Strong Existential Unforgeability and More of MPC-in-the-Head Signatures

मुकुल कुळकर्णी (Mukul Kulkarni) and 草川 恵太 (Keita Xagawa)

Technology Innovation Institute, UAE
{mukul.kulkarni,keita.xagawa}@tii.ae

July 1, 2024

Abstract. NIST started the standardization of additional post-quantum signatures in 2022. Among 40 candidates, a few of them showed their stronger security than existential unforgeability, strong existential unforgeability and BUFF (beyond unforgeability features) securities. Recently, Aulbach, Düzlü, Meyer, Struck, and Weishäupl (PQCrypto 2024) examined the BUFF securities of 17 out of 40 candidates. Unfortunately, on the so-called MPC-in-the-Head (MPCitH) signature schemes, we have no knowledge of strong existential unforgeability and BUFF securities.
This paper studies the strong securities of all nine MPCitH signature candidates: AIMer, Biscuit, FAEST, MIRA, MiRitH, MQOM, PERK, RYDE, and SDitH.
We show that the MPCitH signature schemes are strongly existentially unforgeable under chosen message attacks in the (quantum) random oracle model. To do so, we introduce a new property of the underlying multi-pass identification, which we call *non-divergency*. This property can be considered as a weakened version of the computational unique response for three-pass identification defined by Kiltz, Lyubashevsky, and Schaffner (EUROCRYPT 2018) and its extension to multi-pass identification defined by Don, Fehr, and Majentz (CRYPTO 2020). In addition, we show that the SSH11 protocol proposed by Sakumoto, Shirai, and Hiwatari (CRYPTO 2011) is *not* computational unique response, while Don et al. (CRYPTO 2020) claimed it.
We also survey BUFF securities of the nine MPCitH candidates in the quantum random oracle model. In particular, we show that Biscuit and MiRitH do not have some of the BUFF security.

Keywords: signature · strong existential unforgeability under chosen message attacks · BUFF securities · MPC-in-the-Head signature · quantum random oracle model (QROM)

Table of Contents

1 Introduction

MPC-in-the-Head signatures: To prepare post-quantum cryptography (PQC), which is expected to resist threats of quantum machines against public-key cryptography based on factoring and discrete logarithms, NIST has been standardizing PQC signature schemes[1]. After they selected three digital signature schemes in July 2022, they started an additional PQC signature standardization in Sept. 2022 [NIS22][2]. NIST announced forty additional signature candidates in July 2023.

There are several approaches in those forty signature schemes. One of the promising approaches is MPC-in-the-Head (MPCitH)[3] signatures, which employ the combination of the Fiat-Shamir (FS) transform [FS87] and the zero-knowledge protocol based on the MPCitH paradigm [IKOS07] (or its followers). Nine of the forty candidates are MPCitH signatures: AIMer [KCC+23], Biscuit [BKPV23], FAEST [BBd+23a], MIRA [ABB+23c], MiRitH [ARV+23], MQOM [FR23], PERK [ABB+23a], RYDE [ABB+23b], and SDitH [AFG+23]. See Table 1 for the summary. In this paper, we focus on those nine MPCitH signature schemes.

Background 1: Strong existential unforgeability: The standard security notion for signature is *existential unforgeability under chosen-message attack*, EUF-CMA security in short; roughly speaking, the security states that any efficient adversary cannot forge a signature on *new* message while it can obtain arbitrary signature on its chosen messages. This notion is the basic requirement for the signature schemes and suffices for basic applications of the signature.

However, we sometimes need stronger security notions. One of such notions is *strong* existential unforgeability under chosen-message attack, sEUF-CMA security in short; this security states that any efficient adversary cannot produce a new signature on a message, while the adversary may obtain signatures on the message. This strong security has applications such as chosen-ciphertext secure public-key encrytpion [DDN00, CHK04], authenticated group key exchange [KY03], and unilaterally-authenticated key exchange [DF17].

Suppose we want to upgrade the EUF-CMA security of a signature scheme to the sEUF-CMA security. In that case, we need to employ an additional cryptographic primitive, e.g., a strongly secure one-time signature scheme by following the general transform by Huang, Wong, and Zhao [HWZ07] or by Bellare and Shoup [BS07]. Roughly speaking, those transforms make a signature longer by adding a verification key and signature of a one-time signature scheme. The obtained new signature will be longer when we employ the hash-based one-time signature, e.g., the Lamport one-time signature [Lam79]. Hence, it is important to show the sEUF-CMA security of signature schemes *directly*.

Let us consider a signature scheme based on a three-pass identification scheme via the Fiat-Shamir transform (with or without aborts) [FS87, Lyu09]. In order to show the sEUF-CMA security of such schemes in the random oracle model (ROM) and in the quantum ROM (QROM), we need the underlying three-pass ID scheme to be *computational unique response* (CUR) [KLS18].[4] See e.g., [AFLT16, KLS18, DGJL21, DFPS23, Xag24].

Often, MPCitH signature schemes are based on five/seven-pass ID schemes. Don, Fehr, and Majentz [DFM20] extended the sEUF-CMA proof for three-pass ID into that for $(2n + 1)$-pass ID. Concretely speaking, they considered MQDSS [SCH+17], whose underlying ID is the five-pass SSH11 protocol [SSH11]; they showed the sEUF-CMA security in the QROM by using their extended CUR and insisted that the SSH11 protocol satisfies the extended CUR. Unfortunately, the SSH11 protocol does not satisfy the extended CUR. (See Section 3 for the details.) It is also hard to show that the underlying ID protocols of the MPCitH signature satisfy the extended CUR in a modular fashion. The required property is too strong to achieve while their sEUF-CMA security proof is correct. Our first question is:

> *Are the MPCitH signature schemes* sEUF-CMA *secure in the (Q)ROM? How can we weaken the property of the underlying protocol?*

Background 2: BUFF securities: We also consider more enhanced security notions against malicious key generations; exclusive ownership (M-S-UEO, S-CEO, and S-DEO) [BWM99, MS04, PS05, CDF+21], message-bound signatures (MBS) [PS05, JCCS19, BCJZ21, CDF+21], and non-resignability (NR) [PS05, JCCS19, BCJZ21, CDF+21]. Exclusive ownership requires that a signature is valid only under a single verification key. This prevents an attacker makes another verification key to "hijack" the signature (and some messages). Message-bound signature requires that a signature is valid only under a single message and prevents an attacker from making a weak verification key that allows the verification of a signature under multiple messages.[5] Non-resignability requires that, given a verification key and a signature on a hidden random message, any adversary cannot output a signature and a different valid verification key on the same message.

[1] https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization

[2] https://csrc.nist.gov/Projects/pqc-dig-sig/standardization

[3] MPC = Multi-Party Computation

[4] Any efficient adversary cannot output two valid transcripts (a, c, z) and (a, c, z') with $z \neq z'$.

[5] See example for ECDSA in [SPMS02].

Table 1. Security comparison of the MPCitH signature schemes in Round 1 of the NIST additional PQC signature standardization. "✓" implies that there exists a security proof under appropriate assumptions. "✗" implies that there exists an attack with a success probability larger than $2^{-\kappa}$ with a number of queries 2^{64}, where $\kappa \in \{128, 192, 256\}$ is the security parameter. "✗?" implies that the success probability depends on the parameter sets. "?" implies that showing the security is an open problem.

Name	sEUF	MS-UEO	MBS	wNR	Section	Ref.	version
AIMer	✓	✓	✓	✓	Section F	[KCC+23]	v1.0
Biscuit	✓	✗	✓	✗?	Section 6	[BKPV23]	v1.1
FAEST	✓	✓	✓	✓	subsection H.1	[BBd+23a]	v1.1
MIRA	✓	✓	✓	✓	subsection G.1	[ABB+23c]	v1.0
MiRitH	✓	✗	✓	?	Section D	[ARV+23]	v1.0
MQOM	✓	✓	✓	✓	subsection G.3	[FR23]	v1.0
PERK	✓	✓	✓	✓	Section E	[ABB+23a]	v1.1
RYDE	✓	✓	✓	✓	subsection G.1	[ABB+23b]	v1.0
SDitH	✓	✓	✓	✓	subsection G.2	[AFG+23]	v1.1

In their call for proposal, NIST suggested BUFF securities as desirable properties as well as side-channel-attack resistance, security in the multi-key setting, and misuse-resistance property [NIS22, 4.B.4]. Cremers, Düzlü, Fiedler, Fischlin, Janson [CDF+21] studied all six round-3 candidate signature schemes of NIST PQC standardization. Aulbach, Düzlü, Meyer, Struck, Weishäupl [ADM+24] studied BUFF securities of seventeen signature schemes based on code, isogeny, lattice, or MQ in forty Round-1 candidates of NIST PQC additional signature standardization. To the authors' best knowledge, there are no studies on BUFF securities of the MPCitH signature schemes. Our second question is:

Do the MPCitH signature schemes satisfy BUFF securities?

1.1 Our Contribution

In this paper, we show the MPCitH signature schemes are sEUF-CMA-secure in the (Q)ROM; the assumptions are

1. existential unforgeability under no-message attacks (EUF-NMA security) of the signature scheme in the (Q)ROM,
2. computational honest-verifier zero-knowledge (HVZK) property of the underlying ID protocol, and
3. the non-divergency of the underlying ID protocol,

where *non-divergency* is the weakened version of CUR defined later.
In addition, we survey the BUFF securities of the MPCitH signature schemes and found that the two schemes, Biscuit and MiRitH, do not satisfy some exclusive ownership properties. For comparisons, see Table 1.

1.2 Technical Overview

Let us briefly remind the Fiat-Shamir (FS) transform applied to a $(2n+1)$-pass ID scheme [FS87, EDV+12, DGV+16, CHR+16]: Let $(a_1, c_1, \dots, a_n, c_n, a_{n+1})$ denote a transcript of the underlying ID scheme, where $a_1, \dots, a_{n+1}$ are the messages generated by the prover and $c_1, \dots, c_n$ be public-coin challenges generated by the verifier. On a message μ, the signer sequentially computes the prover's messages $a_1, \dots, a_{n+1}$ by computing the challenges as $c_1 = \mathsf{H}(\mu, a_1)$ and $c_i = \mathsf{H}(i, c_{i-1}, a_i)$ for $i = 2, \dots, n$, where H is the random oracle, and outputs a signature $(a_1, a_2, \dots, a_{n+1})$. The verifier verifies the transcript $(a_1, c_1, \dots, a_n, c_n, a_{n+1})$ via the ID's verification algorithm by computing $c_1 = \mathsf{H}(\mu, a_1)$ and $c_i = \mathsf{H}(i, c_{i-1}, a_i)$ for $i = 2, \dots, n$.

Strong Existential Unforgeability:

Existential unforgeability for 3*-pass ID via reprogramming:* Let us start from the EUF-CMA security proof in the QROM by Grilo, Hövelmanns, Hülsing, and Majenz [GHHM21]. Roughly speaking, they consider the games defined as follows:

- G_0: This is the original EUF-CMA game. The adversary is given vk and has access to the signing oracle. The signing oracle on input μ computes $a_1, c_1 := \mathsf{H}(\mu, a_1)$, and a_2, and returns (a_1, a_2) as a signature. The adversary outputs μ^* and a signature (a_1^*, a_2^*). If it is valid and μ^* is *new*, then the adversary wins.

- G_1: In this game, the challenge c_1 is chosen uniformly at random, and the random oracle is reprogrammed, that is, the hash value of (μ, a_1) is overwritten by c_1. This modification is justified by the adaptive reprogramming technique [GHHM21].
- G_2: In this game, the signing oracle is implemented by the simulator of HVZK. This modification is justified by the HVZK property of the ID.

Due to the EUF-CMA security condition, the adversary should output a new message μ^* never signed before. The challenge c_1^* is comptued as $\mathsf{H}(\mu, a_1^*)$ and this point (μ^*, a_1^*) in the random oracle H is *never reprogrammed* by the challenger since μ^* is new. Therefore, we can construct an EUF-NMA adversary against the signature scheme using the adversary in G_2.

Strong existential unforgeability for 3-pass ID via reprogramming: The situation is a bit changed when we consider the sEUF-CMA security. In the game, the adversary wins if it outputs μ^* and a signature (a_1^*, a_2^*) such that $(\mu^*, (a_1^*, a_2^*))$ is not answered by the signing oracle. Therefore, the adversary can ask μ^* to the signing oracle. Hence, the point (μ^*, a_1^*) might be reprogrammed since μ^* can be queried to the signing oracle. To eliminate this event, we consider an additional game G_3 defined as follows:

- G_3: In this game, the adversary loses if the signing oracle returned answer (a_1^*, a_2) with $a_2^* \neq a_2$ on the query μ^*.

In G_3, the adversary cannot win by outputting a signature that causes the reprogramming. Thus, it is easy to construct an EUF-NMA security against the signature scheme again. To treat this event, Kiltz, Lyubashevsky, and Schafnner [KLS18] defined computational unique response (CUR), in which the adversary is given a verification key and requested to produce (a_1, c_1, a_2, a_2') with $a_2 \neq a_2'$ such that (a_1, c_1, a_2) and (a_1, c_1, a_2') are valid under the verification key.[6]

Strong existential unforgeability for 5-pass ID: We need careful analysis when we consider the multi-pass ID case. Let us consider the 5-pass ID case as an example. On a signing query with message μ, the singing oracle will reprogram values corresponding two points (μ, a_1) and $(2, c_1, a_2)$ as c_1 and c_2, respectively, to produce a signature (a_1, a_2, a_3). If the adversary's forgery $(\mu^*, (a_1^*, a_2^*, a_3^*))$ with challenges $c_1^* = \mathsf{H}(\mu^*, a_1^*)$ and $c_2^* = \mathsf{H}(2, c_1^*, a_2^*)$ is related to the signing oracle's signature (a_1, a_2, a_3) with challenges c_1 and c_2 on a message μ, then the tuple $(\mu, a_1, c_1, a_2, c_2, a_3)$ is classified into the following three cases:

- case 1: $(\mu, a_1, c_1) = (\mu^*, a_1^*, c_1^*)$ and $a_2 \neq a_2^*$;
- case 2: $(\mu, a_1, c_1, a_2, c_2) = (\mu^*, a_1^*, c_1^*, a_2^*, c_2^*)$ and $a_3 \neq a_3^*$; or
- case 3: $(c_1, a_2, c_2) = (c_1^*, a_2^*, c_2^*)$ and $(\mu, a_1) \neq (\mu^*, a_1^*)$.

Fortunately, the third case can be eliminated by using the collision-resistance property of H because, if so, we have $\mathsf{H}(\mu, a_1) = c_1^* = \mathsf{H}(\mu^*, a_1^*)$ with $(\mu, a_1) \neq (\mu^*, a_1^*)$. Therefore, we need to introduce the game to exclude cases 1 and 2 and to define the generalization of CUR.

CUR for $(2n + 1)$-pass ID: Don, Fehr, and Majentz [DFM20] defined CUR for $(2n + 1)$-pass ID. In their definition with $2n + 1 = 5$, the adversary is requested to output two valid transcripts $(a_1, c_1, a_2, c_2, a_3)$ and $(a_1', c_1', a_2', c_2', a_3')$ (and a verification key) such that

- condition 1: $(a_1, c_1) = (a_1', c_1')$ and $a_2 \neq a_2'$; or
- condition 2: $(a_1, c_1, a_2, c_2) = (a_1', c_1', a_2', c_2')$ and $a_3 \neq a_3'$.

Thus, these conditions are what we want to use to eliminate cases 1 and 2. They argued that the 5-pass Sakumoto-Shirai-Hiwatari (SSH11) protocol [SSH11] satisfies their CUR notion and MQDSS [SCH+19], which is obtained by applying the FS transform to the SSH11 protocol, is sEUF-CMA-secure in the QROM under appropriate assumptions.
Unfortunately, the SSH11 protocol is *not* CUR (for the detail, see subsection 3.1). Hence, we must weaken the CUR property to rescue the sEUF-CMA security of MQDSS. Since we can use the collision-resistance property, we could weaken the notion while keeping the security proof by modifying the first condition as:

- condition 1′: $(a_1, c_1) = (a_1', c_1')$, $a_2 \neq a_2'$, and $c_2 \neq c_2'$.

However, we can still show the parallel version of the SSH11 protocol does not satisfy this modified CUR property. In addition, even this stronger condition 1′ is problematic in the context of the underlying ID protocols of the MPCitH signature schemes. We must care about the underlying problem's structure; sometimes, we fail to show this modified CUR.

[6] Their definition is concerning honestly generated verification key.

Non-Divergency: Turning back to the situation in G_3, we observe that one of the two valid transcripts should be generated by *the HVZK simulator.* So, we modify the definition of CUR and dub it *non-divergency.* Roughly speaking, we say that a 5-pass ID is *non-divergent* if an adversary having access to the simulation oracle cannot output a valid transcript $(a_1, c_1, a_2, c_2, a_3)$ and another transcript $(a_1', c_1', a_2', c_2', a_3')$ generated by the simulation oracle satisfying the conditions $1'$ or 2. To treat 7-pass ID schemes and variants of the FS transform, the real conditions differ from the above. See the concrete definition in Section 3.
In the context of MPCitH protocols, if the condition $1'$ is met, then we have $c_2 \neq c_2'$, and the adversary should open a commitment unopened in the simulated transcript, which breaks the one-wayness of the commitment scheme. If the condition 2 is met, then $a_3 \neq a_3'$ implies the violation of the binding property of the commitment or the collision-resistance property of PRG or hash functions.

Does collapsed 3-pass ID help? One might consider that the following approach solves the above problems: Let us consider collapsed 3-pass ID $\mathsf{ID3}$ as Aguilar-Melchor, Hülsing, Joseph, Majenz, Ronen, and Yue [AHJ+23], in which the first prover computes $w = (a_1, a_2)$ by computing $c_1 := \mathsf{H}'(vk, a_1)$ by itself, the verifier chooses a random challenge $c = c_2$, the second prover computes $z = a_3$, and the verifier checks if $\mathsf{V}(vk, a_1, c_1, a_2, c_2, a_3)$ by computing $c_1 := \mathsf{H}'(vk, a_1)$. Applying the Fiat-Shamir transform to $\mathsf{ID3}$, we obtain the signature scheme $\mathsf{FS3}[\mathsf{ID3}, \mathsf{H}]$, where the signer will compute $w = (a_1, a_2)$, $c = \mathsf{H}(\mu, w) = \mathsf{H}(\mu, a_1, a_2)$, and $z = a_3$, and output $\sigma = (w, z)$ (or (c, z)). They showed that the obtained signature scheme is EUF-CMA-secure in the QROM by assuming that the signature is EUF-NMA-secure in the QROM and the HVZK property and the min-entropy of the commitment of the collapsed ID according to Grilo et al. [GHHM21, Thm.3]. They then showed that the collapsed 3-pass ID is EUF-NMA-secure in the QROM by assuming that the underlying problem is hard. We can also show that if the collapsed 3-pass ID is CUR additionally, then the signature scheme is also sEUF-CMA-secure in the QROM.
While the above argument is fine, what we want to treat is the signature scheme obtained from 5-pass ID $\mathsf{ID5}$ because the proposed scheme SDitH is defined as a variant of $\mathsf{FS5}[\mathsf{ID5}, \mathsf{H}]$, where $c_1 = \mathsf{H}(vk, a_1)$ and $c_2 = \mathsf{H}(\mu, c_1, a_2)$ (see Table 2 and subsection G.2 for the details) and there is a subtle gap on how to compute c_2 ($\mathsf{H}(\mu, a_1, a_2)$ or $\mathsf{H}(\mu, c_1, a_2)$). When we prove the sEUF-CMA security of the real signature as the security proof by Grilo et al. [GHHM21, Thm.3], this subtle difference introduces the following possibility: the adversary could output a forgery $(a_1^*, c_1, a_2, c_2, a_3)$ on μ^* while the siging oracle generates a signature $(a_1, c_1, a_2, c_2, a_3)$ on μ^* and $(a_1^*, c_1^*) \neq (a_1, c_1)$. In this case, the forgery involves the point (c_1, a_2) reprogrammed by the signing oracle, and the CUR for 3-pass ID does not help us.
Hülsing, Joseph, Majenz, and Narayanan [HJMN24] recently generalized the above approach to suitable $(2n+1)$-pass IDs and insisted their approach can be applicable to several MPCitH signatures in particular RYDE. We note that the above approach for the EUF-CMA security invokes the fact that the computation of c_2 involves μ, e.g., $c_2 := \mathsf{H}(\mu, c_1, a_2)$, to exclude the event the point (μ^*, c_1^*, a_2^*) is not reprogrammed by the signing oracle. Hence, we will require a few arguments if c_2 does not involve μ directly as AIMer and $\mathsf{Biscuit}$.

BUFF securities: We also examine the BUFF securities of the nine MPCitH signatures because there are differences in the form of a signature and inputs to the hash functions. See Table 2 for the summary of differences. Very roughly speaking, a signature contains the hash values, and it essentially uses the transforms in [PS05, CDF+21]. A signature of all signature schemes containing the hash values involving a message μ. Hence, a weak version of exclusive ownership (S-DEO) and MBS are easily satisfied. If those hash values include a verification key vk too, then it (almost) automatically satisfies exclusive ownership (M-S-UEO) and another weak version (S-CEO). It also satisfies weak non-resignability (wNR).
AIMer, FAEST, MIRA, MQOM, PERK, RYDE, and SDitH satisfy M-S-UEO (under appropriate assumptions) since their hash values include μ and vk as in Table 2. In addition, we can show that they also satisfy wNR under appropriate assumptions since their hash values *in a signature* include μ and vk as in Table 2.
We then examine $\mathsf{Biscuit}$ and MiRitH where vk is not involved in the hash values. Curiously, we find that $\mathsf{Biscuit}$ and MiRitH are vulnerable to S-CEO and M-S-UEO. Very roughly speaking, we propose an attack computing a new verification key vk' when we can obtain many pairs of a message and a signature, say, 2^{64} pairs. The wNR insecurity depends on the parameter sets because we can obtain a *single* pair of a message and signature, while polynomially, many pairs are obtained in the S-CEO and M-S-UEO settings.
See the summary of the securities Table 1.

1.3 Organization

Section 2 reviews basic notations, notions, definitions, and lemmas used in this paper. Section 3 discusses unique response and non-divergency of ID. Section 4 gives our main theorem showing that a signature scheme from multi-pass ID achieves strong unforgeability. Section 5 discusses a variant of the Fiat-Shamir transform used in the MPCitH signature schemes. Section 6 studies $\mathsf{Biscuit}$ as an example of the MPCitH signature

Table 2. Comparison of the candidates in Round 1 of the NIST additional PQC signature standardization. vk is the verification key and μ is the message to be signed. h_i's are hash values and c_i's are challenges computed from the hash values. The last message a_3 or a_4 contains salt.

Name	#pass	h_1 or c_1	h_2 or c_2	h_3 or c_3	σ	Ref.
AIMer	5	μ, vk, a_1	h_1, a_2	–	(h_1, h_2, a_3)	[KCC+23]
Biscuit	5	salt, μ, a_1	salt, h_1, a_2	–	(h_1, h_2, a_3)	[BKPV23]
FAEST	7	salt, $H(\mu, vk)$, a_1	c_1, a_2	c_2, a_3	(h_3, a_4)	[BBd+23a]
MIRA	5	salt, $H(\mu)$, vk, a_1	salt, $H(\mu)$, vk, h_1, a_2	–	(h_1, h_2, a_3)	[ABB+23c]
MiRitH	5	salt, μ, a_1	salt, μ, h_1, a_2	–	(h_1, h_2, a_3)	[ARV+23]
MQOM	7	salt, μ, vk, a_1	salt, μ, h_1, a_2	salt, μ, h_2, a_3	(h_1, h_2, h_3, a_4)	[FR23]
PERK	5	salt, μ, vk, a_1	salt, μ, vk, h_1, a_2	–	(h_1, h_2, a_3)	[ABB+23a]
RYDE	5	salt, $H(\mu)$, vk, a_1	salt, $H(\mu)$, vk, h_1, a_2	–	(h_1, h_2, a_3)	[ABB+23b]
SDitH	5 (3)	salt, vk, a_1	salt, μ, h_1, a_2	–	(h_2, a_3)	[AFG+23]

schemes. Supplement material contains missing definitions, a variant of the FS transform, and studies of other signature schemes. Section A contains missing definitions and proofs. Section B discusses another variant of the Fiat-Shamir transform used in FAEST and SDitH. Section C studies sEUF-CMA security of MQDSS. Section D studies (in)securities of MiRitH, Section E and Section F shows the security of PERK and AIMer, respectively. Section G treats MIRA, RYDE, SDitH, and MQOM. Finally, Section H discusses the security of the VOLEitH signature and its instantiation FAEST.

2 Preliminaries

The security parameter is denoted by $\kappa \in \mathbb{Z}^+$. We use the standard O-notations. For $n \in \mathbb{Z}^+$, we let $[n] := \{1, \dots, n\}$. For a statement P, $\mathsf{boole}(P)$ denotes the truth value of P. DPT and QPT stand for deterministic polynomial time and quantum polynomial time, respectively.
Let $\mathcal{X}$ and $\mathcal{H}$ be two finite sets. $\mathsf{Func}(\mathcal{X}, \mathcal{H})$ denotes a set of all functions whose domain is $\mathcal{X}$ and codomain is $\mathcal{H}$.
For a distribution D, we often write "$x \leftarrow D$," which indicates that we take a sample x according to D. For a finite set $\mathcal{S}$, $U(\mathcal{S})$ denotes the uniform distribution over $\mathcal{S}$. We often write "$x \leftarrow \mathcal{S}$" instead of "$x \leftarrow U(\mathcal{S})$." If inp is a string, then "$\mathsf{out} \leftarrow \mathsf{A}^{O}(\mathsf{inp})$" denotes the output of algorithm A running on input inp with an access to a set of oracles O. If A and oracles are deterministic, then out is a fixed value and we write "$\mathsf{out} := \mathsf{A}^{O}(\mathsf{inp})$." We also use the notation "$\mathsf{out} := \mathsf{A}(\mathsf{inp}; r)$" to make the randomness r of A explicit.
For a function $f\colon \{0,1\}^n \to \{0,1\}^m$, a *quantum access* to f is modeled as oracle access to unitary $O_f\colon |x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$. By convention, we will use the notation $\mathsf{A}^{|f\rangle, g}$ to stress A's *quantum* and classical access to f and g. For a function $f\colon \mathcal{X} \to \mathcal{H}$, we denote the procedure reprogramming $f(x)$ with h by $f := f[x \mapsto h]$.

2.1 Lemmas on Quantum Random Oracles

We use the following two lemmas on quantum random oracles:
Zhandry [Zha15] showed the following lemma on the collision resistance of QROM.

Lemma 2.1 ([Zha15, Thm.3.1] and [Zha12, Cor.7.5]). *Let $\mathsf{H}\colon \mathcal{X} \to \mathcal{Y}$ be a random function. Then any algorithm that makes q quantum queries to H outputs a collision for H with probability at most $632(q+1)^3/|\mathcal{Y}|$.* [7]

Grilo et al. showed that one cannot distinguish whether the random oracle is reprogrammed or not if the min-entropy of the reprogrammed point is sufficiently high [GHHM21].

Lemma 2.2 ([GHHM21, Prop.1]). *Let $\mathcal{X}_1$, $\mathcal{X}_2$, and $\mathcal{H}$ be finite sets. Let $\mathcal{A}$ be an adversary that makes R queries to* Reprogram *and q quantum queries to $|\mathsf{O}_b\rangle$. Then, the distinguishing advantage of $\mathcal{A}$ is bounded by*

$$|\Pr[\mathsf{Repro}_0 = 1] - \Pr[\mathsf{Repro}_1 = 1]| \le \frac{3R}{2}\sqrt{q/|\mathcal{X}_1|},$$

where Repro_b and Reprogram *is defined in Figure 1.*

[7] The constant $632 > 24 \cdot \pi^2 2^3/3$ is taken from $C = 24C'$ in the proof of [Zha15, Thm.3.1] for general $\mathcal{X}$ and $\mathcal{Y}$ with $\#\mathcal{X} > \#\mathcal{Y}$ and $C' = \pi^2 2^3/3$ in [Zha12, Cor.7.5].

$\underline{\mathsf{Game\ Repro}_b}$
$\mathsf{O}_0 \leftarrow \mathsf{Func}(\mathcal{X}_1 \times \mathcal{X}_2, \mathcal{H})$
$\mathsf{O}_1 := \mathsf{O}_0$
$b' \leftarrow \mathcal{A}^{|\mathsf{O}_b\rangle, \textsc{Reprogram}}()$
return b'

$\underline{\textsc{Reprogram}(x_2)}$
$x_1 \leftarrow \mathcal{X}_1$
$y \leftarrow \mathcal{H}$
$\mathsf{O}_1 := \mathsf{O}_1[(x_1, x_2) \mapsto y]$
return x_1

Fig. 1. Adaptive reprogramming games Repro_b for bit $b \in \{0,1\}$ and Reprogram.

$\underline{\mathsf{Expt}^{\text{euf-cma}}_{\mathsf{DS},\mathcal{A}}(1^\kappa)}$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$\mathcal{Q} := \emptyset$
$(\mu^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}}(vk)$
if $\exists \sigma : (\mu^*, \sigma) \in \mathcal{Q}$ **then**
 return false
return $\mathsf{Vrfy}(vk, \mu^*, \sigma^*)$

$\underline{\mathsf{Expt}^{\text{seuf-cma}}_{\mathsf{DS},\mathcal{A}}(1^\kappa)}$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$\mathcal{Q} := \emptyset$;
$(\mu^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}}(vk)$
if $(\mu^*, \sigma^*) \in \mathcal{Q}$ **then**
 return false
return $\mathsf{Vrfy}(vk, \mu^*, \sigma^*)$

$\underline{\textsc{Sign}(\mu)}$
$\sigma \leftarrow \mathsf{Sign}(sk, \mu)$
$\mathcal{Q} := \mathcal{Q} \cup \{(\mu, \sigma)\}$
return σ

Fig. 2. Security games for EUF-CMA and sEUF-CMA security.

2.2 Digital Signature

The model for digital signature schemes is summarized as follows:

Definition 2.1. *A digital signature scheme* DS *consists of the following triple of PPT algorithms* (Gen, Sign, Vrfy)*:*

- $\mathsf{Gen}(1^\kappa) \to (vk, sk)$*: a key-generation algorithm that, on input* 1^κ*, where* κ *is the security parameter, outputs a pair of keys* (vk, sk)*.* vk *and* sk *are verification and signing keys, respectively.*
- $\mathsf{Sign}(sk, \mu) \to \sigma$*: a signing algorithm that takes as input signing key* sk *and message* $\mu \in \mathcal{M}$ *and outputs signature* $\sigma \in \mathcal{S}$*.*
- $\mathsf{Vrfy}(vk, \mu, \sigma) \to \mathsf{true}/\mathsf{false}$*: a verification algorithm that takes as input verification key* vk*, message* $\mu \in \mathcal{M}$*, and signature* σ *and outputs its decision* true *or* false*.*

We require statistical correctness; that is, for any message $\mu \in \mathcal{M}$*, we have*

$$\Pr[(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa), \sigma \leftarrow \mathsf{Sign}(sk, \mu) : \mathsf{Vrfy}(vk, \mu, \sigma) = \mathsf{true}] \geq 1 - \delta(\kappa)$$

for some negligible function δ*.*

Standard security notions: We review the standard security notion, existential unforgeability against chosen-message attack (EUF-CMA), and its variants. We consider a weak version, existential unforgeability against known-message attack (EUF-NMA), in which the adversary cannot access the signing oracle. We also consider a strong version, sEUF-CMA security, in which the adversary wins if its forgery (μ^*, σ^*) is not equal to the pairs returned by Sign. The formal definition follows:

Definition 2.2 (EUF-CMA, sEUF-CMA, **and** EUF-NMA **security).** *Let* $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *be a digital signature scheme. For any* $\mathcal{A}$ *and* goal $\in$ {euf, seuf}*, we define its* goal-cma *advantage against* DS *as*

$$\mathsf{Adv}^{\text{goal-cma}}_{\mathsf{DS},\mathcal{A}}(\kappa) := \Pr[\mathsf{Expt}^{\text{goal-cma}}_{\mathsf{DS},\mathcal{A}}(1^\kappa) = 1],$$

where $\mathsf{Expt}^{\text{goal-cma}}_{\mathsf{DS},\mathcal{A}}(1^\kappa)$ *is an experiment described in Figure 2. For* GOAL $\in$ {EUF, sEUF}*, we say that* DS *is* GOAL-CMA*-secure if* $\mathsf{Adv}^{\text{goal-cma}}_{\mathsf{DS},\mathcal{A}}(\kappa)$ *is negligible for any QPT adversary* $\mathcal{A}$*.*
For any $\mathcal{A}$*, we define its* euf-nma *advantage against* DS *as* $\mathsf{Adv}^{\text{euf-nma}}_{\mathsf{DS},\mathcal{A}}(\kappa) := \Pr[\mathsf{Expt}^{\text{euf-nma}}_{\mathsf{DS},\mathcal{A}}(1^\kappa) = 1]$*, where* $\mathsf{Expt}^{\text{euf-nma}}_{\mathsf{DS},\mathcal{A}}(1^\kappa)$ *is the game* $\mathsf{Expt}^{\text{euf-cma}}_{\mathsf{DS},\mathcal{A}}(1^\kappa)$ *without the signing oracle* Sign*. We say that* DS *is* EUF-NMA*-secure if* $\mathsf{Adv}^{\text{euf-nma}}_{\mathsf{DS},\mathcal{A}}(\kappa)$ *is negligible for any QPT adversary* $\mathcal{A}$*.*

$\underline{\mathsf{Expt}^{\text{s-ceo}}_{\mathsf{DS},\mathcal{A}}(1^\kappa)}$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$\mathcal{Q} := \emptyset;$
$(vk', \mu^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}}(vk)$
$d_1 := \mathsf{V}(vk', \mu^*, \sigma^*)$
$d_2 := \mathsf{boole}((\mu^*, \sigma^*) \in \mathcal{Q})$
$d_k := \mathsf{boole}(vk \neq vk')$
return $d_1 \wedge d_2 \wedge d_k$

$\underline{\mathsf{Expt}^{\text{s-deo}}_{\mathsf{DS},\mathcal{A}}(1^\kappa)}$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$\mathcal{Q} := \emptyset;$
$(vk', \mu^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}}(vk)$
$d_1 := \mathsf{V}(vk', \mu^*, \sigma^*)$
$d_2 := \mathsf{boole}(\exists \mu \neq \mu^* : (\mu, \sigma^*) \in \mathcal{Q})$
$d_k := \mathsf{boole}(vk \neq vk')$
return $d_1 \wedge d_2 \wedge d_k$

$\underline{\textsc{Sign}(\mu)}$
$\sigma \leftarrow \mathsf{Sign}(sk, \mu)$
$\mathcal{Q} := \mathcal{Q} \cup \{(\mu, \sigma)\}$
return σ

$\underline{\mathsf{Expt}^{\text{m-s-ueo}}_{\mathsf{DS},\mathcal{A}}(1^\kappa)}$
$(vk, vk', \mu, \mu', \sigma) \leftarrow \mathcal{A}(1^\kappa)$
$d_1 := \mathsf{Vrfy}(vk, \mu, \sigma)$
$d_2 := \mathsf{Vrfy}(vk', \mu', \sigma)$
$d_k := \mathsf{boole}(vk \neq vk')$
return $d_1 \wedge d_2 \wedge d_k$

$\underline{\mathsf{Expt}^{\text{mbs}}_{\mathsf{DS},\mathcal{A}}(1^\kappa)}$
$(vk, \mu, \mu', \sigma) \leftarrow \mathcal{A}(1^\kappa)$
$d_1 := \mathsf{Vrfy}(vk, \mu, \sigma)$
$d_2 := \mathsf{Vrfy}(vk, \mu', \sigma)$
$d_m := \mathsf{boole}(\mu \neq \mu')$
return $d_1 \wedge d_2 \wedge d_m$

$\underline{\mathsf{Expt}^{\text{wnr}}_{\mathsf{DS},\mathcal{A},\mathcal{D}}(1^\kappa)}$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$\mu \leftarrow \mathcal{MS}$
$\sigma \leftarrow \mathsf{Sign}(sk, \mu)$
$(\sigma', vk') \leftarrow \mathcal{A}(vk, \sigma)$
$d := \mathsf{Vrfy}(vk', \mu, \sigma')$
$d_k := \mathsf{boole}(vk \neq vk')$
return $d \wedge d_k$

Fig. 3. S-CEO, S-DEO, M-S-UEO, MBS, and wNR.

