

Two-Round Threshold Signatures with Adaptive TS-UF-1 Security from MDDH

Wenzhong Li^{1,2} , Shengli Liu^{1,2}  , and You Lyu^{1,2} 

¹ School of Computer Science

Shanghai Jiao Tong University, Shanghai 200240, China

{wzli0920, slliu, vergil}@sjtu.edu.cn

² State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

Abstract. In this paper, we construct threshold signatures (TS) to achieve adaptive TS-UF-1 security and identifiable abort. Our TS construction has two rounds. We prove its adaptive TS-UF-1 security in the random oracle model (ROM) under the MDDH (covering DDH) assumption in pairing-free groups.

To the best of our knowledge, there are only two TS schemes, namely GLRS26 (Gerhart et al., Eurocrypt 2026) and Dazzle (Chen, PKC 2025), enjoying two rounds, identifiable abort and adaptive security from DDH in pairing-free groups in the ROM. However, both schemes only achieve adaptive TS-UF-0 security. Therefore, our construction yields the *first* TS scheme enjoying adaptive TS-UF-1 security and identifiable abort, free of pairing.

Keywords: Threshold Signature, Adaptive Security, Pairing-Free, Identifiable Abort

1 Introduction

A (t, n) -threshold signature (TS) scheme enriches the functionality of digital signature with distributed signing capability among n signers. With a (t, n) -TS scheme, any t signers can jointly generate a valid signature, while any collusion of fewer than t corrupted signers is unable to forge signatures. By distributing the signing capability, this scheme avoids a single point of signing failure and is resilient to up to $t - 1$ malicious (corrupted) signers.

Adaptive Security of TS. Like digital signature, (t, n) -TS requires unforgeability, i.e., it is hard for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ to forge a valid signature for a fresh message, even if $\mathcal{A}$ is able to corrupt up to $t - 1$ signers to obtain their secret keys and states, besides getting valid signatures from the signers.

A natural way of characterizing corruptions in TS is adaptive corruption, where adversary $\mathcal{A}$ may dynamically decide which signers to corrupt during the protocol execution based on the information obtained so far. Adaptive corruption

captures the realistic attacking behaviors in the deployment of TS, so it is preferable to static corruption where the corrupted signers are determined beforehand. The importance of adaptive security has been highlighted by NIST, which explicitly expressed preference for adaptive security in the threshold-signature standardization efforts. Accordingly, the last couple of years have seen great advances of TS in adaptive security, leading to a variety of TS constructions, including schemes from pairing-friendly groups ([4, 19, 14]), from pairing-free groups ([5, 18, 13, 11, 3, 10, 21, 16, 1, 6, 12]), and from lattices ([18, 17]).

TS-UF-0 Security vs. TS-UF-1 Security. The strength of unforgeability for TS crucially depends on how trivial attacks are defined. Most of the adaptive secure TS schemes fall into two types, namely TS-UF-0 security and TS-UF-1 security. These two notions are adapted from the security hierarchies of unforgeability levels in [7, 9].

In this paper, we consider TS-UF-0/1 security under adaptive corruption, which is simply called TS-UF-0/1 security.

TS-UF-0 security considers $\mathcal{A}$'s forgery on a message m^* as trivial as long as $\mathcal{A}$ has ever obtained any partial signature of any round for m^* from an honest signer (even if no other partial signature on m is obtained). This is an overkill, since a single piece of partial signature does not help $\mathcal{A}$ to compute any valid signature for m^* . The TS schemes in [5, 13, 11, 3, 2, 10, 16, 17, 6] only have adaptive TS-UF-0 security.

TS-UF-1 security is much more reasonable than TS-UF-0: $\mathcal{A}$'s attack is considered trivial only if $\mathcal{A}$ presents a forgery on m^* and it has obtained partial signatures on m^* from t_1 honest signers and corrupted t_2 signers for some $t_2 \geq t - t_1$. Note that $t_1 + t_2 \geq t$ does enable $\mathcal{A}$ to compute a valid signature for m^* by the correctness of (t, n) -TS, so $\mathcal{A}$'s forgery on m^* is indeed a trivial attack. The TS schemes in [4, 19, 14, 18] have TS-UF-1 security.

We note that there are also two works [21, 1] achieving the strongest security of adaptive TS-UF-4. In adaptive TS-UF-4, when a signing set $\mathcal{SS}$ with $|\mathcal{SS}| \geq t$ is determined for message m^* , $\mathcal{A}$'s forgery for m^* is accounted trivial only if $\mathcal{A}$ can obtain *all* partial signatures from *all* signers in $\mathcal{SS}$, either from signing oracles or via corruptions. However, the works [21, 1] rely on masking techniques, which hide the round message, but prevent public verification of partial signatures and make non-interactive identifiable abort difficult to achieve.

Identifiable Abort. Another important property of TS is identifiable abort (IA). IA requires that, whenever a signing protocol fails due to malicious behaviors of some signers, it is possible to identify at least one misbehaving signer and avoid blaming honest signers. This property is crucial for practical deployment of TS: it enables accountability, supports exclusion or penalty mechanisms, and prevents malicious signers from repeatedly disrupting the signing process without being detected. The TS schemes in [4, 19, 14, 5, 13, 3, 2, 10, 16, 6] support IA. Note that only schemes in [4, 19, 14] have both TS-UF-1 security and IA, but all of them take advantage of pairing groups. Up to now there is no TS scheme achieving both TS-UF-1 security and IA in pairing-free groups.

Therefore, we ask the following question.

Can we construct a TS scheme that achieves both adaptive TS-UF-1 security and identifiable abort in pairing-free groups?

Our Contributions. We answer the above question affirmatively by constructing the *first* (t, n) -TS scheme enjoying adaptive TS-UF-1 security and IA in pairing-free groups.

- Our (t, n) -TS construction has only two rounds.
- Our (t, n) -TS construction relies on any cyclic group $\mathbb{G}$ of prime order q , over which the MDDH assumption [15] hold. (t, n) -TS construction is also built from a secret sharing scheme $\text{Share}_{(t,n)}$ over $\mathbb{Z}_q$ and a homomorphic commitment scheme HCOM over $\mathbb{G}$ with special equivocality.
- Our (t, n) -TS construction has TS-UF-1 security in the random oracle model if $\text{Share}_{(t,n)}$ has perfect secrecy and linearity, HCOM has the properties of binding and special equivocality, and the MDDH assumption holds over $\mathbb{G}$.
- Our (t, n) -TS construction has identifiable abort, if $\text{Share}_{(t,n)}$ has linearity and HCOM has the homomorphism property.
- When instantiating $\text{Share}_{(t,n)}$ with the Shamir secret sharing scheme and HCOM with our proposed homomorphic commitment scheme HCOM_{DL} (see Fig. 7), we obtain a specific TS scheme named (t, n) - TS_{MDDH} based on the $\mathcal{U}_{k,k-1}$ -MDDH assumption. For $k = 2$, we obtain a (t, n) - TS_{DDH} scheme based on the DDH assumption.

In Table 1, we compare (t, n) - TS_{MDDH} and (t, n) - TS_{DDH} with existing adaptive TS schemes that avoid pairings and the Algebraic Group Model (AGM).

According to Table 1, only Dazzle [10] and GLRS26 [16] enjoy round efficiency (two rounds) and identifiable abort. However, both schemes only have TS-UF-0 security, whereas our (t, n) - TS_{DDH} retains the same round efficiency and identifiable-abort property while achieving the stronger TS-UF-1 security. Moreover, our (t, n) - TS_{DDH} has efficiency comparable to Dazzle, since it saves one group $\mathbb{G}$ element in the verification key vk , adds only one group $\mathbb{G}$ element and two $\mathbb{Z}_q$ elements in communication (the partial signature in two rounds) and adds only two $\mathbb{Z}_q$ elements in the signature σ . Compared to GLRS26, our (t, n) - TS_{DDH} has four more $\mathbb{Z}_q$ elements in the signature, adds one group $\mathbb{G}$ element but saves two $\mathbb{Z}_q$ elements in communication. Moreover, unlike GLRS26, our (t, n) - TS_{DDH} requires no additional rounds for signers to update their public data.

1.1 Technical Overview

Let $\mathbb{G}$ be a group of order q with generator g . For simplicity, we describe it as an additive group. For $a \in \mathbb{Z}_q$, let $[a] \in \mathbb{G}$ denote ag . If $\mathbf{A}$ is a matrix over $\mathbb{Z}_q$, then $[\mathbf{A}]$ is similarly defined over $\mathbb{G}$. Our (t, n) -TS is based on such a group.

First, given $[\mathbf{A}] \in \mathbb{G}^{k \times k}$, define function $f_{\mathbf{A}} : \mathbb{Z}_q^k \rightarrow \mathbb{G}^k$ with $f_{\mathbf{A}}(\mathbf{x}) = [\mathbf{A}]\mathbf{x} = [\mathbf{A}\mathbf{x}]$. Let $[\mathbf{A}] = \begin{bmatrix} \mathbf{A} \\ \mathbf{A} \end{bmatrix}$ with $\overline{\mathbf{A}} \in \mathbb{Z}_q^{(k-1) \times k}$ and $\underline{\mathbf{A}} \in \mathbb{Z}_q^{1 \times k}$. Let $\mathbf{x} = \mathbf{x}' + \mathbf{a}^\perp \mu$ with $\mathbf{a}^\perp \neq 0$ and $\overline{\mathbf{A}}\mathbf{a}^\perp = 0$. Then we have the following observations.

- If $\mathbf{A}$ is of full rank, then function $f_{\mathbf{A}}$ is an injective function. Moreover, given $[\bar{\mathbf{y}}] := [\bar{\mathbf{A}}\mathbf{x}] = [\bar{\mathbf{A}}\mathbf{x}']$ with $\mathbf{x}' \leftarrow_{\mathcal{S}} \mathbb{Z}_q^k$, $[y] := [\underline{\mathbf{A}}\mathbf{x}] = [\underline{\mathbf{A}}\mathbf{x}' + \underline{\mathbf{A}}\mathbf{a}^\perp\mu]$ is still uniformly distributed over $\mathbb{G}$ due to the uniformity of μ and $\underline{\mathbf{A}}\mathbf{a}^\perp \neq 0$.
- If $\underline{\mathbf{A}}^\top \in \text{Span}(\bar{\mathbf{A}}^\top)$, then function $f_{\mathbf{A}}$ is a lossy function. That is, given $[\mathbf{y}] := [\mathbf{A}\mathbf{x}] = [\mathbf{A}\mathbf{x}']$, then $\mathbf{x} = \mathbf{x}' + \mathbf{a}^\perp\mu$ still has minimal entropy at least $\log q$ due to the uniformity of μ . Moreover, given $[\bar{\mathbf{y}}] := [\bar{\mathbf{A}}\mathbf{x}] = [\bar{\mathbf{A}}\mathbf{x}']$, information-theoretically $[y] := [\underline{\mathbf{A}}\mathbf{x}] = [\mathbf{v}^\top \bar{\mathbf{A}}\mathbf{x}']$ does not leak more information than $[\bar{\mathbf{y}}]$.

To show our construction of TS and why it achieves TS-UF-1 security, we will first present a basic signature scheme, then we use Shamir secret sharing to extend it to scheme (t, n) -TS' with TS-UF-0 security. Finally, we upgrade (t, n) -TS' to a TS-UF-1-secure scheme (t, n) -TS using a homomorphic commitment (HCOM) with special equivocality.

THE BASIC SIGNATURE SCHEME. Let $\bar{\mathbf{A}} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{(k-1) \times k}$ be the public parameter. The signature scheme is constructed from function family $\{f_{\mathbf{A}} \mid \mathbf{A} \in \mathbb{Z}_q^{k \times k}\}$ and two hash functions H_0, H_1 .

Table 1. Comparison of adaptively secure threshold signature schemes without pairings or AGM. The column **Round** denotes the number of communication rounds, **Signature Size** denotes the size of the signature, **Comm.** denotes the communication complexity per signer, $|vk|$ denotes the size of the verification key, and **Assumption** lists the underlying hardness assumptions. The column **TS-UF-level** indicates the unforgeability level, where TS-UF-0+ was presented in [12] and it is stronger than TS-UF-0 and weaker than TS-UF-1. The column **IA?** indicates whether the scheme supports identifiable abort.

TS Scheme	Round	Signature Size	Comm.	$ vk $	Assumption	TS-UF-level	IA?
FROST-Mask [11]	3	$\mathbb{G} + \mathbb{Z}_q$	$2\kappa + 2\mathbb{G} + \mathbb{Z}_q$	$\mathbb{G}$	AOMDL	TS-UF-0	×
HBTS-Mask [11]	2	$\mathbb{G} + 2\mathbb{Z}_q$	$\mathbb{G} + 2\mathbb{Z}_q$	$\mathbb{G}$	DL	TS-UF-0	×
Tweed [17]	2	$38\mathcal{R}_q$	$38\mathcal{R}_q$	$270\mathcal{R}_q$	MLWE	TS-UF-0	×
Twinkle _{DDH} [5]	3	$2\mathbb{G} + 3\mathbb{Z}_q$	$2\kappa + 6\mathbb{G} + 2\mathbb{Z}_q$	$2\mathbb{G}$	DDH	TS-UF-0	✓
Twinkle _{AOMCDH} [5]	3	$\mathbb{G} + 2\mathbb{Z}_q$	$2\kappa + 3\mathbb{G} + \mathbb{Z}_q$	$\mathbb{G}$	AOMCDH	TS-UF-0	✓
Twinkle-T _{DDH} [6]	3	$2\kappa + 2\mathbb{G} + 3\mathbb{Z}_q$	$4\kappa + 6\mathbb{G} + 5\mathbb{Z}_q$	$2\mathbb{G}$	DDH	TS-UF-0	✓
Twinkle-T _{AOMCDH} [6]	3	$2\kappa + \mathbb{G} + 2\mathbb{Z}_q$	$4\kappa + 3\mathbb{G} + 3\mathbb{Z}_q$	$\mathbb{G}$	AOMCDH	TS-UF-0	✓
Glacius [3]	5	$\mathbb{G} + \mathbb{Z}_q$	$3\kappa + 3\mathbb{G} + 6\mathbb{Z}_q$	$\mathbb{G}$	DDH	TS-UF-0	✓
Gargos [2]	3	$\mathbb{G} + \mathbb{Z}_q$	$2\kappa + 2\mathbb{G} + 7\mathbb{Z}_q$	$\mathbb{G}$	DDH	TS-UF-0	✓
Dazzle [10]	2	$\mathbb{G} + 3\mathbb{Z}_q$	$4\mathbb{G} + 2\mathbb{Z}_q$	$2\mathbb{G}$	DDH	TS-UF-0	✓
GLRS26 [16]	2	$\mathbb{G} + \mathbb{Z}_q$	$4\mathbb{G} + 6\mathbb{Z}_q$	$\mathbb{G}$	DDH	TS-UF-0	✓
HBTS2[DDH]-Mask [12]	2	$3\mathbb{Z}_q$	$4\mathbb{G} + 2\mathbb{Z}_q$	$2\mathbb{G}$	DDH	TS-UF-0+	×
TRaccoon [18]	5	$5\mathcal{R}_q + 5\mathcal{R}_{q_{40}}$	$4\kappa + 1463 + 9\mathcal{R}_q$	$5\mathcal{R}_{q_{37}}$	MLWE+MSIS	TS-UF-1	×
TSchnorr(Crackle) [18]	5	$2\mathbb{Z}_q$	$4\kappa + 3\mathbb{Z}_q + \mathbb{G}$	$\mathbb{G}$	DL	TS-UF-1	×
Rackle [21]	3	$2\mathbb{Z}_q$	$4\mathbb{G} + 6\mathbb{Z}_q$	$\mathbb{G}$	DL	TS-SUF-4	×
Our (t, n) -TS _{MDDH}	2	$\mathbb{G} + (k+3)\mathbb{Z}_q$	$(k+3)\mathbb{G} + (k+2)\mathbb{Z}_q$	$(k-1)\mathbb{G}$	MDDH	TS-UF-1	✓
Our (t, n) -TS _{DDH}	2	$\mathbb{G} + 5\mathbb{Z}_q$	$5\mathbb{G} + 4\mathbb{Z}_q$	$\mathbb{G}$	DDH	TS-UF-1	✓

- **Key Generation.** The secret key is $sk := \mathbf{x} \leftarrow_{\$} \mathbb{Z}_q^k$ and the verification key is $vk := [\bar{\mathbf{y}}] := [\bar{\mathbf{A}}] \cdot \mathbf{x}$.
- **Signing.** The signature $\sigma := ([y], \mathbf{z}, c)$ for message m is given by

$$([y] := [\underline{\mathbf{A}}] \cdot \mathbf{x} \in \mathbb{G}, \mathbf{z} := \mathbf{r} + c \cdot \mathbf{x} \in \mathbb{Z}_q^k, c := H_1(vk, m, [y], [\mathbf{w}]) \in \mathbb{Z}_q),$$

where $[\underline{\mathbf{A}}] := H_0(vk, m) \in \mathbb{G}^{1 \times k}$, $\mathbf{r} \leftarrow_{\$} \mathbb{Z}_q^k$ and $[\mathbf{w}] := [\mathbf{A}] \cdot \mathbf{r}$.

- **Verification.** Given $\sigma = ([y], \mathbf{z}, c)$ for m , check $c \stackrel{?}{=} H_1(vk, m, [y], [\mathbf{w}])$ where $[\underline{\mathbf{A}}] := H_0(vk, m)$ and $[\mathbf{w}] := \begin{bmatrix} \mathbf{A} \\ \underline{\mathbf{A}} \end{bmatrix} \cdot \mathbf{z} - c \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y \end{bmatrix}$.

The unforgeability requires that it is hard for adversary $\mathcal{A}$ to forge a valid signature $\sigma^* = ([y^*], \mathbf{z}^*, c^*)$ for a fresh message m^* , even if it adaptively obtains signatures from the signing oracle for messages of its choice.

The proof of unforgeability consists of three steps.

- (i) We show that if the forgery $\sigma^* = ([y^*], \mathbf{z}^*, c^*)$ is valid for m^* , then it must satisfy $[y^*] = [\underline{\mathbf{A}}^*] \mathbf{x}$ with $[\underline{\mathbf{A}}^*] = H_0(vk, m^*)$. This is due to the generalized forking lemma and the KerMDH assumption (see Def. 9). More precisely, by rewinding to $\mathcal{A}$'s hash query (vk, m^*) to H_0 , we may obtain two forgeries $(m^*, \sigma_0^* = ([y^*], \mathbf{z}_0^*, c_0^*))$ and $(m^*, \sigma_1^* = ([y^*], \mathbf{z}_1^*, c_1^*))$ with $c_0^* \neq c_1^*$. The validity of both forgeries implies $\begin{bmatrix} \bar{\mathbf{A}} \\ \underline{\mathbf{A}}^* \end{bmatrix} (\mathbf{z}_0^* - \mathbf{z}_1^*) = (c_0^* - c_1^*) \begin{bmatrix} \bar{\mathbf{y}} \\ y^* \end{bmatrix}$. Then $\tilde{\mathbf{z}} := (\mathbf{z}_0^* - \mathbf{z}_1^*)(c_0^* - c_1^*)^{-1}$ yields $[\bar{\mathbf{A}}] \tilde{\mathbf{z}} = [\bar{\mathbf{y}}]$. Together with $[\bar{\mathbf{A}}] \mathbf{x} = [\bar{\mathbf{y}}]$, we have $[\bar{\mathbf{A}}](\tilde{\mathbf{z}} - \mathbf{x}) = [\mathbf{0}]$. If $[y^*] \neq [\underline{\mathbf{A}}^*] \mathbf{x}$, then $(\tilde{\mathbf{z}} - \mathbf{x}) \neq \mathbf{0}$, which gives a valid solution to the KerMDH problem with challenge $[\bar{\mathbf{A}}]$.
- (ii) Recall that $\mathcal{A}$ is not allowed to query m^* to the signing oracles. Now, we program $H_0(vk, m) = [\mathbf{v}^\top \bar{\mathbf{A}}]$ for $\mathbf{v} \leftarrow_{\$} \mathbb{Z}_q^{k-1}$ if $m \neq m^*$ ¹. This change is indistinguishable due to the MDDH assumption. Accordingly, $f_{\bar{\mathbf{A}}}$ is a lossy function. Thus in any signing-oracle query, $[y] = [\underline{\mathbf{A}}] \mathbf{x}$ leaks no more information about $\mathbf{x}$ than $vk = [\bar{\mathbf{A}}] \mathbf{x}$. Moreover, $\mathbf{z} = \mathbf{r} + c \cdot \mathbf{x}$ does not leak information about $\mathbf{x}$ since $\mathbf{r} \leftarrow_{\$} \mathbb{Z}_q^k$ perfectly hides $\mathbf{x}$ (due to random oracle H_0). Thus the signing oracles leak no more information about $\mathbf{x}$ than vk .
- (iii) Random oracle H_0 guarantees $[\underline{\mathbf{A}}^*] = H_0(vk, m^*)$ is uniform, so $[\mathbf{A}^*] := \begin{bmatrix} \bar{\mathbf{A}} \\ \underline{\mathbf{A}}^* \end{bmatrix}$ is of full rank and $f_{\mathbf{A}^*}$ is injective. Recall that $\mathcal{A}$'s information about $\mathbf{x}$ has been bounded by $vk = [\bar{\mathbf{A}}] \mathbf{x}$. The injectivity of $f_{\mathbf{A}^*}$ implies that $[\mathbf{A}^*] \mathbf{x}$ is uniform, given $vk = [\bar{\mathbf{A}}] \mathbf{x}$. Therefore $[y^*] = [\underline{\mathbf{A}}^*] \mathbf{x}$ hardly holds and the unforgeability of the basic signature scheme follows.

EXTENDING THE BASIC SCHEME TO (t, n) -TS' WITH TS-UF-0 SECURITY. Note the basic scheme has good property of linearity, so we can use Shamir secret sharing to extend it to a TS scheme (t, n) -TS'. Let $\mathcal{SS}$ be the signing set.

¹ This is achieved by guessing strategy in the security proof.

- **Key Generation.** The secret key $\mathbf{x}$ is shared among all n signers with Shamir secret sharing. Signer i obtains $sk_i := \mathbf{x}_i$ and $vk_i := [\bar{\mathbf{A}}]\mathbf{x}_i$ for $i \in [n]$.
- **The 1st Round of Signing.** The 1st-round message of signer ℓ is

$$([y_\ell] := [\underline{\mathbf{A}}]\mathbf{x}_\ell, [\mathbf{w}_\ell] := [\mathbf{A}]\mathbf{r}_\ell),$$

where $[\underline{\mathbf{A}}] := H_0(vk, m) \in \mathbb{G}^{1 \times k}$, $[\mathbf{A}] := \begin{bmatrix} \mathbf{A} \\ \underline{\mathbf{A}} \end{bmatrix}$, and $\mathbf{r}_\ell \leftarrow_s \mathbb{Z}_q^k$ is the state.

- **The 2nd Round of Signing.** The 2nd-round message of signer ℓ is

$$\mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{SS,\ell} \cdot \mathbf{x}_\ell,$$

where $\lambda_{SS,\ell}$ is the Lagrange coefficient, $[y] := \sum_{i \in SS} [y_i] \cdot \lambda_{SS,i}$, $[\mathbf{w}] := \sum_{i \in SS} [\mathbf{w}_i]$, and $c := H_1(vk, m, [y], [\mathbf{w}])$.

- **Combination.** The signature $\sigma := ([y], \mathbf{z}, c)$ of message m is obtained by combining the partial signatures $\{([y_i], [\mathbf{w}_i], \mathbf{z}_i)\}_{i \in SS}$ in the following way.

$$[y] := \sum_{i \in SS} [y_i] \cdot \lambda_{SS,i}, [\mathbf{w}] := \sum_{i \in SS} [\mathbf{w}_i], \mathbf{z} := \sum_{i \in SS} \mathbf{z}_i, c := H_1(vk, m, [y], [\mathbf{w}]).$$

Note that the partial signatures $\{([y_i], [\mathbf{w}_i], \mathbf{z}_i)\}_{i \in SS}$ share the same structure of $([y], [\mathbf{w}], \mathbf{z})$ in the basic signature. Therefore, the partial signature can be verified by $[\mathbf{A}\mathbf{z}_i] \stackrel{?}{=} [\mathbf{w}_i] + c\lambda_{SS,i}[y_i]$ with the help vk_i . Together with authenticated channels, it easy to identify misbehaving signers and achieve IA.

The proof of TS-UF-0 security follows a similar strategy to the basic scheme.

- (i) The same as the basic scheme. Therefore $[y^*] = [\underline{\mathbf{A}}^*]\mathbf{x}$ if $\mathcal{A}$ wins.
- (ii) Let $\mathbf{x} = \mathbf{x}' + \mathbf{a}^\perp \mu$ with $\mathbf{a}^\perp$ is a non-zero kernel vector of $\bar{\mathbf{A}}$ s.t. $\bar{\mathbf{A}}\mathbf{a}^\perp = 0$. Then $sk_\ell = \mathbf{x}_\ell = \mathbf{x}'_\ell + \mathbf{a}^\perp \mu_\ell$, where $\mathbf{x}'_\ell$ and μ_ℓ is the secret share of $\mathbf{x}'$ and μ respectively. Thus $vk_\ell = [\bar{\mathbf{A}}]\mathbf{x}_\ell = [\bar{\mathbf{A}}]\mathbf{x}'_\ell$ perfectly hides the randomness μ_ℓ . According to the TS-UF-0 security, $\mathcal{A}$ is not allowed to query m^* to any signing oracles (for any signer). Thus, similarly we program $H_0(vk, m) = [\mathbf{v}^\top \bar{\mathbf{A}}]$ for $\mathbf{v} \leftarrow_s \mathbb{Z}_q^{k-1}$ as long as $m \neq m^*$. Accordingly, in the signing oracle of signer ℓ for $m \neq m^*$, we have $[y_\ell] = [\underline{\mathbf{A}}]\mathbf{x}_\ell$ does not leak more information about $\mathbf{x}_\ell$ than $vk_\ell = [\bar{\mathbf{A}}]\mathbf{x}_\ell$. Similarly, let $\mathbf{r}_\ell = \mathbf{r}'_\ell + \mathbf{a}^\perp r_\ell$, and then $[\mathbf{w}_\ell] = [\mathbf{A}]\mathbf{r}_\ell = [\mathbf{A}]\mathbf{r}'_\ell$ perfectly hides r_ℓ . Accordingly, $\mathbf{z}_\ell = \mathbf{r}_\ell + c \cdot \lambda_{SS,\ell} \mathbf{x}_\ell = (\mathbf{r}'_\ell + \mathbf{a}^\perp r_\ell) + c \cdot \lambda_{SS,\ell} (\mathbf{x}'_\ell + \mathbf{a}^\perp \mu_\ell)$ and r_ℓ perfectly hides μ_ℓ . Therefore, $\mathcal{A}$ learns no more information about $\{\mu_\ell\}_{\ell \in [n]}$ and μ from the signing oracles.
- (iii) With adaptive corruption, adversary $\mathcal{A}$ can obtain the secret key and state from any signer (limited by fewer than t signer). Our security proof makes it easy to deal with adaptive corruption since the security reductions do not involve sk_i and the secret state, so the reduction algorithm can always generate $\{sk_i\}_{i \in [n]}$ and the corresponding states itself. Recall that $\mathcal{A}$ corrupts at most $t - 1$ signers, so $\mathcal{A}$ obtains at most $t - 1$ shares of μ . The perfect secrecy of Shamir secret sharing ensures that μ is perfectly hidden from $\mathcal{A}$. Consequently, the probability that $[y^*] = [\underline{\mathbf{A}}^*]\mathbf{x} = [\underline{\mathbf{A}}^*]\mathbf{x}' + [\underline{\mathbf{A}}^*]\mathbf{a}^\perp \mu$ holds is at most $1/q$ due to the entropy of μ .

UPGRADING TS-UF-0 TO TS-UF-1. It is hard for us to prove the TS-UF-1 security for the above (t, n) -TS scheme. The obstacle is as follows. If $\mathcal{A}$ is allowed to obtain the first-round messages from t signers for m^* (i.e., $\mathcal{A}$ issues t queries with m^* to the first-round signing oracles of t signers $\{i_j\}_{j \in [t]}$), then $\mathcal{A}$ obtains $\{[y_{i_j}], [\mathbf{w}_{i_j}]\}_{j \in [t]}$. However, $[y_{i_j}] = [\mathbf{A}^*] \cdot (\mathbf{x}'_{i_j} + \mathbf{a}^\perp \mu_{i_j})$ might leak μ_{i_j} since $f_{\mathbf{A}^*}$ is injective, and $\{\mu_{i_j}\}_{j \in [t]}$ can recover μ . As a result, we cannot use the entropy of μ to argue the unforgeability of (t, n) -TS' anymore.

To overcome the above obstacle, it is important to hide the secret key $\mathbf{x}_\ell$ used in the first signing round. To this end, we propose a new variant of equivocal homomorphic commitment called HCOM with special equivocality and obtain our final (t, n) -TS scheme with help of HCOM.

First, we replace $[y_\ell]$ with $com_\ell \leftarrow \text{Commit}([y_\ell]; r'_\ell)$ in the first-round message.

- **The 1st Round of Signing:** The 1st-round message of signer ℓ becomes

$$([com_\ell], [\mathbf{w}_\ell]) \text{ with } com_\ell := \text{Commit}([y_\ell]; r'_\ell) \text{ and the state is } ([y_\ell], r'_\ell, \mathbf{r}_\ell).$$

The special equivocality of HCOM ensures $com_\ell \approx_c \overline{com}_\ell$, where $(\overline{com}_\ell, \overline{\mathbf{st}}_\ell) \leftarrow \text{Commit}(1^\kappa)$. Therefore, the first-round messages hide $\mathbf{x}_\ell$ computationally.

In the second round, we replace $[y]$ with $com := \sum_{i \in SS} \lambda_{SS,i} \cdot \text{Commit}([y_i]; r'_i)$ in the computation of $c := H_1(vk, m, com, [\mathbf{w}])$, and open com_ℓ with $([y_\ell], r'_\ell)$.

- **The 2nd Round of Signing:** The 2nd-round message of signer ℓ becomes

$$([y_\ell], r'_\ell, \mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{SS,\ell} \cdot \mathbf{x}_\ell), \text{ with } c := H_1(vk, m, com, [\mathbf{w}]).$$

- **Combination:** Combining the partial signatures $\{([com_i], [\mathbf{w}_i], [y_i], r'_i, \mathbf{z}_i)\}_{i \in SS}$ yields final signature $\sigma = ([y], r', \mathbf{z}, c)$, where

$$\begin{aligned} [y] &:= \sum_{i \in SS} [y_i] \cdot \lambda_{SS,i}, \quad r' := \sum_{i \in SS} r'_i \cdot \lambda_{SS,i}, \quad \mathbf{z} := \sum_{i \in SS} \mathbf{z}_i, \\ com &:= \sum_{i \in SS} com_i \cdot \lambda_{SS,i}, \quad [\mathbf{w}] := \sum_{i \in SS} [\mathbf{w}_i], \quad c := H_1(vk, m, com, [\mathbf{w}]). \end{aligned}$$

- **Verification.** Given $\sigma = ([y], r', \mathbf{z}, c)$ for m , check $c \stackrel{?}{=} H_1(vk, m, com, [\mathbf{w}])$ where $[\mathbf{A}] := H_0(vk, m)$, $[\mathbf{w}] := \begin{bmatrix} \mathbf{A} \\ \mathbf{A} \end{bmatrix} \cdot \mathbf{z} - c \cdot \begin{bmatrix} \mathbf{y} \\ y \end{bmatrix}$, and $com := \text{Commit}([y]; r')$.

The correctness of (t, n) -TS scheme is ensured by the homomorphic property of HCOM, i.e., $com = \sum_{i \in SS} \lambda_{SS,i} \cdot \text{Commit}([y_i]; r'_i) = \text{Commit}(\sum_{i \in SS} \lambda_{SS,i} [y_i]; \sum_{i \in SS} \lambda_{SS,i} r'_i) = \text{Commit}([y]; r')$. Thus the combiner can compute the final signature with (at least) t partial signatures.

Meanwhile, the partial signature can be verified by $\text{Commit}([y_i]; r'_i) \stackrel{?}{=} com_i$ and $[\mathbf{A} \mathbf{z}_i] \stackrel{?}{=} [\mathbf{w}_i] + c \cdot \lambda_{SS^*,i} \cdot [\mathbf{y}_i]$. Therefore, our (t, n) -TS supports IA as well.

Similar to (t, n) -TS', the proof of TS-UF-1 security for (t, n) -TS also contains (i), (ii), (iii), but (ii') is added between (ii) and (iii), and (iii) is re-analyzed.

