Signatures with Tight Adaptive Corruptions from Search Assumptions*

Keitaro Hashimoto¹, Wakaha Ogata², and Yusuke Sakai¹

National Institute of Advanced Industrial Science and Technology (AIST) {keitaro.hashimoto, yusuke.sakai}@aist.go.jp

² Institute of Science Tokyo ogata.w.aa@m.titech.ac.jp

Abstract. We construct the *first* tightly secure signature schemes in the multi-user setting with adaptive corruptions from classical discrete logarithm, RSA, factoring, or post-quantum group action discrete logarithm assumption. In contrast to our scheme, the previous tightly secure schemes are based on the decisional assumption (e.g., (group action) DDH) or interactive search assumptions (e.g., one-more CDH). The security of our schemes is independent of the number of users, signing queries, and RO queries, and forging our signatures is as hard as solving the underlying search problem. Our starting point is an identification scheme with multiple secret keys per public key (e.g., Okamoto identification (CRYPTO'92) and parallel-OR identification (CRYPTO'94)). This property allows a reduction to solve a search problem while answering corruption queries for all users in the signature security game. To convert such an identification scheme into a signature scheme tightly, we employ randomized Fischlin's transformation introduced by Kondi and shelat (Asiacrypt 2022) that provides straight-line extraction. Intuitively, the transformation properties guarantee the tight security of our signature scheme in the programmable random oracle model, but we successfully prove its tight security in the non-programmable random oracle model.

Keywords: Digital signature, Multi-user setting with corruption, Tight security

1 Introduction

1.1 Background

Digital signature. Signature schemes are fundamental cryptographic tools to authenticate the sender of messages. The basic security notion for signature schemes is existential unforgeability against chosen message attacks (UF-CMA) [14]. It guarantees that an efficient adversary, given a single verification key, cannot forge a valid signature for any new message under that key. The

^{*} Author list in alphabetical order.

UF-CMA security considers a single-user setting, but in real-world scenarios, multiple users possess individual signing keys derived from a common public parameter, and some of these keys may occasionally be compromised. To address a more realistic context, Bader et al. [3] defined the notion of UF-CMA in the multi-user setting with adaptive corruption (MU-UF-CMA-C). This framework extends UF-CMA by allowing adversaries to obtain multiple users' verification keys and adaptively corrupt users to access their secret signing keys. Notably, MU-UF-CMA-C security precisely captures the requirements of practical applications that rely on digital signatures, such as authenticated key exchange [3] and identity-based signatures [20].

Tight Security. To prove the security of cryptographic primitives, we generally construct a reduction algorithm, which transforms an efficient adversary who breaks the security of the scheme (e.g., EUF-CMA security of signature scheme) into an algorithm that solves an assumed-to-be-hard computational problem (e.g., discrete logarithm problem). In general, there is a gap between the success probability of the reduction and that of the adversary. This gap is called reduction loss, representing the theoretical difference between the hardness of breaking the scheme's security and the hardness of solving the computational problem. If the reduction has the same success probability as the adversary, we say that the reduction is tight and the scheme is tightly secure. This means the security of the primitive is independent of other factors, such as the number of users or hash function evaluations. Showing tight security is important in both theory and practice. Tight reduction connects the hardness of breaking the security to the hardness of computational problems. Since the hardness of computational problems is well-studied, we can easily understand the security level of the tightly secure schemes. In addition, tightly secure schemes allow for optimal parameter selection that meets the desired security level based on cryptoanalysis against computational problems. As a result, data size (e.g., key, ciphertext, signature size) and computation cost (e.g., signature generation and verification) are reduced compared to non-tight schemes with large parameter sets considering reduction loss.

Tight MU-UF-CMA-C Secure Signature. Constructing MU-UF-CMA-C secure signature schemes is a critical goal in cryptography. It is known that EU-CMA security implies MU-UF-CMA-C security, albeit with a security loss proportional to the number of users. In other words, these signature schemes require a larger parameter to be selected to satisfy sufficient security level to account for security loss. To mitigate this loss, researchers have focused on developing tightly MU-UF-CMA-C-secure signature schemes, where security remains independent of the number of users and other factors such as the number of signing and random oracle queries [2,3,13,27,9]. Table 1 summarizes the existing works. These known schemes are based on decisional assumptions (e.g., (group action) DDH, DLIN, SXDH, φ-hiding) or interactive search assumptions (e.g., one-more CDH), and no tightly MU-UF-CMA-C-secure signature scheme has been constructed under static search assumptions (e.g., CDH or DL).

Table 1: Existing tightly secure signatures in the multi-user setting with corruptions and our result. The column "Settings" indicates whether pairings/the Programmable Random Oracle (PRO)/the Non-Programmable Random Oracle (NPRO) is used. The column "SUF" indicates whether the scheme is proven strongly unforgeable. λ denotes the security parameter, t denotes the bit-length of the challenge space, and ρ denotes the number of repetitions, which satisfy $\rho t = \omega(\lambda)$. |X| denotes the bit-length of the elements in the set X.

(a) Classical group-based schemes. Let \mathbb{G} be a multiplicative group with order q.

Scheme	Public key	Signature	Assumptions	Settings	SUF?
Bader [2]	G	$6 \mathbb{G} $	SXDH	Pairing, PRO	
BHJKL [3]	$\mathcal{O}(\lambda) \mathbb{G} $	$\mathcal{O}(\lambda) \mathbb{G} $	DLIN	Pairing	
GJ [13]	$2 \mathbb{G} $	$2 \mathbb{G} + 4 \mathbb{Z}_q + 2\lambda$	CDH & DDH	PRO	
WLGSZ [27]	G	$2 \mathbb{G} +1$	OM-CDH	Pairing, PRO	
DGJL [9, Sec. 5.1]	4 G	$3 \mathbb{Z}_q $	DDH	NPRO	✓
Ours (based on [21])	G	$\rho(2 \mathbb{Z}_q +t)$	DL	NPRO	√

(b) Classical factoring or RSA-based schemes. Let N be an integer such that N=pq for some primes p and q.

Scheme	Public key	Signature	Assumption	s Settings	SUF?
DGJL [9, Sec. 5.2]	$2 \mathbb{Z}_N $	$2 \mathbb{Z}_N + \lambda/4$	ϕ -hiding	NPRO	✓
Ours (based on [15])	$ \mathbb{Z}_N $	$\rho(2 \mathbb{Z}_N +t)$	RSA	NPRO	√
Ours (based on [12])	$ \mathbb{Z}_N $	$\rho(2 \mathbb{Z}_N + 2t + poly(\lambda))$	FACT	NPRO	✓

(c) Group action-based schemes. Let \mathcal{G} be a group that acts on a set \mathcal{E} (i.e., there exists a group action $\star: \mathcal{G} \times \mathcal{E} \to \mathcal{E}$).

Scheme	Public key	Signature	Assumptions	s Settings	SUF?
PW [24, Sec. 4.2]	$4 \mathcal{E} $	$2\lambda(2 \mathcal{E} + \mathcal{G})$	GADDH	PRO	
Ours (based on $[7,5,26]$)	$2 \mathcal{E} $	$2\rho t(\mathcal{G} +1)$	GADL	NPRO	√

At the same time, several impossibility results regarding tightly MU-UF-CMA-C-secure signatures have been established [4,24,28]. These results identify specific conditions under which tightly MU-UF-CMA-C-secure signatures cannot exist. However, they do not rule out the possibility of constructing such schemes under static search assumptions.

This literature leads to the following research question:

Can we construct a tightly MU-UF-CMA-C secure signature from static search assumptions?³

³ [23,22] left the same question as an open problem.

1.2 Our Contributions

We provide an affirmative answer to the open question above. Specifically, we construct the *first* signature scheme achieving tight (strong) MU-UF-CMA-C security⁴ under static search assumptions such as (group action) discrete logarithm, RSA, or factoring assumptions. The security of our schemes is independent of the number of users, signing queries, and random oracle (RO) queries, and forging a signature is provably as hard as solving static search problems.

Our approach begins with a specialized identification scheme that allows multiple secret keys per public key, such as the Okamoto identification scheme [21] and parallel-OR identification scheme [8]. This unique property enables a reduction, which solves a search problem, to address corruption queries for all users in the MU-UF-CMA-C security game. To tightly convert an identification protocol into a signature scheme, we utilize a randomized variant of Fischlin's transformation [11], introduced by Kondi and shelat [17]. This transformation features straight-line extraction, meaning that the reduction can extract a secret key corresponding to a given public key without rewinding the adversary. While the transformation guarantees tight security in the programmable random oracle model, we advance this result by proving the tight security of our signature scheme in the non-programmable random oracle model.

By instantiating our framework appropriately, we obtain tightly MU-UF-CMA-C secure signature schemes based on classical DL, RSA, or factoring assumptions and post-quantum group action DL (GADL) assumption, as shown in Table 1. It is worth noting that we obtain the first tightly MU-UF-CMA-C secure signature from computational assumptions that are not random self reducible (e.g., RSA assumption). Further, we obtain a GADL-based scheme with a shorter public key and signature than the existing group action DDH (GADDH)-based scheme by Pan and Wagner [24].

Zero-Knowledge of Randomized Fischlin, Recosndiered. As a side contribution, we point out a flaw in the proof for the zero-knowledge property of randomized Fischlin transformation by Kondi and shelat [18, Proof of Theorem 6.4]. They claimed randomized Fischlin transformation is unconditionally zero-knowledge in contrast to the original Fischlin transformation [11]. We notice that the transcripts generated by their ZK simulator are not necessarily indistinguishable from real ones against unconditional adversaries. We employ the strong special soundness of the underlying interactive protocol, which is a newly introduced soundness property by Kondi and shelat, to fix the flaw in the proof. This means the zero-knowledge property of randomized Fischlin transformation is computational if the strong special soundness is computational. The details are in Appendix C.

⁴ As mentioned in [9], some applications require strong unforgeability. So, we also target it.

⁵ Although the definition of strong special soundness [18, Definition 3.2] does not seem to take computational assumptions into account, the authors are probably considering the computational one. See Remark 1.

2 Preliminaries

2.1 Notations

 $\lambda \in \mathbb{N}$ denotes a security parameter. \oplus denotes bit-wise exclusive-or operation. $\operatorname{poly}(\cdot)$ and $\operatorname{negl}(\cdot)$ are any polynomial function and negligible function, respectively. e denotes the base of the natural logarithm (i.e., Napier's constant). For $n \in \mathbb{N}$, we define $[n] \coloneqq \{1, 2, \dots, n\}$ as the set of the first n natural numbers. For a finite set S, we use $s \leftarrow S$ to denote the uniformly random sampling of an element s from S. A probabilistic algorithm A is said to be PPT(probabilistic polynomial time) if its running time T_A can be bounded by a polynomial in its input size. The notation $y \leftarrow A(x)$ means that the variable y is assigned to the output of the algorithm A on input x. We write $y \in A(x)$ to state that y is a possible output of A on input A or input A on input A or input

2.2 Signature Schemes

We recall the syntax and security notions of signature schemes.

Definition 1 (Signature scheme). A signature scheme SIG is a tuple of the following algorithms.

- Setup(1^{λ}) \rightarrow par: On input the security parameter 1^{λ} , the setup algorithm outputs a public parameter par. We assume the following algorithms implicitly take par as input.
- KGen(par) → (svk, ssk): On input a public parameter par, the key generation algorithm outputs a public key svk and a secret key ssk.
- Sign(ssk, m) $\rightarrow \sigma$: On input a secret key ssk and a message m, the signing algorithm outputs a signature σ .
- $Vf(svk, m, \sigma) \rightarrow 1/0$: On input a public key svk, a message m, and a signature σ , the verification algorithm outputs 0 or 1.

Definition 2 (Correctness). We say that a signature scheme SIG is $(1 - \beta)$ -correct if for any $\lambda \in \mathbb{N}$, any par \in Setup (1^{λ}) , any key pair (svk, ssk) \leftarrow KGen(par), and any message m, it holds that

$$\Pr[\mathsf{Vf}(\mathsf{svk}, m, \mathsf{Sign}(\mathsf{ssk}, m)) = 1] \ge 1 - \beta.$$

In this work, we focus on strongly existential unforgeability against adaptive chosen-message attacks in the multi-user setting with adaptive corruptions [9]. As mentioned in [9], strong unforgeability is useful for constructing authenticated key exchange protocols. Thus, we consider it the notion of target security. We call it MU-SUF-CMA-C security in short.

```
N-MU-SUF-CMA-C(\lambda)
                                                                \mathsf{OSign}(i, m)
 1: L_{\text{corr}}, L_{\text{sig}} \leftarrow \emptyset
                                                                 1: if i \in L_{corr} then
 2: par \leftarrow Setup(1^{\lambda})
                                                                             return \perp
                                                                 3: \quad \sigma \leftarrow \mathsf{Sign}(\mathsf{ssk}_i, m)
 3: for each i \in [N] do
                                                                 4: L_{\text{sig}} \coloneqq L_{\text{sig}} \cup \{(i, m, \sigma)\}
             (\mathsf{svk}_i, \mathsf{ssk}_i) \leftarrow \mathsf{KGen}(\mathsf{par})
 4:
                                                                 5: return \sigma
 5: O := (OSign, OCorr)
6: \mathsf{SVK} \coloneqq \{\mathsf{svk}_i\}_{i \in [N]}
                                                                OCorr(i)
7: (i^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{O}}(\mathsf{par}, \mathsf{SVK})
                                                                 1: L_{corr} := L_{corr} \cup \{i\}
8: if i^* \in L_{corr} then return 0
                                                                 2: return ssk,
9: if (i^*, m^*, \sigma^*) \in L_{\text{sig}} then
10:
             return 0
11: ok := \mathsf{Vf}(\mathsf{svk}_{i^*}, m^*, \sigma^*)
12: return ok
```

Fig. 1: Security game for signature scheme.

Definition 3 (N-MU-SUF-CMA-C **Security**). Let $N = \text{poly}(\lambda)$ be a natural number. We say a signature scheme SIG is N-MU-SUF-CMA-C secure if for any PPT adversary \mathcal{A} , it holds that

$$\begin{aligned} \mathsf{Adv}^{\mathsf{MU-SUF-CMA-C}}_{\mathcal{A},\mathsf{SIG}}(\lambda) &\coloneqq \Pr[N\text{-MU-SUF-CMA-C}(\lambda) \Rightarrow 1] \\ &\leq \mathsf{negl}(\lambda), \end{aligned}$$

where the game N-MU-SUF-CMA-C is depicted in Figure 1.

2.3 Canonical Identification Schemes

We follow the syntax of canonical identification schemes in [16].

Definition 4 (Canonical Identification Schemes). A canonical identification scheme ID is a tuple of the following four algorithms.

- $\mathsf{ISetup}(1^\lambda) \to \mathsf{par}$: The setup algorithm takes the security parameter 1^λ and outputs a public parameter par . We assume that par defines the set of challenges ChSet and the following algorithms implicitly take par as input.
- IGen(par) → (pk, sk): The key generation algorithm takes a public parameter par as input and outputs a public and secret key (pk, sk). We assume the secret key sk is chosen uniformly and randomly from the secret key space.
- $-P = (P_1, P_2)$: The prover algorithm is split into two algorithms. P_1 takes as input the key pair (pk, sk) and returns a commitment com and a state st; P_2 takes as input a challenge ch and a state st, and returns a response resp.

- $V(pk, com, ch, resp) \rightarrow 1/0$: The verifier algorithm takes the public key pk and the conversation transcript (com, ch, resp) as input and outputs 1 or 0.

Let $\mathcal{SK}(\mathsf{pk}) \coloneqq \{\mathsf{sk} : (\mathsf{pk}, \mathsf{sk}) \in \mathsf{IGen}(\mathsf{par})\}$ denote the set of all valid secret keys w.r.t. a given public key pk and let $K \coloneqq |\mathcal{SK}(\mathsf{pk})|$. We say that an identification scheme ID has K-multiple secret keys if each pk has K secret keys. When K = 1, we say that ID has a single secret key.

We require that identification schemes ID satisfy the following properties.

Definition 5 (Correctness). We say that ID is correct if for all $\lambda \in \mathbb{N}$, all par \in ISetup(1^{λ}), all (pk, sk) \in IGen(par), all (com, st) \in P₁(pk, sk), all ch \in ChSet and all resp \in P₂(ch, st), we have V(pk, com, ch, resp) = 1.

We say a transcript (com, ch, resp) is valid w.r.t. pk if V(pk, com, ch, resp) = 1.