.

BUFF security notions: We review the definitions of exclusive ownership in Cremers et al. [CDF+21], strong conservative exclusive ownership (S-CEO), strong destructive exclusive ownership (S-DEO), and malicious-strong universal exclusive ownership (M-S-UEO). We also review the definition of message-bounding signatures (MBS) in [CDF+21].
As one of the advanced security notions, Cremers et al. [CDF+21] defined non-resignability (NR). Unfortunately, the original notion is unachievable, as Don, Fehr, Huang, and Struck showed [DFH23]. We here adopt a very weak version of NR, a weak non-resignability (wNR) defined by Aulbach et al. [ADM+24]. For stronger definitions, see [CDF+21, DFH23, DFH+24].

Definition 2.3 (S-CEO, S-DEO, M-S-UEO, MBS, **and** wNR). *Let* $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *be a digital signature scheme. For any* $\mathcal{A}$*, we define its* goal *advantage for* goal $\in$ {s-ceo, s-deo, m-s-ueo, mbs, wnr} *as*

$$\mathsf{Adv}^{\text{goal}}_{\mathsf{DS},\mathcal{A}}(\kappa) := \Pr[\mathsf{Expt}^{\text{goal}}_{\mathsf{DS},\mathcal{A}}(1^\kappa) = 1],$$

where $\mathsf{Expt}^{\text{goal}}_{\mathsf{DS},\mathcal{A}}(1^\kappa)$ *is experiments described in Figure 3 We say that* DS *is* GOAL*-secure for* GOAL $\in$ {S-CEO, S-DEO, M-S-UEO, MBS, wNR} *if* $\mathsf{Adv}^{\text{goal}}_{\mathsf{DS},\mathcal{A}}(\kappa)$ *is negligible for any QPT adversary* $\mathcal{A}$.

2.3 Multi-Pass Identification

We consider multi-pass ID schemes, where the number of passes is $(2n+1)$ for $n = 1, 2, 3$. We only treat public-coin ID schemes; that is, the verifier chooses i-th challenge uniformly at random from the challenge set C_i. The syntax follows:

Definition 2.4 (Multi-pass identification). *A* $(2n+1)$*-pass identification scheme* ID *consists of the following tuple of PPT algorithms* $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$*:*

- $\mathsf{Gen}(1^\kappa) \to (vk, sk)$*: a key-generation algorithm that takes* 1^κ *as input, where* κ *is the security parameter, and outputs a pair of keys* (vk, sk)*.* vk *and* sk *are public verification and secret keys, respectively.*
- $\mathsf{P}(sk, c_{i-1}, \mathsf{state}) \to (a_i, \mathsf{state})$*: a prover algorithm that, in the* i*-th round* $(i = 1, \dots, n+1)$*, takes signing key* sk*, the* $(i-1)$*-th challenge* c_{i-1}*, and state* state *as input, (we let* c_0 *and the initial state* state *be* $\emptyset$*) and outputs the* $(2i-1)$*-th message* a_i *and state* state*.*
- $\mathsf{V}(vk, a_1, c_1, \dots, a_n, c_n, a_{n+1}) \to \mathsf{true}/\mathsf{false}$*: a verification algorithm that takes verification key* vk *and the transcript* $a_1, c_1, \dots, a_n, c_n, a_{n+1}$ *as input and outputs its decision* true *or* false*.*

We assume that a verification key vk *defines the challenge spaces* $C_1, \dots, C_n$*. We also assume perfect correctness; a verifier always outputs* true *for an arbitrary honestly-generated key and transcript.*

We review the property of ID schemes. The first one is the min-entropy of the first message of an ID scheme:

$$
\begin{array}{ll}
\underline{\mathsf{Expt}^{q\text{-hvzk},0}_{\mathsf{ID},\mathcal{A}}(1^\kappa)} & \underline{\mathsf{Expt}^{q\text{-hvzk},1}_{\mathsf{ID},\mathcal{A}}(1^\kappa)} \\
(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa) & (vk, sk) \leftarrow \mathsf{Gen}(1^\kappa) \\
\textbf{for } i \in [q] \textbf{ do} & \textbf{for } i \in [q] \textbf{ do} \\
\quad \mathsf{trans}_i \leftarrow \langle \mathsf{P}(vk, sk), \mathsf{V}(vk) \rangle & \quad (c_1, \dots, c_n) \leftarrow C_1 \times \cdots \times C_n \\
b' \leftarrow \mathcal{A}(vk, (\mathsf{trans}_1, \dots, \mathsf{trans}_q)) & \quad \mathsf{trans}_i \leftarrow \mathsf{Sim}(vk, c_1, \dots, c_n) \\
\textbf{return } b' & b' \leftarrow \mathcal{A}(vk, (\mathsf{trans}_1, \dots, \mathsf{trans}_q)) \\
 & \textbf{return } b'
\end{array}
$$

Fig. 4. The experiments for computational multi-HVZK.

Definition 2.5 (Commitment entropy [KLS18, Def. 2.6], adapted). *We say that $(2n + 1)$-pass ID scheme* ID *has α-commitment entropy if for any (vk, sk) generated by* $\mathsf{Gen}(1^\kappa)$, $H_\infty(a_1 \mid (a_1, \mathsf{state}) \leftarrow \mathsf{P}(sk)) \geq \alpha$.

We next review honest-verifier zero knowledge for multiple transcripts.

Definition 2.6 (Special simulator). *Let (vk, sk) be a key pair generated by* $\mathsf{Gen}(1^\kappa)$. *A* special simulator *is an algorithm* Sim *that takes a public verification key vk and series of challenges $c_1, \dots, c_n$ and outputs a transcript $(a_1, c_1, \dots, a_n, c_n, a_{n+1})$.*

Definition 2.7 (Honest-verifier zero knowledge for multiple transcripts [GHHM21], adapted). *Let* ID *be an ID scheme with a PPT special simulator* Sim. *For a polynomial $q = q(\kappa)$ and an adversary $\mathcal{A}$, we define its q-HVZK advantage as follows:*

$$
\mathsf{Adv}^{q\text{-hvzk}}_{\mathsf{ID},\mathcal{A}}(\kappa) := \left| \Pr[\mathsf{Expt}^{q\text{-hvzk},0}_{\mathsf{ID},\mathcal{A}}(1^\kappa) = 1] - \Pr[\mathsf{Expt}^{q\text{-hvzk},1}_{\mathsf{ID},\mathcal{A}}(1^\kappa) = 1] \right|,
$$

where $\mathsf{Expt}^{q\text{-hvzk},b}_{\mathsf{ID},\mathcal{A}}(1^\kappa)$ is defined in Figure 4. We say that ID *is q-HVZK if $\mathsf{Adv}^{q\text{-hvzk}}_{\mathsf{ID},\mathcal{A}}(\kappa)$ is negligible for any QPT adversary $\mathcal{A}$.*

3 Unique Response and Non-Divegency

We say that three-pass ID scheme ID has *unique responses* if for all a_1 and c_1, there exists at most one a_2 satisfying $\mathsf{V}(vk, a_1, c_1, a_2) = \mathsf{true}$. Kiltz et al. [KLS18] relaxed this notion into a computational one:

Definition 3.1 (Computational unique response [KLS18, Def. 2.7], adapted). *We say that three-pass ID scheme* ID $= (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *has the* computational unique response (CUR) *property if for any QPT adversary $\mathcal{A}$, its advantage defined below is negligible in κ:*

$$
\mathsf{Adv}^{\mathrm{cur}}_{\mathsf{ID},\mathcal{A}}(\kappa) := \Pr\left[\begin{array}{c} (vk, sk) \leftarrow \mathsf{Gen}(1^\kappa), (a_1, c_1, a_2, a_2') \leftarrow \mathcal{A}(vk) : \\ a_2 \neq a_2' \wedge \mathsf{V}(vk, a_1, c_1, a_2) \wedge \mathsf{V}(vk, a_1, c_1, a_2') \end{array} \right].
$$

We can consider that the two transcripts (a_1, c_1, a_2) and (a_1, c_1, a_2') breaking the CUR property *branch at index* 2. Don et al. [DFM20] generalized this idea into $(2n + 1)$-pass ID as follows:

Definition 3.2 (Computational unique response [DFM20, Def. 22], adapted). *We say that $(2n + 1)$-pass ID scheme* ID $= (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *has the* computational unique response (CUR) *property if for any QPT adversary $\mathcal{A}$, its advantage defined below is negligible in κ:*

$$
\mathsf{Adv}^{\mathrm{cur}}_{\mathsf{ID},\mathcal{A}}(\kappa) := \Pr\left[\begin{array}{c} (vk, \mathsf{trans}, \mathsf{trans}') \leftarrow \mathcal{A}(1^\kappa) : \\ \mathsf{BranchCheck}_{\mathsf{DFM}}(\mathsf{trans}, \mathsf{trans}') \wedge \mathsf{V}(vk, \mathsf{trans}) \wedge \mathsf{V}(vk, \mathsf{trans}') \end{array} \right],
$$

where $\mathsf{BranchCheck}_{\mathsf{DFM}}(\mathsf{trans}, \mathsf{trans}')$ is defined as follows:

1. *Parse* $\mathsf{trans} = (a_1, c_1, \dots, a_n, c_n, a_{n+1})$ *and* $\mathsf{trans}' = (a_1', c_1', \dots, a_n', c_n', a_{n+1}')$.
2. *If there exists $k \in [2, n + 1]$ such that $(a_j, c_j) = (a_j', c_j')$ for all $j < k$ but $a_k \neq a_k'$, then return* true;
3. *Otherwise, return* false.

Unfortunately, this definition is too strong for the ID schemes in the wild, as discussed in subsection 3.1. To remedy the situation, we define our branch-checking algorithm as follows:

Definition 3.3 (Branch-checking algorithm). *A branch-checking algorithm* $\mathsf{BranchCheck}(\mathsf{trans}, \mathsf{trans}')$ *is defined as follows:*

1. *Parse* $\mathsf{trans} = (a_1, c_1, \ldots, a_n, c_n, a_{n+1})$ *and* $\mathsf{trans}' = (a'_1, c'_1, \ldots, a'_n, c'_n, a'_{n+1})$.
2. *If one of the following conditions is satisfied, then return* true*:*
   (a) *If there exists* $k \in [2, n]$ *such that* $(a_j, c_j) = (a'_j, c'_j)$ *for all* $j < k$ *but* $a_k \neq a'_k$ *and* $c_l \neq c'_l$ *for all* $l \in [k, n]$, *then return* true*;*
   (b) *If* $(a_j, c_j) = (a'_j, c'_j)$ *for all* $j \leq n$ *and* $a_{n+1} \neq a'_{n+1}$, *then return* true*; or*
   (c) *If* $a_j = a'_j$ *for all* $j \in [n+1]$ *and there exists* $k \in [2, n]$ *such that* $c_j = c'_j$ *for all* $j < k$ *but* $c_l \neq c_l$ *for all* $l \in [k, n]$, *then return* true.
3. *Otherwise, return* false.

Remark 3.1. The first condition (a) captures the case that the branch occurs at index $k < n$. Notice that we require $a_k \neq a'_k$ *and* $c_l \neq c'_l$ for all $l \in [k, n]$ instead of requiring just $a_k \neq a'_k$. This stronger requirement is covered by considering the collision resistance property of the random oracle.
The second condition (b) captures the case that the branch occurs at index $k = n + 1$.
The third condition (c) is introduced to treat the cases where the hash value h_1 *does not* contain the information of μ. This condition can be eliminated if we assume h_1 contains the information of μ.

By using this branch-checking algorithm, we can extend the CUR property for $(2n+1)$-pass ID as follows:

Definition 3.4 (Computational unique response for $(2n+1)$-pass ID). *We say that $(2n+1)$-pass identification scheme* ID *has the* computational unique response (CUR) *property if for any QPT adversary $\mathcal{A}$, its advantage defined below is negligible in κ:*

$$\mathsf{Adv}^{\mathrm{cur}}_{\mathrm{ID},\mathcal{A}}(\kappa) := \Pr\left[\begin{array}{c}(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa), (\mathsf{trans}, \mathsf{trans}') \leftarrow \mathcal{A}(vk) : \\ \mathsf{BranchCheck}(\mathsf{trans}, \mathsf{trans}') \wedge \mathsf{V}(vk, \mathsf{trans}) \wedge \mathsf{V}(vk, \mathsf{trans}')\end{array}\right],$$

where BranchCheck *is defined in Definition 3.3.*

Unfortunately, it is still hard to show that the underlying 5/7-pass ID schemes for the MPCitH signature satisfy this CUR property. To remedy this situation, we further weaken the CUR property: The adversary in CUR can choose two transcripts trans and trans′ by itself. We observe that one transcript should be generated by *the HVZK simulator*, and the adversary outputs a new branch diverged from the transcript in the security proof. We define the new property, *non-divergency*, as follows:

Definition 3.5 (Non-divergency for $(2n+1)$-pass ID). *We say that $(2n+1)$-pass ID scheme* ID *is* q-non-divergent with respect to Sim *if for any QPT adversary $\mathcal{A}$, its advantage defined below is negligible in κ:*

$$\mathsf{Adv}^{q\text{-nd}}_{\mathrm{ID},\mathcal{A}}(\kappa) := \Pr\left[\begin{array}{c}(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa), \\ \textit{for } i \in [q]\ \mathsf{trans}_i \leftarrow \mathsf{Sim}(vk, U(C_1), \ldots, U(C_n)), \\ (i, \mathsf{trans}') \leftarrow \mathcal{A}(vk, \mathsf{trans}_1, \ldots, \mathsf{trans}_q) : \\ \mathsf{BranchCheck}(\mathsf{trans}_i, \mathsf{trans}') \wedge \mathsf{V}(vk, \mathsf{trans}_i) \wedge \mathsf{V}(vk, \mathsf{trans}')\end{array}\right],$$

where BranchCheck *is defined as in Definition 3.3.*

For easiness, we define the stronger version of non-divergency by relaxing the conditions. We define another branch-checking algorithm and strong non-divergency as follows:

Definition 3.6 (Strong Branch-checking algorithm). *Fix a $(2n+1)$-pass ID scheme* ID. *A branch-checking algorithm* $\mathsf{BranchCheck}'(\mathsf{trans}, \mathsf{trans}')$ *is defined as follows:*
1. *Parse* $\mathsf{trans} = (a_1, c_1, \ldots, a_n, c_n, a_{n+1})$ *and* $\mathsf{trans}' = (a'_1, c'_1, \ldots, a'_n, c'_n, a'_{n+1})$.
2. *If one of the following conditions is satisfied, then return* true*:*
   (a) *If* $(a_1, c_1) = (a'_1, c'_1)$ *and* $c_n \neq c'_n$, *then return* true*; or*
   (b) *If* $(a_j, c_j) = (a'_j, c'_j)$ *for all* $j \leq n$ *and* $a_{n+1} \neq a'_{n+1}$, *then return* true*;*
3. *Otherwise, return* false.

Remark 3.2. We remark that the new condition (a) covers the conditions (a) and (c) in BranchCheck.

Definition 3.7 (Strong non-divergency for $(2n+1)$-pass ID). *We say that $(2n+1)$-pass ID scheme* ID *is* strongly q-non-divergent with respect to Sim *if for any QPT adversary $\mathcal{A}$, its advantage* $\mathsf{Adv}^{q\text{-snd}}_{\mathrm{ID},\mathcal{A}}(\kappa)$ *defined below is negligible in κ, where* $\mathsf{Adv}^{q\text{-snd}}_{\mathrm{ID},\mathcal{A}}(\kappa)$ *is the advantage* $\mathsf{Adv}^{q\text{-nd}}_{\mathrm{ID},\mathcal{A}}(\kappa)$ *with* BranchCheck′.

Since the conditions are relaxed, we have the following lemma.

Lemma 3.1. *If* ID *is strongly q-non-divergent with respect to* Sim, *then* ID *is q-non-divergent with respect to* Sim.

3.1 Counterexample of CUR of SSH11

We briefly recall the 5-pass SSH11 protocol in [SSH11]. Let $F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$ be an m quadratic functions in $\mathbb{F}_q[x_1, \ldots, x_n]$, where $f_l(\boldsymbol{x}) = \sum_{i,j} a_{ij}^{(l)} x_i x_j + \sum_i b_i^{(l)} x_i$ with $a_{ij}^{(l)}, b_i^{(l)} \in \mathbb{F}_q$ for $l \in [m]$. A prover and a verifier have $(F, \boldsymbol{v})$ and the prover has a witness $\boldsymbol{s} \in \mathbb{F}_q^n$ satisfying $F(\boldsymbol{s}) = \boldsymbol{v}$. Let Com be a commitment scheme. In the protocol, $G(\boldsymbol{x}, \boldsymbol{y})$ denotes F's polar form, which is defined as $G(\boldsymbol{x}, \boldsymbol{y}) := F(\boldsymbol{x} + \boldsymbol{y}) - F(\boldsymbol{x}) - F(\boldsymbol{y})$. The protocol is defined as follows:

1. The prover chooses $\boldsymbol{r}_0, \boldsymbol{t}_0 \leftarrow \mathbb{F}_q^n$ and $\boldsymbol{e}_0 \leftarrow \mathbb{F}_q^m$ uniformly at random. It computes $\boldsymbol{r}_1 := \boldsymbol{s} - \boldsymbol{r}_0$. It sends $\mathsf{com}_0 := \mathsf{Com}(\boldsymbol{r}_0, \boldsymbol{t}_0, \boldsymbol{e}_0)$ and $\mathsf{com}_1 := \mathsf{Com}(\boldsymbol{r}_1, G(\boldsymbol{t}_0, \boldsymbol{r}_1) + \boldsymbol{e}_0)$.
2. The verifier picks $\alpha \leftarrow \mathbb{F}_q$ and sends it.
3. The prover sends $\boldsymbol{t}_1 := \alpha \boldsymbol{r}_0 - \boldsymbol{t}_0$ and $\boldsymbol{e}_1 := \alpha F(\boldsymbol{r}_0) - \boldsymbol{e}_0$.
4. The verifer picks $b \leftarrow \{0, 1\}$ and sends it.
5. The prover sends $\boldsymbol{r}_b$.
6. The verifier outputs the result of the check defined as follows:
   - If $b = 0$, then check if $\mathsf{com}_0 = \mathsf{Com}\big(\boldsymbol{r}_0, \alpha \boldsymbol{r}_0 - \boldsymbol{t}_1, \alpha F(\boldsymbol{r}_0) - \boldsymbol{e}_1\big)$.
   - If $b = 1$, then check if $\mathsf{com}_1 = \mathsf{Com}\big(\boldsymbol{r}_1, \alpha(\boldsymbol{v} - F(\boldsymbol{r}_1)) - G(\boldsymbol{t}_1, \boldsymbol{r}_1) - \boldsymbol{e}_1\big)$.

Don et al. concluded that this protocol, denoted by Π_{SSH} in their paper, satisfies their CUR definition (Definition 3.2) as follows [DFM20]:

> In Π_{SSH}, the honest prover's first message consists of two commitments, and the second and final messages contain functions of the strings committed to in the first message. This structure, together with the computational binding property (implied by the collapse binding property) of the commitments, immediately implies that Π_{SSH} has computationally unique response.

Unfortunately, this argument is incorrect, and we can falsify it as follows: Let us construct the following adversary who knows $\boldsymbol{s}$ corresponding to F and $\boldsymbol{v}$, because Definition 3.2 allows an adversary to produce vk:

1. Set $b = 1$ and pick $\alpha \leftarrow \mathbb{F}_q$.
2. Pick $\boldsymbol{r}_0, \boldsymbol{t}_0 \leftarrow \mathbb{F}_q^n$ and $\boldsymbol{e}_0 \leftarrow \mathbb{F}_q^m$. Compute $\boldsymbol{r}_1 := \boldsymbol{s} - \boldsymbol{r}_0$. Compute $\mathsf{com}_0 := \mathsf{Com}(\boldsymbol{r}_0, \boldsymbol{t}_0, \boldsymbol{e}_0)$ and $\mathsf{com}_1 := \mathsf{Com}(\boldsymbol{r}_1, G(\boldsymbol{t}_0, \boldsymbol{r}_1) + \boldsymbol{e}_0)$.
3. Compute $\boldsymbol{t}_1 := \alpha \boldsymbol{r}_0 - \boldsymbol{t}_0$ and $\boldsymbol{e}_1 := \alpha F(\boldsymbol{r}_0) - \boldsymbol{e}_0$. Compute $\boldsymbol{t}_1'$ and $\boldsymbol{e}_1'$ such that $G(\boldsymbol{t}_1, \boldsymbol{r}_1) + \boldsymbol{e}_1 = G(\boldsymbol{t}_1', \boldsymbol{r}_1) + \boldsymbol{e}_1'$ by choosing $\boldsymbol{t}_1' \neq \boldsymbol{t}_1$ and setting $\boldsymbol{e}_1' := G(\boldsymbol{t}_1, \boldsymbol{r}_1) + \boldsymbol{e}_1 - G(\boldsymbol{t}_1', \boldsymbol{r}_1)$.
4. Output

$$\begin{aligned} \mathsf{trans}_1 &:= ((\mathsf{com}_0, \mathsf{com}_1), \alpha, (\boldsymbol{t}_1, \boldsymbol{e}_1), 1, \boldsymbol{r}_1), \\ \mathsf{trans}_2 &:= ((\mathsf{com}_0, \mathsf{com}_1), \alpha, (\boldsymbol{t}_1', \boldsymbol{e}_1'), 1, \boldsymbol{r}_1). \end{aligned}$$

The two transcripts are valid since the verifier checks if

$$\begin{aligned} \mathsf{com}_1 &= \mathsf{Com}\big(\boldsymbol{r}_1, \alpha(\boldsymbol{v} - F(\boldsymbol{r}_1)) - G(\boldsymbol{t}_1, \boldsymbol{r}_1) - \boldsymbol{e}_1\big) \\ &= \mathsf{Com}\big(\boldsymbol{r}_1, \alpha(\boldsymbol{v} - F(\boldsymbol{r}_1)) - G(\boldsymbol{t}_1', \boldsymbol{r}_1) - \boldsymbol{e}_1'\big). \end{aligned}$$

Since $a_2 \neq a_2'$, they satisfy the criteria of $\mathsf{Check}_{\mathsf{DFM}}$ in Definition 3.2 and this adversary breaks the CUR property. Even if we modify the definition that vk is given to the adversary, we can construct the CUR adversary since the adversary can choose $\alpha \in \mathbb{F}_q$ and $b = 1$ by itself and run the HVZK simulator for $b = 1$ in [SSH11] who chooses $\boldsymbol{r}_1$ uniformly at random instead of setting $\boldsymbol{r}_1 := \boldsymbol{s} - \boldsymbol{r}_0$.

We can modify the condition checking algorithm $\mathsf{Check}_{\mathsf{DFM}}$ with $\mathsf{BranchCheck}$. If so, an adversary needs to output transcripts such that $a_2 \neq a_2' \wedge c_2 \neq c_2'$ and the above attack does not work. However, if we consider the τ-parallel version for $\tau \geq 2$ as the real use in MQDSS, the above attack revives since one can differentiate the challenges $c_2 = (b_1, \ldots, b_\tau)$ and $c_2' = (b_1', \ldots, b_\tau')$ by setting $b_1 = b_1' = \cdots = b_{\tau-1} = b_{\tau-1}' = 1$ and $b_\tau = 0 \neq b_\tau' = 1$ and the above attack revives. If the adversary is given the verification key vk, then one can use a more malicious simulator: For the index τ, the simulator first chooses $\alpha \leftarrow \mathbb{F}_q$, $\boldsymbol{r}_0, \boldsymbol{r}_1, \boldsymbol{t}_1 \leftarrow \mathbb{F}_q^n$, and $\boldsymbol{e}_1 \leftarrow \mathbb{F}_q^m$, computes $\mathsf{com}_0 := \mathsf{Com}(\boldsymbol{r}_0, \alpha \boldsymbol{r}_0 - \boldsymbol{t}_1, \alpha F(\boldsymbol{r}_0) - \boldsymbol{e}_1)$ and $\mathsf{com}_1 := \mathsf{Com}\big(\boldsymbol{r}_1, \alpha(v - F(\boldsymbol{r}_1)) - G(\boldsymbol{t}_1, \boldsymbol{r}_1) - \boldsymbol{e}\big)$, and outputs $a_1 = (\mathsf{com}_0, \mathsf{com}_1)$, $c_1 = \alpha$, $a_2 = \boldsymbol{t}_1$, $c_2 = 0$, $c_2' = 1$, and $a_3 = \boldsymbol{r}_0$, and $a_3' = \boldsymbol{r}_1$.

We salvage MQDSS's sEUF-CMA security in Section C by showing that the SSH11 protocol is strongly non-divergent with respect to a new HVZK simulator. See Section C for the details.

4 Signature from Multi-Pass Identification

We review a signature scheme constructed from a $(2n+1)$-pass identification scheme $\mathsf{ID} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ via the FS transform [EDV+12, DGV+16, CHR+16]. Let $\mathsf{H} \colon \{0,1\}^* \to \mathcal{H}$ and $\gamma_i \colon \mathcal{H} \to C_i$ for $i \in [n]$ be hash functions modeled as random oracles. The FS transform converts ID into a signature scheme $\mathsf{DS} = \mathsf{FS}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$

$\underline{\mathsf{Sign}_{\mathrm{cmt}}(sk, \mu)}$	$\underline{\mathsf{Vrfy}_{\mathrm{cmt}}(vk, \mu, \sigma)}$
$h_0 := \emptyset$; $c_0 := \emptyset$; $\mathsf{state} := \emptyset$	Parse $\sigma = (a_1, \ldots, a_n, a_{n+1})$
for $i = 1, \ldots, n$ **do**	$h_0 := \emptyset$
$\quad (a_i, \mathsf{state}) \leftarrow \mathsf{P}(sk, c_{i-1}, \mathsf{state})$	**for** $i = 1, \ldots, n$ **do**
$\quad h_i := \mathsf{H}(\mathsf{aux}_i, h_{i-1}, a_i)$	$\quad h_i := \mathsf{H}(\mathsf{aux}_i, h_{i-1}, a_i)$
$\quad c_i := \gamma_i(h_i)$	$\quad c_i := \gamma_i(h_i)$
$a_{n+1} \leftarrow \mathsf{P}(sk, c_n, \mathsf{state})$	$d := \mathsf{V}(vk, a_1, c_1, \ldots, a_n, c_n, a_{n+1})$
return $\sigma := (a_1, \ldots, a_n, a_{n+1})$	**return** d

Fig. 5. Scheme $\mathsf{FS}_{\mathrm{cmt}}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}] = (\mathsf{Gen}, \mathsf{Sign}_{\mathrm{cmt}}, \mathsf{Vrfy}_{\mathrm{cmt}})$, where $\mathsf{ID} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$, $\mathsf{H} : \{0,1\}^* \to \mathcal{H}$ is modeled as the random oracle, and $\gamma_i : \mathcal{H} \to C_i$ for $i \in [n]$ is also modeled as the random oracle. For ease of notation, we let $\mathsf{aux}_i = \mathsf{aux}(i, vk, \mu)$.

by computing i-th challenge c_i from a message μ, previous challenge c_{i-1}, and i-th message a_i and setting $\sigma = (a_1, \ldots, a_n, a_{n+1})$. In the original formulations [EDV+12, DGV+16, CHR+16], they defined

$$c_i := \begin{cases} \mathsf{H}(1, vk, \mu, a_1) & \text{if } i = 1 \\ \mathsf{H}(i, c_{i-1}, a_i) & \text{if } i = 2, \ldots, n. \end{cases}$$

Since almost all MPCitH signature schemes modify the input of the hash functions and use hash values as seeds of challenges, we define the computation of the challenges as follows:

$$h_i := \begin{cases} \mathsf{H}(\mathsf{aux}_1, a_1) & \text{if } i = 1 \\ \mathsf{H}(\mathsf{aux}_i, h_{i-1}, a_i) & \text{if } i = 2, \ldots, n \end{cases} \text{ and } c_i := \gamma_i(h_i),$$

where $\mathsf{aux}_i = \mathsf{aux}(i, vk, \mu)$ is a value computed from μ, vk, and i (and more, e.g., salt). The formal definitions are depicted in Figure 5.

Collision resistance of aux*:* Later, we want to discuss the minimum index $\lambda \in [n]$ satisfying that if $\mu \neq \mu'$ then $\mathsf{aux}(i, vk, \mu) \neq \mathsf{aux}(i, vk, \mu')$ holds (perfectly or computationally). We also use a similar property with respect to vk to discuss the M-S-UEO property. We formalize such property as collision resistance property of aux as follows:

Definition 4.1 (Collision resistance property of aux). *We say that* aux *is collision-resistant with respect to message on index* $\lambda \in [n]$ *if for any QPT adversary* $\mathcal{A}$*, its advantage*

$$\mathsf{Adv}^{\mathrm{cr,msg}}_{\mathsf{aux},\mathcal{A}}(\kappa) := \Pr\left[\begin{array}{c} (vk, vk', \mu, \mu') \leftarrow \mathcal{A}(1^\kappa) : \\ \mu \neq \mu' \wedge \exists l \in [\lambda, n], \forall i \in [1, l], \mathsf{aux}(i, vk, \mu) = \mathsf{aux}(i, vk', \mu') \end{array}\right]$$

is negligible in the security parameter κ*.*
We say that aux *is collision-resistant with respect to verification key on index* $\lambda \in [n]$ *if for any QPT adversary* $\mathcal{A}$*, its advantage*

$$\mathsf{Adv}^{\mathrm{cr,vk}}_{\mathsf{aux},\mathcal{A}}(\kappa) := \Pr\left[\begin{array}{c} (vk, vk', \mu, \mu') \leftarrow \mathcal{A}(1^\kappa) : \\ vk \neq vk' \wedge \exists l \in [\lambda, n], \forall i \in [1, l], \mathsf{aux}(i, vk, \mu) = \mathsf{aux}(i, vk', \mu') \end{array}\right]$$

is negligible in the security parameter κ*.*

4.1 sEUF-CMA Security for $\mathsf{FS}_{\mathrm{cmt}}$

We show the sEUF-CMA security of the signature scheme obtained by applying $\mathsf{FS}_{\mathrm{cmt}}$ to $(2n+1)$-pass ID as follows.

