $(q_{\mathrm{s}}, \tau_{\mathrm{uf}}, \varepsilon_{\mathrm{uf}})$*-adaptively secure in the random oracle model against any adversary that makes at most* q_{h} *hash queries, where essentially* $\tau_{\mathrm{ddh}} \approx \tau_{\mathrm{uf}}$ *and*

$$\varepsilon_{\mathrm{ddh}} \geq \frac{1}{4q_{\mathrm{s}}} \cdot (\varepsilon_{\mathrm{uf}} - \frac{q_{\mathrm{h}} + q_{\mathrm{s}} + 2}{p}) - \frac{q_{\mathrm{h}} + 3}{p}.$$

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

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

$\hat{g},\ h,\ \hat{h} \leftarrow \mathbb{G}$

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

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

$w,\ x \leftarrow \mathbb{Z}_p$

$\left[\begin{smallmatrix} W \\ X \end{smallmatrix}\right] := \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]}$

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

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

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

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

    $\mathsf{sk}_i := (w_i, x_i)$

end for

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

$\underline{\mathsf{Sign}(\mathcal{S}, k, \mathsf{sk}_k, \mu)}$

$(w_k, x_k) := \mathsf{sk}_k$

$(u, \hat{u}) := \mathsf{H}_1(\mu)$

$Y_k := u^{w_k}\hat{u}^{x_k}$

$r_k,\ s_k \leftarrow \mathbb{Z}_p$

$\left[\begin{smallmatrix} R_k \\ S_k \\ T_k \end{smallmatrix}\right] := \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} r_k \\ s_k \end{smallmatrix}\right]}$

$\mathsf{st} := (r_k, s_k)$

$\mathsf{msg} := (Y_k, R_k, S_k, T_k)$

return $(\mathsf{st}, \mathsf{msg})$

$\underline{\mathsf{Sign}'(\mathcal{S}, k, \mathsf{sk}_k, \mu, \mathsf{st}, \mathcal{M})}$

$(w_k, x_k) := \mathsf{sk}_k$

$(r_k, s_k) := \mathsf{st}$

$\{\mathsf{msg}_i\}_{i\in\mathcal{S}} := \mathcal{M}$

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

    $(Y_i, R_i, S_i, T_i) := \mathsf{msg}_i$

end for

$\tilde{Y} := \prod_{i\in\mathcal{S}} Y_i$

$\left[\begin{smallmatrix} \tilde{R} \\ \tilde{S} \\ \tilde{T} \end{smallmatrix}\right] := \prod_{i\in\mathcal{S}} \left[\begin{smallmatrix} R_i \\ S_i \\ T_i \end{smallmatrix}\right]$

$c := \mathsf{H}_2(\mu, \tilde{Y}, \tilde{R}, \tilde{S}, \tilde{T})$

$\left[\begin{smallmatrix} y_k \\ z_k \end{smallmatrix}\right] := \left[\begin{smallmatrix} r_k \\ s_k \end{smallmatrix}\right] + c \cdot \lambda_{\mathcal{S},k} \cdot \left[\begin{smallmatrix} w_k \\ x_k \end{smallmatrix}\right]$

return (y_k, z_k)

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

$\{\mathsf{msg}_i\}_{i\in\mathcal{S}} := \mathcal{M}$

$\{\mathsf{msg}'_i\}_{i\in\mathcal{S}} := \mathcal{M}'$

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

    $(Y_i, R_i, S_i, T_i) := \mathsf{msg}_i$

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

end for

$\tilde{Y} := \prod_{i\in\mathcal{S}} Y_i$

$\left[\begin{smallmatrix} \tilde{R} \\ \tilde{S} \\ \tilde{T} \end{smallmatrix}\right] := \prod_{i\in\mathcal{S}} \left[\begin{smallmatrix} R_i \\ S_i \\ T_i \end{smallmatrix}\right]$

$c := \mathsf{H}_2(\mu, \tilde{Y}, \tilde{R}, \tilde{S}, \tilde{T})$

$\left[\begin{smallmatrix} \tilde{y} \\ \tilde{z} \end{smallmatrix}\right] := \sum_{i\in\mathcal{S}} \left[\begin{smallmatrix} y_i \\ z_i \end{smallmatrix}\right]$

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

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

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

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

$(u, \hat{u}) := \mathsf{H}_1(\mu)$

$\left[\begin{smallmatrix} \tilde{R} \\ \tilde{S} \\ \tilde{T} \end{smallmatrix}\right] := \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} \tilde{y} \\ \tilde{z} \end{smallmatrix}\right]} \cdot \left[\begin{smallmatrix} W \\ X \\ \tilde{Y} \end{smallmatrix}\right]^{-c}$

return $[\![c = \mathsf{H}_2(\mu, \tilde{Y}, \tilde{R}, \tilde{S}, \tilde{T})]\!]$

Figure 3: Scheme description of Dazzle.

Proof. Toward the contradiction, we assume that some adversary A breaks the $(q_s, \tau_{uf}, \varepsilon_{uf})$-adaptive security of Dazzle. We will define a series of hybrids, also shown in Fig. 4.

$\underline{\mathsf{G}_0\text{-}\mathsf{G}_3}$
$\mathcal{Q} := \emptyset$, $\mathcal{C} := \emptyset$
for $i \in [n]$ **do**
  $\mathsf{ctr}_i := 0$
end for
$(\mathbb{G}, p, g) \leftarrow \mathsf{GrGen}(1^\lambda)$
▷ —$\mathsf{KGen}(n, t)$—
$h, \hat{g}, \hat{h} \leftarrow \mathbb{G}$ ▷ G_0–G_2
$h \leftarrow \mathbb{G}$ ▷ G_3
$\alpha \leftarrow \mathbb{Z}_p$ ▷ G_3
$\hat{g} := g^\alpha$, $\hat{h} := h^\alpha$ ▷ G_3
$w, x \leftarrow \mathbb{Z}_p$
$\left[\begin{smallmatrix} W \\ X \end{smallmatrix}\right] := \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right] \left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]$
$\{w_i\}_{i\in[n]} \leftarrow \mathsf{Share}(n, t, w)$
$\{x_i\}_{i\in[n]} \leftarrow \mathsf{Share}(n, t, x)$
$\mathsf{pk} := (W, X)$
▷ ——
$(\mu^*, \sigma^*) \leftarrow \mathsf{A}^{\textsc{Sign},\textsc{Sign}',\textsc{Corr},\mathsf{H}_1,\mathsf{H}_2}(\mathsf{pk})$
return $[\![\mu^* \notin \mathcal{Q} \wedge \mathsf{Vf}(\mathsf{pk}, \mu^*, \sigma^*) = 1]\!]$

$\underline{\textsc{Corr}(k)}$
if $|\mathcal{C}| \geq t - 1$ **then**
  return $\perp$
end if
$\mathcal{C} := \mathcal{C} \cup \{k\}$
return $((w_k, x_k), \{\mathsf{st}_{k,i}\}_{i\in[\mathsf{ctr}_k]})$

$\underline{\textsc{Sign}(\mathcal{S}, k, \mu)}$
if $k \in \mathcal{C}$ **then**
  return $\perp$
end if
$\mathcal{Q} := \mathcal{Q} \cup \{\mu\}$
$\mathsf{ctr}_k := \mathsf{ctr}_k + 1$
$\mathsf{sid} := \mathsf{ctr}_k$
▷ —$\mathsf{Sign}(\mathcal{S}, k, \mathsf{sk}_k, \mu)$—
$(u, \hat{u}) := \mathsf{H}_1(\mu)$
$Y_k := u^{w_k} \hat{u}^{x_k}$
$r_k, s_k \leftarrow \mathbb{Z}_p$
$\left[\begin{smallmatrix} R_k \\ S_k \\ T_k \end{smallmatrix}\right] := \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{smallmatrix}\right] \left[\begin{smallmatrix} r_k \\ s_k \end{smallmatrix}\right]$
$\mathsf{st} := (\mathcal{S}, \mu, r_k, s_k)$
▷ ——
$\mathsf{st}_{k,\mathsf{sid}} := \mathsf{st}$
$\mathsf{st}'_{k,\mathsf{sid}} := (\mathcal{S}, \mu)$
$\mathsf{round}_{k,\mathsf{sid}} := 1$
return (Y_k, R_k, S_k, T_k)