Definition 6 (Key Verifiability [24]). We say that ID is key verifiable if there exists a deterministic polynomial time algorithm VerKey such that for all $\lambda \in \mathbb{N}$, all par $\in \mathsf{ISetup}(1^{\lambda})$ and any $(\mathsf{pk}, \mathsf{sk})$,

$$VerKey(par, pk, sk) = 1 \iff (pk, sk) \in IGen(par).$$

Definition 7 (Min-Entropy of Commitments [16]). We say that ID has κ -bits of commitment min-entropy, if for all (pk, sk) \in IGen(par), the commitment generated by the prover algorithm is chosen from a distribution with at least κ -bits of commitment min-entropy. That is, for all strings com', we have $\Pr[\mathsf{com'} = \mathsf{com}] \leq 2^{-\kappa}$ if (com, *) $\leftarrow \mathsf{P}_1(\mathsf{pk}, \mathsf{sk})$ was honestly generated by the prover.

Given ID as above, we define transcript generation algorithm Tran as follows:

Definition 8 (Special Honest-Verifier Zero-Knowledge, HVZK [24]). We say that ID is ϵ_{ZK} -special honest-verifier zero-knowledge (ϵ_{ZK} -HVZK) if there exists a PPT algorithm Sim, a simulator, such that for all par \in ISetup(1^{λ}) and all (pk, sk) \in IGen(par), the following distributions have statistical distance at most ϵ_{ZK} :

$$\{(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow \mathsf{Tran}(\mathsf{pk}, \mathsf{sk}, \mathsf{ch}) | \mathsf{ch} \leftarrow \mathsf{\$} \mathsf{ChSet}\}$$

and

$$\{(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) | \mathsf{ch} \leftarrow \mathsf{\$} \mathsf{ChSet}; (\mathsf{com}, \mathsf{resp}) \leftarrow \mathsf{Sim}(\mathsf{pk}, \mathsf{ch}) \}.$$

If $\epsilon_{ZK} = 0$, we say ID is perfect HVZK.

In this work, we define strong special soundness, which was introduced by Kondi and shelat [17]. This is a stronger version of special soundness in the sense that the extractor can extract a secret key of pk from two valid transcripts (com, ch, resp) and (com, ch', resp') such that (ch, resp) \neq (ch', resp'). That is, the extractor works even in the case ch = ch' and $resp \neq resp'$. In this work, we define computational strong special soundness.

Definition 9 (Computational Strong Special Soundness). We say that ID is strong special sound if there exists a PPT algorithm Ext, an extractor, such that for all PPT adversaries \mathcal{A} , it holds that

$$\begin{split} \mathsf{Adv}_{\mathcal{A},\mathsf{ID}}^{\mathsf{SSS}}(\lambda) \\ &:= \Pr \begin{bmatrix} (\mathsf{ch},\mathsf{resp}) \neq (\mathsf{ch}',\mathsf{resp}') & \mathsf{par} \leftarrow \mathsf{ISetup}(1^{\lambda}), \\ (\mathsf{pk},\mathsf{com},\mathsf{ch},\mathsf{resp},\mathsf{ch}',\mathsf{resp}') \leftarrow \mathcal{A}(\mathsf{par}), \\ \wedge ok = ok' = 1 \\ \wedge (\mathsf{pk},\mathsf{sk}^*) \notin \mathsf{IGen}(\mathsf{par}) & ok \leftarrow \mathsf{V}(\mathsf{pk},\mathsf{com},\mathsf{ch},\mathsf{resp}'), \\ ok' \leftarrow \mathsf{V}(\mathsf{pk},\mathsf{com},\mathsf{ch}',\mathsf{resp}'), \\ \mathsf{sk}^* \leftarrow \mathsf{Ext}(\mathsf{pk},\mathsf{com},\mathsf{ch},\mathsf{resp},\mathsf{ch}',\mathsf{resp}') \end{bmatrix} \\ \leq \mathsf{negl}(\lambda). \end{split}$$

Remark 1. Kondi and shelat introduced strong special soundness to relax the requirement, "special soundness plus quasi-unique response" required for the underlying protocol in the original Fischlin transformation [11]. The original definition of strong special soundness does not specify "who" generates two transcripts, does not take into account the failure probability of Ext, and leaves no room for computational assumptions. However, it is clear that special soundness plus quasi-unique response does not imply such unconditional strong special soundness. Since they said "Okamoto's identification protocol satisfies strong special soundness," they are probably considering the computational one defined above.

We also introduce a variant of key recovery resistance. The following second key recovery resistance ensures that when ID has multiple secret keys, given a key pair (pk, sk), it is difficult to find another secret key $sk^* \neq sk$ with respect to pk. For our purpose of constructing a tightly secure signature in the multi-user setting, we define second key recovery resistance in the multi-user setting.

Definition 10 (Second Key Recovery in the Multi-User Setting). Let $N = \text{poly}(\lambda)$ be some natural number. We say that ID is second key recovery resistant in the multi-user setting if for all adversaries A, it holds that

$$\begin{split} \mathsf{Adv}^{2^{\mathsf{nd}}\mathsf{KR}}_{\mathcal{A},\mathsf{ID}}(\lambda) &\coloneqq \Pr \begin{bmatrix} (\mathsf{pk}_{i^*},\mathsf{sk}^*) \in \mathsf{IGen}(\mathsf{par}) & \mathsf{par} \leftarrow \mathsf{ISetup}(1^{\lambda}), \\ \wedge \mathsf{sk}^* \neq \mathsf{sk}_{i^*} & (\mathsf{pk}_i,\mathsf{sk}_i) \leftarrow \mathsf{IGen}(\mathsf{par}) \ \forall i \in [N], \\ (i^*,\mathsf{sk}^*) \leftarrow \mathcal{A}(\mathsf{par},\{(\mathsf{pk}_i,\mathsf{sk}_i)\}_{i \in [N]}) \end{bmatrix} \\ &\leq \mathsf{negl}(\lambda). \end{split}$$

3 Signatures from Identification Scheme via Randomized Fischlin Transformation

In this section, we describe the signature scheme from an identification scheme via randomized Fischlin transformation [17]. Then, we prove that the signature scheme tightly archives MU-SUF-CMA-C security.

Let $\mathsf{ID} = (\mathsf{ISetup}, \mathsf{IGen}, \mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V})$ be a canonical identification scheme with the challenge space ChSet . Define the parameters t, ρ, γ for the bit-length of the challenges, the number of repetitions, and the length of the hash value such that $\rho \cdot \gamma = \omega(\lambda), \ t - \gamma = \omega(\lambda), \ t, \rho, \gamma = \mathcal{O}(\lambda)$ and $\gamma \leq t \leq \lfloor \log |\mathsf{ChSet}| \rfloor$. Also, let T be the maximum number of retrying the signing algorithm. Let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\gamma$ be a random oracle. The signature scheme $\mathsf{SIG}[\mathsf{ID}] \coloneqq (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}^\mathsf{H}, \mathsf{Vf}^\mathsf{H})$ (in the random oracle model) is depicted in Figure 2.

We prove that SIG[ID] is correct and MU-SUF-CMA-C secure.

Theorem 1. If ID is correct, then the signature scheme SIG[ID] is $(1-\beta)$ -correct for $\beta = 2^{(-2^{t-\gamma}\log e + \log \rho)T}$.

Proof. Randomization does not affect correctness errors, so we can refer to the existing correctness analysis for the original Fischlin transformation. According to [6, Section 3], the probability that the Sign algorithm outputs \bot is at most $(\rho \cdot e^{-2^{t-\gamma}})^T = 2^{(-2^{t-\gamma}\log e + \log \rho)T}$. Since ID is perfectly correct, SIG[ID] is $(1-\beta)$ -correct for $\beta = 2^{(-2^{t-\gamma}\log e + \log \rho)T}$.

Theorem 2. If ID has κ -bits of commitment min-entropy, K-multiple secret keys for $K \geq 2$, perfect HVZK, strong special sound, and second key recovery resistant in the multi-user setting, then the signature scheme SIG[ID] is MU-SUF-CMA-C secure in the non-programmable random oracle model.

In particular, if there is an adversary $\mathcal A$ that breaks the MU-SUF-CMA-C security of SIG[ID] in time $T_{\mathcal A}$ with success probability $\operatorname{Adv}_{\mathcal A,\operatorname{SIG[ID]}}^{\operatorname{MU-SUF-CMA-C}}(\lambda)$, then there is an algorithm $\mathcal B_1$ breaking the strong special soundness of ID in time $T_{\mathcal B_1}=\mathcal O(T_{\mathcal A})$ with probability $\operatorname{Adv}_{\mathcal B_1,\operatorname{ID}}^{\operatorname{SSS}}(\lambda)$ and an algorithm $\mathcal B_2$ breaking the second key recovery resistance of ID in the multi-user setting in time $T_{\mathcal B_2}=\mathcal O(T_{\mathcal A})$ with probability $\operatorname{Adv}_{\mathcal B_2,\operatorname{ID}}^{\operatorname{2nd} KR}(\lambda)$ such that

$$\begin{split} \mathsf{Adv}^{\mathsf{MU-SUF-CMA-C}}_{\mathcal{A},\mathsf{SIG[ID]}}(\lambda) & \leq \frac{K}{K-1} \mathsf{Adv}^{2^{\mathsf{nd}}\mathsf{KR}}_{\mathcal{B}_1,\mathsf{ID}}(\lambda) + \mathsf{Adv}^{\mathsf{SSS}}_{\mathcal{B}_2,\mathsf{ID}}(\lambda) \\ & + \frac{Q_{\mathsf{RO}}+1}{2^{\rho\kappa}} + \frac{T \cdot Q_{\mathsf{sig}}(Q_{\mathsf{RO}} + T \cdot Q_{\mathsf{sig}})}{2^{\rho\gamma}}. \end{split} \tag{1}$$

Here, Q_{RO} and Q_{sig} are the maximum number of RO queries and signature queries issued by A, respectively.

This theorem shows that the MU-SUF-CMA-C security of our signature scheme is *tightly reduced* to the security against the second key recovery attack and the strong special soundness of ID in the non-programmable ROM. We will provide the proof in Appendix A.

```
\mathsf{Setup}(1^{\lambda})
                                                                                          Sign^{H}(ssk, m)
         \mathsf{par}' \leftarrow \mathsf{ISetup}(1^{\lambda})
                                                                                           1:
                                                                                                    foreach \tau \in [T] do
        return par := par'
                                                                                                         foreach j \in [\rho] do
                                                                                           2:
                                                                                                              (\mathsf{com}_{\tau,j},\mathsf{st}_{\tau,j}) \leftarrow \mathsf{P}_1(\mathsf{pk},\mathsf{sk})
                                                                                           3:
KGen(par)
                                                                                           4:
 1: (pk, sk) \leftarrow IGen(par)
                                                                                           5:
                                                                                                         \vec{\mathsf{com}}_{\tau} \coloneqq (\mathsf{com}_{\tau,1}, \dots, \mathsf{com}_{\tau,\rho})
         return (svk, ssk) := (pk, (pk, sk))
                                                                                                         \mathsf{pfx}_{\tau} \coloneqq (\mathsf{pk}, m, \vec{\mathsf{com}}_{\tau})
                                                                                           6:
                                                                                                         foreach j \in [\rho] do
                                                                                           7:
\mathsf{Vf}^\mathsf{H}(\mathsf{svk}, m, \sigma)
                                                                                                             S_i := \emptyset
                                                                                           8:
                                                                                                             while S_i \neq \{0,1\}^t do
        (\mathsf{com}_j, \mathsf{ch}_j, \mathsf{resp}_j)_{j \in [\rho]} \coloneqq \sigma
                                                                                           9:
                                                                                                                  \mathsf{ch} \leftarrow \$ \{0,1\}^t \setminus S_i
         \vec{com} := (com_1, \dots, com_{\varrho})
                                                                                          10:
         pfx := (pk, m, \overrightarrow{com})
                                                                                                                  \mathsf{resp}_{\tau,j,\mathsf{ch}} \leftarrow \mathsf{P}_2(\mathsf{ch},\mathsf{st}_{\tau,j})
                                                                                          11:
                                                                                                                  h_{	au,j,\mathsf{ch}} \coloneqq \mathsf{H}(\mathsf{pfx}_{	au},j,\mathsf{ch},\mathsf{resp}_{	au,j,\mathsf{ch}})
          foreach j \in [\rho] do
                                                                                          12:
                                                                                                                  if h_{\tau,j,\text{ch}} = 0^{\gamma} then
               if H(pfx, j, ch_j, resp_j) \neq 0^{\gamma} then
                                                                                          13:
                                                                                                                      \mathsf{ch}_{\tau,i} \coloneqq \mathsf{ch}
                   return 0
                                                                                          14:
 6:
                                                                                                                       \mathbf{break} \quad \# \text{ proceed to next } j
               if V(pk, com_i, ch_i, resp_i) = 0 then
                                                                                          15:
 7:
                                                                                                                  else
                                                                                          16:
                   return 0
 8:
                                                                                                                       S_j := S_j \cup \{\mathsf{ch}\}
                                                                                          17:
          endfor
                                                                                                             endwhile
                                                                                          18:
10: return 1
                                                                                                         endfor
                                                                                          19:
                                                                                                         if \forall j \; \exists \mathsf{ch} : \; h_{\tau,j,\mathsf{ch}} = 0^{\gamma} \; \mathsf{then}
                                                                                          20:
                                                                                                             break // succeed in signing
                                                                                          21:
                                                                                          22:
                                                                                                         if \tau = T then
                                                                                                             return \perp
                                                                                          23:
                                                                                                    endfor
                                                                                          24:
                                                                                                    \sigma \coloneqq (\mathsf{com}_{\hat{\tau},j}, \mathsf{ch}_{\hat{\tau},j}, \mathsf{resp}_{\hat{\tau},j,\mathsf{ch}_{\hat{\tau},j}})_{j \in [\rho]}
                                                                                                    return \sigma
                                                                                          27:
```

Fig. 2: The signature scheme SIG[ID].

Remark 2. In our proof, we use the HVZK property of the underlying identification scheme in a different way than in the (typical) way that the HVZK property is used in the single-user setting (i.e., to respond to signing queries), which allows us to construct a reduction in the non-programmable ROM. To prove the security in the non-programmable ROM, we cannot straightforwardly adopt the result by Kondi-shelat [17], which proves the ZK property of the randomized Fischlin transformation in the programmable ROM by assuming the HVZK property of the underlying interactive protocol. We do not use this result

to respond to signing queries. Instead, the reduction uses a secret key sampled by itself to respond to signing queries. This allows us to avoid programming a RO. The perfect HVZK property is used to argue that this secret key possessed by the reduction will not leak to the adversary. On the other hand, according to the proof by Kondi and shelat, zero-knowledge seems to rely only on a high entropy of commitment and perfect zero-knowledge of the underlying Sigma protocol, meaning that they achieve statistical zero-knowledge. However, we found a flaw in their proof; zero knowledge also relies on the strong special soundness of the underlying Sigma protocol. In Appendix C, we will point out the flaw in their proof and provide the correct proof.

3.1 Improving Efficiency

If the verifier algorithm of the underlying ID scheme can be represented as

$$V(pk, com, ch, resp) = 1 \iff com = f^{V}(pk, ch, resp)$$
 (2)

for some efficiently computable function f^{V} , we can eliminate com in the signature as in Schnorr signature. In this case, $\mathsf{Vf}^{\mathsf{H}}(\mathsf{svk} = \mathsf{pk}, m, \sigma = (\mathsf{ch}_j, \mathsf{resp}_j)_j)$ first reconstructs $\mathsf{com}_j \coloneqq f^{\mathsf{V}}(\mathsf{pk}, \mathsf{ch}_j, \mathsf{resp}_j)$, $\mathsf{com} \coloneqq (\mathsf{com}_j)_j$, and then checks if $\mathsf{H}(\mathsf{pk}, m, \mathsf{com}, j, \mathsf{ch}_j, \mathsf{resp}_j) = 0^\gamma$ holds for all $j \in [\rho]$.

4 Instantiations

In this section, we provide concrete instantiations of tightly MU-SUF-CMA-C-secure signatures from our framework. We consider an instantiation from classical groups based on the Okamoto identification [21] and an instantiation from isogenies based on Couveignes-Stolbunov identification [7,5,26] with the parallel-OR technique [8].

4.1 Instantiation based on Classical Groups

Okamoto Identification Scheme. The Okamoto protocol [21], which is based on the discrete-logarithm assumption, is one of the most important instantiations of our framework. Let GGen be a PPT algorithm, called group generator, that on input 1^{λ} generates a prime q, a multiplicative group \mathbb{G} with order q, and a generator $g \in \mathbb{G}$, and outputs (\mathbb{G}, q, g) . The Okamoto protocol $\mathsf{ID}_{\mathsf{Oka}} := (\mathsf{ISetup}_{\mathsf{Oka}}, \mathsf{IGen}_{\mathsf{Oka}}, \mathsf{P}_{\mathsf{Oka}}, \mathsf{V}_{\mathsf{Oka}})$ is defined in Figure 3.

For completeness, we show that ID_{Oka} has correctness, multiple secret keys, perfect HVZK, strong special soundness, and second key recovery resistance in the multi-user setting. To this end, we recall the discrete logarithm assumption.