Theorem 4.1 (EUF-NMA $\Rightarrow$ sEUF-CMA for $\mathsf{FS}_{\mathrm{cmt}}$ on $(2n+1)$-pass ID). *Let* ID *be a* $(2n+1)$*-pass ID scheme that has* α*-commitment entropy. Let* $\mathsf{H} : \{0,1\}^* \to \mathcal{H}$ *and* $\gamma_i : \mathcal{H} \to C_i$ *for* $i \in [n]$ *be random oracles. Suppose that* aux *is collision-resistant with respect to message on index* λ*. Let* $\mathsf{DS} := \mathsf{FS}_{\mathrm{cmt}}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$*. For any quantum adversary* $\mathcal{A}$ *against the* sEUF-CMA *security of* DS *issuing at most* q_S *classical queries to the signing oracle and at most* q_H *and* q_i *quantum queries to the random oracles* H *and* γ_i*, there exist an adversary* $\mathcal{A}_{\mathrm{nma}}$ *against the* EUF-NMA *security of* DS*, an adversary* $\mathcal{A}_{\mathrm{hvzk}}$ *against the* q_S*-HVZK property of* ID*, an adversary* $\mathcal{A}_{\mathrm{cr}}$ *against* aux*'s collision-resistance property with respect to message on index* λ*, and an adversary* $\mathcal{A}_{\mathrm{nd}}$ *against the* q_S*-non-divergency of* ID*, such that*

$$\begin{aligned}
&\mathsf{Adv}^{\text{seuf-cma}}_{\mathsf{DS},\mathcal{A}}(1^\kappa)\\
&\le \mathsf{Adv}^{\text{euf-nma}}_{\mathsf{DS},\mathcal{A}_{\mathrm{nma}}}(1^\kappa) + \mathsf{Adv}^{q_S\text{-hvzk}}_{\mathsf{ID},\mathcal{A}_{\mathrm{hvzk}}}(1^\kappa) + \mathsf{Adv}^{\text{cr,msg}}_{\mathsf{aux},\mathcal{A}_{\mathrm{cr}}}(1^\kappa) + \mathsf{Adv}^{q_S\text{-nd}}_{\mathsf{ID},\mathcal{A}_{\mathrm{nd}}}(1^\kappa)\\
&\quad + 632 \cdot (q_H + nq_S + n + 1)^3 \cdot |\mathcal{H}|^{-1} + 632 \sum_{i\in[n]} (q_i + q_S + 2)^3 \cdot |\mathcal{C}_i|^{-1}\\
&\quad + \sum_{i\in[n]} \frac{3q_S}{2} \sqrt{(q_H + q_S + 2)\cdot 2^{-\alpha_i}} + \sum_{i\in[n]} \frac{3q_S}{2} \sqrt{(q_i + q_S + 2)\cdot |\mathcal{H}|^{-1}},
\end{aligned}$$

where $\alpha_1 = \alpha$ *and* $\alpha_2 = \cdots = \alpha_n = \lg(\#\mathcal{H})$*. The running times of* $\mathcal{A}_{\mathrm{nma}}$*,* $\mathcal{A}_{\mathrm{hvzk}}$*,* $\mathcal{A}_{\mathrm{cr}}$*, and* $\mathcal{A}_{\mathrm{nd}}$ *are approximately that of* $\mathcal{A}$*.*

We prove this theorem by modifying the proof of [GHHM21, Thm.3]. We define eight games $\mathsf{G}_0, \dots, \mathsf{G}_7$ in Figure 6. In G_1, we introduce an algorithm to check a collision of H, denoted by $\mathsf{CollCheck}$. In G_2, we additionally check a collision of γ_i's. Those two changes prohibit the adversary from converging a forgery to the signatures signed by the signing oracle. In G_3, the signing oracle chooses the hash values to produce challenges uniformly at random and then reprograms the random oracle H as in [GHHM21, Thm.3]. In G_4, the signing oracle chooses challenges uniformly at random and then reprograms the random oracles $\gamma_1, \dots, \gamma_n$. We next modify the signing oracle to use the simulator instead of the prover algorithms in G_5. In G_6, we introduce $\mathsf{AuxCheck}$ to check if the adversary submits a forgery diverged from the signature signed by the signing oracle by using the collision of aux. If there is a difference, then we can break the collision-resistance property of aux. In G_7, we again introduce $\mathsf{ForkCheck}$ to check if the adversary submits a forgery diverged from the signature signed by the signing oracle. If there is a difference, then we can break the non-divergency of the underlying ID. We will discuss that the forgery does not involve the reprogrammed points, and we can reduce it to the EUF-NMA security of the signature scheme. In what follows, we define W_i as the event that the adversary wins in G_i.

Game G_0*:* This is the original sEUF-CMA game. We have

$$\Pr[W_0] = \mathsf{Adv}^{\text{seuf-cma}}_{\mathsf{DS},\mathcal{A}}(1^\kappa).$$

Game G_1*:* In this game, the challenger manages the list $\mathcal{L}$ that contains the hash values and challenges that the signing oracle SIGN produced. Receiving a message μ^* and a signature $(a_1^*, \dots, a_{n+1}^*)$, the challenger runs $\mathsf{CollCheck}$ for G_1 (Figure 6) and, if it returns true, then the adversary loses.
If there is a difference between G_0 and G_1, $\mathsf{CollCheck}$ returns true. According to the definition of $\mathsf{CollCheck}$ for G_1, this means that there exists an entry $(\mathsf{aux}_i, h_{i-1}, c_{i-1}, a_i, h_i^*, c_i^*)$ in $\mathcal{L}$ such that $(h_{i-1}, a_i) \neq (h_{i-1}^*, a_i^*)$. This implies a collision for H since we have $(\mathsf{aux}_i, h_{i-1}, a_i) \neq (\mathsf{aux}_i^*, h_{i-1}^*, a_i^*)$ but $\mathsf{H}(\mathsf{aux}_i, h_{i-1}, a_i) = h_i^* = \mathsf{H}(\mathsf{aux}_i^*, h_{i-1}^*, a_i^*)$. Since the number of queries to H is at most $q_H + nq_S + n$, we have the following lemma by using Lemma 2.1.

Lemma 4.1. *We have* $|\Pr[W_0] - \Pr[W_1]| \le 632(q_H + nq_S + n + 1)^3 \cdot |\mathcal{H}|^{-1}$.

Game G_2*:* In this game, receiving the forgery μ^* and $(a_1^*, \dots, a_{n+1}^*)$, the challenger runs $\mathsf{CollCheck}$ for G_2–G_6 and, if it returns true, then the adversary loses.
Notice that the difference between those two $\mathsf{CollCheck}$s, $\exists(\mathsf{aux}_i, h_{i-1}, a_i, \underline{h_i^*}, c_i^*) \in \mathcal{L}$ in G_1 and $\exists(\mathsf{aux}_i, h_{i-1}, a_i, \underline{h_i}, c_i^*) \in \mathcal{L}$ in G_2. Thus, if there is a difference between G_1 and G_2, then there exists $(\mathsf{aux}_i, h_{i-1}, a_i, h_i, c_i^*)$ in $\mathcal{L}$ such that $(h_{i-1}, a_i) \neq (h_{i-1}^*, a_i^*)$ *and* $h_i \neq h_i^*$. This implies a collision for γ_i since we have $h_i \neq h_i^*$ but $\gamma_i(h_i) = c_i^* = \gamma_i(h_i^*)$. Since the number of queries to γ_i is $q_i + q_S + 1$, applying Lemma 2.1 to $\gamma_1, \dots, \gamma_n$, we have the following lemma:

Lemma 4.2. *We have* $|\Pr[W_1] - \Pr[W_2]| \le \sum_{i\in[n]} 632(q_i + q_S + 2)^3 \cdot |\mathcal{C}_i|^{-1}$.

Game G_3*:* In this game, we reprogram the random oracle H by choosing $h_i \leftarrow \mathcal{H}$ as in L.6–7 of SIGN. Applying Lemma 2.2, we have the following lemma:

Lemma 4.3. *We have* $|\Pr[W_2] - \Pr[W_3]| \le \sum_{i\in[n]} \frac{3q_S}{2} \sqrt{(q_H + q_S + 1)/2^{\alpha_i}}$.

The proof is the same as that of [GHHM21, Thm.3], and we omit it.

Game G_4*:* In this game, we reprogram the random oracle γ_i by choosing $c_i \leftarrow C_i$ as in L.9–10 of SIGN. Applying Lemma 2.2, we have the following lemma:

Lemma 4.4. *We have* $|\Pr[W_3] - \Pr[W_4]| \le \sum_{i\in[n]} \frac{3q_S}{2} \sqrt{(q_i + q_S + 1)/|\mathcal{H}|}$.

The proof is the same as that of [GHHM21, Thm.3], and we omit it.

$\underline{\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3, \mathsf{G}_4, \mathsf{G}_5}$

$(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$

$\mathcal{Q} := \emptyset$

$\mathcal{L} := \emptyset$ $\quad$ //G_1-

$(\mu^*, (a_1^*, \ldots, a_n^*, a_{n+1}^*)) \leftarrow \mathcal{A}^{\textsc{Sign},|\mathsf{H}\rangle,|\gamma\rangle}(vk)$

if $(\mu^*, (a_1^*, \ldots, a_n^*, a_{n+1}^*)) \in \mathcal{Q}$ **then**

$\quad$ **return** false

$h_0^* := \emptyset$

for $i = 1, \ldots, n$ **do**

$\quad h_i^* := \mathsf{H}(\mathsf{aux}_i^*, h_{i-1}^*, a_i^*)$

$\quad c_i^* := \gamma_i(h_i^*)$

$\quad$ **if** $\mathsf{CollCheck}_{\mathcal{L}}(\mathsf{aux}_i^*, h_{i-1}^*, a_i^*, h_i^*, c_i^*)$ **then** $\quad$ //G_1-

$\quad\quad$ **return** false $\quad$ //G_1-

return $\mathsf{V}(vk, a_1^*, c_1^*, \ldots, a_n^*, c_n^*, a_{n+1}^*)$

$\underline{\mathsf{CollCheck}_{\mathcal{L}}(\mathsf{aux}_i^*, h_{i-1}^*, a_i^*, h_i^*, c_i^*) \text{ for } \mathsf{G}_1}$

if $\exists(\mathsf{aux}_i, h_{i-1}, a_i, h_i^*, c_i^*) \in \mathcal{L} \colon (h_{i-1}, a_i) \neq (h_{i-1}^*, a_i^*)$ **then**

$\quad$ **return** true

else

$\quad$ **return** false

$\underline{\mathsf{CollCheck}_{\mathcal{L}}(\mathsf{aux}_i^*, h_{i-1}^*, a_i^*, h_i^*, c_i^*) \text{ for } \mathsf{G}_2\text{–}\mathsf{G}_7}$

if $\exists(\mathsf{aux}_i, h_{i-1}, a_i, h_i, c_i^*) \in \mathcal{L} \colon (h_{i-1}, a_i) \neq (h_{i-1}^*, a_i^*)$ **then**

$\quad$ **return** true

else

$\quad$ **return** false

$\underline{\mathsf{G}_6 \text{ and } \mathsf{G}_7}$

$(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$

$\mathcal{Q} := \emptyset; \mathcal{L} := \emptyset$

$(\mu^*, (a_1^*, \ldots, a_{n+1}^*)) \leftarrow \mathcal{A}^{\textsc{Sign},|\mathsf{H}\rangle,|\gamma\rangle}(vk)$

if $(\mu^*, (a_1^*, \ldots, a_{n+1}^*)) \in \mathcal{Q}$ **then**

$\quad$ **return** false

$h_0^* := \emptyset$

for $i = 1, \ldots, n$ **do**

$\quad h_i^* := \mathsf{H}(\mathsf{aux}_i^*, h_{i-1}^*, a_i^*)$

$\quad c_i^* := \gamma_i(h_i^*)$

$\quad$ **if** $\mathsf{CollCheck}_{\mathcal{L}}(\mathsf{aux}_i^*, h_{i-1}^*, a_i^*, h_i^*, c_i^*)$ **then**

$\quad\quad$ **return** false

if $\mathsf{AuxCheck}_{\mathcal{Q}}(\mu^*)$ **then**

$\quad$ **return** false

if $\mathsf{ForkCheck}_{\mathcal{Q}}(\mu^*, a_1^*, \ldots, a_{n+1}^*)$ **then** $\quad$ //G_7

$\quad$ **return** false $\quad$ //G_7

return $\mathsf{V}(vk, a_1^*, c_1^*, \ldots, a_n^*, c_n^*, a_{n+1}^*)$

$\underline{\textsc{Sign}(\mu) \text{ for } \mathsf{G}_0\text{-}\mathsf{G}_4}$

$h_0 := \emptyset; c_0 := \emptyset; \mathsf{state} := \emptyset$

for $i = 1, \ldots, n$ **do**

$\quad (a_i, \mathsf{state}) \leftarrow \mathsf{P}(sk, c_{i-1}, \mathsf{state})$

$\quad h_i := \mathsf{H}(\mathsf{aux}_i, h_{i-1}, a_i)$ $\quad$ //G_0–G_2

$\quad h_i \leftarrow \mathcal{H}$ $\quad$ //G_3–G_4

$\quad \mathsf{H} := \mathsf{H}[(\mathsf{aux}_i, h_{i-1}, a_i) \mapsto h_i]$ $\quad$ //G_3–G_4

$\quad c_i := \gamma_i(h_i)$ $\quad$ //G_0–G_3

$\quad c_i \leftarrow \mathcal{C}_i$ $\quad$ //G_4

$\quad \gamma_i := \gamma_i[h_i \mapsto c_i]$ $\quad$ //G_4

$\quad \mathcal{L} := \mathcal{L} \cup \{(\mathsf{aux}_i, h_{i-1}, a_i, h_i, c_i)\}$ $\quad$ //G_1-

$a_{n+1} \leftarrow \mathsf{P}(sk, c_n, \mathsf{state})$

$\mathcal{Q} := \mathcal{Q} \cup \{(\mu, (a_1, \ldots, a_n, a_{n+1}))\}$

return $\sigma := (a_1, \ldots, a_n, a_{n+1})$

$\underline{\textsc{Sign}(\mu) \text{ for } \mathsf{G}_5 \text{ and } \mathsf{G}_6}$

$h_0 := \emptyset$

for $i = 1, \ldots, n$ **do**

$\quad h_i \leftarrow \mathcal{H}; c_i \leftarrow \mathcal{C}_i$

$(a_1, \ldots, a_n, a_{n+1}) \leftarrow \mathsf{Sim}(vk, c_1, \ldots, c_n)$

for $i = 1, \ldots, n$ **do**

$\quad \mathsf{H} := \mathsf{H}[(\mathsf{aux}_i, h_{i-1}, a_i) \mapsto h_i]$

$\quad \gamma_i := \gamma_i[h_i \mapsto c_i]$

$\quad \mathcal{L} := \mathcal{L} \cup \{(\mathsf{aux}_i, h_{i-1}, a_i, h_i, c_i)\}$

$\mathcal{Q} := \mathcal{Q} \cup \{(\mu, (a_1, \ldots, a_n, a_{n+1}))\}$

return $\sigma := (a_1, \ldots, a_n, a_{n+1})$

$\underline{\mathsf{AuxCheck}_{\mathcal{Q}}(\mu^*) \text{ for } \mathsf{G}_6, \mathsf{G}_7}$

if $\exists(\mu, *) \in \mathcal{Q} \colon \mu \neq \mu^* \wedge \mathsf{aux}_\lambda = \mathsf{aux}_\lambda^*$ **then**

$\quad$ **return** true

return false

$\underline{\mathsf{ForkCheck}_{\mathcal{Q}}(\mu^*, a_1^*, \ldots, a_n^*, a_{n+1}^*) \text{ for } \mathsf{G}_7}$

forall $(\mu, (a_1, a_2, \ldots, a_n, a_{n+1})) \in \mathcal{Q}$ **do**

$\quad$ **if** $\exists k \geq 2 \colon a_j = a_j^* \textit{ for } j < k \textit{ but } a_k \neq a_k^*$ **then**

$\quad\quad$ **return** true

$\quad$ **if** $\exists k \geq 2 \colon c_j = c_j^* \textit{ for } j < k \textit{ but } c_j \neq c_j^* \textit{ for } j \in [k, n]$ $\textit{and } a_i = a_i^* \textit{ for all } i \in [n+1]$ **then**

$\quad\quad$ **return** true

return false

Fig. 6. Games G_0–G_7 for sEUF-CMA security proof of $\mathsf{FS}_{\mathrm{cmt}}$.

Game G_5*:* We then replace P with Sim in the signing oracle as $\textsc{Sign}$ for G_5 and G_6. The HVZK property justifies this replacement.

Lemma 4.5. *There exists an adversary $\mathcal{A}_{\text{hvzk}}$ against the q_S-HVZK property of* ID *such that*

$$|\Pr[W_4] - \Pr[W_5]| \leq \mathsf{Adv}^{q_S\text{-hvzk}}_{\mathsf{ID},\mathcal{A}_{\text{hvzk}}}(1^\kappa).$$

The running time of $\mathcal{A}_{\text{hvzk}}$ is approximately that of $\mathcal{A}$.

The proof is the same as that of [GHHM21, Thm.3], and we omit it.

Game G_6*:* In this game, the challenger runs $\mathsf{AuxCheck}$ and, if the result is true, then the adversary loses.

Lemma 4.6. *There exists an adversary $\mathcal{A}_{\text{cr}}$ against* aux*'s collision-resistance property of with respect to message on index λ such that*

$$|\Pr[W_5] - \Pr[W_6]| \leq \mathsf{Adv}^{\text{cr,msg}}_{\mathsf{aux},\mathcal{A}_{\text{cr}}}(1^\kappa).$$

The running time of $\mathcal{A}_{\text{cr}}$ is approximately that of $\mathcal{A}$.

Proof. The difference between G_5 and G_6 occurs if the adversary submits a *valid* pair of a message and a siganture $(\mu^*, (a_1^*, \dots, a_n^*, a_{n+1}^*))$ such that $(\mu^*, (a_1^*, \dots, a_n^*, a_{n+1}^*)) \notin \mathcal{Q}$, the pair passes the collision checks by $\mathsf{CollCheck}$, *and* $\mathsf{AuxCheck}(\mu^*) = \mathsf{true}$.
The last condition $\mathsf{AuxCheck}(\mu^*) = \mathsf{true}$ implies that we have two messages $\mu \neq \mu^*$ such that $\mathsf{aux}_\lambda = \mathsf{aux}(\lambda, vk, \mu)$ is equivalent to $\mathsf{aux}_\lambda^* = \mathsf{aux}(\lambda, vk, \mu^*)$. This breaks the collision resistance property of aux with respect to the message on index λ, and we can easily construct a reduction. ◻

Game G_7*:* In this game, the challenger runs $\mathsf{ForkCheck}$ and, if the result is true, then the adversary loses. We have the following two lemmas:

Lemma 4.7. *There exists an adversary $\mathcal{A}_{\text{nd}}$ against the non-divergency of* ID *such that*

$$|\Pr[W_6] - \Pr[W_7]| \leq \mathsf{Adv}^{q_S\text{-nd}}_{\mathsf{ID},\mathcal{A}_{\text{nd}}}(1^\kappa).$$

The running time of $\mathcal{A}_{\text{nd}}$ is approximately that of $\mathcal{A}$.

Proof. The difference between G_6 and G_7 happens if the adversary submits *valid* pair of message and siganture $(\mu^*, (a_1^*, \dots, a_n^*, a_{n+1}^*))$ such that $(\mu^*, (a_1^*, \dots, a_n^*, a_{n+1}^*)) \notin \mathcal{Q}$, the pair passes the collision checks by $\mathsf{CollCheck}$ and $\mathsf{AuxCheck}$, *and* $\mathsf{ForkCheck}_{\mathcal{Q}}(\mu^*, a_1^*, \dots, a_{n+1}^*) = \mathsf{true}$.
If the last check by $\mathsf{ForkCheck}$ is true, then we have two valid transcripts $(\mu^*, a_1^*, c_1^*, \dots, a_{k-1}^*, c_{k-1}^*, a_k^*, c_k^*, \dots, a_n^*, c_n^*, a_{n+1}^*)$ generated by the adversary and

- $(\mu, a_1^*, c_1^*, \dots, a_{k-1}^*, c_{k-1}^*, a_k, c_k, \dots, a_n, c_n, a_{n+1})$ generated by the signing oracle, where $k \in [2, n+1]$ satisfying $a_k \neq a_k^*$; or,
- $(\mu, a_1^*, c_1^*, \dots, a_{k-1}^*, c_{k-1}^*, a_k^*, c_k, \dots, a_n^*, c_n, a_{n+1}^*)$ generated by the signing oracle, where $k \in [2, n]$ satisfying $c_l \neq c_l^*$ for all $l \in [k, n]$.

Notice that, on the former condition, if $k \leq n$, then the collision checks force $c_l \neq c_l^*$ for all $l \in [k, n]$. Hence, the former condition covers the conditions (a) and (b) of $\mathsf{BranchCheck}$ in Definition 3.3. The latter condition is equivalent to the condition (c) of it. Therefore, the two valid transcripts apparently violate q_S-non-divergency of ID, and we can easily construct a reduction. ◻

Lemma 4.8. *There exists an adversary $\mathcal{A}_{\text{nma}}$ against the* EUF-NMA *security of* DS *such that*

$$\Pr[W_7] \leq \mathsf{Adv}^{\text{euf-nma}}_{\mathsf{DS},\mathcal{A}_{\text{nma}}}(1^\kappa).$$

The running time of $\mathcal{A}_{\text{nma}}$ is approximately that of $\mathcal{A}$.

Proof. We show that to win in G_7, the adversary should submit a valid pair of message and signature $(\mu^*, (a_1^*, \dots, a_{n+1}^*)) \notin \mathcal{Q}$ such that the values on $(\mathsf{aux}_i^*, h_{i-1}^*, a_i^*)$ for H and h_i^* for γ_i are not reprogrammed by $\textsc{Sign}$.
If not, there exists at least one index $i \in [n]$ such that H is reprogrammed on input $(\mathsf{aux}_i^*, h_{i-1}^*, a_i^*)$ or γ_i is reprogrammed on input h_i^*. Let $\ell \in [n]$ be the minimum of the indices of the reprogrammed points.

- If H is reprogrammed on input $(\mathsf{aux}_\ell^*, h_{\ell-1}^*, a_\ell^*)$, then the simulator generates a transcript $(a_1, c_1, \dots, a_n, c_n, a_{n+1})$ in the computation of $\textsc{Sign}(\mu)$ for some μ satisfying $(\mathsf{aux}_\ell, h_{\ell-1}, a_\ell) = (\mathsf{aux}_\ell^*, h_{\ell-1}^*, a_\ell^*)$ and $h_\ell = h_\ell^*$. Due to the collision check, $h_{\ell-1} = h_{\ell-1}^*$ implies $(\mathsf{aux}_{\ell-1}, h_{\ell-2}, a_{\ell-1}) = (\mathsf{aux}_{\ell-1}^*, h_{\ell-2}^*, a_{\ell-1}^*)$, and so on. Thus, we have

$$(\mathsf{aux}_j, a_j, h_j, c_j) = (\mathsf{aux}_j^*, a_j^*, h_j^*, c_j^*) \text{ for } j = 1, \dots, \ell, \tag{1}$$

  which implies that H is reprogrammed for the indices $1, \dots, \ell$. Since the index ℓ is the *minimum* of the indices of the reprogrammed points, Equation 1 implies that $\ell = 1$.
  We also note that, since $\mathsf{ForkCheck}_{\mathcal{Q}}(\mu^*, a_1^*, \dots, a_{n+1}^*)$ returns false, $a_1 = a_1^*$ implies that $(a_2, \dots, a_{n+1}) = (a_2^*, \dots, a_{n+1}^*)$.[8] Since we have $a_i = a_i^*$ for all $i \in [n+1]$, μ must be distinct from μ^* due to the check in L.5 of G_7.
  We then consider two subcases on λ:

[8] and more by the second condition of $\mathsf{ForkCheck}$.

$\underline{\mathsf{Sign}_{\mathsf{h}}(sk, \mu)}$
$h_0 := \emptyset$; $c_0 := \emptyset$; $\mathsf{state} := \emptyset$
for $i = 1, \dots, n$ **do**
  $(a_i, \mathsf{state}) \leftarrow \mathsf{P}(sk, c_{i-1}, \mathsf{state})$
  $h_i := \mathsf{H}(\mathsf{aux}_i, h_{i-1}, a_i)$
  $c_i := \gamma_i(h_i)$
$a_{n+1} \leftarrow \mathsf{P}(sk, c_n, \mathsf{state})$
return $\sigma := (h_1, \dots, h_n, a_{n+1})$

$\underline{\mathsf{Vrfy}_{\mathsf{h}}(vk, \mu, \sigma)}$
Parse $\sigma = (h_1, \dots, h_n, a_{n+1})$
$c_0 := \emptyset$; $h_0 := \emptyset$
for $i \in [n]$: $c_i := \gamma_i(h_i)$
$(a_1, \dots, a_n) := \mathsf{Rep}(vk, c_1, \dots, c_n, a_{n+1})$
if $(a_1, \dots, a_n) = \bot$ **then return** false
for $i = 1, \dots, n$: $\bar{h}_i := \mathsf{H}(\mathsf{aux}_i, h_{i-1}, a_i)$
return $\mathsf{boole}(\forall i \in [n] : h_i = \bar{h}_i)$

Fig. 7. Scheme $\mathsf{FS}_{\mathsf{h}}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}] = (\mathsf{Gen}, \mathsf{Sign}_{\mathsf{h}}, \mathsf{Vrfy}_{\mathsf{h}})$, where $\mathsf{ID} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$, $\mathsf{H} : \{0,1\}^* \to \mathcal{H}$ is modeled as the random oracle, and $\gamma_i : \mathcal{H} \to C_i$ for $i \in [n]$ is also modeled as the random oracle. For ease of notation, we let $\mathsf{aux}_i = \mathsf{aux}(i, vk, \mu)$.

- Suppose that $\lambda = 1$. Since $\mu \neq \mu^*$, we have $\mathsf{aux}_1 \neq \mathsf{aux}_1^*$ due to AuxCheck. However, this contradicts $\mathsf{aux}_1 = \mathsf{aux}_1^*$ from Equation 1 with $\ell = 1$.
- Next, suppose that $1 < \lambda \leq n$. Then, we have $\mathsf{aux}_\lambda \neq \mathsf{aux}_\lambda^*$ due to AuxCheck and $\mathsf{aux}_l = \mathsf{aux}_l^*$ for $l = 1, \dots, \lambda - 1$. Since $(\mathsf{aux}_j, h_{j-1}, a_j) = (\mathsf{aux}_j^*, h_{j-1}^*, a_j^*)$ implies $(h_j, c_j) = (h_j^*, c_j^*)$ for $j = 1, \dots, \lambda - 1$ by induction, we have $(h_{\lambda-1}, c_{\lambda-1}) = (h_{\lambda-1}^*, c_{\lambda-1}^*)$.
  Now, in the computation of h_λ and h_λ^*, $\mathsf{aux}_\lambda \neq \mathsf{aux}_\lambda^*$ while $h_{\lambda-1} = h_{\lambda-1}^*$ and $a_\lambda = a_\lambda^*$. Due to the collision check, we have $h_\lambda \neq h_\lambda^*$ and $c_\lambda \neq c_\lambda^*$, which yields $c_l \neq c_l^*$ for all $l > \lambda$. Thus, we have two different transcripts $(a_1, c_1, \dots, a_{n+1})$ and $(a_1^*, c_1^*, \dots, a_{n+1}^*)$ satisfying $a_i = a_i^*$ for all $i \in [n+1]$, $c_i = c_i^*$ for all $i \in [\lambda - 1]$, and $c_i \neq c_i^*$ for all $i \in [\lambda, n]$. But, this case is already eliminated by L.5 of ForkCheck, and this leads to the contradiction.

– If γ_i is reprogrammed on input h_i^*, then the simulator generates a transcript $(a_1, c_1, \dots, a_n, c_n, a_{n+1})$ such that $h_i^* = h_i$ and $c_i^* = c_i$. Due to the collision check, $h_i^* = h_i$ implies $(\mathsf{aux}_i, h_{i-1}, a_i) = (\mathsf{aux}_i^*, h_{i-1}^*, a_i^*)$, and so on. Thus, we again have Equation 1. The following argument is the same as the above case, and we omit it.

In both cases, we arrive at the contradiction, and the adversary's forgery never involves the reprogramming. Since the adversary submits a valid pair $(\mu^*, (a_1^*, \dots, a_n^*, a_{n+1}^*)) \notin \mathcal{Q}$ that causes no reprogramming, we can easily construct $\mathcal{B}$ against the EUF-NMA security of DS. $\square$

Remark 4.1. If γ_i is the identity function, then we can skip a part of G_2 because the identity function is perfectly collision-resistant. We can also skip a part of G_4 since we do not need to reprogram γ_i.

5 FS_{h} for Multi-Pass ID

If one can reproduce $a_1, \dots, a_n$ from the challenges $c_1, \dots, c_n$ and last message a_{n+1}, then we have a chance to replace $a_1, \dots, a_n$ in the signature with the hash values $h_1, \dots, h_n$. This replacement drastically shortens a signature because the prover's messages $a_1, \dots, a_n$ are much longer than the hash values $h_1, \dots, h_n$. We call this variant of the FS transform as FS_{h}. In this section, we adopt the notation and notions for three-pass ID by Backendal, Bellare, Sorrell, and Sun [BBSS18], who studied the variants of the FS transform for three-pass ID. To define FS_{h}, we first define the commitment-reproducing algorithm Rep of ID.

Definition 5.1 (Commitment-reproducing algorithm [BBSS18], adapted). *A commitment-reproducing alglorithm* Rep *is a DPT algorithm that takes $(vk, c_1, \dots, c_n, a_{n+1})$ and messages $(a_1, \dots, a_n)$, which might be $\bot$. We require completeness defined as follows: for honestly generated keys (vk, sk) by* Gen *and transcript $(a_1, c_1, \dots, a_n, c_n, a_{n+1})$, if the transcript is valid, then $(a_1, \dots, a_n) = \mathsf{Rep}(vk, c_1, \dots, c_n, a_{n+1})$.*

The signature scheme obtained by FS_{h} is summarized in Figure 7.
In order to consider the security of FS_{h}, We review the soundness of ID defined in [BBSS18]. This is the notion that one cannot output a part of the transcript $(c_1, c_2, \dots, c_n, a_{n+1})$ such that if we reproduce non-$\bot$ messages $(a_1, \dots, a_n)$ by Rep, then the transcript $(a_1, c_1, \dots, c_n, a_{n+1})$ is valid. Since we want to consider FS_{h}, we replace $c_1, \dots, c_n$ with $h_1, \dots, h_n$ as follows:

Definition 5.2 (Soundness of ID [BBSS18, Sec.3], extended for FS_{h}). *A commitment-reproducible ID scheme* ID *is said to be* computationally sound *if, for any QPT adversary $\mathcal{A}$, its advantage is negligible in the security parameter, where the advantage is defined as*

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathsf{ID},\boldsymbol{\gamma},\mathcal{A}}(1^\kappa) := \Pr[\mathsf{Expt}^{\mathsf{sound}}_{\mathsf{ID},\boldsymbol{\gamma},\mathcal{A}}(1^\kappa) = \mathsf{true}],$$

and $\mathsf{Expt}^{\mathsf{sound}}_{\mathsf{ID},\boldsymbol{\gamma},\mathcal{A}}(1^\kappa)$ is defined in Figure 8.
If the advantage is 0 for any unbounded adversary, we say that the scheme is perfectly sound.

$$
\begin{array}{ll}
1: & \underline{\mathsf{Expt}^{\mathrm{sound}}_{\mathsf{ID},\boldsymbol{\gamma},\mathcal{A}}(1^\kappa)} \\
2: & (vk, sk) \leftarrow \mathsf{Gen}(1^\kappa) \\
3: & (h_1, h_2, \dots, h_n, a_{n+1}) \leftarrow \mathcal{A}^{|\gamma\rangle}(vk) \\
4: & \textbf{forall } i \in [n]\text{: } c_i := \gamma_i(h_i) \\
5: & (a_1, \dots, a_n) := \mathsf{Rep}(vk, c_1, \dots, c_n, a_{n+1}) \\
6: & d \leftarrow \mathsf{Vrfy}(vk, a_1, c_1, \dots, a_n, c_n, a_{n+1}) \\
7: & \textbf{return } d \wedge \mathsf{boole}((a_1, \dots, a_n) \neq \bot)
\end{array}
$$

Fig. 8. $\mathsf{Expt}^{\mathrm{sound}}_{\mathsf{ID},\boldsymbol{\gamma},\mathcal{A}}(1^\kappa)$.

It is easy to check that if a verification algorithm internally uses Rep and checks whether the given messages are equivalent to the reproduced messages or not, then the ID scheme is perfectly sound.

Lemma 5.1 (Special verifier means perfect soundness, extended for FS_h). *Suppose that, on input $(vk, a_1, c_1, \dots, c_n, a_{n+1})$, the verification algorithm* V *outputs* $\mathsf{boole}(\mathsf{Rep}(vk, c_1, \dots, c_n, a_{n+1}) = (a_1, \dots, a_n))$. *Then, the identification scheme* ID *is perfectly sound.*

We show the following theorem as [BBSS18]. The proof is in Section A.