- (i),(ii) The same as the proof of (t, n) -TS', except that the binding property of HCOM is needed to ensure $[y^*] = [\mathbf{A}^*]\mathbf{x}$ if $\mathcal{A}$ wins.
- (ii') According to the TS-UF-1 security, $\mathcal{A}$ is allowed to query m^* to any signing oracles (for any signer), as long as $\mathcal{A}$ does not collect enough ($\geq t$) partial signatures $\{(com_i, [\mathbf{w}_i], [y_i], r'_i, \mathbf{z}_i)\}_{i \in SS}$. For such a signing query m^* to the first round signing oracle of signer ℓ , the oracle generates the commitment $\overline{com}_\ell$ with $(\overline{com}_\ell, \overline{st}_\ell) \leftarrow \overline{\text{Commit}}(1^\kappa)$, rather than $com_\ell \leftarrow \text{Commit}([y_\ell]; r'_\ell)$. Later to reply the second-round partial signature, the signing oracle re-explains $\overline{com}_\ell$ as a commitment of $[y_\ell]$ with help of $y_\ell = \mathbf{u}^\top \mathbf{x}_\ell$, where $[\mathbf{A}^*] = H_0(vk, m^*) = [\mathbf{u}^\top]$. This change is indistinguishable to $\mathcal{A}$ due to the special equivocality of HCOM. As a result, the first-round signing for m^* does not leak any information about $\{\mathbf{x}_\ell\}_{\ell \in [n]}$ (thus $\{\mu_\ell\}_{\ell \in [n]}$ as well).
- (iii) According to the TS-UF-1 security, if $\mathcal{A}$ corrupts t_2 signers to obtain their secret keys $\{sk_i = \mathbf{x}_i\}$ and states $\{([y_i], r'_i, \mathbf{r}_i)\}$, then $\mathcal{A}$ obtains at most $t_1 < t - t_2$ second-round messages $\{([y_\ell], r'_\ell, \mathbf{z}_\ell)\}$ w.r.t. m^* from the second-round signing oracles. Therefore $\mathcal{A}$ obtains at most $t - 1$ shares of μ . By an analysis similar to that for (t, n) -TS, we know that μ is uniform to $\mathcal{A}$, and $[y^*] = [\mathbf{A}^*]\mathbf{x} = [\mathbf{A}^*]\mathbf{x}' + [\mathbf{A}^*]\mathbf{a}^\perp \mu$ holds with probability at most $1/q$ due to the entropy of μ . And TS-UF-1 security follows.

Relation to Dazzle [10]. Our (t, n) -TS' scheme is conceptually close to Dazzle, but differs in several important aspects.

- Our (t, n) -TS' scheme is based on the MDDH assumption [15], which naturally generalizes DDH.
- Our (t, n) -TS' scheme from DDH (case $k = 2$ in MDDH) further optimizes the verification key size and the signature size. Unlike Dazzle with $\mathbf{A} \in \mathbb{Z}_q^{2 \times 2}$, our scheme chooses $\mathbf{A} \in \mathbb{Z}_q^{1 \times 2}$. Consequently, the verification key $vk = [\mathbf{A}]\mathbf{x}$ contains only one group element instead of two. Similarly, the committed randomness $[\mathbf{w}] = [\mathbf{A}]\mathbf{r}$ also saves one group element. This optimization relies on the KerMDH assumption [20], which is implied by MDDH.
- Under authenticated channels, we provide an explicit detection algorithm TS.Detect and prove that this scheme achieves identifiable abort (IA).

Moreover, we further extend the (t, n) -TS' scheme to a (t, n) -TS scheme to achieve TS-UF-1 security, with the help of HCOM. This provides security advantage over Dazzle, which only achieves TS-UF-0 security.

2 Preliminaries

Let $n, m \in \mathbb{N}$ with $n > m$ and q be a prime. Denote by $\mathcal{U}_{n,m}$ the uniform distribution over all matrices in $\mathbb{Z}_q^{n \times m}$. Let $[n]$ denote $\{1, \dots, n\}$, $[0, n]$ denote $\{0, 1, \dots, n\}$. If x is a variable, then $y := x$ denotes that we assign the value of x to y . If $\mathcal{S}$ is a set, then $x \leftarrow \mathcal{S}$ denotes that we uniformly sample x from $\mathcal{S}$. We use κ to denote the security parameter.

Let $\mathbb{G}$ be an additive cyclic group of prime order q , and g is the generator of $\mathbb{G}$. We use implicit representation of group elements as in [15]. For $a \in \mathbb{Z}_q$, denote by $[a] = a \cdot g \in \mathbb{G}$ as the implicit representation of a in $\mathbb{G}$. Similarly, for a matrix $\mathbf{A} = (a_{i,j}) \in \mathbb{Z}_q^{n \times m}$, we define $[\mathbf{A}]$ as the implicit representation of $\mathbf{A}$ in $\mathbb{G}$. $\text{Span}(\mathbf{A}) := \{\mathbf{A}\mathbf{r} \mid \mathbf{r} \in \mathbb{Z}_q^m\} \subseteq \mathbb{Z}_q^n$ denotes the linear span of $\mathbf{A}$, and similarly $\text{Span}([\mathbf{A}]) := \{[\mathbf{A}\mathbf{r}] \mid \mathbf{r} \in \mathbb{Z}_q^m\} \subseteq \mathbb{G}^n$.

We write $X \equiv Y$ if X and Y are identically distributed, and write $X \approx_c Y$ to denote the computationally indistinguishability between X and Y .

In Appendix A, we recall the definitions of the $\mathcal{U}_{n,m}$ -MDDH, Q -fold $\mathcal{U}_{n,m}$ -MDDH, and $\mathcal{U}_{n,m}$ -KerMDH assumptions, and the fact that the tight reduction of $\mathcal{U}_{n,m}$ -MDDH $\Rightarrow$ Q -fold $\mathcal{U}_{n,m}$ -MDDH and $\mathcal{U}_{n,m}$ -MDDH $\Rightarrow$ $\mathcal{U}_{n,m}$ -KerMDH. We also recall the Generalized Forking Lemma.

2.1 Secret Sharing Scheme

Definition 1 ((t, n) -Threshold Secret Sharing).

Let $\mathbb{S}$ be a finite additive group with $|\mathbb{S}| \geq 2$. A (t, n) -threshold secret sharing scheme $\text{Share}_{(t,n)}$ for $\mathbb{S}$ with linear reconstruction, perfect secrecy and linearity is defined as follows.

- $\text{Share}_{(t,n)}(s)$ takes $s \in \mathbb{S}$ as input and output n shares $(s_1, s_2, \dots, s_n)$.

Linear Reconstruction. For every subset $\mathcal{SS} \subseteq [n]$ with $|\mathcal{SS}| \geq t$, there exist $\{\lambda_{\mathcal{SS},i}\}_{i \in \mathcal{SS}} \in \mathbb{Z}_q$ s.t. $s = \sum_{i \in \mathcal{SS}} \lambda_{\mathcal{SS},i} \cdot s_i$ for every $(s_1, \dots, s_n) \leftarrow \text{Share}_{(t,n)}(s)$.

Perfect Secrecy. For every subset $\mathcal{SS} \subseteq [n]$ with $|\mathcal{SS}| < t$, for all $a, b \in \mathbb{S}$, if $(s_1, s_2, \dots, s_n) \leftarrow \text{Share}_{(t,n)}(a)$ and $(s'_1, s'_2, \dots, s'_n) \leftarrow \text{Share}_{(t,n)}(b)$, then $\{s_j\}_{j \in \mathcal{SS}} \equiv \{s'_j\}_{j \in \mathcal{SS}}$.

Linearity of $\text{Share}_{(t,n)}$. For all $a, b \in \mathbb{S}$, if $(s'_1, \dots, s'_n) \leftarrow \text{Share}_{(t,n)}(a)$, $(s''_1, \dots, s''_n) \leftarrow \text{Share}_{(t,n)}(b)$ and $(s_1, \dots, s_n) \leftarrow \text{Share}_{(t,n)}(a + b)$. Then, it holds that $(s'_1 + s''_1, \dots, s'_n + s''_n) \equiv (s_1, \dots, s_n)$.

This definition naturally extends to secret vector $\mathbf{s} = (s^{(1)}, \dots, s^{(k)}) \in \mathbb{S}^k$. we can share each coordinate independently using $\text{Share}_{(t,n)}$ to obtain the i -th vector share of $\mathbf{s}$ with $\mathbf{s}_i := (s_i^{(1)}, \dots, s_i^{(k)}) \in \mathbb{S}^k$.

Shamir secret sharing [23] is just a (t, n) -threshold secret sharing. See Appendix B for description of Shamir secret sharing and its security proofs.

2.2 Homomorphic Commitment with Special Equivocality

Homomorphic Commitment (HCOM) with special equivocality scheme is defined over group $\mathbb{G}$ of prime order q . Any element in $\mathbb{G}$ can be written as $[m]$ with $m \in \mathbb{Z}_q$. We describe the operation in $\mathbb{G}$ with “+”. It is homomorphic in both message and randomness. The special equivocality asks that a fake commitment $\overline{com}$ is indistinguishable to a normal one com and $\overline{com}$ can be explained to a commitment of any $[m] \in \mathbb{G}$, but with the help of $m \in \mathbb{Z}_q$.

Definition 2 (Homomorphic Commitment with Special Equivocality).

Let $\mathbb{G}$ be a cyclic group of prime order q with generator g . A homomorphic commitment with special equivocality scheme over group $\mathbb{G}$ is defined by a tuple of PPT algorithms $\text{HCOM} = (\text{CSetup}, \text{Commit}, \text{Open}, \overline{\text{Commit}}, \text{Equiv})$ with message space $\mathbb{G}$, satisfying correctness, binding, special equivocality, and homomorphism.

- $(\text{pp}_{\text{com}}, \text{td}_{\text{com}}) \leftarrow \text{CSetup}(1^\kappa)$: The setup algorithm takes as input the security parameter 1^κ and outputs the public parameters pp_{com} and a trapdoor td_{com} . Here pp_{com} specifies a message space $\mathcal{M}' = \mathbb{G}$, a randomness space $\mathcal{R}$, and a commitment space $\mathcal{C}$, where $\mathcal{R}$ and $\mathcal{C}$ are also additive groups.
- $\text{com} \leftarrow \text{Commit}(\text{pp}_{\text{com}}, [m]; r)$: The commit algorithm takes as input the public parameter pp_{com} , a message $[m] \in \mathbb{G}$ and a randomness $r \in \mathcal{R}$, and it outputs a commitment $\text{com} \in \mathcal{C}$.
- $0/1 \leftarrow \text{Open}(\text{pp}_{\text{com}}, \text{com}, [m], r)$: The opening algorithm takes as input the public parameters pp_{com} , a commitment com , an opening message $[m] \in \mathbb{G}$ and an opening randomness r . It outputs 1 or 0, indicating whether com is a valid commitment w.r.t. m and r .
- $(\overline{\text{com}}, \overline{\text{st}}) \leftarrow \overline{\text{Commit}}(\text{pp}_{\text{com}})$: The fake commit algorithm takes pp_{com} as input and outputs a fake commitment $\overline{\text{com}} \in \mathcal{C}$ and a state $\overline{\text{st}}$.
- $\bar{r} \leftarrow \text{Equiv}(\text{td}_{\text{com}}, \overline{\text{com}}, \overline{\text{st}}, m')$: The equivocation algorithm takes as input the trapdoor td_{com} , a fake commitment $\overline{\text{com}}$, a state $\overline{\text{st}}$, and an opening message $m' \in \mathbb{Z}_q$. It outputs an opening randomness $\bar{r} \in \mathcal{R}$.

$\text{Open}(\text{pp}_{\text{com}}, \text{com}, [m], r)$ can be fulfilled by directly checking $\text{com} \stackrel{?}{=} \text{Commit}(\text{pp}_{\text{com}}, [m]; r)$.

Correctness. For all $[m] \in \mathbb{G}$, all $r \in \mathcal{R}$, if $(\text{pp}_{\text{com}}, \text{td}_{\text{com}}) \leftarrow \text{CSetup}(1^\kappa)$, and $\text{com} \leftarrow \text{Commit}(\text{pp}_{\text{com}}, [m]; r)$, then $\text{Open}(\text{pp}_{\text{com}}, \text{com}, [m], r) = 1$.

Binding. For any PPT adversary $\mathcal{A}$ it holds that, $\text{Adv}_{\text{HE-COM}, \mathcal{A}}^{\text{binding}}(\kappa) :=$

$$\Pr \left[\begin{array}{l} (\text{pp}_{\text{com}}, \text{td}_{\text{com}}) \leftarrow \text{CSetup}(1^\kappa) \\ (\text{com}, [m^0], r^0, [m^1], r^1) \leftarrow \mathcal{A}(\text{pp}_{\text{com}}) \end{array} : \begin{array}{l} [m^0] \neq [m^1] \\ \wedge \text{Open}(\text{pp}_{\text{com}}, \text{com}, [m^0], r^0) = 1 \\ \wedge \text{Open}(\text{pp}_{\text{com}}, \text{com}, [m^1], r^1) = 1 \end{array} \right] = \text{negl}(\kappa).$$

Special Equivocality. For all messages $[m] \in \mathbb{G}$ with $m \in \mathbb{Z}_q$, if $(\text{pp}_{\text{com}}, \text{td}_{\text{com}}) \leftarrow \text{CSetup}(1^\kappa)$, $r \leftarrow \mathcal{R}$, $\text{com} \leftarrow \text{Commit}(\text{pp}_{\text{com}}, [m]; r)$, $(\overline{\text{com}}, \overline{\text{st}}) \leftarrow \overline{\text{Commit}}(\text{pp}_{\text{com}})$, and $\bar{r} \leftarrow \text{Equiv}(\text{td}_{\text{com}}, \overline{\text{com}}, \overline{\text{st}}, m)$ then it holds that $(\text{com}, r) \equiv (\overline{\text{com}}, \bar{r})$ ².

Homomorphism. For all $[m^0], [m^1] \in \mathcal{M}'$, $r^0, r^1 \in \mathcal{R}$, it holds that $\text{Commit}(\text{pp}_{\text{com}}, [m^0] + [m^1]; r^0 + r^1) = \text{Commit}(\text{pp}_{\text{com}}, [m^0]; r^0) + \text{Commit}(\text{pp}_{\text{com}}, [m^1]; r^1)$.

3 Definition of Threshold Signature Scheme

3.1 Syntax of Two-Round (t, n) -TS

Let $n \in \mathbb{N}^+$ denote the total number of signers, and let $t \in \mathbb{N}^+$ be the reconstruction threshold with $t \leq n$. Let $\mathcal{M}$ denote the message space, and $\mathcal{SS}$ represent

² The relation “ $\equiv$ ”, denoting identical distribution, may alternatively be relaxed to computational indistinguishability “ $\approx_c$ ”, i.e., $(\text{com}, r) \approx_c (\overline{\text{com}}, \bar{r})$.

the signing set, where $\mathcal{SS} \subseteq [n]$. Each signer $\ell \in [n]$ maintains its local state $(\text{st}_{1,\ell}, \text{st}_{2,\ell})$ to store the state information generated from the first and second rounds in the current signing session.

Definition 3 (Two-Round Threshold Signature). A two-round (t, n) -threshold signature scheme (t, n) -TS is defined by a tuple of PPT algorithms (TS.Setup, TS.KeyGen, TS.Sign₁, TS.Sign₂, TS.Comb, TS.Vrfy).

- $pp \leftarrow \text{TS.Setup}(1^\kappa)$: The setup algorithm takes as input the security parameter 1^κ and outputs the public parameters pp . This public parameter pp is generated by a trusted setup procedure and serves as an implicit input to the following algorithms below.
- $(vk, \{vk_i\}_{i \in [n]}, \{sk_i\}_{i \in [n]}) \leftarrow \text{TS.KeyGen}(t, n)$: The key generation algorithm takes as input the threshold t and the number of parties n , and outputs a verification key vk , together with verification key shares $\{vk_i\}_{i \in [n]}$ and secret key shares $\{sk_i\}_{i \in [n]}$.
- $(\text{pm}_{1,\ell}, \text{st}_{1,\ell}) \leftarrow \text{TS.Sign}_1(vk, \mathcal{SS}, \ell, sk_\ell, m)$: The first-round signing algorithm takes as input the verification key vk , a signing set $\mathcal{SS} \subseteq [n]$, a signer index $\ell \in \mathcal{SS}$, a secret key share sk_ℓ , and a message $m \in \mathcal{M}$. It outputs a first-round message $\text{pm}_{1,\ell}$ and a local state $\text{st}_{1,\ell}$.
- $(\text{pm}_{2,\ell}, \text{st}_{2,\ell}) \leftarrow \text{TS.Sign}_2(vk, \mathcal{SS}, \ell, sk_\ell, \text{st}_{1,\ell}, m, \{\text{pm}_{1,i}\}_{i \in \mathcal{SS}})$: The second-round signing algorithm takes as input the verification key vk , the signing set $\mathcal{SS}$, a signer index $\ell \in \mathcal{SS}$, a secret key share sk_ℓ , the local state $\text{st}_{1,\ell}$, the message m , and the collection of the first-round messages $\{\text{pm}_{1,i}\}_{i \in \mathcal{SS}}$. It outputs a second-round message $\text{pm}_{2,\ell}$ and an updated state $\text{st}_{2,\ell}$.
- $\sigma \leftarrow \text{TS.Comb}(vk, \mathcal{SS}, m, \{\text{pm}_{1,i}, \text{pm}_{2,i}\}_{i \in \mathcal{SS}})$: The combining algorithm takes as input the verification key vk , a signing set $\mathcal{SS}$, a message m , and a collection of two-round messages $\{\text{pm}_{1,i}, \text{pm}_{2,i}\}_{i \in \mathcal{SS}}$. It outputs a signature σ .
- $0/1 \leftarrow \text{TS.Vrfy}(vk, m, \sigma)$: The verification algorithm takes as input the verification key vk , a message m , and a signature σ . It outputs 1 if the signature is valid, and 0 otherwise.

Correctness of Two-Round Threshold Signature. For all security parameters κ , for all $t, n \in \mathbb{N}^+$ and $t \leq n$, all signer subsets $\mathcal{SS} \subseteq [n]$ with $|\mathcal{SS}| \geq t$, and all messages $m \in \mathcal{M}$, if $pp \leftarrow \text{TS.Setup}(1^\kappa)$, $(vk, \{vk_i\}_{i \in [n]}, \{sk_i\}_{i \in [n]}) \leftarrow \text{TS.KeyGen}(t, n)$, and $(\text{pm}_{1,\ell}, \text{st}_{1,\ell}) \leftarrow \text{TS.Sign}_1(vk, \mathcal{SS}, \ell, sk_\ell, m)$, $(\text{pm}_{2,\ell}, \text{st}_{2,\ell}) \leftarrow \text{TS.Sign}_2(vk, \mathcal{SS}, \ell, sk_\ell, \text{st}_{1,\ell}, m, \{\text{pm}_{1,i}\}_{i \in \mathcal{SS}})$ for all $\ell \in \mathcal{SS}$, then it holds that

$$\text{TS.Vrfy}(vk, m, \text{TS.Comb}(vk, \mathcal{SS}, m, \{\text{pm}_{1,i}, \text{pm}_{2,i}\}_{i \in \mathcal{SS}})) = 1.$$

An (t, n) -TS can be extended to support identifiable abort.

Definition 4 (Two-Round Threshold Signature with Identifiable Abort). A two-round (t, n) -threshold signature scheme (t, n) -TS with identifiable abort is additionally equipped with a detect algorithm, which is used to identify misbehaving signers.

- $\mathcal{J} \leftarrow \text{TS.Detect}(\mathcal{SS}^*, m^*, \{\text{trx}_i\}_{i \in \mathcal{SS}}, \{vk_i\}_{i \in [n]}):$ The detection algorithm takes as input the signing set $\mathcal{SS}^*$, the message m^* , the list of transcripts $\{\text{trx}_i\}_{i \in \mathcal{SS}^*}$, where each transcript trx_i is a tuple of protocol messages, and the verification key shares $\{vk_i\}_{i \in \mathcal{SS}^*}$. It outputs a set $\mathcal{J} \subseteq \mathcal{SS}^*$ of detected malicious signers.

3.2 Adaptive TS-UF-1 Security of Two-Round (t, n) -TS

Adaptive TS-UF-0 security and adaptive TS-UF-1 security are adapted from the security hierarchies of unforgeability levels in [7, 9].

Definition 5 (Adaptive TS-UF-0/TS-UF-1 Security of TS). A two-round (t, n) -threshold signature scheme (t, n) -TS = (TS.Setup, TS.KeyGen, TS.Sign₁, TS.Sign₂, TS.Comb, TS.Vrfy) has adaptive TS-UF- b ($b \in \{0, 1\}$) security, if for any PPT adversary $\mathcal{A}$, it holds that $\text{Adv}_{TS, \mathcal{A}}^{\text{adp-UF-}b}(\kappa) := \Pr[\text{Exp}_{TS, \mathcal{A}}^{\text{adp-UF-}b}(\kappa, t, n) \Rightarrow 1] \leq \text{negl}(\kappa)$, where the experiment $\text{Exp}_{TS, \mathcal{A}}^{\text{adp-UF-}b}$ is defined in Fig. 1.

We stress that we do not need authenticated channels or secure erasures to achieve adaptive TS-UF-0/TS-UF-1 security for (t, n) -TS.

$\text{Exp}_{TS, \mathcal{A}}^{\text{adp-UF-0}}(\kappa, t, n) \setminus \text{Exp}_{TS, \mathcal{A}}^{\text{adp-UF-1}}(\kappa, t, n)$	
1: $\mathcal{Q}_0 := \emptyset, \mathcal{Q}_1[\cdot] := \emptyset, \mathcal{CS} := \emptyset, \mathcal{I}[\cdot, \cdot] := \emptyset$ 2: $\text{round}[\cdot, \cdot] := \emptyset, \text{state}[\cdot, \cdot] := \emptyset$ 3: For $i \in [n]$: 4: $\text{num}_i := 0$ // counting sessions involving signer i 5: $pp \leftarrow \text{TS.Setup}(1^\kappa)$ 6: $(vk, \{vk_i\}_{i \in [n]}, \{sk_i\}_{i \in [n]}) \leftarrow \text{TS.KeyGen}(t, n)$ 7: $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{SIGN}_1, \text{SIGN}_2, \text{CORR}}(pp, vk, \{vk_i\}_{i \in [n]})$ 8: If $ m^* \in \mathcal{Q}_0 $: Return 0 9: If $ \mathcal{Q}_1[m^*] \cup \mathcal{CS} \geq t$: Return 0 10: If $\text{TS.Vrfy}(vk, m^*, \sigma^*) = 1$: Return 1; 11: Else: Return 0 CORR(ℓ) 12: If $ \mathcal{CS} \geq t$: 13: Return $\perp$ 14: $\mathcal{CS} := \mathcal{CS} \cup \{\ell\}$ 15: Return $(sk_\ell, \{\text{state}[\ell, i]\}_{i \in [\text{num}_\ell]})$	SIGN ₁ ($\mathcal{SS}, \ell, m$) 16: If $\ell \in \mathcal{CS}$: Return $\perp$ 17: $\mathcal{Q}_0 := \mathcal{Q}_0 \cup \{m\}$ 18: $\text{num}_\ell := \text{num}_\ell + 1$ 19: $\text{iid} := \text{num}_\ell$ 20: $(\text{pm}_{1, \ell}, \text{st}_{1, \ell}) \leftarrow \text{TS.Sign}_1(vk, \mathcal{SS}, \ell, sk_\ell, m)$ 21: $\text{state}[\ell, \text{iid}] := \text{st}_{1, \ell}$ 22: $\mathcal{I}[\ell, \text{iid}] := (\mathcal{SS}, m)$ 23: $\text{round}[\ell, \text{iid}] := 1$ 24: Return $(\text{iid}, \text{pm}_{1, \ell})$ SIGN ₂ ($\ell, \text{iid}, \{\text{pm}_{1, i}\}_{i \in \mathcal{SS}}$) 25: If $\ell \in \mathcal{CS} \vee \text{round}[\ell, \text{iid}] \neq 1$: Return $\perp$ 26: Parse $\mathcal{I}[\ell, \text{iid}] = (\mathcal{SS}, m)$ 27: Parse $\text{state}[\ell, \text{iid}] = \text{st}_{1, \ell}$ 28: $(\text{pm}_{2, \ell}, \text{st}_{2, \ell}) \leftarrow \text{TS.Sign}_2(vk, \mathcal{SS}, \ell, sk_\ell, \text{st}_{1, \ell}, m, \{\text{pm}_{1, i}\}_{i \in \mathcal{SS}})$ 29: $\text{state}[\ell, \text{iid}] := \text{st}_{2, \ell}$ 30: $\text{round}[\ell, \text{iid}] := 2$ 31: $\mathcal{Q}_1[m] := \mathcal{Q}_1[m] \cup \{\ell\}$ 32: Return $\text{pm}_{2, \ell}$

Fig. 1. The security experiments for a two-round (t, n) -TS scheme: $\text{Exp}_{TS, \mathcal{A}}^{\text{adp-UF-0}}$ [10] (with dashed boxes) and $\text{Exp}_{TS, \mathcal{A}}^{\text{adp-UF-1}}$ (with gray boxes).

3.3 Identifiable Abort of Two-Round (t, n) -TS

Below we adapt the definition of identifiable abort (IA) in [3] to accommodate (t, n) -TS schemes as follows.

$\text{Exp}_{TS, \mathcal{A}}^{\text{adp-IA}}(\kappa, t, n)$ 1: $\mathcal{CS} := \emptyset, \mathcal{I}[\cdot] := \emptyset, \text{pmsg}[\cdot, \cdot] := \emptyset, \text{view}[\cdot, \cdot] := \emptyset$ 2: $\text{round}[\cdot, \cdot] := \emptyset, \text{state}[\cdot, \cdot] := \emptyset, \text{failed} := \emptyset$ 3: $\text{num} := 0$ // counting sessions 4: $pp \leftarrow \text{TS.Setup}(1^\kappa)$ 5: $(vk, \{vk_i\}_{i \in [n]}, \{sk_i\}_{i \in [n]}) \leftarrow \text{TS.KeyGen}(t, n)$ 6: $(\text{sid}^*, \ell^*, \{\text{tr}_i\}_{i \in \mathcal{SS}}) \leftarrow \mathcal{A}^{\text{NEXT}, \text{SIGN}_1, \text{SIGN}_2, \text{COMB}, \text{CORR}}(pp, vk, \{vk_i\}_{i \in [n]})$ 7: $\text{parse } \mathcal{I}[\text{sid}^*] = (SS^*, m^*)$ 8: If $(\ell^* \notin SS^* \setminus \mathcal{CS}) \vee ((\text{sid}^*, \ell^*) \notin \text{failed})$: Return 0 9: If $\mathcal{SS} \neq SS^* \cap \mathcal{CS}$: Return 0 10: For $i \in (SS^* \setminus \mathcal{SS})$: 11: $\text{tr}_i := \text{view}[\text{sid}^*, i]$ 12: $\mathcal{J} \leftarrow \text{TS.Detect}(SS^*, m^*, \{\text{tr}_i\}_{i \in SS^*}, \{vk_i\}_{i \in [n]})$ 13: If $\mathcal{J} \not\subseteq \mathcal{CS} \vee \mathcal{J} \cap SS^* = \emptyset$: 14: Return 1 15: Else: Return 0 16: $\text{NEXT}(SS, m)$ 17: If $ \mathcal{SS} < t \vee SS \not\subseteq [n] \vee m \notin \mathcal{M}$: Return $\perp$ 18: $\text{num} := \text{num} + 1$ 19: $\text{sid} := \text{num}$ 20: $\mathcal{I}[\text{sid}] := (SS, m)$ 21: For $i \in SS$: 22: $\text{round}[i, \text{sid}] := 0$ 23: Return sid 24: $\text{ALLOWED}(\text{sid}, R, \ell, \{\text{pm}_{R-1,i}\}_{i \in \mathcal{SS}})$ // Captures Authenticated Channel 25: If $\text{sid} > \text{num}$: Return 0 26: Parse $\mathcal{I}[\text{sid}] = (SS, m)$ 27: If $\ell \in \mathcal{CS} \vee \ell \notin SS$: Return 0 28: If $\text{round}[\ell, \text{sid}] \neq R - 1$: Return 0 29: If $R = 1$: Return 1 30: If $\mathcal{SS} \neq SS$: Return 0 31: For $i \in (SS \setminus \mathcal{CS})$: 32: If $\text{pm}_{R-1,i} \neq \text{pmsg}[\text{sid}, R - 1, i]$: Return 0 33: Return 1	$\text{CORR}(\ell)$ 34: $\mathcal{CS} := \mathcal{CS} \cup \{\ell\}$ 35: Return $(sk_\ell, \{\text{state}[\ell, i]\}_{i \in [\text{num}]})$ $\text{SIGN}_1(\ell, \text{sid})$ 36: If $\text{ALLOWED}(\text{sid}, 1, \ell, \emptyset) = 0$: Return $\perp$ 37: Parse $\mathcal{I}[\text{sid}] = (SS, m)$ 38: $(\text{pm}_{1,\ell}, \text{st}_{1,\ell}) \leftarrow \text{TS.Sign}_1(vk, SS, \ell, sk_\ell, m)$ 39: $\text{state}[\ell, \text{sid}] := \text{st}_{1,\ell}$ 40: $\text{round}[\ell, \text{sid}] := 1$ 41: $\text{pmsg}[\text{sid}, 1, \ell] := \text{pm}_{1,\ell}$ 42: Return $\text{pm}_{1,\ell}$ $\text{SIGN}_2(\ell, \text{sid}, \{\text{pm}_{1,i}\}_{i \in SS})$ 43: If $\text{ALLOWED}(\text{sid}, 2, \ell, \{\text{pm}_{1,i}\}_{i \in SS}) = 0$: Return $\perp$ 44: Parse $\mathcal{I}[\text{sid}] = (SS, m)$ 45: Parse $\text{state}[\ell, \text{sid}] = \text{st}_{1,\ell}$ 46: $(\text{pm}_{2,\ell}, \text{st}_{2,\ell}) \leftarrow \text{TS.Sign}_2(vk, SS, \ell, sk_\ell, \text{st}_{1,\ell}, m, \{\text{pm}_{1,i}\}_{i \in SS})$ 47: $\text{state}[\ell, \text{sid}] := \text{st}_{2,\ell}$ 48: $\text{round}[\ell, \text{sid}] := 2$ 49: $\text{view}[\text{sid}, \ell] := \{\text{pm}_{1,i}\}_{i \in SS} \cup \{\text{pm}_{2,\ell}\}$ 50: $\text{pmsg}[\text{sid}, 2, \ell] := \text{pm}_{2,\ell}$ 51: Return $\text{pm}_{2,\ell}$ $\text{COMB}(\ell, \text{sid}, \{\text{pm}_{2,i}\}_{i \in SS})$ 52: If $\text{ALLOWED}(\text{sid}, 3, \ell, \{\text{pm}_{2,i}\}_{i \in SS}) = 0$: Return $\perp$ 53: If $(\text{sid}, \ell) \in \text{failed}$: Return $\perp$ 54: Parse $\mathcal{I}[\text{sid}] = (SS, m)$ 55: Parse $\text{view}[\text{sid}, \ell] = \{\text{pm}_{1,i}\}_{i \in SS}$ 56: $\sigma \leftarrow \text{TS.Comb}(vk, SS, m, \{\text{pm}_{1,i}\}_{i \in SS}, \{\text{pm}_{2,i}\}_{i \in SS})$ 57: $\text{view}[\text{sid}, \ell] := \{\text{pm}_{1,i}, \text{pm}_{2,i}\}_{i \in SS}$ 58: If $\text{TS.Vrfy}(vk, m, \sigma) = 0$: 59: $\text{failed} := \text{failed} \cup \{(\text{sid}, \ell)\}$ 60: Return $\perp$ 61: Return σ
--	---

Fig. 2. The identifiable abort experiment $\text{Exp}_{TS, \mathcal{A}}^{\text{adp-IA}}$ for a two-round (t, n) -TS scheme.

Definition 6 (Identifiable Abort of TS). A two-round (t, n) -threshold signature scheme (t, n) -TS supports identifiable abort if it is equipped with TS.Detect algorithm and $\Pr[\text{Exp}_{TS, \mathcal{A}}^{\text{adp-IA}}(\kappa, t, n) \Rightarrow 1] \leq \text{negl}(\kappa)$ for any PPT adversary $\mathcal{A}$, where the experiment $\text{Exp}_{TS, \mathcal{A}}^{\text{adp-IA}}$ is defined in Fig. 2.

For identifiable abort, we assume authenticated channels among signers, and our identifiable abort game captures this via the ALLOWED algorithm. Given a public-key infrastructure (PKI), a non-authenticated channel can be upgraded to an authenticated channel using digital signatures.

4 Construction of Threshold Signatures

The construction of (t, n) -TS' and its adaptive TS-UF-0 security were discussed in the technical overview (see Sect. 1.1). The details of (t, n) -TS' and its security (in fact a stronger notion called TS-UF-0.5) proof are given in Appendix E. Next we present the construction of (t, n) -TS with adaptive TS-UF-1 security.

4.1 Construction of (t, n) -TS

Our construction of (t, n) -TS is based on a cyclic group $\mathbb{G}$ of order q with generator g , and relies on two primitives $\text{Share}_{(t, n)}$ and HCOM . See Fig. 3.