$\boxed{ISetup_{Oka}(1^{\lambda})}$		P _{Oka}	,1(pk, sk)
1:	$(\mathbb{G},q,g) \leftarrow GGen(1^{\lambda})$	1:	$(r_1,r_2) \leftarrow \$ \left(\mathbb{Z}_q ight)^2$
2:	$\alpha \leftarrow \$ \mathbb{Z}_q; g_1 \leftarrow g^{\alpha}$	2:	$R \coloneqq g^{r_1} g_1^{r_2}$
3:	$ChSet \coloneqq \mathbb{Z}_q$	3:	$com \coloneqq R$
4:	$\mathbf{return} par \coloneqq (\mathbb{G}, q, g, g_1)$	4:	$st \coloneqq (sk, r_1, r_2)$
		5:	$\mathbf{return}\ (com,st)$
IGen	_{Oka} (par)		
1:	$sk \coloneqq (s_1, s_2) \leftarrow \!\!\! \ast (\mathbb{Z}_q)^2$	P _{Oka}	$_{,2}(st,ch)$
2:	$pk \coloneqq g^{s_1}g_1^{s_2}$	1:	$y_1 := r_1 + ch \cdot s_1 \bmod q$
3:	$\mathbf{return}\ (pk,sk)$	2:	$y_2 \coloneqq r_2 + ch \cdot s_2 \bmod q$
		3:	$resp \coloneqq (y_1, y_2)$
V_{Oka}	(par, pk, com, ch, resp)	4:	return resp
1:	$(y_1,y_2)\coloneqqresp$		
2:	$\mathbf{if}\ com = g^{y_1}g_1^{y_2}/pk^ch\ \mathbf{then}$		
3:	return 1		
4:	else return 0		

Fig. 3: The Okamoto protocol ID_{Oka}.

Definition 11 (Discrete Logarithm (DL) Assumption). We say that DL assumption holds for GGen if for all PPT adversaries A, it holds that

$$\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{A},\mathsf{GGen}}(\lambda) \coloneqq \Pr \left[g^{\alpha'} = h \, \left| \, \begin{array}{l} (\mathbb{G},q,g) \leftarrow \mathsf{GGen}(1^{\lambda}), \\ \alpha \leftarrow \mathbb{Z}_q, \\ g_1 \coloneqq g^{\alpha}, \\ \alpha' \leftarrow \mathcal{A}(\mathbb{G},q,g,g_1) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

We now prove ID_{Oka} 's properties.

Theorem 3. $\mathsf{ID}_{\mathsf{Oka}}$ is correct, and it has $\log q$ -bits of commitment min-entropy and q-multiple secret keys.

Proof. The correctness of $\mathsf{ID}_{\mathsf{Oka}}$ is clear (proved in [21]). The commitment of $\mathsf{ID}_{\mathsf{Oka}}$ consists of a randomly chosen group element over \mathbb{G} with order q. Thus, $\mathsf{ID}_{\mathsf{Oka}}$ has $\log q$ -bits of commitment min-entropy. Finally, for each pk , there are q pairs of (s_1, s_2) such that $\mathsf{pk} = g^{s_1}g_1^{s_2}$. Thus, $\mathsf{ID}_{\mathsf{Oka}}$ has q-multiple secret keys. \square

Theorem 4. Under the DL assumption, ID_{Oka} is strong special sound. More precisely, there exists an extractor Ext such that, for any adversary $\mathcal A$ breaking strong special soundness of ID_{Oka} with advantage $Adv_{\mathcal A,ID_{Oka}}^{SSS}(\lambda)$, there exists a DL solver $\mathcal B$ whose advantage is

$$\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{B},\mathsf{GGen}}(\lambda) = \mathsf{Adv}^{\mathsf{SSS}}_{\mathcal{A},\mathsf{ID}_{\mathsf{Oka}}}(\lambda).$$

This means that $Adv_{\mathcal{A},ID_{Oka}}^{SSS}(\lambda)$ is upper bounded by $\max_{\mathcal{B}} Adv_{\mathcal{B},GGen}^{DL}(\lambda)$.

Proof. We define the extractor Ext as follows. Let (pk, com, ch, resp, ch', resp') be Ext's input. If $ch \neq ch'$, Ext computes

$$s_1^* \coloneqq (y_1 - y_1')/(\mathsf{ch} - \mathsf{ch}') \bmod q,$$

 $s_2^* \coloneqq (y_2 - y_2')/(\mathsf{ch} - \mathsf{ch}') \bmod q$

and outputs $\mathsf{sk}^* \coloneqq (s_1^*, s_2^*)$. Otherwise, Ext outputs \perp .

When $\operatorname{ch} \neq \operatorname{ch}'$, it is easy to see that Ext outputs a valid secret key. So, $\operatorname{Adv}_{\mathcal{A},\operatorname{ID}_{\mathsf{Oka}}}^{\mathsf{SSS}}(\lambda)$ is the probability that \mathcal{A} outputs $(\mathsf{pk},\mathsf{com},\mathsf{ch},\mathsf{resp},\mathsf{ch}',\mathsf{resp}')$ such that $\operatorname{ch} = \operatorname{ch}'$, $\operatorname{resp} \neq \operatorname{resp}'$, and $\mathsf{V}(\mathsf{pk},\mathsf{com},\mathsf{ch},\mathsf{resp}) = \mathsf{V}(\mathsf{pk},\mathsf{com},\mathsf{ch}',\mathsf{resp}') = 1$. Note that $(\operatorname{ch} = \operatorname{ch}' \wedge \operatorname{resp} \neq \operatorname{resp}')$ implies $y_2 \neq y_2'$ if $\mathsf{V}(\mathsf{pk},\mathsf{com},\mathsf{ch},(y_1,y_2)) = \mathsf{V}(\mathsf{pk},\mathsf{com},\mathsf{ch}',(y_1',y_2')) = 1$.

Now we construct a PPT algorithm $\mathcal B$ that solves the DL problem by using $\mathcal A$. Upon receiving a DL instance $(\mathbb G,q,g,g_1)$, $\mathcal B$ executes $\mathcal A$ on input par := $(\mathbb G,q,g,g_1)$. If $\mathcal A$ outputs pk and two valid transcripts (com, ch, resp) and (com, ch', resp') with respect to pk such that $\mathsf{ch} = \mathsf{ch}'$ and $y_2 \neq y_2'$, $\mathcal B$ can compute a DL of g_1 as

$$\alpha \coloneqq (y_1 - y_1')/(y_2' - y_2).$$

Clearly, $\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{B},\mathsf{GGen}}(\lambda) = \mathsf{Adv}^{\mathsf{SSS}}_{\mathcal{A},\mathsf{ID}_{\mathsf{Oka}}}(\lambda)$ holds. In addition, the running time of \mathcal{B} is \mathcal{A} 's running time plus $\mathsf{poly}(\lambda)$.

Theorem 5. $\mathsf{ID}_{\mathsf{Oka}}$ is perfect special honest verifier zero-knowledge.

Proof. Consider the following simulator: On input (pk, ch), choose $y_1, y_2 \leftarrow \mathbb{Z}_q$, compute $R := g^{y_1}g_1^{y_2}/\mathsf{pk}^{\mathsf{ch}}$, and output $(R, (y_1, y_2))$.

It is easy to confirm that the simulator's output has the same distribution as the real transcript between an honest prover and an honest verifier. \Box

Theorem 6. Under the DL assumption, $\mathsf{ID}_{\mathsf{Oka}}$ satisfies the second key recovery resistance. In particular, if there is an adversary \mathcal{A} that breaks the second key recovery resistance in time $T_{\mathcal{A}}$ with success probability $\mathsf{Adv}_{\mathcal{A},\mathsf{ID}}^{\mathsf{2nd}\mathsf{KR}}(\lambda)$, then there is an algorithm \mathcal{B} solving the DL problem in time $T_{\mathcal{B}} = T_{\mathcal{A}} + N \cdot \mathsf{poly}(\lambda)$ with probability $\mathsf{Adv}_{\mathcal{B},\mathsf{GGen}}^{\mathsf{DL}}(\lambda) = \mathsf{Adv}_{\mathcal{A},\mathsf{ID}}^{\mathsf{2nd}\mathsf{KR}}(\lambda)$.

Proof. Let \mathcal{A} be an adversary that breaks the second key recovery resistance. Consider the following DL solver \mathcal{B} that uses \mathcal{A} as a subroutine: Upon receiving a DL instance (\mathbb{G},q,g,g_1) , \mathcal{B} sets par $:= (\mathbb{G},q,g,g_1)$ and generates $(\mathsf{pk}_i,\mathsf{sk}_i) \leftarrow \mathsf{IGen}(\mathsf{par})$ for each $i \in [N]$. \mathcal{B} executes \mathcal{A} on input $(\mathsf{par},\{(\mathsf{pk}_i,\mathsf{sk}_i)\}_{i\in [N]})$ and receives (i^*,sk^*) from \mathcal{A} such that $\mathsf{sk}^* = (s_1^*,s_2^*) \neq \mathsf{sk}_{i^*} = (s_{1,i^*},s_{2,i^*})$ and $g^{s_1^*}g_1^{s_2^*} = \mathsf{pk}_{i^*} = g^{s_{1,i^*}}g_1^{s_{2,i^*}}$. Then, \mathcal{B} computes

$$\alpha := (s_1^* - s_{1,i^*})/(s_{2,i^*} - s_2^*) \bmod q,$$

and outputs α as the solution of the DL instance.

We can verify that if \mathcal{A} breaks the second key recovery resistance, \mathcal{B} solves the given DL instance. Therefore, we have $\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{B},\mathsf{GGen}}(\lambda) = \mathsf{Adv}^{\mathsf{2nd}}_{\mathcal{A},\mathsf{ID}}(\lambda)$. Note that the running time of \mathcal{B} is $T_{\mathcal{B}} = T_{\mathcal{A}} + N \cdot \mathsf{poly}(\lambda)$ since \mathcal{B} executes \mathcal{A} once and prepares N key pairs.

Table 2: Concrete parameters for $SIG[ID_{Oka}]$ in 128-bit security and corresponding efficiency. We set T=3.

$\rho \gamma t$	β	$\# \mathrm{hash}$	svk	$ \sigma $
				2084 B
				1444 B
16 8 13	$3 2^{-126}$	4096	32 B	1050 B

By instantiating SIG[ID] with ID_{Oka} , we obtain a signature scheme, $SIG[ID_{Oka}]$, whose MU-SUF-CMA-C security is tightly implied from the DL assumption.

Corollary 1. Under the DL assumption, SIG[ID_{Oka}] is MU-SUF-CMA-C secure in the non-programmable random oracle model. In particular, for any adversary A that breaks the MU-SUF-CMA-C security of SIG[ID_{Oka}] in time T_A with probability $Adv_{\mathcal{A},SIG[ID_{Oka}]}^{\text{MU-SUF-CMA-C}}(\lambda)$, there is an algorithm \mathcal{B} solving the DL problem in time $T_{\mathcal{B}} = \mathcal{O}(T_{\mathcal{A}})$ with probability $Adv_{\mathcal{B},GGen}^{\text{DL}}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{MU-SUF-CMA-C}}_{\mathcal{A},\mathsf{SIG}[\mathsf{ID}_{\mathsf{Oka}}]}(\lambda) \leq 3\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{B},\mathsf{GGen}}(\lambda) + \frac{Q_{\mathsf{RO}}+1}{2^{\rho\log_2 q}} + \frac{T \cdot Q_{\mathsf{sig}}(Q_{\mathsf{RO}} + T \cdot Q_{\mathsf{sig}})}{2^{\rho\gamma}}.$$

Remark 3 (Variants of Okamoto identification). It is worth noting that there are versions of Okamoto identification based on RSA assumption [15] or Factoring assumption [10]. Using these identification schemes, we can obtain the tightly MU-SUF-CMA-C-secure signatures from RSA and Factoring assumption, respectively.

Efficiency Evaluation. We provide the efficiency of $SIG[ID_{Oka}]$. First, we explain how to choose parameters for randomized Fischlin transformation. Here, we recall the recent result by Chen and Lindell [6] for the original version since their result is also valid for randomized Fischlin transformation. They proved the correctness error probability, the expected number of hash computations, and the signature size summarized below:

- Probability of correctness errors: $\beta = 2^{-(2^{t-\gamma} \log \hat{e} + \log \rho)T}$.
- Expected number of hash computations: #hash= $\rho \cdot 2^{\gamma}/(1-\beta)$.
- Signature size (bits): $|\sigma| = \rho \cdot (|\mathsf{com}| + t + |\mathsf{resp}|)$.

Further, to achieve λ -bit security, the parameters must satisfy the following conditions: $\rho \cdot \gamma = \omega(\lambda)$ and $t - \gamma = \omega(\lambda)$. Also, Chen and Lindell suggest that for any value of γ , for $\rho \leq 64$ set $t = \gamma + 5$ and for $\rho > 64$ set $t = \gamma + 6$. We then explain the choice of multiplicative group. To achieve 128-bit security, our scheme can use NIST P-256 curve with $\log |\mathbb{G}| = 256$ bits since $SIG[ID_{Oka}]$ is tightly secure.

According to the parameter selections above, $SIG[ID_{Oka}]$ has |resp| = 512 bits, and the total signature size is $\rho(512 + t)$ bits for 128-bit security. Note that we

```
IGenOR(par)
                                                                                            P_{OR,1}(pk, sk)
                                                                                              1: ch_{1-b} \leftarrow s \{0,1\}^{\ell}
 1: b \leftarrow \$ \{0, 1\}
        (\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{IGen}(\mathsf{par})
                                                                                                       (\mathsf{com}_{1-b}, \mathsf{resp}_{1-b}) \leftarrow \mathsf{Sim}(\mathsf{pk}_{1-b}, \mathsf{ch}_{1-b})
          (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{IGen}(\mathsf{par})
                                                                                                       (\mathsf{com}_b, \mathsf{st}_b) \leftarrow \mathsf{P}_1(\mathsf{pk}_b, \mathsf{sk}_b)
        \mathsf{pk} \coloneqq (\mathsf{pk}_0, \mathsf{pk}_1); \mathsf{sk} \coloneqq (b, \mathsf{sk}_b)
                                                                                                       com := (com_0, com_1)
        return (pk, sk)
                                                                                                       \mathsf{st} \coloneqq (\mathsf{st}_b, \mathsf{ch}_{1-b}, \mathsf{resp}_{1-b})
                                                                                                       return (com, st)
V_{OR}(pk, com, ch, resp)
                                                                                            P_{OR,2}(st, ch)
 1: \mathsf{ch}_1 \coloneqq \mathsf{ch} \oplus \mathsf{ch}_0
          if V(pk_0, com_0, ch_0, resp_0) = 0 then
                                                                                              1: \mathsf{ch}_b \coloneqq \mathsf{ch} \oplus \mathsf{ch}_{1-b}
                                                                                                       \mathsf{resp}_b \leftarrow \mathsf{P}_2(\mathsf{ch}_b, \mathsf{st}_b)
                return 0
          if V(pk_1, com_1, ch_1, resp_1) = 0 then
                                                                                                       \mathsf{resp} \coloneqq (\mathsf{ch}_0, \mathsf{resp}_0, \mathsf{resp}_1)
 4:
               return 0
                                                                                                     return resp
 5:
         return 1
```

Fig. 4: The parallel-OR identification scheme $\mathsf{ID}_\mathsf{OR}[\mathsf{ID}]$ constructed from an identification scheme ID , where $\mathsf{ISetup}_\mathsf{OR} \coloneqq \mathsf{ISetup}$.

can remove com from the signature because ID_{Oka} 's verifier algorithm can be represented as follows. (See Section 3.1.)

$$\mathsf{V}(\mathsf{pk},\mathsf{com},\mathsf{ch},(y_1,y_2)) = 1 \iff \mathsf{com} = f_{\mathsf{Oka}}^{\mathsf{V}}(\mathsf{pk},\mathsf{ch},(y_1,y_2)) = g^{y_1}g_1^{y_2}/\mathsf{pk}^{\mathsf{ch}}.$$

We give in Table 2 the efficiency estimations for $SIG[ID_{Oka}]$ in 128-bit security in each parameter. As observed, there is a trade-off between signature size and signing time. Spending more computational cost on signature generation can shorten the signature size.

4.2 Instantiation from Isogenies

We will show an isogeny-based tightly MU-SUF-CMA-C-secure signature derived from our framework. We first show that the so-called parallel-OR identification scheme meets the requirements of our framework. Then, we provide a concrete instantiation of the parallel-OR identification scheme based on isogeny.

Parallel-OR Identification Scheme. We first explain the parallel-OR identification scheme [8]. Let ID = (ISetup, IGen, P, V) be a canonical identification scheme with ℓ bits challenges and let Sim be a special honest verifier zero-knowledge simulator of ID. Then the new identification scheme $ID_{OR}[ID] := (ISetup_{OR}, IGen_{OR}, P_{OR}, V_{OR})$ is constructed as shown in Figure 4.