Theorem 5.1 ($\mathsf{FS}_\mathrm{cmt} \Rightarrow \mathsf{FS}_\mathrm{h}$). *Suppose that* ID *is computationally sound. If $\mathsf{FS}_\mathrm{cmt}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$ is* EUF-CMA/sEUF-CMA-*secure, then $\mathsf{FS}_\mathrm{h}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$ is also, respectively.*

5.1 S-DEO, S-CEO, M-S-UEO, MBS, and wNR of FS_h

FS_h has another advantage on the BUFF securities since a signature inherently includes hash values. Let us consider the BUFF securities of FS_h. For example, PERK [ABB+23a] computes the first hash value h_1 as $\mathsf{H}(\mathtt{0x01}, \mathsf{salt}, \mu, vk, a_1)$, which means that $\mathsf{aux}(1, vk, \mu) = (\mathtt{0x01}, \mathsf{salt}, \mu, vk)$ and aux is perfectly collision-resistant with respect to $\lambda = 1$. PERK's signature is of the form (h_1, h_2, a_3), where a_3 includes salt.

If the adversary breaks the MBS security by outputting vk, $\mu \neq \mu'$, and $\sigma = (h_1, \dots, h_n, a_{n+1})$, then we have a collision of aux_i for some i or a collision of H. In addition, if the adversary breaks the M-S-UEO security by outputting $vk \neq vk'$, μ, μ', and $\sigma = (h_1, \dots, h_n, a_{n+1})$, then such values yield a collision with respect to the verification key of aux_i for some i or a collision of H. Summarizing the above argument, we obtain the following lemma:

Lemma 5.2. *Let* ID *be a $(2n+1)$-pass ID scheme. Let $\mathsf{H}\colon \{0,1\}^* \to \mathcal{H}$ be a hash function. Let $\mathsf{DS} := \mathsf{FS}_\mathrm{h}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$. Assume that* H *is collision-resistant.*

- *If* aux *is collision-resistant with respect to message on index λ, then* DS *satisfies* S-DEO *and* MBS.
- *If* aux *is also collision-resistant with respect to the verification key on index λ', then* DS *further satisfies* M-S-UEO.

Furthermore, we can show wNR security of FS_h in the (Q)ROM.

Lemma 5.3. *Let* H *be a random oracle. Suppose that* aux *is collision-resistant with respect to the verification key on index λ and there exists index $\zeta \in [\lambda, n]$ such that aux_ζ can be written as (μ, η_ζ) for some η_ζ. Then $\mathsf{DS} = \mathsf{FS}_\mathrm{h}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$ satisfies* wNR *in the (Q)ROM.*

The proof is in subsection A.3.

Remark 5.1. See Table 2. AIMer, MQOM, and PERK satisfy the condition of Lemma 5.3. The hash values in MIRA and RYDE involve $H(\mu)$ instead of μ. The proof is obtained similarly by inserting one game. We will require an additional argument to show wNR security of FAEST and SDitH because their signature consists of h_n and a_{n+1}. See Section B, subsection G.2, and subsection H.1 for the details.

6 Biscuit

We briefly review Biscuit v1.1[9], which is an MPCitH signature based on a variant of the multivariate quadratic-equations problem.

[9] Version 1.1 is available at https://www.biscuit-pqc.org/

The signing key is $\boldsymbol{s} \leftarrow \mathbb{F}_q^n$. The verification key consists of $\mathsf{seedF} \in \{0,1\}^\kappa$ and $\boldsymbol{t} \in \mathbb{F}_q^m$; seedF produces a sequence of random elements in $\mathbb{F}_q$ and generates $\boldsymbol{f} = (f_1, \ldots, f_m) \in \mathbb{F}_q[x_1, \ldots, x_n]^m$ with $f_k = A_{k,0} + A_{k,1} \cdot A_{k,2}$ for $k \in [m]$, where $A_{k,j}(x_1, \ldots, x_n) = a_0^{(k,j)} + \sum_{i \in [n]} a_i^{(k,j)} x_i \in \mathbb{F}_q[x_1, \ldots, x_n]$ is a random Affine form; and $\boldsymbol{t}$ is $\boldsymbol{f}(\boldsymbol{s})$. For two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_q^m$, $\boldsymbol{a} \odot \boldsymbol{b}$ is defined as component-wise multiplication. For a vector $\boldsymbol{a} \in \mathbb{F}_q^k$, we denote shares of $\boldsymbol{a}$ via an (N,N)-additive secret share as $[\![\boldsymbol{a}]\!] = ([\![\boldsymbol{a}]\!]_1, \ldots, [\![\boldsymbol{a}]\!]_N) \in (\mathbb{F}_q^k)^N$.
In nutshell, the signer will show the relation that $\boldsymbol{z} = \boldsymbol{t} - A_0(\boldsymbol{s}) = \boldsymbol{x} \odot \boldsymbol{y}$, where $\boldsymbol{x} = A_1(\boldsymbol{s})$ and $\boldsymbol{y} = A_2(\boldsymbol{s})$ via an MPCitH protocol.
We modify the underlying MPCitH protocol $\mathsf{ID}_{\mathsf{Biscuit}}$, $\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3)$ and V with Rep, as depicted in Figure 9 to fit their scheme in our framework. The algorithms in the protocol are summarized as follows:

- $\mathsf{TreePRG}$ computes N pseudorandom seeds by using a binary tree structure.
- Path comptues a path of $\log_2(N)$ values.
- $\mathsf{Reconst}$ computes $N-1$ seeds for $i \neq i_e^*$ by using the path of $\log_2(N)$ values.
- $\mathsf{MakeShares}$ generates pseudorandom shares from the seed $\mathsf{seed}_i^{(e)}$.
- $\mathsf{LinearCircuit}$ computes shares of $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ from a share of $\boldsymbol{s}$ as defined in Figure 9.

Notice that Rep computes $[\![\boldsymbol{v}]\!]_{i^*} := -\sum_{i \neq i^*} [\![\boldsymbol{v}]\!]_i$. Thus, the verifier V checks if $\sum_i [\![\boldsymbol{v}]\!]_i = \boldsymbol{0}$ as the MPC's result. For the details, see the original specification [BKPV23].
The signature scheme $\mathsf{Biscuit} = \mathsf{FS}_{\mathsf{h}}[\mathsf{ID}_{\mathsf{Biscuit}}, \mathsf{H}, \boldsymbol{\gamma}]$ is defined by $\mathsf{aux}_1 = (\mathtt{0x01}, \mathsf{salt}, \mu)$ and $\mathsf{aux}_2 = (\mathtt{0x02}, \mathsf{salt})$.

6.1 Security

sEUF-CMA *security:* To show the sEUF-CMA security, we discuss the protocol's HVZK property and non-divergency. For the definitions of primitives, see Section A.
The HVZK property of $\mathsf{ID}_{\mathsf{Biscuit}}$ is shown in their specification document by following the HVZK proof in [FJR22], but we modify the proof to consider the real protocol as possible. For the proof sketch, see subsection A.4.

Lemma 6.1 (q_S-HVZK). *Suppose that* PRF *is secure,* $\mathsf{TreePRG}$ *and* $\mathsf{MakeShares}$ *are pseudorandom, and* Com *is hiding. Let q_S be a polynomial of 1^κ. Then,* $\mathsf{ID}_{\mathsf{Biscuit}}$ *with simulator* $\mathsf{Sim}_{\mathsf{Biscuit}}$ *in Figure 10 is q_S-HVZK.*

We next show that $\mathsf{ID}_{\mathsf{Biscuit}}$ is strongly non-divergent.

Lemma 6.2 (Strong non-divergency). *Suppose that* Com *is non-invertible and collision-resistant and* $\mathsf{Reconst}$ *is collision-resistant. Then,* $\mathsf{ID}_{\mathsf{Biscuit}}$ *for* $\mathsf{Biscuit}$ *is strongly non-divergent with respect to* $\mathsf{Sim}_{\mathsf{Biscuit}}$.

Proof. For simplicity, we ignore parallelness τ. Suppose that the adversary declines a valid transcript $\mathsf{trans}_i = (a_1, c_1, a_2, c_2, a_3)$ generated by the simulator and outputs a valid transcript $\mathsf{trans}' = (a_1, c_1, a_2', c_2', a_3')$. We parse them as $a_1 = (\mathsf{com}_1, \ldots, \mathsf{com}_N, \boldsymbol{\Delta s}, \boldsymbol{\Delta c})$ and $c_1 = \boldsymbol{\varepsilon}$.
If condition (a) of $\mathsf{BranchCheck}'$ in Definition 3.6 is met, then we have $c_2 \neq c_2'$: We parse $c_2 = i^*$, $c_2' = i^+$, and $a_3' = (\mathsf{salt}', \mathsf{path}', \boldsymbol{\Delta s}', \boldsymbol{\Delta c}', \mathsf{com}_{i^+}', [\![\boldsymbol{\alpha}']\!]_{i^+})$. In this case, the adversary opens com_{i^*} in a_1 as $(\mathsf{salt}', i^*, \mathsf{seed}_{i^*}', \rho_{i^*}')$ computed from path' and i^+ since $i^* \neq i^+$ and the transcript $(a_1, c_1, a_2', c_2', a_3')$ is valid. Thus, we have $\mathsf{com}_{i^*} = \mathsf{Com}(\mathsf{salt}', i^*, \mathsf{seed}_{i^*}'; \rho_{i^*}')$. Since com_{i^*} is chosen uniformly at random in L.11 of the simulator $\mathsf{Sim}_{\mathsf{Biscuit}}$ in Figure 10, this violates the non-invertibility of Com.
If condition (b) of $\mathsf{BranchCheck}'$ is met, then we have $(a_2, c_2) = (a_2', c_2')$ and $a_3 \neq a_3'$. We then parse $a_2 = ([\![\boldsymbol{\alpha}]\!]_i, [\![\boldsymbol{v}]\!]_i)_{i \in [N]}$, $c_2 = i^*$, $a_3 = (\mathsf{salt}, \mathsf{path}, \boldsymbol{\Delta s}, \boldsymbol{\Delta c}, \mathsf{com}_{i^*}, [\![\boldsymbol{\alpha}]\!]_{i^*})$, and $a_3' = (\mathsf{salt}', \mathsf{path}', \boldsymbol{\Delta s}', \boldsymbol{\Delta c}', \mathsf{com}_{i^*}', [\![\boldsymbol{\alpha}']\!]_{i^*})$.
We have the following cases:

- If $\mathsf{salt} \neq \mathsf{salt}'$, then we have a collision for Com.
- If $\mathsf{path} \neq \mathsf{path}'$:
  - If $(\mathsf{state}_i, \rho_i)_{i \neq i^*} = (\mathsf{state}_i', \rho_i')_{i \neq i^*}$, then it implies the collision for $\mathsf{Reconst}$.
  - If $(\mathsf{state}_i, \rho_i)_{i \neq i^*} \neq (\mathsf{state}_i', \rho_i')_{i \neq i^*}$, then we have at least one index i satisfying $(\mathsf{state}_i, \rho_i) \neq (\mathsf{state}_i', \rho_i)$. Since the two transcripts are valid, we have a collision as $\mathsf{com}_i = \mathsf{Com}(\mathsf{salt}, i, \mathsf{state}_i; \rho_i) = \mathsf{Com}(\mathsf{salt}, i, \mathsf{state}_i', \rho_i')$.
- If $(\boldsymbol{\Delta s}, \boldsymbol{\Delta c}, \mathsf{com}_{i^*}, [\![\boldsymbol{\alpha}]\!]_{i^*}) \neq (\boldsymbol{\Delta s}', \boldsymbol{\Delta c}', \mathsf{com}_{i^*}', [\![\boldsymbol{\alpha}]\!]_{i^*})$, then at least one of two transcripts are invalid because of inconsistency with a_1 and a_2, and this never happens.

Using those observations, we can construct reductions easily. □

Since the scheme is (strongly) non-divergent and HVZK, we have the following theorem:

Theorem 6.1 (Biscuit's sEUF-CMA security). *Suppose that* $\mathsf{Biscuit} = \mathsf{FS}_{\mathsf{h}}[\mathsf{ID}_{\mathsf{Biscuit}}, \mathsf{H}, \boldsymbol{\gamma}]$ *is* EUF-NMA*-secure in the (Q)ROM,* PRF, $\mathsf{TreePRG}$, *and* $\mathsf{MakeShares}$ *are pseudorandom,* Com *is hiding, non-invertible, binding, and collision-resistant, and* $\mathsf{Reconst}$ *is collision-resistant. Then,* $\mathsf{Biscuit}$ *is* sEUF-CMA*-secure in the (Q)ROM.*

S-DEO *and* MBS *security:* $\mathsf{Biscuit}$ employs FS_{h} with $\mathsf{aux}_1 = (\mathtt{0x01}, \mathsf{salt}, \mu)$ and $\mathsf{aux}_2 = (\mathtt{0x02}, \mathsf{salt})$. Therefore, h_1 in the signature includes the information of μ. Since aux is perfectly collision-resistant with respect to message on index 1, according to Lemma 5.2, $\mathsf{Biscuit}$ satisfies S-DEO and MBS if H is collision-resistant.

$\underline{\mathsf{P}_1(sk) \text{ for Biscuit}}$

Extract $\mathsf{seedF}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{y}$ from sk

Re-compute $\boldsymbol{f}$ from seedF

Choose rnd uniformly at random

`//Setup MPC`

$(\mathsf{salt}, (\mathsf{seed}^{(e)})_{e\in[\tau]}) := \mathsf{PRF}(\mathsf{rnd}, (sk, \mu))$

`//Run in` τ `parallel. We omit` $^{(e)}$.

`//The original doesn't have` ρ_i

$(\mathsf{seed}_i, \rho_i)_{i\in[N]} := \mathsf{TreePRG}(\mathsf{seed}, (\mathsf{salt}, e))$

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

- $\mathsf{com}_i := \mathsf{Com}((\mathsf{salt}, e, i, \mathsf{seed}_i); \rho_i)$
- $([\![\boldsymbol{s}]\!]_i, [\![\boldsymbol{a}]\!]_i, [\![\boldsymbol{c}]\!]_i) := \mathsf{MakeShares}(\mathsf{seed}_i, (\mathsf{salt}, e, i))$

$\boldsymbol{\Delta s} := \boldsymbol{s} - \sum_{i\in[N]} [\![\boldsymbol{s}]\!]_i$

$\boldsymbol{\Delta c} := \boldsymbol{y} \odot \sum_{i\in[N]} [\![\boldsymbol{a}]\!]_i - \sum_{i\in[N]} [\![\boldsymbol{c}]\!]_i$

$[\![\boldsymbol{s}]\!]_1 := [\![\boldsymbol{s}]\!]_1 + \boldsymbol{\Delta s}$

$[\![\boldsymbol{c}]\!]_1 := [\![\boldsymbol{c}]\!]_1 + \boldsymbol{\Delta c}$ `//`$\boldsymbol{c} = \boldsymbol{y} \odot \boldsymbol{a}$

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

- $([\![\boldsymbol{x}]\!]_i, [\![\boldsymbol{y}]\!]_i, [\![\boldsymbol{z}]\!]_i) := \mathsf{LinearCircuit}([\![\boldsymbol{s}]\!]_i, i, \boldsymbol{t}, \boldsymbol{f})$

$a_1 := \left((\mathsf{com}_i)_{i\in[N]}, \boldsymbol{\Delta s}, \boldsymbol{\Delta c}\right)_{e\in[\tau]}$

$\mathsf{state} := \Big(\mathsf{salt}, \left(\mathsf{seed}, (\mathsf{com}_i)_{i\in[N]}, \boldsymbol{\Delta s}, \boldsymbol{\Delta c}, [\![\boldsymbol{x}]\!], [\![\boldsymbol{y}]\!], [\![\boldsymbol{z}]\!]\right)_{e\in[\tau]}\Big)$

return a_1 and state

$\underline{\mathsf{P}_2(sk, c_2, \mathsf{state}) \text{ for Biscuit}}$

Parse $c_2 = (\boldsymbol{\varepsilon}^{(1)}, \ldots, \boldsymbol{\varepsilon}^{(\tau)})$

`//Simulate MPC`

`//Run in` τ `parallel. We omit` $^{(e)}$.

forall $i \in [N]$: $[\![\boldsymbol{\alpha}]\!]_i := [\![\boldsymbol{x}]\!]_i \odot \boldsymbol{\varepsilon} + [\![\boldsymbol{a}]\!]_i$

$\boldsymbol{\alpha} := \sum_{i\in[N]} [\![\boldsymbol{\alpha}]\!]_i$

forall $i \in [N]$: $[\![\boldsymbol{v}]\!]_i := [\![\boldsymbol{y}]\!]_i \odot \boldsymbol{\alpha} - [\![\boldsymbol{z}]\!]_i \odot \boldsymbol{\varepsilon} - [\![\boldsymbol{c}]\!]_i$

$a_2 := \left(([\![\boldsymbol{\alpha}]\!]_i, [\![\boldsymbol{v}]\!]_i)_{i\in[N]}\right)_{e\in[\tau]}$

$\mathsf{state} := \left(\mathsf{salt}, \left(\mathsf{seed}, (\mathsf{com}_i)_{i\in[N]}, \boldsymbol{\Delta s}, \boldsymbol{\Delta c}, [\![\boldsymbol{\alpha}]\!]\right)_{e\in[\tau]}\right)$

return a_2 and state

$\underline{\mathsf{P}_3(sk, c_3, \mathsf{state}) \text{ for MiRitH}}$

Parse $c_3 = (i_1^*, \ldots, i_\tau^*)$

`//Run in` τ `parallel. We omit` $^{(e)}$ `and` $_e$.

$\mathsf{path} := \mathsf{GetPath}(i^*, \mathsf{seed}, (\mathsf{salt}, e))$

$a_3 := \left(\mathsf{salt}, \left(\mathsf{path}, \boldsymbol{\Delta s}, \boldsymbol{\Delta c}, \mathsf{com}_{i^*}, [\![\boldsymbol{\alpha}]\!]_{i^*}\right)_{e\in[\tau]}\right)$

return a_3

$\underline{\mathsf{Rep}(vk, c_1, c_2, a_3) \text{ for Biscuit}}$

Parse $vk = (\mathsf{seedF}, \boldsymbol{t})$

Re-compute $\boldsymbol{f}$ from seedF

Parse $c_1 = (\boldsymbol{\varepsilon}^{(1)}, \ldots, \boldsymbol{\varepsilon}^{(\tau)})$

Parse $c_2 = (i_1^*, \ldots, i_\tau^*)$

Parse $a_3 = \left(\mathsf{salt}, (\mathsf{path}, \boldsymbol{\Delta s}, \boldsymbol{\Delta c}, \mathsf{com}_{i^*}, [\![\boldsymbol{\alpha}]\!]_{i^*})_{e\in[\tau]}\right)$

`//Reconstruct` a_1.

`//Run in` τ `parallel. We omit` $^{(e)}$ `and` $_e$.

$(\mathsf{seed}_i, \rho_i)_{i\neq i^*} := \mathsf{Reconst}(\mathsf{path}, i^*, (\mathsf{salt}, e))$

forall $i \in [N] \setminus \{i_e^*\}$ **do**

- $\mathsf{com}_i := \mathsf{Com}((\mathsf{salt}, e, i, \mathsf{seed}_i); \rho_i)$
- $([\![\boldsymbol{s}]\!]_i, [\![\boldsymbol{a}]\!]_i, [\![\boldsymbol{c}]\!]_i) := \mathsf{MakeShares}(\mathsf{seed}_i, (salt, e, i))$
- **if** $i = 1$ **then**
  - $[\![\boldsymbol{s}]\!]_1 := [\![\boldsymbol{s}]\!]_1 + \boldsymbol{\Delta s}$
  - $[\![\boldsymbol{c}]\!]_1 := [\![\boldsymbol{c}]\!]_1 + \boldsymbol{\Delta c}$
- $([\![\boldsymbol{x}]\!]_i, [\![\boldsymbol{y}]\!]_i, [\![\boldsymbol{z}]\!]_i) := \mathsf{LinearCircuit}([\![\boldsymbol{s}]\!]_i, i, \boldsymbol{t}, \boldsymbol{f})$

$\bar{a}_1 := \left((\mathsf{com}_i)_{i\in[N]}, \boldsymbol{\Delta s}, \boldsymbol{\Delta c}\right)_{e\in[\tau]}$

`//Reconstruct` a_2.

`//Run in` τ `parallel. We omit` $^{(e)}$ `and` $_e$.

forall $i \in [N] \setminus \{i^*\}$: $[\![\boldsymbol{\alpha}]\!]_i := [\![\boldsymbol{x}]\!]_i \odot \boldsymbol{\varepsilon} + [\![\boldsymbol{a}]\!]_i$

$\boldsymbol{\alpha} := \sum_i [\![\boldsymbol{\alpha}]\!]_i$

forall $i \in [N] \setminus \{i^*\}$: $[\![\boldsymbol{v}]\!]_i := [\![\boldsymbol{y}]\!]_i \odot \boldsymbol{\alpha} - [\![\boldsymbol{z}]\!]_i \odot \boldsymbol{\varepsilon} - [\![\boldsymbol{c}]\!]_i$

$[\![\boldsymbol{v}]\!]_{i^*} := -\sum_{i\neq i^*} [\![\boldsymbol{v}]\!]_i$

$\bar{a}_2 := \left(([\![\boldsymbol{\alpha}]\!]_i, [\![\boldsymbol{v}]\!]_i)_{i\in[N]}\right)_{e\in[\tau]}$

return $\bar{a}_1$ and $\bar{a}_2$

$\underline{\mathsf{V}(vk, a_1, c_1, a_2, c_2, a_3)}$

Compute $(\bar{a}_1, \bar{a}_2) := \mathsf{Rep}(vk, c_1, c_2, a_3)$

return $\mathsf{boole}((\bar{a}_1, \bar{a}_2) = (a_1, a_2))$

$\underline{\mathsf{LinearCircuit}(\boldsymbol{s}, \mathsf{idx}, \boldsymbol{t}, \boldsymbol{f})}$

Parse $\boldsymbol{f} = (f_1, \ldots, f_m)$

Parse $f_k = A_{k,0} + A_{k,1} \cdot A_{k,2}$ for $k \in [m]$

Let $a_0^{(k,j)}$ be a constant term of $A_{k,j}$

if $\mathsf{idx} = 1$ **then**

- $A'_{k,j} := A_{k,j}$

else

- $A'_{k,j} := A_{k,j} - a_0^{(k,j)}$

$\boldsymbol{x} := (A'_{1,1}(\boldsymbol{s}), \ldots, A'_{m,1}(\boldsymbol{s}))$

$\boldsymbol{y} := (A'_{1,2}(\boldsymbol{s}), \ldots, A'_{m,2}(\boldsymbol{s}))$

if $\mathsf{idx} = 1$ **then**

- $\boldsymbol{z} := -(A'_{1,0}(\boldsymbol{s}), \ldots, A'_{m,0}(\boldsymbol{s}))$

else

- $\boldsymbol{z} := \boldsymbol{t} - (A'_{1,0}(\boldsymbol{s}), \ldots, A'_{m,0}(\boldsymbol{s}))$

return $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$

Fig. 9. Prover, reconstruction, and verification algorithms of $\mathsf{ID}_{\mathsf{Biscuit}}$.

$\underline{\mathsf{Sim}_{\mathsf{Biscuit}}(vk, c_1, c_2) \text{ for Biscuit}}$

Parse $vk = (\mathsf{seedF}, \boldsymbol{t})$

Re-compute $\boldsymbol{f}$ from seedF

Parse $c_1 = (\boldsymbol{\varepsilon}^{(1)}, \ldots, \boldsymbol{\varepsilon}^{(\tau)})$

Parse $c_2 = (i_1^*, \ldots, i_\tau^*)$

`//Simulate MPC's setup`

Choose $\mathsf{salt}, \mathsf{seed}^{(1)}, \ldots, \mathsf{seed}^{(\tau)}$ uniformly at random

`//Run in` τ `parallel. We omit` $^{(e)}$ `and` $_e$.

$(\mathsf{seed}_i, \rho_i)_{i \in [N]} := \mathsf{TreePRG}(\mathsf{seed}, (\mathsf{salt}, e))$

forall $i \in [N] \setminus \{i^*\}$ **do**

  $\mathsf{com}_i := \mathsf{Com}((\mathsf{salt}, e, i, \mathsf{seed}_i); \rho_i)$

  $(\llbracket \boldsymbol{s} \rrbracket_i, \llbracket \boldsymbol{a} \rrbracket_i, \llbracket \boldsymbol{c} \rrbracket_i) := \mathsf{MakeShares}(\mathsf{seed}_i, (\mathsf{salt}, e, i))$

Choose com_{i^*} uniformly at random

$\boldsymbol{\Delta s} \leftarrow \mathbb{F}_q^n$, $\boldsymbol{\Delta c} \leftarrow \mathbb{F}_q^m$

$a_1 := \big((\mathsf{com}_i)_{i \in [N]}, \boldsymbol{\Delta s}, \boldsymbol{\Delta c}\big)_{e \in [\tau]}$

`//Simulate MPC's execution`

`//Run in` τ `parallel. We omit` $^{(e)}$ `and` $_e$.

for $e \in [\tau]$ **do**

  forall $i \in [N] \setminus \{i^*\}$: $\llbracket \boldsymbol{\alpha} \rrbracket_i := \llbracket \boldsymbol{x} \rrbracket_i \odot \boldsymbol{\varepsilon} + \llbracket \boldsymbol{a} \rrbracket_i$

  $\llbracket \boldsymbol{\alpha} \rrbracket_{i^*} \leftarrow \mathbb{F}_q^m$

  $\boldsymbol{\alpha} := \sum_i \llbracket \boldsymbol{\alpha} \rrbracket_i$

  forall $i \in [N] \setminus \{i^*\}$:

    $\llbracket \boldsymbol{v} \rrbracket_i := \llbracket \boldsymbol{y} \rrbracket_i \odot \boldsymbol{\alpha} - \llbracket \boldsymbol{z} \rrbracket_i \odot \boldsymbol{\varepsilon} - \llbracket \boldsymbol{c} \rrbracket_i$

  $\llbracket \boldsymbol{v} \rrbracket_{i^*} := -\sum_{i \neq i^*} \llbracket \boldsymbol{v} \rrbracket_i$

$a_2 := \big((\llbracket \boldsymbol{\alpha} \rrbracket_i, \llbracket \boldsymbol{v} \rrbracket_i)_{i \in [N]}\big)_{e \in [\tau]}$

`//Simulate response`

$\mathsf{path} := \mathsf{GetPath}(i^*, \mathsf{seed}, (\mathsf{salt}, e))$

$a_3 := \big(\mathsf{salt}, (\mathsf{path}, \boldsymbol{\Delta s}, \boldsymbol{\Delta c}, \mathsf{com}_{i^*}, \llbracket \boldsymbol{\alpha} \rrbracket_{i^*})_{e \in [\tau]}\big)$

return a_1, a_2, and a_3

Fig. 10. Simulation algorithm for $\mathsf{ID}_{\mathsf{Biscuit}}$.

6.2 S-CEO and wNR Insecurity

Since aux_1 and aux_2 have no information on vk, Biscuit may be S-CEO insecure. We indeed show Biscuit is S-CEO and wNR insecure in some parameter sets.

S-CEO *insecurity:* To break S-CEO security, an adversary needs to output a new verification key vk' on which a message μ and a signature σ is valid, while the adversary obtains (μ, σ) from the signing oracle $\mathsf{Sign}(sk, \cdot)$ many times as in the CMA setting.
We notice that, in the verification procedure, $\boldsymbol{t}$ appears only $\llbracket \boldsymbol{z} \rrbracket_i := \boldsymbol{t} - (A'_{1,0}(\llbracket \boldsymbol{s} \rrbracket_i), \ldots, A'_{m,0}(\llbracket \boldsymbol{s} \rrbracket_i))$ for $i \neq 1$ (L.14 of LinearCircuit).[10] In addition, the direct computation involving $\llbracket \boldsymbol{z} \rrbracket$ is $\llbracket \boldsymbol{z} \rrbracket_i \odot \boldsymbol{\varepsilon}$ in L.18 of Rep.
Exploiting $\boldsymbol{\varepsilon} \in \mathbb{F}_q^m$, we can consider the following attack: Suppose that we have a signature such that $\varepsilon_j^{(1)} = \cdots = \varepsilon_j^{(\tau)} = 0$ for some $j \in [m]$. We then replace $\boldsymbol{t}$ with $\boldsymbol{t}' := \boldsymbol{t} + \boldsymbol{e}_j$ while keeping seedF, where $\boldsymbol{e}_j$ is the j-th unit vector in $\mathbb{F}_q^m$.
This attack is justified as follows: When we consider the computation of $\llbracket \boldsymbol{z} \rrbracket'_i$ in LinearCircuit on this modified $\boldsymbol{t}'$, we have $\llbracket \boldsymbol{z} \rrbracket'_i = \llbracket \boldsymbol{z} \rrbracket_i$ for $i = 1$ and $\llbracket \boldsymbol{z} \rrbracket_i + \boldsymbol{e}_j$ for $i = 2, \ldots, N$. If $\varepsilon_j^{(e)} = 0$ for all $e \in [\tau]$ holds, then we have $\llbracket \boldsymbol{v} \rrbracket'_i = \llbracket \boldsymbol{v} \rrbracket_i$ in the verification algorithm, since $\llbracket \boldsymbol{z} \rrbracket'_i \odot \boldsymbol{\varepsilon} = \llbracket \boldsymbol{z} \rrbracket_i \odot \boldsymbol{\varepsilon}$ for any $i \in [N]$. Thus, the verification is passed on μ, σ, and the shifted verification key $(\mathsf{seedF}, \boldsymbol{t} + \boldsymbol{e}_j)$.
This attack succeeds if we have a signature and an index $j \in [m]$ such that $\varepsilon_j^{(1)} = \cdots = \varepsilon_j^{(\tau)} = 0$. Assuming that γ_1 is the random oracle, each signature satisfies this condition with probability p_1 defined as $p_1 := 1 - (1 - q^{-\tau})^m$. After Q signing queries, there is at least one signature satisfying the condition with probability $p_Q := 1 - (1 - p_1)^Q = 1 - (1 - q^{-\tau})^{mQ}$.
Table 3 summarizes the parameter sets of Biscuit and the success probability of the above S-CEO attack with $Q = 2^{64}$,[11] where, for small a and large b, we use approximations $1 - (1-a)^b \approx ab$ for $ab \ll 1$ and $\approx 1 - \exp(-ab)$ otherwise. Since every p_Q is larger that $2^{-\kappa}$, Biscuit is S-CEO-insecure.

wNR *insecurity:* The above attack for S-CEO insecurity can be used to mount wNR attack. In the wNR game, we are given vk and σ on μ and need to produce $vk' \neq vk$ and σ' such that (vk', μ, σ') is valid while we cannot see μ. Notice that the above attack does not use the information of μ and "hijacks" a given signature. Hence, the above S-CEO adversary works as the wNR attack. In the wNR security game, the adversary is given a single signature instead of Q signatures. Thus, the success probability is p_1 in Table 3. Since p_1 for parameter sets end with s is larger than $2^{-\kappa}$, Biscuit is wNR-insecure depending on the parameter sets. We leave an open problem to find a more sophisticated wNR attack against Biscuit.

[10] One might wonder why $\boldsymbol{t}$ is added for all $i \neq 1$, instead of only for $i = 1$. We can use these offsets since $q = 16$.
[11] "For the purpose of estimating security strengths, it may be assumed that the attacker has access to signatures for no more than 2^{64} chosen messages" [NIS22, 4.B.2].