$ \begin{aligned} pp &\leftarrow \text{TS.Setup}(1^\kappa) : \\ &(\mathbb{G}, g, q) \leftarrow \text{GGen}(1^\kappa) \\ &(\text{pp}_{\text{com}}, \text{td}_{\text{com}}) \leftarrow \text{CSetup}(1^\kappa) \\ &\bar{\mathbf{A}} \leftarrow_s \mathbb{Z}_q^{(k-1) \times k} \\ &\text{Return } pp = (\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\bar{\mathbf{A}}]) \end{aligned} $ $ \begin{aligned} (\text{pm}_{1,\ell}, \text{st}_{1,\ell}) &\leftarrow \text{TS.Sign}_1(vk, \mathcal{SS}, \ell, sk_\ell, m) : \\ &\text{Parse } sk_\ell = \mathbf{x}_\ell \\ &[\bar{\mathbf{A}}] := H_0(vk, m) \in \mathbb{G}^{1 \times k} \\ &[\mathbf{A}] := \begin{bmatrix} \bar{\mathbf{A}} \\ \bar{\mathbf{A}} \end{bmatrix} \\ &[y_\ell] := [\bar{\mathbf{A}}] \cdot \mathbf{x}_\ell = [\bar{\mathbf{A}} \mathbf{x}_\ell] \\ &r'_\ell \leftarrow_s \mathcal{R} \\ &\text{com}_\ell \leftarrow \text{Commit}(\text{pp}_{\text{com}}, [y_\ell]; r'_\ell) \\ &\mathbf{r}_\ell \leftarrow_s \mathbb{Z}_q^k \\ &[\mathbf{w}_\ell] := [\mathbf{A}] \cdot \mathbf{r}_\ell = [\mathbf{A} \mathbf{r}_\ell] \\ &\text{pm}_{1,\ell} := (\text{com}_\ell, [\mathbf{w}_\ell]) \in \mathcal{C} \times \mathbb{G}^k \\ &\text{st}_{1,\ell} := ([y_\ell], r'_\ell, \mathbf{r}_\ell) \\ &\text{Return } (\text{pm}_{1,\ell}, \text{st}_{1,\ell}) \end{aligned} $ $ \begin{aligned} (\text{pm}_{2,\ell}, \text{st}_{2,\ell}) &\leftarrow \text{TS.Sign}_2(vk, \mathcal{SS}, \ell, sk_\ell, \text{st}_{1,\ell}, m, \{\text{pm}_{1,i}\}_{i \in \mathcal{SS}}) : \\ &\text{Parse } sk_\ell = \mathbf{x}_\ell, \text{st}_{1,\ell} = ([y_\ell], r'_\ell, \mathbf{r}_\ell) \\ &\text{For } i \in \mathcal{SS}: \text{ Parse } \text{pm}_{1,i} = (\text{com}_i, [\mathbf{w}_i]) \\ &\text{com} := \sum_{i \in \mathcal{SS}} \text{com}_i \cdot \lambda_{\mathcal{SS},i} \\ &[\mathbf{w}] := \sum_{i \in \mathcal{SS}} [\mathbf{w}_i] \\ &c := H_1(vk, m, \text{com}, [\mathbf{w}]) \\ &\mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{\mathcal{SS},\ell} \cdot \mathbf{x}_\ell \\ &\text{pm}_{2,\ell} := ([y_\ell], r'_\ell, \mathbf{z}_\ell) \in \mathbb{G} \times \mathcal{R} \times \mathbb{Z}_q^k \\ &\text{st}_{2,\ell} := \text{st}_{1,\ell} \\ &\text{Return } (\text{pm}_{2,\ell}, \text{st}_{2,\ell}) \end{aligned} $	$ \begin{aligned} (vk, \{vk_i\}_{i \in [n]}, \{sk_i\}_{i \in [n]}) &\leftarrow \text{TS.KeyGen}(t, n) : \\ &\mathbf{x} \leftarrow \mathbb{Z}_q^k \\ &\{\mathbf{x}_i\}_{i \in [n]} \leftarrow \text{Share}_{(t,n)}(\mathbf{x}) \\ &vk := [\bar{\mathbf{y}}] := [\bar{\mathbf{A}}] \cdot \mathbf{x} = [\bar{\mathbf{A}} \mathbf{x}] \\ &\text{For } i \in [n]: \\ &\quad sk_i := \mathbf{x}_i \\ &\quad vk_i := [\bar{\mathbf{y}}_i] := [\bar{\mathbf{A}}] \cdot \mathbf{x}_i = [\bar{\mathbf{A}} \mathbf{x}_i] \\ &\text{Return } (vk, \{vk_i\}_{i \in [n]}, \{sk_i\}_{i \in [n]}) \end{aligned} $ $ \begin{aligned} \sigma &\leftarrow \text{TS.Comb}(vk, \mathcal{SS}, m, \{\text{pm}_{1,i}, \text{pm}_{2,i}\}_{i \in \mathcal{SS}}) : \\ &\text{For } i \in \mathcal{SS}: \\ &\quad \text{Parse } \text{pm}_{1,i} = (\text{com}_i, [\mathbf{w}_i]) \\ &\quad \text{Parse } \text{pm}_{2,i} = ([y_i], r'_i, \mathbf{z}_i) \\ &\quad [y] := \sum_{i \in \mathcal{SS}} [y_i] \cdot \lambda_{\mathcal{SS},i} = [\sum_{i \in \mathcal{SS}} y_i \cdot \lambda_{\mathcal{SS},i}] \\ &\quad r' := \sum_{i \in \mathcal{SS}} r'_i \cdot \lambda_{\mathcal{SS},i} \\ &\quad \text{com} := \sum_{i \in \mathcal{SS}} \text{com}_i \cdot \lambda_{\mathcal{SS},i} \\ &\quad [\mathbf{w}] := \sum_{i \in \mathcal{SS}} [\mathbf{w}_i] \\ &\quad \mathbf{z} := \sum_{i \in \mathcal{SS}} \mathbf{z}_i \in \mathbb{Z}_q^k \\ &\quad c := H_1(vk, m, \text{com}, [\mathbf{w}]) \\ &\quad \sigma := ([y], r', c) \in \mathbb{G} \times \mathcal{R} \times \mathbb{Z}_q^k \times \mathbb{Z}_q \\ &\text{Return } \sigma \end{aligned} $ $ \begin{aligned} 0/1 &\leftarrow \text{TS.Vrfy}(vk, m, \sigma) : \\ &\text{Parse } vk = [\bar{\mathbf{y}}] \\ &\text{Parse } \sigma = ([y], r', \mathbf{z}, c) \\ &[\bar{\mathbf{A}}] := H_0(vk, m) \in \mathbb{G}^{1 \times k} \\ &[\mathbf{w}] := \begin{bmatrix} \bar{\mathbf{A}} \\ \bar{\mathbf{A}} \end{bmatrix} \cdot \mathbf{z} - c \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y \end{bmatrix} \\ &\text{com} := \text{Commit}(\text{pp}_{\text{com}}, [y]; r') \\ &\text{If } c = H_1(vk, m, \text{com}, [\mathbf{w}]): \\ &\quad \text{Return } 1 \\ &\text{Else: Return } 0 \end{aligned} $
--	--

Fig. 3. The Two-Round Threshold Signature scheme (t, n) -TS.

- $\text{Share}_{(t,n)}$ is a (t, n) -threshold secret sharing scheme with secret space $\mathbb{Z}_q^k$ for $k \geq 2$, satisfying linear reconstruction, perfect secrecy, and linearity property.
- $\text{HCOM} = (\text{CSetup}, \text{Commit}, \text{Open}, \text{Commit}, \text{Equiv})$ is a homomorphic commitment scheme with special equivocality. Its message space, randomness space, and commitment space are $\mathcal{M}' := \mathbb{G}$, $\mathcal{R}$, and $\mathcal{C}$, respectively. It satisfies correctness, binding, special equivocality, and homomorphism.

Correctness. Let $pp = (\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\bar{\mathbf{A}}])$ be the output of $\text{TS.Setup}(1^\kappa)$. Suppose that $\text{TS.KeyGen}(t, n)$ outputs $vk = [\bar{\mathbf{y}}] = [\bar{\mathbf{A}} \mathbf{x}]$, $vk_i = [\bar{\mathbf{y}}_i] = [\bar{\mathbf{A}} \mathbf{x}_i]$, $sk_i = \mathbf{x}_i$ for $i \in [n]$, where $\{\mathbf{x}_i\}_{i \in [n]}$ is the output of $\text{Share}_{(t,n)}(\mathbf{x})$.

Suppose that in the first round, $\text{TS.Sign}_1(vk, \mathcal{SS}, \ell, sk_\ell = \mathbf{x}_\ell, m)$ outputs $\text{pm}_{1,\ell} = (\text{com}_\ell, [\mathbf{w}_\ell] = [\mathbf{A} \mathbf{r}_\ell])$ and $\text{st}_{1,\ell} = ([y_\ell], r'_\ell, \mathbf{r}_\ell)$, where $\text{com}_\ell \leftarrow \text{Commit}([y_\ell]; r'_\ell)$ with $[y_\ell] = [\bar{\mathbf{A}} \mathbf{x}_\ell]$, $r'_\ell \leftarrow_s \mathcal{R}$, and $\mathbf{r}_\ell \leftarrow_s \mathbb{Z}_q^k$. Suppose that in the second round, $\text{TS.Sign}_2(vk, \mathcal{SS}, \ell, sk_\ell = \mathbf{x}_\ell, \text{st}_{1,\ell} = ([y_\ell], r'_\ell, \mathbf{r}_\ell), m, \{\text{pm}_{1,i}\}_{i \in \mathcal{SS}})$ outputs $\text{pm}_{2,\ell} = ([y_\ell], r'_\ell, \mathbf{z}_\ell = \mathbf{r}_\ell + c \cdot \lambda_{\mathcal{SS},\ell} \cdot \mathbf{x}_\ell)$ and $\text{st}_{2,\ell} = \text{st}_{1,\ell} = ([y_\ell], r'_\ell, \mathbf{r}_\ell)$, where $c = H_1(vk, m, \text{com}, [\mathbf{w}])$, $\text{com} = \sum_{i \in \mathcal{SS}} \text{com}_i \cdot \lambda_{\mathcal{SS},i}$, and $[\mathbf{w}] = \sum_{i \in \mathcal{SS}} [\mathbf{w}_i] = [\bar{\mathbf{A}} \cdot \sum_{i \in \mathcal{SS}} \mathbf{r}_i]$.

If $\sigma = ([y], r', \mathbf{z}, c)$ is the output of $\text{TS.Comb}(vk, \mathcal{SS}, m, \{\text{pm}_{1,i} = (\text{com}_i, [\mathbf{w}_i] = [\mathbf{A} \mathbf{r}_i]), \text{pm}_{2,i} = ([y_i], r'_i, \mathbf{z}_i = \mathbf{r}_i + c \cdot \lambda_{\mathcal{SS},i} \cdot \mathbf{x}_i)\}_{i \in \mathcal{SS}})$. Due to the linear reconstruc-

tion of secret sharing, for any $\mathcal{SS}$ with $|\mathcal{SS}| \geq t$, we have $\mathbf{x} = \sum_{i \in \mathcal{SS}} \lambda_{\mathcal{SS},i} \mathbf{x}_i$. Accordingly,

$$\begin{cases} [y] &= \sum_{i \in \mathcal{SS}} [y_i] \cdot \lambda_{\mathcal{SS},i} = [\sum_{i \in \mathcal{SS}} \lambda_{\mathcal{SS},i} \mathbf{A} \mathbf{x}_i] = [\mathbf{A} \mathbf{x}], \\ r' &= \sum_{i \in \mathcal{SS}} r'_i \cdot \lambda_{\mathcal{SS},i}, \\ \mathbf{z} &= \sum_{i \in \mathcal{SS}} \mathbf{z}_i = \sum_{i \in \mathcal{SS}} \mathbf{r}_i + c \cdot \sum_{i \in \mathcal{SS}} \lambda_{\mathcal{SS},i} \mathbf{x}_i = \sum_{i \in \mathcal{SS}} \mathbf{r}_i + c \mathbf{x}, \\ c &= H_1(vk, m, com, [\mathbf{w}]). \end{cases}$$

Finally, we analyze the verification algorithm $\text{TS.Vrfy}(vk = [\bar{y}], m, \sigma = ([y], r', \mathbf{z}, c))$. First, by the homomorphism of HCOM, we have

$$\begin{aligned} com' &:= \text{Commit}([y]; r') = \text{Commit}\left(\sum_{i \in \mathcal{SS}} [y_i] \cdot \lambda_{\mathcal{SS},i}; \sum_{i \in \mathcal{SS}} r'_i \cdot \lambda_{\mathcal{SS},i}\right) \\ &= \sum_{i \in \mathcal{SS}} \text{Commit}([y_i]; r'_i) \cdot \lambda_{\mathcal{SS},i} = \sum_{i \in \mathcal{SS}} com_i \cdot \lambda_{\mathcal{SS},i} = com. \end{aligned}$$

Next, observe that

$$[\mathbf{w}'] := \begin{bmatrix} \mathbf{A} \\ \mathbf{A} \end{bmatrix} \cdot \mathbf{z} - c \cdot \begin{bmatrix} \bar{y} \\ y \end{bmatrix} = \begin{bmatrix} \mathbf{A} \cdot (\sum_{i \in \mathcal{SS}} \mathbf{r}_i + c \mathbf{x}) \\ \mathbf{A} \cdot (\sum_{i \in \mathcal{SS}} \mathbf{r}_i + c \mathbf{x}) \end{bmatrix} - \begin{bmatrix} c \cdot \bar{\mathbf{A}} \mathbf{x} \\ c \cdot \mathbf{A} \mathbf{x} \end{bmatrix} = [\mathbf{A} \cdot (\sum_{i \in \mathcal{SS}} \mathbf{r}_i)] = [\mathbf{w}].$$

Therefore, we have $c = H_1(vk, m, com, [\mathbf{w}]) = H_1(vk, m, com', [\mathbf{w}'])$, and $\text{TS.Vrfy}(vk, m, \sigma)$ always outputs 1.

4.2 Adaptive TS-UF-1 Security of (t, n) -TS

Theorem 1. *Suppose that $\text{Share}_{(t,n)}$ is a (t, n) -threshold secret sharing scheme satisfying perfect secrecy and linearity, and HCOM is a homomorphic commitment scheme with special equivocality satisfying binding and special equivocality. If the $\mathcal{U}_{k,k-1}$ -MDDH assumption holds, then our (t, n) -TS scheme (see Fig. 3) has adaptive TS-UF-1 security in the random oracle model. More precisely, for any PPT adversary $\mathcal{A}$ making at most Q_0 random-oracle queries to H_0 and Q_1 queries to H_1 , we have PPT algorithms $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$, s.t. $\text{Adv}_{\text{TS}, \mathcal{A}}^{\text{adp-UF-1}}(\kappa)$*

$$\leq \frac{Q_0^2 + Q_1^2 + 4Q_0 + Q_1 + 3}{q} + \sqrt{Q_1 \cdot \left(\text{Adv}_{\mathbb{G}, \mathcal{B}_1}^{\mathcal{U}_{k,k-1}\text{-MDDH}}(\kappa) + \frac{3}{q} + \text{Adv}_{\text{HE-COM}, \mathcal{B}_2}^{\text{binding}}(\kappa) \right)} + Q_0 \cdot \text{Adv}_{\mathbb{G}, \mathcal{B}_3}^{\mathcal{U}_{k,k-1}\text{-MDDH}}(\kappa).$$

Proof. We define a series of games $\mathbf{G}_0$ - $\mathbf{G}_9$ and define Win as the event that the game outputs 1. Let $\Pr_i[\text{Win}]$ denote the probability that $\mathbf{G}_i$ outputs 1. We will show $\mathbf{G}_0$ is the original $\text{Exp}_{\text{TS}, \mathcal{A}}^{\text{adp-UF-1}}$, and bound $\Pr[\text{Exp}_{\text{TS}, \mathcal{A}}^{\text{adp-UF-1}}(\kappa, t, n) \Rightarrow 1]$ with $\Pr_i[\text{Win}]$ for $i \in [0, 9]$.

Below we describe the games and the difference between two adjacent games are highlighted by gray background.

Game $\mathbf{G}_0$. $\mathbf{G}_0$ is the original security experiment $\text{Exp}_{\text{TS}, \mathcal{A}}^{\text{adp-UF-1}}$ (see Fig. 1), except that both random oracles H_0, H_1 are simulated by lazy sampling. More precisely, $\mathbf{G}_0$ keeps two sets $\text{List}_0, \text{List}_1$ to record the queries to and answers from

H_0, H_1 respectively. G_0 also initializes $\text{ctr}_0 := 0$ and $\text{ctr}_1 := 0$ to count the number of fresh queries to H_0 and H_1 , respectively.

For each query (vk', m) to H_0 , either from $\mathcal{A}$ or from oracle SIGN_1 , if (vk', m) is a fresh query, i.e., $\nexists (\cdot, vk', m, \cdot) \in \text{List}_0$, then G_0 samples $\mathbf{u} \leftarrow_s \mathbb{Z}_q^k$, updates $\text{ctr}_0 := \text{ctr}_0 + 1$, adds $(\text{ctr}_0, vk', m, [\mathbf{u}^\top])$ to List_0 , and returns $[\mathbf{u}^\top]$ as the hash value of $H_0(vk', m)$; otherwise there must exist $(\cdot, vk', m, [\mathbf{u}^\top]) \in \text{List}_0$, and G_0 returns this $[\mathbf{u}^\top]$ as the hash value of $H_0(vk', m)$.

For each query $(vk', m, \text{com}, [\mathbf{w}])$ to H_1 , if $(vk', m, \text{com}, [\mathbf{w}])$ is a fresh query, i.e., $\nexists (\cdot, vk', m, \text{com}, [\mathbf{w}], \cdot) \in \text{List}_1$, then G_0 samples $c \leftarrow_s \mathbb{Z}_q$, updates $\text{ctr}_1 := \text{ctr}_1 + 1$, adds $(\text{ctr}_1, vk', m, \text{com}, [\mathbf{w}], c)$ to List_1 , and returns c as the hash value of $H_1(vk', m, \text{com}, [\mathbf{w}])$; otherwise there must exist $(\cdot, vk', m, \text{com}, [\mathbf{w}], c) \in \text{List}_1$, and then G_0 returns this c as the hash value of $H_1(vk', m, \text{com}, [\mathbf{w}])$.

Suppose that $\mathcal{A}$ outputs a forged signature $\sigma^* = ([y^*], r'^*, \mathbf{z}^*, c^*)$ for a fresh message m^* . Here, fresh message m^* means that $\mathcal{A}$ cannot obtain any valid signature for m^* through oracle queries, i.e., $|\mathcal{Q}_1[m^*] \cup \mathcal{CS}| < t$. Then Win happens in G_0 (i.e., $\text{Exp}_{TS, \mathcal{A}}^{\text{adp-UF-1}} \Rightarrow 1$) iff m^* is fresh and $\text{TS.Vrfy}(vk, m^*, \sigma^*) = 1$. Clearly,

$$\text{Adv}_{TS, \mathcal{A}}^{\text{adp-UF-1}}(\kappa) = \Pr[\text{Exp}_{TS, \mathcal{A}}^{\text{adp-UF-1}}(\kappa, t, n) \Rightarrow 1] = \Pr_0[\text{Win}]. \quad (1)$$

Game G_1 . G_1 is the same as G_0 , except that G_1 directly returns 0 if hash collisions occur for H_0 or H_1 . In other words, G_1 returns 0 if there exists $(\cdot, vk_0, m_0, [\mathbf{u}_0^\top]) \in \text{List}_0$ and $(\cdot, vk_1, m_1, [\mathbf{u}_1^\top]) \in \text{List}_0$ such that $(vk_0, m_0) \neq (vk_1, m_1)$ but $[\mathbf{u}_0^\top] = [\mathbf{u}_1^\top]$ or there exists $(\cdot, vk_0, m_0, \text{com}_0, [\mathbf{w}_0], c_0) \in \text{List}_1$ and $(\cdot, vk_1, m_1, \text{com}_1, [\mathbf{w}_1], c_1) \in \text{List}_1$ such that $(vk_0, m_0, \text{com}_0, [\mathbf{w}_0]) \neq (vk_1, m_1, \text{com}_1, [\mathbf{w}_1])$ but $c_0 = c_1$.

Due to the uniformity of $\mathbf{u}_0^\top, \mathbf{u}_1^\top$ and c_0, c_1 , G_1 returns 0 with probability at most $\frac{Q_0^2}{q^k} + \frac{Q_1^2}{q} \leq \frac{Q_0^2 + Q_1^2}{q}$. Thus we have

$$\Pr_0[\text{Win}] \leq \Pr_1[\text{Win}] + \frac{Q_0^2 + Q_1^2}{q}. \quad (2)$$

Game G_2 . G_2 is the same as G_1 , except that G_2 directly returns 0 if event **Queried1** (defined below) does not occur. Consequently, $\Pr_2[\text{Win}] = \Pr_1[\text{Win} \wedge \text{Queried1}]$.

Given $\mathcal{A}$'s forgery $(m^*, \sigma^* = ([y^*], r'^*, \mathbf{z}^*, c^*))$, let $[\mathbf{A}^*] := H_0(vk, m^*)$, $[\mathbf{w}^*] := \begin{bmatrix} \mathbf{A}^* \\ \mathbf{A}^* \end{bmatrix} \cdot \mathbf{z}^* - c^* \cdot \begin{bmatrix} \bar{y}^* \\ y^* \end{bmatrix}$ and $\text{com}^* := \text{Commit}(\text{pp}_{\text{com}}, [y^*]; r'^*)$. Define

Queried1 : $\mathcal{A}$ has ever issued a query $(vk, m^*, \text{com}^*, [\mathbf{w}^*])$ to oracle H_1 .

In G_1 , we have $\Pr_1[\text{Win}] = \Pr_1[\text{Win} \wedge \text{Queried1}] + \Pr_1[\text{Win} \wedge \neg \text{Queried1}]$. Recall that Win happens means $\text{TS.Vrfy}(vk, m^*, \sigma^*) = 1$, i.e., $c^* = H_1(vk, m^*, \text{com}^*, [\mathbf{w}^*])$.

If **Queried1** does not occur, then $H_1(vk, m^*, \text{com}^*, [\mathbf{w}^*])$ is uniform over $\mathbb{Z}_q$, and hence $c^* = H_1(vk, m^*, \text{com}^*, [\mathbf{w}^*])$ holds with probability $1/q$. Therefore,

$$\Pr_1[\text{Win} \wedge \neg \text{Queried1}] \leq 1/q, \quad \Pr_1[\text{Win}] \leq \Pr_1[\text{Win} \wedge \text{Queried1}] + 1/q.$$

Given $\Pr_2[\text{Win}] = \Pr_1[\text{Win} \wedge \text{Queried1}]$, we have

$$\Pr_1[\text{Win}] \leq \Pr_1[\text{Win} \wedge \text{Queried1}] + 1/q = \Pr_2[\text{Win}] + 1/q. \quad (3)$$

Note that if Win occurs in G_2 , there must exist $(\text{idx}, vk, m^*, com^*, [\mathbf{w}^*], c^*) \in \text{List}_1$ before $\mathcal{A}$ outputs its forgery.

Game G_3 . G_3 is the same as G_2 , except that G_3 returns 0 if either event **CorrectY** or event **Queried0** (defined below) does not occur. Consequently, $\Pr_3[\text{Win}] = \Pr_2[\text{Win} \wedge \text{CorrectY} \wedge \text{Queried0}]$. Define

CorrectY : $\mathcal{A}$'s forgery (m^*, σ^*) satisfies $[y^*] = [\mathbf{A}^*] \cdot \mathbf{x}$, with $[\mathbf{A}^*] := H_0(vk, m^*)$.

Queried0 : $\mathcal{A}$ has ever issued a query (vk, m^*) to oracle H_0 .

Then we have

$$\begin{aligned} \Pr_2[\text{Win}] &= \Pr_2[\text{Win} \wedge \text{CorrectY} \wedge \text{Queried0}] + \Pr_2[\text{Win} \wedge \neg(\text{CorrectY} \wedge \text{Queried0})] \\ &= \Pr_3[\text{Win}] + \Pr_2[\text{Win} \wedge \neg \text{CorrectY}] + \Pr_2[\text{Win} \wedge \text{CorrectY} \wedge \neg \text{Queried0}]. \end{aligned} \quad (4)$$

It remains to prove the following bound, which follows from Claim 1 and Claim 2 below:

$$\Pr_2[\text{Win}] \leq \Pr_3[\text{Win}] + \frac{Q_1 + 2}{q} + \sqrt{Q_1 \cdot \left(\frac{3}{q} + \text{Adv}_{\mathbb{G}, B_1}^{\mathcal{U}_{k, k-1}\text{-MDDH}}(\kappa) + \text{Adv}_{\text{HE-COM}, B_2}^{\text{binding}}(\kappa) \right)}.$$

$$\text{Claim 1. } \Pr_2[\text{Win} \wedge \neg \text{CorrectY}] \leq \frac{Q_1}{q} + \sqrt{Q_1 \cdot \left(\frac{3}{q} + \text{Adv}_{\mathbb{G}, B_1}^{\mathcal{U}_{k, k-1}\text{-MDDH}}(\kappa) + \text{Adv}_{\text{HE-COM}, B_2}^{\text{binding}}(\kappa) \right)}.$$

Proof. To bound the probability that $\text{Win} \wedge \neg \text{CorrectY}$ occurs in G_2 , we use the generalized forking lemma (see Lemma 3).

Firstly, we define an experiment called **Exp** in Fig. 4. **Exp** takes as input a group description $(\mathbb{G}, g, q)$, $[\mathbf{B}] \in \mathbb{G}^{k \times (k-1)}$, $\mathbf{x} \in \mathbb{Z}_q^k$, Q_1 hash values $h_1, h_2, \dots, h_{Q_1}$ and randomness ρ to simulate G_2 and detect $\text{Win} \wedge \neg \text{CorrectY}$ in G_2 .

Description of Exp (see Fig. 4). It first sets $[\mathbf{A}] := [\mathbf{B}^\top]$ and $pp := (\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{A}])$. It then uses $\mathbf{x}$ to generate $\{sk_i = \mathbf{x}_i\}_{i \in [n]}$ via $\{\mathbf{x}_i\}_{i \in [n]} \leftarrow \text{Share}_{(t, n)}(\mathbf{x})$, and computes $vk = [\mathbf{A}\mathbf{x}]$ and $vk_i = [\mathbf{A}\mathbf{x}_i]$. It simulates oracle H_1 by setting $h_1, h_2, \dots, h_{Q_1}$ as the responses of the Q_1 fresh queries to oracle H_1 , resulting in $\text{List}_1 = \{(i, \cdot, \cdot, \cdot, h_i)\}_{i \in [Q_1]}$. It also simulates $\text{SIGN}_1, \text{SIGN}_2, \text{CORR}, H_0$ just like in G_2 .

After $\mathcal{A}$ outputs a forgery (m^*, σ^*) with $\sigma^* = ([y^*], r'^*, \mathbf{z}^*, c^*)$. **Exp** computes $[\mathbf{A}^*] := H_0(vk, m^*)$, $com^* := \text{Commit}(\text{pp}_{\text{com}}, [y^*]; r'^*)$, $[\mathbf{w}^*] := \begin{bmatrix} \mathbf{A} \\ \mathbf{A}^* \end{bmatrix} \cdot \mathbf{z}^* - c^* \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y^* \end{bmatrix}$, and checks that **Queried1** does occur, the message m^* is fresh, the forgery (m^*, σ^*) is valid, hash collision does not occur, and **CorrectY** does not occur. If any of these checks fails, **Exp** returns $(0, \perp)$, otherwise **Exp** retrieves $(\text{idx}, vk, m^*, com^*, [\mathbf{w}^*], c^*)$ from List_1 , and returns $(\text{idx}, (m^*, \sigma^*))$.

We define the success probability of **Exp** as

$$\text{acc} := \Pr[(\text{idx}, (m^*, \sigma^*)) \leftarrow \text{Exp}(\mathbb{G}, g, q, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{Q_1}; \rho) : \text{idx} \neq 0].$$

$\text{Exp}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{Q_1}; \rho)$ 1: $Q_1[\cdot] := \emptyset, \text{CS} := \emptyset, \text{List}_0 := \emptyset, \text{List}_1 := \emptyset$ 2: $\text{ctr}_0 := 0, \text{ctr}_1 := 0$ 3: For $i \in [n]$: $\text{num}_i := 0$ 4: $[\mathbf{A}] := [\mathbf{B}^\top]$ 5: $pp := (\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{A}])$ 6: $\{\mathbf{x}_i\}_{i \in [n]} \leftarrow \text{Share}_{(t,n)}(\mathbf{x})$ 7: $vk := [\mathbf{y}] := [\mathbf{A}] \cdot \mathbf{x} = [\mathbf{A}\mathbf{x}]$ 8: For $i \in [n]$: 9: $vk_i := [\mathbf{y}_i] := [\mathbf{A}] \cdot \mathbf{x}_i = [\mathbf{A}\mathbf{x}_i]$ 10: $sk_i := \mathbf{x}_i$ 11: $(m^*, \sigma^* = ([y^*], r'^*, \mathbf{z}^*, c^*)) \leftarrow \mathcal{A}^{\text{SIGN}_1, \text{SIGN}_2, \text{CORR}, H_0, H_1}(pp, vk, \{vk_i\}_{i \in [n]})$ 12: $[\mathbf{A}^*] := H_0(vk, m^*)$ 13: $[\mathbf{w}^*] := \begin{bmatrix} \mathbf{A} \\ \mathbf{A}^* \end{bmatrix} \cdot \mathbf{z}^* - c^* \cdot \begin{bmatrix} \mathbf{y} \\ y^* \end{bmatrix}$ 14: $\text{com}^* := \text{Commit}(\text{pp}_{\text{com}}, [y^*]; r'^*)$ 15: If $(\cdot, vk, m^*, \text{com}^*, [\mathbf{w}^*], \cdot) \notin \text{List}_1$: 16: Return $(0, \perp)$ //Queried1 does not occur 17: If $ Q_1[m^*] \cup \text{CS} \geq t \vee H_1(vk, m^*, \text{com}^*, [\mathbf{w}^*]) \neq c^*$: 18: Return $(0, \perp)$ //m* is not fresh or σ^* is not valid 19: If $(\exists(\text{ctr}', \cdot, \cdot, \theta), (\text{ctr}', \cdot, \cdot, \theta') \in \text{List}_0 : \text{ctr} \neq \text{ctr}' \wedge \theta = \theta')$ $\vee (\exists i, j \in [Q_1] : i \neq j \wedge h_i = h_j)$: 20: Return $(0, \perp)$ //Hash collisions occurs in H_0 or H_1 21: If $[y^*] = [\mathbf{A}^*] \cdot \mathbf{x}$: 22: Return $(0, \perp)$ //CorrectY occurs 23: Retrieve $(\text{idx}, vk, m^*, \text{com}^*, [\mathbf{w}^*], c^*)$ from List_1 24: Return $(\text{idx}, (m^*, \sigma^*))$	$\text{SIGN}_1(SS, \ell, m)$ 25: The same as G_2 $\text{SIGN}_2(\ell, \text{iid}, \{\text{pm}_{1,i}\}_{i \in SS})$ 26: The same as G_2 $\text{CORR}(\ell)$ 27: The same as G_2 $H_0(vk', m)$ 28: The same as G_2 $H_1(vk', m, \text{com}, [\mathbf{w}])$ 29: If $(\cdot, vk', m, \text{com}, [\mathbf{w}], c) \in \text{List}_1$: 30: Return c 31: $\text{ctr}_1 := \text{ctr}_1 + 1$ 32: $c := h_{\text{ctr}_1}$ 33: $\text{List}_1 := \text{List}_1 \cup \{(\text{ctr}_1, vk', m, \text{com}, [\mathbf{w}], c)\}$ 34: Return c
--	---

Fig. 4. Description of Experiment Exp .

It is easy to see that

$$\Pr_2[\text{Win} \wedge \neg \text{CorrectY}] = \text{acc}. \quad (5)$$

Next, we will show that

$$\Pr_2[\text{Win} \wedge \neg \text{CorrectY}] = \text{acc} \leq \frac{Q_1}{q} + \sqrt{Q_1 \cdot \left(\frac{1}{q} + \text{Adv}_{\mathbb{G}, \mathcal{B}}^{\mathcal{U}_{k,k-1}\text{-KerMDH}}(\kappa) + \text{Adv}_{\text{HE-COM}, \mathcal{B}_2}^{\text{binding}}(\kappa)\right)}.$$

This is achieved by a security reduction to the KerMDH assumption with the help of the generalized forking lemma. Below we construct a reduction algorithm $\mathcal{B}_{\text{KerMDH}}$ accessing Exp . See Fig. 5.

Given a group description $(\mathbb{G}, g, q)$ and $[\mathbf{B}] \in \mathbb{G}^{k \times k-1}$, $\mathcal{B}_{\text{KerMDH}}$ aims to find a $\mathbf{v} \in \mathbb{Z}_q^k$ s.t. $\mathbf{v}^\top \cdot [\mathbf{B}] = [\mathbf{0}] \wedge \mathbf{v} \neq \mathbf{0}$. To this end, $\mathcal{B}_{\text{KerMDH}}$ invokes $(\text{pp}_{\text{com}}, \text{td}_{\text{com}}) \leftarrow \text{CSetup}(1^\kappa)$, samples $\mathbf{x} \leftarrow_{\$} \mathbb{Z}_q^k$, and then invokes $\text{Fork}^{\text{Exp}}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x})$ (see Fig. 8 for Fork). More precisely, it samples random coins ρ and $h_1, \dots, h_{Q_1} \leftarrow_{\$} \mathbb{Z}_q$, and invokes $\text{Exp}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{Q_1}; \rho)$ to obtain $(\text{idx}, (m_0^*, \sigma_0^*))$.