For completeness, we show that $\mathsf{ID}_{\mathsf{OR}}$ has correctness, multiple secret keys, perfect HVZK, strong special soundness, and 2nd key recovery resistance in the multi-user setting. To this end, we recall the security notions for identification schemes, which are required to prove this property of $\mathsf{ID}_{\mathsf{OR}}$.

Definition 12 (Key Recovery [16, Definition 2.3]). We say that ID is key recovery resistant if for all PPT adversaries A, it holds that

$$\begin{split} \mathsf{Adv}^{\mathsf{KR}}_{\mathcal{A},\mathsf{ID}}(\lambda) &\coloneqq \Pr \left[(\mathsf{pk},\mathsf{sk}^*) \in \mathsf{IGen}(\mathsf{par}) \; \middle| \; \begin{array}{l} \mathsf{par} \leftarrow \mathsf{ISetup}(1^{\lambda}), \\ (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{IGen}(\mathsf{par}), \\ \mathsf{sk}^* \leftarrow \mathcal{A}(\mathsf{par},\mathsf{pk}) \end{array} \right] \\ &\leq \mathsf{negl}(\lambda). \end{split}$$

Definition 13 (Random Self-Reducibility [16, Definition 2.5]). We say that ID is random self-reducible if there is a PPT algorithm ReRand and a deterministic algorithm DeRand such that, for all $(pk, sk) \in IGen(par)$:

- pk' and pk" have the same distribution, where (pk',td') ← ReRand(par, pk) is the rerandomized public key and (pk", sk") ← IGen(par) is a freshly-generated key pair.
- For all (pk',td') ∈ ReRand(par,pk), for all (pk',sk') ∈ IGen(par), and sk* ← DeRand(pk,pk',sk',td'), we have (pk,sk*) ∈ IGen(par). That is, DeRand returns a valid secret key sk* with respect to pk, given any valid secret key sk' for pk'.

We now show ID_{OR}[ID]'s properties.

Theorem 7. If ID is correct and it has κ -bits of commitment min-entropy, then $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ is correct and it has 2κ -bits of commitment min-entropy. Also, if ID has a single secret key, ID has 2-multiple secret keys.

Proof. The correctness of $ID_{OR}[ID]$ is clear. The commitment of $ID_{OR}[ID]$ consists of two independent commitments of ID. Thus, $ID_{OR}[ID]$ has 2κ -bits of commitment min-entropy. Finally, since the underlying identification scheme ID has a single secret key per public key, there are two secret keys $(0, sk_0)$ and $(1, sk_1)$ for each pk of $ID_{OR}[ID]$. Thus, $ID_{OR}[ID]$ has 2-multiple secret keys.

Theorem 8. If ID is strong special sound, ID_{OR}[ID] is also strong special sound. More precisely, if there exists an extractor $\mathsf{Ext}_{\mathsf{ID}}$ such that, for any adversary \mathcal{A} , its advantage is at most $\mathsf{Adv}^{\mathsf{SSS}}_{\mathsf{ID}}(\lambda)$, then there exists an extractor $\mathsf{Ext}_{\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}$ such that, for any adversary \mathcal{B} , its advantage is at most $\mathsf{Adv}^{\mathsf{SSS}}_{\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}(\lambda) = \mathsf{Adv}^{\mathsf{SSS}}_{\mathsf{ID}}(\lambda)$.

Proof. Let $\mathsf{Ext}_{\mathsf{ID}}$ be an extractor of ID . Consider the extractor $\mathsf{Ext}_{\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}$ of $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ as follows. Given two valid transcripts

$$((\mathsf{com}_0, \mathsf{com}_1), \mathsf{ch}, (\mathsf{ch}_0, \mathsf{resp}_0, \mathsf{resp}_1))$$

and

$$((\mathsf{com}_0, \mathsf{com}_1), \mathsf{ch}', (\mathsf{ch}'_0, \mathsf{resp}'_0, \mathsf{resp}'_1))$$

with respect to $pk = (pk_0, pk_1)$,

- if $\mathsf{ch}_0 \neq \mathsf{ch}_0'$ or $\mathsf{resp}_0 \neq \mathsf{resp}_0'$, $\mathsf{Ext}_{\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}$ outputs

$$(0, \mathsf{Ext}_{\mathsf{ID}}(\mathsf{pk}_0, \mathsf{com}_0, \mathsf{ch}_0, \mathsf{resp}_0, \mathsf{ch}_0', \mathsf{resp}_0')).$$

- Otherwise, $\mathsf{ch}_1 \neq \mathsf{ch}_1'$ or $\mathsf{resp}_1 \neq \mathsf{resp}_1'$ must hold, where $\mathsf{ch}_1 := \mathsf{ch} \oplus \mathsf{ch}_0$, $\mathsf{ch}_1' := \mathsf{ch}' \oplus \mathsf{ch}_0'$. In this case, $\mathsf{Ext}_{\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}$ outputs

$$(1, \mathsf{Ext}_{\mathsf{ID}}(\mathsf{pk}_1, \mathsf{com}_1, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}_1', \mathsf{resp}_1')).$$

Let \mathcal{B} be an arbitrary algorithm that breaks strong special soundness of $ID_{OR}[ID]$. In order to estimate \mathcal{B} 's advantage, consider $\mathcal{A}_{\mathcal{B}}$ that works as follows: On input par, $\mathcal{A}_{\mathcal{B}}$ runs \mathcal{B} and obtains \mathcal{B} 's output $(\mathsf{pk}_0,\mathsf{pk}_1),(\mathsf{com}_0,\mathsf{com}_1),\mathsf{ch},(\mathsf{ch}_0,\mathsf{resp}_0,\mathsf{resp}_1),\mathsf{ch}',(\mathsf{ch}_0',\mathsf{resp}_0',\mathsf{resp}_0')$. If $\mathsf{ch}_0 \neq \mathsf{ch}_0'$ or $\mathsf{resp}_0 \neq \mathsf{resp}_0'$, $\mathcal{A}_{\mathcal{B}}$ outputs $(\mathsf{pk}_0,\mathsf{com}_0,\mathsf{ch}_0,\mathsf{resp}_0,\mathsf{ch}_0',\mathsf{resp}_0')$, otherwise outputs $(\mathsf{pk}_1,\mathsf{com}_1,\mathsf{ch}_1,\mathsf{resp}_1',\mathsf{ch}_1',\mathsf{resp}_1')$, where $\mathsf{ch}_1 \coloneqq \mathsf{ch} \oplus \mathsf{ch}_0,\mathsf{ch}_1' \coloneqq \mathsf{ch}' \oplus \mathsf{ch}_0'$.

From the assumption, for this $\mathcal{A}_{\mathcal{B}}$, $\mathsf{Ext}_{\mathsf{ID}}$ successfully extracts a valid secret key with probability at least $1-\mathsf{Adv}_{\mathsf{ID}}^{\mathsf{SSS}}(\lambda)$. On the other hand, the distribution of $\mathsf{Ext}_{\mathsf{ID}}$'s input come from $\mathcal{A}_{\mathcal{B}}$ is identical to the distribution of $\mathsf{Ext}_{\mathsf{ID}}$'s input when $\mathsf{Ext}_{\mathsf{ID}}$ is used as a subroutine of $\mathsf{Ext}_{\mathsf{IDoR}[\mathsf{ID}]}$. Therefore, $\mathsf{Ext}_{\mathsf{IDoR}[\mathsf{ID}]}$ obtains a valid sk_0 or sk_1 with probability at least $1-\mathsf{Adv}_{\mathsf{ID}}^{\mathsf{SSS}}(\lambda)$. That is, \mathcal{B} 's advantage is at most $\mathsf{Adv}_{\mathsf{ID}}^{\mathsf{SSS}}(\lambda)$.

Theorem 9. If ID is perfect special honest verifier zero-knowledge, ID_{OR}[ID] is also perfect special honest verifier zero-knowledge.

Proof. Let Sim be a simulator for ID. We can easily construct a simulator $\mathsf{Sim}_{\mathsf{OR}}$ for $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ by using Sim as follows: $\mathsf{Sim}_{\mathsf{OR}}$'s input is $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$ and $\mathsf{ch} \in \{0,1\}^\ell$. First, choose ch_0 randomly from $\{0,1\}^\ell$ and set $\mathsf{ch}_1 \coloneqq \mathsf{ch} \oplus \mathsf{ch}_0$. Run $(\mathsf{com}_b, \mathsf{resp}_b) \leftarrow \mathsf{Sim}(\mathsf{pk}_b, \mathsf{ch}_b)$ for each $b \in \{0,1\}$, and output $\mathsf{com} \coloneqq (\mathsf{com}_0, \mathsf{com}_1)$, $\mathsf{resp} \coloneqq (\mathsf{ch}_0, \mathsf{resp}_0, \mathsf{resp}_1)$. It is easy to see $\mathsf{Sim}_{\mathsf{OR}}$'s simulation is perfect from the fact that ID is perfect special honest verifier zero-knowledge. \square

Theorem 10. Assuming that ID is key recovery resistant, random self-reducible, and has a single secret key, $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ is the 2nd key recovery resistant in the multi-user setting. In particular, if there is an adversary \mathcal{A} that breaks the second key recovery resistance in time $T_{\mathcal{A}}$ with success probability $\mathsf{Adv}_{\mathcal{A},\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}^{\mathsf{2nd}\mathsf{KR}}(\lambda)$, then there is an algorithm \mathcal{B} breaking the key recovery resistance of ID in time $T_{\mathcal{B}} = T_{\mathcal{A}} + N \cdot \mathsf{poly}(\lambda)$ with probability $\mathsf{Adv}_{\mathcal{B},\mathsf{ID}}^{\mathsf{KR}}(\lambda) = \mathsf{Adv}_{\mathcal{A},\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}^{\mathsf{nd}\mathsf{KR}}(\lambda)$.

Proof. Let \mathcal{A} be an adversary that breaks the second key recovery resistance of $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$. We show a reduction \mathcal{B} that breaks the key recovery resistance of ID by using \mathcal{A} . The description of \mathcal{B} is as follows.

Upon receiving a parameter par and a public key pk , \mathcal{B} generates $(\mathsf{pk}_i, \mathsf{sk}_i)$ for each $i \in [N]$ as follows:

- 1. Sample $b_i \leftarrow \$ \{0, 1\}$.
- 2. Generate $(\mathsf{pk}_{i,b_i}, \mathsf{sk}_{i,b_i}) \leftarrow \mathsf{IGen}(\mathsf{par})$.
- 3. Generate $(\mathsf{pk}_{i,1-b_i}, \mathsf{td}_i) \leftarrow \mathsf{ReRand}(\mathsf{par}, \mathsf{pk})$.
- 4. Set $\mathsf{pk}_i \coloneqq (\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1})$ and $\mathsf{sk}_i \coloneqq (b_i, \mathsf{sk}_{i,b_i})$.

Then, \mathcal{B} executes \mathcal{A} on input $(\mathsf{par}, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i \in [N]})$ and receives $(i^*, (b', \mathsf{sk}'_{i^*, b'}))$ from \mathcal{A} . Then, \mathcal{B} computes $\mathsf{sk}^* \leftarrow \mathsf{DeRand}(\mathsf{pk}, \mathsf{pk}_{i^*, 1 - b_{i^*}}, \mathsf{sk}'_{i^*, b'}, \mathsf{td}_{i^*})$ and outputs sk^* .

 \mathcal{B} perfectly simulates the second key recovery resistance game, since random self-reducibility ensures that the randomized public key embedded in each $\mathsf{pk}_{i,1-b_i}$ is distributed identically to the fresh public key. Moreover, if \mathcal{A} breaks the second key recovery resistance, its output $(i^*, (b', \mathsf{sk}'_{i^*,b'}))$ must satisfy $b' = 1 - b_{i^*}$ and $(\mathsf{pk}_{i^*,b'}, \mathsf{sk}'_{i^*,b'}) \in \mathsf{IGen}(\mathsf{par})$ since $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ has exactly two valid secret keys. Therefore, \mathcal{B} extracts the secret key of given pk , and we have $\mathsf{Adv}^{\mathsf{KR}}_{\mathcal{B},\mathsf{ID}}(\lambda) = \mathsf{Adv}^{\mathsf{2nd}}_{\mathcal{A},\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}(\lambda)$. The running time of \mathcal{B} is $T_{\mathcal{B}} = T_{\mathcal{A}} + N \cdot \mathsf{poly}(\lambda)$ since \mathcal{B} executes \mathcal{A} once and prepares N key pairs.

Remark 4. Let ID be an identification scheme whose verifier algorithm is represented as in Eq.(2) using a function f^{V} , and let define the function f^{V}_{OR} as

$$f_{\mathsf{OR}}^{\mathsf{V}}(\mathsf{pk}_{\mathsf{OR}},\mathsf{ch}_{\mathsf{OR}},\mathsf{resp}_{\mathsf{OR}}) \coloneqq \left(f^{\mathsf{V}}(\mathsf{pk}_{0},\mathsf{ch}_{0},\mathsf{resp}_{0}),f^{\mathsf{V}}(\mathsf{pk}_{1},\mathsf{ch}_{\mathsf{OR}} \oplus \mathsf{ch}_{0},\mathsf{resp}_{1})\right),$$

where $\mathsf{pk}_{\mathsf{OR}} = (\mathsf{pk}_0, \mathsf{pk}_1), \mathsf{resp}_{\mathsf{OR}} = (\mathsf{ch}_0, \mathsf{resp}_0, \mathsf{resp}_1)$. Then, the verifier algorithm of $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ is also represented as in Eq.(2) by using $f_{\mathsf{OR}}^{\mathsf{V}}$. It means that com can be eliminated in the signature of $\mathsf{SIG}[\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]]$.

Couveignes-Stolbunov Identification Scheme. To instantiate the parallel-OR identification scheme from isogeny, we will use the Couveignes-Stolbunov identification scheme $\mathsf{ID}_{\mathsf{CouSto}}$ [7,5,26], depicted in Figure 5. The details of group action are provided in Appendix B. Let GAGen be an efficient algorithm that generates a description of group action. The original protocol has 1 bit challenge. To extend it to t bits challenges, we simply repeat the protocol t times.

For completeness, we show that ID_{CouSto} has correctness, high commitment entropy, single secret keys, perfect HVZK, strong special soundness, and second key recovery resistance in the multi-user setting.

Theorem 11. $\mathsf{ID}_{\mathsf{CouSto}}$ is correct and it has $t \log |\mathcal{G}|$ -bits of commitment minentropy. Also, it has a single secret key.

Proof. The correctness of $\mathsf{ID}_{\mathsf{CouSto}}$ is clear. The commitment of $\mathsf{ID}_{\mathsf{CouSto}}$ is computed from random elements over \mathcal{G} . Thus, $\mathsf{ID}_{\mathsf{CouSto}}$ has $t \log |\mathcal{G}|$ -bits of commitment min-entropy. Also, since the group action is regular, for each pk , there exists a unique secret key sk . Thus, $\mathsf{ID}_{\mathsf{CouSto}}$ has a single secret key.

Theorem 12. $\mathsf{ID}_{\mathsf{CouSto}}$ is strong special sound. That is, $\mathsf{Adv}_{\mathcal{A},\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}^{\mathsf{SSS}}(\lambda) = 0$ for all (including unlimited bounded) adversaries \mathcal{A} .

Proof. We define the extractor Ext as follows. On input (pk, com, ch, resp, ch', resp'), if $\mathsf{ch} = \mathsf{ch}'$, Ext outputs \bot . If $\mathsf{ch} \neq \mathsf{ch}'$, there must exist an index I such that the I'th bit of them are different, i.e., $\{\mathsf{ch}_I, \mathsf{ch}_I'\} = \{0, 1\}$. For such an I, Ext outputs $(\mathsf{resp}_I' \cdot (\mathsf{resp}_I)^{-1})^{\mathsf{ch}_I - \mathsf{ch}_I'}$.

$ISetup_{CouSto}(1^\lambda)$	$P_{CouSto,1}(pk,sk)$
1: $(\mathcal{G}, \mathcal{E}, E_0, \star) \leftarrow GAGen(1^{\lambda})$) 1: foreach $i \in [t]$ do
$2: ChSet \coloneqq \{0,1\}^t$	$2: \qquad b_i \leftarrow \$ \ \mathcal{G}$
$3: \mathbf{return} \ par \coloneqq (\mathcal{G}, \mathcal{E}, E_0, \star)$	$) 3: \hat{E}_i := b_i \star E_0$
16	$4: \;\; com \coloneqq (\hat{E}_1, \dots, \hat{E}_t)$
$\frac{IGen_{CouSto}(par)}{}$	$5: \;\;st \coloneqq (sk, (b_1, \dots, b_t))$
$1: a_1 \leftarrow \$ \mathcal{G}$	$6: \mathbf{return} (com, st)$
$a: sk \coloneqq a_1$	
$3: pk := E_1 = a_1 \star E_0$	$P_{CouSto,2}(ch,st)$
$4: \mathbf{return} (pk, sk)$	$a_1: (a_1,(b_1,\ldots,b_t)) \coloneqq st$
V (1 1)	$a_0 \coloneqq 1_{\mathcal{G}}$
$V_{CouSto}(pk,com,ch,resp)$	$3: (ch_1, \ldots, ch_t) \coloneqq ch$
1: if com = $(\operatorname{resp}_i \star E_{\operatorname{ch}_i})_{i \in [t]}$	then 4: for each $i \in [t]$ do
2: return 1	$5: \qquad r_i \coloneqq b_i \cdot a_{ch_i}^{-1}$
3: return 0	$6: \mathbf{return} \ resp \coloneqq (r_1, \dots, r_t)$

Fig. 5: The Couveignes-Stolbunov identification scheme ID_{CouSto}.