Table 3. Parameter sets in Biscuit's specification v1.1 and success probabitliy p_Q of S-CEO attack with $Q = 2^{64}$ signing queries.

name	q	n	m	τ	N	p_1	p_Q
biscuit128f	16	64	67	34	16	$\approx 2^{-129.934}$	$\approx 2^{-65.934}$
biscuit128s	16	64	67	18	256	$\approx 2^{-65.934}$	≈ 0.230
biscuit192f	16	87	90	55	16	$\approx 2^{-213.508}$	$\approx 2^{-149.508}$
biscuit192s	16	87	90	31	256	$\approx 2^{-117.508}$	$\approx 2^{-53.508}$
biscuit256f	16	118	121	74	16	$\approx 2^{-289.081}$	$\approx 2^{-225.081}$
biscuit256s	16	118	121	42	256	$\approx 2^{-161.081}$	$\approx 2^{-97.081}$

References

ABB+23a. Najwa Aaraj, Slim Bettaieb, Loïc Bidoux, Alessandro Budroni, Victor Dyseryn, Andre Esser, Philippe Gaborit, Mukul Kulkarni, Victor Mateu, Marco Palumbi, Lucas Perin, and Jean-Pierre Tillich. PERK. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. 2, 3, 6, 17, 34, 35

ABB+23b. Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Matthieu Rivain, Jean-Pierre Tillich, and Adrien Vinçotte. RYDE. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. 2, 3, 6, 37

ABB+23c. Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Matthieu Rivain, and Jean-Pierre Tillich. MIRA. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. 2, 3, 6, 37

ADM+24. Thomas Aulbach, Samed Düzlü, Michael Meyer, Patrick Struck, and Maximiliane Weishäupl. Hash your keys before signing: BUFF security of the additional NIST PQC signatures. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *PQCrypto 2024, Part II*, volume 14772 of *LNCS*, pages 301–335. Springer, 2024. See also https://eprint.iacr.org/2024/591. 3, 8

AFG+23. Carlos Aguilar-Melchor, Thibauld Feneuil, Nicolas Gama, Shay Gueron, James Howe, David Joseph, Antoine Joux, Edoardo Persichetti, Tovohery H. Randrianarisoa, Matthieu Rivain, and Dongze Yue. SDitH — Syndrome Decoding in the Head. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. 2, 3, 6, 38

AFLT16. Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly secure signatures from lossy identification schemes. *Journal of Cryptology*, 29(3):597–631, July 2016. 2

AGH+23. Carlos Aguilar-Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. The return of the SDitH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 564–596. Springer, Heidelberg, April 2023. 37

AHJ+23. Carlos Aguilar Melchor, Andreas Hülsing, David Joseph, Christian Majenz, Eyal Ronen, and Dongze Yue. SDitH in the QROM. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VII*, volume 14444 of *LNCS*, pages 317–350. Springer, Heidelberg, December 2023. 5, 29, 38

AHU19. Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 269–295. Springer, Heidelberg, August 2019. 26

ARV+23. Gora Adj, Luis Rivera-Zamarripa, Javier Verbel, Emanuele Bellini, Stefano Barbero, Andre Esser, Carlo Sanna, and Floyd Zweydinger. MiRitH — MinRank in the Head. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. 2, 3, 6, 31, 33

BBd+23a. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. FAEST. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. 2, 3, 6, 39

BBD+23b. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 581–615. Springer, Heidelberg, August 2023. 39, 42

BBSS18. Matilda Backendal, Mihir Bellare, Jessica Sorrell, and Jiahao Sun. The Fiat-Shamir zoo: Relating the security of different signature. In Nils Gruschka, editor, *NordSec 2018*, volume 11252 of *LNCS*, pages 154–170. Springer, Heidelberg, 2018. See also https://eprint.iacr.org/2018/775. 16, 17

BCJZ21. Jacqueline Brendel, Cas Cremers, Dennis Jackson, and Mang Zhao. The provable security of Ed25519: Theory and practice. In *2021 IEEE Symposium on Security and Privacy*, pages 1659–1676. IEEE Computer Society Press, May 2021. 2

BKPV23. Luk Bettale, Delaram Kahrobaei, Ludovic Perret, and Javier Verbel. Biscuit. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. 2, 3, 6, 18

BS07. Mihir Bellare and Sarah Shoup. Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 201–216. Springer, Heidelberg, April 2007. 2

BWM99. Simon Blake-Wilson and Alfred Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In Hideki Imai and Yuliang Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 154–170. Springer, Heidelberg, March 1999. 2

CDF+21. Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In *2021 IEEE Symposium on Security and Privacy*, pages 1696–1714. IEEE Computer Society Press, May 2021. 2, 3, 5, 8

CHK04. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer, Heidelberg, May 2004. 2

CHR+16. Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. From 5-pass MQ-based identification to MQ-based signatures. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 135–165. Springer, Heidelberg, December 2016. 3, 11, 12

DDN00. Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. 2

DF17. Yevgeniy Dodis and Dario Fiore. Unilaterally-authenticated key exchange. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 542–560. Springer, Heidelberg, April 2017. 2

DFH23. Jelle Don, Serge Fehr, and Patrick Huang, Yu-Hsuan Struck. On the (in)security of the BUFF transform. Cryptology ePrint Archive, Report 2023/1634, 2023. To appear at CRYPTO 2024. See https://eprint.iacr.org/2023/1634. 8, 26

DFH+24. Jelle Don, Serge Fehr, Yu-Hsuan Huang, Jyun-Jie Liao, and Patrick Struck. Hide-and-seek and the non-resignability of the BUFF transform. Cryptology ePrint Archive, Report 2024/793, 2024. https://eprint.iacr.org/2024/793. 8, 26

DFM20. Jelle Don, Serge Fehr, and Christian Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 602–631. Springer, Heidelberg, August 2020. 2, 4, 9, 11, 31

DFPS23. Julien Devevey, Pouria Fallahpour, Alain Passelègue, and Damien Stehlé. A detailed analysis of Fiat-Shamir with aborts. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 327–357. Springer, Heidelberg, August 2023. 2

DGJL21. Denis Diemert, Kai Gellert, Tibor Jager, and Lin Lyu. Digital signatures with memory-tight security in the multi-challenge setting. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 403–433. Springer, Heidelberg, December 2021. 2

DGV+16. Özgür Dagdelen, David Galindo, Pascal Véron, Sidi Mohamed El Yousfi Alaoui, and Pierre-Louis Cayrel. Extended security arguments for signature schemes. *Des. Codes Cryptogr.*, 78:441–461, 2016. 3, 11, 12

DHSY24. Sanjay Deshpande, James Howe, Jakub Szefer, and Dongze Yue. SDitH in hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(2):215–251, Mar. 2024. 29

EDV+12. Sidi Mohamed El Yousfi Alaoui, Özgür Dagdelen, Pascal Véron, David Galindo, and Pierre-Louis Cayrel. Extended security arguments for signature schemes. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *AFRICACRYPT 12*, volume 7374 of *LNCS*, pages 19–34. Springer, Heidelberg, July 2012. 3, 11, 12

EGM90. Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital schemes. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 263–275. Springer, Heidelberg, August 1990. 29

FJR22. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 541–572. Springer, Heidelberg, August 2022. 18, 29, 31, 33, 35, 37

FR23. Thibauld Feneuil and Matthieu Rivain. MQOM — MQ on my Mind. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. 2, 3, 6, 38

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. 2, 3

GHHM21. Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Tight adaptive reprogramming in the QROM. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part I*, volume 13090 of *LNCS*, pages 637–667. Springer, Heidelberg, December 2021. 3, 4, 5, 6, 9, 13, 15, 38

HJMN24. Andreas Hülsing, David Joseph, Christian Majenz, and Anand Kumar Narayanan. On the (in)security of the BUFF transform. Cryptology ePrint Archive, Report 2024/904, 2024. To appear at CRYPTO 2024. See https://eprint.iacr.org/2024/904. 5

HWZ07. Qiong Huang, Duncan S. Wong, and Yiming Zhao. Generic transformation to strongly unforgeable signatures. In Jonathan Katz and Moti Yung, editors, *ACNS 07*, volume 4521 of *LNCS*, pages 1–17. Springer, Heidelberg, June 2007. 2

IKOS07. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007. 2

JCCS19. Dennis Jackson, Cas Cremers, Katriel Cohn-Gordon, and Ralf Sasse. Seems legit: Automated analysis of subtle attacks on protocols that use signatures. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2165–2180. ACM Press, November 2019. 2

KCC+23. Seongkwang Kim, Jihoon Cho, Mingyu Cho, Jincheol Ha, Jihoon Kwon, Byeonghak Lee, Joohee Lee, Jooyoung Lee, Sangyub Lee, Dukjae Moon, Mincheol Son, and Hyojin Yoon. AIMer. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. 2, 3, 6, 37

KLS18. Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 552–586. Springer, Heidelberg, April / May 2018. 2, 4, 9, 38

KY03. Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 110–125. Springer, Heidelberg, August 2003. 2

KZ22. Daniel Kales and Greg Zaverucha. Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures. Cryptology ePrint Archive, Report 2022/588, 2022. https://eprint.iacr.org/2022/588. 37

Lam79. Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979. 2

Lyu09. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Heidelberg, December 2009. 2

MS04. Alfred Menezes and Nigel P. Smart. Security of signature schemes in a multi-user setting. *DCC*, 33(3):261–274, 2004. 2

NIS22. NIST. Call for additional digital signature schemes for the post-quantum cryptography standardization process, October 2022. 2, 3, 20

PS05. Thomas Pornin and Julien P. Stern. Digital signatures do not guarantee exclusive ownership. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 138–150. Springer, Heidelberg, June 2005. 2, 5

Roy22. Lawrence Roy. SoftSpokenOT: Quieter OT extension from small-field silent VOLE in the minicrypt model. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 657–687. Springer, Heidelberg, August 2022. 42

SCH+17. Simona Samardjiska, Ming-Shing Chen, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. MQDSS. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions. 2

SCH+19. Simona Samardjiska, Ming-Shing Chen, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. MQDSS. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions. 4

SPMS02. Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 93–110. Springer, Heidelberg, August 2002. 2

SSH11. Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. Public-key identification schemes based on multivariate quadratic polynomials. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 706–723. Springer, Heidelberg, August 2011. 2, 4, 11

Xag24. Keita Xagawa. Signatures with memory-tight security in the quantum random oracle model. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 30–58. Springer, 2024. See also https://eprint.iacr.org/2023/. 2

YSWW21. Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2986–3001. ACM Press, November 2021. 39, 42

Zha12. Mark Zhandry. How to construct quantum random functions. In *53rd FOCS*, pages 679–687. IEEE Computer Society Press, October 2012. 6

Zha15. Mark Zhandry. A note on the quantum collision and set equality problems. *Quantum Info. Comput.*, 15(7–8):557–567, May 2015. 6

A Missing Definitions and Proofs

A.1 Missing Definitions

Definition A.1 (Pseudorandom Functions (PRF)). *We say that PRF* $\mathsf{PRF}\colon \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^{p(\kappa)}$ *is secure if for any QPT adversary* $\mathcal{A}$*, its advantage*

$$\left|\Pr[\mathcal{A}^{|\mathsf{RF}(\cdot)\rangle}(1^\kappa)=1] - \Pr_{\mathsf{seed}\leftarrow\{0,1\}^\kappa}[\mathcal{A}^{|\mathsf{PRF}(\mathsf{seed},\cdot)\rangle}(1^\kappa)=1]\right|$$

is negligible in the security parameter, where $\mathsf{RF}\colon \{0,1\}^* \to \{0,1\}^{p(\kappa)}$ *is a random function.*

Definition A.2 (Pseudorandom Generator (PRG)). *We say that pseudo-random generator* $\mathsf{PRG}\colon \{0,1\}^\kappa \to \{0,1\}^{p(\kappa)}$ *is secure if for any QPT adversary* $\mathcal{A}$*, its advantage*

$$\left|\Pr_{s\leftarrow\{0,1\}^{p(\kappa)}}[\mathcal{A}(s)=1] - \Pr_{\mathsf{seed}\leftarrow\{0,1\}^\kappa}[\mathcal{A}(\mathsf{PRG}(\mathsf{seed}))=1]\right|$$

is negligible in the security parameter.

Definition A.3 (Tree PRG). *A tree PRG scheme consists of the follwoing three DPT algorithms, where an auxiliary information* aux *can be placed:*

- $\mathsf{TreePRG}(\mathsf{seed},\mathsf{aux}) \to (r_1,\dots,r_N)$*: the tree-PRG algorithm takes* $\mathsf{seed} \in \{0,1\}^\kappa$ *as input and outputs* $(r_1,\dots,r_N) \in (\{0,1\}^{p(\kappa)})^N$.
- $\mathsf{GetPath}(i^*,\mathsf{seed},\mathsf{aux}) \to \mathsf{path}$*: the path finding algorithm takes index* $i^* \in [N]$ *and* $\mathsf{seed} \in \{0,1\}^\kappa$ *as input and outputs a path information* path.
- $\mathsf{Reconst}(i^*,\mathsf{path},\mathsf{aux}) \to (r_i)_{i\neq i^*}$*: the reconstruction algorithm takes* path *and index* i^* *as input and outputs* $(r_i)_{i\neq i^*} \in (\{0,1\}^{p(\kappa)})^{N-1}$.

For correctness, we require that for any $\mathsf{seed} \in \{0,1\}^\kappa$, $i^* \in [N]$*, and* $\mathsf{aux} \in \{0,1\}^*$*, we have* $r_i = r'_i$ *for all* $i \neq i^*$*, where* $(r_i)_{i\in[N]} := \mathsf{TreePRG}(\mathsf{seed},\mathsf{aux})$, $\mathsf{path} := \mathsf{GetPath}(i^*,\mathsf{seed},\mathsf{aux})$*, and* $(r'_i)_{i\neq i^*} := \mathsf{Reconstruct}(i^*,\mathsf{path},\mathsf{aux})$.
We say that a tree PRG scheme is secure *if for any QPT adversary* $\mathcal{A}$*, for any* $i^* \in [N]$*, (and for any* $\mathsf{aux} \in \{0,1\}^*$*,) its advantage*

$$\left| \begin{array}{l} \Pr\left[\begin{array}{c} \mathsf{seed} \leftarrow \{0,1\}^\kappa, (r_i)_{i\in[N]} := \mathsf{TreePRG}(\mathsf{seed},\mathsf{aux}) : \\ \mathcal{A}((r_i)_{i\neq i^*}, r_{i^*}, \mathsf{GetPath}(i^*,\mathsf{seed},\mathsf{aux})) = 1 \end{array}\right] \\ -\Pr\left[\begin{array}{c} \mathsf{seed} \leftarrow \{0,1\}^\kappa, (r_i)_{i\in[N]} := \mathsf{TreePRG}(\mathsf{seed},\mathsf{aux}), s \leftarrow \{0,1\}^{p(\kappa)} : \\ \mathcal{A}((r_i)_{i\neq i^*}, s, \mathsf{GetPath}(i^*,\mathsf{seed},\mathsf{aux})) = 1 \end{array}\right] \end{array} \right|$$

is negligible in the security parameter.

Definition A.4 (Collision-resistance of Reconst**).** *We say that* Reconst *is* collision-resistant *if for any QPT adversary* $\mathcal{A}$, $i^* \in [N]$*, and* $\mathsf{aux} \in \{0,1\}^*$*, its advantage*

$$\Pr\left[\begin{array}{c} (\mathsf{path},\mathsf{path}') \leftarrow \mathcal{A}(1^\kappa) : \\ \mathsf{path} \neq \mathsf{path}' \wedge \mathsf{Reconst}(i^*,\mathsf{path},\mathsf{aux}) = \mathsf{Reconst}(i^*,\mathsf{path}',\mathsf{aux}) \end{array}\right]$$

is negligible in κ.

1: $\underline{\mathsf{G}_0, \mathsf{G}_T, \mathsf{G}_F}$
2: $(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$; $\mathcal{Q} := \emptyset$
3: $(\mu^*, (h_1^*, \ldots, h_n^*, a_{n+1}^*)) \leftarrow \mathcal{A}^{\textsc{Sign}, |\mathsf{H}\rangle, |\gamma\rangle}(vk)$
4: **if** $(\mu^*, (h_1^*, \ldots, h_n^*, a_{n+1}^*)) \in \mathcal{Q}$ **then**
5: $\quad$ **return** false
6: **for** $i \in [n]$: $c_i^* := \gamma_i(h_i^*)$
7: $(a_1^*, \ldots, a_n^*) := \mathsf{Rep}(vk, c_1^*, \ldots, c_n^*, a_{n+1}^*)$
8: **if** $(a_1^*, \ldots, a_n^*) = \bot$ **then return** false
9: $h_0^* := \emptyset$
10: **for** $i = 1, \ldots, n$: $\bar{h}_i := \mathsf{H}(\mathsf{aux}_i^*, h_{i-1}^*, a_i^*)$
11: $d := \mathsf{V}(vk, a_1^*, c_1^*, \ldots, a_n^*, c_n^*, a_{n+1}^*)$ $\quad$ //G_T, G_F
12: **return** $\mathsf{boole}(\forall i \in [n] : h_i^* = \bar{h}_i)$ $\quad$ //G_0
13: **return** $d \land \mathsf{boole}(\forall i \in [n] : h_i^* = \bar{h}_i)$ $\quad$ //G_T
14: **return** $\neg d \land \mathsf{boole}(\forall i \in [n] : h_i^* = \bar{h}_i)$ $\quad$ //G_F

$\underline{\textsc{Sign}(\mu)}$
$h_0 := \emptyset$; $\mathsf{state} := \emptyset$
for $i = 1, \ldots, n$ **do**
$\quad (a_i, \mathsf{state}) \leftarrow \mathsf{P}(sk, c_{i-1}, \mathsf{state})$
$\quad h_i := \mathsf{H}(\mathsf{aux}_i, h_{i-1}, a_i)$
$\quad c_i := \gamma_i(h_i)$
$a_{n+1} \leftarrow \mathsf{P}(sk, c_n, \mathsf{state})$
$\mathcal{Q} := \mathcal{Q} \cup \{(\mu, (h_1, \ldots, h_n, a_{n+1}))\}$
return $\sigma := (h_1, \ldots, h_n, a_{n+1})$

$\underline{\mathcal{A}_{\mathsf{FS}_{\mathrm{cmt}}}^{\textsc{Sign}', |\mathsf{H}\rangle, |\gamma\rangle}(vk) \text{ against } \mathsf{FS}_{\mathrm{cmt}}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]}$
$(\mu^*, (h_1^*, \ldots, h_n^*, a_{n+1}^*)) \leftarrow \mathcal{A}^{\textsc{Sign}, |\mathsf{H}\rangle, |\gamma\rangle}(vk)$
for $i \in [n]$: $c_i^* := \gamma_i(h_i^*)$
$(a_1^*, \ldots, a_n^*) := \mathsf{Rep}(vk, c_1^*, \ldots, c_n^*, a_{n+1}^*)$
return $(\mu^*, (a_1^*, \ldots, a_n^*, a_{n+1}^*))$

$\underline{\mathcal{A}_{\mathsf{FS}_{\mathrm{cmt}}}\text{'s simulation of } \textsc{Sign}(\mu)}$
$(a_1, \ldots, a_n, a_{n+1}) \leftarrow \textsc{Sign}'(\mu)$
$h_0 := \emptyset$
for $i = 1, \ldots, n$ **do**
$\quad h_i := \mathsf{H}(\mathsf{aux}_i, h_{i-1}, a_i)$
return $\sigma := (h_1, \ldots, h_n, a_{n+1})$

Fig. 11. Games G_0, G_T, and G_F and an adversary $\mathcal{A}_{\mathsf{FS}_{\mathrm{cmt}}}$ for sEUF-CMA security proof of $\mathsf{FS_h}$.

Definition A.5 (Commitment). *We say that a commitment scheme* $\mathsf{Com} : \{0,1\}^* \times \{0,1\}^\kappa \to \{0,1\}^\kappa$ *is*

- non-invertible *if for any QPT adversary $\mathcal{A}$, its advantage*
$$\Pr[\mathsf{com} \leftarrow \{0,1\}^\kappa, (x, \rho) \leftarrow \mathcal{A}(\mathsf{com}) : \mathsf{Com}(x; \rho) = \mathsf{com}]$$
is negligible in the security parameter;
- binding *if for any QPT adversary $\mathcal{A}$, its advantage*
$$\Pr[(x, \rho, x', \rho') \leftarrow \mathcal{A}(1^\kappa) : x \neq x' \land \mathsf{Com}(x; \rho) = \mathsf{Com}(x'; \rho')]$$
is negligible in the security parameter;
- collision-resistant *if for any QPT adversary $\mathcal{A}$, its advantage*
$$\Pr[(x, \rho, x', \rho') \leftarrow \mathcal{A}(1^\kappa) : (x, \rho) \neq (x', \rho') \land \mathsf{Com}(x; \rho) = \mathsf{Com}(x'; \rho')]$$
is negligible in the security parameter;
- hiding *if for any QPT adversary $\mathcal{A}$ and for any $x \in \{0,1\}^*$, its advantage*
$$\left| \Pr_{\mathsf{com} \leftarrow \{0,1\}^\kappa}[\mathcal{A}(\mathsf{com}) = 1] - \Pr_{\rho \leftarrow \{0,1\}^\kappa}[\mathcal{A}(\mathsf{Com}(x; \rho)) = 1] \right|$$
is negligible in the security parameter.

A.2 (Strong) Existential Unforgeability of $\mathsf{FS_h}$

Proof (Proof of Theorem 5.1). We only consider sEUF-CMA security since the proof for EUF-CMA security is essentially the same.

We consider the following games:

- G_0: This is the original sEUF-CMA game as in Figure 11. The challenger checks if $h_i^* = \bar{h}_i$ for all $i \in [n]$ (See L.12).
- G_T: In this game, the challenger checks if $h_i^* = \bar{h}_i$ for all $i \in [n]$ and $\mathsf{V}(vk, a_1^*, c_1^*, \ldots, a_{n+1}^*) = \mathsf{true}$ (See L.13).
- G_F: In this game, the challenger checks if $h_i^* = \bar{h}_i$ for all $i \in [n]$ and $\mathsf{V}(vk, a_1^*, c_1^*, \ldots, a_{n+1}^*) = \mathsf{false}$ (See L.14).

Apparently, we have $\mathsf{Adv}^{\text{seuf-cma}}_{\mathsf{FS_h}[\mathsf{ID},\mathsf{H},\gamma]}(1^\kappa) = \Pr[W_0] \le \Pr[W_T] + \Pr[W_F]$.
On G_T, we can construct an adversary $\mathcal{A}_{\mathsf{FS_{cmt}}}$ against $\mathsf{FS_{cmt}}[\mathsf{ID},\mathsf{H},\gamma]$ that simulates Sign as in Figure 11.
We argue that if $\mathcal{A}$'s output $(\mu^*, (h_1^*, \dots, h_n^*, a_{n+1}^*))$ is fresh then $\mathcal{A}_{\mathsf{FS_{cmt}}}$'s output $(\mu^*, (a_1^*, \dots, a_{n+1}^*))$ is also fresh. Suppose that $\mathcal{A}_{\mathsf{FS_{cmt}}}$'s output $(\mu^*, (a_1^*, \dots, a_{n+1}^*))$, which is produced from $\mathcal{A}$'s output $(\mu^*, (h_1^*, \dots, h_n^*, a_{n+1}^*))$, is in the list. This means that μ^* is queried by $\mathcal{A}$, $\mathcal{A}_{\mathsf{FS_{cmt}}}$ receives $(a_1^*, \dots, a_{n+1}^*)$ from its sigining oracle Sign$'$, $\mathcal{A}_{\mathsf{FS_{cmt}}}$ computes $h_i^* := \mathsf{H}(\mathsf{aux}_i, h_{i-1}^*, a_i^*)$ for $i = 1, \dots, n$, and returns $(h_1^*, \dots, h_n^*, a_{n+1}^*)$ to $\mathcal{A}$. Thus, $\mathcal{A}$'s output $(\mu^*, (h_1^*, \dots, h_n^*, a_{n+1}^*))$ also should be in the list.
Hence, if $\mathcal{A}$ wins, then $\mathcal{A}_{\mathsf{FS_{cmt}}}$ also wins. We have

$$\Pr[W_T] \le \mathsf{Adv}^{\text{seuf-cma}}_{\mathsf{FS_{cmt}}[\mathsf{ID},\mathsf{H},\gamma],\mathcal{A}_{\mathsf{FS_{cmt}}}}(1^\kappa).$$

On G_F, if non-$\perp$ $(a_1^*, \dots, a_n^*)$ is produced by Rep in L.7, then $\mathsf{V}(vk, a_1^*, c_1^*, \dots, c_n^*, a_{n+1}^*)$ should be true since ID is computationally sound. In other words, if it is violated, we can construct an adversary $\mathcal{A}_{\mathrm{snd}}$ against ID by using $\mathcal{A}$ against $\mathsf{FS_h}$ such that

$$\Pr[W_F] \le \mathsf{Adv}^{\text{sound}}_{\mathsf{ID},\gamma,\mathcal{A}_{\mathrm{snd}}}(1^\kappa).$$

This completes the proof. □

A.3 Weak Non-Resignability of $\mathsf{FS_h}$

In this subsection, we show Lemma 5.3.
In order to treat multi-point reprogramming, we review the one-way-to-hiding (O2H) lemma in [AHU19, Thm.3] stated as follows:

Lemma A.1 (One-way-to-Hiding Lemma, Revisited [AHU19, Thm.3], adapted). *Let $\mathcal{S} \subseteq \mathcal{X}$ be random. Let $G, H \colon \mathcal{X} \to \mathcal{Y}$ be random functions satisfying $\forall x \notin \mathcal{S}$, $G(x) = H(x)$. Let z be a random string. Note that $\mathcal{S}, G, H, z$ may have arbitrary joint distribution.*
Let $\mathcal{A}$ be a q-query oracle algorithm. Let $\mathcal{B}^{|G\rangle}$ be an algorithm that on input z chooses $i \leftarrow [q]$, runs $\mathcal{A}^{|G\rangle}(z)$ until the i-th query, then measure all query input registers in the computational basis and outputs an element $s \in \mathcal{X}$ of measurement outcomes. Let

$$P_l := \Pr[b \leftarrow \mathcal{A}^{|H\rangle}(z) : b = 1],$$
$$P_r := \Pr[b \leftarrow \mathcal{A}^{|G\rangle}(z) : b = 1],$$
$$P_g := \Pr[s \leftarrow \mathcal{B}^{|G\rangle}(z) : s \in \mathcal{S}].$$

Then, we have

$$|P_l - P_r| \le 2q\sqrt{P_g} \text{ and } \left|\sqrt{P_l} - \sqrt{P_r}\right| \le 2q\sqrt{P_g}.$$

If z and $\mathcal{S}$ are independent, the bound can be $4q \cdot \max_{x \in \mathcal{X}} \Pr[x \in \mathcal{S}]$. But, in our context, z and $\mathcal{S}$ are correlated. Don, Fehr, Huang, and Struck [DFH23] showed that the BUFF conversion with *salt* ($-BUFF),[12] satisfies their revised non-resignability in the (Q)ROM, where the adversary is given auxiliary information $\mathsf{AUX}(\mu, vk)$ independent of H whose statistical entropy is sufficiently high. The proof below can be considered as a simplified version of their QROM proof adapted to the case for $\mathsf{FS_h}$ *without salt*. Very recently, Don, Fehr, Huang, Liao, and Struck [DFH+24] showed that the standard BUFF conversion is enough in the QROM for somewhat stronger non-resignability where the adversary can get $\mathsf{AUX}(\mu, sk)$ whose computational entropy is sufficiently high.

Proof (Proof of Lemma 5.3). We consider the following games defined in Figure 12 and Figure 13.

- G_0: This is the original wNR security game. $\mathcal{A}$ is given vk and σ, which is produced on a message $\mu \leftarrow \mathcal{M}$, and outputs $vk' \ne vk$ and σ'. If $\mathsf{Vrfy}_{\mathsf{h}}^{\mathsf{H},\gamma}(vk', \mu, \sigma') = \mathsf{true}$, then the adversary wins.
- G_1: In this game, we introduce a collision-check procedure for aux as follows: Receiving $vk' \ne vk$ and σ', the challenger computes $\mathsf{aux}'_\lambda := \mathsf{aux}(\lambda, vk', \mu)$. If $\mathsf{aux}_\lambda = \mathsf{aux}'_\lambda$, then the adversary loses. This modification is justified by the collision-resistance property of aux with respect to the verification key on index λ.
- G_2: In this game, we introduce a collision-check procedure for H as follows: Receiving $vk' \ne vk$ and σ', the challenger checks if $\mathsf{H}(\mathsf{aux}_j, h_{j-1}, a_j) = \mathsf{H}(\mathsf{aux}'_j, h'_{j-1}, a'_j)$ while $(\mathsf{aux}_j, h_{j-1}, a_j) \ne (\mathsf{aux}'_j, h'_{j-1}, a'_j)$ for some $j \in [\lambda, n]$, where $\mathsf{aux}'_j, h'_j, a'_j$ are values in the verification of vk', μ, σ'. If such a pair is found, then the adversary loses. This modification is justified by the collision-resistance property of H.
  Notice that the adversary should output vk' and σ' such that $(\mathsf{aux}_j, h_{j-1}) \ne (\mathsf{aux}'_j, h'_{j-1})$ for all $j \in [\lambda, n]$. Let ζ be a minimum index in $[\lambda, n]$ such that $\mathsf{aux}_\zeta = (\mu, \eta_\zeta)$. Now, H should be asked at least one point $(\mu, \eta'_\zeta, h'_{\zeta-1}, a'_\zeta)$ to compute h'_ζ in the verification of (vk', μ, σ'), while this point is not asked in the signing/verification of (vk, μ, σ).

[12] The signer first chooses salt salt, computes $y = F(vk, \mu, \mathsf{salt})$, and generates a signature $\sigma \leftarrow \mathsf{Sign}(sk, \mu)$, and outputs $(\sigma, y, \mathsf{salt})$, where F is the random oracle.

$\underline{\mathsf{G}_0}$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$\mu \leftarrow \mathcal{MS}$
$\sigma \leftarrow \mathsf{Sign}_\mathrm{h}^{\mathsf{H},\gamma}(sk, \mu)$
$(\sigma', vk') \leftarrow \mathcal{A}^{|\mathsf{H}\rangle,|\gamma\rangle}(vk, \sigma)$
if $vk = vk'$ **then return** false
return $\mathsf{Vrfy}_\mathrm{h}^{\mathsf{H},\gamma}(vk', \mu, \sigma')$

$\underline{\mathsf{Sign}_\mathrm{h}^{\mathsf{H},\gamma}(sk, \mu)}$
$h_0 := \emptyset$; $\mathsf{state} := \emptyset$
for $i = 1, \dots, n$ **do**
  $(a_i, \mathsf{state}) \leftarrow \mathsf{P}(sk, c_{i-1}, \mathsf{state})$
  $h_i := \mathsf{H}(\mathsf{aux}_i, h_{i-1}, a_i)$
  $c_i := \gamma_i(h_i)$
$a_{n+1} \leftarrow \mathsf{P}(sk, c_n, \mathsf{state})$
return $\sigma := (h_1, \dots, h_n, a_{n+1})$

$\underline{\mathsf{G}_1 \text{ and } \mathsf{G}_2}$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$\mu \leftarrow \mathcal{MS}$
$\sigma \leftarrow \mathsf{Sign}_\mathrm{h}^{\mathsf{H},\gamma}(sk, \mu)$
$(\sigma', vk') \leftarrow \mathcal{A}^{|\mathsf{H}\rangle,|\gamma\rangle}(vk, \sigma)$
if $vk = vk'$ **then return** false
parse $\sigma' = (h'_1, \dots, h'_n, a'_{n+1})$
for $i \in [n]$: $c'_i := \gamma_i(h'_i)$
$(a'_1, \dots, a'_n) := \mathsf{Rep}(vk, c'_1, \dots, c'_n, a'_{n+1})$
if $(a'_1, \dots, a'_n) = \bot$ **then return** false
if $\mathsf{aux}_\lambda = \mathsf{aux}'_\lambda$ **then return** false //G_1-
$h'_0 := \emptyset$
for $i \in [n]$: $\bar{h}_i := \mathsf{H}(\mathsf{aux}'_i, h'_{i-1}, a'_i)$
for $i \in [\lambda, n]$ **do** //G_2-
  if $(\mathsf{aux}_i, h_{i-1}, a_i) \neq (\mathsf{aux}'_i, h'_{i-1}, a'_i)$ **and** $h_i = \bar{h}_i$ **then**
    return false //G_2-
return $\mathsf{boole}(\forall i \in [n] : h'_i = \bar{h}_i)$

Fig. 12. Games G_0, G_1, and G_2 for the wNR security proof of FS_h.

$\underline{\mathsf{G}_3}$
$\mathsf{H} \leftarrow \mathsf{Func}(\{0,1\}^*, \mathcal{H})$
$\mathsf{G} := \mathsf{H}$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$\mu \leftarrow \mathcal{MS}$
$\sigma \leftarrow \mathsf{Sign}_\mathsf{h}^{\mathsf{H},\gamma}(sk, \mu)$
$\mathcal{S} := \{(\mathsf{aux}_i, h_{i-1}, a_i) : \mathsf{aux}_i \text{ contains } \mu\}$
for $(\mathsf{aux}_i, h_{i-1}, a_i) \in \mathcal{S}$ **do**
  $\hat{h}_i \leftarrow \mathcal{H}$
  $\mathsf{G} := \mathsf{G}[(\mathsf{aux}_i, h_{i-1}, a_i) \mapsto \hat{h}_i]$
$(\sigma', vk') \leftarrow \mathcal{A}^{|\mathsf{G}\rangle,|\gamma\rangle}(vk, \sigma)$
if $vk = vk'$ **then return** false
parse $\sigma' = (h'_1, \ldots, h'_n, a'_{n+1})$
for $i \in [n]$: $c'_i := \gamma_i(h'_i)$
$(a'_1, \ldots, a'_n) := \mathsf{Rep}(vk, c'_1, \ldots, c'_n, a'_{n+1})$
if $(a'_1, \ldots, a'_n) = \bot$ **then return** false
if $\mathsf{aux}_\lambda = \mathsf{aux}'_\lambda$ **then return** false
$h'_0 := \varnothing$
for $i \in [n]$: $\bar{h}_i := \mathsf{H}(\mathsf{aux}'_i, h'_{i-1}, a'_i)$
for $i \in [\lambda, n]$ **do**
  if $(\mathsf{aux}_i, h_{i-1}, a_i) \neq (\mathsf{aux}'_i, h'_{i-1}, a'_i)$ **and** $h_i = \bar{h}_i$ **then**
    return false
return $\mathsf{boole}(\forall i \in [n] : h'_i = \bar{h}_i)$

$\underline{\mathsf{G}_4}$
$\mathsf{H} \leftarrow \mathsf{Func}(\{0,1\}^*, \mathcal{H})$
$\mathsf{G} := \mathsf{H}$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$\mu \leftarrow \mathcal{MS}$
$\sigma \leftarrow \mathsf{Sign}_\mathsf{h}^{\mathsf{H},\gamma}(sk, \mu)$
$\mathcal{S} := \{(\mu, \cdot)\}$
for $(\mu, x) \in \mathcal{S}$ **do**
  $\mathsf{G} := \mathsf{G}[(\mu, x) \mapsto \bot]$
$(\sigma', vk') \leftarrow \mathcal{A}^{|\mathsf{G}\rangle,|\gamma\rangle}(vk, \sigma)$
if $vk = vk'$ **then return** false
parse $\sigma' = (h'_1, \ldots, h'_n, a'_{n+1})$
for $i \in [n]$: $c'_i := \gamma_i(h'_i)$
$(a'_1, \ldots, a'_n) := \mathsf{Rep}(vk, c'_1, \ldots, c'_n, a'_{n+1})$
if $(a'_1, \ldots, a'_n) = \bot$ **then return** false
if $\mathsf{aux}_\lambda = \mathsf{aux}'_\lambda$ **then return** false
$h'_0 := \varnothing$
for $i \in [n]$: $\bar{h}_i := \mathsf{H}(\mathsf{aux}'_i, h'_{i-1}, a'_i)$
for $i \in [\lambda, n]$ **do**
  if $(\mathsf{aux}_i, h_{i-1}, a_i) \neq (\mathsf{aux}'_i, h'_{i-1}, a'_i)$ **and** $h_i = \bar{h}_i$ **then**
    return false
return $\mathsf{boole}(\forall i \in [n] : h'_i = \bar{h}_i)$

$\underline{\mathsf{G}_{g,3}}$
$\mathsf{H} \leftarrow \mathsf{Func}(\{0,1\}^*, \mathcal{H})$
$\mathsf{G} := \mathsf{H}$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$\mu \leftarrow \mathcal{MS}$
$\sigma \leftarrow \mathsf{Sign}_\mathsf{h}^{\mathsf{H},\gamma}(sk, \mu)$
$\mathcal{S} := \{(\mathsf{aux}_i, h_{i-1}, a_i) : \mathsf{aux}_i \text{ contains } \mu\}$
for $(\mathsf{aux}_i, h_{i-1}, a_i) \in \mathcal{S}$ **do**
  $\hat{h}_i \leftarrow \mathcal{H}$
  $\mathsf{G} := \mathsf{G}[(\mathsf{aux}_i, h_{i-1}, a_i) \mapsto \hat{h}_i]$
$z := (vk, \sigma)$
$s \leftarrow \mathcal{B}^{|\mathsf{G}\rangle,|\gamma\rangle}(vk, \sigma)$
return $\mathsf{boole}(s \in \mathcal{S})$

$\underline{\mathsf{G}_{g,4}}$
$\mathsf{F} \leftarrow \mathsf{Func}(\{0,1\}^*, \mathcal{H})$
$\mathsf{H} := \mathsf{F}$
$(vk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
$\mu \leftarrow \mathcal{MS}$
$\sigma \leftarrow \mathsf{Sign}_\mathsf{h}^{\mathsf{F},\gamma}(sk, \mu)$
$\mathcal{S}_3 := \{(\mathsf{aux}_i, h_{i-1}, a_i) : \mathsf{aux}_i \text{ contains } \mu\}$
for $(\mathsf{aux}_i, h_{i-1}, a_i) \in \mathcal{S}_3$ **do**
  $\hat{h}_i \leftarrow \mathcal{H}$
  $\mathsf{H} := \mathsf{H}[(\mathsf{aux}_i, h_{i-1}, a_i) \mapsto \hat{h}_i]$
$\mathsf{G} := \mathsf{H}$
$\mathcal{S} := \{(\mu, \cdot)\}$
for $(\mu, x) \in \mathcal{S}$ **do**
  $\mathsf{G} := \mathsf{G}[(\mu, x) \mapsto \bot]$
$z := (vk, \sigma)$
$s \leftarrow \mathcal{B}^{|\mathsf{G}\rangle,|\gamma\rangle}(vk, \sigma)$
return $\mathsf{boole}(s \in \mathcal{S})$

Fig. 13. Games G_3 and G_4 for the wNR security proof of FS_h.

