Signatures with Tight Adaptive Corruptions from Search Assumptions

Keitaro Hashimoto[1], Wakaha Ogata[2], and Yusuke Sakai[1]

[1] 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 static search assumptions, such as classical discrete logarithm, RSA, factoring, or post-quantum group action discrete logarithm assumptions. In contrast to our scheme, the previous tightly secure schemes are based on decisional assumptions (e.g., (group action) DDH) or interactive search assumptions (e.g., one-more CDH). The security of our schemes is independent of the numbers of users, signing queries, and random oracle queries, and forging our signatures is as hard as solving the underlying static search problems. Our signature schemes are based on an identification scheme with multiple secret keys per public key and "second-key recovery resistance," difficulty of finding another secret key of a given public and secret key pair (e.g., Okamoto identification (CRYPTO'92) and Parallel-OR identification (CRYPTO'94)). These properties allow a reduction in solving a search problem while answering signing and 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 transformation introduced by Kondi and shelat (Asiacrypt 2022) that provides improved straight-line extraction. Intuitively, the transformation guarantees 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. Also, 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. This paper summarizes what they overlooked in the proof of zero-knowledge property of the transformation, the difficulty of correcting their proof, and how to overcome it.

Keywords: 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) [18]. It guarantees that an efficient adversary, given a single verification key, cannot forge a valid signature for any new message under that key. 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 [3,27,28,19] and identity-based signature [25].

Tight security. The security of cryptographic primitives is proved by constructing a reduction algorithm that transforms an efficient adversary who breaks the security of the scheme (e.g., the UF-CMA security of the signature scheme) into an algorithm that solves an assumed-to-be-hard computational problem (e.g.,

discrete logarithm problem). In general, the success probability of the reduction and that of the adversary have some gap 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, the reduction is said to be *tight*, and the scheme is said to be tightly secure. This means the primitive's security is independent of the adversary's behavior (e.g., the number of hash computations and the number of signing queries) and the number of users in the system. Tight security is important in both theory and practice. A tight reduction reveals the connection between the hardness of breaking the security and that of solving computational problems. Since computational problem hardness is generally well-studied, we can easily understand the security level of tightly secure schemes. In addition, tightly secure schemes allow for optimal parameter selection that meets the desired security level based on cryptoanalysis against underlying computational problems. As a result, data size (e.g., key and signature sizes) and computation cost (e.g., signing and verification) can be minimized, obtaining more efficient than non-tight schemes with large parameter sets considering reduction loss.

Tight MU-UF-CMA-C secure signature. It is known that EU-CMA security implies MU-UF-CMA-C security, albeit with a reduction loss proportional to the number of users. In other words, such signature schemes require a larger parameter to be selected to satisfy sufficient security level to account for reduction loss. To mitigate this loss, researchers have focused on developing *tightly* MU-UF-CMA-C-secure signature schemes, where the security remains independent of the number of users and other factors such as the numbers of signing and random oracle queries [2,3,17,34,10], summarized in Table 1. We notice that they 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). At the same time, several impossibility results regarding tightly MU-UF-CMA-C-secure signatures have been established [4,30,35]. These results identify specific conditions under which tightly MU-UF-CMA-C-secure signatures cannot exist. However, they do not completely rule out the possibility of constructing such schemes under static search assumptions.

This literature leads to the following research question[3]:

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

1.2 Our Contributions

We provide an affirmative answer to the open question above. Specifically, we construct the *first* signature schemes achieving tight (strong) MU-UF-CMA-C security under static search assumptions such as (group action) discrete logarithm, RSA, or factoring assumptions in the (classical) random oracle model (ROM). 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 signature is a generic construction based on a specific identification scheme that allows multiple secret keys per public key, such as Okamoto identification scheme [26] and the Parallel-OR identification scheme [8]. To tightly convert such an identification scheme into a signature scheme, we utilize randomized Fischlin transformation introduced by Kondi and shelat [22], which improves Fischlin's transformation [15]. While the transformation guarantees tight security in the programmable ROM, we advance this result by proving the tight security of our signature scheme in the non-programmable ROM.

By instantiating the underlying identification scheme appropriately, we obtain tightly MU-UF-CMA-C secure signature schemes based on various static search assumptions such as 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, our group action-based scheme is superior to the scheme by Pan and Wagner [30] regarding efficiency, security level, and computational assumption. Further, our signature gives us the first tightly secure authenticated key exchange protocol, tightly secure identity-based signature [25] and tightly secure certificates signature [20] based on search assumptions.

[3] Some previous works [29,27,28] raised the same question as an open problem.

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 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]	$\|\mathbb{G}\|$	$6\|\mathbb{G}\|$	SXDH	Pairing, PRO	—
BHJKL [3]	$\mathcal{O}(\lambda)\|\mathbb{G}\|$	$\mathcal{O}(\lambda)\|\mathbb{G}\|$	DLIN	Pairing	—
GJ [17]	$2\|\mathbb{G}\|$	$2\|\mathbb{G}\| + 4\|\mathbb{Z}_q\| + 2\lambda$	CDH & DDH	PRO	—
WLGSZ [34]	$\|\mathbb{G}\|$	$2\|\mathbb{G}\| + 1$	OM-CDH	Pairing, PRO	—
DGJL [10, Sec. 5.1]	$4\|\mathbb{G}\|$	$3\|\mathbb{Z}_q\|$	DDH	NPRO	✓
Ours (based on [26])	$\|\mathbb{G}\|$	$\rho(2\|\mathbb{Z}_q\| + t)$	DL	NPRO	✓

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

Scheme	Public key	Signature	Assumptions	Settings	SUF?
DGJL [10, Sec. 5.2]	$2\|\mathbb{Z}_N\|$	$2\|\mathbb{Z}_N\| + \lambda/4$	ϕ-hiding	NPRO	✓
Ours (based on [26])	$\|\mathbb{Z}_N\|$	$\rho(2\|\mathbb{Z}_N\| + t)$	RSA	NPRO	✓
Ours (based on [16])	$\|\mathbb{Z}_N\|$	$\rho(2\|\mathbb{Z}_N\| + 2t + \mathsf{poly}(\lambda))$	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	Settings	SUF?
PW [30, Sec. 4.2]	$4\|\mathcal{E}\|$	$2\lambda(2\|\mathcal{E}\| + \|\mathcal{G}\|)$	GADDH	PRO	—
Ours (based on [7,5,33])	$2\|\mathcal{E}\|$	$2\rho t(\|\mathcal{G}\| + 1)$	GADL	NPRO	✓

1.3 Technical Overview

This section provides a technical overview of our main result and a side contribution.

Intuition of our signature scheme. Our construction starts with a specialized identification scheme with multiple secret keys per public key and a property called "second-key recovery resistance." This property means that for a given key pair $(\mathsf{pk}, \mathsf{sk})$, computing another valid secret key sk' with respect to pk can be reduced to breaking a search assumption. To tightly convert such an identification scheme into a signature scheme, we utilize the randomized Fischlin transformation [22]. In this transformation, the prover, given a statement x and a witness w, first generates a commitment of the underlying Sigma protocol com. Then, the prover searches a challenge ch in a random order such that the hash value $h = \mathsf{H}(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ is the all-zero string where resp is the response under com and ch. The proof is the transcript $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$, i.e., a valid transcript of the Sigma protocol that yields the all-zero hash value.[4] This mechanism realizes straight-line extraction in the ROM, since the prover is imposed to query the RO for multiple valid transcripts that share the same commitment until getting the all-zero answer. That is, the extractor can obtain sufficient valid transcripts to extract a witness by observing the RO queries.

Now, let us explain how the reduction $\mathcal{B}$ from the MU-UF-CMA-C security of the obtained scheme to a search assumption works. $\mathcal{B}$ first generates key pairs $(\mathsf{sk}_i, \mathsf{pk}_i)$ for all users by itself, and runs MU-UF-CMA-C adversary $\mathcal{A}$. Indeed, $\mathcal{B}$ can simulate the singing oracle and the corruption oracle perfectly using the secret keys sk_i. If $\mathcal{A}$ outputs a valid signature under a public key pk_i, $\mathcal{B}$ can extract a secret key sk'_i corresponding to pk_i due to the straight-line extractability of randomized Fischlin transformation. If the

[4] To balance the soundness error and the efficiency, in the actual protocol, the prover does this process in parallel, but this is not important for the moment.

extracted key sk'_i is different from sk_i generated by $\mathcal{B}$, $\mathcal{B}$ succeeds in breaking the search assumption from the second-key recovery resistance of the underlying identification scheme. Now, we hope $\mathsf{sk}'_i \neq \mathsf{sk}_i$ holds with high probability, but our concern is that $\mathcal{A}$ might be able to get some information about sk_i from the answers of the signing oracle (i.e., signatures) simulated with sk_i and make $\mathsf{sk}'_i = \mathsf{sk}_i$. (Of course, $\mathcal{A}$ is disallowed to get sk_i from the corruption oracle.) Roughly, signatures of our scheme are non-interactive zero-knowledge (NIZK) proofs obtained by randomized Fischlin transformation, but, unfortunately, we cannot conclude easily that signatures give no information about the used key sk_i, since the transformation does not necessarily provide *perfect* zero-knowledge property. (See below for details.) Instead, we prove that $\mathcal{A}$ actually has no information about sk_i unless $\mathcal{A}$ issues special queries to the random oracle, and that it is infeasible to make such special queries for PPT adversaries.

Interestingly, the tight security of our signature scheme is shown in the *non-programmable* ROM, although the transformation guarantees (possibly computational) zero-knowledge property in the *programmable* ROM. This is because the programmability of RO is only used to show $\mathcal{A}$ has no information about sk_i in our security proof, and the reduction $\mathcal{B}$ does not need to program RO.

Randomized Fischlin transformation, reconsidered. 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 [23, Proof of Theorem 6.4]. Here, we summarize what Kondi and shelat overlooked in the proof of ZK property of their transformation, the difficulty of correcting their proof, and how to overcome it.

We first briefly review two worlds, the real world and a simulated world, used to define the ZK property for NIZK in the ROM [15,12]. In both worlds, an adversary $\mathcal{D}$ first outputs a statement x and a witness w, receives a challenge proof, and tries to guess which world it lives in accessing the RO. In the real world, the challenge proof is generated by an honest prover having (x, w). On the other hand, in the simulated world, the challenge proof is generated by a simulator Sim having only x, and the random oracle $\mathcal{D}$ accesses to is controlled by Sim. The important point is that Sim has to not only simulate a proof (typically while programming the RO) but also mimic the behavior of the real RO (considering the programmed parts in response to queries issued by the honest prover when generating the simulated proof as if it were honestly generated.

We observe that, although each answer of the RO in the real world is uniformly distributed, its distribution *conditioned on the fact that the challenge proof is the first transcript that the honest prover would find yielding the all-zero hash value* is not necessary the uniform distribution; while the hash value of the transcript in the challenge proof is guaranteed to be all-zero, the hash values of other valid transcripts that the honest prover would query are less likely to be all-zero, and those to the other queries are uniformly distributed. (We will give a tiny concrete example in Section 6.) However, as far as we understand, Kondi and shelat overlooked this point. In their proof, the simulator honestly simulates the random oracle, with the only exception that the simulator programs the all-zero hash value to simulate a challenge proof.

To prove the ZK property of randomized Fischlin transformation correctly, the simulator needs to recognize RO queries that the honest prover would make and respond to them differently than other RO queries. However, difficulties arise due to the two facts: RO queries issued by the honest prover depend on the honest prover's witness, and the simulator does not know the honest prover's witness. We elaborate on this difficulty using an example of a Parallel-OR protocol. In a Parallel-OR protocol for two statements, the transcript consists of two "sub-transcripts", each proving the knowledge of each witness, and the sum of the two sub-challenges is equal to the challenge of OR protocol. If an honest prover knows a witness of the first statement w.l.o.g., the second sub-transcript is first simulated and fixed, and the first sub-transcript is honestly generated based on the challenge. Thus, to generate a proof of randomized Fischlin transformation, the prover repeatedly queries the RO for transcripts with different first sub-transcripts but the common second sub-transcripts. That is, the sequence of queries issued by the honest prover is completely different depending on the witness that the prover has. In the simulated world in the proof of the ZK property, the simulator has to simulate the RO differently depending on whether the query would be issued by the honest prover or not, but it seems impossible for the simulator to distinguish them.