$\underline{\textsc{Sign}'(k, \mathsf{sid}, \mathcal{M})}$
if $k \in \mathcal{C} \vee \mathsf{round}_{k,\mathsf{sid}} \neq 1$ **then**
  return $\perp$
end if
$(\mathcal{S}, \mu) := \mathsf{st}'_{k,\mathsf{sid}}$
▷ —$\mathsf{Sign}'(\mathcal{S}, k, \mathsf{sk}_k, \mu, \mathsf{st}_{k,\mathsf{sid}}, \mathcal{M})$—
$(r_k, s_k) := \mathsf{st}_{k,\mathsf{sid}}$
$\{\mathsf{msg}_i\}_{i\in\mathcal{S}\setminus\{k\}} := \mathcal{M}$
for $i \in \mathcal{S} \setminus \{k\}$ **do**
  $(Y_i, R_i, S_i, T_i) := \mathsf{msg}_i$
end for
$\tilde{Y} := \prod_{i\in\mathcal{S}} Y_i$, $\tilde{R} := \prod_{i\in\mathcal{S}} R_i$
$\tilde{S} := \prod_{i\in\mathcal{S}} S_i$, $\tilde{T} := \prod_{i\in\mathcal{S}} T_i$
$c := \mathsf{H}_2(\mu, \tilde{Y}, \tilde{R}, \tilde{S}, \tilde{T})$
$\left[\begin{smallmatrix} y_k \\ z_k \end{smallmatrix}\right] := \left[\begin{smallmatrix} r_k \\ s_k \end{smallmatrix}\right] + c \cdot \lambda_{\mathcal{S},k} \cdot \left[\begin{smallmatrix} w_k \\ x_k \end{smallmatrix}\right]$
▷ ——
$\mathsf{round}_{k,\mathsf{sid}} := 2$
return (y_k, z_k)

$\underline{\mathsf{H}_1(\mu)}$
if $\mathcal{T}(\mu) = \perp$ **then**
  With prob. $\frac{q_s}{q_s+1}$: $\mathsf{B}(\mu) := 1$ ▷ G_1–G_3
  Otherwise $\mathsf{B}(\mu) := 0$ ▷ G_1–G_3
  $\mathcal{T}(\mu) \leftarrow \mathbb{G}^2$ ▷ G_0–G_1
  if $\mathsf{B}(\mu) = 1$ **then** ▷ G_2–G_3
    $d, e \leftarrow \mathbb{Z}_p$ ▷ G_2–G_3
    $u := g^d h^e$, $\hat{u} := \hat{g}^d \hat{h}^e$ ▷ G_2–G_3
    $\mathcal{T}(\mu) := (u, \hat{u})$ ▷ G_2–G_3
  else ▷ G_2–G_3
    $\mathcal{T}(\mu) \leftarrow \mathbb{G}^2$ ▷ G_2–G_3
  end if ▷ G_2–G_3
end if
return $\mathcal{T}(\mu)$

Figure 4: Hybrid games G_0–G_3. H_2 is a normal random oracle.

G_0. Let G_0 denote the original security game adp-UF against Dazzle. Suppose that A outputs message μ^* and forged signature $\sigma^* = (\tilde{Y}^*, c^*, \tilde{y}^*, \tilde{z}^*)$. Let event WIN indicate the winning of A (i.e. the event that $\mathsf{adp\text{-}UF}^{\mathsf{A}}_{\mathsf{TS}}(\lambda, n, t)$ returns 1). By the assumption, we have $\Pr[\text{WIN} \mid \mathsf{G}_0] = \varepsilon_{\text{uf}}$. Let event CORRECTY indicate that $\tilde{Y}^* = (u^*)^w(\hat{u}^*)^x$, where $(u^*, \hat{u}^*) = \mathsf{H}_1(\mu^*)$.

Claim. $\varepsilon_{\text{uf}} \leq \Pr[\text{WIN} \wedge \text{CORRECTY} \mid \mathsf{G}_0] + (q_{\text{h}} + q_{\text{s}} + 2)/p$.

Proof of Claim. Define BADCHAL as the following event: there exists a hash query $c = \mathsf{H}_2(\mu, \tilde{Y}, \tilde{R}, \tilde{S}, \tilde{T})$ that has been made and satisfies:

- $\tilde{Y} \neq u^w \hat{u}^x$; and
- there exists $(\tilde{y}, \tilde{z})$ such that $\begin{bmatrix} \tilde{R} \\ \tilde{S} \\ \tilde{T} \end{bmatrix} = \begin{bmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{bmatrix}^{\left[\begin{smallmatrix} \tilde{y} \\ \tilde{z} \end{smallmatrix}\right]} \cdot \begin{bmatrix} W \\ X \\ \tilde{Y} \end{bmatrix}^{-c}$,

where $(u, \hat{u}) = \mathsf{H}_1(\mu)$.

Let us bound the probability of BADCHAL. We claim that: if $(g, \hat{g}, h, \hat{h})$ is not a DH tuple, and $\tilde{Y} \neq u^w \hat{u}^x$, then there is at most one value of c that makes the second condition holds. Therefore, each query satisfies the two conditions with probability at most $1/p$. There are at most $q_{\text{h}} + q_{\text{s}} + 1$ queries to H_2, including at most q_{h} external queries from A and at most $q_{\text{s}} + 1$ internal queries in SIGN and the final verification. The probability that $(g, \hat{g}, h, \hat{h})$ is a DH tuple is $1/p$. By the union bound, we have $\Pr[\text{BADCHAL} \mid \mathsf{G}_0] \leq (q_{\text{h}} + q_{\text{s}} + 2)/p$.

To see the above claim, assume that there are two such values $c \neq c'$. Then there exist $(\tilde{y}, \tilde{z})$ and $(\tilde{y}', \tilde{z}')$ such that

$$\begin{bmatrix} \tilde{R} \\ \tilde{S} \\ \tilde{T} \end{bmatrix} = \begin{bmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{bmatrix}^{\begin{bmatrix} \tilde{y} \\ \tilde{z} \end{bmatrix}} \cdot \begin{bmatrix} W \\ X \\ \tilde{Y} \end{bmatrix}^{-c} = \begin{bmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{bmatrix}^{\begin{bmatrix} \tilde{y}' \\ \tilde{z}' \end{bmatrix}} \cdot \begin{bmatrix} W \\ X \\ \tilde{Y} \end{bmatrix}^{-c'}.$$

It follows that

$$\begin{bmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{bmatrix}^{\begin{bmatrix} y \\ z \end{bmatrix}} = \begin{bmatrix} W \\ X \\ \tilde{Y} \end{bmatrix},$$

where $y = (\tilde{y} - \tilde{y}')/(c - c')$, $z = (\tilde{z} - \tilde{z}')/(c - c')$. Since $(g, \hat{g}, h, \hat{h})$ is not a DH tuple, $\left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} y \\ z \end{smallmatrix}\right]} = \left[\begin{smallmatrix} W \\ X \end{smallmatrix}\right]$ implies $(y, z) = (w, x)$. Therefore, we have $\tilde{Y} = u^w \hat{u}^x$.