It is sufficient to show that Ext can extract a_1 when $(ch, resp) \neq (ch', resp')$ and both (pk, com, ch, resp) and (pk, com, ch', resp') are accepted by V_{CouSto} . Further, from the regularity of group action and the fact that com and ch uniquely determine resp, we can assume $ch \neq ch'$.

When $\operatorname{ch} \neq \operatorname{ch}'$, define I as above. Since $E_b = a_1^b \star E_0$ holds for both bit $b \in \{0,1\}$, we have $\operatorname{com}_I = \operatorname{resp}_I \star E_{\operatorname{ch}_I} = (\operatorname{resp}_I \cdot a_1^{\operatorname{ch}_I}) \star E_0$ and $\operatorname{com}_I = (\operatorname{resp}_I' \cdot a_1^{\operatorname{ch}_I'}) \star E_0$. Thus, we have $(\operatorname{resp}_I' \cdot (\operatorname{resp}_I)^{-1})^{\operatorname{ch}_I - \operatorname{ch}_I'} = (a_1^{\operatorname{ch}_I - \operatorname{ch}_I'})^{\operatorname{ch}_I - \operatorname{ch}_I'} = a_1$. Therefore, Ext succeeds in extracting a_1 as desired.

Theorem 13. ID_{CouSto} is perfect special honest verifier zero-knowledge.

Proof. Consider the following simulator: On input ($pk = E_1, ch$), choose $resp_i \leftarrow \mathcal{G}$, compute $com_i := resp_i \star E_{ch_i}$ for all $i \in [t]$, and output

$$(\mathsf{com}, \mathsf{resp}) = ((\mathsf{com}_1, \dots, \mathsf{com}_t), (\mathsf{resp}_1, \dots, \mathsf{resp}_t)).$$

It is easy to confirm that the simulator's output has the same distribution as the real transcript between an honest prover and an honest verifier. \Box

Theorem 14. Under the GADL assumption, $\mathsf{ID}_{\mathsf{CouSto}}$ satisfies the key recovery resistance. In particular, if there is an adversary $\mathcal A$ that breaks the key recovery resistance in time $T_{\mathcal A}$ with success probability $\mathsf{Adv}^{\mathsf{KR}}_{\mathcal A,\mathsf{ID}_{\mathsf{CouSto}}}(\lambda)$, then there is an algorithm $\mathcal B$ solving GADL problems in time $T_{\mathcal B} = T_{\mathcal A}$ with probability $\mathsf{Adv}^{\mathsf{GADL}}_{\mathcal B,\mathsf{GAGen}}(\lambda) = \mathsf{Adv}^{\mathsf{KR}}_{\mathcal A,\mathsf{ID}_{\mathsf{CouSto}}}(\lambda)$.

Proof. Let \mathcal{A} be an adversary that breaks the key recovery resistance. Consider the following GADL solver \mathcal{B} that uses \mathcal{A} as a subroutine: Upon receiving a GADL instance $(\mathcal{G}, \mathcal{E}, E_0, \star, E_1)$, \mathcal{B} sets par $:= (\mathcal{G}, \mathcal{E}, E_0, \star)$ and pk $:= E_1$. \mathcal{B} executes \mathcal{A} on input (par, pk) and receives sk $\in \mathcal{G}$ from \mathcal{A} . \mathcal{B} outputs sk as the solution of the GADL instance.

We can verify that if \mathcal{A} breaks the key recovery resistance, \mathcal{B} solves the given GADL instance since \mathcal{A} 's output sk satisfies $\mathsf{pk} = \mathsf{sk} \star E_0$, meaning that sk is the GADL of $\mathsf{pk} = E_1$ w.r.t. E_0 . Therefore, we have $\mathsf{Adv}_{\mathcal{B},\mathsf{GAGen}}^\mathsf{GADL}(\lambda) = \mathsf{Adv}_{\mathcal{A},\mathsf{ID}_\mathsf{CouSto}}^\mathsf{KR}(\lambda)$. Note that the running time of \mathcal{B} is $T_{\mathcal{B}} = T_{\mathcal{A}}$ since \mathcal{B} executes \mathcal{A} once.

Theorem 15. ID_{CouSto} is random self-reducible.

Proof. ReRand and DeRand are defined as follows:

ReRand(par, pk): Let $E_1 := pk$. Choose $c_1 \leftarrow \mathcal{G}$ and output $pk' := c_1 \star E_1$ and $td' := c_1$.

DeRand(pk, pk', sk', td'): Let $a_1' := \mathsf{sk}'$ satisfying $E_1' = a' \star E_0$ and let $c_1 := \mathsf{td}$. Output $\mathsf{sk}^* := a_1' \cdot (c_1)^{-1}$.

We have that, for all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{IGen}(\mathsf{par})$, pk' is uniformly distributed and has the same distribution as a freshly-generated key pair. Also, for all $(\mathsf{pk}', \mathsf{td}') \leftarrow \mathsf{ReRand}(\mathsf{par}, \mathsf{pk})$ and $(\mathsf{pk}', \mathsf{sk}') = (E_1', a_1') \in \mathsf{IGen}(\mathsf{par})$, we have $E_1' = a_1' \star E_0$ and $E_1' = c_1 \star E_1$. Thus, we have $a_1' \star E_0 = c_1 \star E_1 = (c_1 a_1) \star E_0$. Therefore, $\mathsf{sk}^* = a_1'(c_1)^{-1} = a_1$.

By instantiating SIG[ID] with $ID = ID_{OR}[ID_{CouSto}]$, we obtain a signature scheme, $SIG[ID_{OR}[ID_{CouSto}]]$, whose MU-SUF-CMA-C security is tightly implied from the GADL assumption.

Corollary 2. Under the GADL assumption, SIG[ID_{OR}[ID_{CouSto}]] is MU-SUF-CMA-C secure in the non-programmable random oracle model. In particular, for any adversary \mathcal{A} that breaks the MU-SUF-CMA-C security of SIG[ID_{OR}[ID_{CouSto}]] in time $T_{\mathcal{A}}$ with probability $\operatorname{Adv}_{\mathcal{A},\operatorname{SIG[ID}_{OR}[ID_{CouSto}]]}^{\operatorname{MU-SUF-CMA-C}}(\lambda)$, there is an algorithm \mathcal{B} solving a GADL problem in time $T_{\mathcal{B}} = \mathcal{O}(T_{\mathcal{A}})$ with probability $\operatorname{Adv}_{\mathcal{B},\operatorname{GAGen}}^{\operatorname{GADL}}(\lambda)$ such that

$$\begin{split} \mathsf{Adv}_{\mathcal{A},\mathsf{SIG[ID}_{\mathsf{OR}[\mathsf{ID}_{\mathsf{CouSto}}]]}^{\mathsf{GADL}}(\lambda) \\ & \leq 2\mathsf{Adv}_{\mathcal{B},\mathsf{GAGen}}^{\mathsf{GADL}}(\lambda) + \frac{Q_{\mathsf{RO}} + 1}{2^{\rho t \log|\mathcal{G}|}} + \frac{T \cdot Q_{\mathsf{sig}}(Q_{\mathsf{RO}} + T \cdot Q_{\mathsf{sig}})}{2^{\rho \gamma}}, \end{split}$$

Acknowledgements. This work was partially supported by JST CREST JP-MJCR22M1, Japan.

References

 Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 411–439. Springer, Cham (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_14

- Bader, C.: Efficient signatures with tight real world security in the random-oracle model. In: Gritzalis, D., Kiayias, A., Askoxylakis, I.G. (eds.) CANS 14. LNCS, vol. 8813, pp. 370–383. Springer, Cham (Oct 2014). https://doi.org/10.1007/978-3-319-12280-9_24
- Bader, C., Hofheinz, D., Jager, T., Kiltz, E., Li, Y.: Tightly-secure authenticated key exchange. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 629-658. Springer, Berlin, Heidelberg (Mar 2015). https://doi.org/10.1007/978-3-662-46494-6_26
- Bader, C., Jager, T., Li, Y., Schäge, S.: On the impossibility of tight cryptographic reductions. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 273–304. Springer, Berlin, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_10
- 5. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: Efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 227–247. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-34578-5_9
- Chen, Y.H., Lindell, Y.: Optimizing and implementing fischlin's transform for UC-secure zero knowledge. IACR Communications in Cryptology 1(2) (2024). https://doi.org/10.62056/a66chey6b
- Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
- Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y. (ed.) CRYPTO'94. LNCS, vol. 839, pp. 174–187. Springer, Berlin, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48658-5_19
- 9. Diemert, D., Gellert, K., Jager, T., Lyu, L.: More efficient digital signatures with tight multi-user security. In: Garay, J. (ed.) PKC 2021, Part II. LNCS, vol. 12711, pp. 1–31. Springer, Cham (May 2021). https://doi.org/10.1007/978-3-030-75248-4_1
- Fischlin, M.: On the impossibility of constructing non-interactive statisticallysecret protocols from any trapdoor one-way function. In: Preneel [25], pp. 79–95. https://doi.org/10.1007/3-540-45760-7_7
- 11. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Berlin, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_10
- 12. Fischlin, M., Fischlin, R.: The representation problem based on factoring. In: Preneel [25], pp. 96–113. https://doi.org/10.1007/3-540-45760-7_8
- 13. Gjøsteen, K., Jager, T.: Practical and tightly-secure digital signatures and authenticated key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 95–125. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_4
- Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (Apr 1988)
- 15. Guillou, L.C., Quisquater, J.J.: A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO'88. LNCS, vol. 403, pp. 216–231. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34799-2_16
- 16. Kiltz, E., Masny, D., Pan, J.: Optimal security proofs for signatures from identification schemes. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016,

- Part II. LNCS, vol. 9815, pp. 33–61. Springer, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53008-5_2
- 17. Kondi, Y., shelat, a.: Improved straight-line extraction in the random oracle model with applications to signature aggregation. In: Agrawal, S., Lin, D. (eds.) ASI-ACRYPT 2022, Part II. LNCS, vol. 13792, pp. 279–309. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22966-4_10
- 18. Kondi, Y., shelat, a.: Improved straight-line extraction in the random oracle model with applications to signature aggregation. Cryptology ePrint Archive, Report 2022/393 (2022), https://eprint.iacr.org/2022/393
- Lai, Y.F.: Capybara and tsubaki: Verifiable random functions from group actions and isogenies. IACR Communications in Cryptology (CiC) 1(3), 1 (2024). https://doi.org/10.62056/avr-11zn4
- 20. Lee, Y., Park, J.H., Lee, K., Lee, D.H.: Tight security for the generic construction of identity-based signature (in the multi-instance setting). Theoretical Computer Science 847, 122–133 (2020). https://doi.org/https://doi.org/10.1016/j.tcs.2020.09.044
- 21. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 31–53. Springer, Berlin, Heidelberg (Aug 1993). https://doi.org/10.1007/3-540-48071-4_3
- Pan, J., Qian, C., Ringerud, M.: Signed Diffie-Hellman key exchange with tight security. In: Paterson, K.G. (ed.) CT-RSA 2021. LNCS, vol. 12704, pp. 201–226.
 Springer, Cham (May 2021). https://doi.org/10.1007/978-3-030-75539-3_9
- 23. Pan, J., Ringerud, M.: Signatures with tight multi-user security from search assumptions. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) ES-ORICS 2020, Part II. LNCS, vol. 12309, pp. 485–504. Springer, Cham (Sep 2020). https://doi.org/10.1007/978-3-030-59013-0_24
- 24. Pan, J., Wagner, B.: Lattice-based signatures with tight adaptive corruptions and more. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022, Part II. LNCS, vol. 13178, pp. 347–378. Springer, Cham (Mar 2022). https://doi.org/10.1007/978-3-030-97131-1_12
- Preneel, B. (ed.): CT-RSA 2002, LNCS, vol. 2271. Springer, Berlin, Heidelberg (Feb 2002)
- 26. Stolbunov, A.: Cryptographic Schemes Based on Isogenies. Ph.D. University thesis, Norwegian of Science and Technology (2012),https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/262577
- 27. Wu, G., Lai, J.C., Guo, F.C., Susilo, W., Zhang, F.T.: Tightly secure public-key cryptographic schemes from one-more assumptions. Journal of Computer Science and Technology 34, 1366–1379 (2019). https://doi.org/10.1007/s11390-019-1980-2
- 28. Yoshioka, H., Ogata, W., Hashimoto, K.: Towards a tightly secure signature in multi-user setting with corruptions based on search assumptions. Cryptology ePrint Archive, Report 2024/1286 (2024), https://eprint.iacr.org/2024/1286

A Proof of Theorem 2

Proof. Let \mathcal{A} be a PPT adversary against the MU-SUF-CMA-C security of SIG[ID] and \mathcal{C} be a challenger managing the security game. To prove the theorem, we consider the following sequence of games.

Game₀. This is the original MU-SUF-CMA-C game. By definition, we have

$$\Pr[\mathsf{Game}_0 \Rightarrow 1] = \mathsf{Adv}^{\mathsf{MU-SUF-CMA-C}}_{\mathcal{A},\mathsf{SIG[ID]}}(\lambda).$$

In the following, for a query $(pk, m, \overrightarrow{com}, j, ch, resp)$ to the RO, we call $(pk, m, \overrightarrow{com})$ a prefix of the query. If $V(pk, com_j, ch, resp) = 1$, the query is called a "valid transcript", otherwise, it is called an "invalid transcript".

<u>Game_1</u>. In this game, \mathcal{C} simulates RO using three lists $L_{\mathsf{RO}}^{\mathsf{valid}}$, $L_{\mathsf{RO}}^{\mathsf{invalid}}$, $L_{\mathsf{RO}}^{\mathsf{ignore}}$, and simulates $\mathsf{OSign}(i,m)$ as in Figure 6 (without boxed lines). In the simulation of OSign , \mathcal{C} computes the responses and the hash values for all j and ch instead of computing them individually.

On the other hand, \mathcal{C} simulates $\mathsf{RO}(\mathsf{pk}, m, \mathsf{com}, j, \mathsf{ch}, \mathsf{resp})$ as follows: \mathcal{C} first checks if the same query has already been issued based on $L_{\mathsf{RO}} := L_{\mathsf{RO}}^{\mathsf{valid}} \cup L_{\mathsf{RO}}^{\mathsf{invalid}} \cup L_{\mathsf{RO}}^{\mathsf{ignore}}$. If so, \mathcal{C} returns the consistent value. Otherwise, chooses $h \leftarrow \$ \{0,1\}^{\gamma}$, adds a query-answer tuple $(\mathsf{pk}, m, \mathsf{com}, j, \mathsf{ch}, \mathsf{resp}, h)$ to one of three lists according to the following conditions, and returns h as the hash value.

- If the query is issued by \mathcal{A} : if it is a valid transcript, the tuple $(\mathsf{pk}, m, \mathsf{com}, j, \mathsf{ch}, \mathsf{resp}, h)$ is added to $L^{\mathrm{valid}}_{\mathsf{RO}}$. Otherwise, if it is an invalid transcript, the tuple is added to $L^{\mathrm{invalid}}_{\mathsf{RO}}$.
- If the query $(pk_i, m, \overrightarrow{com}, j, ch, resp)$ is issued internally in the simulation of $\mathsf{OSign}(i, m)$:
 - if $\vec{\mathsf{com}}$ and $(\mathsf{com}_j, j, \mathsf{ch}, \mathsf{resp})$ are used in the signature, then the tuple $(\mathsf{pk}_i, m, \vec{\mathsf{com}}, j, \mathsf{ch}, \mathsf{resp}, h)$ is added to $L^{\mathrm{valid}}_{\mathsf{RO}}$,
 - if \vec{com} is used in the signature, but $(com_j, j, ch, resp)$ is not, then the tuple is added to L_{RO}^{ignore} ,
 - if common is not used in the signature, 2^t queries (pfx, j, ch', resp', h') with the same (pfx, j) must be asked. From these queries, one query and its answer, chosen at random in Line 29, is added to L^{valid} , and the rest are added to L^{ignore} .
- When \mathcal{C} verifies the forged signature, we consider \mathcal{A} makes hash queries $(\mathsf{pk}_{i^*}, m^*, \mathsf{com}^*, j, \mathsf{ch}_j^*, \mathsf{resp}_j^*)$ for all j. $L_{\mathsf{RO}}^{\mathsf{valid}}$ or $L_{\mathsf{RO}}^{\mathsf{invalid}}$ are updated depending on $\mathsf{V}(\mathsf{pk}_{i^*}, \mathsf{com}_j^*, \mathsf{ch}_j^*, \mathsf{resp}_j^*) = 1$ or not.