- If $\text{idx} = 0$, it returns $\perp$.
- If $\text{idx} \neq 0$, it samples fresh values $h'_{\text{idx}}, \dots, h'_{Q_1} \leftarrow \mathbb{Z}_q$, and re-invokes $(\text{idx}', (m_1^*, \sigma_1^*)) \leftarrow \text{Exp}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{\text{idx}-1}, h'_{\text{idx}}, \dots, h'_{Q_1}; \rho)$.
 - If $\text{idx} \neq \text{idx}'$, then it returns $\perp$.
 - If $\text{idx} = \text{idx}'$, let $\sigma_0^* = ([y_0^*], r_0'^*, \mathbf{z}_0^*, c_0^*)$ and $\sigma_1^* = ([y_1^*], r_1'^*, \mathbf{z}_1^*, c_1^*)$. Then $c_0^* = h_{\text{idx}}$ and $c_1^* = h'_{\text{idx}}$. If $[y_0^*] \neq [y_1^*] \vee c_0^* = c_1^*$, $\mathcal{B}_{\text{KerMDH}}$ outputs $\perp$. Otherwise, $\mathcal{B}_{\text{KerMDH}}$ defines $\tilde{\mathbf{z}} := (\mathbf{z}_0^* - \mathbf{z}_1^*) / (c_0^* - c_1^*)$ and outputs $\tilde{\mathbf{z}} - \mathbf{x}$.

Algorithm $\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}])$:

```

1:  $(\text{pp}_{\text{com}}, \text{td}_{\text{com}}) \leftarrow \text{CSetup}(1^\kappa)$ 
2:  $\mathbf{x} \leftarrow \mathbb{Z}_q^k$ 
   //Begin Fork $^{\text{Exp}}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x})$ 
3: Choose the random coins  $\rho$  for Exp
4:  $\{h_1, \dots, h_{Q_1}\} \leftarrow \mathbb{Z}_q^{Q_1}$ 
5:  $(\text{idx}, (m_0^*, \sigma_0^*)) \leftarrow \text{Exp}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{Q_1}; \rho)$ 
6: If  $\text{idx} = 0$ : return  $\perp$ 
7:  $\{h'_{\text{idx}}, \dots, h'_{Q_1}\} \leftarrow \mathbb{Z}_q^{Q_1 - \text{idx} + 1}$ 
8:  $(\text{idx}', (m_1^*, \sigma_1^*)) \leftarrow \text{Exp}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{\text{idx}-1}, h'_{\text{idx}}, \dots, h'_{Q_1}; \rho)$ 
9: If  $\text{idx} \neq \text{idx}'$ : return  $\perp$ 
   //End Fork $^{\text{Exp}}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x})$ 
10: Parse  $\sigma_0^* = ([y_0^*], r_0^*, \mathbf{z}_0^*, c_0^*)$ ,  $\sigma_1^* = ([y_1^*], r_1^*, \mathbf{z}_1^*, c_1^*)$ 
11: If  $[y_0^*] \neq [y_1^*]$ : return  $\perp$ 
12: If  $c_0^* = c_1^*$ : return  $\perp$ 
13:  $\tilde{\mathbf{z}} := (\mathbf{z}_0^* - \mathbf{z}_1^*) / (c_0^* - c_1^*)$ 
14: Return  $\tilde{\mathbf{z}} - \mathbf{x}$ 

```

Fig. 5. Description of Algorithm $\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}$.

We define the forking probability $\text{frk} := \Pr[\text{idx} \neq 0 \wedge \text{idx}' = \text{idx}]$. Then

$$\begin{aligned}
\text{frk} &\leq \Pr[c_0^* = c_1^* \wedge \text{idx} \neq 0 \wedge \text{idx}' = \text{idx}] + \Pr[[y_0^*] \neq [y_1^*] \wedge \text{idx} \neq 0 \wedge \text{idx}' = \text{idx}] \\
&\quad + \Pr[\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}]) \Rightarrow (\tilde{\mathbf{z}} - \mathbf{x})] \\
&\leq \Pr[c_0^* = c_1^* \mid \text{idx} \neq 0 \wedge \text{idx}' = \text{idx}] + \Pr[[y_0^*] \neq [y_1^*] \wedge \text{idx} \neq 0 \wedge \text{idx}' = \text{idx}] \\
&\quad + \Pr[\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}]) \Rightarrow (\tilde{\mathbf{z}} - \mathbf{x})] \tag{6}
\end{aligned}$$

Suppose that $\text{idx} \neq 0 \wedge \text{idx}' = \text{idx}$ occurs. Due to the uniformity of h_{idx} and h'_{idx} , we have $c_0^* = h_{\text{idx}} = h'_{\text{idx}} = c_1^*$ with probability $1/q$. Thus

$$\Pr[c_0^* = c_1^* \mid \text{idx} \neq 0 \wedge \text{idx}' = \text{idx}] \leq 1/q. \tag{7}$$

Recall that in the first invocation of Exp, we have $(\text{idx}, vk, m_0^*, \text{com}_0^*, [\mathbf{w}_0^*], c_0^*) \in \text{List}_1$ with $\text{com}_0^* := \text{Commit}(\text{pp}_{\text{com}}, [y_0^*]; r_0^*)$ and $[\mathbf{w}_0^*] = \begin{bmatrix} \mathbf{A} \\ \mathbf{A}_0^* \end{bmatrix} \cdot \mathbf{z}_0^* - c_0^* \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y_0^* \end{bmatrix}$, and the output is $(\text{idx}, (m_0^*, \sigma_0^* = ([y_0^*], r_0^*, \mathbf{z}_0^*, c_0^*)))$. In the second invocation of Exp, we have $(\text{idx}', vk, m_1^*, \text{com}_1^*, [\mathbf{w}_1^*], c_1^*) \in \text{List}_1$ with $\text{com}_1^* := \text{Commit}(\text{pp}_{\text{com}}, [y_1^*]; r_1^*)$ and $[\mathbf{w}_1^*] = \begin{bmatrix} \mathbf{A} \\ \mathbf{A}_1^* \end{bmatrix} \cdot \mathbf{z}_1^* - c_1^* \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y_1^* \end{bmatrix}$, and the output is $(\text{idx}', (m_1^*, \sigma_1^* = ([y_1^*], r_1^*, \mathbf{z}_1^*, c_1^*)))$. Given $\text{idx} = \text{idx}' \neq 0$, and the same random coins ρ for the two invocations of Exp, $\text{Exp}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{\text{idx}-1}, h_{\text{idx}}, \dots, h_{Q_1}; \rho)$ and $\text{Exp}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{\text{idx}-1}, h'_{\text{idx}}, \dots, h'_{Q_1}; \rho)$ are identical until receiving the reply of the idx -th fresh query from oracle H_1 . Obviously, the idx -th query $(vk, m^*, \text{com}^*, [\mathbf{w}^*])$ is the same in the two invocations of Exp. Therefore, we have

$$\begin{aligned}
m_0^* &= m_1^* = m^*, \quad [\mathbf{A}_0^*] = [\mathbf{A}_1^*] = [\mathbf{A}^*] = H_0(vk, m^*), \\
\text{com}_0^* &= \text{com}_1^* = \text{com}^*, \quad [\mathbf{w}_0^*] = [\mathbf{w}_1^*] = [\mathbf{w}^*]. \tag{8}
\end{aligned}$$

Note that $\text{com}_0^* = \text{com}_1^* = \text{com}^*$ means $\text{com}_0^* = \text{Commit}(\text{pp}_{\text{com}}, [y_0^*]; r_0^*) = \text{com}^* = \text{Commit}(\text{pp}_{\text{com}}, [y_1^*]; r_1^*) = \text{com}_1^*$. If $[y_0^*] \neq [y_1^*]$, then the resulting

$(com^*, [y_0^*], r_0'^*, [y_1^*], r_1'^*)$ breaks the binding of HCOM. Thus we have

$$\Pr[[y_0^*] \neq [y_1^*] \wedge \text{idx} \neq 0 \wedge \text{idx}' = \text{idx}] \leq \text{Adv}_{\text{HE-COM}, \mathcal{B}_2}^{\text{binding}}(\kappa). \quad (9)$$

The formal proof is shown in Appendix C.

Finally, let us bound $\Pr[\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}]) \Rightarrow (\tilde{\mathbf{z}} - \mathbf{x})]$. If $\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}])$ outputs $(\tilde{\mathbf{z}} - \mathbf{x})$, then we have $\text{idx} = \text{idx}' \neq 0$, $c_0^* \neq c_1^*$, and $[y_0^*] = [y_1^*]$. According

to (8), we have $\begin{bmatrix} \mathbf{A} \\ \mathbf{A}^* \end{bmatrix} \cdot \mathbf{z}_0^* - c_0^* \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y_0^* \end{bmatrix} = [\mathbf{w}^*] = \begin{bmatrix} \mathbf{A} \\ \mathbf{A}^* \end{bmatrix} \cdot \mathbf{z}_1^* - c_1^* \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y_1^* \end{bmatrix}$.

From $[y_0^*] = [y_1^*]$, we have $\begin{bmatrix} \mathbf{A} \\ \mathbf{A}^* \end{bmatrix} \cdot (\mathbf{z}_0^* - \mathbf{z}_1^*) = (c_0^* - c_1^*) \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y_0^* \end{bmatrix}$. Given $c_0^* \neq c_1^*$, and $\tilde{\mathbf{z}} := (\mathbf{z}_0^* - \mathbf{z}_1^*) \cdot (c_0^* - c_1^*)^{-1}$, we have $\begin{bmatrix} \mathbf{A} \\ \mathbf{A}^* \end{bmatrix} \cdot \tilde{\mathbf{z}} = \begin{bmatrix} \bar{\mathbf{y}} \\ y_0^* \end{bmatrix}$, and hence $[\mathbf{A}] \cdot \tilde{\mathbf{z}} = [\bar{\mathbf{y}}]$ and $[\mathbf{A}^*] \cdot \tilde{\mathbf{z}} = [y_0^*]$.

On the one hand, we have

$$[\mathbf{A}] \cdot \tilde{\mathbf{z}} = [\bar{\mathbf{y}}] = [\mathbf{A}] \cdot \mathbf{x} \Rightarrow [\mathbf{A}] \cdot (\tilde{\mathbf{z}} - \mathbf{x}) = \mathbf{0} \Rightarrow [\mathbf{B}^\top] \cdot (\tilde{\mathbf{z}} - \mathbf{x}) = \mathbf{0}.$$

On the other hand, $\neg \text{CorrectY} \Rightarrow [\mathbf{A}^*] \cdot \mathbf{x} \neq [y_0^*]$. Together with $[\mathbf{A}^*] \cdot \tilde{\mathbf{z}} = [y_0^*]$, we have $\tilde{\mathbf{z}} - \mathbf{x} \neq \mathbf{0}$.

Therefore, the output $\tilde{\mathbf{z}} - \mathbf{x}$ of $\mathcal{B}_{\text{KerMDH}}$ is a valid non-zero solution to the KerMDH problem. So

$$\Pr[\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}]) \Rightarrow (\tilde{\mathbf{z}} - \mathbf{x})] \leq \text{Adv}_{\mathbb{G}, \mathcal{B}}^{\mathcal{U}_{k, k-1}\text{-KerMDH}}(\kappa). \quad (10)$$

Combining (6), (7), (9), (10) yields

$$\text{frk} \leq \frac{1}{q} + \text{Adv}_{\mathbb{G}, \mathcal{B}}^{\mathcal{U}_{k, k-1}\text{-KerMDH}}(\kappa) + \text{Adv}_{\text{HE-COM}, \mathcal{B}_2}^{\text{binding}}(\kappa). \quad (11)$$

According to the generalized forking lemma (see Lemma 3), we have

$$\text{acc} \leq \frac{Q_1}{q} + \sqrt{Q_1 \cdot \text{frk}}. \quad (12)$$

Combining (5), (11), (12) yields

$$\Pr_2[\text{Win} \wedge \neg \text{CorrectY}] = \text{acc} \leq \frac{Q_1}{q} + \sqrt{Q_1 \cdot \left(\frac{1}{q} + \text{Adv}_{\mathbb{G}, \mathcal{B}}^{\mathcal{U}_{k, k-1}\text{-KerMDH}}(\kappa) + \text{Adv}_{\text{HE-COM}, \mathcal{B}_2}^{\text{binding}}(\kappa) \right)}.$$

According to Lemma 2, we have a PPT algorithm $\mathcal{B}_1$ such that

$$\Pr_2[\text{Win} \wedge \neg \text{CorrectY}] \leq \frac{Q_1}{q} + \sqrt{Q_1 \cdot \left(\frac{3}{q} + \text{Adv}_{\mathbb{G}, \mathcal{B}_1}^{\mathcal{U}_{k, k-1}\text{-MDDH}}(\kappa) + \text{Adv}_{\text{HE-COM}, \mathcal{B}_2}^{\text{binding}}(\kappa) \right)}.$$

□

Claim 2. $\Pr_3[\text{Win} \wedge \text{CorrectY} \wedge \neg \text{Queried0}] \leq \frac{2}{q}$.

Proof. If **Queried0** does not occur in the game, then $\mathcal{A}$ must have not queried $H_0(vk, m^*)$. Thus, $[\mathbf{A}^*] := H_0(vk, m^*)$ is uniformly distributed. Given that $\mathbf{x}$ is uniformly sampled from $\mathbb{Z}_q^k$, thus $\mathbf{x} \neq \mathbf{0}$ except with probability $\frac{1}{q^k}$. If $\mathbf{x} \neq \mathbf{0}$, then $[\mathbf{A}^*] \cdot \mathbf{x}$ is uniformly distributed, and hence $[y^*] = [\mathbf{A}^*] \cdot \mathbf{x}$ with probability $\frac{1}{q}$. Consequently, we have $\Pr_3[\text{Win} \wedge \text{CorrectY} \wedge \neg \text{Queried0}] \leq \frac{1}{q} + \frac{1}{q^k} \leq \frac{2}{q}$. $\square$

Combining (4), Claim 1 and Claim 2, we have

$$\Pr_2[\text{Win}] \leq \Pr_3[\text{Win}] + \frac{Q_1+2}{q} + \sqrt{Q_1 \cdot \left(\frac{3}{q} + \text{Adv}_{\mathbb{G}, \mathcal{B}_1}^{\mathcal{U}_{k,k-1}\text{-MDDH}}(\kappa) + \text{Adv}_{\text{HE-COM}, \mathcal{B}_2}^{\text{binding}}(\kappa)\right)}. \quad (13)$$

Game $\mathcal{G}_4$. $\mathcal{G}_4$ samples $\text{ctr}^* \leftarrow_{\$} [Q_0]$, and checks if **GoodGuess** (defined below) occurs, where event **GoodGuess** is defined as

GoodGuess : The ctr^* -th fresh hash query (vk', m) to H_0 satisfies $vk' = vk \wedge m = m^*$.

$\mathcal{G}_4$ is the same as $\mathcal{G}_3$, except that $\mathcal{G}_4$ returns 0 if **GoodGuess** does not occur. Consequently, $\Pr_4[\text{Win}] = \Pr_3[\text{Win} \wedge \text{GoodGuess}]$.

If **Win** occurs in $\mathcal{G}_4$, then **Queried0** must occur (thanks to $\mathcal{G}_3$). Due to the uniformity of ctr^* , we have $\Pr_3[\text{GoodGuess}] \geq \frac{1}{Q_0}$.

Note that **GoodGuess** is independent of $\mathcal{A}$'s view, so we have $\Pr_4[\text{Win}] = \Pr_3[\text{Win} \wedge \text{GoodGuess}] = \Pr_3[\text{Win}] \cdot \Pr_3[\text{GoodGuess}] \geq \frac{1}{Q_0} \cdot \Pr_3[\text{Win}]$. Therefore,

$$\Pr_3[\text{Win}] \leq Q_0 \cdot \Pr_4[\text{Win}] \quad (14)$$

Game $\mathcal{G}_5$. $\mathcal{G}_5$ is the same as $\mathcal{G}_4$, except that $\mathcal{G}_5$ changes the way of answering hash queries to H_0 . More precisely, upon the ctr -th fresh query (vk', m) to oracle H_0 for $\text{ctr} \in [Q_0]$, $\mathcal{G}_5$ replies this query with

$$\begin{cases} H_0(vk', m) := [\mathbf{u}_{\text{ctr}}^\top] \text{ with } \mathbf{u}_{\text{ctr}} \leftarrow_{\$} \mathbb{Z}_q^k, & \text{if } \text{ctr} = \text{ctr}^* \\ H_0(vk', m) := \mathbf{v}_{\text{ctr}}^\top [\overline{\mathbf{A}}] \text{ with } \mathbf{v}_{\text{ctr}} \leftarrow_{\$} \mathbb{Z}_q^{k-1}, & \text{if } \text{ctr} \neq \text{ctr}^*. \end{cases}$$

Recall that in $\mathcal{G}_4$, $H_0(vk', m) := [\mathbf{u}_{\text{ctr}}^\top]$ with $\mathbf{u}_{\text{ctr}} \leftarrow_{\$} \mathbb{Z}_q^k$ for the ctr -th fresh hash query $H_0(vk', m)$ for all $\text{ctr} \in [Q_0]$. We show that $\mathcal{G}_4$ and $\mathcal{G}_5$ are indistinguishable due to the $Q_0\text{-}\mathcal{U}_{k,k-1}\text{-MDDH}$ assumption, as shown by Claim 3.

Claim 3. $|\Pr_5[\text{Win}] - \Pr_4[\text{Win}]| \leq \text{Adv}_{\mathbb{G}, \mathcal{B}'_3}^{Q_0\text{-}\mathcal{U}_{k,k-1}\text{-MDDH}}(\kappa)$.

Proof. To prove Claim 3, we construct a PPT algorithm $\mathcal{B}'_3$ against the $Q_0\text{-}\mathcal{U}_{k,k-1}\text{-MDDH}$ assumption. Given the challenge $([\mathbf{B}], [\mathbf{C}] = [\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{Q_0}])$, $\mathcal{B}'_3$ has to tell whether $[\mathbf{C}] = [\mathbf{B}\mathbf{V}] = [\mathbf{B}\mathbf{v}_1, \mathbf{B}\mathbf{v}_2, \dots, \mathbf{B}\mathbf{v}_{Q_0}]$ or $[\mathbf{C}] = [\mathbf{U}] = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{Q_0}]$.

Now $\mathcal{B}'_3$ simulates the game for $\mathcal{A}$ just like $\mathcal{G}_5$, except that $\mathcal{B}'_3$ sets $[\overline{\mathbf{A}}] := [\mathbf{B}^\top]$, and answers the ctr -th fresh query (vk', m) to oracle H_0 as follows,

$$\begin{cases} H_0(vk', m) := [\mathbf{u}_{\text{ctr}}^\top] \text{ with } \mathbf{u}_{\text{ctr}} \leftarrow_{\$} \mathbb{Z}_q^k, & \text{if } \text{ctr} = \text{ctr}^* \\ H_0(vk', m) := [\mathbf{c}_{\text{ctr}}^\top], & \text{if } \text{ctr} \neq \text{ctr}^*. \end{cases}$$

$\mathcal{B}'_3$ can perfectly simulate oracles $\text{SIGN}_1, \text{SIGN}_2, \text{CORR}$ and H_1 , since it generates $\text{pp}_{\text{com}}, vk, \{sk_i\}_{i \in [n]}, \{vk_i\}_{i \in [n]}$ by itself.

Finally, $\mathcal{B}'_3$ receives a forgery (m^*, σ^*) from $\mathcal{A}$, and $\mathcal{B}'_3$ outputs 1 iff m^* is fresh, σ^* is valid and $\text{Queried1} \wedge \text{CorrectY} \wedge \text{Queried0} \wedge \text{GoodGuess}$ occurs.

It is easy to see that $\mathcal{B}'_3$ perfectly simulates G_4 for $\mathcal{A}$ if $[\mathbf{C}] = [\mathbf{U}]$, and $\mathcal{B}'_3$ perfectly simulates G_5 for $\mathcal{A}$ if $[\mathbf{C}] = [\mathbf{BV}]$. Meanwhile $\mathcal{B}'_3$ outputs 1 iff Win occurs. Hence,

$$|\Pr_5[\text{Win}] - \Pr_4[\text{Win}]| \leq \text{Adv}_{\mathbb{G}, \mathcal{B}'_3}^{Q_0 - \mathcal{U}_{k, k-1} - \text{MDDH}}(\kappa).$$

□

Therefore, we have $\Pr_4[\text{Win}] \leq \Pr_5[\text{Win}] + \text{Adv}_{\mathbb{G}, \mathcal{B}'_3}^{Q_0 - \mathcal{U}_{k, k-1} - \text{MDDH}}(\kappa)$.

According to the Lemma 1, we have a PPT algorithm $\mathcal{B}_3$ such that

$$\Pr_4[\text{Win}] \leq \Pr_5[\text{Win}] + \text{Adv}_{\mathbb{G}, \mathcal{B}_3}^{\mathcal{U}_{k, k-1} - \text{MDDH}}(\kappa) + 2/q. \quad (15)$$

Game G_6 . G_6 is the same as G_5 , except for the replies of the signing oracles $\text{SIGN}_1, \text{SIGN}_2$ and corruption oracle CORR when (vk, m) is the ctr^* -th fresh random oracle query to H_0 . In G_6 , if (vk, m) is the ctr^* -th fresh random oracle query to H_0 , $\text{SIGN}_1(\mathcal{SS}, \ell, m)$ invokes $\overline{\text{Commit}}$ to generate a fake commitment $\overline{com}$ and $\overline{st}$ without using $[y_\ell]$, and later $\text{SIGN}_2, \text{CORR}$ opens $\overline{com}$ to $([y_\ell], \overline{r}_\ell)$ via invoking Equiv . More precisely,

- For any query $(\mathcal{SS}, \ell, m)$ to SIGN_1 , if $\ell \in \mathcal{CS}$, SIGN_1 returns $\perp$. Otherwise SIGN_1 sets $\text{num}_\ell := \text{num}_\ell + 1$, $\text{iid} := \text{num}_\ell$ and obtains $[\mathbf{A}] := H_0(vk, m)$.
 - If (vk, m) is the ctr^* -th fresh query to H_0 , then SIGN_1 proceeds as follows. It first invokes $(\overline{com}_\ell, \overline{st}_\ell) \leftarrow \overline{\text{Commit}}(\text{pp}_{\text{com}}, 1^\kappa)$ and samples $\mathbf{r}_\ell \leftarrow \mathbb{Z}_q^k$. It then sets $[\mathbf{w}_\ell] := \begin{bmatrix} \mathbf{A} \\ \mathbf{A} \end{bmatrix} \cdot \mathbf{r}_\ell$. Finally, it returns $(\text{iid}, \text{pm}_{1,\ell} := (\overline{com}_\ell, [\mathbf{w}_\ell]))$, and stores the local state $\text{state}[\ell, \text{iid}] := \text{st}_{1,\ell} := (\overline{com}_\ell, \overline{st}_\ell, \mathbf{r}_\ell)$.
 - Otherwise, SIGN_1 behaves just the same as in G_5 .
- For any query $(\ell, \text{iid}, \{\text{pm}_{1,i}\}_{i \in \mathcal{SS}})$ for SIGN_2 , SIGN_2 retrieves $\mathcal{I}[\ell, \text{iid}] = (\mathcal{SS}, m)$ and sets $\mathcal{Q}_1[m] := \mathcal{Q}_1[m] \cup \{\ell\}$. Then, SIGN_2 combines the first-round message $\{\text{pm}_{1,i}\}_{i \in \mathcal{SS}}$ to obtain com and $[\mathbf{w}]$. Then, it computes $c := H_1(vk, m, com, [\mathbf{w}])$.
 - If (vk, m) is the ctr^* -th fresh query to H_0 , suppose that $H_0(vk, m) = [\mathbf{u}_{\text{ctr}^*}^\top]$. Then SIGN_2 parses $\text{st}_{1,\ell} := (\overline{com}_\ell, \overline{st}_\ell, \mathbf{r}_\ell)$ and computes $[y_\ell] := \mathbf{u}_{\text{ctr}^*}^\top \cdot \mathbf{x}_\ell$. Next, SIGN_2 computes $\mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{\mathcal{SS}, \ell} \cdot \mathbf{x}_\ell$ and invokes $[\overline{r}_\ell] \leftarrow \text{Equiv}(\text{td}_{\text{com}}, \overline{com}_\ell, \overline{st}_\ell, y_\ell)$. Finally, SIGN_2 returns $\text{pm}_{2,\ell} := ([y_\ell], [\overline{r}_\ell], \mathbf{z}_\ell)$ and updates its local state $\text{state}[\ell, \text{iid}] := \text{st}_{2,\ell} := ([y_\ell], [\overline{r}_\ell], \mathbf{r}_\ell)$.
 - Otherwise, SIGN_2 behaves just the same as in G_5 .
- For any corrupt query ℓ , oracle CORR updates the internal states $\text{state}[\ell, \text{iid}]$ for $\text{iid} \in [\text{num}_\ell]$ accordingly. More precisely, for each $\text{iid} \in [\text{num}_\ell]$, CORR retrieves $\mathcal{I}[\ell, \text{iid}] = (\mathcal{SS}, m)$ and does the following.

- If (vk, m) is the ctr^* -th fresh query to H_0 and $\text{round}[\ell, \text{iid}] = 1$ (i.e., $\mathcal{A}$ did not ever query $\text{SIGN}_2(\ell, \text{iid}, \cdot)$ yet), suppose that $H_0(vk, m) = [\mathbf{u}_{\text{ctr}^*}^\top]$. Then CORR retrieves $\text{state}[\ell, \text{iid}] = (\overline{\text{com}}_\ell, \overline{\text{st}}_\ell, \mathbf{r}_\ell)$, and computes $\underline{y}_\ell := \mathbf{u}_{\text{ctr}^*}^\top \cdot \mathbf{x}_\ell$. Next, CORR invokes $\bar{r}_\ell \leftarrow \text{Equiv}(\text{td}_{\text{com}}, \overline{\text{com}}_\ell, \overline{\text{st}}_\ell, \underline{y}_\ell)$ and sets $\text{state}[\ell, \text{iid}] := ([\underline{y}_\ell], \bar{r}_\ell, \mathbf{r}_\ell)$.
 - Otherwise, CORR retrieves $\text{state}[\ell, \text{iid}]$ just like G_5 .
- Finally CORR returns signer ℓ 's secret key sk_ℓ and its (updated) internal state $\{\text{state}[\ell, i]\}_{i \in [\text{num}_\ell]}$.

Recall that in G_5 , $r'_\ell \leftarrow \mathcal{R}$ and $\text{com}_\ell \leftarrow \text{Commit}(\text{pp}_{\text{com}}, [\underline{y}_\ell]; r'_\ell)$. In G_6 , by contrast, $(\overline{\text{com}}_\ell, \overline{\text{st}}_\ell) \leftarrow \overline{\text{Commit}}(\text{pp}_{\text{com}})$ and $\bar{r}_\ell \leftarrow \text{Equiv}(\text{td}_{\text{com}}, \overline{\text{com}}_\ell, \overline{\text{st}}_\ell, \underline{y}_\ell)$. According to the special equivocality of HCOM, we have $(\text{com}_\ell, r'_\ell) \equiv (\overline{\text{com}}_\ell, \bar{r}_\ell)$. Therefore, G_6 is identical to G_5 and hence

$$\Pr_5[\text{Win}] = \Pr_6[\text{Win}]. \quad (16)$$

Game G_7 . G_7 is the same as G_6 , except that G_7 generates $\mathbf{x}$ and $\mathbf{x}_i$ in an equivalent way. More precisely,

- For the key generation, G_7 chooses a non-zero vector $\mathbf{a}^\perp \in \mathbb{Z}_q^k$, such that $\overline{\mathbf{A}}\mathbf{a}^\perp = \mathbf{0}$. It then samples $\mathbf{x}' \leftarrow \mathbb{Z}_q^k$ and $\mu \leftarrow \mathbb{Z}_q$, and sets $\mathbf{x} := \mathbf{x}' + \mathbf{a}^\perp \mu$. Next, it invokes $\{\mathbf{x}'_i\}_{i \in [n]} \leftarrow \text{Share}_{(t,n)}(\mathbf{x}')$, $\{\mu_i\}_{i \in [n]} \leftarrow \text{Share}_{(t,n)}(\mu)$ and sets $sk_i := \mathbf{x}_i = \mathbf{x}'_i + \mathbf{a}^\perp \mu_i$.

Next we will show that G_7 is identical to G_6 .

Due to $\mathbf{x}' \leftarrow \mathbb{Z}_q^k$ and the independence between $\mathbf{x}'$ and $\mathbf{a}^\perp \mu$, we know that $\mathbf{x}$ is uniform over $\mathbb{Z}_q^k$, just like G_6 .

Due to the linearity and perfect secrecy of $\text{Share}_{(t,n)}$, we know that $\mathbf{x}_i := \mathbf{x}'_i + \mathbf{a}^\perp \mu_i$ is an output of $\text{Share}_{(t,n)}(\mathbf{x})$ and $\{sk_i = \mathbf{x}_i\}_{i \in [n]}$ in G_7 has the same distribution as in G_6 . Therefore, G_7 is identical to G_6 and hence

$$\Pr_6[\text{Win}] = \Pr_7[\text{Win}]. \quad (17)$$

Game G_8 . G_8 is the same as G_7 , except for the key generation and replies of the first-round signing oracle SIGN_1 .

- For the key generation, G_8 generates $\mathbf{x} = \mathbf{x}' + \mathbf{a}^\perp \mu$ and $sk_i := \mathbf{x}_i = \mathbf{x}'_i + \mathbf{a}^\perp \mu_i$ just like G_7 . Then G_8 sets $vk := [\overline{\mathbf{A}}\mathbf{x}']$ and $vk_i := [\overline{\mathbf{A}}\mathbf{x}'_i]$ for all $i \in [n]$, rather than setting $vk := [\overline{\mathbf{A}}\mathbf{x}]$ and $vk_i := [\overline{\mathbf{A}}\mathbf{x}_i]$ as in G_7 .
- For any query $(\mathcal{SS}, \ell, m)$ to SIGN_1 , if $\ell \in \mathcal{CS}$, SIGN_1 returns $\perp$. Otherwise SIGN_1 sets $\text{num}_\ell := \text{num}_\ell + 1$, $\text{iid} := \text{num}_\ell$, and obtains $[\mathbf{A}] := H_0(vk, m)$, just like G_7 . Next, SIGN_1 replies the query $(\mathcal{SS}, \ell, m)$ in two cases.

- If (vk, m) is the ctr^* -th fresh random oracle query to H_0 , SIGN_1 behaves just the same as in G_7 .
- If (vk, m) is not the ctr^* -th fresh random oracle query to H_0 , then SIGN_1 sets $[y_\ell] := [\mathbf{A}] \cdot \mathbf{x}'_\ell$, rather than setting $[y_\ell] := [\mathbf{A}] \cdot \mathbf{x}_\ell$ as in G_7 .
Next, G_8 uses $[y_\ell]$ to generate com_ℓ , samples $\mathbf{r}_\ell \leftarrow \mathbb{Z}_q^k$ to generate $[\mathbf{w}_\ell]$, returns $(\text{iid}, \text{pm}_{1,\ell})$ and locally stores $\text{state}[\ell, \text{iid}] = \text{st}_{1,\ell}$, just like G_7 .

Next we will show that G_8 is identical to G_7 .

In G_7 , $vk := [\mathbf{A}]\mathbf{x} = [\mathbf{A}] \cdot (\mathbf{x}' + \mathbf{a}^\perp \mu) = [\mathbf{A}] \cdot \mathbf{x}'$ and $vk_i := [\mathbf{A}]\mathbf{x}_i = [\mathbf{A}] \cdot (\mathbf{x}'_i + \mathbf{a}^\perp \mu_i) = [\mathbf{A}] \cdot \mathbf{x}'_i$ for $i \in [n]$, which are identical to $vk, \{vk_i\}_{i \in [n]}$ in G_8 .

In G_7 , if (vk, m) is not the ctr^* -th fresh hash query to H_0 , we have $[\mathbf{A}] := [\mathbf{v}^\top \mathbf{A}]$ for some $\mathbf{v} \in \mathbb{Z}_q^{k-1}$ (due to G_5), and

$$[y_\ell] := [\mathbf{A}] \cdot \mathbf{x}_\ell = [\mathbf{v}^\top \mathbf{A}] \cdot \mathbf{x}_\ell = [\mathbf{v}^\top \mathbf{A}] \cdot (\mathbf{x}'_\ell + \mathbf{a}^\perp \mu_\ell) = [\mathbf{A}] \cdot \mathbf{x}'_\ell, \quad (18)$$

which is identical to $[y_\ell]$ in G_8 .

Therefore, G_8 is identical to G_7 and hence

$$\Pr_7[\text{Win}] = \Pr_8[\text{Win}]. \quad (19)$$

Game G_9 . G_9 is the same as G_8 , except for the replies of the second-round signing oracle SIGN_2 and the corrupt oracle CORR .