On the other hand, if A wins, and $\tilde{Y}^* \neq (u^*)^w(\hat{u}^*)^x$, then event BADCHAL must occur. To see this, just note that the second condition is necessary to pass the final verification. Therefore, we have

$$\varepsilon_{\text{uf}} \leq \Pr[\text{WIN} \wedge \text{CORRECTY} \vee \text{BADCHAL} \mid \mathsf{G}_0] \leq \Pr[\text{WIN} \wedge \text{CORRECTY} \mid \mathsf{G}_0] + \frac{q_{\text{h}} + q_{\text{s}} + 2}{p}.$$

□

G_1. In G_1, for each fresh hash query $\mathsf{H}_1(\mu)$, a biased coin $\mathsf{coin}(\mu)$ is tossed, with probability ρ to be 1 and $1 - \rho$ to be 0. Since this modification does not change the view of A, we have

$$\Pr[\text{WIN} \wedge \text{CORRECTY} \mid \mathsf{G}_1] = \Pr[\text{WIN} \wedge \text{CORRECTY} \mid \mathsf{G}_0] \tag{1}$$

Define GOODTOSSES as the event that: for each μ that A makes signing queries on, $\mathsf{coin}(\mu) = 1$, while $\mathsf{coin}(\mu^*) = 0$. Note that WIN requires that A has never made signing queries on μ^*. In this

case, GoodTosses requires the at most q_{s} coins corresponding to signing queries to be 1 and a different coin $\mathsf{B}(\mu^*)$ to be 0. Therefore,

$$\begin{aligned}&\Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodTosses} \mid \mathsf{G}_1]\\ &= \Pr[\textsc{GoodTosses} \mid \mathsf{G}_1, \textsc{Win}] \cdot \Pr[\textsc{Win} \wedge \textsc{CorrectY} \mid \mathsf{G}_1]\\ &\geq \rho^{q_{\mathrm{s}}}(1-\rho) \cdot \Pr[\textsc{Win} \wedge \textsc{CorrectY} \mid \mathsf{G}_1].\end{aligned}$$

We set $\rho = q_{\mathrm{s}}/(q_{\mathrm{s}}+1)$ to maximize $\rho^{q_{\mathrm{s}}}(1-\rho)$ to

$$\rho^{q_{\mathrm{s}}}(1-\rho) = \frac{1}{q_{\mathrm{s}}}(1-\frac{1}{q_{\mathrm{s}}+1})^{q_{\mathrm{s}}+1} \geq \frac{1}{4q_{\mathrm{s}}}.$$

Then we have

$$\Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodTosses} \mid \mathsf{G}_1] \geq \frac{1}{4q_{\mathrm{s}}} \cdot \Pr[\textsc{Win} \wedge \textsc{CorrectY} \mid \mathsf{G}_1]. \tag{2}$$

G_2. In G_2, we change the way to answer $\mathsf{H}_1(\mu)$ queries depending on $\mathsf{coin}(\mu)$. If $\mathsf{coin}(\mu) = 0$, we still uniformly sample $(u, \hat{u}) \leftarrow \mathbb{G}^2$. However, if $\mathsf{coin}(\mu) = 1$, then we sample a, $b \leftarrow \mathbb{Z}_p$ and let $\left[\begin{smallmatrix} u & \hat{u} \end{smallmatrix}\right] = \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right]}$. If $(g, \hat{g}, h, \hat{h})$ is not a DH tuple, which is true except with probability $1/p$, then $(u, \hat{u})$ sampled in this way is still uniformly distributed over $\mathbb{G}^2$, so G_2 is identical to G_1. Thus, we have

$$\begin{aligned}&\Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodTosses} \mid \mathsf{G}_2]\\ &\geq \Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodTosses} \mid \mathsf{G}_1] - \frac{1}{p}.\end{aligned} \tag{3}$$

G_3. In G_3, we sample $(\hat{g}, h, \hat{h})$ such that $(g, \hat{g}, h, \hat{h})$ is a DH tuple. We can construct a distinguisher against DDH that uses the input tuple as $(g, \hat{g}, h, \hat{h})$, performs the games, and verifies the occurrence of event $\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodTosses}$. It follows that

$$\begin{aligned}&\Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodTosses} \mid \mathsf{G}_3]\\ &\geq \Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodTosses} \mid \mathsf{G}_2] - \varepsilon_{\mathrm{ddh}}.\end{aligned} \tag{4}$$

Combining the following claim and Eqs. (1) to (4) completes this proof.

Claim. $\Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodTosses} \mid \mathsf{G}_3] \leq (q_{\mathrm{h}}+2)/p$.

□

Proof of Claim. We continue defining more hybrids. Our final target is a hybrid where the sampling of w and x are deferred to the end of the game, which makes CorrectY unlikely. The arguments will be all statistical, so the hybrids are not necessary to be efficient. Fig. 5 shows these hybrids.

G_4. In G_4, we change the way to answer signing queries and corruption queries. Recall that $(g, \hat{g}, h, \hat{h})$ is a DH tuple, and suppose that $(\hat{g}, \hat{h}) = (g^\alpha, h^\alpha)$. We let the signing oracles use $\delta_k = w_k + \alpha x_k$, instead of directly use (w_k, x_k), to sign. However, they can only answer queries that do not violate event GoodTosses, i.e. signing queries on message μ with $\mathsf{coin}(\mu) = 1$.

Since $\mathsf{coin}(\mu) = 1$, $(u, \hat{u})$ is parallel to $(g, \hat{g})$. Suppose $u = g^\beta$ and $\hat{u} = \hat{g}^\beta = g^{\alpha\beta}$. Oracle Sign returns $(Y_k, R_k, S_k, T_k) = (g^{\delta_k \beta}, g^\gamma, h^\gamma, u^\gamma)$ with $\gamma \leftarrow \mathbb{Z}_p$. Oracle Sign$'$ of the same signing session returns $z_k \leftarrow \mathbb{Z}_p$ and $y_k = \gamma + c \cdot \lambda_{\mathcal{S},k} \cdot \delta_k - \alpha z_k$.

Note that now the secret state of signing sessions, (r_k, s_k), are missing. We change the corruption oracle Corr accordingly to generate (r_k, s_k) for each signing session at the time of corruption. If the signing session is already completed, then (r_k, s_k) is computed as $(y_k - c \cdot \lambda_{\mathcal{S},k} \cdot w_k, z_k - c \cdot \lambda_{\mathcal{S},k} \cdot x_k)$.

$\underline{\mathsf{G}_4\text{–}\mathsf{G}_5}$

$\mathcal{Q} := \emptyset$, $\mathcal{C} := \emptyset$
for $i \in [n]$ **do**
  $\mathsf{ctr}_i := 0$
end for
$(\mathbb{G}, p, g) \leftarrow \mathsf{GrGen}(1^\lambda)$
$h \leftarrow \mathbb{G}$
$\alpha \leftarrow \mathbb{Z}_p$
$\hat{g} := g^\alpha$, $\hat{h} := h^\alpha$
$w, x \leftarrow \mathbb{Z}_p$ ▷ G_4
$\left[\begin{smallmatrix} W \\ X \end{smallmatrix}\right] := \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]}$ ▷ G_4
$\{w_i\}_{i\in[n]} \leftarrow \mathsf{Share}(n, t, w)$ ▷ G_4
$\{x_i\}_{i\in[n]} \leftarrow \mathsf{Share}(n, t, x)$ ▷ G_4
for $i \in [n]$ **do** ▷ G_4
  $\delta_i := w_i + \alpha x_i$ ▷ G_4
end for ▷ G_4
$\delta \leftarrow \mathbb{Z}_p$ ▷ G_5
$\left[\begin{smallmatrix} W \\ X \end{smallmatrix}\right] := \left[\begin{smallmatrix} g \\ h \end{smallmatrix}\right]^{\delta}$ ▷ G_5
$\{\delta_i\}_{i\in[n]} \leftarrow \mathsf{Share}(n, t, \delta)$ ▷ G_5
$\mathsf{pk} := (W, X)$
$(\mu^*, \sigma^*) \leftarrow \mathsf{A}^{\textsc{Sign},\textsc{Sign}',\textsc{Corr},\mathsf{H}_1,\mathsf{H}_2}(\mathsf{pk})$
$x \leftarrow \mathbb{Z}_p$ ▷ G_5
$w \leftarrow \delta - \alpha x$ ▷ G_5
return $[\![\mu^* \notin \mathcal{Q} \wedge \mathsf{Vf}(\mathsf{pk}, \mu^*, \sigma^*) = 1]\!]$