– G_3: In this game, after obtaining $\sigma = (h_1, \dots, h_n, a_{n+1})$, we reprogram the points related to μ with random values.

  Due to the O2H theorem (Lemma A.1), the difference between the two games G_2 and G_3 is upper-bounded by $2q\sqrt{\Pr[\mathsf{G}_{g,3} \Rightarrow 1]}$, where $\mathsf{G}_{g,3}$ is defined in Figure 13.

  Notice that the problem $\mathsf{G}_{g,3}$ in our context is boiled down to an unstructured database search since $\mathcal{B}$ is given *no* information of $\mathsf{G}(\mu, \cdot)$ via $z = (vk, \sigma)$. Therefore, the probability $\Pr[\mathsf{G}_{g,3} \Rightarrow 1]$ is at most $1/|\mathcal{M}|$.

– G_4: Next, the challenger gives a filtered random oracle H', which returns $\perp$ if the input is $(\mu, \cdot)$ to the adversary. Notice that in this game, the adversary has no information of the hash value $\mathsf{H}(\mu, \eta'_\zeta, h'_{\zeta-1}, a'_\zeta)$, while it outputs h'_ζ in the signature. Therefore, the winning probability in this game is at most $1/|\mathcal{H}|$.

  The difference between the two games G_3 and G_4 is bounded by the O2H and we have $2q\sqrt{\Pr[\mathsf{G}_{g,4} \Rightarrow 1]}$, where game $\mathsf{G}_{g,4}$ is defined in Figure 13. Again, since $\mathcal{B}$ is given *no* information of $\mathsf{G}(\mu, \cdot)$ via $z = (vk, \sigma)$, the probability $\Pr[\mathsf{G}_{g,4} \Rightarrow 1]$ is at most $1/|\mathcal{M}|$.

This completes the proof. □

A.4 Proof Sketch of HVZK Property of Biscuit

Proof (Proof sketch of Lemma 6.1). The proof in [FJR22] considered four games G_i for $i = 0, 1, 2, 3$. But, we consider seven games defined as follows:

– G_0: In this game, the adversary can obtain the transcript generated by the real prover and verifier.
– G_1: In this game, the challenger first chooses challenges c_1 and c_2 and then runs the prover using those challenges. Since ID is public-coin, this modification is conceptual.
– G_2: In this game, the prover chooses $\left(\mathsf{salt}, (\mathsf{seed}^{(e)})_{e \in [\tau]}\right)$ uniformly at random. This modification is justified by the security of PRF.
– G_3: Next, the prover chooses $\mathsf{seed}^{(e)}_{i^*_e}$ and $\rho^{(e)}_{i^*_e}$ for $e \in [\tau]$ uniformly at random. This modification is justified by the security of $\mathsf{TreePRG}$.
– G_4: Next, we make the prover choose $(\llbracket \boldsymbol{s} \rrbracket^{(e)}_{i^*_e}, \llbracket \boldsymbol{a} \rrbracket^{(e)}_{i^*_e}, \llbracket \boldsymbol{c} \rrbracket^{(e)}_{i^*_e})$ uniformly at random. This modification is justified by the security of $\mathsf{MakeShares}$ used for those shares.
– G_5: In this game, the prover is modified to choose com_{1,i^*} uniformly at random. This modification is justified by the hiding property of the commitment scheme Com.
– G_6: Next, we make the prover compute $\llbracket \boldsymbol{v} \rrbracket^{(e)}_{i^*_e} := -\sum_{i \neq i^*_e} \llbracket \boldsymbol{v} \rrbracket^{(e)}_i$. This modification is justified by the correctness of the MPCitH protocol.
– G_7: Finally, the prover chooses $\boldsymbol{\Delta s}^{(e)}$, $\boldsymbol{\Delta c}^{(e)}$, and $\llbracket \boldsymbol{\alpha} \rrbracket^{(e)}_{i^*_e}$ uniformly at random and now the modified prover is equivalent to the simulator.

  Let us show that the distributions of the output of the prover in G_6 and G_7 are equivalent: For simplicity of notation, we omit e: We note that the shares for party $i \neq i^*$ are the same in both games. However, since $\llbracket \boldsymbol{s} \rrbracket_{i^*}$, $\llbracket \boldsymbol{c} \rrbracket_{i^*}$, and $\llbracket \boldsymbol{a} \rrbracket_{i^*}$ are hidden from the adversary, they mask the distribution of $\boldsymbol{\Delta s}^{(e)}$, $\boldsymbol{\Delta c}^{(e)}$, and $\llbracket \boldsymbol{\alpha} \rrbracket_{i^*}$ in G_6. Thus, the distributions of the views from the adversary are the same in both games.

B Variant of FS_h

We notice that FAEST (in our formulation in subsection H.1) and SDitH put only the last hash value h_n in a signature; we call this transform $\mathsf{FS}_{\mathsf{h,last}}$ defined later. If $h_1, \dots, h_{n-1}$ are independent of a message and only the last h_n involves a message, then we can treat such signature schemes as *online/offline signature* [EGM90] as Deshpande, Howe, Szefer, and Yue [DHSY24] pointed out. From practical views, we can store several pre-signature values by using $\mathsf{P}_1, \dots, \mathsf{P}_n$ since $h_1, \dots, h_{n-1}$ are independent of a message and, receiving a message μ to be signed, then pick up informations to produce a_{n+1}. While this nature came from the collapsed three-pass ID protocol [AHJ+23], we can show its security without considering the collapsed one.

To eliminate $h_1, \dots, h_{n-1}$, the commitment-reproducing algolrithm Rep should be able to reproduce $a_1, \dots, a_n$ from the last challenge $c_n = \gamma_n(h_n)$ and the last message a_{n+1}. In typical MPCitH protocol, Rep can be decomposed into n algorithms as follows:

Definition B.1 (Decomposable commitment-reproducing algorithm). *Assume that there exists a commitment-reproducing alglorithm* Rep *that takes* $(vk, c_1, \dots, c_n, a_{n+1})$ *and outputs messages* $(a_1, \dots, a_n)$*, which may be* $\perp$*. We*

$\underline{\mathsf{Sign}_{\mathsf{h,last}}(sk, \mu)}$
$h_0 := \emptyset;\ c_0 := \emptyset;\ \mathsf{state} := \emptyset$
for $i = 1, \ldots, n$ **do**
  $(a_i, \mathsf{state}) \leftarrow \mathsf{P}(sk, c_{i-1}, \mathsf{state})$
  $h_i := \mathsf{H}(\mathsf{aux}_i, h_{i-1}, a_i)$
  $c_i := \gamma_i(h_i)$
$a_{n+1} \leftarrow \mathsf{P}(sk, c_n, \mathsf{state})$
return $\sigma := (h_n, a_{n+1})$

$\underline{\mathsf{Vrfy}_{\mathsf{h,last}}(vk, \mu, \sigma)}$
Parse $\sigma = (h_n, a_{n+1})$
$\hat{h}_0 := \emptyset;\ \mathsf{state} := \emptyset$
$c_n := \gamma_n(h_n)$
for $i = 1, \ldots, N$ **do**
  $(a_i, \mathsf{state}) := \mathsf{Rep}_i(vk, (a_j, c_j)_{j \in [i-1]}, c_n, a_{n+1}, \mathsf{state})$
  if $a_i = \perp$ **then**
    return $\perp$
  $\hat{h}_i := \mathsf{H}(\mathsf{aux}_i, \hat{h}_{i-1}, a_i)$
  $c_i := \gamma_i(\hat{h}_i)$
return $\mathsf{boole}(h_n = \hat{h}_i)$

Fig. 14. Scheme $\mathsf{FS}_{\mathsf{h,last}}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}] = (\mathsf{Gen}, \mathsf{Sign}_{\mathsf{h,last}}, \mathsf{Vrfy}_{\mathsf{h,last}})$, where $\mathsf{ID} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$, $\mathsf{H} : \{0,1\}^* \to \mathcal{H}$ is modeled as the random oracle, and $\gamma_i : \mathcal{H} \to C_i$ for $i \in [n]$ is also modeled as the random oracle. For ease of notation, we let $\mathsf{aux}_i = \mathsf{aux}(i, vk, \mu)$.

$\underline{\mathcal{A}_{\mathsf{FS_h}}^{\textsc{Sign}', |\mathsf{H}\rangle, |\boldsymbol{\gamma}\rangle}(vk) \text{ against } \mathsf{FS_h}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]}$
$(\mu^*, (h_n^*, a_{n+1}^*)) \leftarrow \mathcal{A}^{\textsc{Sign}, |\mathsf{H}\rangle, |\boldsymbol{\gamma}\rangle}(vk)$
$\hat{h}_0 := \emptyset;\ \mathsf{state} := \emptyset$
$c_n^* := \gamma_n(h_n^*)$
for $i = 1, \ldots, N$ **do**
  $(a_i^*, \mathsf{state}) := \mathsf{Rep}_i(vk, (a_j^*, c_j^*)_{j \in [i-1]}, c_n^*, a_{n+1}^*, \mathsf{state})$
  if $a_i^* = \perp$ **then return** false
  $\hat{h}_i^* := \mathsf{H}(\mathsf{aux}_i^*, \hat{h}_{i-1}^*, a_i^*)$
  $c_i^* := \gamma_i(\hat{h}_i^*)$
return $(\mu^*, (\hat{h}_1^*, \ldots, \hat{h}_{n-1}^*, h_n^*, a_{n+1}^*))$

$\underline{\mathcal{A}_{\mathsf{FS_h}}\text{'s simulation of } \textsc{Sign}(\mu)}$
$(h_1, \ldots, h_n, a_{n+1}) \leftarrow \textsc{Sign}'(\mu)$
return $\sigma := (h_n, a_{n+1})$

Fig. 15. An adversary $\mathcal{A}_{\mathsf{FS_h}}$ for sEUF-CMA security proof of $\mathsf{FS}_{\mathsf{h,last}}$.

say that Rep *is decomposable if there exist DPT algorithms* $\mathsf{Rep}_1, \ldots, \mathsf{Rep}_n$ *such that* Rep *is written as follows:*

$\underline{\mathsf{Rep}(vk, c_1, c_2, \ldots, c_n, a_{n+1})}$ `//Ignore` $c_1, \ldots, c_{n-1}$
$\hat{h}_0 := \emptyset;\ \mathsf{state} := \emptyset$
for $i = 1, \ldots, N$ **do**
  $(a_i, \mathsf{state}) := \mathsf{Rep}_i(vk, (a_j, c_j)_{j \in [i-1]}, c_n, a_{n+1}, \mathsf{state})$
  if $a_i = \perp$ **then**
    return $\perp$
  $\hat{h}_i := \mathsf{H}(\mathsf{aux}_i, \hat{h}_{i-1}, a_i)$
  $c_i := \gamma_i(\hat{h}_i)$ `//Overwite` c_i
return $(a_1, \ldots, a_n)$

If Rep is decomposable, then we can consider the signature scheme $\mathsf{FS}_{\mathsf{h,last}}$ as the variant of $\mathsf{FS_h}$, defined in Figure 14.
We have the following theorem:

Theorem B.1 ($\mathsf{FS_h} \Rightarrow \mathsf{FS}_{\mathsf{h,last}}$). *Suppose that* Rep *is decomposable. If* $\mathsf{FS_h}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$ *is* EUF-CMA/sEUF-CMA-*secure, then* $\mathsf{FS}_{\mathsf{h,last}}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$ *is also, respectively.*

Combined with Theorem 5.1, we obtain the following corollary.

Corollary B.1 ($\mathsf{FS}_{\mathsf{cmt}} \Rightarrow \mathsf{FS}_{\mathsf{h,last}}$). *Suppose that* ID *is computationally sound and* Rep *is decomposable. If* $\mathsf{FS}_{\mathsf{cmt}}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$ *is* EUF-CMA/sEUF-CMA-*secure, then* $\mathsf{FS}_{\mathsf{h,last}}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$ *is also, respectively.*

Proof (Proof of Theorem B.1). We only consider sEUF-CMA security since the proof for EUF-CMA security is essentially the same.

Let us consider the reduction algorithm $\mathcal{A}_{\mathsf{FS_h}}$ as in Figure 15. Apparently, the simulation of the signing oracle is perfect. We show that if $\mathcal{A}$'s output is valid for $\mathsf{FS}_{\mathrm{h,last}}$, then the output of $\mathcal{A}_{\mathsf{FS_h}}$ is also valid for $\mathsf{FS_h}$.
Let $(\mu^*, (h_n^*, a_{n+1}^*))$ be $\mathcal{A}$'s output and let $(\mu^*, (\hat{h}_1^*, \dots, \hat{h}_{n-1}^*, h_n^*, a_{n+1}^*)$ be $\mathcal{A}_{\mathsf{FS_h}}$'s output. Since $\mathcal{A}$'s output is valid, we have $h_n^* = \hat{h}_n^*$. We next check how to compute the hash values in the verification algorithm $\mathsf{Vrfy_h}$ (see Figure 7). Let $\bar{h}_1^*, \dots, \bar{h}_n^*$ be hash values computed in L.7 of the verification algorithm $\mathsf{Vrfy_h}$ on input vk, μ^*, and $\sigma^* = (\hat{h}_1^*, \dots, \hat{h}_{n-1}^*, \hat{h}_n^*, a_{n+1}^*)$. To compute them by Rep, we first compute $\hat{c}_n = \gamma_i(\hat{h}_n)$; We then compute for $i = 1, \dots, n$, $(a_i^*, \mathsf{state}) := \mathsf{Rep}_i(vk, (a_j^*, \hat{c}_j^*)_{j \in [i-1]}, \hat{c}_n^*, a_{n+1}^*, \mathsf{state})$ (and reject if $a_i^* = \bot$), $\hat{h}_i^* := \mathsf{H}(\mathsf{aux}_i^*, \hat{h}_{i-1}^*, a_i^*)$, and $c_i^* := \gamma_i(\hat{h}_i^*)$; After the recomputation of $a_1^*, \dots, a_n^*$ by this procedure, we compute $\bar{h}_i^*$ as $\mathsf{H}(\mathsf{aux}_i^*, \hat{h}_{i-1}^*, a_i^*)$. Thus, we have $\hat{h}_i^* = \bar{h}_i^*$ for all $i \in [n]$ and the pair $(\mu^*, (\hat{h}_1^*, \dots, \hat{h}_{n-1}^*, h_n^*, a_{n+1}^*)$ is also valid for $\mathsf{FS_h}$.
Finally, if $(\mu^*, (h_n^*, a_{n+1}^*))$ is new, then the converted signature is also new. This completes the proof. □

We also note that the above proof can be used to show wNR security.

Corollary B.2 ($\mathsf{FS_{cmt}} \Rightarrow \mathsf{FS}_{\mathrm{h,last}}$). *Suppose that* Rep *is decomposable. If* $\mathsf{FS_h}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$ *is* wNR*-secure, then* $\mathsf{FS}_{\mathrm{h,last}}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$ *is also.*

C MQDSS

To discuss HVZK and non-divergency, we propose a new simulator for the SSH11 protocol $\mathsf{SSH11}$. The simulator $\mathsf{Sim_{SSH11}}$ is defined as follows, where we omit the randomness for Com and set $\tau = 1$ for brevity:

1. Receive input $vk = (F, \boldsymbol{v})$, $c_1 = \alpha \in \mathbb{F}_q$, and $c_2 = b \in \{0, 1\}$.
2. Compute messages as follows:
   - If $b = 0$, then pick $\boldsymbol{r}_0, \boldsymbol{t}_0 \leftarrow \mathbb{F}_q^n$ and $\boldsymbol{e}_0 \leftarrow \mathbb{F}_q^m$, compute $\mathsf{com}_0 := \mathsf{Com}(\boldsymbol{r}_0, \boldsymbol{t}_0, \boldsymbol{e}_0)$, pick a random $\mathsf{com}_1 \leftarrow \{0,1\}^\kappa$, compute $a_2 = (\boldsymbol{t}_1, \boldsymbol{e}_1) = (\alpha \boldsymbol{r}_0 - \boldsymbol{t}_0, \alpha F(\boldsymbol{r}_0) - \boldsymbol{e}_0)$, and set $a_3 = \boldsymbol{r}_0$.
   - If $b = 1$, then pick $\boldsymbol{r}_1, \boldsymbol{t}_1 \leftarrow \mathbb{F}_q^n$ and $\boldsymbol{e}_1 \leftarrow \mathbb{F}_q^m$, compute $\mathsf{com}_1 := \mathsf{Com}(\boldsymbol{r}_1, \alpha(\boldsymbol{v} - F(\boldsymbol{r}_1)) - G(\boldsymbol{t}_1, \boldsymbol{r}_1) - \boldsymbol{e}_1)$, pick a random $\mathsf{com}_0 \leftarrow \{0,1\}^\kappa$, set $a_2 := (\boldsymbol{t}_1, \boldsymbol{e}_1)$, and set $a_3 := \boldsymbol{r}_1$.
3. Output $(a_1 = (\mathsf{com}_0, \mathsf{com}_1), a_2, a_3)$.

It is easy to show that $\mathsf{SSH11}$ is q-HVZK, assuming Com is hiding.
It is also easy to that $\mathsf{SSH11}$ is strongly non-divergent: If the condition (a) is met, then the adversary should break the non-invertibility of Com. If the condition (b) is met, then the adversary should break the binding property of Com. Hence, assuming Com's security, the protocol is strongly non-divergent.
By using those properties, we can salvage the sEUF-CMA security of MQDSS in [DFM20, Cor.24] by using the EUF-NMA security of MQDSS in [DFM20].

D MiRitH

We briefly review MiRitH. The signing key is $\boldsymbol{\alpha} \in \mathbb{F}_q^k$ and $K \in \mathbb{F}_q^{r \times (n-r)}$. The verification key consists of a seed seed_{vk}, which produces $M_1, \dots, M_k \in \mathbb{F}_q^{m \times n}$ via PRG, and a matrix $M_0 \in \mathbb{F}_q^{m \times n}$ such that $M_{\boldsymbol{\alpha}} \left[\begin{smallmatrix} I_{n-r} \\ -K \end{smallmatrix}\right] = O$, where $M_{\boldsymbol{\alpha}} := M_0 + \sum_i \alpha_i M_i$. The condition means that the rank of the matrix $M_{\boldsymbol{\alpha}}$ is at most r.
We modify the underlying MPCitH protocol $\mathsf{ID_{MiRitH}}$, $\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3)$ and V with Rep, as depicted in Figure 16.

- The first challenge R is chosen from $\mathbb{F}_q^{s \times m}$, where $s < m$.
- For $M \in \mathbb{F}_q^{m \times n}$, $M_R \in \mathbb{F}_q^{m \times r}$ and $M_L \in \mathbb{F}_q^{m \times (n-r)}$ denotes the matrices consisting of the first r columns of M and the last $(n-r)$ columns of M, respectively.
- $\mathsf{MakeShares}$ generates pseudorandom shares from the seed and an auxiliary information (salt, i).
- The specification sheet just says that "The parties locally compute $[\![M_{\boldsymbol{\alpha},L}]\!]$ and $[\![M_{\boldsymbol{\alpha},R}]\!]$" in P_2. In the reference implementation, $M_{0,L}$ and $M_{0,R}$ are added in a *single* index, and we let this index be $i = 1$.
- In P_3, a_3 contains all $N - 1$ state informations. But, this can be made compact by using $\mathsf{GetPath}$.

For the details, see the original specification [ARV+23]. The signature scheme $\mathsf{MiRitH} = \mathsf{FS_h}[\mathsf{ID_{MiRitH}}, \mathsf{H}, \boldsymbol{\gamma}]$ is defined by $\mathsf{aux}_1 = (\mathsf{salt}, \mu)$ and $\mathsf{aux}_2 = (\mathsf{salt}, \mu)$. They used implicit domain separation of H for h_1 and h_2 [ARV+23, Sec.6.5], because the lengthes of a_1 and (h_1, a_2) differ.

D.1 Security

sEUF-CMA *security:*

Lemma D.1 (q_S-HVZK). *Suppose that* $\mathsf{TreePRG}$ *and* $\mathsf{MakeShares}$ *are pseudorandom and* Com *is hiding. Then,* $\mathsf{ID_{MiRitH}}$ *is* q_S*-HVZK.*

Proof (Proof sketch). Following the proofs in [ARV+23, Sec.9.3] and [FJR22, Sec.E of ePrint], we give a sketch of the proof:

$\underline{\mathsf{P}_1(sk) \text{ for MiRitH}}$

Choose salt at random

//Setup MPC

//Run the following procedure in parallel

Choose seed at random

//The original doesn't have ρ_i

$(\mathsf{seed}_i, \rho_i)_{i\in[N]} := \mathsf{TreePRG}(\mathsf{seed}, \mathsf{salt})$

for $i = 1$ *to* $N-1$ **do**

  $([\![A]\!]_i, [\![\boldsymbol{\alpha}]\!]_i, [\![C]\!]_i, [\![K]\!]_i) := \mathsf{MakeShares}(\mathsf{seed}_i, \mathsf{salt})$

  $\mathsf{state}_i := \mathsf{seed}_i$

//The first part only for $i = N$

$[\![A]\!]_N := \mathsf{MakeShares}(\mathsf{seed}_N, \mathsf{salt})$

$A := \sum_i [\![A]\!]_i$

$[\![\boldsymbol{\alpha}]\!]_N := \boldsymbol{\alpha} - \sum_{i\in[N-1]} [\![\boldsymbol{\alpha}]\!]_i$

$[\![K]\!]_N := K - \sum_{i\in[N-1]} [\![K]\!]_i$

$[\![C]\!]_N := AK - \sum_{i\in[N-1]} [\![C]\!]_i$

$\mathsf{state}_N := (\mathsf{state}_i, [\![\boldsymbol{\alpha}]\!]_N, [\![K]\!]_N, [\![C]\!]_N)$

//Commit the input of MPC

forall $i \in [N]$: $\mathsf{com}_i := \mathsf{Com}((\mathsf{salt}, i, \mathsf{state}_i); \rho_i)$

$a_1 := (\mathsf{com}_1, \ldots, \mathsf{com}_N)_{e\in[\tau]}$

$\mathsf{state} := \big(\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i\in[N]}, (\mathsf{com}_i)_{i\in[N]}, ([\![A]\!]_i, [\![\boldsymbol{\alpha}]\!]_i, [\![K]\!]_i, [\![C]\!]_i)_{i\in[N]}\big)$

return a_1 and state

$\underline{\mathsf{P}_2(sk, R, \mathsf{state}) \text{ for MiRitH}}$

//Simulate MPC

//The offset follows the reference implementation

$[\![M_{\boldsymbol{\alpha},L}]\!]_1 := M_{0,L} + \sum_{j\in[k]} [\![\alpha_j]\!]_1 M_{j,L}$

forall $i \in [2, N]$: $[\![M_{\boldsymbol{\alpha},L}]\!]_i := \sum_{j\in[k]} [\![\alpha_j]\!]_i M_{j,L}$

$[\![M_{\boldsymbol{\alpha},R}]\!]_i := M_{0,R} + \sum_{j\in[k]} [\![\alpha_j]\!]_i M_{j,R}$

forall $i \in [2, N]$: $[\![M_{\boldsymbol{\alpha},R}]\!]_i := \sum_{j\in[k]} [\![\alpha_j]\!]_i M_{j,R}$

forall $i \in [N]$: $[\![S]\!]_i := R \cdot [\![M_{\boldsymbol{\alpha},R}]\!]_i + [\![A]\!]_i$

$S := \sum_{i\in[N]} [\![S]\!]_i$

forall $i \in [N]$: $[\![V]\!]_i := S \cdot [\![K]\!]_i - R \cdot [\![M_{\boldsymbol{\alpha},L}]\!]_i - [\![C]\!]_i$

$a_2 := ([\![S]\!]_i, [\![V]\!]_i)_{i\in[N]}$

$\mathsf{state} := \big(\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i\in[N]}, (\mathsf{com}_i)_{i\in[N]}, ([\![S]\!]_i)_{i\in[N]}\big)$

return a_2 and state

$\underline{\mathsf{P}_3(sk, i^*, \mathsf{state}) \text{ for MiRitH}}$

Parse $\mathsf{state} = \big(\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i\in[N]}, (\mathsf{com}_i)_{i\in[N]}, ([\![S]\!]_i)_{i\in[N]}\big)$

$a_3 := (\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i\neq i^*}, \mathsf{com}_{i^*}, [\![S]\!]_{i^*})$

return a_3

$\underline{\mathsf{Rep}(vk, c_1, c_2, a_3) \text{ for MiRitH}}$

Parse $c_1 = R$ and $c_2 = i^*$

Parse $a_3 = (\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i\neq i^*}, \mathsf{com}_{i^*}, [\![S]\!]_{i^*})$

//Setup MPC

forall $i \in [N] \setminus \{i^*\}$ **do**

  if $i \neq N$ **then**

    Parse $\mathsf{state}_i = \mathsf{seed}_i$

    Compute $[\![A]\!]_i, [\![\boldsymbol{\alpha}]\!]_i, [\![C]\!]_i, [\![K]\!]_i$ from salt and seed_i

  else

    Parse $\mathsf{state}_N = (\mathsf{seed}_N, [\![\boldsymbol{\alpha}]\!]_N, [\![K]\!]_N, [\![C]\!]_N)$

    Compute $[\![A]\!]_N$ from salt and seed_N

  Compute $\mathsf{com}_i := \mathsf{Com}((\mathsf{salt}, i, \mathsf{state}_i); \rho_i)$;

$\bar{a}_1 := (\mathsf{com}_i)_{i\in[N]}$

//Run MPC except i^*

forall $i \in [N] \setminus \{i^*\}$: Compute $[\![M_{\boldsymbol{\alpha},L}]\!]_i$ and $[\![M_{\boldsymbol{\alpha},R}]\!]_i$ from vk and $[\![\boldsymbol{\alpha}]\!]_i$

forall $i \in [N] \setminus \{i^*\}$: $[\![S]\!]_i := R \cdot [\![M_{\boldsymbol{\alpha},R}]\!]_i + [\![A]\!]_i$

$S := \sum_i [\![S]\!]_i$

forall $i \in [N] \setminus \{i^*\}$: $[\![V]\!]_i := S \cdot [\![K]\!]_i - R \cdot [\![M_{\boldsymbol{\alpha},L}]\!]_i - [\![C]\!]_i$

$[\![V]\!]_{i^*} := -\sum_{i\neq i^*} [\![V]\!]_i$

$\bar{a}_2 := ([\![S]\!]_i, [\![V]\!]_i)_{i\in[N]}$

return $\bar{a}_1$ and $\bar{a}_2$

$\underline{\mathsf{V}(vk, a_1, c_1, a_2, c_2, a_3) \text{ for MiRitH}}$

Compute $(\bar{a}_1, \bar{a}_2) := \mathsf{Rep}(vk, c_1, c_2, a_3)$

return $\mathsf{boole}((\bar{a}_1, \bar{a}_2) = (a_1, a_2))$

$\underline{\mathsf{Sim}_{\mathsf{MiRitH}}(vk, c_1, c_2) \text{ for MiRitH}}$

Choose salt at random

//Run the following procedure in parallel

Parse $c_1 = R$ and $c_2 = i^*$

//Simulate MPC's setup

Choose seed at random

$(\mathsf{seed}_i, \rho_i)_{i\in[N]} := \mathsf{TreePRG}(\mathsf{seed}, \mathsf{salt})$

forall $i \in [N] \setminus \{i^*\}$ **do**

  if $i \neq N$ **then**

    Compute $[\![A]\!]_i, [\![\boldsymbol{\alpha}]\!]_i, [\![C]\!]_i, [\![K]\!]_i$ from salt and seed_i

    $\mathsf{state}_i := \mathsf{seed}_i$

  else

    Compute $[\![A]\!]_N$ from salt and seed_N

    Choose $[\![\boldsymbol{\alpha}]\!]_N, [\![K]\!]_N, [\![C]\!]_N$ at random

    $\mathsf{state}_N := (\mathsf{seed}_N, [\![\boldsymbol{\alpha}]\!]_N, [\![K]\!]_N, [\![C]\!]_N)$

  $\mathsf{com}_i := \mathsf{Com}((i, \mathsf{state}_i); \rho_i)$;

Choose com_{i^*} at random

//Simulate MPC's execution

forall $i \in [N] \setminus \{i^*\}$: compute $[\![M_{\boldsymbol{\alpha},L}]\!]$ and $[\![M_{\boldsymbol{\alpha},R}]\!]$ from vk and $[\![\boldsymbol{\alpha}]\!]_i$

forall $i \in [N] \setminus \{i^*\}$: $[\![S]\!]_i := R \cdot [\![M_{\boldsymbol{\alpha},R}]\!]_i + [\![A]\!]_i$

Choose $[\![S]\!]_{i^*}$ at random

$S := \sum_i [\![S]\!]_i$

forall $i \in [N] \setminus \{i^*\}$: $[\![V]\!]_i := S \cdot [\![K]\!]_i - R \cdot [\![M_{\boldsymbol{\alpha},L}]\!]_i - [\![C]\!]_i$

$[\![V]\!]_{i^*} := -\sum_{i\neq i^*} [\![V]\!]_i$

$a_2 := ([\![S]\!]_i, [\![V]\!]_i)_{i\in[N]}$

//Simulate response

$a_3 := (\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i\neq i^*}, \mathsf{com}_{i^*}, [\![S]\!]_{i^*})$

return a_1, a_2, and a_3

Fig. 16. Prover, reconstruction, verification, and simulation algorithms of $\mathsf{ID}_{\mathsf{MiRitH}}$. We run the protocol in τ-parallel way sharing salt.

- G_0: In this game, the transcripts are generated by the real prover.
- G_1: In this game, the challenger chooses challenges c_1 and c_2 and runs the prover using those challenges. This change is just conceptual.
- G_2: In this game, the prover chooses seed_{i^*} and ρ_{i^*} uniformly at random. This modification is justified by the security of TreePRG.
- G_3: Next, the prover chooses $[\![A]\!]_{i^*}$ (and $[\![\alpha]\!]_{i^*}$, $[\![K]\!]_{i^*}$, and $[\![C]\!]_{i^*}$ if $i^* \neq N$) uniformly at random. This modification is justified by the pseudorandomness of MakeShares.
- G_4: Next, the prover chooses $[\![\alpha]\!]_N$, $[\![K]\!]_N$, and $[\![C]\!]_N$ uniformly at random and computes $[\![V]\!]_{i^*} := -\sum_{i \neq \pi^*} [\![V]\!]_i$. The distributions of G_3 and G_4 are equivalent as discussed in [ARV+23, Sec.9.3] and [FJR22, Sec.E of ePrint].
- G_5: Finally, the prover generates $[\![S]\!]_{i^*}$ and com_{i^*} uniformly at random. Now, the prover is the equivalent to Sim. This modification is justified by the hiding property of Com and pseudorandomness of PRG.