The function UpdateLists (Figure 7) specifies the concrete way to update $L_{\mathsf{RO}}^{\mathsf{valid}}$ and $L_{\mathsf{RO}}^{\mathsf{ignore}}$. Note that the way $\mathsf{ch}_{\tau,j}$ is determined depends on $\tau = \hat{\tau}$ or not. From this process, for any $(\mathsf{pk}_i, m, \mathsf{com}, j)$, at most one tuple $(\mathsf{pk}_i, m, \mathsf{com}, j, *, *, *)$ is added to $L_{\mathsf{RO}}^{\mathsf{valid}}$ in the process of OSign.

This change does not affect A's view. Therefore, we have

$$\Pr[\mathsf{Game}_1 \Rightarrow 1] = \Pr[\mathsf{Game}_0 \Rightarrow 1].$$

<u>Game_2</u>. In this game, the simulation of OSign is further changed as follows. After setting pfx_{τ} in line 9 of Figure 6, $\mathcal C$ browses the list $L^{\mathrm{valid}}_{\mathsf{RO}}$ to find a tuple whose prefix is pfx_{τ} . If such a tuple exists, skip Lines 11–31, i.e., move on to the next τ . Otherwise, $\mathcal C$ proceeds the signing process as in $\mathsf{Game_1}$.

```
\mathsf{OSign}(i, m)
           if i \in L_{\mathsf{corr}} then
 1:
                 \mathbf{return} \perp
 2:
            endif
            foreach \tau \in [T] do
  4:
                 foreach j \in [\rho] do
  5:
                      (\mathsf{com}_{\tau,j}, \mathsf{st}_{\tau,j}) \leftarrow \mathsf{P}_1(\mathsf{par}, \mathsf{pk}_i, \mathsf{sk}_i)
 6:
 7:
                 \vec{\mathsf{com}}_{	au} \coloneqq (\mathsf{com}_{	au,1}, \dots, \mathsf{com}_{	au,
ho})
  8:
                 \mathsf{pfx}_\tau \coloneqq (\mathsf{pk}_i, m, \vec{\mathsf{com}}_\tau)
 9:
                  if (\mathsf{pfx}_{\tau}, *, *, *) \notin L_{\mathsf{RO}}^{\mathrm{valid}} then
                                                                                            /\!\!/ Check if \mathsf{pfx}_\tau was not generated before
10:
                     for
each j \in [\rho] do
11:
                           foreach ch \in \{0,1\}^t do
12:
                                \mathsf{resp}_{\tau,j,\mathsf{ch}} \leftarrow \mathsf{P}_2(\mathsf{ch},\mathsf{st}_{\tau,j})
13:
14:
                               h_{\tau,j,\mathsf{ch}} \leftarrow \mathsf{H}(\mathsf{pfx}_{\tau}, j, \mathsf{ch}, \mathsf{resp}_{\tau,j,\mathsf{ch}})
                           endfor
15:
                      endfor
16:
17:
                     if \forall j \; \exists \mathsf{ch} : \; h_{\tau,j,\mathsf{ch}} = 0^{\gamma} \; \mathsf{then}
                           for
each j \in [\rho] do
18:
                               let \pi be a random permutation over \{0,1\}^t
19:
                               k \coloneqq \pi \left( \min_{k \in \{0,1\}^t} \{ k \mid h_{\tau,j,\pi(k)} = 0^{\gamma} \} \right)
20:
                                \mathsf{ch}_{\tau,i} := k
21:
22:
                                \mathsf{UpdateLists}(\mathsf{pfx}_{\tau}, j, \{\mathsf{resp}_{\tau, j, \mathsf{ch}}, h_{\tau, j, \mathsf{ch}}\}_{\mathsf{ch} \in \{0, 1\}^t}, \mathsf{ch}_{\tau, j})
23:
                           endfor
24:
                           \hat{\tau} \coloneqq \tau
                           \sigma \coloneqq (\mathsf{com}_{\hat{\tau},j}, \mathsf{ch}_{\hat{\tau},j}, \mathsf{resp}_{\hat{\tau},j,\mathsf{ch}_{\hat{\tau},j}})_{j \in [\rho]}
25:
26:
                           break
27:
                      else
                           for
each j \in [\rho] do
28:
                               \mathsf{ch}_{\tau,j} \leftarrow \$ \left\{0,1\right\}^t
29:
                                \mathsf{UpdateLists}(\mathsf{pfx}_\tau, j, \{\mathsf{resp}_{\tau, j, \mathsf{ch}}, h_{\tau, j, \mathsf{ch}}\}_{\mathsf{ch} \in \{0, 1\}^t}, \mathsf{ch}_{\tau, j})
30:
31:
                      endif
                 endif
32:
33:
                 if \tau = T then \sigma \coloneqq \bot
            endfor
34:
            L_{\text{sig}} \coloneqq L_{\text{sig}} \cup \{(i, m, \sigma)\}
35:
           return \sigma
```

Fig. 6: The sign oracle in Game_1 (without boxed lines) and Game_2 (with boxed lines).

Fig. 7: Function UpdateLists used in OSign.

Because honestly generated commitments $\vec{\text{com}}$ has $\rho\kappa$ min-entropy and the number of varieties of $\vec{\text{com}}$ in $L_{\text{RO}}^{\text{valid}}$ is at most $Q_{\text{RO}} + T \cdot Q_{\text{SIG}}$, the difference between Game_1 and Game_2 is upper bounded by $T \cdot Q_{\text{SIG}} \times (Q_{\text{RO}} + T \cdot Q_{\text{SIG}})/2^{\rho\kappa}$. Therefore,

```
|\Pr[\mathsf{Game}_2 \Rightarrow 1] - \Pr[\mathsf{Game}_1 \Rightarrow 1]| \le T \cdot Q_{\mathsf{SIG}}(Q_{\mathsf{RO}} + T \cdot Q_{\mathsf{SIG}})2^{-\rho\kappa}.
```

<u>Game_3.</u> In this game, we introduce a flag F_{find} which is initialized as $F_{\text{find}} := \text{false}$. The simulation of RO is changed as follows: When $(\mathsf{pk}_i, m, \mathsf{com}, j, \mathsf{ch}, \mathsf{resp}, h)$ is added to $L^{\text{valid}}_{\text{RO}}$ as the response of \mathcal{A} 's query and $i \notin L_{\text{corr}}$ and $F_{\text{find}} = \text{false}$ hold, \mathcal{C} browses the list $L^{\text{valid}}_{\text{RO}}$ and finds a tuple $(\mathsf{pk}_i, m, \mathsf{com}, j, \mathsf{ch}', \mathsf{resp}', h')$ such that $(\mathsf{ch}', \mathsf{resp}') \neq (\mathsf{ch}, \mathsf{resp})$. If such a tuple exists, \mathcal{C} sets

$$pair := (pk_i, com_j, ch, resp, ch', resp'),$$

 $F_{find} := true.$

Further, we add a condition " $F_{\text{find}} = \text{true}$ " to the requirements that the game outputs 1.

Because \mathcal{A} cannot see F_{find} and pair, the change of RO simulation does not affect \mathcal{A} 's view. Thus, $|\Pr[\mathsf{Game}_3 \Rightarrow 1] - \Pr[\mathsf{Game}_2 \Rightarrow 1]|$ is bounded by the probability that \mathcal{A} 's final outputs is valid and $F_{\mathrm{find}} = \mathsf{false}$ at the end of Game_2 . Let Lucky be the event that both of these two conditions hold. We will evaluate $\Pr[\mathsf{Lucky}]$ in the following three cases. Let $\sigma^* = (\mathsf{com}_j^*, \mathsf{ch}_j^*, \mathsf{resp}_j^*)_{j \in [\rho]}$ be \mathcal{A} 's forged signature, and $\mathsf{pfx}^* \coloneqq (\mathsf{pk}_{i^*}, m^*, \mathsf{com}^*)$.

- (1) pfx* was used in OSign and the oracle returned a signature including it.
- (2) pfx* was used in OSign but the oracle returned a signature that does not include pfx*.
- (3) pfx* was not used in OSign.

Due to the condition that F_{find} is set to true and the fact that $L_{\text{RO}}^{\text{valid}}$ includes all transcripts in the signatures returned from OSign and one transcript per each j whose prefix was considered in OSign but discarded, in cases (1) and (2),

query-answer tuples $(\mathsf{pfx}^*, j, \mathsf{ch}_j^*, \mathsf{resp}_j^*, h_j)$ for all j should have been added in the process of OSign. In case (1), if all transcripts in σ^* are valid and $F_{\text{find}} = \mathsf{false}$, then $(i^*, m^*, \sigma^*) \in L_{\mathsf{SIG}}$ holds. Therefore, $\Pr[\mathsf{Lucky}] = 0$. In case (2), if all transcripts in σ^* are valid and $F_{\mathsf{find}} = \mathsf{false}$, then there should exist j such that $h_j = \mathsf{H}(\mathsf{pfx}^*, j, \mathsf{ch}_j^*, \mathsf{resp}_j^*) \neq 0^\gamma$. Therefore, $\Pr[\mathsf{Lucky}] = 0$.

On the other hand, in case (3), the event Lucky occurs only if \mathcal{A} obtains valid transcripts $(\mathsf{com}_j^*, \mathsf{ch}_j, \mathsf{resp}_j)$ such that $h_j = \mathsf{H}(\mathsf{pk}_{i^*}, m^*, \mathsf{com}^*, j, \mathsf{ch}_j, \mathsf{resp}_j) = 0^\gamma$ for all $j \in [\rho]$ with a single hash computation. Since hash values are chosen independently and uniformly at random from $\{0,1\}^\gamma$, the probability $h_j = 0^\gamma$ is $2^{-\gamma}$ for each $j \in [\rho]$. Thus, for a fixed $(\mathsf{pk}_{i^*}, m^*, \mathsf{com}^*)$, the probability $h_j = 0^\gamma$ for all $j \in [\rho]$ is bounded by $2^{-\rho\gamma}$. Since \mathcal{A} issues Q_{RO} RO queries with any prefix $(\mathsf{pk}, m, \mathsf{com})$ and finally outputs $(\mathsf{pk}_{i^*}, m^*, \mathsf{com}^*)$ as a part of the forged signature, $\Pr[\mathsf{Lucky}]$ is bounded by $(Q_{\mathsf{RO}} + 1)/2^{\rho\gamma}$, and we have

$$|\Pr[\mathsf{Game}_3 \Rightarrow 1] - \Pr[\mathsf{Game}_2 \Rightarrow 1]| \le (Q_{\mathsf{RO}} + 1)2^{-\rho\gamma}.$$

In the following, let \hat{i} be the index of the first element of $pair = (pk_{\hat{i}}, \ldots)$. Game₄. In this game, \mathcal{C} computes the following when F_{find} is set to true.

$$\mathsf{sk}^* \leftarrow \mathsf{Ext}(pair) = \mathsf{Ext}(\mathsf{pk}_{\hat{i}}, \mathsf{com}_{i}, \mathsf{ch}_{i}, \mathsf{resp}_{i}, \mathsf{ch}_{i}', \mathsf{resp}_{i}').$$

Further, we add a condition "VerKey(par, $pk_{\hat{i}}$, sk^*) = 1" to the requirements that the game outputs 1. (Note that if $F_{\text{find}} = \text{false}$, the game outputs 0.)

 $|\Pr[\mathsf{Game}_4 \Rightarrow 1] - \Pr[\mathsf{Game}_3 \Rightarrow 1]|$ is bounded by the probability that pair is assigned but the extractor Ext fails to find a valid secret key. We will evaluate this probability of failure. From the condition of setting F_{find} , $pair = (\mathsf{pk}_{\hat{i}}, \mathsf{com}_j, \mathsf{ch}_j, \mathsf{resp}_j, \mathsf{ch}_j', \mathsf{resp}_j')$ satisfies

$$\mathsf{V}(\mathsf{pk}_{\hat{i}},\mathsf{com}_{j},\mathsf{ch}_{j},\mathsf{resp}_{j}) = \mathsf{V}(\mathsf{pk}_{\hat{i}},\mathsf{com}_{j},\mathsf{ch}_{j}',\mathsf{resp}_{j}') = 1$$

and

$$(\mathsf{ch}_j, \mathsf{resp}_j) \neq (\mathsf{ch}'_j, \mathsf{resp}'_j).$$

Therefore, the probability of this failure is at most $Adv_{\mathcal{B}_1,|D}^{SSS}(\lambda)$ for some algorithm \mathcal{B}_1 . Formally, we can construct an adversary \mathcal{B}_1 that breaks the strong special soundness by using \mathcal{A} as follows: Upon receiving a parameter par, \mathcal{B}_1 simulates Game_2 against \mathcal{A} . If \mathcal{B}_1 obtains pair, it outputs pair. We can see \mathcal{B}_1 breaks the strong special soundness when Ext fails extraction. Thus, we have

$$|\Pr[\mathsf{Game}_4 \Rightarrow 1] - \Pr[\mathsf{Game}_3 \Rightarrow 1]| \leq \mathsf{Adv}_{\mathcal{B}_1,\mathsf{ID}}^{\mathsf{SSS}}(\lambda).$$

We also evaluate the running time of \mathcal{B}_1 . It executes \mathcal{A} once and answers oracle queries from \mathcal{A} . Since each oracle query can be answered in the time of $\mathsf{poly}(\lambda)$, the running time of \mathcal{B}_1 is $T_{\mathcal{B}_1} = T_{\mathcal{A}} + (Q_{\mathsf{RO}} + Q_{\mathsf{sig}} + Q_{\mathsf{corr}}) \cdot \mathsf{poly}(\lambda)$, where $Q_{\mathsf{RO}}, Q_{\mathsf{sig}}, Q_{\mathsf{corr}}$ denote the maximum number of each oracle query \mathcal{A} makes. Since $Q_{\mathsf{RO}}, Q_{\mathsf{sig}}, Q_{\mathsf{corr}}$ are about $\mathcal{O}(T_{\mathcal{A}})$, we conclude that $T_{\mathcal{B}_1} = \mathcal{O}(T_{\mathcal{A}})$.

<u>Games.</u> This game outputs 1 if $(F_{\text{find}} = \text{true}) \land (\text{VerKey}(\text{par}, \text{pk}_{\hat{i}}, \text{sk}^*) = 1)$ holds regardless of whether the forgery was successful or not.

Since the conditions ($F_{\text{find}} = \text{true}$) and (VerKey(par, pk₁, sk*) = 1) are already included in the requirements that Game₃ outputs 1, clearly,

$$\Pr[\mathsf{Game}_5 \Rightarrow 1] \ge \Pr[\mathsf{Game}_4 \Rightarrow 1]$$

holds.

 $\frac{\mathsf{Game}_{5'}.}{1} \text{ We add a condition } \mathsf{sk}^* \neq \mathsf{sk}_{\hat{i}} \text{ to the requirements that the game outputs}$

Intuitively, from the perfect HVZK of ID, \mathcal{A} cannot know which secret key \mathcal{C} has among K secret keys corresponding pk_i . Therefore, we hope

$$\Pr[\mathsf{Game}_{5'} \Rightarrow 1] pprox rac{K-1}{K} \Pr[\mathsf{Game}_5 \Rightarrow 1].$$

Actually, we can prove the following lemma.

Lemma 1. If ID is perfect HVZK,

$$\Pr[\mathsf{Game}_{5'} \Rightarrow 1] = \frac{K-1}{K} \Pr[\mathsf{Game}_5 \Rightarrow 1].$$

Before proving Lemma 1, we will upper-bound $\Pr[\mathsf{Game}_{5'} \Rightarrow 1]$ and conclude the proof. We can construct an adversary \mathcal{B}_2 against the second key recovery resistance of ID using \mathcal{A} . \mathcal{B}_2 receives a public parameter par and key pairs of ID $\{(\mathsf{pk}_i,\mathsf{sk}_i)\}_{i\in[N]}$. It sets $(\mathsf{svk}_i,\mathsf{ssk}_i) \coloneqq (\mathsf{pk}_i,(\mathsf{pk}_i,\mathsf{sk}_i))$, initializes the lists $L^{\mathrm{valid}}_{\mathsf{RO}}, L^{\mathrm{invalid}}_{\mathsf{RO}}, L^{\mathrm{ignore}}_{\mathsf{RO}}$, L_{corr} and L_{sig} and executes \mathcal{A} on input $(\mathsf{par}, \{\mathsf{svk}_i\}_{i\in[N]})$ and answers \mathcal{A} 's oracle queries as in $\mathsf{Game}_{5'}$. If \mathcal{B}_2 obtains a $\mathsf{pk}_{\hat{i}}$'s valid secret key $\mathsf{sk}^*(\neq \mathsf{sk}_{\hat{i}})$, \mathcal{B}_2 outputs (\hat{i},sk^*) .