$\underline{\textsc{Corr}(k)}$

if $|\mathcal{C}| \geq t - 1$ **then**
  return $\perp$
end if
$\mathcal{C} := \mathcal{C} \cup \{k\}$
$x_k \leftarrow \mathbb{Z}_p$ ▷ G_5
$w_k := \delta_k - \alpha x_k$ ▷ G_5
for $i \in \mathsf{ctr}_k$ **do**
  if $\mathsf{round}_{k,\mathsf{sid}} = 2$ **then**
    $(y_k, z_k) := \mathsf{rsp}_{k,\mathsf{sid}}$
    $(\mathcal{S}, \mu) := \mathsf{st}'_{k,\mathsf{sid}}$
    $r_k := y_k - c \cdot \lambda_{\mathcal{S},k} \cdot w_k$
    $z_k := z_k - c \cdot \lambda_{\mathcal{S},k} \cdot x_k$
  else
    $s_k \leftarrow \mathbb{Z}_p$
    $r_k := \gamma - \alpha s_k$
  end if
  $\mathsf{st}_{k,\mathsf{sid}} := (r_k, s_k)$
end for
return $((w_k, x_k), \{\mathsf{st}_{k,i}\}_{i\in[\mathsf{ctr}_k]})$

$\underline{\textsc{Sign}(\mathcal{S}, k, \mu)}$

if $k \in \mathcal{C}$ **then**
  return $\perp$
end if
$\mathcal{Q} := \mathcal{Q} \cup \{\mu\}$
$\mathsf{ctr}_k := \mathsf{ctr}_k + 1$
$\mathsf{sid} := \mathsf{ctr}_k$
$(u, \hat{u}) := \mathsf{H}_1(\mu)$
$Y_k := u^{\delta_k}$
$\gamma \leftarrow \mathbb{Z}_p$
$\left[\begin{smallmatrix} R_k \\ S_k \\ T_k \end{smallmatrix}\right] := \left[\begin{smallmatrix} g \\ h \\ u \end{smallmatrix}\right]^{\gamma}$
$\mathsf{st}'_{k,\mathsf{sid}} := (\mathcal{S}, \mu, \gamma)$
$\mathsf{round}_{k,\mathsf{sid}} := 1$
return (Y_k, R_k, S_k, T_k)

$\underline{\textsc{Sign}'(k, \mathsf{sid}, \mathcal{M})}$

if $k \in \mathcal{C} \vee \mathsf{round}_{k,\mathsf{sid}} \neq 1$ **then**
  return $\perp$
end if
$(\mathcal{S}, \mu, \gamma) := \mathsf{st}'_{k,\mathsf{sid}}$
$\{\mathsf{msg}_i\}_{i\in\mathcal{S}\setminus\{k\}} := \mathcal{M}$
for $i \in \mathcal{S} \setminus \{k\}$ **do**
  $(Y_i, R_i, S_i, T_i) := \mathsf{msg}_i$
end for
$\tilde{Y} := \prod_{i\in\mathcal{S}} Y_i$, $\tilde{R} := \prod_{i\in\mathcal{S}} R_i$
$\tilde{S} := \prod_{i\in\mathcal{S}} S_i$, $\tilde{T} := \prod_{i\in\mathcal{S}} T_i$
$c := \mathsf{H}_2(\mu, \tilde{Y}, \tilde{R}, \tilde{S}, \tilde{T})$
$z_k \leftarrow \mathbb{Z}_p$
$y_k := \gamma + c \cdot \lambda_{\mathcal{S},k} \cdot \delta_k - \alpha z_k$
$\mathsf{round}_{k,\mathsf{sid}} := 2$
$\mathsf{rsp}_{k,\mathsf{sid}} := (y_k, z_k)$
return (y_k, z_k)

Figure 5: Hybrid games G_4–G_5. H_1 and H_2 are the same as in G_3.

If the session is incomplete, i.e., only the first round is finished, then Corr samples $s_k \leftarrow \mathbb{Z}_p$ and computes $r_k = \gamma - \alpha s_k$.

We show that G_4 is identical to G_3 until GoodTosses is violated. First, we show that if $\mathsf{coin}(\mu) = 1$, then the output of the signing oracles in G_3 and G_4 are identically distributed, where

the randomnesses are (r_k, s_k) and (α, z_k) in each signing session, respectively.

In G_3, Sign returns $Y_k = u^{w_k}\hat{u}_k^x = g^{\beta w_k}\hat{g}^{\beta x_k} = g^{\delta_k \beta}$, the same as G_4. It also returns

$$\begin{bmatrix} R_k \\ S_k \\ T_k \end{bmatrix} = \begin{bmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{bmatrix} \begin{bmatrix} r_k \\ s_k \end{bmatrix} = \begin{bmatrix} g \\ h \\ u \end{bmatrix}^{r_k + \alpha s_k}.$$

Since r_k is uniformly distributed over $\mathbb{Z}_p$, (R_k, S_k, T_k) is identically distributed as $(g^\gamma, h^\gamma, u^\gamma)$ in G_4. Moreover, s_k is independent of the output of Sign. Therefore, Sign$'$ returns $z_k = s_k + c\lambda_{\mathcal{S},k} x_k$ which is uniformly distributed over $\mathbb{Z}_p$ and independent of the output of Sign. Then

$$\begin{aligned} y_k &= r_k + c\lambda_{\mathcal{S},k} w_k = \gamma - \alpha s_k + c\lambda_{\mathcal{S},k}(\delta_k - \alpha x_k) \\ &= \gamma + c\lambda_{\mathcal{S},k}\delta_k - \alpha(s_k + c\lambda_{\mathcal{S},k} x_k) = \gamma + c\lambda_{\mathcal{S},k}\delta_k - \alpha z_k, \end{aligned}$$

so (y_k, z_k) is also identically distributed as in G_4.

Then we show the outputs of the corruption oracle in G_3 and G_4 are also identical. For completed sessions this is clear. For each incomplete session, this is true because s_k is independent of the output of Sign.

Since G_3 and G_4 only differ after GoodTosses is violated, we have

$$\Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodTosses} \mid \mathsf{G}_4] = \Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodTosses} \mid \mathsf{G}_3]. \quad (5)$$

G_5. In G_5, we change the initialization of the game. We already let Sign use δ_k instead of the secret share (w_k, x_k). Now the secret key shares are only used in Corr. In this game, we only generate δ and its shares $\{\delta_i\}_{i\in[n]}$ at the beginning. The generation of secret key shares is deferred to the corresponding corruption queries, and the generation of the secret key will be deferred to the end of the game.

At the beginning, the game samples $\delta \leftarrow \mathbb{Z}_p$ and shares it as $\{\delta_i\}_{i\in[n]} \leftarrow \mathsf{Share}(n, t, \delta)$. It sets $(W, X) = (g^\delta, h^\delta)$. To answer a valid corruption query Corr(k), the game samples $x_k \leftarrow \mathbb{Z}_p$ and calculates $w_k = \delta_k - \alpha x_k$. At the end of the game, it samples $x \leftarrow \mathbb{Z}_p$ and computes $w = \delta - \alpha x$.

We show that G_5 is identical to G_4. In G_4, $\delta = w + \alpha x$, which is uniformly random so identically distributed as in G_5. Then from the linearity of Shamir's secret sharing, $\delta_i = w_i + \alpha x_i$ for $i \in [n]$ are also identically distributed to that in G_5. Moreover, δ and δ_i are independent of x and x_i, respectively. The property of Shamir's secret sharing guarantees that x and the at most $t-1$ shares of x_i output by Corr are mutually independent. Therefore, those corrupted x_i and x are also identical to that in G_5. In conclusion, we have

$$\Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodTosses} \mid \mathsf{G}_5] = \Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodTosses} \mid \mathsf{G}_4]. \quad (6)$$

Finally, we show that $\Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodTosses} \mid \mathsf{G}_5] \le (q_\mathrm{h} + 2)/p$. It suffices to show $\Pr[\textsc{CorrectY} \mid \mathsf{G}_5, \textsc{GoodTosses}] \le (q_\mathrm{h} + 2)/p$. For each hash query $\mathsf{H}_1(\mu)$ with $\mathsf{coin}(\mu) = 0$, $(u, \hat{u})$ is uniformly sampled from $\mathbb{G}^2$, which is not parallel to $(g, \hat{g})$ except with probability $1/p$. Hence, except with probability at most $(q_\mathrm{h} + 1)/p$, every query on μ with $\mathsf{coin}(\mu) = 0$ that is made by A or in the final verification return $(u, \hat{u})$ not parallel to $(g, \hat{g})$. If GoodTosses occurs, then μ^* corresponds to one of such queries, and $(u^*, \hat{u}^*) = \mathsf{H}_1(\mu^*)$ is not parallel to $(g, \hat{g})$. Suppose $\hat{u}^* = (u^*)^\alpha v$ with $v \neq \mathbb{1}$. At the end of the game, $x \leftarrow \mathbb{Z}_p$ and $w = \delta - \alpha x$ are generated. Then $(u^*)^w(\hat{u}^*)^x = (u^*)^w((\hat{u}^*)^\alpha v)^x = (u^*)^\delta v^x$. Each $x \in \mathbb{Z}_p$ yields a distinct value of $(u^*)^w(\hat{u}^*)^x$, so the probability that $Y^* = (u^*)^w(\hat{u}^*)^x$ is $1/p$. It follows by the union bound that $\Pr[\textsc{CorrectY} \mid \mathsf{G}_5, \textsc{GoodTosses}] \le (q_\mathrm{h} + 2)/p$.