□

Lemma D.2 (Strong non-divergency). *Suppose that* Com *is non-invertible and collision-resistant. Then,* $\mathsf{ID}_{\mathsf{MiRitH}}$ *for* MiRitH *is strongly non-divergent with respect to* $\mathsf{Sim}_{\mathsf{MiRitH}}$.

Proof. For simplicity, we ignore parallelness τ. Suppose that the adversary declines a valid transcript $\mathsf{trans}_i = (a_1, c_1, a_2, c_2, a_3)$ generated by the simulator and outputs a valid transcript $\mathsf{trans}' = (a_1, c_1, a_2', c_2', a_3')$. Note that they are valid and share a_1 and c_1. We parse them as $a_1 = (\mathsf{com}_1, \dots, \mathsf{com}_N)$ and $c_1 = R$.
If the condition (a) is met, then we have $c_2 \neq c_2'$: We parse $a_2 = ([\![S]\!]_i, [\![V]\!]_i)_{i\in[N]}$, $c_2 = i^*$, $c_2' = i^+$, and $a_3' = (\mathsf{salt}', (\mathsf{state}_i', \rho_i')_{i\neq i^+}, \mathsf{com}_{i^+}', [\![S']\!]_{i^+})$. Since the adversary opens com_{i^*} as $(\mathsf{salt}', \mathsf{state}_{i^*}', \rho_{i^*}')$ in the valid transcript $(a_1, c_1, a_2', c_2', a_3')$, this breaks the non-invertibility of Com.
If the conditon (b) is met, then we have $(a_2, c_2) = (a_2', c_2')$ and $a_3 \neq a_3'$. We then parse $a_2 = ([\![S]\!]_i, [\![V]\!]_i)_{i\in[N]}$, $c_2 = i^*$, $a_3 = (\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i\neq i^*}, \mathsf{com}_{i^*}, [\![S]\!]_{i^*})$, and $a_3' = (\mathsf{salt}', (\mathsf{state}_i', \rho_i')_{i\neq i^*}, \mathsf{com}_{i^*}', [\![S']\!]_{i^*})$.
We have the following cases:
- If $\mathsf{salt} \neq \mathsf{salt}'$, then we have a collision for Com.
- If $(\mathsf{state}_i, \rho_i)_{i\neq i^*} \neq (\mathsf{state}_i', \rho_i')_{i\neq i^*}$, then we have at least one index i satisfying $(\mathsf{state}_i, \rho_i) \neq (\mathsf{state}_i', \rho_i)$. Since the two transcripts are valid, we have $\mathsf{com}_i = \mathsf{Com}(\mathsf{salt}, \mathsf{state}_i; \rho_i) = \mathsf{Com}(\mathsf{salt}, \mathsf{state}_i', \rho_i')$. This implies a collision for Com.
- If $(\mathsf{com}_{i^*}, [\![S]\!]_{i^*}) \neq (\mathsf{com}_{i^*}', [\![S']\!]_{i^*})$, then at least one of two transcripts are invalid and this never happens.

Using those observations, we can construct reductions easily. □

Due to the definitions of V and Rep, the underlying ID scheme is perfectly sound.

Lemma D.3 (Perfect soundness). $\mathsf{ID}_{\mathsf{MiRitH}}$ *is perfectly sound.*

Since the scheme is (strongly) non-divergent and HVZK, we have the following theorem:

Theorem D.1 (MiRitH's sEUF-CMA security). *Suppose that* $\mathsf{MiRitH} = \mathsf{FS}_\mathsf{h}[\mathsf{ID}_{\mathsf{MiRitH}}, \mathsf{H}, \boldsymbol{\gamma}]$ *is* EUF-NMA*-secure in the (Q)ROM,* TreePRG*, and* MakeShares *are pseudorandom,* Com *is hiding, non-invertible, binding, and collision-resistant. Then,* MiRitH *is* sEUF-CMA*-secure in the (Q)ROM. (If* P_3 *employs* GetPath*, then we need the collision-resistance property of* Reconst*.)*

S-DEO *and* MBS *security:* MiRitH employs FS_h with $\mathsf{aux}_1 = (\mathsf{salt}, \mu)$ and $\mathsf{aux}_2 = (\mathsf{salt}, \mu)$. Therefore, h_1 and h_2 in the signature include the information of μ. Since aux is perfectly collision-resistant with respect to message on index 1, according to Lemma 5.2, MiRitH satisfies S-DEO and MBS if H is collision-resistant.

D.2 S-CEO and wNR Insecurity

We examine the similar strategy of the S-CEO attack against Biscuit in Section 6.
Suppose that we are given $vk = (\mathsf{seed}_{vk}, M_0)$ and seed_{vk} produces $M_1, \dots, M_k$. As the attack against Biscuit, we keep seed_{vk} and modify M_0 into M_0'. If the signature is fixed, then on the second message $a_2 = \left(([\![S]\!]_i, [\![V]\!]_i)_{i\in[N]}\right)_{e\in[\tau]}$, we have $[\![S]\!]_i = R \cdot [\![M_{\alpha,R}]\!]_i + [\![A]\!]_i = R \cdot [\![M_{\alpha,R}']\!]_i + [\![A]\!]_i$ and $[\![V]\!]_i = S \cdot [\![K]\!]_i - R \cdot [\![M_{\alpha,L}]\!]_i + [\![C]\!]_i = S \cdot [\![K]\!]_i - R \cdot [\![M_{\alpha,L}']\!]_i + [\![C]\!]_i$ for $i \in [N] \setminus \{i_e^*\}$, which implies

$$R \cdot ([\![M_\alpha]\!]_i - [\![M_\alpha']\!]_i) = O, \tag{2}$$

where $[\![M_\alpha]\!]_i \in \mathbb{F}_q^{m\times n}$ is the concatenation of $[\![M_{\alpha,R}]\!]_i$ and $[\![M_{\alpha,L}]\!]_i$. Due to the computation of $[\![M_\alpha]\!]_i$, Equation 2 holds for any $i \neq 1$. Therefore, if, for $e \in [\tau]$, $i_e^* \neq 1$ and $R^{(e)} \cdot (M_0 - M_0') = O$ hold, then Equation 2 and the siganture is valid for modified $vk' = (\mathsf{seed}_{vk}, M_0')$. In other words, if we can find such good $(R^{(1)}, \dots, R^{(\tau)}) = \gamma_2(h_1)$ with M_0', we can mount S-CEO and M-S-UEO attacks.

Table 4. Parameter sets in MiRitH's specification v1.0 and success probabitliy with $Q = 2^{64}$.

name	q	m	n	k	r	s	N	τ	p_1	p_Q
Ia-f	16	15	15	78	6	5	16	39	$> 2^{-134.920}$	$> 2^{-70.921}$
Ia-s	16	15	15	78	6	9	256	19	$> 2^{-132.591}$	$> 2^{-68.591}$
IIIa-f	16	19	19	142	4	5	16	55	$> 2^{-192.739}$	$> 2^{-128.739}$
IIIa-s	16	19	19	142	6	9	256	29	$> 2^{-207.346}$	$> 2^{-143.346}$
Va-f	16	21	21	189	7	7	16	74	$> 2^{-272.148}$	$> 2^{-208.148}$
Va-s	16	21	21	189	7	10	256	38	$> 2^{-278.478}$	$> 2^{-214.478}$

Let us calculate a probability p_1 that the above holds for random signature. Let T be the set of indices satisfying $i_e^* = 1$, that is, $T = \{e \in [\tau] : i_e^* = 1\}$ and let τ' be the number of such indices. We can find M_0' by taking a non-trivial vector $\boldsymbol{a}$ from the intersection of kernels $\bigcap_{e \in [\tau] \setminus T} \ker(R^{(e)})$ and setting $M_0' = M_0 + [\boldsymbol{a}, \boldsymbol{0}, \dots, \boldsymbol{0}]$ if and only if $\bigcap_{e \in [\tau] \setminus T} \ker(R^{(e)}) \neq \{\boldsymbol{0}\}$. The condition can be written as $\mathrm{rank}([R^{(i_1)}; \dots; R^{(i_{\tau-\tau'})}]) < m$, where, for $A \in \mathbb{F}_q^{n \times m}$ and $B \in \mathbb{F}_q^{n' \times m}$, $[A; B]$ denotes the block matrix $\binom{A}{B} \in \mathbb{F}_q^{(n+n') \times m}$. Using this argument, we can compute p_1 as

$$p_1 := \sum_{\tau' \in \{0,\dots,\tau\}} p_{\mathrm{num},\tau'} \cdot p_{\mathrm{rank},\tau'},$$

where $p_{\mathrm{num},\tau'} := \Pr_{i_1^*,\dots,i_\tau^* \leftarrow [N]}[\#\{j \in [\tau] : i_j^* = 1\} = \tau']$ and $p_{\mathrm{rank},\tau'} := \Pr_{R_1,\dots,R_{\tau-\tau'} \leftarrow \mathbb{F}_q^{s \times m}}[\mathrm{rank}([R_1; \dots; R_{\tau-\tau'}]) < m] = \Pr_{R' \leftarrow \mathbb{F}_q^{s(\tau-\tau') \times m}}[\mathrm{rank}(R') < m]$. By routine calculation, we have

$$p_{\mathrm{num},\tau'} = \left(\binom{\tau}{\tau'} (N-1)^{\tau-\tau'} / N^\tau \right),$$

$$p_{\mathrm{rank},\tau'} = \begin{cases} 1 & \text{if } s(\tau - \tau') < m, \\ 1 - \prod_{j=s(\tau-\tau')-m+1}^{s(\tau-\tau')} (1 - q^{-j}) & \text{otherwise.} \end{cases}$$

We note that $1 - \prod_{j=s(\tau-\tau')-m+1}^{s(\tau-\tau')} (1-q^{-j}) \leq 2mq^{-(s(\tau-\tau')-m+1)}$. Thus, if $s(\tau-\tau')$ is larger than m, then the probability converges to 0 rapidly. After Q ($\approx 2^{64}$) signing queries, we will have a chance with probability p_Q defined by

$$p_Q := 1 - (1 - p_1)^Q,$$

whose approximation is $Q \cdot p_1$ if $Q \cdot p_1 \ll 1$ and $1 - \exp(-Q \cdot p_1)$ otherwise.
The parameter sets of MiRitH are summarized in Table 4.

- Ia-f: We have $m = 15$, $s = 5$, $N = 16$, and $\tau = 39$. Adding up the probability for $\tau' = 36, 37, 38, 39$, we have $p_1 \geq 2^{-134.92079\dots}$ and $p_Q \geq 1 - (1 - 2^{-134.92079\dots})^{2^{64}} \approx 2^{-134.92079+64} = 2^{-70.92079\dots}$.
- Ia-s: We have $m = 15$, $s = 9$, $N = 256$, and $\tau = 19$. Adding up the probability for $\tau' = 17, 18, 19$, we have $p_1 \geq 2^{-132.59069\dots}$ and $p_Q \geq 1 - (1 - p_1)^{2^{64}} \approx 2^{-132.59069\dots+64} = 2^{-68.59069\dots}$.
- IIIa-f: We have $m = 19$, $s = 5$, $N = 16$, and $\tau = 55$. Summing up the probability for $\tau' \in 51, 52, 53, 54, 55$, we have $p_1 \geq 2^{-192.73929\dots}$ and $p_Q \geq 1 - (1 - p_1)^{2^{64}} \approx 2^{-192.73929\dots+64} = 2^{-128.73929\dots}$.
- IIIa-s: We have $m = 19$, $s = 9$, $N = 256$, and $\tau = 29$. Adding up the probability for $\tau' = 26, 27, 28, 29$, we have $p_1 \geq 2^{-207.34555\dots}$ and $p_Q \geq 1 - (1 - p_1)^{2^{64}} \approx 2^{-207.34555\dots+64} = 2^{-143.34555\dots}$.
- Va-f: We have $m = 21$, $s = 7$, $N = 16$, and $\tau = 74$. Summing up the probability for $\tau' = 71, 72, 73, 74$, we have $p_1 \geq 2^{-272.14842\dots}$ and $p_Q \geq 1 - (1 - p_1)^{2^{64}} \approx 2^{-272.14842\dots+64} = 2^{-208.14842\dots}$.
- Va-s: We have $m = 21$, $s = 10$, $N = 256$, and $\tau = 38$. Adding up the probability for $\tau' = 36, 37, 38, 39$, we have $p_1 \geq 2^{-278.47767\dots}$ and $p_Q \geq 1 - (1 - p_1)^{2^{64}} \approx 2^{-278.47767\dots+64} = 2^{-214.47767\dots}$.

We note that p_Q's in Table 4 are larger than $2^{-\kappa}$, and the above attack for S-CEO is effective. Since p_1 is smaller than $2^{-\kappa}$, we cannot say that MiRitH is vulnerable to wNR. We leave to determine MiRitH is wNR or not as an open problem.

E PERK

We next examine the candidates from PERK v1.1 [ABB+23a].[13] The signing key is a random permutation $\pi \in \mathcal{S}_n$. The verification key consists of pk_seed and $\boldsymbol{y}_1, \dots, \boldsymbol{y}_t \in \mathbb{F}_q^n$; pk_seed produces a sequence of random elements in $\mathbb{F}_q$ to construct random $H \in \mathbb{F}_q^{m \times n}$ and $\boldsymbol{x}_1, \dots, \boldsymbol{x}_t \in \mathbb{F}_q^n$; and $\boldsymbol{y}_j = H \cdot \pi(\boldsymbol{x}_j)$ for all $j = 1, \dots, t$.
Intuitively speaking, the signer will show the relation between $\boldsymbol{y}_j$ and $\boldsymbol{x}_j$. We modify the underlying MPCitH protocol $\mathrm{ID}_{\mathsf{PERK}}$, P and V with Rep, as described in Figure 17.

[13] The version 1.1 is available at https://pqc-perk.org/.

$\underline{\mathsf{P}_1(sk)\text{ for PERK}}$
Choose salt and mseed uniformly at random
$(\mathsf{seed}^{(1)}, \dots, \mathsf{seed}^{(\tau)}) := \mathsf{PRG}(\mathsf{salt}, \mathsf{mseed})$
//Run the following procedure in parallel for $e \in [\tau]$
$(\mathsf{seed}_i, \rho_i)_{i \in [N]} := \mathsf{TreePRG}(\mathsf{seed}, \mathsf{salt})$
for $i = N$ *to* 2 **do**
  $(\pi_i, \boldsymbol{v}_i) := \mathsf{MakeShares}(\mathsf{seed}_i, \mathsf{salt})$
  $\mathsf{state}_i := \mathsf{seed}_i$
//The second part only for $i = 1$
$\boldsymbol{v}_1 := \mathsf{MakeShares}(\mathsf{seed}_1, \mathsf{salt})$
$\pi_1 := \pi_2^{-1} \circ \cdots \circ \pi_N^{-1} \circ \pi$
$\mathsf{state}_1 := (\pi_1, \mathsf{seed}_1)$
forall $i \in [N]$: $\mathsf{com}_{1,i} := \mathsf{Com}((\mathsf{salt}, e, i, \mathsf{state}_i); \rho_i)$
$\boldsymbol{v} := \boldsymbol{v}_N + \sum_{i \in [N-1]} \pi_N \circ \cdots \circ \pi_{i+1}(\boldsymbol{v}_i)$
$\mathsf{com}_1 := \mathsf{H}_0(\mathsf{salt}, e, H\boldsymbol{v})$
$a_1 := (\mathsf{com}_1, (\mathsf{com}_{1,i})_{i \in [N]})$
$\mathsf{state} := (\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i \in [N]}, (\mathsf{com}_{1,i})_{i \in [N]})$
return a_1 and state

$\underline{\mathsf{P}_2(sk, \boldsymbol{\kappa}, \mathsf{state})\text{ for PERK}}$
parse $\mathsf{state} = (\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i \in [N]}, (\mathsf{com}_{1,i})_{i \in [N]})$
$\boldsymbol{s}_0 := \sum_{j \in [t]} \kappa_j \boldsymbol{x}_j$
forall $i \in [N]$ **do**
  $\boldsymbol{s}_i := \pi_i(\boldsymbol{s}_{i-1}) + \boldsymbol{v}_i$
$a_2 := (\boldsymbol{s}_i)_{i \in [N]}$
$\mathsf{state} := (\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i \in [N]}, (\mathsf{com}_{1,i})_{i \in [N]}, (\boldsymbol{s}_i)_{i \in [N]})$
return a_2 and state

$\underline{\mathsf{P}_3(sk, i^*, \mathsf{state})\text{ for PERK}}$
parse $\mathsf{state} = (\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i \in [N]}, (\mathsf{com}_{1,i})_{i \in [N]}, (\boldsymbol{s}_i)_{i \in [N]})$
$a_3 := (\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i \neq i^*}, \mathsf{com}_{1,i^*}, \boldsymbol{s}_{i^*})$
return a_3

$\underline{\mathsf{Rep}(vk, c_1, c_2, a_3)\text{ for PERK}}$
Parse $c_1 = \boldsymbol{\kappa} = (\kappa_1, \dots, \kappa_t)$
Parse $c_2 = i^*$
Parse $a_3 = (\mathsf{salt}, (\overline{\mathsf{state}}_i, \bar{\rho}_i)_{i \neq i^*}, \overline{\mathsf{com}}_{1,i^*}, \bar{\boldsymbol{s}}_{i^*})$
//Setup MPC
forall $i \in [N] \setminus \{i^*\}$ **do**
  if $i \neq 1$ **then**
    Parse $\overline{\mathsf{state}}_i = \overline{\mathsf{seed}}_i$
    Compute $(\bar{\pi}_i, \bar{\boldsymbol{v}}_i)$ from salt and $\overline{\mathsf{seed}}_i$
  else
    Parse $\overline{\mathsf{state}}_1 = (\bar{\pi}_1, \overline{\mathsf{seed}}_1)$
    Compute $\bar{\boldsymbol{v}}_1$ from salt and $\overline{\mathsf{seed}}_1$
  $\overline{\mathsf{com}}_{1,i} := \mathsf{Com}((\mathsf{salt}, e, i, \overline{\mathsf{state}}_i); \rho_i)$
//Run MPC except i^*
$\bar{\boldsymbol{s}}_0 := \sum_{j \in [t]} \kappa_j \boldsymbol{x}_j$
forall $i \in [N] \setminus \{i^*\}$ **do**
  $\bar{\boldsymbol{s}}_i = \bar{\pi}_i(\bar{\boldsymbol{s}}_{i-1}) + \bar{\boldsymbol{v}}_i$
//Wrap up
$\overline{\mathsf{com}}_1 := \mathsf{H}_0\left(\mathsf{salt}, e, H\bar{\boldsymbol{s}}_N - \sum_{j \in [t]} \kappa_j \boldsymbol{y}_j\right)$
$\bar{a}_1 = (\overline{\mathsf{com}}_1, (\overline{\mathsf{com}}_{1,i})_{i \in [N]})$
$\bar{a}_2 := (\bar{\boldsymbol{s}}_i)_{i \in [N]}$
return $\bar{a}_1$ and $\bar{a}_2$

$\underline{\mathsf{V}(vk, a_1, c_1, a_2, c_2, a_3)\text{ for PERK}}$
$(\bar{a}_1, \bar{a}_2) := \mathsf{Rep}(vk, c_1, c_2, a_3)$
return $\mathsf{boole}((a_1, a_2) = (\bar{a}_1, \bar{a}_2))$

Fig. 17. Prover, reconstruction, and verification algorithms for $\mathsf{ID}_{\mathsf{PERK}}$. We run the protocol in τ-parallel way sharing salt.

- MakeShares generates pseudorandom shares $(\pi_i^{(e)}, \boldsymbol{v}_i^{(e)})$ from the seed $\mathsf{seed}_i^{(e)}$ with an auxiliary information salt, where $\pi_i^{(e)} \in \mathcal{S}_n$ and $\boldsymbol{v}_i^{(e)} \in \mathbb{F}_q^n$.
- In P_3, a_3 contains all $N-1$ state informations. This can be made compact by using GetPath.

For the details, see the original specification [ABB+23a]. The signature scheme $\mathsf{PERK} = \mathsf{FS}_\mathsf{h}[\mathsf{ID}_{\mathsf{PERK}}, \mathsf{H}, \boldsymbol{\gamma}]$ is defined by $\mathsf{aux}_1 = (\mathtt{0x01}, \mathsf{salt}, \mu, vk)$ and $\mathsf{aux}_2 = (\mathtt{0x02}, \mathsf{salt}, \mu)$.

E.1 Security

sEUF-CMA *security:* Since we modify the protocol, we need to modify the simulator, which is described in Figure 18. The HVZK property of $\mathsf{ID}_{\mathsf{PERK}}$ is shown in their specification document by following the HVZK proof in [FJR22], but we modify the proof to consider the real protocol as possible. It is easy to check the above simulator Sim yields q-HVZK for polynomial $q = q(1^\kappa)$ as in the proof for Lemma 6.1 and Lemma D.1 by following the original proofs in [FJR22] and [ABB+23a, Thm.3.3].

Lemma E.1 **(q_S-HVZK).** *Suppose that* PRG, TreePRG, *and* MakeShares *are pseudorandom and* Com *is hiding. Then,* $\mathsf{ID}_{\mathsf{PERK}}$ *with simulator* $\mathsf{Sim}_{\mathsf{PERK}}$ *in Figure 18 is* q_S*-HVZK.*

Lemma E.2 **(Strong non-divergency).** *Suppose that* H_0 *is collision-resistant.* Com *is non-invertible and collision-resistant. Then,* $\mathsf{ID}_{\mathsf{PERK}}$ *is* q_S*-non-divergent with respect to* $\mathsf{Sim}_{\mathsf{PERK}}$*.*

Proof. For simplicity, we ignore parallelness τ. Suppose that the adversary declines a valid transcript $\mathsf{trans}_i = (a_1, c_1, a_2, c_2, a_3)$ generated by the simulator and outputs a valid transcript $\mathsf{trans}' = (a_1, c_1, a_2', c_2', a_3')$. Note that they are valid and share a_1 and c_1. We parse them as $a_1 = (\mathsf{com}_1, \mathsf{com}_{1,1}, \dots, \mathsf{com}_{1,N})$ and $c_1 = \boldsymbol{\kappa}$.

$\underline{\mathsf{Sim}_{\mathsf{PERK}}(vk, c_1, c_2) \text{ for PERK}}$

Choose salt uniformly at random

//Run the following procedure in parallel for $e \in [\tau]$

Parse $c_1 = \kappa = (\kappa_1, \dots, \kappa_t)$ and $c_2 = i^*$

Choose seed uniformly at random

$(\mathsf{seed}_i, \rho_i)_{i \in [N]} := \mathsf{TreePRG}(\mathsf{salt}, \mathsf{seed})$

//Simulate MPC's setup

forall $i \in [N] \setminus \{i^*\}$ **do**

  if $i \neq 1$ **then**

    $(\pi_i, \boldsymbol{v}_i) := \mathsf{MakeShares}(\mathsf{seed}_i, \mathsf{salt})$

    $\mathsf{state}_i := \mathsf{seed}_i$

  else

    //The second part only for $i = 1$

    $\boldsymbol{v}_1 := \mathsf{MakeShares}(\mathsf{seed}_i, \mathsf{salt})$

    Choose π_1 at random

    $\mathsf{state}_1 := (\pi_1, \mathsf{seed}_1)$

  $\mathsf{com}_{1,i} := \mathsf{Com}((\mathsf{salt}, e, i, \mathsf{state}_i); \rho_i)$

Choose π_{i^*}, $\boldsymbol{v}_{i^*}$, and com_{1,i^*} uniformly at random

$\boldsymbol{v} := \boldsymbol{v}_N + \sum_{i \in [N-1]} \pi_N \circ \cdots \circ \pi_{i+1}(\boldsymbol{v}_i)$

$\mathsf{com}_1 := \mathsf{H}_0(\mathsf{salt}, e, H\boldsymbol{v})$

$a_1 := (\mathsf{com}_1, (\mathsf{com}_{1,i})_{i \in [N]})$

//Simulate MPC's execution

$\tilde{\pi} := \pi_N \circ \cdots \circ \pi_1$

Compute $\tilde{\boldsymbol{x}}$ s.t. $H\tilde{\boldsymbol{x}} = \sum_j \kappa_j \boldsymbol{y}_j$

$\boldsymbol{s}_0 := \sum_j \kappa_j \boldsymbol{x}_j$

foreach $i \in \{1, \dots, i^* - 1\}$: $\boldsymbol{s}_i := \pi_i(\boldsymbol{s}_{i-1}) + \boldsymbol{v}_i$

$\boldsymbol{s}_{i^*} := \pi_{i^*}(\boldsymbol{s}_{i^*-1}) + \boldsymbol{v}_{i^*} + \pi_{i^*+1}^{-1} \circ \cdots \circ \pi_N^{-1}\left(\tilde{\boldsymbol{x}} - \tilde{\pi}(\boldsymbol{s}_0)\right)$

foreach $i \in \{i^* + 1, \dots, N\}$: compute

  $\boldsymbol{s}_i := \pi_i(\boldsymbol{s}_{i-1}) + \boldsymbol{v}_i$

$a_2 := (\boldsymbol{s}_i)_{i \in [N]}$

//Simulate response

$a_3 := (\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i \neq i^*}, \mathsf{com}_{1,i^*}, \boldsymbol{s}_{i^*})$

return a_1, a_2, and a_3

Fig. 18. Simulation algorithm for $\mathsf{ID}_{\mathsf{PERK}}$. We run the protocol in τ-parallel way sharing salt.

If the condition (a) is met, then we have $c_2 \neq c_2'$. We parse $c_2 = i^*$, $c_2' = i^+$, and $a_3' = (\mathsf{salt}', (\mathsf{state}_i', \rho_i')_{i \neq i^+}, \mathsf{com}_{1,i^+}', \boldsymbol{s}_{i^+}')$. Notice that the adversary opens com_{1,i^*} as $(\mathsf{salt}', e^*, i^*, \mathsf{state}_{i^*}', \rho_{i^*}')$ due to the validity of the transcript $(a_1, c_1, a_2', c_2', a_3')$. Thus, we have $\mathsf{com}_{1,i^*} = \mathsf{Com}((\mathsf{salt}', e^*, i^*, \mathsf{state}_{i^*}'); \rho_{i^*}')$. Since com_{1,i^*} is chosen uniformly at random by the simulator, this violates the non-invertibility of Com.
If the condition (b) is met, then we have $(a_2, c_2) = (a_2', c_2')$ and $a_3 \neq a_3'$. We parse $a_2 = (\boldsymbol{s}_i)_{i \in [N]}$, $c_2 = i^*$, $a_3 = (\mathsf{salt}, (\mathsf{state}_i, \rho_i)_{i \neq i^*}, \mathsf{com}_{1,i^*}, \boldsymbol{s}_{i^*})$, and $a_3' = (\mathsf{salt}', (\mathsf{state}_i', \rho_i')_{i \neq i^*}, \mathsf{com}_{1,i^*}', \boldsymbol{s}_{i^*}')$. We have the following cases:

- If $\mathsf{salt} \neq \mathsf{salt}'$, then we have a collision H_0 and break the binding property of Com.
- If $(\mathsf{state}_i, \rho_i)_{i \neq i^*} \neq (\mathsf{state}_i', \rho_i')_{i \neq i^*}$, then we have at least one index i satisfying $(\mathsf{state}_i, \rho_i) \neq (\mathsf{state}_i', \rho_i')$. Since the two transcripts are valid, we have $\mathsf{com}_{1,i} = \mathsf{Com}(\mathsf{salt}, e, i, \mathsf{state}_i; \rho_i) = \mathsf{Com}(\mathsf{salt}, e, i, \mathsf{state}_i'; \rho_i')$. This implies a break of the collision-resistance property of Com.
- If $\mathsf{com}_{1,i^*} \neq \mathsf{com}_{1,i^*}'$, then this contradicts with a_1 and the validity of the transcripts.
- If $\boldsymbol{s}_{i^*} \neq \boldsymbol{s}_{i^*}'$, then this contradicts with a_2 and the validity of the transcripts.

Using those observations, we can construct reductions easily. □

Due to the definitions of V and Rep, the underlying ID scheme is perfectly sound.

Lemma E.3 (Perfect soundness). $\mathsf{ID}_{\mathsf{PERK}}$ *is perfectly sound.*

Since the scheme is (strongly) non-divergent and HVZK, we have the following theorem:

Theorem E.1 (PERK's sEUF-CMA security). *Suppose that* $\mathsf{PERK} = \mathsf{FS}_{\mathsf{h}}[\mathsf{ID}_{\mathsf{PERK}}, \mathsf{H}, \boldsymbol{\gamma}]$ *is* EUF-NMA*-secure in the (Q)ROM,* PRG, $\mathsf{TreePRG}$, *and* $\mathsf{MakeShares}$ *are pseudorandom,* H_0 *is collision-resistant,* Com *is hiding, non-invertible, and collision-resistant. Then,* PERK *is* sEUF-CMA*-secure in the (Q)ROM. (If* P_3 *employs* $\mathsf{GetPath}$*, then we need the collision-resistance property of* $\mathsf{Reconst}$*.)*

BUFF security: Recall that $\mathsf{aux}_1 = (\mathtt{0x01}, \mathsf{salt}, \mu, vk)$ and $\mathsf{aux}_2 = (\mathtt{0x02}, \mathsf{salt}, \mu)$ in PERK. It is obvious that aux is perfectly collision-resistant with respect to the message on index 1. Thus, applying Lemma 5.2, PERK sasifies MBS and M-S-UEO. In addition, aux is perfectly collision-resistant with respect to the verification key on index 1, and both aux_1 and aux_2 can be written as (μ, η_1) and (μ, η_2), respectively. Hence, PERK satisfies wNR due to Lemma 5.3.

Theorem E.2. *Assume that* H *is collision-resistant. Then,* $\mathsf{PERK} = \mathsf{FS}_{\mathsf{h}}[\mathsf{ID}_{\mathsf{PERK}}, \mathsf{H}, \boldsymbol{\gamma}]$ *satisfies* MBS *and* M-S-UEO*. If* H *is a random oracle, then* PERK *satisfies* wNR*.*

F AIMer

We briefly review AIMer [KCC+23].

Let $\mathsf{AIM} : \{0,1\}^\kappa \times \mathbb{F}_{2^\kappa} \to \mathbb{F}_{2^\kappa}$ be a tweakable one-way function defined in [KCC+23]. The signing key is $\mathsf{pt} \in \mathbb{F}_{2^\kappa}$. The verification key is $(\mathsf{iv}, \mathsf{ct})$ such that $\mathsf{AIM}(\mathsf{iv}, \mathsf{pt}) = \mathsf{ct}$. The abstract structure of the underlying MPCitH protocol $\mathsf{ID}_{\mathsf{AIMer}}$ is very similar to that in $\mathsf{Biscuit}$, and we do not give the full details of AIMer. (Their MPCitH protocol is based on $\mathsf{BN{+}{+}}$ proposed by Kales and Zaverucha [KZ22].) In AIMer, the signature is computed as follows:

- Compute $a_1, h_1 := \mathsf{H}(\mathtt{0x01}, \mu, vk, \mathsf{salt}, a_1)$, and $c_1 := \gamma_1(h_1)$.
- Compute $a_2, h_2 := \mathsf{H}(\mathtt{0x02}, \mathsf{salt}, h_1, a_2)$, and $c_2 := \gamma_2(h_2)$.
- Compute a_3, which includes salt, and output $\sigma := (h_1, h_2, a_3)$.

The verifier verifies a signature as follows:

- Compute $c_1 := \gamma_1(h_1)$ and $c_2 := \gamma_2(h_2)$.
- Reconstruct $\bar{a}_1$ and $\bar{a}_2$ from c_1, c_2, a_3.
- Compute $\bar{h}_1 := \mathsf{H}(\mathtt{0x01}, \mu, vk, \mathsf{salt}, \bar{a}_1)$ and $\bar{h}_2 := (\mathtt{0x02}, \mathsf{salt}, h_1, \bar{a}_2)$
- Output $\mathsf{boole}(h_1 = \bar{h}_1 \wedge h_2 = \bar{h}_2)$.

We can consider AIMer as $\mathsf{FS}_{\mathrm{h}}[\mathsf{ID}_{\mathsf{AIMer}}, \mathsf{H}, \boldsymbol{\gamma}]$ with $\mathsf{aux}_1 = (\mathtt{0x01}, \mu, vk, \mathsf{salt})$ and $\mathsf{aux}_2 = (\mathtt{0x02}, \mathsf{salt})$. aux is perfectly collision-resistant with respect to the message and verification key on index 1.

It is easy to check the underlying protocol is HVZK and strongly non-divergent under appropriate assumptions on the primitives used in the protocol. Therefore, AIMer is sEUF-CMA-secure in the (Q)ROM if it is EUF-NMA-secure in the (Q)ROM and used primitives are secure.

Since aux is collision-resistant with respect to the message and verification key on index 1, AIMer enjoys M-S-UEO and MBS securities if H is collision-resistant. In addition, aux_1 an be written as (μ, η_1). Hence, AIMer is wNR-secure if H is the random oracle.

G Generic MPCitH using Embedding

This section treats MIRA, RYDE, SDitH, and MQOM. Essentially speaking, the signer of those schemes shows the relation between the verification key and the signing key over $\mathbb{F}_q$ via MPC using polynomials and the extension field $\mathbb{F}_{q^\eta}$ by using the framework proposed by Feneuil, Joux, and Rivain [FJR22]. They also used the Hypercube-in-the-Head techniques proposed by Aguilar-Melchor, Gama, Howe, Hülsing, Joseph, and Yue [AGH+23].