We overcome this difficulty by requiring the underlying Sigma protocol to have three properties in addition to perfect ZK and high commitment entropy: Strong 2-special soundness[5], witness collision resistance[6], and a property that we call "re-simulatability."[7] The key observation here is that once an adversary issues an RO query that looks like a query that the honest prover would make, the simulator can extract a witness from the challenge transcript and the queried transcript, due to the strong 2-special soundness. Furthermore, due to the witness collision resistance, this extracted witness should be equal to the witness specified by the adversary. Finally, using the re-simulatability, the simulator can generate the transcript that would be queried by the honest prover. Using these facts, the simulator can distinguish which RO queries the honest prover would make, and then the simulator can simulate the real experiment.

1.4 Future Work and Open Problem

We leave the following interesting future works.

Regarding tight MU-UF-CMA-C secure signature. First, our scheme's signature size is not constant; it is proportional to the security parameter. Constructing a tight MU-UF-CMA-C secure signature with a constant signature size from search assumptions is an important future work to improve efficiency. Second, proving or disproving the existence of tight MU-UF-CMA-C secure signatures based on static search assumptions in the standard model is an interesting theoretical question. Finally, the security of randomized Fischlin transform in the quantum ROM (QROM) has not been shown, so the tight security of our construction is not guaranteed in the QROM either. A promising alternative approach to achieve a tight MU-UF-CMA-C signature from post-quantum search assumption (e.g., GADL) in the QROM is the use of Pass's transformation [31], i.e., a Sigma protocol is first transformed into a commit-then-open type interactive protocol and then it is converted to a non-interactive protocol using Fiat-Shaimr (FS) transformation [13], since the recent result by Don et al. [11] allows us to convert commit-then-open type protocols into NIZK via FS transformation in the QROM tightly. A drawback of Pass's transformation is its inefficiency. The proof length is $\mathcal{O}(\lambda)$ times longer than the total length of a transcript in the underlying Sigma protocol. However, it is comparable to our group action-based scheme in which the signature consists of $\mathcal{O}(\lambda)$ group elements.

Regarding randomized Fischlin transformation. To correct the proof for the ZK property of randomized Fischlin transformation, we require three additional properties to the underlying Sigma protocol: strong 2-special soundness, witness collision resistance, and re-simulatability. At this point, it is unclear whether all of them are necessary for the ZK property, and we leave the analysis for future work. Especially assuming witness collision resistance prevents us from converting Parallel-OR protocols into NIZKs because Parallel-OR protocols do not have witness collision resistance (cf. Remark 6). Thus, relaxing the underlying assumption is an important work to expand the applicability of randomized Fischlin transformation.

2 Preliminaries

This section reviews basic notations and definitions of cryptographic primitives used in this paper.

2.1 Notations

$\lambda \in \mathbb{N}$ denotes a security parameter. $\oplus$ denotes the bit-wise exclusive-or operation. $\mathsf{poly}(\cdot)$ and $\mathsf{negl}(\cdot)$ are any polynomial function and negligible function, respectively. e denotes the base of the natural logarithm

[5] Intuitively, strong 2-special soundness requires that a witness can be efficiently extracted from two different transcripts that share the same commitment.

[6] Witness collision resistance ensures that it is hard to find two different witnesses for the same statement. It was known as the hardness of representation problem [9].

[7] Intuitively, re-simulatability requires that given a witness w, a transcript $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ honesty generated from w and a (unknown) random coin r, and any challenge ch', one can efficiently compute a response resp' that the prover would compute using the same w and r to respond to ch' instead of ch.

(i.e., Napier's constant). For $n \in \mathbb{N}$, we define $[n] \coloneqq \{1, 2, \ldots, n\}$ 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 $\mathcal{A}$ is said to be PPT (probabilistic polynomial time) if its running time $T_{\mathcal{A}}$ can be bounded by a polynomial in its input size. The notation $y \leftarrow \mathcal{A}(x)$ means that the variable y is assigned to the output of the algorithm $\mathcal{A}$ on input x. We write $y \in \mathcal{A}(x)$ to state that y is a possible output of $\mathcal{A}$ on input x. Whenever we deal with statistically negligible terms, we denote them by Greek letters, e.g., ϵ_{XX}; for computationally negligible terms, we use notation like $\mathsf{Adv}^{\mathsf{game}}_{\mathcal{A},\Pi}(\lambda)$.

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.*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{par}$*: 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.*
- $\mathsf{KGen}(\mathsf{par}) \to (\mathsf{svk}, \mathsf{ssk})$*: On input a public parameter* par*, the key generation algorithm outputs a public key* svk *and a secret key* ssk*.*
- $\mathsf{Sign}(\mathsf{ssk}, m) \to \sigma$*: On input a secret key* ssk *and a message* m*, the signing algorithm outputs a signature* σ*.*
- $\mathsf{Vf}(\mathsf{svk}, m, \sigma) \to 1/0$*: 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* $\mathsf{par} \in \mathsf{Setup}(1^\lambda)$*, any key pair* $(\mathsf{svk}, \mathsf{ssk}) \in \mathsf{KGen}(\mathsf{par})$*, and any message* m*, it holds that*

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

In this work, we focus on strong existential unforgeability against adaptive chosen-message attacks in the multi-user setting with adaptive corruptions [10, Definition 2].[8] We call it MU-SUF-CMA-C security in short.

Definition 3 (N-MU-SUF-CMA-C Security [10]). *Let* $N = \mathsf{poly}(\lambda)$ *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*

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

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 [21].

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.*
- $\mathsf{IGen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$*: The key generation algorithm takes a public parameter* par *as input and outputs public and secret keys* $(\mathsf{pk}, \mathsf{sk})$*. We assume the secret key* sk *is chosen uniformly and randomly from the secret key space.*

[8] As mentioned in [10], strong unforgeability is useful for constructing cryptographic protocols such as authenticated key exchange.

N-MU-SUF-CMA-C(λ)

1 : $L_{\mathsf{corr}}, L_{\mathsf{sig}} \leftarrow \emptyset$
2 : $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$
3 : **foreach** $i \in [N]$ **do**
4 : $\quad (\mathsf{svk}_i, \mathsf{ssk}_i) \leftarrow \mathsf{KGen}(\mathsf{par})$
5 : $\mathsf{O} := (\mathsf{OSign}, \mathsf{OCorr})$
6 : $\mathsf{SVK} := \{\mathsf{svk}_i\}_{i \in [N]}$
7 : $(i^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{O}}(\mathsf{par}, \mathsf{SVK})$
8 : **if** $i^* \in L_{\mathsf{corr}}$ **then return** 0
9 : **if** $(i^*, m^*, \sigma^*) \in L_{\mathsf{sig}}$ **then**
10 : $\quad$ **return** 0
11 : $ok := \mathsf{Vf}(\mathsf{svk}_{i^*}, m^*, \sigma^*)$
12 : **return** ok

$\mathsf{OSign}(i, m)$

1 : **if** $i \in L_{\mathsf{corr}}$ **then**
2 : $\quad$ **return** $\perp$
3 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{ssk}_i, m)$
4 : $L_{\mathsf{sig}} := L_{\mathsf{sig}} \cup \{(i, m, \sigma)\}$
5 : **return** σ

$\mathsf{OCorr}(i)$

1 : $L_{\mathsf{corr}} := L_{\mathsf{corr}} \cup \{i\}$
2 : **return** ssk_i

Fig. 1: Security game for signature scheme.

- $\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$*: The prover algorithm is split into two algorithms.* P_1 *takes as input a key pair* $(\mathsf{pk}, \mathsf{sk})$ *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.
- $\mathsf{V}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp}) \to 1/0$*: The verifier algorithm takes a public key* pk *and a conversation transcript* $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ *as input and outputs* 1 *or* 0.

Let $K := |\{\mathsf{sk} : (\mathsf{pk}, \mathsf{sk}) \in \mathsf{IGen}(\mathsf{par})\}|$ denote the number of valid secret keys w.r.t. a public key 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* $\mathsf{par} \in \mathsf{ISetup}(1^\lambda)$*, all* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{IGen}(\mathsf{par})$*, all* $(\mathsf{com}, \mathsf{st}) \in \mathsf{P}_1(\mathsf{pk}, \mathsf{sk})$*, all* $\mathsf{ch} \in \mathsf{ChSet}$ *and all* $\mathsf{resp} \in \mathsf{P}_2(\mathsf{ch}, \mathsf{st})$*, we have* $\mathsf{V}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp}) = 1$.

We say a transcript $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ is *valid* w.r.t. pk if $\mathsf{V}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp}) = 1$.

Definition 6 (Key Verifiability [30]). *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* $\mathsf{par} \in \mathsf{ISetup}(1^\lambda)$ *and any* $(\mathsf{pk}, \mathsf{sk})$*,*

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

Definition 7 (Min-Entropy of Commitments [21]). *We say that* ID *has* κ-bits of commitment min-entropy*, if for all* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{IGen}(\mathsf{par})$*, 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* $(\mathsf{com}, *) \leftarrow \mathsf{P}_1(\mathsf{pk}, \mathsf{sk})$ *was honestly generated by the prover.*

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

$\mathsf{Tran}(\mathsf{pk}, \mathsf{sk}, \mathsf{ch})$

1 : $(\mathsf{com}, \mathsf{st}) \leftarrow \mathsf{P}_1(\mathsf{pk}, \mathsf{sk})$
2 : $\mathsf{resp} \leftarrow \mathsf{P}_2(\mathsf{ch}, \mathsf{st})$
3 : **return** $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$

Definition 8 (Special Honest-Verifier Zero-Knowledge (HVZK) [30]). *We say that* ID *is perfect special honest-verifier zero-knowledge (perfect HVZK) if there exists a* PPT *algorithm* Sim*, a simulator, such that for all* $\mathsf{par} \in \mathsf{ISetup}(1^\lambda)$ *and all* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{IGen}(\mathsf{par})$*, the following distributions are identical:*

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

and

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

Kondi and shelat [22] introduced strong special soundness, which is a stronger version of special soundness in the sense that the extractor can extract a secret key of pk from two valid transcripts $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ and $(\mathsf{com}, \mathsf{ch}', \mathsf{resp}')$ such that $(\mathsf{ch}, \mathsf{resp}) \neq (\mathsf{ch}', \mathsf{resp}')$. That is, the extractor works even in the case $\mathsf{ch} = \mathsf{ch}'$ and $\mathsf{resp} \neq \mathsf{resp}'$. In this work, we define computational strong special soundness as follows.

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*

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

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 [15]. Their original definition 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 unconditional strong special soundness. Since they said "Okamoto's identification protocol satisfies strong special soundness," they are probably considering the computational one as our definition.

We 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 $(\mathsf{pk}, \mathsf{sk})$, it is difficult to find another secret key $\mathsf{sk}^* \neq \mathsf{sk}$ 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* $N = \mathsf{poly}(\lambda)$ *be some natural number. We say that* ID *is* second-key recovery resistant in the multi-user setting *if for all adversaries* $\mathcal{A}$*, it holds that*

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{2^{nd}KR}}_{\mathcal{A},\mathsf{ID}}(\lambda) &:= \Pr\left[\begin{array}{c|c} (\mathsf{pk}_{i^*}, \mathsf{sk}^*) \in \mathsf{IGen}(\mathsf{par}) & \mathsf{par} \leftarrow \mathsf{ISetup}(1^\lambda), \\ \wedge \mathsf{sk}^* \neq \mathsf{sk}_{i^*} & (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{IGen}(\mathsf{par})\ \forall i \in [N], \\ & (i^*, \mathsf{sk}^*) \leftarrow \mathcal{A}(\mathsf{par}, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i \in [N]}) \end{array}\right] \\
&\leq \mathsf{negl}(\lambda).
\end{aligned}$$

3 Signature Scheme via Randomized Fischlin Transformation

In this section, we describe the signature scheme from an identification scheme via randomized Fischlin transformation [22]. Then, we prove that the signature scheme tightly archives MU-SUF-CMA-C security.