- For any query $(\ell, \text{iid}, \{\text{pm}_{1,i}\}_{i \in \mathcal{SS}})$ for SIGN_2 , SIGN_2 retrieves $\mathcal{I}[\ell, \text{iid}] = (\mathcal{SS}, m)$, sets $\mathcal{Q}_1[m] := \mathcal{Q}_1[m] \cup \{\ell\}$, combines the first-round message $\{\text{pm}_{1,i}\}_{i \in \mathcal{SS}}$ to get com and $[\mathbf{w}]$, and computes $c := H_1(vk, m, \text{com}, [\mathbf{w}])$, just like G_8 .
Next, SIGN_2 replies the query $(\ell, \text{iid}, \{\text{pm}_{1,i}\}_{i \in \mathcal{SS}})$ in two cases.
Case I: (vk, m) is the ctr^* -th fresh random oracle query to H_0 . SIGN_2 behaves just the same as G_8 .
Case II: (vk, m) is not the ctr^* -th fresh random oracle query to H_0 . Oracle SIGN_2 retrieves $\text{st}_{1,\ell} = ([y_\ell], r'_\ell, \mathbf{r}_\ell)$, computes $\mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{\mathcal{SS},\ell} \cdot \mathbf{x}'_\ell$, sets $\text{state}[\ell, \text{iid}] := \text{st}_{2,\ell} := ([y_\ell], r'_\ell, \mathbf{r}_\ell)$, and returns $\text{pm}_{2,\ell} := ([y_\ell], r'_\ell, \mathbf{z}_\ell)$.
- For any corrupt query ℓ , oracle CORR updates the internal states $\text{state}[\ell, \text{iid}]$ with $\text{iid} \in [\text{num}_\ell]$. For each $\text{iid} \in [\text{num}_\ell]$, CORR retrieves $\mathcal{I}[\ell, \text{iid}] = (\mathcal{SS}, m)$. If (vk, m) is not the ctr^* -th fresh query to H_0 , then it retrieves $\text{state}[\ell, \text{iid}] = ([y_\ell], r'_\ell, \mathbf{r}_\ell)$, computes $\mathbf{r}'_\ell := \mathbf{r}_\ell - c \cdot \lambda_{\mathcal{SS},\ell} \mathbf{a}^\perp \mu_\ell$, and updates $\text{state}[\ell, \text{iid}] := ([y_\ell], r'_\ell, \mathbf{r}'_\ell)$. Otherwise it updates $\text{state}[\ell, \text{iid}]$ as in G_8 . Finally CORR returns signer ℓ 's secret key sk_ℓ and its updated internal states $\{\text{state}[\ell, \text{iid}]\}_{\text{iid} \in [\text{num}_\ell]}$.

The difference between G_9 and G_8 lies in the computation of $\mathbf{z}_\ell$ in Case II by SIGN_2 and the later reveal $\mathbf{r}'_\ell$ of the state $\text{state}[\ell, \text{iid}]$ by CORR .

Recall that in Case II, (vk, m) is not the ctr^* -th fresh random oracle query to H_0 , so $[\mathbf{A}] := H_0(vk, m) := [\mathbf{v}^\top \mathbf{A}]$ for some $\mathbf{v} \in \mathbb{Z}_q^{k-1}$. Also recall that in this case, SIGN_2 computes $\mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{\mathcal{SS},\ell} \cdot \mathbf{x}_\ell$ in G_8 , but it computes $\mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{\mathcal{SS},\ell} \cdot \mathbf{x}'_\ell$ in G_9 .

- Given $\mathbf{r}'_\ell := \mathbf{r}_\ell - c \cdot \lambda_{SS,\ell} \mathbf{a}^\perp \mu_\ell$, we know that $\mathbf{r}'_\ell$ is uniformly distributed due to the uniformity of $\mathbf{r}_\ell$.
- In SIGN_2 , $\mathbf{z}_\ell = \mathbf{r}_\ell + c \cdot \lambda_{SS,\ell} \cdot \mathbf{x}'_\ell = \mathbf{r}'_\ell + c \cdot \lambda_{SS,\ell} \mathbf{a}^\perp \mu_\ell + c \cdot \lambda_{SS,\ell} \cdot \mathbf{x}'_\ell = \mathbf{r}'_\ell + c \cdot \lambda_{SS,\ell} \cdot \mathbf{x}_\ell$. Thus the computation of $\mathbf{z}_\ell$ in $\mathbf{G}_8$ is the same as $\mathbf{G}_9$.
- In the later corruption, CORR reveals $\mathbf{r}'_\ell = \mathbf{r}_\ell - c \cdot \lambda_{SS,\ell} \mathbf{a}^\perp \mu_\ell$ to $\mathcal{A}$, which is consistent to SIGN_2 in $\mathbf{G}_9$. Note that in SIGN_1 , $[\mathbf{w}_\ell] := [\mathbf{A}] \cdot \mathbf{r}_\ell = \begin{bmatrix} \overline{\mathbf{A}} \\ \mathbf{v}^\top \overline{\mathbf{A}} \end{bmatrix} \cdot (\mathbf{r}'_\ell + c \cdot \lambda_{SS,\ell} \mathbf{a}^\perp \mu_\ell) = \begin{bmatrix} \overline{\mathbf{A}} \\ \mathbf{v}^\top \overline{\mathbf{A}} \end{bmatrix} \cdot \mathbf{r}'_\ell = [\mathbf{A}] \cdot \mathbf{r}'_\ell$, so $\mathbf{r}'_\ell$ is also consistent to SIGN_1 .

Therefore, $\mathbf{G}_9$ is identical to $\mathbf{G}_8$ and we obtain

$$\Pr_8[\text{Win}] = \Pr_9[\text{Win}]. \quad (20)$$

Claim 4. $\Pr_9[\text{Win}] \leq \frac{2}{q}$.

Proof. Recall that Win in $\mathbf{G}_9$ implies CorrectY and GoodGuess . Thus it must hold that the ctr^* -th fresh hash query (vk', m) to H_0 satisfies $vk' = vk \wedge m = m^*$.

If GoodGuess occurs in $\mathbf{G}_9$, then $[\mathbf{A}^*] := H_0(vk, m^*) \leftarrow \mathbb{G}^{1 \times k}$. Now we show that μ remains perfectly hidden to $\mathcal{A}$ in this case.

- $pp := (\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{A}])$, $vk := [\overline{\mathbf{A}} \mathbf{x}']$ and $vk_i := [\overline{\mathbf{A}} \mathbf{x}'_i]$ for all $i \in [n]$. Thus $\mathcal{A}$ obtains no information of μ from $pp, vk, \{vk_i\}_{i \in [n]}$.
- Upon a signing query $(\mathcal{SS}, \ell, m)$, oracle SIGN_1 will return $(\text{iid}, \text{pm}_{1,\ell} = (\text{com}_\ell, [\mathbf{w}_\ell]))$. Upon a signing query $(\ell, \text{iid}, \{\text{pm}_{1,i}\}_{i \in \mathcal{SS}})$, oracle SIGN_2 will return $\text{pm}_{2,\ell} = ([y_\ell], r'_\ell, \mathbf{z}_\ell)$. After SIGN_1 's reply, there must exist $(\text{ctr}, vk, m, \cdot) \in \text{List}_0$ since SIGN_1 must have issued a hash query to H_0 . We distinguish two cases:
 - Case $\text{ctr} \neq \text{ctr}^*$.** In this case, we have $[y_\ell] := [\underline{\mathbf{A}}] \cdot \mathbf{x}'_\ell$, $r'_\ell \leftarrow \mathcal{R}$, $\text{com}_\ell \leftarrow \text{Commit}(\text{pp}_{\text{com}}, [y_\ell]; r'_\ell)$, $[\mathbf{w}_\ell] := \begin{bmatrix} \overline{\mathbf{A}} \\ \underline{\mathbf{A}} \end{bmatrix} \cdot \mathbf{r}_\ell$, and $\mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{SS,\ell} \cdot \mathbf{x}'_\ell$. Thus $\mathcal{A}$ obtains no information of μ from $\text{SIGN}_1, \text{SIGN}_2$ in this case.
 - Case $\text{ctr} = \text{ctr}^*$.** In this case, we distinguish SIGN_1 and SIGN_2 :
 - Oracle SIGN_1 computes $(\overline{\text{com}}_\ell, \overline{\text{st}}_\ell) \leftarrow \overline{\text{Commit}}(\text{pp}_{\text{com}})$ and $[\mathbf{w}_\ell] := \begin{bmatrix} \overline{\mathbf{A}} \\ \underline{\mathbf{A}} \end{bmatrix} \cdot \mathbf{r}_\ell$. Thus $\mathcal{A}$ obtains no information of μ from SIGN_1 in this case.
 - Oracle SIGN_2 sets $\mathcal{Q}_1[m^*] := \mathcal{Q}_1[m^*] \cup \{\ell\}$. Suppose that $H_0(vk, m) = [\mathbf{u}_{\text{ctr}^*}^\top]$, SIGN_2 computes $\underline{y}_\ell := \mathbf{u}_{\text{ctr}^*}^\top \cdot (\mathbf{x}'_\ell + \mathbf{a}^\perp \mu_\ell)$, $[\mathbf{w}_\ell] := \begin{bmatrix} \overline{\mathbf{A}} \\ \underline{\mathbf{A}} \end{bmatrix} \cdot \mathbf{r}_\ell$, $\mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{SS,\ell} \cdot (\mathbf{x}'_\ell + \mathbf{a}^\perp \mu_\ell)$, and $\bar{r}_\ell \leftarrow \text{Equiv}(\text{td}_{\text{com}}, \overline{\text{com}}_\ell, \overline{\text{st}}_\ell, \underline{y}_\ell)$. Thus $\mathcal{A}$ learns at most one share μ_ℓ from SIGN_2 in this case.
- Upon a corruption query ℓ , oracle CORR sets $\mathcal{CS} := \mathcal{CS} \cup \{\ell\}$ and returns $(sk_\ell, \{\text{state}[\ell, i]\}_{i \in [\text{num}_\ell]})$, where $sk_\ell = \mathbf{x}'_\ell + \mathbf{a}^\perp \mu_\ell$ and $\text{state}[\ell, i] = ([y_\ell], r'_\ell, \mathbf{r}'_\ell = \mathbf{r}_\ell - c \cdot \lambda_{SS,\ell} \mathbf{a}^\perp \mu_\ell)$ or $\text{state}[\ell, i] = ([y_\ell], \bar{r}_\ell, \mathbf{r}_\ell)$ where $\bar{r}_\ell \leftarrow \text{Equiv}(\text{td}_{\text{com}}, \overline{\text{com}}_\ell, \overline{\text{st}}_\ell, \underline{y}_\ell)$. In this case, $\mathcal{A}$ learns at most one share μ_ℓ from CORR .

$\mathcal{J} \leftarrow \text{TS.Detect}(\mathcal{SS}^*, m^*, \{\text{trx}_i\}_{i \in \mathcal{SS}^*}, \{vk_i\}_{i \in [n]}):$	
1: For $i \in \mathcal{SS}^*$:	11: For $i \in \mathcal{SS}^* \setminus \mathcal{J}_1$:
2: Parse $\text{trx}_i = \{\text{pm}_{1,j}^{(i)}, \text{pm}_{2,j}^{(i)}\}_{j \in \mathcal{SS}^*}$	12: Parse $\{\text{pm}_{1,j}^{(i)} = (\text{com}_j^{(i)}, [\mathbf{w}_j^{(i)}]), \text{pm}_{2,j}^{(i)} = ([y_j^{(i)}], r_j^{(i)}, \mathbf{z}_j^{(i)})\}_{j \in \mathcal{SS}^*}$
3: $\mathcal{J}_1 := \emptyset, \mathcal{J}_2 := \emptyset$	13: Parse $vk_i = [\bar{\mathbf{y}}_i]$
4: //Detect all equivocating signers	14: If $\text{Open}(\text{pp}_{\text{com}}, \text{com}_i^{(i)}, [\underline{y}_i^{(i)}], r_i^{(i)}) \neq 1$:
5: For $i \in \mathcal{SS}^*$ and $R \in [2]$:	15: $\mathcal{J}_2 := \mathcal{J}_2 \cup \{i\}$ // caught i as misbehaving signer
6: If $\exists j_1, j_2 \in \mathcal{SS}^*$, s.t. $(\text{pm}_{R,i}^{(j_1)} \neq \text{pm}_{R,i}^{(j_2)})$:	16: $[\mathbf{y}_i^{(i)}] := \begin{bmatrix} \bar{\mathbf{y}}_i \\ \underline{y}_i^{(i)} \end{bmatrix}, \text{com}^{(i)} := \sum_{j \in \mathcal{SS}} \text{com}_j^{(i)} \cdot \lambda_{\mathcal{SS}^*, j}$
7: $\mathcal{J}_1 := \mathcal{J}_1 \cup \{i\}$ // caught i as equivocating signer	17: $[\mathbf{w}^{(i)}] := \sum_{j \in \mathcal{SS}} [\mathbf{w}_j^{(i)}]$
8: //Detect signers who sent invalid signature shares	18: $c^{(i)} := H_1(vk, m^*, \text{com}^{(i)}, [\mathbf{w}^{(i)}])$
9: $[\underline{\mathbf{A}}] := H_0(vk, m^*)$	19: If $[\mathbf{A}\mathbf{z}_i^{(i)}] \neq [\mathbf{w}^{(i)}] + c^{(i)} \cdot \lambda_{\mathcal{SS}^*, i} \cdot [\mathbf{y}_i^{(i)}]$:
10: $[\mathbf{A}] := \begin{bmatrix} \underline{\mathbf{A}} \\ \mathbf{A} \end{bmatrix}$	20: $\mathcal{J}_2 := \mathcal{J}_2 \cup \{i\}$ // caught i as misbehaving signer
	21: Return $\mathcal{J}_1 \cup \mathcal{J}_2$

Fig. 6. The TS.Detect Algorithm for (t, n) -TS Scheme

If Win occurs, it must hold that $|\mathcal{Q}_1[m^*] \cup \mathcal{CS}| < t$. Therefore $\mathcal{A}$ learns less than t shares of μ in $\mathcal{G}_9$. By the perfect secrecy of the secret sharing scheme the leakage of fewer than t shares $\{\mu_\ell\}$ reveals no information about the underlying secret μ . Therefore, μ remains perfectly hidden from $\mathcal{A}$ in $\mathcal{G}_9$.

If Win occurs, then CorrectY must happen, so $\mathcal{A}$'s forgery $(m^*, \sigma^* = ([y^*], r'^*, \mathbf{z}^*, c^*))$ satisfies

$$[y^*] = [\underline{\mathbf{A}}^*] \cdot \mathbf{x} = [\underline{\mathbf{A}}^*] \cdot (\mathbf{x}' + \mathbf{a}^\perp \mu) = [\underline{\mathbf{A}}^*] \cdot \mathbf{x}' + [\underline{\mathbf{A}}^*] \cdot \mathbf{a}^\perp \mu. \quad (21)$$

Recall that $[\underline{\mathbf{A}}^*] := H_0(vk, m^*) \leftarrow_{\mathcal{S}} \mathbb{G}^{1 \times k}$. We can see that $[\underline{\mathbf{A}}^*]^\top \notin \text{Span}([\underline{\mathbf{A}}]^\top)$ except with probability $1/q$. Thus $[\underline{\mathbf{A}}^*] \cdot \mathbf{a}^\perp \neq \mathbf{0}$ except with probability $1/q$. In this case, the uniformity of μ makes sure that $[\underline{\mathbf{A}}^*] \cdot \mathbf{a}^\perp \cdot \mu$ is uniformly distributed over $\mathbb{G}$. As a result, (21) holds with probability at most $1/q$. Then we have

$$\Pr_9[\text{Win}] \leq \Pr_9[\text{CorrectY}] \leq \frac{2}{q}.$$

□

The proof finishes by combining (1), (2), (3), (13), (14), (15), (16), (17), (19), (20) and Claim 4.

□

4.3 Identifiable Abort of (t, n) -TS

To support identifiable abort, we require an authenticated channel. The authenticated channel can be realized by attaching each round-message $\text{pm}_{R,j}$ with a signature generated by signer j . Therefore, upon receiving each message, one can ensure that $\text{pm}_{R,j}$ is originated from signer j .

To prove the identifiable abort property, we design algorithm TS.Detect in Fig. 6 and prove the following theorem.

Theorem 2. *Suppose that $\text{Share}_{(t,n)}$ is a (t, n) -threshold secret sharing scheme with linear reconstruction and linearity property, HCOM is a homomorphic commitment scheme with special equivocality satisfying correctness and homomorphism properties. Then, we have $\Pr[\text{Exp}_{\text{TS}, \mathcal{A}}^{\text{adp-IA}}(\kappa, t, n) \Rightarrow 1] = 0$ for any (even all powerful) adversary $\mathcal{A}$.*

Proof. Recall that identifiable abort requires that TS.Detect algorithm identify at least one misbehaving signer while never accusing an honest signer.

Suppose that $\mathcal{A}$ outputs $(\text{sid}^*, \ell^*, \{\text{tr}_i\}_{i \in \widetilde{\mathcal{SS}}})$ with $\mathcal{I}[\text{sid}^*] = (\mathcal{SS}^*, m^*)$. If $\ell^* \in \mathcal{SS}^* \setminus \mathcal{CS}$, $(\text{sid}^*, \ell^*) \in \text{failed}$ and $\widetilde{\mathcal{SS}} = \mathcal{SS}^* \cap \mathcal{CS}$ then TS.Detect is invoked to output $\mathcal{J} = (\mathcal{J}_1 \cup \mathcal{J}_2)$. According to the specification of $\mathcal{J} = (\mathcal{J}_1 \cup \mathcal{J}_2) \leftarrow \text{TS.Detect}(\mathcal{SS}^*, \cdot, \cdot, \cdot)$ algorithm, it must hold that $\mathcal{J} \subseteq \mathcal{SS}^*$. To prove Theorem 2, it suffices to show that $(\mathcal{J} \subseteq \mathcal{CS}) \wedge (\mathcal{J} \cap \mathcal{SS}^* \neq \emptyset)$ since $\mathcal{J} \subseteq \mathcal{CS} \wedge \mathcal{J} \cap \mathcal{SS}^* \neq \emptyset = \neg(\mathcal{J} \not\subseteq \mathcal{CS} \vee \mathcal{J} \cap \mathcal{SS}^* = \emptyset)$.

Firstly, we show that $\mathcal{J} \subseteq \mathcal{CS}$. In other words, we show that an honest signer will never be accused by TS.Detect .

Note that an honest signer $\ell \in \mathcal{SS}^* \setminus \mathcal{CS}$ always follows the protocol faithfully, so it always honestly generates round-messages and sends them to all other signers. Therefore, an honest signer never equivocates by sending inconsistent round-message to different parties. Furthermore, with authenticated channel, the adversary $\mathcal{A}$ can neither modify messages sent by honest signers nor impersonate them. Consequently, for every honest signer $\ell \in \mathcal{SS}^* \setminus \mathcal{CS}$ we have $\ell \notin \mathcal{J}_1$.

Next, for every honest signer $\ell \in \mathcal{SS}^* \setminus \mathcal{CS}$, it must have honestly generated $\text{com}_\ell^{(\ell)}, [\mathbf{w}_\ell^{(\ell)}], [\mathbf{y}_\ell^{(\ell)}], \mathbf{z}_\ell^{(\ell)}$, where $c^{(\ell)} := H_1(vk, m, \sum_{i \in \mathcal{SS}^*} \text{com}_i^{(\ell)} \cdot \lambda_{\mathcal{SS}^*, i}, \sum_{i \in \mathcal{SS}^*} [\mathbf{w}_i^{(\ell)}])$ is computed from its view of the first-round messages $\{\text{pm}_{1,j}^{(\ell)} = (\text{com}_j^{(\ell)}, [\mathbf{w}_j^{(\ell)}])\}_{j \in \mathcal{SS}^*}$.

Due to the correctness of HCOM , we have $\text{Open}(\text{pp}_{\text{com}}, \text{com}_\ell^{(\ell)}, [\underline{y}_\ell^{(\ell)}], r_\ell'^{(\ell)}) = 1$. Moreover we know that $[\mathbf{Az}_\ell^{(\ell)}] = [\mathbf{w}_\ell^{(\ell)}] + c \cdot \lambda_{\mathcal{SS}, \ell} \cdot [\mathbf{y}_\ell^{(\ell)}]$, where the analysis is similar to the correctness analysis of (t, n) -TS.

Hence, every honest signer always passes the verification check and will never be added to $\mathcal{J}_2$. Thus, for any honest signer $\ell \in \mathcal{SS}^* \setminus \mathcal{CS}$ we have $\ell \notin \mathcal{J}_2$.

Combined with the fact that $\mathcal{J} \subseteq \mathcal{SS}^*$, thus we conclude that for any honest signer $\ell \in [n] \setminus \mathcal{CS}$ we have $\ell \notin (\mathcal{J}_1 \cup \mathcal{J}_2) = \mathcal{J}$. Therefore we have $\mathcal{J} \subseteq \mathcal{CS}$.

Secondly, we show that $\mathcal{J} \cap \mathcal{SS}^* \neq \emptyset$. In other words, we show that TS.Detect identifies at least one misbehaving signer.

Since $(\text{sid}^*, \ell^*) \in \text{failed}$, $\mathcal{A}$ must have queried $(\ell^*, \text{sid}^*, \{\text{pm}_{2,i}\}_{i \in \mathcal{SS}^*})$ to the COMB oracle, and the resulting combined signature σ^* fails verification, i.e.,

$$\text{TS.Vrfy}(vk, m^*, \sigma^*) = 0. \quad (22)$$

Assume toward contradiction that $\mathcal{J} \cap \mathcal{SS}^* = \emptyset$. Given the fact that $(\mathcal{J}_1 \cup \mathcal{J}_2) = \mathcal{J} \subseteq \mathcal{SS}^*$, then $(\mathcal{J}_1 \cup \mathcal{J}_2) \cap \mathcal{SS}^* = \emptyset$ implies $\mathcal{J}_1 = \mathcal{J}_2 = \emptyset$. We show that, in this case, the resulting signature σ^* must pass verification.

Given $\mathcal{J}_1 = \emptyset$, no signer is identified as equivocating. Hence, every first-round and second-round message is consistently delivered to all signers, i.e., $\forall i_1, i_2 \in \mathcal{SS}^*, \{\text{pm}_{R,j}^{(i_1)} = \text{pm}_{R,j}^{(i_2)}\}_{j \in \mathcal{SS}^*}$. In particular, the combined values $\text{com} = \sum_{i \in \mathcal{SS}} \text{com}_i \cdot \lambda_{\mathcal{SS}, i}, [\mathbf{w}] = \sum_{i \in \mathcal{SS}} [\mathbf{w}_i]$ are identical for all signers, and therefore the hash value $c := H_1(vk, m^*, \text{com}, [\mathbf{w}])$ is also consistent.

Moreover, since $\mathcal{J}_2 = \emptyset$, every signer must pass the local verification inside TS.Detect . Therefore, for every $\ell \in \mathcal{SS}$, it holds that $\text{Open}(\text{pp}_{\text{com}}, \text{com}_\ell, [\underline{y}_\ell], r_\ell') = 1$ which, by the definition of Open , is equivalent to $\text{com}_\ell = \text{Commit}(\text{pp}_{\text{com}}, [\underline{y}_\ell]; r_\ell')$.

By the homomorphism property of HCOM, we obtain

$$\begin{aligned} com &= \sum_{i \in \mathcal{SS}} com_i \cdot \lambda_{\mathcal{SS},i} = \sum_{i \in \mathcal{SS}} \text{Commit}(\text{pp}_{\text{com}}, [\underline{y}_i] r'_i) \cdot \lambda_{\mathcal{SS},i} \\ &= \text{Commit}(\text{pp}_{\text{com}}, \sum_{i \in \mathcal{SS}} [\underline{y}_i] \cdot \lambda_{\mathcal{SS},i}; \sum_{i \in \mathcal{SS}} r'_i \cdot \lambda_{\mathcal{SS},i}) = \text{Commit}(\text{pp}_{\text{com}}, [\underline{y}]; r'). \end{aligned}$$

Furthermore, for every $\ell \in \mathcal{SS}$, it also holds that $[\mathbf{A}\mathbf{z}_\ell] = [\mathbf{w}_\ell] + c \cdot \lambda_{\mathcal{SS},\ell} \cdot [\mathbf{y}_\ell]$. Summing over all $i \in \mathcal{SS}$, we have $\sum_{i \in \mathcal{SS}} [\mathbf{A}\mathbf{z}_i] = \sum_{i \in \mathcal{SS}} ([\mathbf{w}_i] + c \cdot \lambda_{\mathcal{SS},i} \cdot [\mathbf{y}_i])$

$$\Rightarrow [\mathbf{A}] \sum_{i \in \mathcal{SS}} \mathbf{z}_i = \sum_{i \in \mathcal{SS}} [\mathbf{w}_i] + c \cdot \sum_{i \in \mathcal{SS}} \lambda_{\mathcal{SS},i} \cdot [\mathbf{y}_i] \Rightarrow [\mathbf{A}]\mathbf{z} = [\mathbf{w}] + c \cdot [\mathbf{y}].$$

Equivalently, we have $[\mathbf{w}] = [\mathbf{A}]\mathbf{z} - c \cdot [\mathbf{y}]$.

Therefore, in the verification algorithm $\text{TS.Vrfy}(vk, m^*, \sigma^*)$, the check of $c = H_1(vk, m^*, com, [\mathbf{w}])$ always succeeds. As a result, $\text{TS.Vrfy}(vk, m^*, \sigma^*) = 1$ contradicts to (22).

In conclusion, we have $\mathcal{J} \subseteq \mathcal{CS}$ and $\mathcal{J} \cap \mathcal{SS}^* \neq \emptyset$, so $\text{Exp}_{\text{TS}, \mathcal{A}}^{\text{adp-IA}}(\kappa, t, n)$ always outputs 0. $\square$

5 Instantiation of (t, n) -TS

To obtain a concrete (t, n) -TS scheme, we instantiate the underlying components as follows.

- The group $(\mathbb{G}, g, q) \leftarrow \text{GGen}(1^\kappa)$ is instantiated as an arbitrary cyclic group of prime order q with generator g .
- $\text{Share}_{(t,n)}$ is instantiated by Shamir (t, n) -threshold secret sharing scheme over the secret space $\mathbb{S} = \mathbb{Z}_q^k$. As shown in Appendix B, Shamir scheme satisfies linear reconstruction, perfect secrecy, and linearity.
- HCOM is instantiated by the commitment scheme HCOM_{DL} in Figure 7. Briefly, HCOM_{DL} uses Pedersen commitment to commit a random r_1 , and then uses $[r_1]$ to mask the committed group element $[m]$. We prove that HCOM_{DL} has correctness, binding from the DL assumption, special equivocality, and homomorphism in Appendix D. Its message space, randomness space, and commitment space are set as $\mathcal{M}' := \mathbb{G}$, $\mathcal{R} := \mathbb{Z}_q^2$, and $\mathcal{C} := \mathbb{G}^2$, respectively.

Given $\sigma = ([\underline{y}], r', \mathbf{z}, c) \in \mathbb{G} \times \mathcal{R} \times \mathbb{Z}_q^k \times \mathbb{Z}_q$, $\text{pm}_{1,\ell} = (com_\ell, [\mathbf{w}_\ell]) \in \mathcal{C} \times \mathbb{G}^k$, $\text{pm}_{2,\ell} = ([\underline{y}_\ell], r'_\ell, \mathbf{z}_\ell) \in \mathbb{G} \times \mathcal{R} \times \mathbb{Z}_q^k$ and $vk = [\underline{\mathbf{y}}] \in \mathbb{G}^{k-1}$, the above instantiations result in a specific (t, n) - TS_{MDDH} scheme based on the MDDH assumption with signature size $|\sigma| = (k+3)\mathbb{Z}_q + \mathbb{G}$, communication complexity per signer $|\text{pm}_{1,\ell}| + |\text{pm}_{2,\ell}| = ((k+2)\mathbb{G}) + (\mathbb{G} + (k+2)\mathbb{Z}_q) = (k+3)\mathbb{G} + (k+2)\mathbb{Z}_q$, and $|vk| = (k-1)\mathbb{G}$.

When $k = 2$, we obtain a specific (t, n) - TS_{DDH} scheme based on the DDH assumption with signature size $|\sigma| = 5\mathbb{Z}_q + \mathbb{G}$, communication complexity per

$(\text{pp}_{\text{com}}, \text{td}_{\text{com}}) \leftarrow \text{CSetup}(1^\kappa) :$ 1: $(\mathbb{G}, g, q) \leftarrow \text{GGen}(1^\kappa)$ 2: $\alpha \leftarrow \mathbb{Z}_q$ 3: $\text{pp}_{\text{com}} := ([1], [\alpha])$ 4: $\text{td}_{\text{com}} := (\alpha, [1], [\alpha])$ 5: $\mathcal{M}' := \mathbb{G}$ 6: $\mathcal{R} := \mathbb{Z}_q^2$ 7: $\mathcal{C} := \mathbb{G}^2$ 8: Return $(\text{pp}_{\text{com}}, \text{td}_{\text{com}})$ $\text{com} \leftarrow \text{Commit}(\text{pp}_{\text{com}}, [m]; r) :$ 9: Parse $\text{pp}_{\text{com}} := ([1], [\alpha])$ 10: Parse $r = (r_1, r_2) \in \mathbb{Z}_q^2$ 11: $\text{com}_1 := [m] + [r_1]$ 12: $\text{com}_2 := [r_1] + r_2 \cdot [\alpha]$ 13: Return $\text{com} := (\text{com}_1, \text{com}_2)$	$0/1 \leftarrow \text{Open}(\text{pp}_{\text{com}}, \text{com}, [m], r) :$ 14: Parse $\text{pp}_{\text{com}} := ([1], [\alpha])$ 15: Parse $r = (r_1, r_2) \in \mathbb{Z}_q^2$ 16: Parse $\text{com} := (\text{com}_1, \text{com}_2)$ 17: If $\text{com}_1 = [m] + [r_1] \wedge \text{com}_2 = [r_1] + r_2 \cdot [\alpha]$: 18: Return 1 19: Else: Return 0 $(\overline{\text{com}}, \overline{\text{st}}) \leftarrow \overline{\text{Commit}}(\text{pp}_{\text{com}}) :$ 20: Parse $\text{pp}_{\text{com}} := ([1], [\alpha])$ 21: $\beta, \gamma \leftarrow \mathbb{Z}_q$ 22: $\overline{\text{com}} := ([\beta], [\gamma])$ 23: $\overline{\text{st}} := (\beta, \gamma)$ 24: Return $(\overline{\text{com}}, \overline{\text{st}})$ $\bar{r} \leftarrow \text{Equiv}(\text{td}_{\text{com}}, \overline{\text{com}}, \overline{\text{st}}, m') :$ 25: Parse $\text{td}_{\text{com}} = (\alpha, [1], [\alpha])$ 26: Parse $\overline{\text{st}} = (\beta, \gamma)$ 27: $\bar{r}_1 := \beta - m' \pmod{q}$ 28: $\bar{r}_2 := \alpha^{-1}(\gamma - \beta + m') \pmod{q}$ 29: Return $\bar{r} := (\bar{r}_1, \bar{r}_2)$
---	--

Fig. 7. Our Instantiation of the HCOM_{DL} Scheme.

signer $|\text{pm}_{1,\ell}| + |\text{pm}_{2,\ell}| = 5\mathbb{G} + 4\mathbb{Z}_q$, and $|vk| = \mathbb{G}$. We refer Table 1 for the comparison of existing works with adaptive security (pairing-free, without AGM, without secure state erasure). We stress that our (t, n) -TS is the only scheme enjoying round efficiency (two rounds), identifiable abort and TS-UF-1 security.

Acknowledgements. We would like to thank the reviewers for their valuable comments. This work was supported by National Natural Science Foundation of China (Grant Nos. 62632012, 62572295, 625B2114), National Cryptologic Science Fund of China (Grant No. 2025NCSF01009), and National Cyber Security-National Science and Technology Major Project (Grant No. 2026ZD1500900).