□

4 Three-Round Scheme from DDH with Fully Tight Security

We describe our scheme Dazzle-T below. We give a formal pseudocode description in Fig. 6. The scheme is defined with respect to a group generation algorithm GrGen.

Setup, Key Generation, and Verification. Same as Dazzle.

Signing Protocol. Suppose a group $\mathcal{S}$ of signers want to jointly sign a message μ. In the first stage, each participant (whose index is denoted by k) samples $r_k,\ s_k \leftarrow \mathbb{Z}_p$ and sends its partial commitment $\left[\begin{smallmatrix} R_k \\ S_k \end{smallmatrix}\right] := \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} r_k \\ s_k \end{smallmatrix}\right]}$ as its protocol message.

In the second stage, the signer hashes the signer group $\mathcal{S}$, the message μ, and all protocol messages $\{(R_i, S_i)\}_{i\in\mathcal{S}}$ it receives into a public mask $b := \mathsf{H}_1(\mathcal{S}, \mu, \{(R_i, S_i)\}_{i\in\mathcal{S}})$. It aggregates all protocol messages it receives with the public mask to get the partial aggregated commitment $\left[\begin{smallmatrix} \tilde{R} \\ \tilde{S} \end{smallmatrix}\right] = \left[\begin{smallmatrix} g \\ h \end{smallmatrix}\right]^b \cdot \prod_{i\in\mathcal{S}} \left[\begin{smallmatrix} R_i \\ S_i \end{smallmatrix}\right]$. Then it hashes the message together with $(\tilde{R}, \tilde{S})$ to obtain $(u, \hat{u}) := \mathsf{H}_2(\mu, \tilde{R}, \tilde{S})$. It computes its ephemeral public key $Y_k := u^{w_k}\hat{u}^{x_k}$ and the remaining commitment $T_k := u^{r_k}\hat{u}^{s_k}$ as its protocol message.

In the third stage, the signer aggregates all the protocol messages into the aggregated ephemeral public key $\tilde{Y} := \prod_{i\in\mathcal{S}} Y_i$ and the aggregated remaining commitment, also masked by b, $\tilde{T} := u^b \cdot \prod_{i\in\mathcal{S}} T_i$. It hashes the aggregated ephemeral public key, the commitment, and the message to get the challenge $c := \mathsf{H}_3(\mu, \tilde{Y}, \tilde{R}, \tilde{S}, \tilde{T})$. It computes its response $\left[\begin{smallmatrix} y_k \\ z_k \end{smallmatrix}\right] := \left[\begin{smallmatrix} r_k \\ s_k \end{smallmatrix}\right] + c \cdot \lambda_{\mathcal{S},k} \cdot \left[\begin{smallmatrix} w_k \\ x_k \end{smallmatrix}\right]$, where $\lambda_{\mathcal{S},k}$ is the corresponding Lagrange coefficient for $\mathcal{S}$. It sends the response as its protocol message.

In the last stage, a designated combiner recompute b, $\tilde{R}$, $\tilde{S}$ based on the first-round protocol messages. Then it aggregates the second-round protocol messages to recompute $\tilde{Y}$, $\tilde{T}$, and c. Finally, it aggregates the second-round protocol messages with the public mask into the aggregated response $\left[\begin{smallmatrix} \tilde{y} \\ \tilde{z} \end{smallmatrix}\right] := \left[\begin{smallmatrix} b \\ 0 \end{smallmatrix}\right] + \sum_{i\in\mathcal{S}} \left[\begin{smallmatrix} y_k \\ z_k \end{smallmatrix}\right]$. It outputs the threshold signature $\sigma := (\tilde{Y}, c, \tilde{y}, \tilde{z})$.

Theorem 2. *If* DDH *is* $(\tau_{\mathrm{ddh}}, \varepsilon_{\mathrm{ddh}})$*-hard for* GrGen*, then* Dazzle-T *is* $(q_{\mathrm{s}}, \tau_{\mathrm{uf}}, \varepsilon_{\mathrm{uf}})$*-adaptively secure in the random oracle model against any adversary that makes at most* q_{h} *hash queries, where essentially* $\tau_{\mathrm{ddh}} \approx \tau_{\mathrm{uf}}$ *and*

$$\varepsilon_{\mathrm{ddh}} \geq \varepsilon_{\mathrm{uf}} - \frac{(q_{\mathrm{h}} + 2q_{\mathrm{s}})(q_{\mathrm{h}} + q_{\mathrm{s}}) + 2q_{\mathrm{h}} + q_{\mathrm{s}} + 5}{p}.$$

Proof. Assume that there exists an adversary A that breaks the adp-UF security of Dazzle-T. We will define a series of hybrid games.

G_0. Let G_0 be the original security game adp-UF against Dazzle-T. Suppose that A outputs message μ^* and forged signature $\sigma^* = (\tilde{Y}^*, c^*, \tilde{y}^*, \tilde{z}^*)$, and commitments $\tilde{R}^*$, $\tilde{S}^*$ are recovered during the final verification. Let event Win indicate the win of A. By the assumption, we have $\Pr[\textsc{Win} \mid \mathsf{G}_0] = \varepsilon_{\mathrm{uf}}$. Let event CorrectY indicate that $\tilde{Y}^* = (u^*)^w(\hat{u}^*)^x$, where $(u^*, \hat{u}^*) = \mathsf{H}_2(\tilde{R}^*, \tilde{S}^*, \mu^*)$. For the same reason as in the proof of Theorem 1, we have

$$\varepsilon_{\mathrm{uf}} \leq \Pr[\textsc{Win} \wedge \textsc{CorrectY} \mid \mathsf{G}_0] + \frac{q_{\mathrm{h}} + q_{\mathrm{s}} + 2}{p}. \tag{7}$$

G_1. In G_1, we set a coin when answering H_2 query. We make the following definitions:

- For any H_1 query $\mathsf{H}_1(\mu, \mathcal{S}, \{(R_i, S_i)\}_{i\in\mathcal{S}})$, let b be the returned value, $\tilde{R} = g^b \prod_{i\in\mathcal{S}} R_i$, $\tilde{S} = h^b \prod_{i\in\mathcal{S}} S_i$. We call this H_1 query a *preceding hash query* to the query $\mathsf{H}_2(\mu, \tilde{R}, \tilde{S})$.
- For any Sign query $\textsc{Sign}(k, \mathcal{S}, \mu)$, let (R_k, S_k) be the returned value. We call this Sign query a *preceding signing query* to any query in form of $\mathsf{H}_1(\mu, \mathcal{S}, \{(R_i, S_i)\}_{i\in\mathcal{S}})$ if the input list $\{(R_i, S_i)\}_{i\in\mathcal{S}}$ contains (R_k, S_k) at index k.