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

$\mathsf{par}' \leftarrow \mathsf{ISetup}(1^\lambda)$
return $\mathsf{par} := \mathsf{par}'$

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

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{IGen}(\mathsf{par})$
return $(\mathsf{svk}, \mathsf{ssk}) := (\mathsf{pk}, (\mathsf{pk}, \mathsf{sk}))$

$\mathsf{Vf}^{\mathsf{H}}(\mathsf{svk}, m, \sigma)$

$(\mathsf{com}_j, \mathsf{ch}_j, \mathsf{resp}_j)_{j\in[\rho]} := \sigma$
$\vec{\mathsf{com}} := (\mathsf{com}_1, \ldots, \mathsf{com}_\rho)$
$\mathsf{pfx} := (\mathsf{pk}, m, \vec{\mathsf{com}})$
foreach $j \in [\rho]$ **do**
  if $\mathsf{H}(\mathsf{pfx}, j, \mathsf{ch}_j, \mathsf{resp}_j) \neq 0^\gamma$ **then**
    return 0
  if $\mathsf{V}(\mathsf{pk}, \mathsf{com}_j, \mathsf{ch}_j, \mathsf{resp}_j) = 0$ **then**
    return 0
endfor
return 1

$\mathsf{Sign}^{\mathsf{H}}(\mathsf{ssk}, m)$

foreach $\tau \in [T]$ **do**
  foreach $j \in [\rho]$ **do**
    $(\mathsf{com}_{\tau,j}, \mathsf{st}_{\tau,j}) \leftarrow \mathsf{P}_1(\mathsf{pk}, \mathsf{sk})$
  endfor
  $\vec{\mathsf{com}}_\tau := (\mathsf{com}_{\tau,1}, \ldots, \mathsf{com}_{\tau,\rho})$
  $\mathsf{pfx}_\tau := (\mathsf{pk}, m, \vec{\mathsf{com}}_\tau)$
  foreach $j \in [\rho]$ **do**
    $S_j := \emptyset$
    while $S_j \neq \{0,1\}^t$ **do**
      $\mathsf{ch} \leftarrow\!\!\$\ \{0,1\}^t \setminus S_j$
      $\mathsf{resp}_{\tau,j,\mathsf{ch}} \leftarrow \mathsf{P}_2(\mathsf{ch}, \mathsf{st}_{\tau,j})$
      $h_{\tau,j,\mathsf{ch}} := \mathsf{H}(\mathsf{pfx}_\tau, j, \mathsf{ch}, \mathsf{resp}_{\tau,j,\mathsf{ch}})$
      if $h_{\tau,j,\mathsf{ch}} = 0^\gamma$ **then**
        $\mathsf{ch}_{\tau,j} := \mathsf{ch}$
        break // proceed to next j
      else
        $S_j := S_j \cup \{\mathsf{ch}\}$
    endwhile
  endfor
  if $\forall j\ \exists \mathsf{ch}:\ h_{\tau,j,\mathsf{ch}} = 0^\gamma$ **then**
    break // succeed in signing
  if $\tau = T$ **then**
    return $\perp$
endfor
$\hat{\tau} := \tau$
$\sigma := (\mathsf{com}_{\hat{\tau},j}, \mathsf{ch}_{\hat{\tau},j}, \mathsf{resp}_{\hat{\tau},j,\mathsf{ch}_{\hat{\tau},j}})_{j\in[\rho]}$
return σ

Fig. 2: The signature scheme $\mathsf{SIG}[\mathsf{ID}, \mathsf{H}]$.

3.1 Proposed Signature Scheme

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 hash function modeled as a random oracle. The signature scheme $\mathsf{SIG}[\mathsf{ID}, \mathsf{H}] := (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}^{\mathsf{H}}, \mathsf{Vf}^{\mathsf{H}})$ (in the ROM) is depicted in Figure 2.

We prove that $\mathsf{SIG}[\mathsf{ID}, \mathsf{H}]$ is correct and MU-SUF-CMA-C secure.

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

Proof. Randomization of Fischlin transformation 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 Sign outputs $\perp$ is at most $(\rho \cdot e^{-2^{t-\gamma}})^T = 2^{(-2^{t-\gamma} \log e + \log \rho)T}$. Since ID is perfectly correct, $\mathsf{SIG}[\mathsf{ID}, \mathsf{H}]$ 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* $\mathsf{SIG}[\mathsf{ID}, \mathsf{H}]$ *is* $\mathsf{MU\text{-}SUF\text{-}CMA\text{-}C}$ *secure in the non-programmable random oracle model.*

In particular, if there is an adversary $\mathcal{A}$ *that breaks the* $\mathsf{MU\text{-}SUF\text{-}CMA\text{-}C}$ *security of* $\mathsf{SIG}[\mathsf{ID}, \mathsf{H}]$ *in time* $T_{\mathcal{A}}$ *with success probability* $\mathsf{Adv}^{\mathsf{MU\text{-}SUF\text{-}CMA\text{-}C}}_{\mathcal{A},\mathsf{SIG}[\mathsf{ID},\mathsf{H}]}(\lambda)$*, 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* $\mathsf{Adv}^{\mathsf{SSS}}_{\mathcal{B}_1,\mathsf{ID}}(\lambda)$ *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* $\mathsf{Adv}^{\mathsf{2^{nd}KR}}_{\mathcal{B}_2,\mathsf{ID}}(\lambda)$ *such that*

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

Here, Q_{RO} *and* Q_{sig} *are the maximum numbers of* RO *queries and signature queries issued by* $\mathcal{A}$*, respectively.*

This theorem shows that the $\mathsf{MU\text{-}SUF\text{-}CMA\text{-}C}$ security of our signature scheme is *tightly reduced* to the security against the second-key recovery resistance and the strong special soundness of ID in the non-programmable ROM. Here, we provide a proof sketch. The full proof is provided in Appendix A.

Proof sketch. For simplicity, assume that the underlying ID scheme has perfect strong special soundness ($\mathsf{Adv}^{\mathsf{MU\text{-}SUF\text{-}CMA\text{-}C}}_{\mathcal{A},\mathsf{SIG}[\mathsf{ID},\mathsf{H}]}(\lambda) = 0$), honestly generated $\vec{\mathsf{com}}$ is always fresh (i.e., $1/2^{\rho\kappa} \approx 0$), and the adversary cannot find ch_j that yields the hash value 0^γ for each $j \in [\rho]$ with one oracle query (i.e., $1/2^{\rho\gamma} \approx 0$).

Consider the following reduction $\mathcal{R}$ that reduces the $\mathsf{MU\text{-}SUF\text{-}CMA\text{-}C}$ security of $\mathsf{SIG}[\mathsf{ID}, \mathsf{H}]$ to the second-key recovery resistance. On input $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]}$, $\mathcal{R}$ runs $\mathcal{A}$ with input $(\mathsf{pk}_i)_{i\in[N]}$. $\mathcal{R}$ simulates the signing and the corruption oracles using sk_i. On the simulation of the RO, $\mathcal{R}$ always returns a random hash value (without programming), and maintains the query-answer list L as usual. In addition, for each query $(\mathsf{pk}_i, m, \vec{\mathsf{com}}, \mathsf{ch}, \mathsf{resp})$, $\mathcal{R}$ checks if $(\mathsf{com}_j, \mathsf{ch}, \mathsf{resp})$ is a valid transcript w.r.t. pk_i, i.e., $\mathsf{V}(\mathsf{pk}_i, \mathsf{com}_j, \mathsf{ch}, \mathsf{resp}) = 1$ holds, and if there is another valid query $(\mathsf{pk}_i, m, \vec{\mathsf{com}}, \mathsf{ch}', \mathsf{resp}')$ that shares the same $(\mathsf{pk}_i, m, \vec{\mathsf{com}})$ but $(\mathsf{ch}', \mathsf{resp}') \neq (\mathsf{ch}, \mathsf{resp})$ in L. If exists, $\mathcal{R}$ sets the flag F_{find} and computes $\hat{\mathsf{sk}} = \mathsf{Ext}(\mathsf{pk}_i, \mathsf{com}_j, \mathsf{ch}, \mathsf{resp}, \mathsf{ch}', \mathsf{resp}')$. The perfect strong special soundness ensures that $\hat{\mathsf{sk}}$ is a valid secret key w.r.t. pk_i, and $\mathcal{R}$ successfully finds the second secret key if $\hat{\mathsf{sk}} \neq \mathsf{sk}_i$.

Next, we estimate the probability that $\hat{\mathsf{sk}} = \mathsf{sk}_i$ holds. If randomized Fischlin transformation preserves the perfect (HV)ZK property of ID, we can conclude that $\hat{\mathsf{sk}} = \mathsf{sk}_i$ holds with probability $1/K$, that implies $\mathcal{R}$'s success probability is $(K-1)/K$ times that of $\mathcal{A}$. However, as we will see in Section 6, the transformation does not preserve the perfect (HV)ZK property. More precisely, simulated signatures themselves never leak information about the used secret key sk_i, but the responses of the RO for special queries depend on sk_i. Let $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ be a valid transcript included in a signature generated by $\mathcal{R}$. During the singing process, $\mathcal{R}$ internally issues not only $(\mathsf{pk}_i, m, \vec{\mathsf{com}}, \mathsf{ch}, \mathsf{resp})$ but also other queries $(\mathsf{pk}_i, m, \vec{\mathsf{com}}, \mathsf{ch}'(\neq \mathsf{ch}), \mathsf{resp}')$ to the RO, and the conditional probability distribution of $H(\mathsf{pk}_i, m, \vec{\mathsf{com}}, \mathsf{ch}', \mathsf{resp}')$ under the condition "$(\mathsf{com}_i, \mathsf{ch}', \mathsf{resp}')$ was not included in the signature" is different from the uniform distribution. Furthermore, the set of such $(\mathsf{pk}_i, m, \vec{\mathsf{com}}, \mathsf{ch}', \mathsf{resp}')$ that has a biased probability distribution depends on the used secret key. Therefore, $\mathcal{A}$ may get some information about sk_i by asking to the RO for such $(\mathsf{pk}_i, m, \vec{\mathsf{com}}, \mathsf{ch}', \mathsf{resp}')$.

We avoid this situation using the observation that once $(\mathsf{pk}_i, m, \vec{\mathsf{com}}, \mathsf{ch}', \mathsf{resp}')$ that may leak the information about sk_i is queried to the RO, the flag F_{find} is set. In other words, $\mathcal{A}$'s view up to the point of issuing a query that triggers the flag to be set is independent of sk_i. Thus we can conclude that $\hat{\mathsf{sk}}$ extracted from the trigger query is equal to sk_i with probability $1/K$. □

Remark 2. In [22], the ZK property of the randomized Fischlin transformation is shown in the programmable ROM. Unfortunately, the proof has a flaw as shown in Section 6, and its ZK property is conditional. That is, randomized Fischlin transformation has conditional ZK property in the PROM. In contrast, we have successfully proven the security of the signature scheme in the non-programmable ROM (NPROM). This is

because, rather than using the computational ZK property of the transformation straightforwardly, we take advantage of the fact that its ZK property holds *unconditionally* until a query is requested to the RO that triggers the flag F_{find} to be set, and use the unconditional ZK property to show that no information about sk_i is leaked to $\mathcal{A}$ before requesting such a query.

3.2 Improving Efficiency

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

$$\mathsf{V}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp}) = 1 \quad \Longleftrightarrow \quad \mathsf{com} = f^{\mathsf{V}}(\mathsf{pk}, \mathsf{ch}, \mathsf{resp}) \tag{2}$$

for some efficiently computable function f^{V}, we can eliminate $\vec{\mathsf{com}}$ 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 $\mathsf{com}_j := f^{\mathsf{V}}(\mathsf{pk}, \mathsf{ch}_j, \mathsf{resp}_j)$, $\vec{\mathsf{com}} := (\mathsf{com}_j)_j$, and then checks if $\mathsf{H}(\mathsf{pk}, m, \vec{\mathsf{com}}, j, \mathsf{ch}_j, \mathsf{resp}_j) = 0^\gamma$ holds for all $j \in [\rho]$.