References

- [1] Bacho, R., Chen, Y., Loss, J., Tessaro, S., Zhu, C.: Adaptively secure partially non-interactive threshold schnorr signatures in the AGM. IACR Cryptol. ePrint Arch. 2025, 1953 (2025), <https://eprint.iacr.org/2025/1953>
- [2] Bacho, R., Das, S., Loss, J., Ren, L.: Adaptively secure three-round threshold schnorr signatures from DDH. In: Kalai, Y.T., Kamara, S.F. (eds.) Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part VI. pp. 390–422. Lecture Notes in Computer Science, Springer (2025), https://doi.org/10.1007/978-3-032-01887-8_13
- [3] Bacho, R., Das, S., Loss, J., Ren, L.: Glacius: Threshold schnorr signatures from DDH with full adaptive security. In: Fehr, S., Fouque, P. (eds.) Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part II. pp. 304–334. Lecture Notes in Computer Science, Springer (2025), https://doi.org/10.1007/978-3-031-91124-8_11
- [4] Bacho, R., Loss, J.: On the adaptive security of the threshold BLS signature scheme. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS

- 2022, Los Angeles, CA, USA, November 7-11, 2022. pp. 193–207. ACM (2022), <https://doi.org/10.1145/3548606.3560656>
- [5] Bacho, R., Loss, J., Tessaro, S., Wagner, B., Zhu, C.: Twinkle: Threshold signatures from DDH with full adaptive security. In: Joye, M., Leander, G. (eds.) *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I. pp. 429–459. *Lecture Notes in Computer Science*, Springer (2024), https://doi.org/10.1007/978-3-031-58716-0_15
 - [6] Bacho, R., Wagner, B.: Tightly secure threshold signatures over pairing-free groups. *IACR Commun. Cryptol.* 2(4), 8 (2026), <https://doi.org/10.62056/ay73z10k>
 - [7] Bellare, M., Crites, E.C., Komlo, C., Maller, M., Tessaro, S., Zhu, C.: Better than advertised security for non-interactive threshold signatures. In: Dodis, Y., Shrimpton, T. (eds.) *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference*, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV. pp. 517–550. *Lecture Notes in Computer Science*, Springer (2022), https://doi.org/10.1007/978-3-031-15985-5_18
 - [8] Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006. pp. 390–399. ACM (2006), <https://doi.org/10.1145/1180405.1180453>
 - [9] Bobolz, J., Crites, E.C., Kohlweiss, M., Takahashi, A.: Uc4free! existing threshold signatures are UC secure. In: Daemen, J., Thomé, E. (eds.) *Advances in Cryptology - EUROCRYPT 2026 - 45th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Rome, Italy, May 10-14, 2026, Proceedings, Part I. *Lecture Notes in Computer Science*, vol. 16541, pp. 63–93. Springer (2026), https://doi.org/10.1007/978-3-032-25291-3_3
 - [10] Chen, Y.: Dazzle: Improved adaptive threshold signatures from DDH. In: Jager, T., Pan, J. (eds.) *Public-Key Cryptography - PKC 2025 - 28th IACR International Conference on Practice and Theory of Public-Key Cryptography*, Røros, Norway, May 12-15, 2025, Proceedings, Part III. pp. 233–261. *Lecture Notes in Computer Science*, Springer (2025), https://doi.org/10.1007/978-3-031-91826-1_8
 - [11] Chen, Y.: Round-efficient adaptively secure threshold signatures with rewinding. *IACR Commun. Cryptol.* 2(2), 16 (2025), <https://doi.org/10.62056/ah893z10k>
 - [12] Chen, Y.: Partially non-interactive two-round threshold and multi-signatures with tighter and adaptive security. *IACR Cryptol. ePrint Arch.* 2026, 373 (2026), <https://eprint.iacr.org/2026/373>
 - [13] Crites, E.C., Katz, J., Komlo, C., Tessaro, S., Zhu, C.: On the adaptive security of FROST. In: Kalai, Y.T., Kamara, S.F. (eds.) *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference*, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part VI. pp. 480–511. *Lecture Notes in Computer Science*, Springer (2025), https://doi.org/10.1007/978-3-032-01887-8_16
 - [14] Das, S., Ren, L.: Adaptively secure BLS threshold signatures from DDH and co-dh. In: Reyzin, L., Stebila, D. (eds.) *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference*, Santa Barbara, CA, USA, Au-

- gust 18-22, 2024, Proceedings, Part VII. pp. 251–284. Lecture Notes in Computer Science, Springer (2024), https://doi.org/10.1007/978-3-031-68394-7_9
- [15] Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.L.: An algebraic framework for diffie-hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference*, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II. pp. 129–147. Lecture Notes in Computer Science, Springer (2013), https://doi.org/10.1007/978-3-642-40084-1_8
 - [16] Gerhart, P., Calsi, D.L., Russo, L., Schröder, D.: Fully-adaptive two-round threshold schnorr signatures from DDH. *IACR Cryptol. ePrint Arch.* 2025, 1478 (2025), <https://eprint.iacr.org/2025/1478>
 - [17] Jiang, K., Tessaro, S., Wee, H., Zhu, C.: Tweed: Adaptively secure lattice-based two-round threshold signatures. *Cryptology ePrint Archive*, Paper 2026/417 (2026), <https://eprint.iacr.org/2026/417>
 - [18] Katsumata, S., Reichle, M., Takemure, K.: Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding. In: Reyzin, L., Stebila, D. (eds.) *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference*, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VII. pp. 459–491. Lecture Notes in Computer Science, Springer (2024), https://doi.org/10.1007/978-3-031-68394-7_15
 - [19] Mitrokotsa, A., Mukherjee, S., Sedaghat, M., Slamanig, D., Tomy, J.: Threshold structure-preserving signatures: Strong and adaptive security under standard assumptions. In: Tang, Q., Teague, V. (eds.) *Public-Key Cryptography - PKC 2024 - 27th IACR International Conference on Practice and Theory of Public-Key Cryptography*, Sydney, NSW, Australia, April 15-17, 2024, Proceedings, Part I. pp. 163–195. Lecture Notes in Computer Science, Springer (2024), https://doi.org/10.1007/978-3-031-57718-5_6
 - [20] Morillo, P., Ràfols, C., Villar, J.L.: The kernel matrix diffie-hellman assumption. In: Cheon, J.H., Takagi, T. (eds.) *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security*, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. pp. 729–758. Lecture Notes in Computer Science (2016), https://doi.org/10.1007/978-3-662-53887-6_27
 - [21] Niot, G., Reichle, M., Takemure, K.: Adaptively-secure three-round threshold schnorr from DL. *IACR Cryptol. ePrint Arch.* 2025, 1941 (2025), <https://eprint.iacr.org/2025/1941>
 - [22] Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U.M. (ed.) *Advances in Cryptology - EUROCRYPT '96*, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding. pp. 387–398. Lecture Notes in Computer Science, Springer (1996), https://doi.org/10.1007/3-540-68339-9_33
 - [23] Shamir, A.: How to share a secret. *Commun. ACM* 22(11), 612–613 (1979), <https://doi.org/10.1145/359168.359176>

Table of Contents

1	Introduction	1
1.1	Technical Overview	3
2	Preliminaries	8
2.1	Secret Sharing Scheme	9
2.2	Homomorphic Commitment with Special Equivocality	9
3	Definition of Threshold Signature Scheme	10
3.1	Syntax of Two-Round (t, n) -TS	10
3.2	Adaptive TS-UF-1 Security of Two-Round (t, n) -TS	12
3.3	Identifiable Abort of Two-Round (t, n) -TS	12
4	Construction of Threshold Signatures	13
4.1	Construction of (t, n) -TS	13
4.2	Adaptive TS-UF-1 Security of (t, n) -TS	15
4.3	Identifiable Abort of (t, n) -TS	26
5	Instantiation of (t, n) -TS	28
A	Assumptions and Generalized Forking Lemma	33
A.1	Assumptions	33
A.2	Generalized Forking Lemma	34
B	Shamir Secret Sharing Scheme	34
C	Proof of (9)	35
D	Homomorphic Commitment HCOM_{DL} with Special Equivocality	36
E	Construction of Threshold Signatures with TS-UF-0.5 Security	38
E.1	Adaptive TS-UF-0.5 Security of Two-Round (t, n) -TS	38
E.2	Construction of (t, n) -TS'	38
E.3	Adaptive TS-UF-0.5 Security of (t, n) -TS'	40

A Assumptions and Generalized Forking Lemma

A.1 Assumptions

Definition 7 ($\mathcal{U}_{n,m}$ -MDDH Assumption). *The $\mathcal{U}_{n,m}$ -MDDH assumption holds over group $\mathbb{G}$, if for any PPT adversary $\mathcal{A}$, it holds that $\text{Adv}_{\mathbb{G},\mathcal{A}}^{\mathcal{U}_{n,m}\text{-MDDH}}(\kappa) := |\Pr[\mathcal{A}([\mathbf{B}], [\mathbf{B}\mathbf{v}]) = 1] - \Pr[\mathcal{A}([\mathbf{B}], [\mathbf{u}]) = 1]| \leq \text{negl}(\kappa)$, where the probability is over $\mathbf{B} \leftarrow_{\$} \mathcal{U}_{n,m}$, $\mathbf{v} \leftarrow_{\$} \mathbb{Z}_q^m$ and $\mathbf{u} \leftarrow_{\$} \mathbb{Z}_q^n$.*

Several relations among MDDH assumptions parameterized by different matrix distributions were established in [15]. When $n = 2, m = 1$, $\mathcal{U}_{2,1}$ -MDDH assumption becomes the DDH assumption.

For $Q \geq 1$, we define Q -fold $\mathcal{U}_{n,m}$ -MDDH as below.

Definition 8 (Q -fold $\mathcal{U}_{n,m}$ -MDDH Assumption). *The Q -fold $\mathcal{U}_{n,m}$ -MDDH assumption holds over group $\mathbb{G}$, if for any PPT adversary $\mathcal{A}$, it holds that $\text{Adv}_{\mathbb{G},\mathcal{A}}^{Q\text{-}\mathcal{U}_{n,m}\text{-MDDH}}(\kappa) := |\Pr[\mathcal{A}([\mathbf{B}], [\mathbf{B}\mathbf{V}]) = 1] - \Pr[\mathcal{A}([\mathbf{B}], [\mathbf{U}]) = 1]| \leq \text{negl}(\kappa)$, where the probability is over $\mathbf{B} \leftarrow_{\$} \mathcal{U}_{n,m}$, $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_Q) \leftarrow_{\$} \mathbb{Z}_q^{m \times Q}$ and $\mathbf{U} \leftarrow_{\$} \mathbb{Z}_q^{n \times Q}$.*

$\mathcal{U}_{n,m}$ -MDDH problem is random self-reducible [15], namely Q -fold and (1-fold) $\mathcal{U}_{n,m}$ -MDDH problems can be tightly reduced to each other.

Lemma 1 (Random Self-Reducibility of MDDH). *For any adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$, such that $T(\mathcal{B}) \approx T(\mathcal{A}) + Q \cdot \text{poly}(\kappa)$ with $\text{poly}(\kappa)$ independent of $T(\mathcal{A})$, and $\text{Adv}_{\mathbb{G},\mathcal{A}}^{Q\text{-}\mathcal{U}_{n,m}\text{-MDDH}}(\kappa) \leq \text{Adv}_{\mathbb{G},\mathcal{B}}^{\mathcal{U}_{n,m}\text{-MDDH}}(\kappa) + 1/(q-1) \leq \text{Adv}_{\mathbb{G},\mathcal{B}}^{\mathcal{U}_{n,m}\text{-MDDH}}(\kappa) + 2/q$.*

We also define the $\mathcal{U}_{n,m}$ -Kernel Matrix DH ($\mathcal{U}_{n,m}$ -KerMDH) assumption according to [20] which is a natural search variant of the $\mathcal{U}_{n,m}$ -MDDH assumption.

Definition 9 ($\mathcal{U}_{n,m}$ -KerMDH Assumption). *The $\mathcal{U}_{n,m}$ -KerMDH assumption holds over group $\mathbb{G}$, if for any PPT adversary $\mathcal{A}$, it holds that*

$$\text{Adv}_{\mathbb{G},\mathcal{A}}^{\mathcal{U}_{n,m}\text{-KerMDH}}(\kappa) := \Pr[\mathbf{v} \in \mathbb{Z}^n \leftarrow_{\$} \mathcal{A}([\mathbf{B}]) : \mathbf{v}^\top [\mathbf{B}] = [\mathbf{0}] \wedge \mathbf{v} \neq \mathbf{0}] \leq \text{negl}(\kappa),$$

where the probability is over $\mathbf{B} \leftarrow_{\$} \mathcal{U}_{n,m}$.

The following lemma shows that the $\mathcal{U}_{n,m}$ -KerMDH assumption is implied by the $\mathcal{U}_{n,m}$ -MDDH assumption, since one can use a non-zero $\mathbf{v}$ satisfying $\mathbf{v}^\top [\mathbf{B}] = [\mathbf{0}]$ to test membership in $\text{Span}([\mathbf{B}])$.

Lemma 2 ($\mathcal{U}_{n,m}$ -MDDH $\Rightarrow$ $\mathcal{U}_{n,m}$ -KerMDH [20]). *For any PPT adversary $\mathcal{A}$, there exists a PPT algorithm $\mathcal{B}$ s.t.*

$$\text{Adv}_{\mathbb{G},\mathcal{A}}^{\mathcal{U}_{n,m}\text{-KerMDH}}(\kappa) \leq \text{Adv}_{\mathbb{G},\mathcal{B}}^{\mathcal{U}_{n,m}\text{-MDDH}}(\kappa) + 1/(q-1) \leq \text{Adv}_{\mathbb{G},\mathcal{B}}^{\mathcal{U}_{n,m}\text{-MDDH}}(\kappa) + 2/q.$$

Algorithm Fork^A(x):
1: Choose the random coins ρ for $\mathcal{A}$
2: $\{h_1, \dots, h_Q\} \leftarrow_{\$} \mathcal{H}^Q$
3: $(\text{idx}, \text{aux}) \leftarrow \mathcal{A}(x, h_1, \dots, h_Q; \rho)$
4: If $\text{idx} = 0$: return $\perp$
5: $\{h'_{\text{idx}}, \dots, h'_Q\} \leftarrow_{\$} \mathcal{H}^{Q-\text{idx}+1}$
6: $(\text{idx}', \text{aux}') \leftarrow \mathcal{A}(x, h_1, \dots, h_{\text{idx}-1}, h'_{\text{idx}}, \dots, h'_Q; \rho)$
7: If $\text{idx} \neq \text{idx}'$: return $\perp$
8: Return $(\text{idx}, \text{aux}, \text{aux}')$

Fig. 8. The generalized forking algorithm Fork.

A.2 Generalized Forking Lemma

We recall the generalized forking lemma [22] [8].

Lemma 3. *Let $Q \geq 1$ be an integer, and $\mathcal{H}$ be a set. Let $\mathcal{A}$ be a randomized algorithm that on input $x, h_1, h_2, \dots, h_Q$ outputs a pair (idx, aux) , where $\text{idx} \in [0, Q]$ and aux is a side output. Let IG be a randomized algorithm that generates x . The accepting probability of $\mathcal{A}$ is defined as:*

$$\text{acc} := \Pr_{\substack{x \leftarrow \text{IG} \\ h_1, h_2, \dots, h_Q \leftarrow_{\$} \mathcal{H}}} [(\text{idx}, \text{aux}) \leftarrow \mathcal{A}(x, h_1, \dots, h_Q) : \text{idx} \neq 0].$$

Consider algorithm Fork^A described in Fig. 8. The accepting probability of Fork^A is defined as:

$$\text{frk} := \Pr_{x \leftarrow \text{IG}} [v \leftarrow \text{Fork}^{\mathcal{A}}(x) : v \neq \perp].$$

Then, we have:

$$\text{frk} \geq \text{acc} \left(\frac{\text{acc}}{Q} - \frac{1}{|\mathcal{H}|} \right) \implies \text{acc} \leq \frac{Q}{|\mathcal{H}|} + \sqrt{Q \cdot \text{frk}}.$$

B Shamir Secret Sharing Scheme

A standard instantiation of (t, n) -threshold secret sharing is Shamir secret sharing [23] over $\mathbb{S} := \mathbb{Z}_q$. To share a secret $s \in \mathbb{Z}_q$, the sharing algorithm $\text{Share}_{(t,n)}(s)$ samples a random polynomial

$$f(X) = s + a_1X + \dots + a_{t-1}X^{t-1} \in \mathbb{Z}_q[X],$$

where $(a_1, \dots, a_{t-1}) \leftarrow_{\$} \mathbb{Z}_q^{t-1}$. It then outputs $(s_1, \dots, s_n) = (f(1), \dots, f(n))$.

For every subset $\mathcal{SS} \subseteq [n]$ with $|\mathcal{SS}| \geq t$, the reconstruction coefficients are the Lagrange coefficients

$$\lambda_{\mathcal{SS},i} := \prod_{\substack{j \in \mathcal{SS} \\ j \neq i}} \frac{j}{j-i} \in \mathbb{Z}_q.$$

They satisfy

$$s = f(0) = \sum_{i \in \mathcal{SS}} \lambda_{\mathcal{SS},i} f(i) = \sum_{i \in \mathcal{SS}} \lambda_{\mathcal{SS},i} s_i,$$

which gives linear reconstruction.

Perfect secrecy follows from the fact that any set of fewer than t shares is consistent with every possible secret $s \in \mathbb{Z}_q$, with the same number of degree- $(t-1)$ polynomials. Hence, such shares reveal no information about the secret.

Finally, the linearity of $\text{Share}_{(t,n)}$ follows from the linearity of polynomials: if $(s'_1, \dots, s'_n) \leftarrow \text{Share}_{(t,n)}(a)$ is generated by f_a and $(s''_1, \dots, s''_n) \leftarrow \text{Share}_{(t,n)}(b)$ is generated by f_b , then $f_a + f_b$ is a random degree- $(t-1)$ polynomial with constant term $a + b$. Therefore, $(s'_1 + s''_1, \dots, s'_n + s''_n) \equiv \text{Share}_{(t,n)}(a + b)$.

The vector-space extension to $\mathbb{Z}_q^k$ is obtained by applying the above procedure independently to each coordinate.

C Proof of (9)

To prove $\Pr[[y_0^*] \neq [y_1^*] \wedge \text{idx} \neq 0 \wedge \text{idx}' = \text{idx}] \leq \text{Adv}_{\text{HE-COM}, \mathcal{B}_2}^{\text{binding}}(\kappa)$, we construct a reduction algorithm $\mathcal{B}_2$. Given the public parameter pp_{com} of HCOM, $\mathcal{B}_2$ aims to output two distinct valid openings for a same commitment. To this end, $\mathcal{B}_2$ simulates $\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}])$, as shown in Fig. 9.

More specifically, $\mathcal{B}_2$ parses $(\mathbb{G}, g, q)$ from pp_{com} , samples $[\mathbf{B}] \leftarrow_{\$} \mathbb{G}^{k \times k-1}$, $\mathbf{x} \leftarrow_{\$} \mathbb{Z}_q^k$, $h_1, \dots, h_{Q_1} \leftarrow_{\$} \mathbb{Z}_q$ and randomness ρ . Then it invokes $\text{Exp}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{Q_1}; \rho)$ to obtain $(\text{idx}, (m_0^*, \sigma_0^*))$.

- If $\text{idx} = 0$, then $\mathcal{B}_2$ returns $\perp$.
- If $\text{idx} \neq 0$, then $\mathcal{B}_2$ samples fresh values $h'_{\text{idx}}, \dots, h'_{Q_1} \leftarrow \mathbb{Z}_q$, and re-invokes $(\text{idx}', (m_1^*, \sigma_1^*)) \leftarrow \text{Exp}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{\text{idx}-1}, h'_{\text{idx}}, \dots, h'_{Q_1}; \rho)$.
 - If $\text{idx} \neq \text{idx}'$, then $\mathcal{B}_2$ returns $\perp$.
 - If $\text{idx} = \text{idx}'$, let $\sigma_0^* = ([y_0^*], r_0'^*, \mathbf{z}_0^*, c_0^*)$ and $\sigma_1^* = ([y_1^*], r_1'^*, \mathbf{z}_1^*, c_1^*)$.
 - * If $[y_0^*] = [y_1^*]$, $\mathcal{B}_2$ outputs $\perp$.
 - * If $[y_0^*] \neq [y_1^*]$, $\mathcal{B}_2$ computes $\text{com}^* := \text{Commit}(\text{pp}_{\text{com}}, [y_0^*]; r_0'^*)$, and outputs $(\text{com}^*, [y_0^*], r_0'^*, [y_1^*], r_1'^*)$.

Note that $\mathcal{B}_2$ parses $(\mathbb{G}, g, q)$ from pp_{com} and samples $[\mathbf{B}] \leftarrow_{\$} \mathbb{G}^{k \times k-1}$ itself in the simulation. With the help of pp_{com} , $\mathcal{B}_2$ perfectly simulates $\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}])$.

Algorithm $\mathcal{B}_2(\text{pp}_{\text{com}})$:

- 1: Parse $(\mathbb{G}, g, q)$ from pp_{com} //With $\mathcal{M}' = \mathbb{G}$
- 2: $[\mathbf{B}] \leftarrow_{\$} \mathbb{G}^{k \times (k-1)}$
- 3: $\mathbf{x} \leftarrow_{\$} \mathbb{Z}_q^k$
//Begin Fork^{Exp} $(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x})$
- 4: Choose the random coins ρ for Exp
- 5: $\{h_1, \dots, h_{Q_1}\} \leftarrow_{\$} \mathbb{Z}_q^{Q_1}$
- 6: $(\text{idx}, (m_0^*, \sigma_0^*)) \leftarrow \text{Exp}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{Q_1}; \rho)$
- 7: If $\text{idx} = 0$: return $\perp$
- 8: $\{h'_{\text{idx}}, \dots, h'_{Q_1}\} \leftarrow_{\$} \mathbb{Z}_q^{Q_1 - \text{idx} + 1}$
- 9: $(\text{idx}', (m_1^*, \sigma_1^*)) \leftarrow \text{Exp}(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{\text{idx}-1}, h'_{\text{idx}}, \dots, h'_{Q_1}; \rho)$
- 10: If $\text{idx} \neq \text{idx}'$: return $\perp$
//End Fork^{Exp} $(\mathbb{G}, g, q, \text{pp}_{\text{com}}, [\mathbf{B}], \mathbf{x})$
- 11: Parse $\sigma_0^* = ([y_0^*], r_0'^*, \mathbf{z}_0^*, c_0^*)$, $\sigma_1^* = ([y_1^*], r_1'^*, \mathbf{z}_1^*, c_1^*)$
- 12: If $[y_0^*] = [y_1^*]$: return $\perp$
- 13: $\text{com}^* := \text{Commit}(\text{pp}_{\text{com}}, [y_0^*]; r_0'^*)$
- 14: Return $(\text{com}^*, [y_0^*]; r_0'^*, [y_1^*]; r_1'^*)$

Fig. 9. Description of Algorithm $\mathcal{B}_2$.

If $\text{idx} = \text{idx}' \neq 0$, then the two invocations of Exp must have used the same random coins ρ . Hence, the two executions of Exp are identical until the reply to the idx -th fresh query to H_1 . In particular, the idx -th query $(vk, m^*, \text{com}^*, [\mathbf{w}^*])$ in the two executions must be identical. Moreover, by the definition of Exp (see Fig. 4), the final forgery output in each execution is exactly associated with this idx -th query to H_1 . Consequently, (8) holds, namely,

$$\begin{aligned} m_0^* &= m_1^* = m^*, \quad [\mathbf{A}_0^*] = [\mathbf{A}_1^*] = [\mathbf{A}^*] = H_0(vk, m^*), \\ \text{com}_0^* &= \text{com}_1^* = \text{com}^*, \quad [\mathbf{w}_0^*] = [\mathbf{w}_1^*] = [\mathbf{w}^*], \end{aligned}$$

where $\text{com}_0^* := \text{Commit}(\text{pp}_{\text{com}}, [y_0^*]; r_0'^*)$ and $\text{com}_1^* := \text{Commit}(\text{pp}_{\text{com}}, [y_1^*]; r_1'^*)$.

Since $\text{com}_0^* = \text{com}_1^* = \text{com}^*$, we have $\text{Open}(\text{pp}_{\text{com}}, \text{com}^*, [y_0^*], r_0'^*) = 1$ and $\text{Open}(\text{pp}_{\text{com}}, \text{com}^*, [y_1^*], r_1'^*) = 1$ by the correctness of HCOM. If $[y_0^*] \neq [y_1^*]$, then the tuple $(\text{com}^*, [y_0^*], r_0'^*, [y_1^*], r_1'^*)$ provides two distinct valid openings $([y_0^*], r_0'^*)$ and $([y_1^*], r_1'^*)$ for the same commitment com^* , thereby breaking the binding property of HCOM. Therefore, $\mathcal{B}_2$ wins as long as $[y_0^*] \neq [y_1^*] \wedge \text{idx} \neq 0 \wedge \text{idx}' = \text{idx}$ happens, leading to

$$\Pr[[y_0^*] \neq [y_1^*] \wedge \text{idx} \neq 0 \wedge \text{idx}' = \text{idx}] \leq \text{Adv}_{\text{HE-COM}, \mathcal{B}_2}^{\text{binding}}(\kappa).$$

D Homomorphic Commitment HCOM_{DL} with Special Equivocality

In this appendix, we will show that our instantiation of the HCOM_{DL} scheme (see Fig. 7) satisfies correctness, binding, special equivocality and homomorphism.

Correctness. The correctness of this commitment scheme follows directly from the fact that **Open** checks whether $com = \text{Commit}(\text{pp}_{\text{com}} = ([1], [\alpha]), [m]; r)$ holds.

Binding. The binding property of commitment scheme is based on the discrete logarithm (DL) assumption in $\mathbb{G}$, a cyclic group of order q . More precisely, suppose there exists a PPT adversary $\mathcal{A}$ that breaks binding. We construct a reduction algorithm $\mathcal{B}$ that, given a DL challenge $([1], [\alpha])$, where $[1]$ is a generator of $\mathbb{G}$, computes the discrete logarithm α .

$\mathcal{B}$ sends $\text{pp}_{\text{com}} := ([1], [\alpha])$ to $\mathcal{A}$, and then $\mathcal{A}$ outputs $(com, [m^0], r^0, [m^1], r^1)$ where $com = (com_1, com_2)$ and $r^b = (r_1^b, r_2^b) \in \mathbb{Z}_q^2$ for $b \in \{0, 1\}$. If $\mathcal{A}$ wins, then we have $[m^0] \neq [m^1]$ and $\text{Open}(\text{pp}_{\text{com}}, com, [m^0], r^0) = 1 \wedge \text{Open}(\text{pp}_{\text{com}}, com, [m^1], r^1) = 1$. By the definition of **Open**, we have

$$\begin{cases} com_1 = [m^0] + [r_1^0] = [m^1] + [r_1^1], \\ com_2 = [r_1^0] + r_2^0[\alpha] = [r_1^1] + r_2^1[\alpha]. \end{cases}$$

Since $[m^0] \neq [m^1]$, the first equality implies $[r_1^0] \neq [r_1^1]$. Hence, from the second equality, we must have $r_2^0 \neq r_2^1$. Writing the second equality in the exponent gives $r_1^0 + r_2^0\alpha = r_1^1 + r_2^1\alpha \pmod{q}$.

Finally, since $r_2^0 - r_2^1 \neq 0$, $\mathcal{B}$ returns $(r_1^1 - r_1^0)(r_2^0 - r_2^1)^{-1} \pmod{q}$ and successfully solves the discrete logarithm problem in $\mathbb{G}$.

Hence, assuming that the DL assumption holds in $\mathbb{G}$, the binding advantage $\text{Adv}_{\text{HE-COM}, \mathcal{A}}^{\text{binding}}(\kappa)$ of any PPT adversary $\mathcal{A}$ is negligible.

Equivocality. Let $(\text{pp}_{\text{com}} = ([1], [\alpha]), \text{td}_{\text{com}} = (\alpha, [1], [\alpha]))$ be the output of $\text{CSetup}(1^\kappa)$. Let $[m] \in \mathbb{G}$ be any message. In the normal mode, we have $(com = ([m] + [r_1], [r_1] + r_2 \cdot [\alpha]), r = (r_1, r_2))$ with $r_1, r_2 \leftarrow_{\$} \mathbb{Z}_q$. In the fake mode, $(\overline{com} = ([\beta], [\gamma]), \bar{r} = (\beta - m' \pmod{q}, \alpha^{-1}(\gamma - \beta + m') \pmod{q}))$ with $\beta, \gamma \leftarrow_{\$} \mathbb{Z}_q$.

Note that $\bar{r} = (\bar{r}_1, \bar{r}_2) = (\beta - m' \pmod{q}, \alpha^{-1}(\gamma - \beta + m') \pmod{q})$ implies $\beta = m' + \bar{r}_1, \gamma = \bar{r}_2\alpha + \bar{r}_1$, so $\overline{com} = ([\beta], [\gamma]) = ([m'] + [\bar{r}_1], [\bar{r}_1] + \bar{r}_2[\alpha])$. The uniformity of (β, γ) implies that $(\bar{r}_1, \bar{r}_2)$ is also independently uniformly distributed. Therefore, we have $(com, r) \equiv (\overline{com}, \bar{r})$.

Homomorphism. Let $(\text{pp}_{\text{com}}, \text{td}_{\text{com}}) \leftarrow \text{CSetup}(1^\kappa)$, where $\text{pp}_{\text{com}} = ([1], [\alpha])$. For any $[m^0], [m^1] \in \mathbb{G}$ and any $r^0 = (r_1^0, r_2^0), r^1 = (r_1^1, r_2^1) \in \mathbb{Z}_q^2$, we have

$$\begin{aligned} \text{Commit}(\text{pp}_{\text{com}}, [m^0 + m^1]; r^0 + r^1) &= ([m^0 + m^1] + [r_1^0 + r_1^1], [r_1^0 + r_1^1] + (r_2^0 + r_2^1) \cdot [\alpha]) \\ &= ([m^0] + [m^1] + [r_1^0] + [r_1^1], [r_1^0] + [r_1^1] + r_2^0 \cdot [\alpha] + r_2^1 \cdot [\alpha]) \\ &= ([m^0] + [r_1^0], [r_1^0] + r_2^0 \cdot [\alpha]) + ([m^1] + [r_1^1], [r_1^1] + r_2^1 \cdot [\alpha]) \\ &= \text{Commit}(\text{pp}_{\text{com}}, [m^0]; r^0) + \text{Commit}(\text{pp}_{\text{com}}, [m^1]; r^1). \end{aligned}$$

Thus, the homomorphism property holds.

$\text{Exp}_{TS,\mathcal{A}}^{\text{adp-UF-0}}(\kappa, t, n)$	$\text{Exp}_{TS,\mathcal{A}}^{\text{adp-UF-0.5}}(\kappa, t, n)$
1: $\mathcal{Q}_0 := \emptyset, \mathcal{Q}_{0.5}[\cdot] := \emptyset, \mathcal{CS} := \emptyset, \mathcal{I}[\cdot, \cdot] := \emptyset$ 2: $\text{round}[\cdot, \cdot] := \emptyset, \text{state}[\cdot, \cdot] := \emptyset$ 3: For $i \in [n]$: 4: $\text{num}_i := 0$ // counting sessions involving signer i 5: $pp \leftarrow \text{TS.Setup}(1^\kappa)$ 6: $(vk, \{vk_i\}_{i \in [n]}, \{sk_i\}_{i \in [n]}) \leftarrow \text{TS.KeyGen}(t, n)$ 7: $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{SIGN}_1, \text{SIGN}_2, \text{CORR}}(pp, vk, \{vk_i\}_{i \in [n]})$ 8: If $ m^* \in \mathcal{Q}_0 $: Return 0 9: If $ \mathcal{Q}_{0.5}[m^*] \cup \mathcal{CS} \geq t$: Return 0 10: If $\text{TS.Vrfy}(vk, m^*, \sigma^*) = 1$: Return 1; 11: Else: Return 0 $\text{CORR}(\ell)$ 12: If $ \mathcal{CS} \geq t$: 13: Return $\perp$ 14: $\mathcal{CS} := \mathcal{CS} \cup \{\ell\}$ 15: Return $(sk_\ell, \{\text{state}[\ell, i]\}_{i \in [\text{num}_\ell]})$	$\text{SIGN}_1(SS, \ell, m)$ 16: If $\ell \in \mathcal{CS}$: Return $\perp$ 17: $\mathcal{Q}_0 := \mathcal{Q}_0 \cup \{m\}$ 18: $\text{num}_\ell := \text{num}_\ell + 1$ 19: $\text{iid} := \text{num}_\ell$ 20: $(\text{pm}_{1,\ell}, \text{st}_{1,\ell}) \leftarrow \text{TS.Sign}_1(vk, SS, \ell, sk_\ell, m)$ 21: $\text{state}[\ell, \text{iid}] := \text{st}_{1,\ell}$ 22: $\mathcal{I}[\ell, \text{iid}] := (SS, m)$ 23: $\mathcal{Q}_{0.5}[m] := \mathcal{Q}_{0.5}[m] \cup \{\ell\}$ 24: $\text{round}[\ell, \text{iid}] := 1$ 25: Return $(\text{iid}, \text{pm}_{1,\ell})$ $\text{SIGN}_2(\ell, \text{iid}, \{\text{pm}_{1,i}\}_{i \in SS})$ 26: If $\ell \in \mathcal{CS} \vee \text{round}[\ell, \text{iid}] \neq 1$: Return $\perp$ 27: Parse $\mathcal{I}[\ell, \text{iid}] = (SS, m)$ 28: Parse $\text{state}[\ell, \text{iid}] = \text{st}_{1,\ell}$ 29: $(\text{pm}_{2,\ell}, \text{st}_{2,\ell}) \leftarrow \text{TS.Sign}_2(vk, SS, \ell, sk_\ell, \text{st}_{1,\ell}, m, \{\text{pm}_{1,i}\}_{i \in SS})$ 30: $\text{state}[\ell, \text{iid}] := \text{st}_{2,\ell}$ 31: $\text{round}[\ell, \text{iid}] := 2$ 32: Return $\text{pm}_{2,\ell}$

Fig. 10. The security experiments for a two-round (t, n) -TS scheme: $\text{Exp}_{TS,\mathcal{A}}^{\text{adp-UF-0}}$ [10] (with $\boxed{}$ dashed boxes) and $\text{Exp}_{TS,\mathcal{A}}^{\text{adp-UF-0.5}}$ (with $\boxed{}$ frame boxes).

E Construction of Threshold Signatures with TS-UF-0.5 Security

E.1 Adaptive TS-UF-0.5 Security of Two-Round (t, n) -TS

Unlike TS-UF-0 which excludes a message from forgery once any honest signer is queried on it, TS-UF-0.5 allows first-round queries to multiple honest signers as long as the number of signing queries plus corruption queries remains below t . Thus, TS-UF-0.5 provides a stronger security guarantee than TS-UF-0.

Definition 10 (Adaptive TS-UF-0.5 Security of TS). A two-round (t, n) -threshold signature scheme (t, n) -TS = (TS.Setup, TS.KeyGen, TS.Sign₁, TS.Sign₂, TS.Comb, TS.Vrfy) has adaptive TS-UF-0.5 security, if for any PPT adversary $\mathcal{A}$, it holds that $\text{Adv}_{TS,\mathcal{A}}^{\text{adp-UF-0.5}}(\kappa) := \Pr[\text{Exp}_{TS,\mathcal{A}}^{\text{adp-UF-0.5}}(\kappa, t, n) \Rightarrow 1] \leq \text{negl}(\kappa)$, where the experiment $\text{Exp}_{TS,\mathcal{A}}^{\text{adp-UF-0.5}}$ is defined in Fig. 10.

