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)
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. After they selected three digital signature schemes in July 2022, they started an additional PQC signature standardization in Sept. 2022 [NIS22]. 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) 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: AdMer [KCC+23], Biscuit [BKPV23], FAEST [BBD+23a], MIRA [ABB+23c], MiRiT [ARV+23], MQOM [FR23], PERK [ABB+23a], RYDE [ABB+23b], and SDiH [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 signatures 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 encryption [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, Lyu99]. 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]. See e.g., [AFLT16, KLS18, DGLJ21, 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 [SSH1], 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 to alter the verification of a signature under multiple messages.4 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.

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 ≠ 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 $2^{-κ}$ with a number of queries $2^κ$, where $κ \in \{128, 192, 256\}$ is the security parameter. "?$" implies that showing the security is an open problem.

<table>
<thead>
<tr>
<th>Name</th>
<th>sEUF</th>
<th>MS-UEO</th>
<th>MBS</th>
<th>wNR</th>
<th>Section</th>
<th>Ref.</th>
<th>version</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIMer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Section F</td>
<td>[KCC+23]</td>
<td>v1.0</td>
</tr>
<tr>
<td>Biscuit</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>Section 6</td>
<td>[BKPV23]</td>
<td>v1.1</td>
</tr>
<tr>
<td>FAEST</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>subsection H.1</td>
<td>[BBD+23a]</td>
<td>v1.1</td>
</tr>
<tr>
<td>MIRA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>subsection G.1</td>
<td>[ABB+23c]</td>
<td>v1.0</td>
</tr>
<tr>
<td>MiRitH</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Section D</td>
<td>[ARV+23]</td>
<td>v1.0</td>
</tr>
<tr>
<td>MQOM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>subsection G.3</td>
<td>[FR23]</td>
<td>v1.0</td>
</tr>
<tr>
<td>PERK</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Section E</td>
<td>[ABB+23a]</td>
<td>v1.1</td>
</tr>
<tr>
<td>RYDE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>subsection G.1</td>
<td>[ABB+23b]</td>
<td>v1.0</td>
</tr>
<tr>
<td>SDiH</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>subsection G.2</td>
<td>[AFG+23]</td>
<td>v1.1</td>
</tr>
</tbody>
</table>

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+21] studied BUFF securities of seventeen signature schemes based on code, isogeny, lattice, or MQ in forty Round-1 candidates of NIST PQC additional 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 $(a_1, c_1, ..., a_n, c_n, a_{n+1})$ denote a transcript of the underlying ID scheme, where $a_1, ..., a_{n+1}$ are the messages generated by the prover and $c_1, ..., c_n$ be public-coin challenges generated by the verifier. On a message $\mu$, the signer sequentially computes the prover’s messages $a_1, ..., a_{n+1}$ by computing the challenges as $c_1 = H(\mu, a_1)$ and $c_i = H(c_{i-1}, a_i)$ for $i = 2, ..., n$, where $H$ is the random oracle, and outputs a signature $(a_1, a_2, ..., a_{n+1})$. The verifier verifies the transcript $(a_1, c_1, ..., a_n, c_n, a_{n+1})$ via the ID’s verification algorithm by computing $c_1 = H(\mu, a_1)$ and $c_i = H(c_{i-1}, a_i)$ for $i = 2, ..., n$.

**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:

- $G_1$: This is the original EUF-CMA game. The adversary is given $\nu k$ and has access to the signing oracle.
  The signing oracle on input $\mu$ computes $a_1, c_1 := H(\mu, a_1)$, and $a_2$, and returns $(a_1, a_2)$ as a signature. The adversary outputs $\mu'$ and a signature $(a_1', a_2')$. If it is valid and $\mu'$ is new, then the adversary wins.
- G₁: In this game, the challenge \( c_i \) is chosen uniformly at random, and the random oracle is reprogrammed, that is, the hash value of \((\mu, a_i)\) is overwritten by \( c_i \). This modification is justified by the adaptive reprogramming technique [GHHM21].

- G₂: 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 \( \mu' \) never signed before. The challenge \( c_i \) is computed as \( H(\mu, a_i) \) and this point \((\mu', a_i')\) in the random oracle \( H \) is never reprogrammed by the challenger since \( \mu' \) is new. Therefore, we can construct an EUF-NMA adversary against the signature scheme using the adversary in G₂.

**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 \( \mu' \) and a signature \((a'_i, a'_2)\) such that \((\mu', (a'_1, a'_2))\) is not answered by the signing oracle. Therefore, the adversary can ask \( \mu' \) to the signing oracle. Hence, the point \((\mu', a'_i)\) might be reprogrammed since \( \mu' \) can be queried to the signing oracle. To eliminate this event, we consider an additional game G₃ defined as follows:

- G₃: In this game, the adversary loses if the signing oracle returned answer \((a'_i, a_i)\) with \( a'_i \neq a_i \) on the query \( \mu' \).