4 Instantiations of Signature Schemes

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 [26] and an instantiation from isogenies based on Couveignes-Stolbunov identification [7,5,33] with the Parallel-OR technique [8].

4.1 Instantiation from Classical Groups

Okamoto identification scheme. Okamoto protocol [26, Scheme 1], which is based on discrete-logarithm assumption, is one of the most important instantiations of our framework. Let GGen be a PPT algorithm 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 $(\mathbb{G}, q, g)$. 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 $\mathsf{ID}_{\mathsf{Oka}}$ 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 discrete logarithm assumption.

Definition 11 (Discrete Logarithm (DL) Assumption). *We say that* DL assumption holds for GGen *if for all* PPT *adversaries* $\mathcal{A}$*, it holds that*

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

We now prove $\mathsf{ID}_{\mathsf{Oka}}$'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 [26]). 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. □

Theorem 4. *Under the* DL *assumption,* $\mathsf{ID}_{\mathsf{Oka}}$ *is strong special sound. More precisely, there exists an extractor* Ext *such that, for any adversary* $\mathcal{A}$ *breaking strong special soundness of* $\mathsf{ID}_{\mathsf{Oka}}$ *with advantage* $\mathsf{Adv}^{\mathsf{SSS}}_{\mathcal{A},\mathsf{ID}_{\mathsf{Oka}}}(\lambda)$*, 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 $\mathsf{Adv}^{\mathsf{SSS}}_{\mathcal{A},\mathsf{ID}_{\mathsf{Oka}}}(\lambda)$ *is upper bounded by* $\max_{\mathcal{B}} \mathsf{Adv}^{\mathsf{DL}}_{\mathcal{B},\mathsf{GGen}}(\lambda)$.

$\mathsf{ISetup}_{\mathsf{Oka}}(1^\lambda)$

1 : $(\mathbb{G}, q, g) \leftarrow \mathsf{GGen}(1^\lambda)$
2 : $\alpha \leftarrow\!\!\$\ \mathbb{Z}_q; g_1 \leftarrow g^\alpha$
3 : $\mathsf{ChSet} \coloneqq \mathbb{Z}_q$
4 : **return** $\mathsf{par} \coloneqq (\mathbb{G}, q, g, g_1)$

$\mathsf{IGen}_{\mathsf{Oka}}(\mathsf{par})$

1 : $\mathsf{sk} \coloneqq (s_1, s_2) \leftarrow\!\!\$\ (\mathbb{Z}_q)^2$
2 : $\mathsf{pk} \coloneqq g^{s_1} g_1^{s_2}$
3 : **return** $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{V}_{\mathsf{Oka}}(\mathsf{par}, \mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$

1 : $(y_1, y_2) \coloneqq \mathsf{resp}$
2 : **if** $\mathsf{com} = g^{y_1} g_1^{y_2} / \mathsf{pk}^{\mathsf{ch}}$ **then**
3 : **return** 1
4 : **else return** 0

$\mathsf{P}_{\mathsf{Oka},1}(\mathsf{pk}, \mathsf{sk})$

1 : $(r_1, r_2) \leftarrow\!\!\$\ (\mathbb{Z}_q)^2$
2 : $R \coloneqq g^{r_1} g_1^{r_2}$
3 : $\mathsf{com} \coloneqq R$
4 : $\mathsf{st} \coloneqq (\mathsf{sk}, r_1, r_2)$
5 : **return** $(\mathsf{com}, \mathsf{st})$

$\mathsf{P}_{\mathsf{Oka},2}(\mathsf{st}, \mathsf{ch})$

1 : $y_1 \coloneqq r_1 + \mathsf{ch} \cdot s_1 \bmod q$
2 : $y_2 \coloneqq r_2 + \mathsf{ch} \cdot s_2 \bmod q$
3 : $\mathsf{resp} \coloneqq (y_1, y_2)$
4 : **return** resp

Fig. 3: The Okamoto protocol $\mathsf{ID}_{\mathsf{Oka}}$.

Proof. We define the extractor Ext as follows. Let $(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp}, \mathsf{ch}', \mathsf{resp}')$ be Ext's input. If $\mathsf{ch} \neq \mathsf{ch}'$, 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 $\mathsf{ch} \neq \mathsf{ch}'$, it is easy to see that Ext outputs a valid secret key. So, $\mathsf{Adv}^{\mathsf{SSS}}_{\mathcal{A},\mathsf{ID}_{\mathsf{Oka}}}(\lambda)$ is the probability that $\mathcal{A}$ outputs $(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp}, \mathsf{ch}', \mathsf{resp}')$ such that $\mathsf{ch} = \mathsf{ch}'$, $\mathsf{resp} \neq \mathsf{resp}'$, and $\mathsf{V}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp}) = \mathsf{V}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}', \mathsf{resp}') = 1$. Note that $(\mathsf{ch} = \mathsf{ch}' \wedge \mathsf{resp} \neq \mathsf{resp}')$ 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 $\mathsf{par} \coloneqq (\mathbb{G}, q, g, g_1)$. If $\mathcal{A}$ outputs pk and two valid transcripts $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ and $(\mathsf{com}, \mathsf{ch}', \mathsf{resp}')$ 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 HVZK.*

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

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*

$\mathsf{Adv}^{\mathsf{2nd\text{-}KR}}_{\mathcal{A},\mathsf{ID}}(\lambda)$, *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* $\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{B},\mathsf{GGen}}(\lambda) = \mathsf{Adv}^{\mathsf{2nd\text{-}KR}}_{\mathcal{A},\mathsf{ID}}(\lambda)$.

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 $\mathsf{par} := (\mathbb{G}, q, g, g_1)$ 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 $\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{B},\mathsf{GGen}}(\lambda) = \mathsf{Adv}^{\mathsf{2nd\text{-}KR}}_{\mathcal{A},\mathsf{ID}}(\lambda)$. 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. □

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

Corollary 1. *Under the* DL *assumption,* $\mathsf{SIG}[\mathsf{ID}_{\mathsf{Oka}}, \mathsf{H}]$ *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* $\mathsf{SIG}[\mathsf{ID}_{\mathsf{Oka}}, \mathsf{H}]$ *in time* $T_{\mathcal{A}}$ *with probability* $\mathsf{Adv}^{\mathsf{MU\text{-}SUF\text{-}CMA\text{-}C}}_{\mathcal{A},\mathsf{SIG}[\mathsf{ID}_{\mathsf{Oka}},\mathsf{H}]}(\lambda)$*, there is an algorithm* $\mathcal{B}$ *solving the* DL *problem in time* $T_{\mathcal{B}} = \mathcal{O}(T_{\mathcal{A}})$ *with probability* $\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{B},\mathsf{GGen}}(\lambda)$ *such that*

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

Remark 3 (Variants of Okamoto identification). It is worth noting that there are versions of Okamoto identification based on RSA assumption [26, Scheme 2] and Factoring assumption [14]. Using these identification schemes, we can obtain tightly MU-SUF-CMA-C-secure signatures from RSA and Factoring assumptions, respectively.

4.2 Instantiation from Group Action

We will show a group action-based tightly MU-SUF-CMA-C-secure signature derived from our framework. We first show that the so-called Parallel-OR identification scheme [8] meets the requirements of our framework. Then, we provide a concrete instantiation of the Parallel-OR identification scheme based on group action.

Parallel-OR identification scheme. Let $\mathsf{ID} = (\mathsf{ISetup}, \mathsf{IGen}, \mathsf{P}, \mathsf{V})$ be a canonical identification scheme with ℓ bits challenges and let Sim be a special honest verifier zero-knowledge simulator of ID. Then the Parallel-OR identification scheme $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}] := (\mathsf{ISetup}_{\mathsf{OR}}, \mathsf{IGen}_{\mathsf{OR}}, \mathsf{P}_{\mathsf{OR}}, \mathsf{V}_{\mathsf{OR}})$ is defined as shown in Figure 4.

For completeness, we show that $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ has all properties required for the underlying identification scheme to apply our framework in Section 3, if ID has correctness, single secret key, perfect HVZK, strong special soundness, second-key recovery resistance in the multi-user setting, and the following basic properties.

Definition 12 (Key Recovery [21, Definition 2.3]). *We say that* ID *is* key recovery resistant *if for all* PPT *adversaries* $\mathcal{A}$*, it holds that*

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

Definition 13 (Random Self-Reducibility [21, 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* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{IGen}(\mathsf{par})$*:*

$\mathsf{IGen}_{\mathsf{OR}}(\mathsf{par})$

$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{IGen}(\mathsf{par})$
$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{IGen}(\mathsf{par})$
$\mathsf{pk} := (\mathsf{pk}_0, \mathsf{pk}_1); \mathsf{sk} := (b, \mathsf{sk}_b)$
return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{V}_{\mathsf{OR}}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$

$\mathsf{ch}_1 := \mathsf{ch} \oplus \mathsf{ch}_0$
$b_0 \leftarrow \mathsf{V}(\mathsf{pk}_0, \mathsf{com}_0, \mathsf{ch}_0, \mathsf{resp}_0)$
$b_1 \leftarrow \mathsf{V}(\mathsf{pk}_1, \mathsf{com}_1, \mathsf{ch}_1, \mathsf{resp}_1)$
if $b_0 = 1 \wedge b_1 = 1$ **then**
  return 1
else return 0

$\mathsf{P}_{\mathsf{OR},1}(\mathsf{pk}, \mathsf{sk})$

$\mathsf{ch}_{1-b} \leftarrow\!\!\$\ \{0,1\}^{\ell}$
$(\mathsf{com}_{1-b}, \mathsf{resp}_{1-b}) \leftarrow \mathsf{Sim}(\mathsf{pk}_{1-b}, \mathsf{ch}_{1-b})$
$(\mathsf{com}_b, \mathsf{st}_b) \leftarrow \mathsf{P}_1(\mathsf{pk}_b, \mathsf{sk}_b)$
$\mathsf{com} := (\mathsf{com}_0, \mathsf{com}_1)$
$\mathsf{st} := (\mathsf{st}_b, \mathsf{ch}_{1-b}, \mathsf{resp}_{1-b})$
return $(\mathsf{com}, \mathsf{st})$

$\mathsf{P}_{\mathsf{OR},2}(\mathsf{st}, \mathsf{ch})$

$\mathsf{ch}_b := \mathsf{ch} \oplus \mathsf{ch}_{1-b}$
$\mathsf{resp}_b \leftarrow \mathsf{P}_2(\mathsf{ch}_b, \mathsf{st}_b)$
$\mathsf{resp} := (\mathsf{ch}_0, \mathsf{resp}_0, \mathsf{resp}_1)$
return resp

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

- pk' *and* pk'' *have the same distribution, where* $(\mathsf{pk}', \mathsf{td}') \leftarrow \mathsf{ReRand}(\mathsf{par}, \mathsf{pk})$ *is the rerandomized public key and* $(\mathsf{pk}'', \mathsf{sk}'') \leftarrow \mathsf{IGen}(\mathsf{par})$ *is a freshly-generated key pair.*
- *For all* $(\mathsf{pk}', \mathsf{td}') \in \mathsf{ReRand}(\mathsf{par}, \mathsf{pk})$*, for all* $(\mathsf{pk}', \mathsf{sk}') \in \mathsf{IGen}(\mathsf{par})$*, and* $\mathsf{sk}^* \leftarrow \mathsf{DeRand}(\mathsf{pk}, \mathsf{pk}', \mathsf{sk}', \mathsf{td}')$*, we have* $(\mathsf{pk}, \mathsf{sk}^*) \in \mathsf{IGen}(\mathsf{par})$*. That is,* DeRand *returns a valid secret key* sk^* *with respect to* pk*, given any valid secret key* sk' *for* pk'*.*

We now show $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$'s properties.