Aguilar-Melchor et al. [AGH+23] showed the 1-HVZK property of the underlying 5-pass MPCitH protocol. It is easy to check the underlying protocol is also q-HVZK by tracing their proof. It is also easy to check the protocol is strongly non-divergent under appropriate assumptions on the primitives used in the protocol.

G.1 MIRA and RYDE

We briefly review MIRA [ABB+23c] and RYDE [ABB+23b], which share the framework. Since the difference of RYDE from MIRA is only the underlying problem, we here review MIRA. Let $\mathsf{ID}_{\mathsf{MIRA}}$ be the underlying 5-pass MPCitH protocol. Let $\mathsf{DS\_1}, \mathsf{DS\_2}, \mathsf{DS\_M} \in \{0,1\}^\kappa$ be domain separators for the random oracle H. In MIRA, the signature is computed as follows:

1. Let $\mathsf{md} := \mathsf{H}_m(\mu)$, where $\mathsf{H}_m(\mu) = \mathsf{H}(\mathsf{DS\_M}, \mu)$.
2. Compute $a_1, h_1 := \mathsf{H}(\mathsf{DS\_1}, \mathsf{salt}, vk, \mathsf{md}, a_1)$, and $c_1 := \gamma_1(h_1)$
3. Compute $a_2, h_2 := \mathsf{H}(\mathsf{DS\_2}, \mathsf{salt}, vk, \mathsf{md}, h_1, a_2)$, and $c_2 := \gamma_2(h_2)$
4. Compute a_3, which includes salt, and output $\sigma := (h_1, h_2, a_3)$

The verification algorithm verifies a signature as follows:

1. Let $\mathsf{md} := \mathsf{H}_m(\mu)$.
2. Compute $c_1 := \gamma_1(h_1)$ and $c_2 := \gamma_2(h_2)$.
3. Reconstruct $\bar{a}_1$ and $\bar{a}_2$ from c_1, c_2, a_3.
4. Compute $\bar{h}_1 := \mathsf{H}(\mathsf{DS\_1}, \mathsf{salt}, vk, \mathsf{md}, \bar{a}_1)$ and $\bar{h}_2 := \mathsf{H}(\mathsf{DS\_2}, \mathsf{salt}, vk, \mathsf{md}, h_1, \bar{a}_2)$.
5. Output $\mathsf{boole}(h_1 = \bar{h}_1 \wedge h_2 = \bar{h}_2)$.

Thus, we can consider MIRA as $\mathsf{FS}_{\mathrm{h}}[\mathsf{ID}_{\mathsf{MIRA}}, \mathsf{H}, \boldsymbol{\gamma}]$ with $\mathsf{aux}_1 = (\mathsf{DS\_1}, \mathsf{salt}, vk, \mathsf{md})$ and $\mathsf{aux}_2 = (\mathsf{DS\_2}, \mathsf{salt}, vk, \mathsf{md})$, where $\mathsf{md} = \mathsf{H}(\mathsf{DS\_M}, \mu)$. aux is collision-resistant with respect to the message and verification key on index 1 if H is collision-resistant.

It is easy to check the underlying protocol is HVZK and strongly non-divergent under appropriate assumptions on the primitives used in the protocol. Therefore, MIRA is sEUF-CMA-secure in the (Q)ROM if it is EUF-NMA-secure in the (Q)ROM and used primitives are secure. Since aux is collision-resistant with respect to the message and verification key on index 1, MIRA enjoys M-S-UEO and MBS securities if H is collision-resistant. By replacing μ with md, we can apply Lemma 5.3 and show that MIRA is wNR-secure if H is the random oracle.

G.2 SDitH – SDitH-HC

We briefly review SDitH v.1.1 [AFG+23].[14] Here, we only consider the hypercubic MPCitH version, which we call SDitH-HC. Let $\mathsf{ID}_{\mathsf{SDitH\text{-}HC}}$ be the underlying 5-pass MPCitH protocol. In SDitH-HC, the signature is computed as follows:

1. Compute $a_1, h_1 := \mathsf{H}(\mathtt{0x01}, \mathsf{salt}, vk, a_1)$, and $c_1 := \gamma_1(h_1)$.
2. Compute $a_2, h_2 := \mathsf{H}(\mathtt{0x02}, \mathsf{salt}, \mu, h_1, a_2)$, and $c_2 := \gamma_2(h_2)$.
3. Compute a_3, which includes salt, and output $\sigma := (h_2, a_3)$.

The verification algorithm verifies a signature as follows:

1. Compute $c_2 := \gamma_2(h_2)$.
2. Reconstruct $\bar{a}_1$ from c_2 and a_3.
3. Compute $\bar{h}_1 := \mathsf{H}(\mathtt{0x01}, \mathsf{salt}, vk, \bar{a}_1)$ and $\bar{c}_1 := \gamma_1(\bar{h}_1)$.
4. Reconstruct $\bar{a}_2$ from $\bar{c}_1$ and so on. and $\bar{h}_2 := \mathsf{H}(\mathtt{0x02}, \mathsf{salt}, \mu, h_1, \bar{a}_2)$.
5. Output $\mathsf{boole}(h_2 = \bar{h}_2)$.

Thus, we can consider SDitH-HC as $\mathsf{FS}_{\mathsf{h,last}}[\mathsf{ID}_{\mathsf{SDitH\text{-}HC}}, \mathsf{H}, \boldsymbol{\gamma}]$ with $\mathsf{aux}_1 = (\mathtt{0x01}, \mathsf{salt}, vk)$ and $\mathsf{aux}_2 = (\mathtt{0x02}, \mathsf{salt}, \mu)$. aux is collision-resistant with respect to message on index 2 and collision-resistant with respect to verification key on index 1.

Since aux is collision-resistant with respect to message on index 2 and h_2 is included in the signature, SDitH-HC is MBS-secure.

To show M-S-UEO security, we need a short (routine) discussion since h_1 is not in the signature. If there is an adversary against the M-S-UEO security, then its output contains two different verification keys vk and vk', two messages μ and μ', and a signature $\sigma = (h_2, a_3)$, where a_3 contains salt. Let a_1 (or a'_1, resp.) be the first messages reconstructed from vk (or vk', resp.), a_3, and $c_2 = \gamma_2(h_2)$. We then let $\hat{h}_1 = \mathsf{H}(\mathtt{0x01}, \mathsf{salt}, vk, a_1)$ and $\hat{h}'_1 = \mathsf{H}(\mathtt{0x01}, \mathsf{salt}, vk, a_1)$.

- If $\hat{h}_1 = \hat{h}'_1$, then we find a collision of H.
- Otherwise, we let $\hat{h}_2 = \mathsf{H}(\mathtt{0x02}, \mathsf{salt}, \mu, \hat{h}_1, a_1)$ and $\hat{h}'_2 = \mathsf{H}(\mathtt{0x02}, \mathsf{salt}, \mu, \hat{h}'_1, a'_1)$. Since the signature is valid for both messages and verification keys, we have $\hat{h}_2 = h_2 = \hat{h}'_2$ and find a collision of H.

Thus, if H is collision-resistant, then SDitH-HC is M-S-UEO-secure.

If we consider $\mathsf{SDitH\text{-}HC}' := \mathsf{FS}_{\mathsf{h}}[\mathsf{ID}_{\mathsf{SDitH\text{-}HC}}, \mathsf{H}, \boldsymbol{\gamma}]$, then we can apply Lemma 5.3 and SDitH-HC′ is wNR-secure if H is the random oracle since aux_1 is collision-resistant with respect to verification key on index 1 and aux_2 can be written as (μ, η_2). Corollary B.2 states that if $\mathsf{FS}_{\mathsf{h}}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$ is wNR-secure, then $\mathsf{FS}_{\mathsf{h,last}}[\mathsf{ID}, \mathsf{H}, \boldsymbol{\gamma}]$ is also wNR-secure. Hence, SDitH-HC is also wNR-secure if H is the random oracle.

Remark G.1. Aguilar-Melchor et al. [AHJ+23] treat the underlying ID protocol as *collapsed* 3-pass ID protocol, where the prover computes (a_1, a_2) by computing h_1 and c_1 by itself, the verifier sends a random challenge c_2, and the prover answers sends a_3. They then apply the FS transform and show the obtained signature SDitH-HC is EUF-CMA-secure in the QROM as Grilo et al. [GHHM21]. We can show the collapsed 3-pass ID protocol is CUR [KLS18] and extend their proof into the sEUF-CMA security proof.

G.3 MQOM

We briefly review MQOM [FR23]. Let $\mathsf{ID}_{\mathsf{MQOM}}$ be the underlying 7-pass MPCitH protocol. In MQOM, the signature is computed as follows: [15]

1. Compute $a_1, h_1 := \mathsf{H}(\mathtt{0x01}, \mathsf{salt}, vk, \mu, a_1)$, and $c_1 := \gamma_1(h_1)$.
2. Compute $a_2, h_2 := \mathsf{H}(\mathtt{0x02}, \mathsf{salt}, \mu, h_1, a_2)$, and $c_2 := \gamma_2(h_2)$.
3. Compute $a_3, h_3 := \mathsf{H}(\mathtt{0x03}, \mathsf{salt}, \mu, h_2, a_3)$, and $c_3 := \gamma_3(h_3)$.
4. Compute a_4, which includes salt, and output $\sigma := (h_1, h_2, h_3, a_4)$.

The verification algorithm verifies a signature as follows:

1. Compute $c_1 := \gamma_1(h_1)$, $c_2 := \gamma_2(h_2)$, and $c_3 := \gamma_3(h_3)$.
2. Reconstruct $(\bar{a}_1, \bar{a}_2, \bar{a}_3)$ from c_1, c_2, c_3, a_4.
3. Compute $\bar{h}_1 := \mathsf{H}(\mathtt{0x01}, \mathsf{salt}, vk, \mu, \bar{a}_1)$, $\bar{h}_2 := \mathsf{H}(\mathtt{0x02}, \mathsf{salt}, \mu, h_1, \bar{a}_2)$, and $\bar{h}_3 := \mathsf{H}(\mathtt{0x03}, \mathsf{salt}, \mu, h_2, \bar{a}_3)$
4. Output $\mathsf{boole}(h_1 = \bar{h}_1 \wedge h_2 = \bar{h}_2 \wedge h_3 = \bar{h}_3)$.

Thus, we can consider MQOM as $\mathsf{FS}_{\mathsf{h}}[\mathsf{ID}_{\mathsf{MQOM}}, \mathsf{H}, \boldsymbol{\gamma}]$ with $\mathsf{aux}_1 = (\mathtt{0x01}, \mathsf{salt}, vk, \mu)$, $\mathsf{aux}_2 = (\mathtt{0x02}, \mathsf{salt}, \mu)$, and $\mathsf{aux}_3 = (\mathtt{0x03}, \mathsf{salt}, \mu)$. aux is perfectly collision-resistant with respect to the message and verification key on index 1.

We can routinely show $\mathsf{ID}_{\mathsf{MQOM}}$'s HVZK and strong non-divergency under appropriate assumptions. Therefore, MQOM is sEUF-CMA-secure in the (Q)ROM if it is EUF-NMA-secure in the (Q)ROM and used primitives are secure. Since aux is collision-resistant with respect to the message and verification key on index 1, MQOM is M-S-UEO and MBS securities if H is collision-resistant. In addition, aux_1 can be written as (μ, η_1). Thus, MQOM is wNR-secure if H is the random oracle (Lemma 5.3).

[14] Version 1.1 is available at `https://sdith.org/resources.html`.

[15] On the input of hash functions, we adopt the definitions in the implementation (`mqrom_cat1_gf31_fast` in reference implementations), since there is an inconsistency between high-level description (Figures 2 and 3) and low-level description (Algorithms 8, 9, 10, and 11) in the specification documents [FR23].

H Generic VOLEitH

Recently, a close variant of MPCitH-type signatures called *VOLE-in-the-Head* [16] *or VOLEitH* type signature was introduced [BBD+23b] to design FAEST signature scheme based on symmetric key primitives (block ciphers). In this approach, one begins by proving knowledge of a witness (such as secret key of block cipher) with the help of zero-knowledge proof of knowledge system based on VOLE correlations and then convert this ZKPoK into signature scheme via Fiat-Shamir transformation. In spirit this is similar to constructing MPCitH-type ZKPoK with only 2 parties (prover and verifier) using correlated randomness.

H.1 Example: FAEST

We review FAEST v1.1[17] briefly below. The signing key is the secret key sk of a block cipher (from here onward we will consider AES as the underlying block cipher) where as the verification key consists of plaintext $\boldsymbol{x}$ and ciphertext $\boldsymbol{y}$ such that $\boldsymbol{y} := \mathsf{Enc}_{sk}(\boldsymbol{x})$. Additionally, the prover (signer) and verifier interact with an ideal functionality $\mathcal{F}_{\mathsf{VOLE}}$ which generates correlated random values u, v, Δ, q such that $q = u \cdot \Delta + v$ and sends (u, v) to the prover and (Δ, q) to the verifier. This ideal functionality is implemented using puncturable PRF by building a GGM tree from a length-doubling secure pseudorandom generator PRG. The protocol proceeds as follows:

1. Prover embeds the witness w corresponding to the secret sk in the VOLE correlation such that $q = w \cdot \Delta + v$. Specifically, prover computes $d := w - u$ and sends d to the verifier. Since verifier does not know u sending d does not leak anything about the witness w. The verifier can then *locally* update q as $q := q + d \cdot \Delta$ which corresponds to the VOLE correlation with respect to the witness as $q = w \cdot \Delta + v$ and since the mask v is known only to the prover updated q does not leak any information about the witness.
2. The prover and verifier then run the QuickSilver protocol [YSWW21] with the help of VOLE correlation $q = w \cdot \Delta + v$, to check that on the input witness w and verification key $(\boldsymbol{x}, \boldsymbol{y})$ the AES circuit evaluates to 1.

In order to achieve the desired security level (such as 128-bit security) the above protocol is repeated τ times with independent VOLE correlations $(u_i, v_i, q_i, \Delta_i)$ for $i \in [\tau]$. We present the underlying VOLEitH protocol (which is implicit in FAEST signature specification) as $\mathsf{ID}_{\mathsf{FAEST}}$, $\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3, \mathsf{P}_4)$ and V with Rep, as depicted in Figure 19 and Figure 20 to fit their scheme in our framework.

As stated earlier, the ideal VOLE functionality $\mathcal{F}_{\mathsf{VOLE}}$ is implemented by constructing GGM tree using a secure length-doubling pseudorandom generator PRG. The prover gets values u, v by scaling and adding all the (N) leaves of the GGM tree. Whereas, the verifier is given all-but-one leaves of the GGM tree (this can be done efficiently since GGM tree is a puncturable PRF). The verifier can then compute the value q by scaling and adding $(N - 1)$ leaves, while the index i^* serves as Δ. Since scaling and adding is a linear operation, this method results in prover and verifier obtaining the desired VOLE correlation.

In practice, the GGM tree is created by the prover and verifier selects the index i^* which serves as Δ. The prover then sends the relevant seeds (path from GGM tree) to the verifier so that it receives all the leaves except the i^*-th leaf, from which the verifier can compute q. Note that since this reveals the value Δ to the prover, this step is only done after the prover has computed and committed to VOLE correlations proving the AES circuit.

Another optimization used by FAEST facilitates the AES proof part using only single VOLE correlation (say u_1) instead of τ correlations, however this requires the prover to prove the consistency of this proof with remaining $\tau - 1$ correlations. This requirement of proving that all the τ indepedent VOLE correlations are generated honestly using the GGM trees and they are consistent with each other requires an additional round in the proof system, therefore the protocol is a 7-round protocol.

Following algorithms are used in the protocol:

- UniversalHash: Used to prove the consistency of the τ VOLE instances efficiently.
- ExtendWitness: Extends the secret key sk to VOLE witness w.
- Lines 5 to 18 of P_3 in Figure 19 computes the AES proof using the QuickSilver protocol. The universal hash ZKHash masks the information related to the AES circuit when providing extra information required to prove the computation of multiplication gates in the circuit.
- PartialOpen: This refers to opening all-but-one leaves of the GGM tree.
- VOLEReconstruct: Reconstructs the value q from masked witness d sent by the prover and random challenge ch_3 generated by the verifier after receiving all-but-one leaves of the GGM tree. Specifically, the values Δ and q are generated by running all deterministic operations such as computing hash functions and PRG on the inputs.
- VOLECorrect: Used to check the consistency of all τ VOLE correlations.
- AESVerify: Runs the verification steps of QuickSilver protocol to check the computation of AES circuit.

For details, refer to the original specification [BBd+23a].

[16] VOLE is abbreviation of Vector Oblivious Linear Evaluation.

[17] Version 1.1 is available at https://faest.info/

$\underline{\mathsf{P}_1(sk) \text{ for } \mathsf{FAEST}}$

Choose $\mathsf{salt}, \mathsf{mseed}$ at random

Sample $(\mathsf{seed}_i)_{i\in[\tau]} := \mathsf{PRG}_1(\mathsf{salt}, \mathsf{mseed})$

`//Generate VOLE secrets and tags`

for $i = 1$ *to* τ **do**

  Compute $(\mathsf{com}_i, dec_i, u_i, V_i,)$ from salt and seed_i using length-doubling PRG

$V := [V_0\, V_1\, \cdots\, V_\tau]$

$u := u_1$

for $i = 1$ *to* $\tau - 1$ **do**

  Compute $c_{i+1} := u \oplus u_i$

`//Commit to VOLE secrets, tags, and commitments`

$h_{\mathsf{com}} := \mathsf{H}_1(\mathsf{com}_1 \| \mathsf{com}_2 \| \cdots \| \mathsf{com}_\tau)$

$a_1 := \left(h_{\mathsf{com}}, (c_{i+1})_{i\in[\tau-1]}\right)$

$\mathsf{state}_1 := \left(\mathsf{salt}, h_{\mathsf{com}}, (c_{i+1})_{i\in[\tau-1]}, (dec_i)_{i\in[\tau]}, u, V\right)$

$\mathsf{state} := (\mathsf{state}_1)$

return a_1 and state

$\underline{\mathsf{P}_2(sk, ch_1, \mathsf{state}) \text{ for } \mathsf{FAEST}}$

parse $\mathsf{state} = (\mathsf{state}_1)$

parse $\mathsf{state}_1 = \left(\mathsf{salt}, h_{\mathsf{com}}, (c_{i+1})_{i\in[\tau-1]}, (dec_i)_{i\in[\tau]}, u, V\right)$

`//Universal hash for VOLE consistency`

$\tilde{u} := \mathsf{UniversalHash}(ch_1, u)$

$\tilde{V} := \mathsf{UniversalHash}(ch_1, V)$

$h_V := \mathsf{H}_1(\tilde{V})$

`//Mask witness and generate VOLE MACs for` w

$w := \mathsf{ExtendWitness}(sk)$

$d := w \oplus u$

$a_2 := (\tilde{u}, h_V, d)$

$\mathsf{state}_2 := (w, \tilde{u}, d)$

$\mathsf{state} := (\mathsf{state}_1, \mathsf{state}_2)$

return a_2 and state

$\underline{\mathsf{P}_3(sk, ch_2, \mathsf{state}) \text{ for } \mathsf{FAEST}}$

parse $\mathsf{state} = (\mathsf{state}_1, \mathsf{state}_2)$

parse $\mathsf{state}_1 = \left(\mathsf{salt}, h_{\mathsf{com}}, (c_{i+1})_{i\in[\tau-1]}, (dec_i)_{i\in[\tau]}, u, V\right)$

parse $\mathsf{state}_2 = (w, \tilde{u}, d)$

`//Prove` $C(w) = 1$ `for AES circuit` C `using` u, V, w

for *each gate* $g \in C$ **do**

  `//`w_θ, w_ϕ `are input wires and` w_η `is the output`

  if g *is linear* **then**

    `//`p, q, r `are coefficients of the linear function`

    $w_\eta := p \cdot w_\theta \oplus q \cdot w_\phi \oplus r$

    $v_\eta := p \cdot v_\theta \oplus q \cdot v_\phi$

  if g *is multiplicative* **then**

    `//`m_g `be unique identifier for` g

    $w_\eta := w_\theta \cdot w_\phi$

    $d_{m_g} := w_\eta \oplus u_{m_g}$

    `//Generate multiplication checking tags`

    $a_{m_g} := v_\theta \cdot v_\phi$

    $b_{m_g} := w_\theta \cdot v_\phi \oplus w_\phi \cdot v_\theta \oplus v_\eta$

`//Compress multiplication check tags in ZK`

$\widehat{a} := \{ a_{m_g} \}$

$\widehat{b} := \{ b_{m_g} \}$

$\widehat{d} := \{ d_{m_g} \}$

$\tilde{a} := \mathsf{ZKHash}(ch_2, \widehat{a})$

$\tilde{b} := \mathsf{ZKHash}(ch_2, \widehat{b})$

$a_3 := \left(\tilde{a}, \tilde{b}\right)$

$\mathsf{state}_3 := (\tilde{a})$

$\mathsf{state} := (\mathsf{state}_1, \mathsf{state}_2, \mathsf{state}_3)$

return a_3 and state

$\underline{\mathsf{P}_4(sk, ch_3, \mathsf{state}) \text{ for } \mathsf{FAEST}}$

parse $\mathsf{state} = (\mathsf{state}_1, \mathsf{state}_2, \mathsf{state}_3)$

parse $\mathsf{state}_1 = \left(\mathsf{salt}, h_{\mathsf{com}}, (c_{i+1})_{i\in[\tau-1]}, (dec_i)_{i\in[\tau]}, u, V\right)$

parse $\mathsf{state}_2 = (w, \tilde{u}, d)$

parse $\mathsf{state}_3 = (\tilde{a})$

`//Generate partial decommitments for VOLE`

for $i = 1$ *to* τ **do**

  $pdec_i := \mathsf{PartialOpen}(ch_3, dec_i)$

$a_4 := \left((c_{i+1})_{i\in[\tau-1]}, \tilde{u}, d, \tilde{a}, (pdec_i)_{i\in[\tau]}, \mathsf{salt}\right)$

return a_4

Fig. 19. Prover algorithms for $\mathsf{ID}_{\mathsf{FAEST}}$.

$\underline{\mathsf{Rep}(vk, ch_1, ch_2, ch_3, a_4)}$

Parse $a_4 = \left((c_{i+1})_{i\in[\tau-1]}, \tilde{u}, d, \tilde{a}, (pdec_i)_{i\in[\tau]}, \mathsf{salt}\right)$

`//Reconstruct VOLE correlations`

Compute $\left(\overline{h}_{\mathsf{com}}, Q'\right) :=$ $\mathsf{VOLEReconstruct}(ch_3, (pdec_i)_{i\in[\tau]}, \mathsf{salt})$

$\overline{a}_1 := \left(\overline{h}_{\mathsf{com}}, (c_{i+1})_{i\in[\tau-1]}\right)$

`//Apply VOLE corrections`

$(Q, \overline{D}) := \mathsf{VOLECorrect}\left(ch_3, \tilde{u}, (c_{i+1})_{i\in[\tau-1]}, Q'\right)$

$\overline{Q} := \mathsf{UniversalHash}(ch_1, Q)$

$\overline{h}_V := \mathsf{H}_1\left(\overline{Q} \oplus \overline{D}\right)$

$\overline{a}_2 := \left(\tilde{u}, \overline{h}_V, d\right)$

`//Verify AES relation`

$\overline{b} := \mathsf{AESVerify}(d, \overline{Q}, ch_2, ch_3, \tilde{a}, vk)$

$\overline{a}_3 := \left(\tilde{a}, \overline{b}\right)$

return $(\overline{a}_1, \overline{a}_2, \overline{a}_3)$

$\underline{\mathsf{V}(vk, a_1, ch_1, a_2, ch_2, a_3, ch_3, a_4)}$

Compute $(\overline{a}_1, \overline{a}_2, \overline{a}_3) := \mathsf{Rep}(vk, ch_1, ch_2, ch_3, a_4)$

return $\mathsf{boole}((\overline{a}_1, \overline{a}_2, \overline{a}_3) = (a_1, a_2, a_3))$

$\underline{\mathsf{Sim}_{\mathsf{FAEST}}(vk, ch_1, ch_2, ch_3)}$

Choose salt, mseed at random

Sample $(\mathsf{seed}_i)_{i\in[\tau]} := \mathsf{PRG}_1(\mathsf{salt}, \mathsf{mseed})$

`//Generate VOLE secrets and tags`

for $i = 1$ *to* τ **do**

  Compute $(\mathsf{com}_i, dec_i, u_i, V_i,)$ from salt and seed_i using length-doubling PRG

$V := [V_0\, V_1 \cdots V_\tau]$

$u := u_1$

for $i = 1$ *to* $\tau - 1$ **do**

  Compute $c_{i+1} := u \oplus u_i$

`//Commit to VOLE secrets, tags, and commitments`

$h_{\mathsf{com}} := \mathsf{H}_1(\mathsf{com}_1 \| \mathsf{com}_2 \| \cdots \| \mathsf{com}_\tau)$

$a_1 := \left(h_{\mathsf{com}}, (c_{i+1})_{i\in[\tau-1]}\right)$

Choose d uniform randomly

Set $w := d \oplus u$

Compute Δ from ch_3

Set $V := V + d\Delta$

`//Universal hash for VOLE consistency`

$\tilde{u} := \mathsf{UniversalHash}(ch_1, u)$

$\tilde{V} := \mathsf{UniversalHash}(ch_1, V)$

$h_V := \mathsf{H}_1(\tilde{V})$

$a_2 := (\tilde{u}, h_V, d)$

Prove $C(w) = 1$ using u, V, w as in P_3

$a_3 := \left(\tilde{a}, \tilde{b}\right)$

`//Generate partial decommitments for VOLE`

for $i = 1$ *to* τ **do**

  $pdec_i := \mathsf{PartialOpen}(ch_3, dec_i)$

$a_4 := \left((c_{i+1})_{i\in[\tau-1]}, \tilde{u}, \widehat{d}, \tilde{a}, (pdec_i)_{i\in[\tau]}, \mathsf{salt}\right)$

return a_1, a_2, a_3 and a_4

Fig. 20. Reconstruction, verification, and simulation algorithms for $\mathrm{ID}_{\mathsf{FAEST}}$.

Security of FAEST Signature

Lemma H.1 **(q_S-HVZK).** *Suppose that* PRG *is a length-doubling secure PRG and* H_1 *is a hash function modelled as random oracle. Let* q_S *be a polynomial of* 1^κ*. Then,* $\mathsf{ID}_{\mathsf{FAEST}}$ *with simulator* $\mathsf{Sim}_{\mathsf{FAEST}}$ *in Figure 20 is* q_S*-HVZK.*

Proof. The length-doubling PRG PRG is used to implement the ideal functionality $\mathcal{F}_{\mathsf{VOLE}}$ using the GGM trees. The rest of the proof follows from the proof for the malicious verifier case from SoftSpoken [Roy22] and QuickSilver [YSWW21] protocols, as explained in [BBD+23b].

Lemma H.2 **(Strong Non-divergency).** *Suppose that hash functions* H_0 *and* H_1 *are collision resistant. Then,* $\mathsf{ID}_{\mathsf{FAEST}}$ *is strongly-non-divergent with respect to* $\mathsf{Sim}_{\mathsf{FAEST}}$*.*

Proof. Let a legitimate transcript be $(a_1, ch_1, a_2, ch_2, a_3, ch_3, a_4)$ and let the adversary's transcript be $(a_1, ch_1, a_2', ch_2', a_3', ch_3', a_4')$. Recalling the conditions from Definition 3.3, we have following cases

1. $ch_3 \neq ch_3'$ (condition (2*a*) of Definition 3.3.)
2. $(a_1, ch_1, a_2, ch_2, a_3, ch_3) = \left(a_1, ch_1, a_2', ch_2', a_3', ch_3'\right)$ and $a_4 \neq a_4'$ (condition (2*b*) of Definition 3.3.)

When $ch_3 \neq ch_3'$*:* In this case, let $\overline{h}_{\mathsf{com}}$ and $\overline{h'}_{\mathsf{com}}$ be the values recovered by running $\mathsf{VOLEReconstruct}$ with inputs ch_3 and ch_3' respectively. Then if $\overline{h}_{\mathsf{com}} = \overline{h'}_{\mathsf{com}}$, we have found a collision (during internal computation of $\mathsf{VOLEReconstruct}$) for the hash function H_1. Otherwise, there is a contradiction since $a_1 \neq a_1'$.

When $a_4 \neq a_4'$*:* Note that, in this case $(a_1, a_2, a_3) = \left(a_1, a_2', a_3'\right)$ therefore the only possible case is $\left((pdec_i)_{i\in[\tau]}, \mathsf{salt}\right) \neq \left((pdec_i')_{i\in[\tau]}, \mathsf{salt}'\right)$. If $(pdec_i)_{i\in[\tau]} \neq (pdec_i')_{i\in[\tau]}$, then again during computation of $\overline{h}_{\mathsf{com}}$ and $\overline{h'}_{\mathsf{com}}$ from $\mathsf{VOLEReconstruct}$ with inputs $(pdec_i)_{i\in[\tau]}$ and $(pdec_i')_{i\in[\tau]}$ respectively we can find a collision (during the internal computation of $\mathsf{VOLEReconstruct}$) for either the hash function H_1 or H_0. Similarly, when $\mathsf{salt} \neq \mathsf{salt}'$ we can find a collision for the hash function H_0 while computing $\overline{h}_{\mathsf{com}}$ and $\overline{h'}_{\mathsf{com}}$ from $\mathsf{VOLEReconstruct}$ with inputs salt and salt' respectively.

□

Since Rep is decomposable, we can obtain signature scheme $\mathsf{FAEST} = \mathsf{FS}_{\mathrm{h,last}}[\mathsf{ID}_{\mathsf{FAEST}}, \mathsf{H}, \boldsymbol{\gamma}]$ as follows: [18]

- Let H be a random oracle.
- Let $\boldsymbol{\gamma} := (\gamma_1, \gamma_2, \gamma_3)$, where γ_i is identity function for $i \in \{1, 2, 3\}$.
- For message μ, compute $M := \mathsf{H}(\mathtt{0x01}, vk, \mu)$.
- Set $\mathsf{aux}_1 := (\mathtt{0x02}, \mathtt{0x01}, M, \mathsf{salt})$ and $h_1 := \mathsf{H}(\mathsf{aux}_1, a_1)$.
- Set $\mathsf{aux}_2 := (\mathtt{0x02}, \mathtt{0x02})$ and $h_2 := \mathsf{H}(\mathsf{aux}_2, h_1, a_2)$.
- Set $\mathsf{aux}_3 := (\mathtt{0x02}, \mathtt{0x03})$ and $h_3 := \mathsf{H}(\mathsf{aux}_3, h_2, a_3)$.

As the scheme is HVZK and strongly non-divergent, we get the following theorem:

Theorem H.1. *Suppose that* $\mathsf{FAEST} = \mathsf{FS}_{\mathrm{h,last}}[\mathsf{ID}_{\mathsf{FAEST}}, \mathsf{H}, \boldsymbol{\gamma}]$ *is* EUF-NMA*-secure in the (Q)ROM,* PRG *is length-doubling PRG,* $\mathsf{UniversalHash}$*,* ZKHash *are hiding universal hashes Then,* FAEST *is* sEUF-CMA*-secure in the (Q)ROM.*

Assuming that H is a random oracle (and therefore collision-resistant) we get that aux is also collision-resistant with respect to the message and verification key on index 1, therefore FAEST is M-S-UEO secure and MBS secure following Lemma 5.2.

Let us discuss the wNR security of FAEST. Because of Corollary B.2, it is enough to show that a variant $\mathsf{FAEST}' := \mathsf{FS}_{\mathrm{h}}[\mathsf{ID}_{\mathsf{FAEST}}, \mathsf{H}, \boldsymbol{\gamma}]$ is wNR-secure if H is the random oracle. We can show this by modifying the wNR security proof for FS_{h} in subsection A.3 as follows.

- G_0: This is the original wNR game with FAEST'.
- G_1: In this game, if the adversary outputs $vk' \neq vk$ such that $M = \mathsf{H}(\mathtt{0x01}, vk, \mu) = \mathsf{H}(\mathtt{0x01}, vk', \mu)$, then the adversary loses. Since we have a collision $(\mathtt{0x01}, vk, \mu) \neq (\mathtt{0x01}, vk', \mu)$ for H, this modification is justified by the fact that random oracle H is collision resistant.
- G_2: We skip this game.
- G_3: Before giving vk and σ to the adversary, we reprogram the point $(\mathtt{0x01}, vk, \mu)$ with random value M. As in the wNR security proof in subsection A.3, we can invoke the O2H lemma and the difference is at most $1/|\mathcal{M}|$.
- G_4: Next, we filter the random oracle H by reprogramming the values on the points $(\mathtt{0x01}, \cdot, \mu)$ with $\perp$. Since the adversary cannot obtain any information of the hash value $M' = \mathsf{H}(\mathtt{0x01}, vk', \mu)$, the winning probability is at most $1/|\mathcal{H}|$. As in the wNR security proof in subsection A.3, we can invoke the O2H lemma and the difference between G_3 and G_4 is at most $1/|\mathcal{M}|$.

[18] We introduce $\mathtt{0x01}, \mathtt{0x02}, \mathtt{0x03}$ in the computation of h_i values to split domains while the original specification implicitly did it by the length of inputs and outputs.