We can verify that \mathcal{B}_2 perfectly simulates $\mathsf{Game}_{5'}$ against \mathcal{A} . In addition, due to the modifications we made in the previous games, $\mathsf{Game}_{5'}$ outputs 1 only if $F_{\mathrm{find}} = \mathsf{true}$ and sk^* is a valid secret key w.r.t. $\mathsf{pk}_{\hat{\imath}}$ that is different from $\mathsf{sk}_{\hat{\imath}}$. Thus, \mathcal{B}_2 breaks the second key recovery resistance in the multi-user setting of ID. Therefore, we have

$$\Pr[\mathsf{Game}_{5'} \Rightarrow 1] \leq \mathsf{Adv}_{\mathcal{B}_2,\mathsf{ID}}^{2^{\mathsf{nd}}\mathsf{KR}}(\lambda).$$

We also evaluate the running time of \mathcal{B}_2 . It executes \mathcal{A} once and answers oracle queries from \mathcal{A} . Since each oracle query can be answered in the time of $\mathsf{poly}(\lambda)$, the running time of \mathcal{B}_2 is $T_{\mathcal{B}_2} = T_{\mathcal{A}} + (Q_{\mathsf{RO}} + Q_{\mathsf{sig}} + Q_{\mathsf{corr}}) \cdot \mathsf{poly}(\lambda)$, where $Q_{\mathsf{RO}}, Q_{\mathsf{sig}}, Q_{\mathsf{corr}}$ denote the maximum number of each oracle query \mathcal{A} made. Since $Q_{\mathsf{RO}}, Q_{\mathsf{sig}}, Q_{\mathsf{corr}}$ are about $\mathcal{O}(T_{\mathcal{A}})$, we conclude that $T_{\mathcal{B}_2} = \mathcal{O}(T_{\mathcal{A}})$.

Combining everything, we have

$$\begin{split} \mathsf{Adv}^{\mathsf{MU-SUF-CMA-C}}_{\mathcal{A},\mathsf{SIG[ID]}}(\lambda) & \leq \mathsf{Adv}^{\mathsf{SSS}}_{\mathcal{B}_1,\mathsf{ID}}(\lambda) + \frac{K}{K-1} \mathsf{Adv}^{2^{\mathsf{nd}}}_{\mathcal{B}_2,\mathsf{ID}}^{\mathsf{KR}}(\lambda) \\ & + \frac{(Q_{\mathsf{RO}}+1)}{2^{\rho\kappa}} + \frac{T \cdot Q_{\mathsf{sig}}(Q_{\mathsf{RO}} + T \cdot Q_{\mathsf{sig}})}{2^{\rho\gamma}}. \end{split}$$

Our remaining task is proving Lemma 1. To do so, we use two game sequences started at Game_5 and $\mathsf{Game}_{5'}$. In the following, $\mathsf{Game}_{X'}$ is exactly the same as Game_X except the condition $\mathsf{sk}^* \neq \mathsf{sk}_{\hat{i}}$ is added to the requirements that the game outputs 1.

Game₆ and Game_{6'}. These games are the same as $Game_5$ and $Game_{5'}$ expect that line 13 and 14 in Figure 6 are replaced with

$$h_{\tau,j,\mathsf{ch}} \leftarrow \$ \left\{ 0,1 \right\}^{\gamma}$$

and the following lines are inserted before line 3 and line 5 in Figure 7:

$$\begin{split} \mathsf{resp}_{\tau,j,\mathsf{ch}} &\leftarrow \mathsf{P}_2(\mathsf{ch},\mathsf{st}_{\tau,j}) \\ \mathsf{H}(\mathsf{pfx}_{\tau},j,\mathsf{ch},\mathsf{resp}_{\tau,j,\mathsf{ch}}) &\coloneqq h_{\tau,j,\mathsf{ch}} \quad \# \; \mathsf{program} \; \mathsf{RO} \end{split}$$

This is a conceptual change. Thus, we have

$$\begin{split} \Pr[\mathsf{Game}_6 \Rightarrow 1] &= \Pr[\mathsf{Game}_5 \Rightarrow 1], \\ \Pr[\mathsf{Game}_{6'} \Rightarrow 1] &= \Pr[\mathsf{Game}_{5'} \Rightarrow 1]. \end{split}$$

<u>Game₇ and Game_{7'}.</u> In this game, we introduce another list, L^{tmp} . We delete the line before line $\frac{5}{5}$ that was added in Game₆, and replace line $\frac{5}{5}$ in Figure 7 with the next one.

add
$$(\mathsf{pfx}_{\tau}, j, \mathsf{ch}, \mathsf{resp}_{\tau, j, \mathsf{ch}}, h_{\tau, j, \mathsf{ch}})$$
 to L^{tmp}

Further, if $\mathsf{OCorr}(i)$ is queried, for each tuple $(\mathsf{pk}_i, m, \mathsf{com}, j, \mathsf{ch}, \mathsf{resp}, h) \in L^{\mathsf{tmp}}$, \mathcal{C} programs the RO as $\mathsf{H}(\mathsf{pk}_i, m, \mathsf{com}, j, \mathsf{ch}, \mathsf{resp}) \coloneqq h$ and moves the tuple to $L^{\mathsf{ignore}}_{\mathsf{RO}}$. That is, \mathcal{C} programs only one hash value for each (pfx_τ, j) at the time of OSign simulation, and others will be programmed when sk_i is revealed.

 \mathcal{A} 's views in Game_6 and Game_7 (resp. $\mathsf{Game}_{6'}$ and $\mathsf{Game}_{7'}$) are exactly the same until \mathcal{A} makes a query whose hash value is programmed in Game_6 (resp. $\mathsf{Game}_{6'}$) but not in Game_7 (resp. $\mathsf{Game}_{7'}$). Let $(\mathsf{pfx}, j, \mathsf{ch'}, \mathsf{resp'})$ be such a query. Then, there must exist a pair $(\mathsf{ch''}, \mathsf{resp''})$ such that $(\mathsf{pfx}, j, \mathsf{ch''}, \mathsf{resp''})$ has been programmed and added to $L^{\mathrm{valid}}_{\mathsf{RO}}$, and $\mathsf{ch''} \neq \mathsf{ch'}$. This means that when \mathcal{A} issues such a query for the first time, F_{find} is set to true and pair and sk^* are assigned values. Therefore, the probability that $(F_{\mathrm{find}} = \mathsf{true}) \wedge (\mathsf{VerKey}(\mathsf{pk}_i, \mathsf{sk}^*) = 1)$ holds is the same in Game_6 (resp. $\mathsf{Game}_{6'}$) and Game_7 (resp. $\mathsf{Game}_{7'}$). Thus, we have

$$\begin{aligned} \Pr[\mathsf{Game}_7 \Rightarrow 1] &= \Pr[\mathsf{Game}_6 \Rightarrow 1], \\ \Pr[\mathsf{Game}_{7'} \Rightarrow 1] &= \Pr[\mathsf{Game}_{6'} \Rightarrow 1]. \end{aligned}$$

Game₈ and Game_{8'}. In this game, line 21 in Figure 6 is replaced with

where swap(a, b) exchanges the values of variables a and b.

The modification changes how the values of $(h_{\tau,j,\mathsf{ch}})_{\mathsf{ch}\in\{0,1\}^t}$ and $\mathsf{ch}_{\hat{\tau},j}$ are determined. However, we can prove that the distribution of $((h_{\tau,j,\mathsf{ch}})_{\mathsf{ch}\in\{0,1\}^t},\mathsf{ch}_{\hat{\tau},j})$ is not changed. It is enough to show the distributions of $((h_k)_{k\in\{0,1\}^t},\mathsf{ch})$ in Experiment A and B defined below are identical.

Experiment A: Experiment B:
$$\mathbf{h} = (h_k)_{k \in \{0,1\}^t} \leftarrow \$ (\{0,1\}^\gamma)^{2^t}$$
 if $h_k \neq 0^\gamma$ for all $k \in \{0,1\}^t$ return \bot ch $\leftarrow \$ \{k \in \{0,1\}^t \mid h_k = 0^\gamma\}$ output $(\mathbf{h}, \mathsf{ch})$
$$ch \leftarrow \$ \{k \in \{0,1\}^t \mid h_k = 0^\gamma\}$$
 output $(\mathbf{h}, \mathsf{ch})$
$$ch \leftarrow \$ \{0,1\}^t \mid h_k = 0^\gamma\}$$
 ch $\leftarrow \$ \{0,1\}^t \mid h_k' = 0^\gamma\}$ ch $\leftarrow \$ \{0,1\}^t \mid h_k' = 0^\gamma\}$ define $\mathbf{h} = (h_k)_{k \in \{0,1\}^t}$ as
$$\begin{cases} h'_{\mathsf{ch}} & \text{if } k = \mathsf{ch}' \\ h'_{\mathsf{ch}} & \text{otherwise} \end{cases}$$
 output $(\mathbf{h}, \mathsf{ch})$

In the following, if $h_k = 0^{\gamma}$ for some $k \in \{0,1\}^t$, we say "**h** is good", and define $W(\mathbf{h}) := |\{k \in \{0,1\}^t \mid h_k = 0^{\gamma}\}|$. Then, we have the following facts:

- The probability that good \mathbf{h} is chosen in Experient A and that good \mathbf{h}' is chosen in Experiment B are the same.
- For any good $\hat{\mathbf{h}}$, the probability that $\mathbf{h} = \hat{\mathbf{h}}$ holds in Experiment A and that $\mathbf{h}' = \hat{\mathbf{h}}$ holds in Experiment B are constant, say p.
- In Experiment A, for any good $\hat{\mathbf{h}}$ and any $\hat{\mathbf{ch}} \in \{0,1\}^t$, it holds that

$$\Pr[(\mathbf{h},\mathsf{ch}) = (\hat{\mathbf{h}},\hat{\mathsf{ch}})] = \begin{cases} \frac{p}{W(\hat{\mathbf{h}})} & \text{if } \hat{h}_{\hat{\mathsf{ch}}} = 0^{\gamma} \\ 0 & \text{otherwise.} \end{cases}$$

- For any good $\hat{\mathbf{h}}$ and any $\hat{\mathbf{ch}}$, $\hat{\mathbf{ch}}' \in \{0,1\}^t$, define $\hat{\mathbf{h}}(\hat{\mathbf{ch}} \Leftrightarrow \hat{\mathbf{ch}}')$ as

$$\hat{h}(\hat{\mathsf{ch}} \Leftrightarrow \hat{\mathsf{ch}}')_k = \begin{cases} \hat{h}_{\hat{\mathsf{ch}}} & \text{if } k = \hat{\mathsf{ch}}' \\ \hat{h}_{\hat{\mathsf{ch}}'} & \text{if } k = \hat{\mathsf{ch}} \\ \hat{h}_k & \text{otherwise} \end{cases}$$

Then, in Experiment B, we have

$$\begin{split} &\Pr[(\mathbf{h},\mathsf{ch}) = (\hat{\mathbf{h}},\hat{\mathsf{ch}})] \\ &= \sum_{\hat{\mathsf{ch}}' \in \{0,1\}^t} \Pr[\mathbf{h}' = \hat{\mathbf{h}}(\hat{\mathsf{ch}} \Leftrightarrow \hat{\mathsf{ch}}') \wedge \mathsf{ch}' = \hat{\mathsf{ch}}'] \times \Pr[\mathsf{ch} = \hat{\mathsf{ch}}]. \end{split}$$

The first probability on the right side is estimated as above:

$$\Pr[\mathbf{h}' = \hat{\mathbf{h}}(\hat{\mathsf{ch}} \Leftrightarrow \hat{\mathsf{ch}}') \land \mathsf{ch}' = \hat{\mathsf{ch}}'] = \begin{cases} \frac{p}{W(\hat{\mathbf{h}})} & \text{if } \hat{h}_{\hat{\mathsf{ch}}} = 0^{\gamma} \\ 0 & \text{otherwise,} \end{cases}$$

since $W(\hat{\mathbf{h}}(\hat{\mathsf{ch}} \Leftrightarrow \hat{\mathsf{ch}}')) = W(\hat{\mathbf{h}})$ holds. On the other hand, the second probability on the right side is $\Pr[\mathsf{ch} = \hat{\mathsf{ch}}] = \frac{1}{2t}$. Thus,

$$\begin{split} \Pr[(\mathbf{h},\mathsf{ch}) = (\hat{\mathbf{h}},\hat{\mathsf{ch}})] &= \begin{cases} \sum_{\hat{\mathsf{ch}}' \in \{0,1\}^t} \frac{p}{W(\hat{\mathbf{h}})} \times \frac{1}{2^t} & \text{if } \hat{h}_{\hat{\mathsf{ch}}} = 0^{\gamma} \\ \sum_{\hat{\mathsf{ch}}' \in \{0,1\}^t} 0 \times \frac{1}{2^t} & \text{otherwise} \end{cases} \\ &= \begin{cases} \frac{p}{W(\hat{\mathbf{h}})} & \text{if } \hat{h}_{\hat{\mathsf{ch}}} = 0^{\gamma} \\ 0 & \text{otherwise}, \end{cases} \end{split}$$

which is the same as in Experiment A.

Therefore, we have

$$\begin{split} \Pr[\mathsf{Game}_8 \Rightarrow 1] &= \Pr[\mathsf{Game}_7 \Rightarrow 1], \\ \Pr[\mathsf{Game}_{8'} \Rightarrow 1] &= \Pr[\mathsf{Game}_{7'} \Rightarrow 1]. \end{split}$$

Now, $\mathsf{ch}_{\tau,j}$ is chosen randomly and uniformly independent from the choice of $h_{\tau,j,\mathsf{ch}}$ and the value of $\hat{\tau}$.

Game₉ and Game_{9'}. In this game, OSign and UpdateLists are further modified as in Figure 8 and Figure 9, respectively. The difference is that the challenge $ch_{\tau,j}$ is chosen and $resp_{\tau,j,ch}$ for $ch = ch_{\tau,j}$ is calculated immediately after $com_{\tau,j}$ is computed.

This change is completely conceptual. Thus, we have

$$\begin{split} \Pr[\mathsf{Game}_9 \Rightarrow 1] &= \Pr[\mathsf{Game}_8 \Rightarrow 1], \\ \Pr[\mathsf{Game}_{9'} \Rightarrow 1] &= \Pr[\mathsf{Game}_{8'} \Rightarrow 1]. \end{split}$$

 $\underline{\mathsf{Game}_{10}}$ and $\underline{\mathsf{Game}_{10'}}$. Now, let consider the following (possibly inefficient) function ReSim.

- ReSim takes a valid key pair (pk, sk) and a valid transcript $(com, ch, resp) \in Tran(pk, sk, ch)$ as input,
- chooses $r \leftarrow \$ \{r \mid (\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow \mathsf{Tran}(\mathsf{pk}, \mathsf{sk}, \mathsf{ch}; r)\}$, where r is the random coins used to compute P_1 in Tran algorithm,
- regenerates (com, st) $\leftarrow P_1(pk, sk; r)$,
- computes $\operatorname{resp'_{ch'}} \leftarrow \mathsf{P}_2(\operatorname{\mathsf{st}},\operatorname{\mathsf{ch'}})$ for all $\operatorname{\mathsf{ch'}} \in \{0,1\}^t \setminus \{\operatorname{\mathsf{ch}}\}$, and
- outputs a list $(\operatorname{resp'_{ch'}})_{\operatorname{ch'} \in \{0,1\}^t \setminus \{\operatorname{ch}\}}$.

If the identification scheme is perfect HVZK, there exists r such that $(com, ch, resp) \leftarrow Tran(pk, sk, ch; r)$ for any transcript (com, ch, resp) simulated by Sim and for any valid sk. Thus, ReSim can generate the response resp' that would be computed by an honest prover having the specified secret key sk as the response to the different challenge ch'.