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

Proof. The correctness of $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ is clear. The commitment of $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ consists of two independent commitments of ID. Thus, $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ has 2κ-bits of commitment min-entropy. Finally, if the underlying identification scheme ID has a single secret key per public key, there are two secret keys $(0, \mathsf{sk}_0)$ and $(1, \mathsf{sk}_1)$ for each pk of $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$. Thus, $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ has 2-multiple secret keys. □

Theorem 8. *If* ID *is strong special sound,* $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ *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

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

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

with respect to $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$,

- 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 $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$. 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 $(\mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{com}_0, \mathsf{com}_1), \mathsf{ch}, (\mathsf{ch}_0, \mathsf{resp}_0, \mathsf{resp}_1), \mathsf{ch}', (\mathsf{ch}'_0, \mathsf{resp}'_0, \mathsf{resp}'_1)$. 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 $\mathsf{ch}_1 := \mathsf{ch} \oplus \mathsf{ch}_0, \mathsf{ch}'_1 := \mathsf{ch}' \oplus \mathsf{ch}'_0$.

From the assumption, for this $\mathcal{A}_{\mathcal{B}}$, $\mathsf{Ext}_{\mathsf{ID}}$ successfully extracts a valid secret key with probability at least $1 - \mathsf{Adv}^{\mathsf{SSS}}_{\mathsf{ID}}(\lambda)$. 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 $\mathsf{Ext}_{\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}$. Therefore, $\mathsf{Ext}_{\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}$ obtains a valid sk_0 or sk_1 with probability at least $1 - \mathsf{Adv}^{\mathsf{SSS}}_{\mathsf{ID}}(\lambda)$. That is, $\mathcal{B}$'s advantage is at most $\mathsf{Adv}^{\mathsf{SSS}}_{\mathsf{ID}}(\lambda)$. □

Theorem 9. *If* ID *is perfect HVZK,* $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ *is also perfect HVZK.*

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 $\mathsf{ch}_1 := \mathsf{ch} \oplus \mathsf{ch}_0$. Run $(\mathsf{com}_b, \mathsf{resp}_b) \leftarrow \mathsf{Sim}(\mathsf{pk}_b, \mathsf{ch}_b)$ for each $b \in \{0,1\}$, and output $\mathsf{com} := (\mathsf{com}_0, \mathsf{com}_1), \mathsf{resp} := (\mathsf{ch}_0, \mathsf{resp}_0, \mathsf{resp}_1)$. It is easy to see $\mathsf{Sim}_{\mathsf{OR}}$'s simulation is perfect from the fact that ID is perfect HVZK. □

Theorem 10. *If* ID *is key recovery resistant, random self-reducible, and has a single secret key, then* $\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]$ *is the second-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* $\mathsf{Adv}^{\mathsf{2^{nd}KR}}_{\mathcal{A},\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}(\lambda)$*, 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* $\mathsf{Adv}^{\mathsf{KR}}_{\mathcal{B},\mathsf{ID}}(\lambda) = \mathsf{Adv}^{\mathsf{2^{nd}KR}}_{\mathcal{A},\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}(\lambda)$.

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 $\mathsf{pk}_i := (\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1})$ and $\mathsf{sk}_i := (b_i, \mathsf{sk}_{i,b_i})$.

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 the random self-reducibility ensures that the randomized public key embedded in each $\mathsf{pk}_{i,1-b_i}$ is distributed identically to a 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 $\mathsf{Adv}^{\mathsf{KR}}_{\mathcal{B},\mathsf{ID}}(\lambda) = \mathsf{Adv}^{\mathsf{2^{nd}KR}}_{\mathcal{A},\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}(\lambda)$. 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 $f^{\mathsf{V}}_{\mathsf{OR}}$ as

$$f^{\mathsf{V}}_{\mathsf{OR}}(\mathsf{pk}_{\mathsf{OR}}, \mathsf{ch}_{\mathsf{OR}}, \mathsf{resp}_{\mathsf{OR}}) := \left(f^{\mathsf{V}}(\mathsf{pk}_0, \mathsf{ch}_0, \mathsf{resp}_0), f^{\mathsf{V}}(\mathsf{pk}_1, \mathsf{ch}_{\mathsf{OR}} \oplus \mathsf{ch}_0, \mathsf{resp}_1)\right),$$

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 $f^{\mathsf{V}}_{\mathsf{OR}}$. It means that $\vec{\mathsf{com}}$ can be eliminated in the signature of $\mathsf{SIG}[\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}], \mathsf{H}]$.

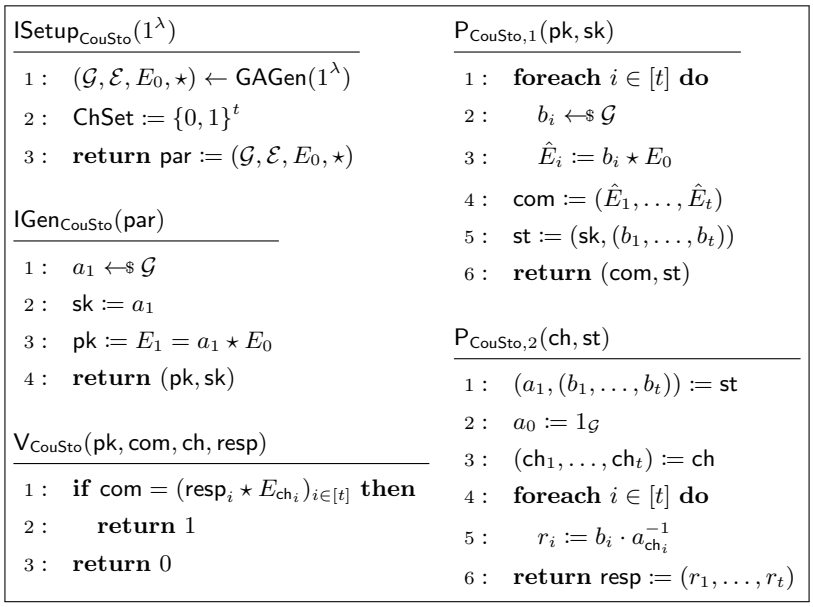

Fig. 5: The Couveignes-Stolbunov identification scheme $\mathsf{ID}_{\mathsf{CouSto}}$.

Couveignes-Stolbunov identification scheme. To instantiate the Parallel-OR identification scheme from group action, we will use Couveignes-Stolbunov identification scheme $\mathsf{ID}_{\mathsf{CouSto}}$ [7,5,33], 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 challenge, we simply repeat the protocol t times.

For completeness, we show that $\mathsf{ID}_{\mathsf{CouSto}}$ has correctness, high commitment entropy, single secret keys, perfect HVZK, strong special soundness, the key recovery resistance in the multi-user setting, and random self-reducible under the GADL assumption.

Theorem 11. $\mathsf{ID}_{\mathsf{CouSto}}$ *has correctness,* $t\log|\mathcal{G}|$*-bits of commitment min-entropy, and a single secret key.*

Proof. The correctness of $\mathsf{ID}_{\mathsf{CouSto}}$ is clear. The commitment of $\mathsf{ID}_{\mathsf{CouSto}}$ consists of t 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. More precisely,* $\mathsf{Adv}^{\mathsf{SSS}}_{\mathcal{A},\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}]}(\lambda) = 0$ *for all (including computationally unbounded) adversaries* $\mathcal{A}$*.*

Proof. We define the extractor Ext as follows. On input $(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp}, \mathsf{ch}', \mathsf{resp}')$, if $\mathsf{ch} = \mathsf{ch}'$, Ext outputs $\perp$. 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 $(\mathsf{resp}'_I \cdot (\mathsf{resp}_I)^{-1})^{\mathsf{ch}_I - \mathsf{ch}'_I}$.

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

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

Theorem 13. $\mathsf{ID}_{\mathsf{CouSto}}$ *is perfect HVZK.*

Proof. Consider the following simulator: On input $(\mathsf{pk} = E_1, \mathsf{ch})$, choose $\mathsf{resp}_i \leftarrow\!\!\$\ \mathcal{G}$, compute $\mathsf{com}_i := \mathsf{resp}_i \star E_{\mathsf{ch}_i}$ for all $i \in [t]$, and output $\mathsf{com} = (\mathsf{com}_1, \ldots, \mathsf{com}_t)$ and $\mathsf{resp} = (\mathsf{resp}_1, \ldots, \mathsf{resp}_t)$.

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

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 $\mathsf{par} := (\mathcal{G}, \mathcal{E}, E_0, \star)$ and $\mathsf{pk} := E_1$. $\mathcal{B}$ executes $\mathcal{A}$ on input $(\mathsf{par}, \mathsf{pk})$ and receives $\mathsf{sk} \in \mathcal{G}$ 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 $\mathsf{Adv}^{\mathsf{GADL}}_{\mathcal{B},\mathsf{GAGen}}(\lambda) = \mathsf{Adv}^{\mathsf{KR}}_{\mathcal{A},\mathsf{ID}_{\mathsf{CouSto}}}(\lambda)$. Note that the running time of $\mathcal{B}$ is $T_{\mathcal{B}} = T_{\mathcal{A}}$ since $\mathcal{B}$ executes $\mathcal{A}$ once. □

Theorem 15. $\mathsf{ID}_{\mathsf{CouSto}}$ *is random self-reducible.*

Proof. ReRand and DeRand are defined as follows:

$\mathsf{ReRand}(\mathsf{par}, \mathsf{pk})$**:** Let $E_1 := \mathsf{pk}$. Choose $c_1 \leftarrow\!\!\$\ \mathcal{G}$ and output $\mathsf{pk}' := c_1 \star E_1$ and $\mathsf{td}' := c_1$.
$\mathsf{DeRand}(\mathsf{pk}, \mathsf{pk}', \mathsf{sk}', \mathsf{td}')$**:** Let $a'_1 := \mathsf{sk}'$ and $c_1 := \mathsf{td}'$. Output $\mathsf{sk}^* := a'_1 \cdot (c_1)^{-1}$.

We have that, for all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{IGen}(\mathsf{par})$, pk' output from $\mathsf{ReRand}(\mathsf{par}, \mathsf{pk})$ 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 $\mathsf{sk}^* \leftarrow \mathsf{DeRand}(\mathsf{pk}, \mathsf{pk}', \mathsf{sk}', \mathsf{td}')$, if $\mathsf{pk}' = a'_1$ is a valid secret key of $\mathsf{pk}' = c_1 \star E_1$, we have $\mathsf{sk}^* \star E_0 = (a'_1 \cdot (c_1)^{-1}) \star E_0 = (c_1)^{-1} \star (a'_1 \star E_0) = (c_1)^{-1} \star (c_1 \star E_1) = E_1$. Therefore, sk^* is a valid secret key of $\mathsf{pk} = E_1$. □

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

Corollary 2. *Under the* GADL *assumption,* $\mathsf{SIG}[\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}_{\mathsf{CouSto}}], \mathsf{H}]$ *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* $\mathsf{SIG}[\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}_{\mathsf{CouSto}}], \mathsf{H}]$ *in time* $T_{\mathcal{A}}$ *with probability* $\mathsf{Adv}^{\mathsf{MU\text{-}SUF\text{-}CMA\text{-}C}}_{\mathcal{A},\mathsf{SIG}[\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}_{\mathsf{CouSto}}],\mathsf{H}]}(\lambda)$*, there is an algorithm* $\mathcal{B}$ *solving a* GADL *problem in time* $T_{\mathcal{B}} = \mathcal{O}(T_{\mathcal{A}})$ *with probability* $\mathsf{Adv}^{\mathsf{GADL}}_{\mathcal{B},\mathsf{GAGen}}(\lambda)$ *such that*

$$\mathsf{Adv}^{\mathsf{MU\text{-}SUF\text{-}CMA\text{-}C}}_{\mathcal{A},\mathsf{SIG}[\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}_{\mathsf{CouSto}}],\mathsf{H}]}(\lambda) \leq 2\mathsf{Adv}^{\mathsf{GADL}}_{\mathcal{B},\mathsf{GAGen}}(\lambda) + \frac{Q_{\mathsf{RO}}+1}{2^{\rho t \log|\mathcal{G}|}} + \frac{T \cdot Q_{\mathsf{sig}}(Q_{\mathsf{RO}} + T \cdot Q_{\mathsf{sig}})}{2^{\rho\gamma}}.$$

5 Efficiency Evaluation

In this section, we evaluate the efficiency of the classical group-based scheme $\mathsf{SIG}[\mathsf{ID}_{\mathsf{Oka}}, \mathsf{H}]$ and that of the isogeny-based instantiation derived from $\mathsf{SIG}[\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}_{\mathsf{CouSto}}], \mathsf{H}]$. Here, we consider 128-bit classical security (i.e., NIST Level-I).