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

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

$\hat{g},\ h,\ \hat{h} \leftarrow \mathbb{G}$

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

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

$w,\ x \leftarrow \mathbb{Z}_p$

$\left[\begin{smallmatrix} W \\ X \end{smallmatrix}\right] := \begin{bmatrix} g & \hat{g} \\ h & \hat{h} \end{bmatrix}^{\left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]}$

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

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

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

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

  $\mathsf{sk}_i := (w_i, x_i)$

end for

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

$\underline{\mathsf{Sign}(\mathcal{S}, k, \mathsf{sk}_k, \mu)}$

$(w_k, x_k) := \mathsf{sk}_k$

$r_k,\ s_k \leftarrow \mathbb{Z}_p$

$\left[\begin{smallmatrix} R_k \\ S_k \end{smallmatrix}\right] := \begin{bmatrix} g & \hat{g} \\ h & \hat{h} \end{bmatrix}^{\left[\begin{smallmatrix} r_k \\ s_k \end{smallmatrix}\right]}$

$\mathsf{st} := (r_k, s_k)$

$\mathsf{msg} := (R_k, S_k)$

return $(\mathsf{st}, \mathsf{msg})$

$\underline{\mathsf{Sign}'(\mathcal{S}, k, \mathsf{sk}_k, \mu, \mathsf{st}, \mathcal{M})}$

$(w_k, x_k) := \mathsf{sk}_k$

$(r_k, s_k) := \mathsf{st}$

$\{\mathsf{msg}_i\}_{i\in\mathcal{S}} := \mathcal{M}$

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

  $(R_i, S_i) := \mathsf{msg}_i$

end for

$b := \mathsf{H}_1(\mathcal{S}, \mu, \{R_i, S_i\}_{i\in\mathcal{S}})$

$\left[\begin{smallmatrix} \tilde{R} \\ \tilde{S} \end{smallmatrix}\right] := \left[\begin{smallmatrix} g \\ h \end{smallmatrix}\right]^b \cdot \prod_{i\in\mathcal{S}} \left[\begin{smallmatrix} R_i \\ S_i \end{smallmatrix}\right]$

$(u, \hat{u}) := \mathsf{H}_2(\mu, \tilde{R}, \tilde{S})$

$Y_k := u^{w_k} \hat{u}^{x_k}$

$T_k := u^{r_k} \hat{u}^{s_k}$

$\mathsf{st}' := (r_k, s_k, \tilde{R}, \tilde{S}, b)$

$\mathsf{msg}' := (Y_k, T_k)$

return $(\mathsf{st}', \mathsf{msg}')$

$\underline{\mathsf{Sign}''(\mathcal{S}, k, \mathsf{sk}_k, \mu, \mathsf{st}, \mathcal{M}')}$

$(w_k, x_k) := \mathsf{sk}_k$

$(r_k, s_k, \tilde{R}, \tilde{S}, b) := \mathsf{st}'$

$\{\mathsf{msg}'_i\}_{i\in\mathcal{S}} := \mathcal{M}'$

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

  $(Y_i, T_i) := \mathsf{msg}'_i$

end for

$\tilde{Y} := \prod_{i\in\mathcal{S}} Y_i$

$\tilde{T} := u^b \cdot \prod_{i\in\mathcal{S}} T_i$

$c := \mathsf{H}_3(\mu, \tilde{Y}, \tilde{R}, \tilde{S}, \tilde{T})$

$\left[\begin{smallmatrix} y_k \\ z_k \end{smallmatrix}\right] := b \cdot \left[\begin{smallmatrix} r_k \\ s_k \end{smallmatrix}\right] + c \cdot \lambda_{\mathcal{S},k} \cdot \left[\begin{smallmatrix} w_k \\ x_k \end{smallmatrix}\right]$

return (y_k, z_k)

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

$\{\mathsf{msg}_i\}_{i\in\mathcal{S}} := \mathcal{M}$

$\{\mathsf{msg}'_i\}_{i\in\mathcal{S}} := \mathcal{M}'$

$\{\mathsf{msg}''_i\}_{i\in\mathcal{S}} := \mathcal{M}''$

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

  $(R_i, S_i) := \mathsf{msg}_i$

  $(Y_i, T_i) := \mathsf{msg}'_i$

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

end for

$b := \mathsf{H}_1(\mathcal{S}, \mu, \{R_i, S_i\}_{i\in\mathcal{S}})$

$\tilde{Y} := \prod_{i\in\mathcal{S}} Y_i$

$\left[\begin{smallmatrix} \tilde{R} \\ \tilde{S} \end{smallmatrix}\right] := \left[\begin{smallmatrix} g \\ h \end{smallmatrix}\right]^b \cdot \prod_{i\in\mathcal{S}} \left[\begin{smallmatrix} R_i \\ S_i \end{smallmatrix}\right]$

$(u, \hat{u}) := \mathsf{H}_2(\mu, \tilde{R}, \tilde{S})$

$\tilde{T} := u^b \cdot \prod_{i\in\mathcal{S}} T_i$

$c := \mathsf{H}_3(\mu, \tilde{Y}, \tilde{R}, \tilde{S}, \tilde{T})$

$\left[\begin{smallmatrix} \tilde{y} \\ \tilde{z} \end{smallmatrix}\right] := \left[\begin{smallmatrix} b \\ 0 \end{smallmatrix}\right] + \sum_{i\in\mathcal{S}} \left[\begin{smallmatrix} y_i \\ z_i \end{smallmatrix}\right]$

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

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

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

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

$(u, \hat{u}) := \mathsf{H}_1(\mu)$

$\left[\begin{smallmatrix} \tilde{R} \\ \tilde{S} \\ \tilde{T} \end{smallmatrix}\right] := \begin{bmatrix} g & \hat{g} \\ h & \hat{h} \\ u & \hat{u} \end{bmatrix}^{\left[\begin{smallmatrix} \tilde{y} \\ \tilde{z} \end{smallmatrix}\right]} \cdot \left[\begin{smallmatrix} W \\ X \\ \tilde{Y} \end{smallmatrix}\right]^{-c}$

return $[\![c = \mathsf{H}_3(\mu, \tilde{Y}, \tilde{R}, \tilde{S}, \tilde{T})]\!]$

Figure 6: Scheme description of Dazzle-T.

- If a signing query precedes a H_1 query that precedes a H_2 query, then we also say the signing query is a *preceding signing query* to the H_2 query.

Now we specify how to set the coin. On each $\mathsf{H}_2(\mu, \tilde{R}, \tilde{S})$ query, if a preceding signing query has been made, then we set $\mathsf{coin}(\mu, \tilde{R}, \tilde{S}) = 1$. Otherwise, we set $\mathsf{coin}(\mu, \tilde{R}, \tilde{S}) = 0$.

Define GoodCoins as the event that:

- for each signing session that has completed the second round, $\mathsf{coin}(\mu, \tilde{R}, \tilde{S}) = 1$, where μ is the message and $(\tilde{R}, \tilde{S})$ are the aggregated commitments computed in the second round; and
- $\mathsf{coin}(\mu^*, \tilde{R}^*, \tilde{S}^*) = 0$.

By the following claim we have

$$\begin{aligned} &\Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodCoins} \mid \mathsf{G}_1] \\ &\geq \Pr[\textsc{Win} \wedge \textsc{CorrectY} \mid \mathsf{G}_1] - \frac{(q_\mathrm{h} + 2q_\mathrm{s})(q_\mathrm{h} + q_\mathrm{s})}{p}. \end{aligned} \tag{8}$$

Claim. $\Pr[\neg\textsc{GoodCoins} \mid \mathsf{G}_1, \textsc{Win}] < (q_\mathrm{h} + 2q_\mathrm{s})(q_\mathrm{h} + q_\mathrm{s})/p$.

Proof of Claim. For A to win, it must not have made any signing query on μ^*. Thus, query $\mathsf{H}_2(\mu^*, \tilde{R}^*, \tilde{S}^*)$ does not have a preceding signing query, so the coin would be set as 0.

It remains to consider the coin corresponding to each signing query. From the scheme description, we can see that the query $\mathsf{H}_2(\mu, \tilde{R}, \tilde{S})$ corresponding to each signing query must have a preceding H_1 query and a preceding signing query being made throughout the game. For the coin to be set to 0, one of the following must hold when answering $\mathsf{H}_2(\mu, \tilde{R}, \tilde{S})$:

- No preceding H_1 query has been made.
- A preceding H_1 query has been made, but no preceding signing query has been made.