E.2 Construction of (t, n) -TS'

Our construction of (t, n) -TS' is based on a cyclic group $\mathbb{G}$ of order q with generator g , and relies on one cryptographic primitive $\text{Share}_{(t,n)}$. See Fig. 11.

- $\text{Share}_{(t,n)}$ is a (t, n) -threshold secret sharing scheme with secret space $\mathbb{Z}_q^k$ for $k \geq 2$, satisfying linear reconstruction, perfect secrecy, and linearity property.

<pre> $pp \leftarrow \text{TS.Setup}(1^\kappa) :$ $(\mathbb{G}, g, q) \leftarrow \text{GGen}(1^\kappa)$ $\bar{\mathbf{A}} \leftarrow_{\\$} \mathbb{Z}_q^{(k-1) \times k}$ Return $pp = (\mathbb{G}, g, q, [\bar{\mathbf{A}}])$ $(\text{pm}_{1,\ell}, \text{st}_{1,\ell}) \leftarrow \text{TS.Sign}_1(vk, \mathcal{SS}, \ell, sk_\ell, m) :$ Parse $sk_\ell = \mathbf{x}_\ell$ $[\bar{\mathbf{A}}] := H_0(vk, m) \in \mathbb{G}^{1 \times k}$ $[\mathbf{A}] := \begin{bmatrix} \bar{\mathbf{A}} \\ \mathbf{A} \end{bmatrix}$ $[\underline{y}_\ell] := [\bar{\mathbf{A}}] \cdot \mathbf{x}_\ell = [\bar{\mathbf{A}}\mathbf{x}_\ell]$ $\mathbf{r}_\ell \leftarrow_{\\$} \mathbb{Z}_q^k$ $[\mathbf{w}_\ell] := [\mathbf{A}] \cdot \mathbf{r}_\ell = [\mathbf{A}\mathbf{r}_\ell]$ $\text{pm}_{1,\ell} := ([\underline{y}_\ell], [\mathbf{w}_\ell]) \in \mathbb{G} \times \mathbb{G}^k$ $\text{st}_{1,\ell} := \mathbf{r}_\ell$ Return $(\text{pm}_{1,\ell}, \text{st}_{1,\ell})$ $(\text{pm}_{2,\ell}, \text{st}_{2,\ell}) \leftarrow \text{TS.Sign}_2(vk, \mathcal{SS}, \ell, sk_\ell, \text{st}_{1,\ell}, m, \{\text{pm}_{1,i}\}_{i \in \mathcal{SS}}) :$ Parse $sk_\ell = \mathbf{x}_\ell, \text{st}_{1,\ell} = \mathbf{r}_\ell$ For $i \in \mathcal{SS}$: Parse $\text{pm}_{1,i} = ([\underline{y}_i], [\mathbf{w}_i])$ $[\underline{y}] := \sum_{i \in \mathcal{SS}} [\underline{y}_i] \cdot \lambda_{\mathcal{SS},i} = [\sum_{i \in \mathcal{SS}} \underline{y}_i \cdot \lambda_{\mathcal{SS},i}]$ $[\mathbf{w}] := \sum_{i \in \mathcal{SS}} [\mathbf{w}_i]$ $c := H_1(vk, m, [\underline{y}], [\mathbf{w}])$ $\text{pm}_{2,\ell} := \mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{\mathcal{SS},\ell} \cdot \mathbf{x}_\ell \in \mathbb{Z}_q^k$ $\text{st}_{2,\ell} := \text{st}_{1,\ell}$ Return $(\text{pm}_{2,\ell}, \text{st}_{2,\ell})$ </pre>	<pre> $(vk, \{vk_i\}_{i \in [n]}, \{sk_i\}_{i \in [n]}) \leftarrow \text{TS.KeyGen}(t, n) :$ $\mathbf{x} \leftarrow \mathbb{Z}_q^k$ $\{\mathbf{x}_i\}_{i \in [n]} \leftarrow \text{Share}_{(t,n)}(\mathbf{x})$ $vk := [\bar{\mathbf{y}}] := [\bar{\mathbf{A}}] \cdot \mathbf{x} = [\bar{\mathbf{A}}\mathbf{x}]$ For $i \in [n]$: $sk_i := \mathbf{x}_i$ $vk_i := [\bar{\mathbf{y}}_i] := [\bar{\mathbf{A}}] \cdot \mathbf{x}_i := [\bar{\mathbf{A}}\mathbf{x}_i]$ Return $(vk, \{vk_i\}_{i \in [n]}, \{sk_i\}_{i \in [n]})$ $\sigma \leftarrow \text{TS.Comb}(vk, \mathcal{SS}, m, \{vk_i\}_{i \in \mathcal{SS}}, \{\text{pm}_{1,i}, \text{pm}_{2,i}\}_{i \in \mathcal{SS}}) :$ For $i \in \mathcal{SS}$: Parse $\text{pm}_{1,i} = ([\underline{y}_i], [\mathbf{w}_i])$ Parse $\text{pm}_{2,i} = \mathbf{z}_i$ $[\underline{y}] := \sum_{i \in \mathcal{SS}} [\underline{y}_i] \cdot \lambda_{\mathcal{SS},i} = [\sum_{i \in \mathcal{SS}} \underline{y}_i \cdot \lambda_{\mathcal{SS},i}]$ $\mathbf{z} := \sum_{i \in \mathcal{SS}} \mathbf{z}_i$ $[\mathbf{w}] := \sum_{i \in \mathcal{SS}} [\mathbf{w}_i]$ $c := H_1(vk, m, [\underline{y}], [\mathbf{w}])$ $\sigma := ([\underline{y}], \mathbf{z}, c) \in \mathbb{G} \times \mathbb{Z}_q^k \times \mathbb{Z}_q$ Return σ $0/1 \leftarrow \text{TS.Vrfy}(vk, m, \sigma) :$ Parse $vk = [\bar{\mathbf{y}}]$ Parse $\sigma = ([\underline{y}], \mathbf{z}, c)$ $[\bar{\mathbf{A}}] := H_0(vk, m) \in \mathbb{G}^{1 \times k}$ $[\mathbf{w}] := \begin{bmatrix} \bar{\mathbf{A}} \\ \mathbf{A} \end{bmatrix} \cdot \mathbf{z} - c \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ \underline{y} \end{bmatrix}$ If $c = H_1(vk, m, [\underline{y}], [\mathbf{w}])$: Return 1 Else: Return 0 </pre>
---	---

Fig. 11. The Two-Round Threshold Signature scheme (t, n) -TS'.

Correctness. Let $pp = (\mathbb{G}, g, q, [\bar{\mathbf{A}}])$ be the output of $\text{TS.Setup}(1^\kappa)$. Suppose that $\text{TS.KeyGen}(t, n)$ outputs $vk = [\bar{\mathbf{y}}] = [\bar{\mathbf{A}}\mathbf{x}]$, $vk_i = [\bar{\mathbf{y}}_i] = [\bar{\mathbf{A}}\mathbf{x}_i]$, $sk_i = \mathbf{x}_i$ for $i \in [n]$, where $\{\mathbf{x}_i\}_{i \in [n]}$ is the output of $\text{Share}_{(t,n)}(\mathbf{x})$.

Suppose that in the first round, $\text{TS.Sign}_1(vk, \mathcal{SS}, \ell, sk_\ell = \mathbf{x}_\ell, m)$ outputs $\text{pm}_{1,\ell} = ([\underline{y}_\ell] = [\bar{\mathbf{A}}\mathbf{x}_\ell], [\mathbf{w}_\ell] = [\mathbf{A}\mathbf{r}_\ell])$ and $\text{st}_{1,\ell} = \mathbf{r}_\ell$, where $\mathbf{r} \leftarrow_{\$} \mathbb{Z}_q^k$.

Suppose that in the second round, $\text{TS.Sign}_2(vk, \mathcal{SS}, \ell, sk_\ell = \mathbf{x}_\ell, \text{st}_{1,\ell} = \mathbf{r}_\ell, m, \{\text{pm}_{1,i}\}_{i \in \mathcal{SS}})$ outputs $\text{pm}_{2,\ell} = \mathbf{z}_\ell = \mathbf{r}_\ell + c \cdot \lambda_{\mathcal{SS},\ell} \cdot \mathbf{x}_\ell$ and $\text{st}_{2,\ell} = \text{st}_{1,\ell} = \mathbf{r}_\ell$, where $c = H_1(vk, m, [\underline{y}], [\mathbf{w}])$, $[\underline{y}] = \sum_{i \in \mathcal{SS}} [\underline{y}_i] \cdot \lambda_{\mathcal{SS},i} = [\sum_{i \in \mathcal{SS}} \lambda_{\mathcal{SS},i} \bar{\mathbf{A}}\mathbf{x}_i]$, $[\mathbf{w}] = \sum_{i \in \mathcal{SS}} [\mathbf{w}_i] = [\mathbf{A} \cdot \sum_{i \in \mathcal{SS}} \mathbf{r}_i]$.

If $\sigma = ([\underline{y}], \mathbf{z}, c)$ is the output of $\text{TS.Comb}(vk, \mathcal{SS}, m, \{vk_i = [\bar{\mathbf{y}}_i] = [\bar{\mathbf{A}}\mathbf{x}_i]\}_{i \in \mathcal{SS}}, \{\text{pm}_{1,i} = ([\underline{y}_i] = [\bar{\mathbf{A}}\mathbf{x}_i], [\mathbf{w}_i] = [\mathbf{A}\mathbf{r}_i]), \text{pm}_{2,i} = \mathbf{z}_i = \mathbf{r}_i + c \cdot \lambda_{\mathcal{SS},i} \cdot \mathbf{x}_i\}_{i \in \mathcal{SS}})$. Due to the linear reconstruction of secret sharing, for any $\mathcal{SS}$ with $|\mathcal{SS}| \geq t$, we have $\mathbf{x} = \sum_{i \in \mathcal{SS}} \lambda_{\mathcal{SS},i} \mathbf{x}_i$. Accordingly,

$$\begin{cases}
[\underline{y}] &= \sum_{i \in \mathcal{SS}} [\underline{y}_i] \cdot \lambda_{\mathcal{SS},i} = [\sum_{i \in \mathcal{SS}} \lambda_{\mathcal{SS},i} \bar{\mathbf{A}}\mathbf{x}_i] = [\bar{\mathbf{A}}\mathbf{x}], \\
\mathbf{z} &= \sum_{i \in \mathcal{SS}} \mathbf{z}_i = \sum_{i \in \mathcal{SS}} \mathbf{r}_i + c \cdot \sum_{i \in \mathcal{SS}} \lambda_{\mathcal{SS},i} \mathbf{x}_i = \sum_{i \in \mathcal{SS}} \mathbf{r}_i + c\mathbf{x}, \\
c &= H_1(vk, m, [\underline{y}], [\mathbf{w}]) = H_1(vk, m, [\underline{y}], [\mathbf{A} \cdot (\sum_{i \in \mathcal{SS}} \mathbf{r}_i)]).
\end{cases}$$

Finally, we analyze the verification algorithm $\text{TS.Vrfy}(vk = [\bar{\mathbf{y}}], m, \sigma = ([\underline{y}], \mathbf{z}, c))$, observe that

$$\begin{aligned}
[\mathbf{w}'] &:= \begin{bmatrix} \mathbf{A} \\ \mathbf{A} \end{bmatrix} \cdot \mathbf{z} - c \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y \end{bmatrix} = \begin{bmatrix} \mathbf{A} \cdot (\sum_{i \in \mathcal{SS}} \mathbf{r}_i + c\mathbf{x}) \\ \mathbf{A} \cdot (\sum_{i \in \mathcal{SS}} \mathbf{r}_i + c\mathbf{x}) \end{bmatrix} - \begin{bmatrix} c \cdot \mathbf{A}\mathbf{x} \\ c \cdot \mathbf{A}\mathbf{x} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A} \cdot (\sum_{i \in \mathcal{SS}} \mathbf{r}_i) \\ \mathbf{A} \cdot (\sum_{i \in \mathcal{SS}} \mathbf{r}_i) \end{bmatrix} = [\mathbf{A} \cdot (\sum_{i \in \mathcal{SS}} \mathbf{r}_i)] = [\mathbf{w}].
\end{aligned}$$

Thus, we have $H_1(vk, m, [y], [\mathbf{w}']) = H_1(vk, m, [y], [\mathbf{w}]) = c$ holds, yielding $\text{TS.Vrfy}(vk, m, \sigma) = 1$.

E.3 Adaptive TS-UF-0.5 Security of (t, n) -TS'

Theorem 3. Suppose that $\text{Share}_{(t,n)}$ is a (t, n) -threshold secret sharing scheme with perfect secrecy and linearity. If the $\mathcal{U}_{k,k-1}$ -MDDH assumption holds, then our (t, n) -TS' construction (see Fig. 11) achieves adaptive TS-UF-0.5 security in the random oracle model. More precisely, for any PPT adversary $\mathcal{A}$ making at most Q_0 random-oracle queries to H_0 and Q_1 queries to H_1 , we have PPT algorithms $\mathcal{B}_1, \mathcal{B}_2$, s.t. $\text{Adv}_{\text{TS}, \mathcal{A}}^{\text{adp-UF-0.5}}(\kappa)$

$$\leq \frac{Q_0^2 + Q_1^2 + 4Q_0 + Q_1 + 3}{q} + \sqrt{Q_1 \cdot \left(\text{Adv}_{\mathbb{G}, \mathcal{B}_1}^{\mathcal{U}_{k,k-1}\text{-MDDH}}(\kappa) + \frac{3}{q} \right) + Q_0 \cdot \text{Adv}_{\mathbb{G}, \mathcal{B}_2}^{\mathcal{U}_{k,k-1}\text{-MDDH}}(\kappa)}$$

Proof. We define a series of games $\mathbf{G}_0$ - $\mathbf{G}_7$ and define Win as the event that the game outputs 1. Let $\Pr_i[\text{Win}]$ denote the probability that $\mathbf{G}_i$ outputs 1. We will show $\mathbf{G}_0$ is the original $\text{Exp}_{\text{TS}, \mathcal{A}}^{\text{adp-UF-0.5}}$, and bound $\Pr[\text{Exp}_{\text{TS}, \mathcal{A}}^{\text{adp-UF-0.5}}(\kappa, t, n) \Rightarrow 1]$ with $\Pr_i[\text{Win}]$ for $i \in [0, 7]$.

Below we describe the games and the difference between two adjacent games are highlighted by gray background.

Game $\mathbf{G}_0$. $\mathbf{G}_0$ is the original security experiment $\text{Exp}_{\text{TS}, \mathcal{A}}^{\text{adp-UF-0.5}}$ (see Fig. 1), except that both random oracles H_0, H_1 are simulated by lazy sampling. More precisely, $\mathbf{G}_0$ keeps two sets $\text{List}_0, \text{List}_1$ to record the queries to and answers from H_0, H_1 respectively. And $\mathbf{G}_0$ initializes $\text{ctr}_0 := 0$ and $\text{ctr}_1 := 0$ to count the number of fresh queries to H_0 and H_1 , respectively.

For each query (vk', m) to H_0 , either from $\mathcal{A}$ or from oracle SIGN_1 , if (vk', m) is a fresh query, i.e., $\nexists (\cdot, vk', m, \cdot) \in \text{List}_0$, then $\mathbf{G}_0$ samples $\mathbf{u} \leftarrow_{\$} \mathbb{Z}_q^k$, updates $\text{ctr}_0 := \text{ctr}_0 + 1$, adds $(\text{ctr}_0, vk', m, [\mathbf{u}^\top])$ to List_0 , and returns $[\mathbf{u}^\top]$ as the hash value of $H_0(vk', m)$; otherwise there must exist $(\cdot, vk', m, [\mathbf{u}^\top]) \in \text{List}_0$, and $\mathbf{G}_0$ returns $[\mathbf{u}^\top]$ as the hash value of $H_0(vk', m)$.

For each query $(vk', m, [y], [\mathbf{w}])$ to H_1 , if $(vk', m, [y], [\mathbf{w}])$ is a fresh query, i.e., $\nexists (\cdot, vk', m, [y], [\mathbf{w}], \cdot) \in \text{List}_1$, then $\mathbf{G}_0$ samples $c \leftarrow_{\$} \mathbb{Z}_q$, updates $\text{ctr}_1 := \text{ctr}_1 + 1$, adds $(\text{ctr}_1, vk', m, [y], [\mathbf{w}], c)$ to List_1 , and returns c as the hash value of $H_1(vk', m, [y], [\mathbf{w}])$; otherwise there must exist $(\cdot, vk', m, [y], [\mathbf{w}], c) \in \text{List}_1$, and then $\mathbf{G}_0$ returns c as the hash value of $H_1(vk', m, [y], [\mathbf{w}])$.

Suppose that $\mathcal{A}$ outputs a forged signature $\sigma^* = ([y^*], \mathbf{z}^*, c^*)$ for a fresh message m^* . Here, fresh message m^* means that $\mathcal{A}$ cannot issue first-round signing oracle queries for m^* to t_1 signers and corrupt t_2 signers such that $t_1 + t_2 \geq t$; equivalently, $|\mathcal{Q}_{0.5}[m^*] \cup \mathcal{CS}| < t$. Then Win happens in G_0 (i.e., $\text{Exp}_{TS, \mathcal{A}}^{\text{adp-UF-0.5}}$ outputs 1) if and only if m^* is fresh and $\text{TS.Vrfy}(vk, m^*, \sigma^*) = 1$.

Clearly,

$$\text{Adv}_{TS, \mathcal{A}}^{\text{adp-UF-0.5}}(\kappa) = \Pr[\text{Exp}_{TS, \mathcal{A}}^{\text{adp-UF-0.5}}(\kappa, t, n) \Rightarrow 1] = \Pr_0[\text{Win}]. \quad (23)$$

Game G_1 . G_1 is the same as G_0 , except that G_1 directly returns 0 if hash collisions occur for H_0 or H_1 . In other words, G_1 returns 0 if there exist $(\cdot, vk_0, m_0, [\mathbf{u}_0^\top]) \in \text{List}_0$ and $(\cdot, vk_1, m_1, [\mathbf{u}_1^\top]) \in \text{List}_0$ such that $(vk_0, m_0) \neq (vk_1, m_1)$ but $[\mathbf{u}_0^\top] = [\mathbf{u}_1^\top]$ or there exist $(vk_0, m_0, [y_0], [\mathbf{w}_0], c_0) \in \text{List}_1$ and $(vk_1, m_1, [y_1], [\mathbf{w}_1], c_1) \in \text{List}_1$ such that $(vk_0, m_0, [y_0], [\mathbf{w}_0]) \neq (vk_1, m_1, [y_1], [\mathbf{w}_1])$ but $c_0 = c_1$.

Due to the uniformity of $\mathbf{u}_0^\top, \mathbf{u}_1^\top$ and c_0, c_1 , G_1 returns 0 with probability at most $\frac{Q_0^2}{q^k} + \frac{Q_1^2}{q} \leq \frac{Q_0^2 + Q_1^2}{q}$. Thus we have

$$\Pr_0[\text{Win}] \leq \Pr_1[\text{Win}] + \frac{Q_0^2 + Q_1^2}{q}. \quad (24)$$

Game G_2 . G_2 is the same as G_1 , except that G_2 directly returns 0 if event Queried1 (defined below) does not occur. Consequently, $\Pr_2[\text{Win}] = \Pr_1[\text{Win} \wedge \text{Queried1}]$.

Given $\mathcal{A}$'s forgery $(m^*, \sigma^* = ([y^*], \mathbf{z}^*, c^*))$, let $[\mathbf{A}^*] := H_0(vk, m^*)$ and $[\mathbf{w}^*] := \begin{bmatrix} \mathbf{A} \\ \mathbf{A}^* \end{bmatrix} \cdot \mathbf{z}^* - c^* \cdot \begin{bmatrix} \mathbf{y} \\ y^* \end{bmatrix}$. Define

Queried1 : $\mathcal{A}$ has ever issued a query $(vk, m^*, [y^*], [\mathbf{w}^*])$ to oracle H_1 .

In G_1 , we have $\Pr_1[\text{Win}] = \Pr_1[\text{Win} \wedge \text{Queried1}] + \Pr_1[\text{Win} \wedge \neg \text{Queried1}]$. Recall that Win happens means $\text{TS.Vrfy}(vk, m^*, \sigma^*) = 1$, i.e.,

$$c^* = H_1(vk, m^*, [y^*], [\mathbf{w}^*]).$$

If Queried1 does not occur, then $H_1(vk, m^*, [y^*], [\mathbf{w}^*])$ is uniform over $\mathbb{Z}_q$, and hence $c^* = H_1(vk, m^*, [y^*], [\mathbf{w}^*])$ holds with probability $1/q$. Therefore,

$$\Pr_1[\text{Win} \wedge \neg \text{Queried1}] \leq 1/q, \quad \Pr_1[\text{Win}] \leq \Pr_1[\text{Win} \wedge \text{Queried1}] + 1/q.$$

Given $\Pr_2[\text{Win}] = \Pr_1[\text{Win} \wedge \text{Queried1}]$, we have

$$\Pr_1[\text{Win}] \leq \Pr_1[\text{Win} \wedge \text{Queried1}] + 1/q = \Pr_2[\text{Win}] + 1/q. \quad (25)$$

Note that in if Win occurs in G_2 , there must exist $(\text{idx}, vk, m^*, [y^*], [\mathbf{w}^*], c^*) \in \text{List}_1$ before $\mathcal{A}$ outputs its forgery.

Game G_3 . G_3 is the same as G_2 , except that G_3 returns 0 if either event **CorrectY** or event **Queried0** (defined below) does not occur. Consequently, $\Pr_3[\text{Win}] = \Pr_2[\text{Win} \wedge \text{CorrectY} \wedge \text{Queried0}]$. Define

CorrectY : $\mathcal{A}$'s forgery (m^*, σ^*) satisfies $[y^*] = [\mathbf{A}^*] \cdot \mathbf{x}$, with $[\mathbf{A}^*] := H_0(vk, m^*)$.
Queried0 : $\mathcal{A}$ has ever issued a query (vk, m^*) to oracle H_0 .

Then we can have

$$\begin{aligned} \Pr_2[\text{Win}] &= \Pr_2[\text{Win} \wedge \text{CorrectY} \wedge \text{Queried0}] + \Pr_2[\text{Win} \wedge \neg(\text{CorrectY} \wedge \text{Queried0})] \\ &= \Pr_3[\text{Win}] + \Pr_2[\text{Win} \wedge \neg\text{CorrectY}] + \Pr_2[\text{Win} \wedge \text{CorrectY} \wedge \neg\text{Queried0}]. \end{aligned} \quad (26)$$

We will prove $\Pr_2[\text{Win}] \leq \Pr_3[\text{Win}] + \frac{Q_1+2}{q} + \sqrt{Q_1 \cdot \left(\frac{3}{q} + \text{Adv}_{\mathbb{G}, \mathcal{B}}^{\mathcal{U}_{k,k-1}\text{-MDDH}}(\kappa)\right)}$ via Claim 5 and Claim 6.

$$\text{Claim 5. } \Pr_2[\text{Win} \wedge \neg\text{CorrectY}] \leq \frac{Q_1}{q} + \sqrt{Q_1 \cdot \left(\frac{3}{q} + \text{Adv}_{\mathbb{G}, \mathcal{B}}^{\mathcal{U}_{k,k-1}\text{-MDDH}}(\kappa)\right)}.$$

Proof. To bound the probability that $\text{Win} \wedge \neg\text{CorrectY}$ occurs in G_2 , we use the generalized forking lemma (see Lemma 3).

Firstly, we define an experiment called **Exp** (see Fig. 12). **Exp** takes as input a group description $(\mathbb{G}, g, q)$, $[\mathbf{B}] \in \mathbb{G}^{k \times (k-1)}$, $\mathbf{x} \in \mathbb{Z}_q^k$, Q_1 hash values $h_1, h_2, \dots, h_{Q_1}$ and randomness ρ to simulate G_2 and detect $\text{Win} \wedge \neg\text{CorrectY}$ in G_2 .

Description of Exp (see Fig. 12). It first sets $[\bar{\mathbf{A}}] := [\mathbf{B}^\top]$ and $pp := (\mathbb{G}, g, q, [\bar{\mathbf{A}}])$. It then uses $\mathbf{x}$ to generate $\{sk_i = \mathbf{x}_i\}_{i \in [n]}$ via $\{\mathbf{x}_i\}_{i \in [n]} \leftarrow \text{Share}_{(t,n)}(\mathbf{x})$, and computes $vk = [\bar{\mathbf{A}}\mathbf{x}]$ and $vk_i = [\bar{\mathbf{A}}\mathbf{x}_i]$. It simulates oracle H_1 by setting $h_1, h_2, \dots, h_{Q_1}$ as the responses of the Q_1 fresh queries to oracle H_1 , resulting in $\text{List}_1 = \{(i, \cdot, \cdot, \cdot, h_i)\}_{i \in [Q_1]}$. It also simulates **SIGN**₁, **SIGN**₂, **CORR**, H_0 just like in G_2 .

After $\mathcal{A}$ outputs a forgery (m^*, σ^*) with $\sigma^* = ([y^*], \mathbf{z}^*, c^*)$. **Exp** computes $[\mathbf{A}^*] := H_0(vk, m^*)$, $[\mathbf{w}^*] := \begin{bmatrix} \bar{\mathbf{A}} \\ \mathbf{A}^* \end{bmatrix} \cdot \mathbf{z}^* - c^* \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y^* \end{bmatrix}$, and checks that **Queried1** does occur, the message m^* is fresh, the forgery (m^*, σ^*) is valid, hash collision does not occur, and **CorrectY** does not occur. If any of these checks fails, **Exp** returns $(0, \perp)$, otherwise **Exp** retrieves $(\text{idx}, vk, m^*, [y^*], [\mathbf{w}^*], c^*)$ from List_1 , and returns $(\text{idx}, (m^*, \sigma^*))$.

We define the success probability of **Exp** as

$$\text{acc} := \Pr[(\text{idx}, (m^*, \sigma^*)) \leftarrow \text{Exp}(\mathbb{G}, g, q, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{Q_1}; \rho) : \text{idx} \neq 0].$$

It is easy to see that

$$\Pr_2[\text{Win} \wedge \neg\text{CorrectY}] = \text{acc}. \quad (27)$$

Next, we will show that

$$\Pr_2[\text{Win} \wedge \neg\text{CorrectY}] = \text{acc} \leq \frac{Q_1}{q} + \sqrt{Q_1 \cdot \left(\frac{1}{q} + \text{Adv}_{\mathbb{G}, \mathcal{B}}^{\mathcal{U}_{k,k-1}\text{-KerMDH}}(\kappa)\right)}.$$

$\text{Exp}(\mathbb{G}, g, q, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{Q_1}; \rho)$ 1: $\mathcal{Q}_{0.5}[\cdot] := \emptyset, \mathcal{CS} := \emptyset, \text{List}_0 := \emptyset, \text{List}_1 := \emptyset$ 2: $\text{ctr}_0 := 0, \text{ctr}_1 := 0$ 3: For $i \in [n]$: $\text{num}_i := 0$ 4: $[\mathbf{A}] := [\mathbf{B}^\top]$ 5: $pp := (\mathbb{G}, g, q, [\mathbf{A}])$ 6: $\{\mathbf{x}_i\}_{i \in [n]} \leftarrow \text{Share}_{(t,n)}(\mathbf{x})$ 7: $vk := [\mathbf{y}^*] := [\mathbf{A}] \cdot \mathbf{x} = [\mathbf{A}\mathbf{x}]$ 8: For $i \in [n]$: 9: $vk_i := [\mathbf{y}^*] := [\mathbf{A}] \cdot \mathbf{x}_i = [\mathbf{A}\mathbf{x}_i]$ 10: $sk_i := \mathbf{x}_i$ 11: $(m^*, \sigma^* = ([\underline{y}^*], \mathbf{z}^*, c^*)) \leftarrow \mathcal{A}^{\text{SIGN}_1, \text{SIGN}_2, \text{CORR}, H_0, H_1}(pp, vk, \{vk_i\}_{i \in [n]})$ 12: $[\mathbf{A}^*] := H_0(vk, m^*)$ 13: $[\mathbf{w}^*] := \begin{bmatrix} \mathbf{A} \\ \mathbf{A}^* \end{bmatrix} \cdot \mathbf{z}^* - c^* \cdot \begin{bmatrix} \mathbf{y}^* \\ \underline{y}^* \end{bmatrix}$ 14: If $(\cdot, vk, m^*, [\underline{y}^*], [\mathbf{w}^*], \cdot) \notin \text{List}_1$: 15: Return $(0, \perp)$ // Queried1 does not occur 16: If $ \mathcal{Q}_{0.5}[m^*] \cup \mathcal{CS} \geq t \vee H_1(vk, m^*, [\underline{y}^*], [\mathbf{w}^*]) \neq c^*$: 17: Return $(0, \perp)$ // m^* is not fresh or σ^* is not valid 18: If $(\exists (\text{ctr}', \cdot, \cdot, \theta), (\text{ctr}', \cdot, \cdot, \theta') \in \text{List}_0 : \text{ctr}' \neq \text{ctr}_0 \wedge \theta = \theta')$ $\vee (\exists i, j \in [Q_1] : i \neq j \wedge h_i = h_j)$: 19: Return $(0, \perp)$ // Hash collisions occurs in H_0 or H_1 20: If $[\underline{y}^*] = [\mathbf{A}^*] \cdot \mathbf{x}$: 21: Return $(0, \perp)$ // CorrectY occurs 22: Retrieve $(\text{idx}, vk, m^*, [\underline{y}^*], [\mathbf{w}^*], c^*)$ from List_1 23: Return $(\text{idx}, (m^*, \sigma^*))$	$\text{SIGN}_1(SS, \ell, m)$ 24: The same as $\mathbb{G}_2$ $\text{SIGN}_2(\ell, \text{iid}, \{\mathbf{pm}_{1,i}\}_{i \in SS})$ 25: The same as $\mathbb{G}_2$ $\text{CORR}(\ell)$ 26: The same as $\mathbb{G}_2$ $H_0(vk', m)$ 27: The same as $\mathbb{G}_2$ $H_1(vk', m, [\underline{y}], [\mathbf{w}])$ 28: If $(\cdot, vk', m, [\underline{y}], [\mathbf{w}], c) \in \text{List}_1$: 29: Return c 30: $\text{ctr}_1 := \text{ctr}_0 + 1$ 31: $c := h_{\text{ctr}_1}$ 32: $\text{List}_1 := \text{List}_1 \cup \{(\text{ctr}_1, vk', m, [\underline{y}], [\mathbf{w}], c)\}$ 33: Return c
--	--

Fig. 12. Description of Experiment Exp.

Algorithm $\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}])$: 1: $\mathbf{x} \leftarrow \mathbb{Z}_q^k$ // Begin Fork $^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}], \mathbf{x})$ 2: Choose the random coins ρ for Exp 3: $\{h_1, \dots, h_{Q_1}\} \leftarrow \mathbb{Z}_q^{Q_1}$ 4: $(\text{idx}, (m_0^*, \sigma_0^*)) \leftarrow \text{Exp}(\mathbb{G}, g, q, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{Q_1}; \rho)$ 5: If $\text{idx} = 0$: return $\perp$ 6: $\{h'_{\text{idx}}, \dots, h'_{Q_1}\} \leftarrow \mathbb{Z}_q^{Q_1 - \text{idx} + 1}$ 7: $(\text{idx}', (m_1^*, \sigma_1^*)) \leftarrow \text{Exp}(\mathbb{G}, g, q, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{\text{idx}-1}, h'_{\text{idx}}, \dots, h'_{Q_1}; \rho)$ 8: If $\text{idx} \neq \text{idx}'$: return $\perp$ // End Fork $^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}], \mathbf{x})$ 9: Parse $\sigma_0^* = ([\underline{y}_0^*], \mathbf{z}_0^*, c_0^*)$, $\sigma_1^* = ([\underline{y}_1^*], \mathbf{z}_1^*, c_1^*)$ 10: If $c_0^* = c_1^*$: return $\perp$ 11: $\tilde{\mathbf{z}} := (\mathbf{z}_0^* - \mathbf{z}_1^*) / (c_0^* - c_1^*)$ 12: Return $\tilde{\mathbf{z}} - \mathbf{x}$
--

Fig. 13. Description of Algorithm $\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}$.

This is achieved by a security reduction to the KerMDH assumption with the help of the generalized forking lemma. Below we construct a reduction algorithm $\mathcal{B}_{\text{KerMDH}}$ accessing Exp. See Fig. 13.

Given a group description $(\mathbb{G}, g, q)$ and $[\mathbf{B}] \in \mathbb{G}^{k \times k-1}$, $\mathcal{B}_{\text{KerMDH}}$ aims to find a $\mathbf{v} \in \mathbb{Z}_q^k$ s.t. $\mathbf{v}^\top \cdot [\mathbf{B}] = [\mathbf{0}] \wedge \mathbf{v} \neq \mathbf{0}$. To achieve this, $\mathcal{B}_{\text{KerMDH}}$ samples $\mathbf{x} \leftarrow_s \mathbb{Z}_q^k$, and then implicitly invokes $\text{Fork}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}], \mathbf{x})$ (see Fig. 8 for Fork). More precisely, it samples random coins ρ and $h_1, \dots, h_{Q_1} \leftarrow_s \mathbb{Z}_q$, and invoke $\text{Exp}(\mathbb{G}, g, q, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{Q_1}; \rho)$ to obtain $(\text{idx}, (m_0^*, \sigma_0^*))$.