Parameters for Fischlin transformation. For the choice of parameters used in randomized Fischlin transformation, we recall the recent result by Chen and Lindell [6] for the original version since their result is also valid for the randomized one. To achieve λ-bit security, the parameters must satisfy the following conditions: $\rho \cdot \gamma = \omega(\lambda)$ and $t - \gamma = \omega(\lambda)$. Chen and Lindell suggest that for any value of γ, for $\rho \leq 64$ set $t = \gamma + 5$ and for $\rho > 64$ set $t = \gamma + 6$. They also proved the correctness error probability and the expected number of hash computations summarized below:

Table 2: Efficiency of $\mathsf{SIG}[\mathsf{ID}_{\mathsf{Oka}}, \mathsf{H}]$ in 128-bit security. We set $T = 3$.

ρ	γ	t	β	#hash	$\lvert\mathsf{svk}\rvert$	$\lvert\sigma\rvert$
32	4	9	2^{-123}	512	32 B	2084 B
22	6	11	2^{-123}	1241	32 B	1444 B
16	8	13	2^{-126}	4096	32 B	1050 B

Table 3: Efficiency of $\mathsf{SIG}[\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}_{\mathsf{CouSto}}], \mathsf{H}]$ with CSIDH-512 parameter. We set $T = 3$.

ρ	γ	t	β	#hash	$\lvert\mathsf{svk}\rvert$	$\lvert\sigma\rvert$
32	4	9	2^{-123}	512	128 B	18.5 KB
22	6	11	2^{-123}	1241	128 B	15.5 KB
16	8	13	2^{-126}	4096	128 B	13.4 KB

– Probability of correctness error: $\beta = 2^{-(2^{t-\gamma}\log\hat{e}+\log\rho)T}$.
– Expected number of hash computations: #hash$= \rho \cdot 2^{\gamma}/(1-\beta)$.

Parameters for classical groups and efficiency of $\mathsf{SIG}[\mathsf{ID}_{\mathsf{Oka}}, \mathsf{H}]$. For the choice of a multiplicative group, we can use the NIST P-256 curve with $\log|\mathbb{G}| = 256$, which achieves 128-bit security since $\mathsf{SIG}[\mathsf{ID}_{\mathsf{Oka}}, \mathsf{H}]$ is tightly secure. Thus, $|\mathsf{resp}| = 512$ bits and the total signature size is $|\sigma| = \rho(t + |\mathsf{resp}|) = \rho(t + 512)$ bits for 128-bit security. Note that we can remove com from the signature because $\mathsf{ID}_{\mathsf{Oka}}$'s verifier algorithm can be represented as follows. (See Section 3.2.)

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

We give in Table 2 the efficiency estimations for $\mathsf{SIG}[\mathsf{ID}_{\mathsf{Oka}}, \mathsf{H}]$ in 128-bit security in some parameter sets. As observed, there is a trade-off between the signature sizes and the signing times. Spending more computational cost on signature generation can shorten the signature size. The signature size of $\mathsf{SIG}[\mathsf{ID}_{\mathsf{Oka}}, \mathsf{H}]$ is about 1.5 KB, but the DDH-based tight MU-SUF-CMA-C signature scheme [10, Sec. 5.1] has only 96 B signature. Thus, while $\mathsf{SIG}[\mathsf{ID}_{\mathsf{Oka}}, \mathsf{H}]$ relies on weaker DL assumption, the signature is about 15 times larger than the DDH-based scheme. This can be viewed as a trade-off between the strength of the assumptions on which security is based and the efficiency, but improving the efficiency of DL-based schemes is a future challenge.

Parameters for isogeny and efficiency of $\mathsf{SIG}[\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}_{\mathsf{CouSto}}], \mathsf{H}]$. We use the CSIDH-512 prime p and define the group action $g \star \mathcal{E}$ exactly as in CSI-FiSh [5]. According to [5, Sec. 2], the group elements require roughly 256 bits, and elements in $\mathcal{E}$ require about 512 bits. Thus, the public key size is $2|\mathcal{E}| = 1024$ bits and the signature size is $|\sigma| = 2\rho t(|\mathcal{G}| + 1) = 514\rho t$ bits in the CSIDH-512 parameter set. Note that, similar to $\mathsf{SIG}[\mathsf{ID}_{\mathsf{Oka}}, \mathsf{H}]$, we can remove com's from the signature because $\mathsf{SIG}[\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}_{\mathsf{CouSto}}], \mathsf{H}]$'s verifier algorithm can be represented as follows. (See Section 3.2.)

$$\mathsf{V}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp}) = 1 \iff \mathsf{com} = f^{\mathsf{V}}_{\mathsf{CouSto}}(\mathsf{pk}, \mathsf{ch}, \mathsf{resp}) = (\mathsf{resp}_i \star E_{\mathsf{ch}_i})_{i \in [t]}.$$

We give in Table 3 the efficiency estimations for $\mathsf{SIG}[\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}_{\mathsf{CouSto}}], \mathsf{H}]$ in some parameter sets. The concrete signature size of $\mathsf{SIG}[\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}_{\mathsf{CouSto}}], \mathsf{H}]$ instantiated with CSIDH-512 parameter is about 15 KB. On the other hand, the signature size of Pan and Wagner's group action-based scheme [30] with CSIDH-512 parameter is $2\lambda(2|\mathcal{E}| + |\mathcal{G}|) = 40.96$ KB. Thus, our signature is about 75 % shorter than Pan and Wagner's. Moreover, our signature is tightly MU-SUF-CMA-C secure based on GADL assumption in the NPROM while Pan and Wagner's one is tight MU-UF-CMA-C secure based on stronger GADDH assumption in the PROM. Therefore, $\mathsf{SIG}[\mathsf{ID}_{\mathsf{OR}}[\mathsf{ID}_{\mathsf{CouSto}}], \mathsf{H}]$ is superior to Pan and Wagner's scheme in terms of the efficiency, the achieved security level, and the underlying computational assumptions on which security is based.

6 Zero-Knowledge of Randomized Fischlin Transformation, Reconsidered

In [23, Theorem 6.4] (which is the full version of [22]), Kondi and shelat showed that 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 their proof and provide a new proof.

Flaw in the proof by Kondi and shelat. 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 for 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 for; 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 modified random oracle. Concretely, the modified random oracle H is implemented as follows:

1. For the first ρ queries by the prover $Q_1, \ldots, Q_\rho$, return 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 $\mathcal{H}_1$.

- $(h_0, h_1, c, e) = (0, 0, 0, 0)$.
- $(h_0, h_1, c, e) = (0, 0, 1, 1)$.
- $(h_0, h_1, c, e) = (0, 1, 0, 0)$.
- $(h_0, h_1, c, e) = (0, 1, 1, 0)$.
- $(h_0, h_1, c, e) = (1, 0, 0, 1)$.
- $(h_0, h_1, c, e) = (1, 0, 1, 1)$.
- $(h_0, h_1, c, e) = (1, 1, 0, \perp)$.
- $(h_0, h_1, c, e) = (1, 1, 1, \perp)$.

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 $e = 0$ appears in a proof. 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 fact leads to the conclusion that the adversary may take the following strategy to distinguish two hybrid experiments; ask $h_{1-e} = H(tr_{1-e})$ to the random oracle, and distinguish two experiments based on $h_{1-e} = 0$ or not. Note that the adversary knows the witness w used by the honest prover, thus can compute tr_{1-e} from w and tr_e in the proof.

We have to emphasize 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 for a simulator that correctly matches both witnesses because the simulator does not know which witness the prover uses.

New proof for ZK property. We solve this problem by requiring the underlying Sigma protocol to have the following properties.

- Strong 2-special soundness (cf. Definition 9), i.e., there exists an extractor that can extract a witness with overwhelming probability from two valid transcripts generated by a PPT adversary if the transcripts share the same commitment.
- A property we call "witness collision resistance", that guarantees it is hard to find a statement x and its two different witnesses w, w'. [9]
- The existence of an efficient algorithm ReSim that realizes re-simulatability; it requires that given a witness w, a transcript $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ honesty generated from w and a (unknown) random coin r, and any challenge ch', one can efficiently compute a response resp' that the prover would compute using the same w and r to respond to ch' instead of ch.[10]

To correct Kondi and shelat's proof, we only change the second hybrid experiment $\mathcal{H}_2$ to $\mathcal{H}'_2$. In $\mathcal{H}'_2$, the modified random oracle returns 0 for the first ρ queries by the honest prover as in the original $\mathcal{H}_2$, but for other queries by the adversary, uses fair coins and biased coins depending on the query described as follows.

First, we note the fact that the only difference between the random oracle simulation in $\mathcal{H}_2$ (by Kondi and shelat) and the real random oracle is the behavior of how to respond to queries *an honest prover would issue to search for a good challenge*. Considering that such queries are valid transcripts that share the same commitment with the transcript in the (simulated) proof, the query allows a witness to be extracted from strong 2-special soundness. Further, the extracted witness can be assumed to be the witness the honest prover has; otherwise, witness collision resistance would be broken. From the extracted witness, the efficient ReSim algorithm can reproduce transcripts *an honest prover would compute to search for a good challenge*, so we can confirm that the issued query is indeed a query that *an honest prover would issue*. If so, the modified random oracle uses a biased coin to respond; otherwise, it uses a fair coin.

Consequently, the difference between $\mathcal{H}_1$ and $\mathcal{H}'_2$ is bounded by $|Q|/2^\lambda$ plus negligible terms if the underlying Sigma protocol has strong 2-special soundness and the witness collision resistance.

Remark 5. The above proof for the ZK property of randomized Fischlin transformation requires three additional properties to the underlying Sigma protocol. At this point, it is unclear whether all of them are necessary for the ZK property, and we leave the analysis for future work.

Remark 6 (Witness collision resistance of existing ID schemes). If ID has a single secret key as in Schnorr identification scheme, it satisfies perfect witness collision resistance. Okamoto identification scheme satisfies witness collision resistance under DL assumption [9], since, if both $\mathsf{sk} = (s_1, s_2)$ and $\mathsf{sk}' = (s'_1, s'_2)$ are valid secret keys of pk, the discrete logarithm of h based on g can be obtained as $-(s_1 - s'_1)/(s_2 - s'_2)$.

On the other hand, the Parallel-OR identification scheme does not have witness collision resistance, because it is easy to generate a public key and two valid secret keys, $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1), \mathsf{sk} = (0, \mathsf{sk}_0), \mathsf{sk}' = (1, \mathsf{sk}_1)$. In contrast, as shown above, the Parallel-OR identification scheme has second-key recovery resistance (cf. Theorem 10), which can be seen as a weaker notion of witness collision resistance. This means that Parallel-OR protocols can be tightly converted into signature schemes but it is unclear whether it can be converted into NIZKs via randomized Fischlin transformation.

[9] Witness collision resistance was previously formalized as the hardness of representation problem [9].

[10] Re-simulatability is also used to prove Theorem 2 (the detail is in Appendix A). For the proof of Theorem 2, i.e., proof for our signature scheme, (possibly inefficient) algorithm is sufficient.

Remark 7 (Re-simulatability of existing ID schemes). Here, we note that Schnorr, Okamoto, and Paralell-OR identification schemes have an efficient algorithm ReSim realizing re-simulatability.