The first implies that there exists a query $\mathsf{H}_1(\mu, \mathcal{S}, \{R_i, S_i\}_{i\in\mathcal{S}}) = b$ such that $\tilde{R} = g^b \prod_{i\in\mathcal{S}} R_i$ and $\tilde{S} = h^b \prod_{i\in\mathcal{S}} S_i$ hit an earlier query $\mathsf{H}_2(\mu, \tilde{R}, \tilde{S})$. Since b is independently, uniformly chosen, each H_1 query hits an earlier H_2 query with probability at most $(q_\mathrm{h} + q_\mathrm{s})/p$. In total, such an event occurs throughout the game with probability at most $(q_\mathrm{h} + q_\mathrm{s})^2/p$.

The second implies that there exists a query $\textsc{Sign}(\mathcal{S}, k, \mu)$ with output (R_k, S_k) that hits an earlier query $\mathsf{H}_1(\mu, \mathcal{S}, \{R_i, S_i\}_{i\in\mathcal{S}})$. Each Sign query hits an earlier H_1 query with probability at most $(q_\mathrm{h} + q_\mathrm{s})/p$. In total this occurs with probability at most $q_\mathrm{s}(q_\mathrm{h} + q_\mathrm{s})/p$. □

G_2. In G_2, we answer $\mathsf{H}_2(\mu, \tilde{R}, \tilde{S})$ query based on $\mathsf{coin}(\mu, \tilde{R}, \tilde{S})$. If $\mathsf{coin}(\mu, \tilde{R}, \tilde{S}) = 0$, we still return $(u, \hat{u}) \leftarrow \mathbb{G}^2$. If $\mathsf{coin}(\mu, \tilde{R}, \tilde{S}) = 1$, we let $(u, \hat{u})$ be a uniform linear combination of $(g, \hat{g})$ and $(h, \hat{h})$, i.e. $(u, \hat{u}) = (g^d h^e, \hat{g}^d \hat{h}^e)$ with $d, e \leftarrow \mathbb{Z}_p$. Unless $(g, \hat{g}, h, \hat{h})$ is a DH tuple, which occurs with probability $1/p$, this modification does not change the view of A. Thus, we have

$$\Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodCoins} \mid \mathsf{G}_2] \geq \Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodCoins} \mid \mathsf{G}_1] - \frac{1}{p}. \tag{9}$$

G_3. In this game, we let $(g, \hat{g}, h, \hat{h})$ be a uniform DH tuple instead of independent group elements. A distinguisher against DDH can simulate $\mathsf{G}_2/\mathsf{G}_3$ and verify event $\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodCoins}$. Thus

$$\Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodCoins} \mid \mathsf{G}_3] \geq \Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodCoins} \mid \mathsf{G}_2] - \varepsilon_\mathrm{ddh}. \tag{10}$$

Combining Eqs. (7) to (10) and the following claim, we can conclude this proof.

Claim. $\Pr[\textsc{Win} \wedge \textsc{CorrectY} \wedge \textsc{GoodCoins} \mid \mathsf{G}_3] \leq (q_\mathrm{h} + 2)/p$.

□

The proof of this claim is almost identical to the final claim in the proof of Theorem 1, so we omit it.

5 Generic Construction And AOMCDH-Based Scheme

Bacho et al. [BLT+24] presented Twinkle as a generic construction for linear functions and provided two instantiations based on AOMCDH and DDH, respectively. In this section, we briefly discuss how our improvements apply to the generic construction and the AOMCDH-based scheme.

Bacho et al.'s AOMCDH-Based Scheme. The one-party scheme underlying their AOMCDH-based instantation is the Chaum-Pedersen signature scheme [CP93]. It works over a cyclic group $\mathbb{G}$ of prime order p with generator g. The public key is $X = g^x$, where x is the secret key. To sign a message μ, the signer hashes μ into element $h := \mathsf{H}_1(\mu)$. It computes the ephemeral public key $Y = h^x$ and produces a Schnorr-like proof of knowledge of x satisfying $\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right] := \left[\begin{smallmatrix} g \\ h \end{smallmatrix}\right]^x$. The signature is output as $\sigma := (Y, c, z)$, where (c, z) is the proof. The Chaum-Pedersen scheme can be proved secure based on CDH with a rewinding-free reduction.

By applying the simple one-party to threshold transformation to the Chaum-Pedersen scheme, we can obtain the following two-round threshold signature scheme. The secret key x is distributed using t-out-of-n Shamir's secret sharing. Suppose that a set of signers $\mathcal{S}$ want to jointly sign a message μ. In the first round of the signing protocol, each signer (indexed by k) calculates $h := \mathsf{H}_1(\mu)$ and computes its individual ephemeral public key $Y_k := h^{x_k}$ and commitment $\left[\begin{smallmatrix} R_k \\ S_k \end{smallmatrix}\right] := \left[\begin{smallmatrix} g \\ h \end{smallmatrix}\right]^{r_k}$ with $r_k \leftarrow \mathbb{Z}_p$ and broadcasts them. Then the aggregated ephemeral key and commitment are calculated as $\tilde{Y} := \prod_{i\in\mathcal{S}} Y_i$ and $\left[\begin{smallmatrix} \tilde{R} \\ \tilde{S} \end{smallmatrix}\right] := \prod_{i\in\mathcal{S}} \left[\begin{smallmatrix} R_i \\ S_i \end{smallmatrix}\right]$. In the second round, the signer calculates the challenge as $c := \mathsf{H}_2(\mu, \tilde{Y}, \tilde{R}, \tilde{S})$ and computes its individual response $z_k := r_k + c \cdot \lambda_{\mathcal{S},k} \cdot x_k$. Finally, a combiner computes the aggregated response $\tilde{z} := \sum_{i\in\mathcal{S}} z_i$ and outputs $(\tilde{Y}, c, \tilde{z})$ as the signature.

The CDH-based reduction for the Chaum-Pedersen scheme embeds an instance of CDH in the public key X. As a result, the reduction does not know the secret key x. To prove the adaptive security of the threshold scheme, Bacho et al. introduced the algebraic one-more computational Diffie-Hellman (AOMCDH) assumption. Given uniformly random generator g and elements h, g^{α_1}, $\ldots$, g^{α_m}, the m-AOMCDH problem is to output h^{α_1}, $\ldots$, h^{α_m} with $(m-1)$-time access to a discrete logarithm oracle. Every query to the discrete-log oracle is required to be represented as a linear combination of the input elements. The reduction for the threshold scheme embeds g^{α_1}, $\ldots$, g^{α_t} in the public key shares $\{X_i\}_{i\in[n]} = \{g^{x_i}\}_{i\in[n]}$. The reduction handles each corruption by querying X_k to the discrete-log oracle. The $t-1$ queries to the discrete-log oracle and the forgery of the adversary provide t equations with unknown h^{α_1}, $\ldots$, h^{α_t}, which allows the reduction to solve t-AOMCDH.

Improved AOMCDH-Based Scheme. The above reduction does not know the secret key shares from the beginning. It may be asked to sign under some public key share with unknown discrete logarithm. Bacho et al. turned to a three-round scheme with an additional hashing round, where the reduction applies the HVZK simulator to produce signatures. However, like Dazzle, the two-round scheme is already secure.

Our key observation is that, instead of applying the HVZK simulator, the signing queries can also be handled with the discrete-log oracle. We embed AOMCDH inputs not only in public key shares but also in the commitment R_k for each signing query. For every μ it needs to sign, the reduction knows the discrete logarithm w of $\mathsf{H}_1(\mu) = g^w$, so it can compute the ephemeral public key $Y_k = X_k^w$ and the remaining commitment $S_k = R_k^w$ in the first round. To respond to challenge c in the second round, it queries $R_k X_k^{c\cdot\lambda_{\mathcal{S},k}}$ to the oracle to obtain z_k. When user k is corrupted, the reduction should also output r_k for each signing query. If the signing session is completed, then it can compute r_k as $z_k - c \cdot \lambda_{\mathcal{S},k} x_k$. Else, it simply queries R_k to the discrete-log oracle. Each signing session is handled by one query to the discrete-log oracle, either when it is completed or when the signer is corrupted. At the end, a total of $q_\mathrm{s} + t - 1$ queries to the discrete-log oracle and the forgery of the adversary provide $q_\mathrm{s} + t$ equations, which allow the reduction to solve $(q_\mathrm{s} + t)$-AOMCDH.