- If $\text{idx} = 0$, it returns $\perp$.
- If $\text{idx} \neq 0$, it samples fresh values $h'_{\text{idx}}, \dots, h'_{Q_1} \leftarrow \mathbb{Z}_q$, and re-invokes $(\text{idx}', (m_1^*, \sigma_1^*)) \leftarrow \text{Exp}(\mathbb{G}, g, q, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{\text{idx}-1}, h'_{\text{idx}}, \dots, h'_{Q_1}; \rho)$.
 - If $\text{idx} \neq \text{idx}'$, then it returns $\perp$.
 - If $\text{idx} = \text{idx}'$, let $\sigma_0^* = ([y_0^*], \mathbf{z}_0^*, c_0^*)$ and $\sigma_1^* = ([y_1^*], \mathbf{z}_1^*, c_1^*)$. Then $c_0^* = h_{\text{idx}}$ and $c_1^* = h'_{\text{idx}}$. If $[y_0^*] \neq [y_1^*] \vee c_0^* = c_1^*$, $\mathcal{B}_{\text{KerMDH}}$ outputs $\perp$. Otherwise, $\mathcal{B}_{\text{KerMDH}}$ defines $\tilde{\mathbf{z}} := (\mathbf{z}_0^* - \mathbf{z}_1^*) / (c_0^* - c_1^*)$ and outputs $\tilde{\mathbf{z}} - \mathbf{x}$.

We define the forking probability

$$\text{frk} := \Pr[\text{idx} \neq 0 \wedge \text{idx}' = \text{idx}].$$

Then frk

$$\begin{aligned} &\leq \Pr[c_0^* = c_1^* \wedge \text{idx} \neq 0 \wedge \text{idx}' = \text{idx}] + \Pr[\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}]) \Rightarrow (\tilde{\mathbf{z}} - \mathbf{x})] \\ &\leq \Pr[c_0^* = c_1^* \mid \text{idx} \neq 0 \wedge \text{idx}' = \text{idx}] + \Pr[\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}]) \Rightarrow (\tilde{\mathbf{z}} - \mathbf{x})] \end{aligned} \quad (28)$$

Suppose that $\text{idx} \neq 0 \wedge \text{idx}' = \text{idx}$ occurs. Due to the uniformity of h_{idx} and h'_{idx} , we have $c_0^* = h_{\text{idx}} = h'_{\text{idx}} = c_1^*$ with probability $1/q$. Thus

$$\Pr[c_0^* = c_1^* \mid \text{idx} \neq 0 \wedge \text{idx}' = \text{idx}] \leq 1/q. \quad (29)$$

Recall that in the first invocation of Exp , we have $(\text{idx}, vk, m_0^*, [y_0^*], [\mathbf{w}_0^*], c_0^*) \in \text{List}_1$ with $[\mathbf{w}_0^*] = \begin{bmatrix} \bar{\mathbf{A}} \\ \mathbf{A}_0^* \end{bmatrix} \cdot \mathbf{z}_0^* - c_0^* \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y_0^* \end{bmatrix}$, and the output is $(\text{idx}, (m_0^*, \sigma_0^* = ([y_0^*], \mathbf{z}_0^*, c_0^*)))$. In the second invocation of Exp , we have $(\text{idx}', vk, m_1^*, [y_1^*], [\mathbf{w}_1^*], c_1^*) \in \text{List}_1$ with $[\mathbf{w}_1^*] = \begin{bmatrix} \bar{\mathbf{A}} \\ \mathbf{A}_1^* \end{bmatrix} \cdot \mathbf{z}_1^* - c_1^* \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y_1^* \end{bmatrix}$, and the output is $(\text{idx}', (m_1^*, \sigma_1^* = ([y_1^*], \mathbf{z}_1^*, c_1^*)))$. Given $\text{idx} = \text{idx}' \neq 0$, and the same random coins ρ for the two invocations of Exp , $\text{Exp}(\mathbb{G}, g, q, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{\text{idx}-1}, h_{\text{idx}}, \dots, h_{Q_1}; \rho)$ and $\text{Exp}(\mathbb{G}, g, q, [\mathbf{B}], \mathbf{x}, h_1, \dots, h_{\text{idx}-1}, h'_{\text{idx}}, \dots, h'_{Q_1}; \rho)$ are identical until receiving the reply of the idx -th fresh query from oracle H_1 . Obviously, the idx -th query $(vk, m^*, [y^*], [\mathbf{w}^*])$ is the same in the two invocations of Exp . Therefore, we have

$$\begin{aligned} m_0^* &= m_1^* = m^*, \quad [\mathbf{A}_0^*] = [\mathbf{A}_1^*] = [\mathbf{A}^*] = H_0(vk, m^*), \\ [y_0^*] &= [y_1^*] = [y^*], \quad [\mathbf{w}_0^*] = [\mathbf{w}_1^*] = [\mathbf{w}^*]. \end{aligned} \quad (30)$$

Finally, let us bound $\Pr[\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}]) \Rightarrow (\tilde{\mathbf{z}} - \mathbf{x})]$.

If $\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}])$ outputs $(\tilde{\mathbf{z}} - \mathbf{x})$, then we have $\text{idx} = \text{idx}' \neq 0$, $c_0^* \neq c_1^*$. According to (30), we have $\begin{bmatrix} \bar{\mathbf{A}} \\ \mathbf{A}^* \end{bmatrix} \cdot \mathbf{z}_0^* - c_0^* \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y^* \end{bmatrix} = [\mathbf{w}^*] = \begin{bmatrix} \bar{\mathbf{A}} \\ \mathbf{A}^* \end{bmatrix} \cdot \mathbf{z}_1^* - c_1^* \cdot \begin{bmatrix} \bar{\mathbf{y}} \\ y^* \end{bmatrix}$.

Given $c_0^* \neq c_1^*$, and $\tilde{\mathbf{z}} := (\mathbf{z}_0^* - \mathbf{z}_1^*) \cdot (c_0^* - c_1^*)^{-1}$, we have $\begin{bmatrix} \overline{\mathbf{A}} \\ \underline{\mathbf{A}^*} \end{bmatrix} \cdot \tilde{\mathbf{z}} = \begin{bmatrix} \overline{\mathbf{y}} \\ \underline{y^*} \end{bmatrix}$, and hence $\overline{\mathbf{A}} \cdot \tilde{\mathbf{z}} = \overline{\mathbf{y}}$ and $\underline{\mathbf{A}^*} \cdot \tilde{\mathbf{z}} = \underline{y^*}$.

On the one hand, we have

$$\overline{\mathbf{A}} \cdot \tilde{\mathbf{z}} = \overline{\mathbf{y}} = \overline{\mathbf{A}} \cdot \mathbf{x} \Rightarrow \overline{\mathbf{A}} \cdot (\tilde{\mathbf{z}} - \mathbf{x}) = \mathbf{0} \Rightarrow [\mathbf{B}^\top] \cdot (\tilde{\mathbf{z}} - \mathbf{x}) = \mathbf{0}.$$

On the other hand, $\neg \text{CorrectY} \Rightarrow [\underline{\mathbf{A}^*}] \cdot \mathbf{x} \neq \underline{y^*}$. Together with $[\underline{\mathbf{A}^*}] \cdot \tilde{\mathbf{z}} = \underline{y^*}$, we have $\tilde{\mathbf{z}} - \mathbf{x} \neq \mathbf{0}$.

Therefore, the output $\tilde{\mathbf{z}} - \mathbf{x}$ of $\mathcal{B}_{\text{KerMDH}}$ is a valid non-zero solution to the KerMDH problem. So

$$\Pr[\mathcal{B}_{\text{KerMDH}}^{\text{Exp}}(\mathbb{G}, g, q, [\mathbf{B}]) \Rightarrow (\tilde{\mathbf{z}} - \mathbf{x})] \leq \text{Adv}_{\mathbb{G}, \mathcal{B}}^{\mathcal{U}_{k, k-1}\text{-KerMDH}}(\kappa). \quad (31)$$

Combining (28), (29), (31) yields

$$\text{frk} \leq \frac{1}{q} + \text{Adv}_{\mathbb{G}, \mathcal{B}}^{\mathcal{U}_{k, k-1}\text{-KerMDH}}(\kappa). \quad (32)$$

According to the generalized forking lemma (see Lemma 3), we have

$$\text{acc} \leq \frac{Q_1}{q} + \sqrt{Q_1 \cdot \text{frk}}. \quad (33)$$

Combining (27), (32), (33) yields

$$\Pr_2[\text{Win} \wedge \neg \text{CorrectY}] = \text{acc} \leq \frac{Q_1}{q} + \sqrt{Q_1 \cdot \left(\frac{1}{q} + \text{Adv}_{\mathbb{G}, \mathcal{B}}^{\mathcal{U}_{k, k-1}\text{-KerMDH}}(\kappa) \right)}.$$

According to Lemma 2, we have a PPT algorithm $\mathcal{B}_1$ such that

$$\Pr_2[\text{Win} \wedge \neg \text{CorrectY}] \leq \frac{Q_1}{q} + \sqrt{Q_1 \cdot \left(\frac{3}{q} + \text{Adv}_{\mathbb{G}, \mathcal{B}_1}^{\mathcal{U}_{k, k-1}\text{-MDDH}}(\kappa) \right)}.$$

□

Claim 6. $\Pr_3[\text{Win} \wedge \text{CorrectY} \wedge \neg \text{Queried0}] \leq \frac{2}{q}$.

Proof. If Queried0 does not occur, then $\mathcal{A}$ must have not queried $H_0(vk, m^*)$. Thus, $[\underline{\mathbf{A}^*}] := H_0(vk, m^*)$ is uniformly distributed. Given that $\mathbf{x}$ is uniformly sampled from $\mathbb{Z}_q^k$, thus $\mathbf{x} \neq \mathbf{0}$ except with probability $\frac{1}{q^k}$. As a result, if $\mathbf{x} \neq \mathbf{0}$, then $[\underline{\mathbf{A}^*}] \cdot \mathbf{x}$ is uniformly distributed, and hence $\underline{y^*} = [\underline{\mathbf{A}^*}] \cdot \mathbf{x}$ with probability $\frac{1}{q}$. Consequently, we have $\Pr_3[\text{Win} \wedge \text{CorrectY} \wedge \neg \text{Queried0}] \leq \frac{1}{q} + \frac{1}{q^k} \leq \frac{2}{q}$. □

Combining (26), Claim 5 and Claim 6, we have

$$\Pr_2[\text{Win}] \leq \Pr_3[\text{Win}] + \frac{Q_1 + 2}{q} + \sqrt{Q_1 \cdot \left(\frac{3}{q} + \text{Adv}_{\mathbb{G}, \mathcal{B}}^{\mathcal{U}_{k, k-1}\text{-MDDH}}(\kappa) \right)}. \quad (34)$$

Game G_4 . G_4 samples $\text{ctr}^* \leftarrow_{\$} [Q_0]$, and checks if **GoodGuess** (defined below) occurs, where event **GoodGuess** defined as

GoodGuess : The ctr^* -th fresh hash query (vk', m) to H_0 satisfies $vk' = vk \wedge m = m^*$.

G_4 is the same as G_3 , except that G_4 returns 0 if **GoodGuess** does not occur. Consequently, $\Pr_4[\text{Win}] = \Pr_3[\text{Win} \wedge \text{GoodGuess}]$.

If **Win** occurs in G_4 , then **Queried0** must occur (thanks to G_3). Due to the uniformity of ctr^* , we have $\Pr_3[\text{GoodGuess}] \geq \frac{1}{Q_0}$.

Note that **GoodGuess** is independent of $\mathcal{A}$'s view, so we have $\Pr_4[\text{Win}] = \Pr_3[\text{Win} \wedge \text{GoodGuess}] = \Pr_3[\text{Win}] \cdot \Pr_3[\text{GoodGuess}] \geq \frac{1}{Q_0} \cdot \Pr_3[\text{Win}]$. In other words,

$$\Pr_3[\text{Win}] \leq Q_0 \cdot \Pr_4[\text{Win}] \quad (35)$$

Game G_5 . G_5 is the same as G_4 , except that G_5 changes the way of answering hash queries to H_0 . More precisely, upon the ctr -th fresh query (vk', m) to oracle H_0 for $\text{ctr} \in Q_1$, game G_5 replies with

$$\begin{cases} H_0(vk', m) := [\mathbf{u}_{\text{ctr}}^\top] \text{ with } \mathbf{u}_{\text{ctr}} \leftarrow_{\$} \mathbb{Z}_q^k, & \text{if } \text{ctr} = \text{ctr}^* \\ H_0(vk', m) := [\mathbf{v}_{\text{ctr}}^\top \overline{\mathbf{A}}] \text{ with } \mathbf{v}_{\text{ctr}} \leftarrow_{\$} \mathbb{Z}_q^{k-1}, & \text{if } \text{ctr} \neq \text{ctr}^*. \end{cases}$$

Recall that in G_4 , $H_0(vk', m) := [\mathbf{u}_{\text{ctr}}^\top]$ with $\mathbf{u}_{\text{ctr}} \leftarrow_{\$} \mathbb{Z}_q^k$ for the ctr -th hash query $H_0(vk', m)$ for all $\text{ctr} \in [Q_0]$. G_4 and G_5 is indistinguishable due to the $Q_0\text{-}\mathcal{U}_{k,k-1}\text{-MDDH}$ assumption.

Claim 7. $|\Pr_5[\text{Win}] - \Pr_4[\text{Win}]| \leq \text{Adv}_{\mathbb{G}, \mathcal{B}'_2}^{Q_0\text{-}\mathcal{U}_{k,k-1}\text{-MDDH}}(\kappa)$.

Proof. To prove Claim 7, we construct a PPT algorithm $\mathcal{B}'_2$ against the $Q_0\text{-MDDH}$ assumption. Given the challenge $([\mathbf{B}], [\mathbf{C}] = [\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{Q_0}])$. $\mathcal{B}'_2$ has to tell whether $[\mathbf{C}] = [\mathbf{B}\mathbf{V}] = [\mathbf{B}\mathbf{v}_1, \mathbf{B}\mathbf{v}_2, \dots, \mathbf{B}\mathbf{v}_{Q_0}]$ or $[\mathbf{C}] = [\mathbf{U}] = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{Q_0}]$.

Now $\mathcal{B}'_2$ simulates the game for $\mathcal{A}$ just like G_5 , except that $\mathcal{B}'_2$ sets $[\overline{\mathbf{A}}] := [\mathbf{B}^\top]$, and answers the ctr -th fresh query (vk', m) to oracle H_0 as follows,

$$\begin{cases} H_0(vk', m) := [\mathbf{u}_{\text{ctr}}^\top] \text{ with } \mathbf{u}_{\text{ctr}} \leftarrow_{\$} \mathbb{Z}_q^k, & \text{if } \text{ctr} = \text{ctr}^* \\ H_0(vk', m) := [\mathbf{c}_{\text{ctr}}^\top], & \text{if } \text{ctr} \neq \text{ctr}^*. \end{cases}$$

$\mathcal{B}'_2$ can perfectly simulate oracles $\text{SIGN}_1, \text{SIGN}_2, \text{CORR}$ and H_1 , since it generates $vk, \{sk_i\}_{i \in [n]}, \{vk_i\}_{i \in [n]}$ by itself.

Finally, $\mathcal{B}'_2$ receives a forgery (m^*, σ^*) from $\mathcal{A}$, and $\mathcal{B}'_2$ outputs 1 iff m^* is fresh, σ^* is valid and **Queried1** $\wedge$ **CorrectY** $\wedge$ **Queried0** $\wedge$ **GoodGuess** occurs.

It is easy to see that $\mathcal{B}'_2$ perfectly simulates G_4 for $\mathcal{A}$ if $[\mathbf{C}] = [\mathbf{U}]$, and $\mathcal{B}'_2$ perfectly simulates G_5 for $\mathcal{A}$ if $[\mathbf{C}] = [\mathbf{B}\mathbf{V}]$. Thus

$$|\Pr_5[\text{Win}] - \Pr_4[\text{Win}]| \leq \text{Adv}_{\mathbb{G}, \mathcal{B}'_2}^{Q_0\text{-}\mathcal{U}_{k,k-1}\text{-MDDH}}(\kappa).$$

□

Therefore, we can obtain that

$$\Pr_4[\text{Win}] \leq \Pr_5[\text{Win}] + \text{Adv}_{\mathbb{G}, \mathcal{B}'_2}^{Q_0 - \mathcal{U}_{k, k-1} - \text{MDDH}}(\kappa).$$

According to the Lemma 1, we have a PPT algorithm $\mathcal{B}_2$ such that

$$\Pr_4[\text{Win}] \leq \Pr_5[\text{Win}] + \text{Adv}_{\mathbb{G}, \mathcal{B}_2}^{\mathcal{U}_{k, k-1} - \text{MDDH}}(\kappa) + \frac{2}{q}. \quad (36)$$

Game $\mathcal{G}_6$. $\mathcal{G}_6$ is the same as $\mathcal{G}_5$, except for the key generation and replies of the first round signing oracle SIGN_1 .

- For the key generation, $\mathcal{G}_6$ chooses a non-zero vector $\mathbf{a}^\perp \in \mathbb{Z}_q^k$, such that $\overline{\mathbf{A}}\mathbf{a}^\perp = \mathbf{0}$. It then samples $\mathbf{x}' \leftarrow_{\$} \mathbb{Z}_q^k$ and $\mu \leftarrow_{\$} \mathbb{Z}_q$. Next, $\mathcal{G}_6$ invoke $\{\mathbf{x}'_i\}_{i \in [n]} \leftarrow \text{Share}_{(t,n)}(\mathbf{x}')$, $\{\mu_i\}_{i \in [n]} \leftarrow \text{Share}_{(t,n)}(\mu)$ and sets $sk_i := \mathbf{x}'_i + \mathbf{a}^\perp \mu_i$. Finally, $\mathcal{G}_6$ sets $vk := [\overline{\mathbf{A}}\mathbf{x}']$ and $vk_i := [\overline{\mathbf{A}}\mathbf{x}'_i]$ for all $i \in [n]$.
- For any query $(\mathcal{SS}, \ell, m)$ to SIGN_1 . If $\ell \in \mathcal{CS}$, SIGN_1 returns $\perp$. Otherwise SIGN_1 sets $\text{num}\ell := \text{num}\ell + 1$, $\text{iid} := \text{num}\ell$ and $\mathcal{Q}_{0.5}[m] := \mathcal{Q}_{0.5}[m] \cup \{\ell\}$. SIGN_1 obtains $[\underline{\mathbf{A}}] := H_0(vk, m)$. Then, SIGN_1 replies the query $(\mathcal{SS}, \ell, m)$ in two cases:
 - If (vk, m) is the ctr^* -th fresh random oracle query to H_0 : SIGN_1 sets $[y_\ell] := [\underline{\mathbf{A}}] \cdot \mathbf{x}_\ell = [\underline{\mathbf{A}}] \cdot (\mathbf{x}'_\ell + \mathbf{a}^\perp \mu_\ell)$, exactly the same as in $\mathcal{G}_5$.
 - If (vk, m) is not the ctr^* -th fresh random oracle query to H_0 : SIGN_1 sets $[y_\ell] := [\underline{\mathbf{A}}] \cdot \mathbf{x}'_\ell$.

After that, SIGN_1 samples $\mathbf{r}_\ell \leftarrow_{\$} \mathbb{Z}_q^k$, and computes $[\mathbf{w}_\ell] := \begin{bmatrix} \underline{\mathbf{A}} \\ \underline{\mathbf{A}} \end{bmatrix} \cdot \mathbf{r}_\ell$. Finally, SIGN_1 returns $(\text{iid}, \text{pm}_{1,\ell} := ([y_\ell], [\mathbf{w}_\ell]))$ and locally stores $\text{st}_{1,\ell} := \mathbf{r}_\ell$.

Next we will show that $\mathcal{G}_6$ is identical to $\mathcal{G}_5$.

Define $\mathbf{x} := \mathbf{x}' + \mathbf{a}^\perp \mu$ and $\mathbf{x}_i := \mathbf{x}'_i + \mathbf{a}^\perp \mu_i$ for all $i \in [n]$. Then $\mathbf{x}$ is uniform over $\mathbb{Z}_q^k$ due to $\mathbf{x}' \leftarrow_{\$} \mathbb{Z}_q^k$, just like $\mathcal{G}_5$. Due to the linearity of $\text{Share}_{(t,n)}$, we know that $\mathbf{x}_i := \mathbf{x}'_i + \mathbf{a}^\perp \mu_i$ is an output of $\text{Share}_{(t,n)}(\mathbf{x})$, so $\{sk_i\}_{i \in [n]}$ in $\mathcal{G}_6$ has the same distribution as $\mathcal{G}_5$. Consequently, $vk, \{vk_i\}_{i \in [n]}$ has the same distribution as well.

Let us consider $\text{SIGN}_1(\mathcal{SS}, \ell, m)$. In the computation of $[y_\ell]$, SIGN_1 first obtains $[\underline{\mathbf{A}}] := H_0(vk, m)$. Suppose that (vk, m) is the ctr -th fresh hash query to H_0 . There are two cases:

Case $\text{ctr} = \text{ctr}^*$: In this case, the computation of $[y_\ell] := [\underline{\mathbf{A}}] \cdot \mathbf{x}_\ell$ is exactly the same as in $\mathcal{G}_5$.

Case $\text{ctr} \neq \text{ctr}^*$: In this case, $[\underline{\mathbf{A}}] := [\mathbf{v}^\top \overline{\mathbf{A}}]$ for some $\mathbf{v} \in \mathbb{Z}_q^{k-1}$ (due to $\mathcal{G}_5$). Hence, we always have

$$[y_\ell] := [\underline{\mathbf{A}}] \cdot \mathbf{x}'_\ell = [\mathbf{v}^\top \overline{\mathbf{A}}] \cdot (\mathbf{x}_\ell - \mathbf{a}^\perp \mu_\ell) = [\mathbf{v}^\top \overline{\mathbf{A}}] \cdot \mathbf{x}_\ell = [\underline{\mathbf{A}}] \cdot \mathbf{x}_\ell.$$

which is identical to $[y_\ell]$ in $\mathcal{G}_5$.

Therefore, G_6 is identical to G_5 and hence

$$\Pr_5[\text{Win}] = \Pr_6[\text{Win}]. \quad (37)$$

Game G_7 . G_7 is the same as G_6 , except for the replies of the second round signing oracle SIGN_2 and the corrupt oracle CORR .

- For any query $(\ell, \text{iid}, \{\text{pm}_{1,i}\}_{i \in \text{SS}})$ for SIGN_2 , SIGN_2 combines the first round message $\{\text{pm}_{1,i}\}_{i \in \text{SS}}$ to get $[y]$ and $\mathbf{w}$. Then, it computes $c := H_1(vk, m, [y], [\mathbf{w}])$ just like G_6 . Next, SIGN_2 replies the query $(\ell, \text{iid}, \{\text{pm}_{1,i}\}_{i \in \text{SS}})$ in two cases.
Case I: (vk, m) is the ctr^* -th fresh random oracle query to H_0 . SIGN_2 sets $\mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{\text{SS},\ell} \cdot \mathbf{x}_\ell = \mathbf{r}_\ell + c \cdot \lambda_{\text{SS},\ell} \cdot (\mathbf{x}'_\ell + \mathbf{a}^\perp \mu_\ell)$, as in G_6 .
Case II: Otherwise, SIGN_2 sets $\mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{\text{SS},\ell} \cdot \mathbf{x}'_\ell$.
Then SIGN_2 returns $\text{pm}_{2,\ell} := \mathbf{z}_\ell$ and sets $\text{state}[\ell, \text{iid}] := \text{st}_{2,\ell} := \text{st}_{1,\ell}$.
– For any corrupt query ℓ , oracle CORR updates all internal states $\text{state}_{\ell, \text{iid}}$ for $\text{iid} \in [\text{num}\ell]$. For each $\text{iid} \in [\text{num}\ell]$, CORR retrieves $\mathcal{I}[\ell, \text{iid}] = (\text{SS}, m)$. If (vk, m) is not the ctr^* -th fresh query to H_0 , then it retrieves $\text{state}_{\ell, \text{iid}} = \mathbf{r}_\ell$, computes $\mathbf{r}'_\ell := \mathbf{r}_\ell - c \cdot \lambda_{\text{SS},\ell} \mathbf{a}^\perp \mu_\ell$ and updates $\text{state}_{\ell, \text{iid}} := \mathbf{r}'_\ell$. Finally CORR returns signer ℓ 's secret key sk_ℓ and its updated internal states $\{\text{state}_{\ell, \text{iid}}\}_{\text{iid} \in [\text{num}\ell]}$.

The difference between G_7 and G_6 lies in the computation of $\mathbf{z}_\ell$ in Case II by SIGN_2 and the later reveal $\mathbf{r}'_\ell$ of the state $\text{state}[\ell, \text{iid}]$ by CORR .

Recall that in Case II, (vk, m) is not the ctr^* -th fresh random oracle query to H_0 , so $[\underline{\mathbf{A}}] := H_0(vk, m) := [\mathbf{v}^\top \mathbf{A}]$ for some $\mathbf{v} \in \mathbb{Z}_q^{k-1}$. Accordingly,

- Given $\mathbf{r}'_\ell := \mathbf{r}_\ell - c \cdot \lambda_{\text{SS},\ell} \mathbf{a}^\perp \mu_\ell$, $\mathbf{r}'_\ell$ is uniformly distributed since $\mathbf{r}_\ell$ is uniformly sampled in SIGN_1 .
- In SIGN_2 , $\mathbf{z}_\ell = \mathbf{r}_\ell + c \cdot \lambda_{\text{SS},\ell} \cdot \mathbf{x}'_\ell = \mathbf{r}'_\ell + c \cdot \lambda_{\text{SS},\ell} \mathbf{a}^\perp \mu_\ell + c \cdot \lambda_{\text{SS},\ell} \cdot \mathbf{x}'_\ell = \mathbf{r}'_\ell + c \cdot \lambda_{\text{SS},\ell} \cdot \mathbf{x}_\ell$. Thus the computation of $\mathbf{z}_\ell$ in G_7 is the same as G_6 .
- In the later corruption, CORR reveals $\mathbf{r}'_\ell = \mathbf{r}_\ell - c \cdot \lambda_{\text{SS},\ell} \mathbf{a}^\perp \mu_\ell$ to $\mathcal{A}$, which is consistent to SIGN_2 in G_7 . Note that in SIGN_1 , $[\mathbf{w}_\ell] := [\mathbf{A}] \cdot \mathbf{r}_\ell = \begin{bmatrix} \overline{\mathbf{A}} \\ \mathbf{v}^\top \overline{\mathbf{A}} \end{bmatrix}$.
 $(\mathbf{r}'_\ell + c \cdot \lambda_{\text{SS},\ell} \mathbf{a}^\perp \mu_\ell) = \begin{bmatrix} \overline{\mathbf{A}} \\ \mathbf{v}^\top \overline{\mathbf{A}} \end{bmatrix} \cdot \mathbf{r}'_\ell = [\mathbf{A}] \cdot \mathbf{r}'_\ell$, so $\mathbf{r}'_\ell$ is also consistent to SIGN_1 .

Therefore, G_7 is identical to G_6 and we obtain

$$\Pr_6[\text{Win}] = \Pr_7[\text{Win}]. \quad (38)$$

Claim 8. $\Pr_7[\text{Win}] \leq \frac{2}{q}$.

Proof. Recall that Win in G_7 implies **CorrectY** and **GoodGuess**. Thus it must hold that the ctr^* -th fresh hash query (vk', m) to H_0 satisfies $vk' = vk \wedge m = m^*$.

If **GoodGuess** occurs in G_7 , then $[\underline{\mathbf{A}}^*] := H_0(vk, m^*) \leftarrow_s \mathbb{G}^{1 \times k}$. Now we show that μ remains perfectly hidden to $\mathcal{A}$ in this case.

- $pp := (\mathbb{G}, g, q, [\bar{\mathbf{A}}]), vk := [\bar{\mathbf{A}}\mathbf{x}']$ and $vk_i := [\bar{\mathbf{A}}\mathbf{x}'_i]$ for all $i \in [n]$.
Thus $\mathcal{A}$ obtains no information of μ from $pp, vk, \{vk_i\}_{i \in [n]}$.
- Upon a signing query $(\mathcal{SS}, \ell, m)$, oracle SIGN_1 will return $(\text{iid}, ([y_\ell], [\mathbf{w}_\ell]))$.
Upon a signing query $(\ell, \text{iid}, \{\mathbf{pm}_{1,i}\}_{i \in \mathcal{SS}})$, oracle SIGN_2 will return $\mathbf{z}_\ell$. After SIGN_1 's reply, there must exist $(\text{ctr}, vk, m, \cdot) \in \text{List}_0$ since SIGN_1 must have issued a hash query to H_0 . We distinguish two cases:

Case $\text{ctr} \neq \text{ctr}^*$. In this case, we have $[y_\ell] := [\underline{\mathbf{A}}] \cdot \mathbf{x}'_\ell$, $[\mathbf{w}_\ell] := \begin{bmatrix} \underline{\mathbf{A}} \\ \underline{\mathbf{A}} \end{bmatrix} \cdot \mathbf{r}_\ell$, and

$$\mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{\mathcal{SS}, \ell} \cdot \mathbf{x}'_\ell.$$

Thus $\mathcal{A}$ obtains no information of μ from $\text{SIGN}_1, \text{SIGN}_2$ in this case.

Case $\text{ctr} = \text{ctr}^*$. In this case, $\mathcal{Q}_{0.5}[m^*] := \mathcal{Q}_{0.5}[m^*] \cup \{\ell\}$ by SIGN_1 , and we

$$\text{have } [y_\ell] := [\underline{\mathbf{A}}] \cdot (\mathbf{x}'_\ell + \mathbf{a}^\perp \mu_\ell), [\mathbf{w}_\ell] := \begin{bmatrix} \underline{\mathbf{A}} \\ \underline{\mathbf{A}} \end{bmatrix} \cdot \mathbf{r}_\ell, \text{ and } \mathbf{z}_\ell := \mathbf{r}_\ell + c \cdot \lambda_{\mathcal{SS}, \ell} \cdot (\mathbf{x}'_\ell + \mathbf{a}^\perp \mu_\ell).$$

Thus $\mathcal{A}$ learns at most the information of μ_ℓ from $\text{SIGN}_1, \text{SIGN}_2$ in this case.

- Upon a corruption query ℓ , oracle CORR sets $\mathcal{CS} := \mathcal{CS} \cup \{\ell\}$ and returns $(sk_\ell, \{\text{state}_{\ell,i}\}_{i \in [\text{num}_\ell]})$, where $sk_\ell = \mathbf{x}'_\ell + \mathbf{a}^\perp \mu_\ell$ and $\text{state}_{\ell,i} = \mathbf{r}'_\ell = \mathbf{r}_\ell - c \cdot \lambda_{\mathcal{SS}, \ell} \mathbf{a}^\perp \mu_\ell$ or $\text{state}_{\ell,i} = \mathbf{r}_\ell$. In this case, $\mathcal{A}$ learns at most the information of one share μ_ℓ from CORR .

If **Win** occurs, it must hold that $|\mathcal{Q}_{0.5}[m^*] \cup \mathcal{CS}| < t$. Therefore $\mathcal{A}$ learns less than t shares of μ in $\mathbf{G}_7$. By the perfect secrecy of the secret sharing scheme $\mathcal{SS}$, the leakage of fewer than t shares $\{\mu_\ell\}$ reveals no information about the underlying secret μ . Therefore, μ remains perfectly hidden from $\mathcal{A}$ in $\mathbf{G}_7$.

If **Win** occurs, then **CorrectY** must happen, so $\mathcal{A}$'s forgery $(m^*, \sigma^* = ([y^*], \mathbf{z}^*, c^*))$ satisfies

$$[y^*] = [\underline{\mathbf{A}}^*] \cdot \mathbf{x} = [\underline{\mathbf{A}}^*] \cdot (\mathbf{x}' + \mathbf{a}^\perp \mu) = [\underline{\mathbf{A}}^*] \cdot \mathbf{x}' + [\underline{\mathbf{A}}^*] \cdot \mathbf{a}^\perp \mu. \quad (39)$$

Recall that $[\underline{\mathbf{A}}^*] := H_0(vk, m^*) \leftarrow_s \mathbb{G}^{1 \times k}$. We can see that $[\underline{\mathbf{A}}^*]^\top \notin \text{Span}([\bar{\mathbf{A}}]^\top)$ except with probability at most $1/q$. Thus $[\underline{\mathbf{A}}^*] \cdot \mathbf{a}^\perp \neq \mathbf{0}$ except with probability at most $1/q$. In this case, the uniformity of μ makes sure that $[\underline{\mathbf{A}}^*] \cdot \mathbf{a}^\perp \mu$ is uniformly distributed over $\mathbb{G}$. As a result, (39) holds with probability at most $1/q$. Then we have

$$\Pr_7[\text{Win}] \leq \Pr_7[\text{CorrectY}] \leq \frac{2}{q}.$$

□

The proof finishes by combining (23), (24), (25), (34), (35), (36), (37), (38) and Claim 8. □