Schnorr ID: Upon receiving $w = x \in \mathbb{Z}_q$, $\mathsf{com} = g^r$, ch, $\mathsf{resp} = r + \mathsf{ch} \cdot x$ and another $\mathsf{ch}' \in \mathbb{Z}_q$, the algorithm outputs $\mathsf{resp}' := \mathsf{resp} + (\mathsf{ch}' - \mathsf{ch}) \cdot x \bmod q$.

Okamoto ID: Upon receiving $w = (s_1, s_2) \in (\mathbb{Z}_q)^2$, $\mathsf{com} = (g^{r_1} h^{r_2})$, ch, $\mathsf{resp} = (r_1 + \mathsf{ch} \cdot s_1, r_2 + \mathsf{ch} \cdot s_2)$ and another $\mathsf{ch}' \in \mathbb{Z}_q$, the algorithm outputs $\mathsf{resp}' := \mathsf{resp} + ((\mathsf{ch}' - \mathsf{ch}) \cdot s_1, (\mathsf{ch}' - \mathsf{ch}) \cdot s_2)$.

Parallel-OR ID: Assume the underlying ID scheme ID has $\mathsf{ReSim}_{\mathsf{ID}}$. The $\mathsf{ReSim}_{\mathsf{ID}_{\mathsf{OR}}}$ works as follows: It receives $w = (b, w_b)$, $\mathsf{com} = (\mathsf{com}_0, \mathsf{com}_1)$, ch, $\mathsf{resp} = (\mathsf{ch}_0, \mathsf{resp}_0, \mathsf{resp}_1)$ and another ch' as input. If $b = 0$, it computes $\mathsf{ch}'_0 := \mathsf{ch}_0 \oplus \mathsf{ch} \oplus \mathsf{ch}'$ and $\mathsf{resp}'_0 \leftarrow \mathsf{ReSim}_{\mathsf{ID}}(w_0, \mathsf{com}_0, \mathsf{ch}_0, \mathsf{resp}_0, \mathsf{ch}'_0)$ and outputs $\mathsf{resp}' := (\mathsf{ch}'_0, \mathsf{resp}'_0, \mathsf{resp}_1)$; Else if $b = 1$, it computes $\mathsf{ch}'_1 := \mathsf{ch}_0 \oplus \mathsf{ch}'$ and $\mathsf{resp}'_1 \leftarrow \mathsf{ReSim}_{\mathsf{ID}}(w_1, \mathsf{com}_1, \mathsf{ch}_1, \mathsf{resp}_1, \mathsf{ch}'_1)$ and outputs $\mathsf{resp}' := (\mathsf{ch}_0, \mathsf{resp}_0, \mathsf{resp}'_1)$.

Acknowledgements. This work was partially supported by JST CREST JPMJCR22M1, Japan.

References

1. 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
2. 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
3. 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
4. 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
6. 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
7. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
8. 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. Dagdelen, Ö., Venturi, D.: A second look at Fischlin's transformation. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT 14. LNCS, vol. 8469, pp. 356–376. Springer, Cham (May 2014). https://doi.org/10.1007/978-3-319-06734-6_22
10. 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
11. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Online-extractability in the quantum random-oracle model. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 677–706. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2_24
12. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat-Shamir transform. In: Galbraith, S.D., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 60–79. Springer, Berlin, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34931-7_5
13. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12

14. Fischlin, M.: On the impossibility of constructing non-interactive statistically-secret protocols from any trapdoor one-way function. In: Preneel [32], pp. 79–95. https://doi.org/10.1007/3-540-45760-7_7
15. 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
16. Fischlin, M., Fischlin, R.: The representation problem based on factoring. In: Preneel [32], pp. 96–113. https://doi.org/10.1007/3-540-45760-7_8
17. 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
18. 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)
19. Han, S., Jager, T., Kiltz, E., Liu, S., Pan, J., Riepel, D., Schäge, S.: Authenticated key exchange and signatures with tight security in the standard model. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 670–700. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84259-8_23
20. Hashimoto, K., Ogata, W., Tomita, T.: Tight reduction for generic construction of certificateless signature and its instantiation from DDH assumption. Cryptology ePrint Archive, Report 2019/1367 (2019), https://eprint.iacr.org/2019/1367
21. 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
22. Kondi, Y., shelat, a.: Improved straight-line extraction in the random oracle model with applications to signature aggregation. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part II. LNCS, vol. 13792, pp. 279–309. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22966-4_10
23. 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
24. 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
25. 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
26. 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
27. 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
28. Pan, J., Qian, C., Ringerud, M.: Signed (group) Diffie-Hellman key exchange with tight security. Journal of Cryptology **35**(4), 26 (Oct 2022). https://doi.org/10.1007/s00145-022-09438-y
29. Pan, J., Ringerud, M.: Signatures with tight multi-user security from search assumptions. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) ESORICS 2020, Part II. LNCS, vol. 12309, pp. 485–504. Springer, Cham (Sep 2020). https://doi.org/10.1007/978-3-030-59013-0_24
30. 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
31. Pass, R.: On deniability in the common reference string and random oracle model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Berlin, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_19
32. Preneel, B. (ed.): CT-RSA 2002, LNCS, vol. 2271. Springer, Berlin, Heidelberg (Feb 2002)
33. Stolbunov, A.: Cryptographic Schemes Based on Isogenies. Ph.D. thesis, Norwegian University of Science and Technology (2012), https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/262577
34. 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
35. 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

This section provides the full proof of Theorem 2.

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

$\underline{\mathsf{Game}_0}$. This is the original MU-SUF-CMA-C game. By definition, we have

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

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

$\underline{\mathsf{Game}_1}$. In this game, $\mathcal{C}$ simulates RO using three lists $L^{\text{valid}}_{\mathsf{RO}}, L^{\text{invalid}}_{\mathsf{RO}}, L^{\text{ignore}}_{\mathsf{RO}}$, 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 $\mathsf{RO}(\mathsf{pk}, m, \vec{\mathsf{com}}, j, \mathsf{ch}, \mathsf{resp})$ as follows: $\mathcal{C}$ first checks if the same query has already been issued based on $L_{\mathsf{RO}} := L^{\text{valid}}_{\mathsf{RO}} \cup L^{\text{invalid}}_{\mathsf{RO}} \cup L^{\text{ignore}}_{\mathsf{RO}}$. If so, $\mathcal{C}$ returns the consistent value. Otherwise, chooses $h \leftarrow\!\!\$\ \{0,1\}^{\gamma}$, adds a query-answer tuple $(\mathsf{pk}, m, \vec{\mathsf{com}}, j, \mathsf{ch}, \mathsf{resp}, h)$ to one of three lists according to the following conditions, and returns h as the hash value.

- When the query is issued by $\mathcal{A}$:
  - if it is a valid transcript, the tuple $(\mathsf{pk}, m, \vec{\mathsf{com}}, j, \mathsf{ch}, \mathsf{resp}, h)$ is added to $L^{\text{valid}}_{\mathsf{RO}}$.
  - if it is an invalid transcript, the tuple is added to $L^{\text{invalid}}_{\mathsf{RO}}$.
- When the query 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 $L^{\text{valid}}_{\mathsf{RO}}$,
  - if $\vec{\mathsf{com}}$ is used in the signature, but $(\mathsf{com}_j, j, \mathsf{ch}, \mathsf{resp})$ is not, then the tuple is added to $L^{\text{ignore}}_{\mathsf{RO}}$,
  - if $\vec{\mathsf{com}}$ is not used in the signature, 2^t queries $(\mathsf{pfx}, j, \mathsf{ch}', \mathsf{resp}', h')$ 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 $L^{\text{ignore}}_{\mathsf{RO}}$.
- When $\mathcal{C}$ verifies the forged signature, we consider $\mathcal{A}$ makes hash queries $(\mathsf{pk}_{i^*}, m^*, \vec{\mathsf{com}}^*, j, \mathsf{ch}^*_j, \mathsf{resp}^*_j)$ for all j. According to these queries, $L^{\text{valid}}_{\mathsf{RO}}$ or $L^{\text{invalid}}_{\mathsf{RO}}$ are updated depending on $\mathsf{V}(\mathsf{pk}_{i^*}, \mathsf{com}^*_j, \mathsf{ch}^*_j, \mathsf{resp}^*_j) = 1$ or not.

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

This change does not affect $\mathcal{A}$'s view. Therefore, we have

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

$\underline{\mathsf{Game}_2}$. 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 $L^{\text{valid}}_{\mathsf{RO}}$ to find a tuple whose prefix is pfx_τ. If such a tuple exists, skip Lines 11 to 31, i.e., move on to the next τ. Otherwise, $\mathcal{C}$ proceeds the signing process as in Game_1.

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

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

$\underline{\mathsf{Game}_3}$. In this game, we introduce a flag F_{find} which is initialized as $F_{\text{find}} := \mathsf{false}$. The simulation of RO is changed as follows: When $(\mathsf{pk}_i, m, \vec{\mathsf{com}}, j, \mathsf{ch}, \mathsf{resp}, h)$ is added to $L^{\text{valid}}_{\mathsf{RO}}$ as the response of $\mathcal{A}$'s query and

$\mathsf{OSign}(i, m)$

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

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

$\mathsf{UpdateLists}(\mathsf{pfx}_\tau, j, \{\mathsf{resp}_{\tau,j,\mathsf{ch}}, h_{\tau,j,\mathsf{ch}}\}_{\mathsf{ch}\in\{0,1\}^t}, \mathsf{ch}_{\tau,j})$

1 : **foreach** $\mathsf{ch} \in \{0,1\}^t$
2 : **if** $\mathsf{ch} = \mathsf{ch}_{\tau,j}$ **then**
3 : add $(\mathsf{pfx}_\tau, j, \mathsf{ch}, \mathsf{resp}_{\tau,j,\mathsf{ch}}, h_{\tau,j,\mathsf{ch}})$ to $L_{\mathsf{RO}}^{\mathrm{valid}}$
4 : **else**
5 : add $(\mathsf{pfx}_\tau, j, \mathsf{ch}, \mathsf{resp}_{\tau,j,\mathsf{ch}}, h_{\tau,j,\mathsf{ch}})$ to $L_{\mathsf{RO}}^{\mathrm{ignore}}$
6 : **endif**
7 : **endfor**

Fig. 7: Function UpdateLists used in OSign.

$i \notin L_{\mathsf{corr}}$ and $F_{\mathrm{find}} = \mathsf{false}$ hold, $\mathcal{C}$ browses the list $L_{\mathsf{RO}}^{\mathrm{valid}}$ and finds a tuple $(\mathsf{pk}_i, m, \vec{\mathsf{com}}, j, \mathsf{ch}', \mathsf{resp}', h')$ such that $(\mathsf{ch}', \mathsf{resp}') \neq (\mathsf{ch}, \mathsf{resp})$. If such a tuple exists, $\mathcal{C}$ sets

$$\begin{aligned} \mathit{pair} &\coloneqq (\mathsf{pk}_i, \mathsf{com}_j, \mathsf{ch}, \mathsf{resp}, \mathsf{ch}', \mathsf{resp}'), \\ F_{\mathrm{find}} &\coloneqq \mathsf{true}. \end{aligned}$$

Further, we add a condition "$F_{\mathrm{find}} = \mathsf{true}$" 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 $\mathsf{pfx}^* \coloneqq (\mathsf{pk}_{i^*}, m^*, \vec{\mathsf{com}}^*)$.

(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 $L_{\mathsf{RO}}^{\mathrm{valid}}$ 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 $F_{\mathrm{find}} = \mathsf{false}$, 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 $F_{\mathrm{find}} = \mathsf{false}$, 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 $h_j = \mathsf{H}(\mathsf{pk}_{i^*}, m^*, \vec{\mathsf{com}}^*, j, \mathsf{ch}_j, \mathsf{resp}_j) = 0^\gamma$ 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 $(\mathsf{pk}_{i^*}, m^*, \vec{\mathsf{com}}^*)$, 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 $(\mathsf{pk}, m, \vec{\mathsf{com}})$ and finally outputs $(\mathsf{pk}_{i^*}, m^*, \vec{\mathsf{com}}^*)$ as a part of the forged signature, $\Pr[\mathsf{Lucky}]$ is bounded by $(Q_{\mathsf{RO}} + 1)/2^{\rho\gamma}$, and we have

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

In the following, let $\hat{i}$ be the index of the first element of $\mathit{pair} = (\mathsf{pk}_{\hat{i}}, \ldots)$.