Answering signing queries in this way does not require fixing the challenge c in advance, so it works perfectly in the two-round scheme.

To construct a tight variant, we can just replace the Chaum-Pedersen scheme with the Chevallier-Mames scheme [Che05, KLP17], i.e., let $h := \mathsf{H}_1(\mu, R)$ instead of $h := \mathsf{H}_1(\mu)$. Then it can be turned to a three-round threshold scheme, with our technique to preserve the tightness.

Improved Generic Construction. Starting from the AOMCDH-based scheme, we can achieve Bacho et al.'s generic construction by replacing group exponentiation g^x with a generic tagged linear function $\mathsf{T}(g, x)$. The above improvements in round complexity and tightness directly apply to the generic construction. It only remains to discuss how to modify the generalization, such that the DDH-based instantiation has shorter signatures like ours.

Their DDH-based instantiation is obtained by choosing tagged linear function $\mathsf{T}\left(\left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right], \left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]\right) = \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]}$. They showed that for this function, the generic AOMCDH assumption tightly reduces to DDH. Their ephemeral public key contains two group elements like the real public key.

We observe that the construction can be further generalized by using different linear functions for the real public key and the ephemeral public key. In Dazzle, the real public key is computed by $\mathsf{T}\left(\left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right], \left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]\right) = \left[\begin{smallmatrix} g & \hat{g} \\ h & \hat{h} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]}$, while the ephemeral public key is computed by another function $\mathsf{T}'\left(\left[\begin{smallmatrix} u & \hat{u} \end{smallmatrix}\right], \left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]\right) = \left[\begin{smallmatrix} u & \hat{u} \end{smallmatrix}\right]^{\left[\begin{smallmatrix} w \\ x \end{smallmatrix}\right]}$, which yields shorter ephemeral keys. Accordingly, the generic AOMCDH assumption should be defined asymmetrically: given a tag g for T, a tag h for T', and $\mathsf{T}(g, \alpha_1)$, …, $\mathsf{T}(g, \alpha_m)$, output $\mathsf{T}'(h, \alpha_1)$, …, $\mathsf{T}'(h, \alpha_m)$. For our choice of T and T', this asymmetric AOMCDH assumption also tightly reduces to DDH.

Acknowledgements

We would like to thank Andrej Bogdanov, Biming Zhou, and the anonymous PKC reviewers for their helpful discussions and comments.

This work was supported by an NSERC discovery grant.

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. 6

[AF04] Masayuki Abe and Serge Fehr. Adaptively secure feldman VSS and applications to universally-composable threshold cryptography. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 317–334. Springer, Berlin, Heidelberg, August 2004. 7

[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. 6

[BCK+22] Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 517–550. Springer, Cham, August 2022. 1, 7

[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. 6

[BHK+24] Fabrice Benhamouda, Shai Halevi, Hugo Krawczyk, Yiping Ma, and Tal Rabin. SPRINT: High-throughput robust distributed Schnorr signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part V*, volume 14655 of *LNCS*, pages 62–91. Springer, Cham, May 2024. 7

[BLL+21] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 33–53. Springer, Cham, October 2021. 5

[BLSW24] Renas Bacho, Julian Loss, Gilad Stern, and Benedikt Wagner. HARTS: High-threshold, adaptively secure, and robust threshold schnorr signatures. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 104–140. Springer, Singapore, December 2024. 1, 7

[BLT+24] Renas Bacho, Julian Loss, Stefano Tessaro, Benedikt Wagner, and Chenzhi Zhu. Twinkle: Threshold signatures from DDH with full adaptive security. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 429–459. Springer, Cham, May 2024. 1, 2, 3, 7, 20

[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. 5, 6, 7

[BP23] Luis Brandao and Rene Peralta. Nist first call for multi-party threshold schemes. *doi*, 10:6028, 2023. 1

[BW24] Renas Bacho and Benedikt Wagner. Tightly secure threshold signatures over pairing-free groups. Cryptology ePrint Archive, Paper 2024/1557, 2024. 2

[CGJ+99] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 98–115. Springer, Berlin, Heidelberg, August 1999. 7

[CGRS23] Hien Chu, Paul Gerhart, Tim Ruffing, and Dominique Schröder. Practical Schnorr threshold signatures without the algebraic group model. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 743–773. Springer, Cham, August 2023. 1, 7

[Che05] Benoît Chevallier-Mames. An efficient CDH-based signature scheme with a tight security reduction. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 511–526. Springer, Berlin, Heidelberg, August 2005. 6, 21

[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. 1, 2, 3, 7

[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. 6

[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. 20

[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. 5, 6

[Des90] Yvo Desmedt. Abuses in cryptography and how to fight them. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 375–389. Springer, New York, August 1990. 1

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, New York, August 1990. 1

[DGJL21] Denis Diemert, Kai Gellert, Tibor Jager, and Lin Lyu. More efficient digital signatures with tight multi-user security. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 1–31. Springer, Cham, May 2021. 4

[DYX+22] Sourav Das, Thomas Yurek, Zhuolun Xiang, Andrew K. Miller, Lefteris Kokoris-Kogias, and Ling Ren. Practical asynchronous distributed key generation. In *2022 IEEE Symposium on Security and Privacy*, pages 2518–2534. IEEE Computer Society Press, May 2022. 7

[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. 7

[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. 2

[GJ18] Kristian Gjøsteen and Tibor Jager. Practical and tightly-secure digital signatures and authenticated key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 95–125. Springer, Cham, August 2018. 4

[GJKR07] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, January 2007. 7

[GS24] Jens Groth and Victor Shoup. Fast batched asynchronous distributed key generation. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part V*, volume 14655 of *LNCS*, pages 370–400. Springer, Cham, May 2024. 7

[JL00] Stanislaw Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 221–242. Springer, Berlin, Heidelberg, May 2000. 7

[KG20] Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson, Jr., and Colin O'Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 34–65. Springer, Cham, October 2020. 1, 2, 3, 7

[KGS23] Chelsea Komlo, Ian Goldberg, and Douglas Stebila. A formal treatment of distributed key generation, and new constructions. Cryptology ePrint Archive, Report 2023/292, 2023. 7

[KLP17] Eike Kiltz, Julian Loss, and Jiaxin Pan. Tightly-secure signatures from five-move identification protocols. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 68–94. Springer, Cham, December 2017. 6, 21

[KMS20] Eleftherios Kokoris-Kogias, Dahlia Malkhi, and Alexander Spiegelman. Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1751–1767. ACM Press, November 2020. 7

[KRT24] Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 459–491. Springer, Cham, August 2024. 1, 2, 3, 7

[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. 7

[Lin24] Yehuda Lindell. Simple three-round multiparty schnorr signing with full simulatability. *CiC*, 1(1):25, 2024. 1, 2, 7

[LPQ18] Benoît Libert, Thomas Peters, and Chen Qian. Logarithmic-size ring signatures with tight security from the DDH assumption. In Javier López, Jianying Zhou, and Miguel Soriano, editors, *ESORICS 2018, Part II*, volume 11099 of *LNCS*, pages 288–308. Springer, Cham, September 2018. 4

[MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. *DCC*, 87(9):2139–2164, 2019. 6

[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. 6

[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. 6

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Berlin, Heidelberg, August 1992. 7

[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. 6, 7

[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. 6, 7

[RRJ+22] Tim Ruffing, Viktoria Ronge, Elliott Jin, Jonas Schneider-Bensch, and Dominique Schröder. ROAST: Robust asynchronous schnorr threshold signatures. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 2551–2564. ACM Press, November 2022. 7

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. 8

[Sho23] Victor Shoup. The many faces of schnorr. Cryptology ePrint Archive, Report 2023/1019, 2023. 7

[SS01] Douglas R. Stinson and Reto Strobl. Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. In Vijay Varadharajan and Yi Mu, editors, *ACISP 01*, volume 2119 of *LNCS*, pages 417–434. Springer, Berlin, Heidelberg, July 2001. 7

[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. 1, 2, 3, 6