In games Game_{10} and $\mathsf{Game}_{10'}$, lines 6 to 8 in Figure 8 are replaced with the following.

$$\begin{aligned} \mathsf{ch}_{\tau,j} &\leftarrow \!\! \$ \left\{ 0,1 \right\}^t \\ &\left(\mathsf{com}_{\tau,j}, \mathsf{resp}_{\tau,j,\mathsf{ch}_{\tau,j}} \right) \leftarrow \mathsf{Sim}(\mathsf{pk}_i, \mathsf{ch}_{\tau,j}) \end{aligned}$$

```
\mathsf{OSign}(i, m)
 1: if i \in L_{corr} then
                 return \perp
           endif
           for
each \tau \in [T] do
                 foreach j \in [\rho] do
 5:
                     (\mathsf{com}_{\tau,j},\mathsf{st}_{\tau,j}) \leftarrow \mathsf{P}_1(\mathsf{par},\mathsf{pk}_i,\mathsf{sk}_i)
 6:
                     \mathsf{ch}_{\tau,j} \leftarrow \$ \{0,1\}^t
 7:
                     \mathsf{resp}_{\tau,j,\mathsf{ch}_{\tau,j}} \leftarrow \mathsf{P}_2(\mathsf{ch}_{\tau,j},\mathsf{st}_{\tau,j})
 8:
                 endfor
 9:
10:
                \vec{\mathsf{com}}_{\tau} \coloneqq (\mathsf{com}_{\tau,1}, \dots, \mathsf{com}_{\tau,\rho})
                \mathsf{pfx}_\tau \coloneqq (\mathsf{pk}_i, m, \vec{\mathsf{com}}_\tau)
11:
                if (\mathsf{pfx}_{\tau}, *, *, *) \notin L_{\mathsf{RO}}^{\mathrm{valid}} then
                                                                                   /\!\!/ Check if \mathsf{pfx}_\tau was not generated before
12:
                     for
each j \in [\rho] do
13:
                          foreach ch \in \{0,1\}^t do
14:
                               h_{\tau,j,\mathrm{ch}} \leftarrow \$ \{0,1\}^{\gamma}
15:
                          endfor
16:
                     endfor
17:
                     if \forall j \; \exists \mathsf{ch} : \; h_{\tau,j,\mathsf{ch}} = 0^{\gamma} \; \mathsf{then}
18:
                          for
each j \in [\rho] do
19:
                               let \pi be a random permutation over \{0,1\}^t
20:
                               k := \pi \left( \min_{k \in \{0,1\}^t} \{ k \mid h_{\tau,j,\pi(k)} = 0^\gamma \} \right)
21:
                               swap(h_{\tau,j,k},h_{\tau,j,\mathsf{ch}_{\tau,j}})
                               \mathsf{UpdateLists}(\mathsf{pfx}_\tau, j, \{\mathsf{resp}_{\tau, j, \mathsf{ch}}, h_{\tau, j, \mathsf{ch}}\}_{\mathsf{ch}}, \mathsf{ch}_{\tau, j})
23:
24:
                          endfor
                          \hat{\tau}\coloneqq \tau
25:
                          \sigma \coloneqq (\mathsf{com}_{\hat{\tau},j}, \mathsf{ch}_{\hat{\tau},j}, \mathsf{resp}_{\hat{\tau},j,\mathsf{ch}_{\hat{\tau},i}})_{j \in [\rho]}
27:
                          break
28:
                     else
                          for
each j \in [\rho] do
29:
                               \mathsf{UpdateLists}(\mathsf{pfx}_{\tau}, j, \{\mathsf{resp}_{\tau, j, \mathsf{ch}}, h_{\tau, j, \mathsf{ch}}\}_{\mathsf{ch}}, \mathsf{ch}_{\tau, j})
31:
                     endif
                 endif
32:
                 if \tau = T then \sigma := \bot
33:
           endfor
34:
         L_{\mathsf{sig}} \coloneqq L_{\mathsf{sig}} \cup \{(i, m, \sigma)\}
36: return \sigma
```

Fig. 8: The sign oracle in Game₉ and Game₉.

```
\begin{split} & \frac{\mathsf{UpdateLists}(\mathsf{pfx}_{\tau}, j, \{\mathsf{resp}_{\tau, j, \mathsf{ch}}, h_{\tau, j, \mathsf{ch}}\}_{\mathsf{ch}}, \mathsf{ch}_{\tau, j})}{1: & \mathbf{foreach} \ \mathsf{ch} \in \{0, 1\}^t \\ 2: & \mathbf{if} \ \mathsf{ch} = \mathsf{ch}_{\tau, j} \ \mathbf{then} \\ 3: & \mathsf{H}(\mathsf{pfx}_{\tau}, j, \mathsf{ch}, \mathsf{resp}_{\tau, j, \mathsf{ch}}) \coloneqq h_{\tau, j, \mathsf{ch}} \ /\!\!/ \ \mathsf{program} \ \mathsf{RO} \\ 4: & \mathrm{add} \ (\mathsf{pfx}_{\tau}, j, \mathsf{ch}, \mathsf{resp}_{\tau, j, \mathsf{ch}}, h_{\tau, j, \mathsf{ch}}) \ \mathsf{to} \ L^{\mathrm{valid}}_{\mathsf{RO}} \\ 5: & \mathbf{else} \\ 6: & \mathsf{resp}_{\tau, j, \mathsf{ch}} \leftarrow \mathsf{P}_2(\mathsf{ch}, \mathsf{st}_{\tau, j}) \\ 7: & \mathrm{add} \ (\mathsf{pfx}_{\tau}, j, \mathsf{ch}, \mathsf{resp}_{\tau, j, \mathsf{ch}}, h_{\tau, j, \mathsf{ch}}) \ \mathsf{to} \ L^{\mathrm{tmp}} \\ 8: & \mathbf{endif} \\ 9: & \mathbf{endfor} \end{split}
```

Fig. 9: Function UpdateLists used in Game₉.

Further, insert the next line

$$\{\mathsf{resp}_\mathsf{ch}'\}_{\mathsf{ch} \in \{0,1\}^t \setminus \{\mathsf{ch}_{\tau,j}\}} \leftarrow \mathsf{ReSim}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{com}_{\tau,j}, \mathsf{ch}_{\tau,j}, \mathsf{resp}_{\tau,j, \mathsf{ch}_{\tau,j}})$$

at the beginning of Figure 9, and line 6 is replaced with

$$\mathsf{resp}_{\tau,j,\mathsf{ch}} \coloneqq \mathsf{resp}_\mathsf{ch}'$$

If the identification scheme is perfect HVZK, \mathcal{A} 's view does not change from the definition of ReSim. Thus,

$$\begin{split} \Pr[\mathsf{Game}_{10} \Rightarrow 1] &= \Pr[\mathsf{Game}_{9} \Rightarrow 1], \\ \Pr[\mathsf{Game}_{10'} \Rightarrow 1] &= \Pr[\mathsf{Game}_{9'} \Rightarrow 1]. \end{split}$$

In Game_{10} and $\mathsf{Game}_{10'}$, sk_i is used only in $\mathsf{UpdateLists}$ to compute resp' , and the tuple including resp' is added to L^{tmp} which is perfectly hidden to $\mathcal A$ until i is corrupted. From the definition of $\hat i$, by the time F_{find} is set true and pair is assigned, $\hat i$ has not been corrupted and $\mathcal A$ has no information about $\mathsf{sk}_{\hat i}$. Thus we have

$$\begin{split} \Pr[\mathsf{Game}_{10'} \Rightarrow 1] &= \Pr[F_{\mathrm{find}} = \mathsf{true} \wedge \mathsf{VerKey}(\mathsf{par}, \mathsf{pk}_{\hat{i}}, \mathsf{sk}^*) = 1 \wedge \mathsf{sk}_{\hat{i}} \neq \mathsf{sk}^*] \\ &= \Pr[F_{\mathrm{find}} = \mathsf{true} \wedge \mathsf{VerKey}(\mathsf{par}, \mathsf{pk}_{\hat{i}}, \mathsf{sk}^*) = 1] \\ &\times \Pr[\mathsf{sk}_{\hat{i}} \neq \mathsf{sk}^* \mid F_{\mathrm{find}} = \mathsf{true} \wedge \mathsf{VerKey}(\mathsf{par}, \mathsf{pk}_{\hat{i}}, \mathsf{sk}^*) = 1] \\ &= \Pr[\mathsf{Game}_{10} \Rightarrow 1] \times \frac{K-1}{K}. \end{split}$$

This completes the proof of Lemma 1.

B Cryptographic Group Actions

Here, we recall cryptographic group actions.

Definition 14 (Group Action). A group \mathcal{G} is said to act on a set \mathcal{E} if there is a map $\star : \mathcal{G} \times \mathcal{E} \to \mathcal{E}$ that satisfies the following two properties:

- 1. Identity: If $1_{\mathcal{G}}$ is the identity element of \mathcal{G} , then for all $E \in \mathcal{E}$, we have $1_{\mathcal{G}} \star E = E$.
- 2. Compatibility: For any $g, h \in \mathcal{G}$ and any $E \in \mathcal{E}$, we have $(gh) \star E = g \star (h \star E)$.

We may denote a group action by using the abbreviated notation $(\mathcal{G}, \mathcal{E}, \star)$. For cryptographic purposes, we need the following propositions.

Definition 15. A group action $(\mathcal{G}, \mathcal{E}, \star)$ is said to be

- 1. transitive if, for every $E_1, E_2 \in \mathcal{E}$, there exists a unique $g \in \mathcal{G}$ such that $E_2 = g \star E_1$,
- 2. free if, for all $E \in \mathcal{E}$, $E = g \star E$ implies $g = 1_{\mathcal{G}}$.

If a group action is transitive and free, it is said to be regular.

Note that if a group action is regular, then for any $E \in \mathcal{E}$, the map $f_E : g \to g \star E$ defines a bijection between \mathcal{G} and \mathcal{E} ; especially if \mathcal{G} or \mathcal{E} is finite, then we must have $|\mathcal{G}| = |\mathcal{E}|$.

To construct feasible cryptographic primitives from a group action, we require some efficient PPT algorithms. We recall the *effective group action* framework introduced in [1].

Definition 16 (Effective Group Action [1]). A group action $(\mathcal{G}, \mathcal{E}, E_0, \star)$ is effective if the following properties are satisfied:

- 1. The group G is finite, and there exist PPT algorithms for (1) the membership testing, (2) equality testing, (3) group operations, (4) element inversions, and (5) random sampling over G. The sampling method is required to be statistically indistinguishable from the uniform distribution over G.
- 2. The set \mathcal{E} is finite, and there exist PPT algorithms for (1) the membership testing and (2) generating a unique bit-string representation for every element in \mathcal{E} .
- 3. There exists a distinguished element $E_0 \in \mathcal{E}$ and its bit-string representation is publicly known.
- 4. There exists a PPT algorithm that given any $(g, E) \in \mathcal{G} \times \mathcal{E}$ outputs $g \star E$.

An effective group action is denoted using the abbreviated notation $(\mathcal{G}, \mathcal{E}, E_0, \star)$. Let GAGen be a PPT algorithm that takes 1^{λ} as input and outputs a description of an effective group action $(\mathcal{G}, \mathcal{E}, E_0, \star)$. The next hardness assumption of group actions is often used. Definition 17 (Group Action Discrete Logarithm (GADL) Assumption [19, Definition 16]). We say that GADL assumption holds for GAGen if for all PPT adversaries A, it holds that

$$\mathsf{Adv}^{\mathsf{GADL}}_{\mathcal{A},\mathsf{GAGen}}(\lambda) \coloneqq \Pr \left[\alpha \star E_0 = E \; \middle| \; \begin{array}{c} (\mathcal{G},\mathcal{E},E_0,\star) \leftarrow \mathsf{GAGen}(1^\lambda), \\ E \hookleftarrow \!\!\!\!\! & \mathcal{E}, \\ \alpha \leftarrow \mathcal{A}(\mathcal{G},\mathcal{E},E_0,\star,E) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

C Note on Zero-Knowledge of Randomized Fischlin Transformation

In [18, Theorem 6.4] (the full version of [17]), Kondi and shelat showed that the randomized Fischlin transformation preserves the zero-knowledge property of the underlying Sigma protocol. Especially their proof only depends on the existence of a perfect ZK simulator and the property that the entropy of commitments is λ , which is large enough. Here, we will show a flaw in the proof.

In their proof of ZK, they construct a simulator and show that simulated proofs and real proofs are indistinguishable using a sequence of hybrid experiments. Starting from the real proof, the change to Hybrid \mathcal{H}_1 is merely syntactic. Each transcript in \mathcal{H}_1 is generated using the prover's algorithm, i.e., the prover searches a "good" challenge by asking many transcripts to the random oracle, and the random oracle honestly answers a random value to each query. In Hybrid \mathcal{H}_2 , a "good" challenge is no longer searched; instead, the first chosen challenge is made a good one by programming the random oracle. Thus, in this experiment, the prover asks only ρ queries to the random oracle. Concretely, the random oracle H is implemented as follows:

- 1. The first ρ queries by the prover Q_1, \ldots, Q_{ρ} will receives 0 as a response.
- 2. Emulate H as a random oracle honestly for every other query.

To show the difference between \mathcal{H}_1 and \mathcal{H}_2 is negligibly small, the authors first claimed that each "good" challenge e_i appeared in the proof is distributed uniformly in $\{0,1\}^t$ in both \mathcal{H}_1 and \mathcal{H}_2 . Next, they said "the only distinguishing event corresponds to the programming of H, i.e., if the adversary is able to query H on some index that \mathcal{H}_2 subsequently programs to a different value." "this distinguishing event happens with probability no greater than $|Q|/2^{\lambda}$, where |Q| is the number of queries made by the adversary to the random oracle."

It is true that each e_i is distributed uniformly in $\{0,1\}^t$ in both \mathcal{H}_1 and \mathcal{H}_2 . However, even if the above-mentioned distinguishing event never occurs, the random oracle simulation in \mathcal{H}_2 is not perfect, as shown below.

We give a tiny example in which t=1 (i.e., the challenge is 0 or 1), the hash length is 1 (i.e., hash values are 0 or 1), $\rho=1$ (i.e., the proof includes only one transcript). Let tr_0 and tr_1 be the potential transcripts corresponding to challenge 0 and 1, respectively, and $h_0:=H(tr_0), h_1:=H(tr_1)$ be their hash values chosen by the random oracle (or simulated values).

Let c be the challenge first chosen to search for a "good" one, e be the "good" challenge actually put in the proof in \mathcal{H}_1 . Then, the following 8 cases occur with equal probability in \mathcal{H}_1 .

```
 - (h_0, h_1, c, e) = (0, 0, 0, 0). 
 - (h_0, h_1, c, e) = (0, 0, 1, 1). 
 - (h_0, h_1, c, e) = (0, 1, 0, 0). 
 - (h_0, h_1, c, e) = (0, 1, 1, 0). 
 - (h_0, h_1, c, e) = (1, 0, 0, 1). 
 - (h_0, h_1, c, e) = (1, 0, 1, 1). 
 - (h_0, h_1, c, e) = (1, 1, 0, \bot). 
 - (h_0, h_1, c, e) = (1, 1, 1, \bot).
```

Ignoring the last two cases, indeed, e is 0 or 1 with the same probability.

On the other hand, in \mathcal{H}_2 , e is first chosen randomly, and fix $h_e = 0$, while the hash value h_{1-e} is chosen randomly. Then, the following 4 cases occur with probability 1/4.

```
-(h_0, h_1, c, e) = (0, 0, -, 0).
-(h_0, h_1, c, e) = (0, 1, -, 0).
-(h_0, h_1, c, e) = (0, 0, -, 1).
-(h_0, h_1, c, e) = (1, 0, -, 1).
```

Now assume e = 0. Then, $\Pr_{\mathcal{H}_1}[h_1 = 0 \mid e = 0] = 1/3$, $\Pr_{\mathcal{H}_1}[h_1 = 1 \mid e = 0] = 2/3$ hold in \mathcal{H}_1 , while $\Pr_{\mathcal{H}_2}[h_1 = 0 \mid e = 0] = \Pr_{\mathcal{H}_2}[h_1 = 1 \mid e = 0] = 1/2$ hold in \mathcal{H}_2 . This means that there exists a possibility that the adversary distinguishes \mathcal{H}_1 and \mathcal{H}_2 even if the adversary does not ask for h_0 in advance.

We note that using a biased coin rather than a fair coin to decide the value $h_{1-e} = H(tr_{1-e})$ cannot solve the above problem. This is because tr_{1-e} may depend on the witness and random coins the prover used. E.g., the transcript corresponding to challenge 1-e is tr if w is used as a witness, but the transcript corresponding to challenge 1-e is $tr'(\neq tr)$ if $w'(\neq w)$ is used. In such cases, it is impossible to emulate a random oracle in a way that correctly matches both witnesses.

We can solve this problem by using the property of the underlying sigma protocol, strong 2-special soundness. The property ensures that any polynomial-time algorithms $\mathcal A$ cannot find two valid transcripts, and thus the adversary does not query tr_{1-e} (except negligible probability $\mathsf{Adv}_{\mathcal A,\Sigma}^{\mathsf{SSS}}(\lambda)$) even if the adversary knows one valid transcript tr_e . So, we can estimate the difference between $\mathcal H_1$ and $\mathcal H_2$ as $|Q|/2^{\lambda} + \mathsf{Adv}_{\mathcal A,\Sigma}^{\mathsf{SSS}}(\lambda)$.