$\underline{\mathsf{Game}_4}$**.** In this game, $\mathcal{C}$ computes the following when F_{find} is set to true.

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

Further, we add a condition "$\mathsf{VerKey}(\mathsf{par}, \mathsf{pk}_{\hat{i}}, \mathsf{sk}^*) = 1$" to the requirements that the game outputs 1. (Note that if $F_{\mathrm{find}} = \mathsf{false}$, 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 $\mathsf{Adv}^{\mathsf{SSS}}_{\mathcal{B}_1,\mathsf{ID}}(\lambda)$ 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_4 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

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

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 numbers 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 $T_{\mathcal{B}_1} = \mathcal{O}(T_{\mathcal{A}})$.

$\underline{\mathsf{Game}_5}$**.** This game outputs 1 if $(F_{\mathrm{find}} = \mathsf{true}) \wedge (\mathsf{VerKey}(\mathsf{par}, \mathsf{pk}_{\hat{i}}, \mathsf{sk}^*) = 1)$ holds regardless of whether the forgery was successful or not.

Since the conditions $(F_{\mathrm{find}} = \mathsf{true})$ and $(\mathsf{VerKey}(\mathsf{par}, \mathsf{pk}_{\hat{i}}, \mathsf{sk}^*) = 1)$ are already included in the requirements that Game_3 outputs 1, clearly,

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

holds.

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

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

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 $(\mathsf{svk}_i, \mathsf{ssk}_i) := (\mathsf{pk}_i, (\mathsf{pk}_i, \mathsf{sk}_i))$, 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 $(\hat{i}, \mathsf{sk}^*)$.

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. $\mathsf{pk}_{\hat{i}}$ that is different from $\mathsf{sk}_{\hat{i}}$. Thus, $\mathcal{B}_2$ breaks the second key recovery resistance in the multi-user setting of ID. Therefore, we have

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

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

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

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 $\mathsf{sk}^* \neq \mathsf{sk}_{\hat{\imath}}$ is added to the requirements that the game outputs 1.

Game_6 and $\mathsf{Game}_{6'}$. These games are the same as Game_5 and $\mathsf{Game}_{5'}$ expect that Lines 13 to 14 in Figure 6 are replaced with

$$\boxed{h_{\tau,j,\mathsf{ch}} \leftarrow\!\!\$\ \{0,1\}^\gamma}$$

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

$$\boxed{\begin{array}{l}\mathsf{resp}_{\tau,j,\mathsf{ch}} \leftarrow \mathsf{P}_2(\mathsf{ch},\mathsf{st}_{\tau,j})\\ \mathsf{H}(\mathsf{pfx}_\tau, j, \mathsf{ch}, \mathsf{resp}_{\tau,j,\mathsf{ch}}) \coloneqq h_{\tau,j,\mathsf{ch}} \quad /\!/ \text{ program RO}\end{array}}$$

This is a conceptual change. Thus, we have

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

Game_7 and $\mathsf{Game}_{7'}$. In this game, we introduce another list, L^{tmp}. We delete the line before Line 5 that was added in Game_6, and replace Line 5 in Figure 7 with the next one.

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

Further, if $\mathsf{OCorr}(i)$ is queried, for each tuple $(\mathsf{pk}_i, m, \vec{\mathsf{com}}, j, \mathsf{ch}, \mathsf{resp}, h) \in L^{\text{tmp}}$, $\mathcal{C}$ programs the RO as $\mathsf{H}(\mathsf{pk}_i, m, \vec{\mathsf{com}}, j, \mathsf{ch}, \mathsf{resp}) \coloneqq h$ and moves the tuple to $L^{\text{ignore}}_{\mathsf{RO}}$. 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 $L^{\text{valid}}_{\mathsf{RO}}$, 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 $(F_{\text{find}} = \mathsf{true}) \wedge (\mathsf{VerKey}(\mathsf{pk}_i, \mathsf{sk}^*) = 1)$ holds is the same in Game_6 (resp. $\mathsf{Game}_{6'}$) and Game_7 (resp. $\mathsf{Game}_{7'}$). Thus, we have

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

Game_8 and $\mathsf{Game}_{8'}$. In this game, Line 21 in Figure 6 is replaced with

$$\boxed{\begin{array}{l}\mathsf{ch}_{\tau,j} \leftarrow\!\!\$\ \{0,1\}^t\\ \mathsf{swap}(h_{\tau,j,k}, h_{\tau,j,\mathsf{ch}_{\tau,j}})\end{array}}$$

where $\mathsf{swap}(a,b)$ 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:
$\mathbf{h} = (h_k)_{k\in\{0,1\}^t} \leftarrow\!\$\ (\{0,1\}^\gamma)^{2^t}$	$\mathbf{h}' = (h'_k)_{k\in\{0,1\}^t} \leftarrow\!\$\ (\{0,1\}^\gamma)^{2^t}$
if $h_k \neq 0^\gamma$ for all $k \in \{0,1\}^t$	if $h'_k \neq 0^\gamma$ for all $k \in \{0,1\}^t$
return $\perp$	return $\perp$
$\mathsf{ch} \leftarrow\!\$\ \{k \in \{0,1\}^t \mid h_k = 0^\gamma\}$	$\mathsf{ch}' \leftarrow\!\$\ \{k \in \{0,1\}^t \mid h'_k = 0^\gamma\}$
output $(\mathbf{h}, \mathsf{ch})$	$\mathsf{ch} \leftarrow\!\$\ \{0,1\}^t$
	define $\mathbf{h} = (h_k)_{k\in\{0,1\}^t}$ as $h_k = \begin{cases} h'_{\mathsf{ch}} & \text{if } k = \mathsf{ch}' \\ h'_{\mathsf{ch}'} & \text{if } k = \mathsf{ch} \\ h'_k & \text{otherwise} \end{cases}$
	output $(\mathbf{h}, \mathsf{ch})$

In the following, if $h_k = 0^\gamma$ for some $k \in \{0,1\}^t$, we say "$\mathbf{h}$ is good", and define $W(\mathbf{h}) \coloneqq |\{k \in \{0,1\}^t \mid h_k = 0^\gamma\}|$. 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 $\hat{\mathsf{ch}} \in \{0,1\}^t$, 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 $\hat{\mathsf{ch}}, \hat{\mathsf{ch}}' \in \{0,1\}^t$, define $\hat{\mathbf{h}}(\hat{\mathsf{ch}} \Leftrightarrow \hat{\mathsf{ch}}')$ 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

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

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

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

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 $\Pr[\mathsf{ch} = \hat{\mathsf{ch}}] = \frac{1}{2^t}$. Thus,

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

which is the same as in Experiment A.

Therefore, we have

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

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

Game_9 and $\mathsf{Game}_{9'}$. In this game, OSign and UpdateLists are further modified as in Figure 8 and Figure 9, respectively. The difference is that the challenge $\mathsf{ch}_{\tau,j}$ is chosen and $\mathsf{resp}_{\tau,j,\mathsf{ch}}$ for $\mathsf{ch} = \mathsf{ch}_{\tau,j}$ is calculated immediately after $\mathsf{com}_{\tau,j}$ is computed.

This change is completely conceptual. Thus, we have

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

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

- ReSim takes a valid key pair $(\mathsf{pk}, \mathsf{sk})$ and a valid transcript $(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \in \mathsf{Tran}(\mathsf{pk}, \mathsf{sk}, \mathsf{ch})$ 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 $(\mathsf{com}, \mathsf{st}) \leftarrow \mathsf{P}_1(\mathsf{pk}, \mathsf{sk}; r)$,
- computes $\mathsf{resp}'_{\mathsf{ch}'} \leftarrow \mathsf{P}_2(\mathsf{st}, \mathsf{ch}')$ for all $\mathsf{ch}' \in \{0,1\}^t \setminus \{\mathsf{ch}\}$, and
- outputs a list $(\mathsf{resp}'_{\mathsf{ch}'})_{\mathsf{ch}' \in \{0,1\}^t \setminus \{\mathsf{ch}\}}$.

If the identification scheme is perfect HVZK, there exists r such that $(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow \mathsf{Tran}(\mathsf{pk}, \mathsf{sk}, \mathsf{ch}; r)$ for any transcript $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ 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.

$$\boxed{\begin{array}{l}\mathsf{ch}_{\tau,j} \leftarrow\!\!\$\ \{0,1\}^t \\ (\mathsf{com}_{\tau,j}, \mathsf{resp}_{\tau,j,\mathsf{ch}_{\tau,j}}) \leftarrow \mathsf{Sim}(\mathsf{pk}_i, \mathsf{ch}_{\tau,j})\end{array}}$$

Further, insert the next line

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

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

$$\boxed{\mathsf{resp}_{\tau,j,\mathsf{ch}} := \mathsf{resp}'_{\mathsf{ch}}}$$

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

$$\Pr[\mathsf{Game}_{10} \Rightarrow 1] = \Pr[\mathsf{Game}_9 \Rightarrow 1],$$
$$\Pr[\mathsf{Game}_{10'} \Rightarrow 1] = \Pr[\mathsf{Game}_{9'} \Rightarrow 1].$$

In Game_{10} and $\mathsf{Game}_{10'}$, sk_i is used only in 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 $\hat{\imath}$, by the time F_{find} is set true and *pair* is assigned, $\hat{\imath}$ has not been corrupted and $\mathcal{A}$ has no information about $\mathsf{sk}_{\hat{\imath}}$. Thus we have

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

This completes the proof of Lemma 1. □

$\mathsf{OSign}(i, m)$

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

Fig. 8: The sign oracle in Game_9 and $\mathsf{Game}_{9'}$.

UpdateLists($\mathsf{pfx}_\tau, j, \{\mathsf{resp}_{\tau,j,\mathsf{ch}}, h_{\tau,j,\mathsf{ch}}\}_{\mathsf{ch}}, \mathsf{ch}_{\tau,j}$)

1 : **foreach** $\mathsf{ch} \in \{0,1\}^t$
2 : **if** $\mathsf{ch} = \mathsf{ch}_{\tau,j}$ **then**
3 : $\mathsf{H}(\mathsf{pfx}_\tau, j, \mathsf{ch}, \mathsf{resp}_{\tau,j,\mathsf{ch}}) := h_{\tau,j,\mathsf{ch}}$ // program RO
4 : add $(\mathsf{pfx}_\tau, j, \mathsf{ch}, \mathsf{resp}_{\tau,j,\mathsf{ch}}, h_{\tau,j,\mathsf{ch}})$ to $L_{\mathsf{RO}}^{\mathrm{valid}}$
5 : **else**
6 : $\mathsf{resp}_{\tau,j,\mathsf{ch}} \leftarrow \mathsf{P}_2(\mathsf{ch}, \mathsf{st}_{\tau,j})$
7 : add $(\mathsf{pfx}_\tau, j, \mathsf{ch}, \mathsf{resp}_{\tau,j,\mathsf{ch}}, h_{\tau,j,\mathsf{ch}})$ to L^{tmp}
8 : **endif**
9 : **endfor**

Fig. 9: Function UpdateLists used in Game_9.

B Cryptographic Group Action

Here, we recall cryptographic group action.

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 properties.

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 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 $\mathcal{G}$ 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 $\mathcal{G}$. The sampling method is required to be statistically indistinguishable from the uniform distribution over $\mathcal{G}$.*
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 effective group action $(\mathcal{G}, \mathcal{E}, E_0, \star)$. The next hardness assumption on group action is often used.

Definition 17 (Group Action Discrete Logarithm (GADL) Assumption [24, Definition 16]). *We say that* GADL assumption holds for GAGen *if for all* PPT *adversaries* $\mathcal{A}$*, it holds that*

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

Table of Contents