In G₃, 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 Schaffner [KLS18] defined computational unique response (CUR), in which the adversary is given a property. In addition, even this stronger condition 1 is problematic in the context of the underlying ID protocols of the MPCih 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 $G_2$, 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 $c_3 ≠ c'_3$, 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 $a_3 ≠ a'_3$ 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 ID3 as Aguilar-Melchor, Hülsing, Joseph, Majenz, Ronen, and Yue [AHJY23], in which the first prover computes $w = (a_1, a_2)$ by computing $c_1 := H(\mu, a_1)$ by itself, the verifier chooses a random challenge $c = c_2$, the second prover computes $z = a_1$, and the verifier checks if $H(\mu, a_2, c_1, a_2, c_2, a_3)$ by computing $c_1 := H(\mu, a_1)$. Applying the Fiat-Shamir transform to ID3, we obtain the signature scheme FS3 [ID3, H], where the signer will compute $w = (a_1, a_2), c = H(\mu, w) = H(\mu, a_2, a_3)$, and $z = a_1$, 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 Grillo et al. [GHMM21, 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 $s$EUF-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 IDE because the proposed scheme SDiTh is defined as a variant of FS3 [ID5, H], where $c_1 = H(\mu, a_1)$ and $c_3 = H(\mu, c_1, a_3)$ (see Table 2 and subsection G.2 for the details) and there is a subtle gap on how to compute $c_2 (H(\mu, a_2, a_3) or H(\mu, c_1, a_3)).$ When we prove the $s$EUF-CMA security of the real signature as the security proof by Grillo et al. [GHMM21, 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 $\mu'$ while the signing oracle generates a signature $(a_1, c_1, a_2, c_2, a_3)$ on $\mu$ and $(a'_1, c'_1) ≠ (a_1, c_1).$ In this case, the forgery involves the point $(c_1, a_3)$ 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 $\mu,$ e.g., $c_2 := H(\mu, c_1, a_3),$ to exclude the event the point $(\mu', c'_1, a'_3)$ is not reprogrammed by the signing oracle. Hence, we will require a few arguments if $c_2$ does not involve $\mu$ directly as AIMer and 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, CDF21]. A signature of all signature schemes containing the hash values involving a message $\mu.$ Hence, a weak version of exclusive ownership (S-DEO) and MBS are easily satisfied. If those hash values include a verification key $ok$ (or, 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 $\mu$ and $ok$ 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 $\mu$ and $ok$ as in Table 2. We then examine Biscuit and MiRItH where $ok$ is not involved in the hash values. Curiously, we find that Biscuit and MiRItH are vulnerable to S-CEO and M-S-UEO. Very roughly speaking, we propose an attack computing a new verification key $ok'$ 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 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 $\mu$ 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 $a_i$ or $a_t$ contains salt.

<table>
<thead>
<tr>
<th>Name</th>
<th>#pass $h_1$ or $c_1$</th>
<th>$h_2$ or $c_2$</th>
<th>$h_3$ or $c_3$</th>
<th>$\sigma$</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AImer</td>
<td>5</td>
<td>$\mu, vk, a_1$</td>
<td>$h_1, a_2$</td>
<td>-</td>
<td>[KCC+23]</td>
</tr>
<tr>
<td>Biscuit</td>
<td>5</td>
<td>salt, $h_1, a_2$</td>
<td>-</td>
<td>$(h_1, h_2, a_1)$</td>
<td>[BKPV23]</td>
</tr>
<tr>
<td>FAEST</td>
<td>7</td>
<td>salt, $H(\mu, vk)$, $a_1$, $c_1$, $a_2$</td>
<td>$c_2, a_3$</td>
<td>$(h_1, a_4)$</td>
<td>[BBd+23a]</td>
</tr>
<tr>
<td>MIRA</td>
<td>5</td>
<td>salt, $H(\mu)$, $vk$, $a_1$, salt, $H(\mu)$, $vk$, $h_1, a_2$</td>
<td>-</td>
<td>$(h_1, h_2, a_1)$</td>
<td>[ABB+23c]</td>
</tr>
<tr>
<td>MiRitH</td>
<td>5</td>
<td>salt, $\mu$, $a_1$</td>
<td>salt, $\mu$, $h_1, a_2$</td>
<td>-</td>
<td>$(h_1, h_2, a_1)$</td>
</tr>
<tr>
<td>MQOM</td>
<td>7</td>
<td>salt, $\mu, vk, a_1$</td>
<td>salt, $\mu, h_2, a_3$</td>
<td>$(h_1, h_2, h_1, a_1)$</td>
<td>[FR23]</td>
</tr>
<tr>
<td>PERK</td>
<td>5</td>
<td>salt, $\mu, vk, a_1$</td>
<td>salt, $\mu, vw, a_1$, $h_1, a_2$</td>
<td>-</td>
<td>$(h_1, h_2, a_1)$</td>
</tr>
<tr>
<td>RYDE</td>
<td>5</td>
<td>salt, $H(\mu)$, $vk$, $a_1$, salt, $H(\mu)$, $vk$, $h_1, a_2$</td>
<td>-</td>
<td>$(h_1, h_2, a_1)$</td>
<td>[ABB+23b]</td>
</tr>
<tr>
<td>SDitH</td>
<td>5 (3)</td>
<td>salt, $vk, a_1$</td>
<td>salt, $\mu, h_1, a_2$</td>
<td>-</td>
<td>$(h_1, h_2, a_1)$</td>
</tr>
</tbody>
</table>

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 $x \in \mathbb{Z}^+$. We use the standard O-notations. For $n \in \mathbb{Z}^+$, we let $[n] := \{1, \ldots, n\}$. For a statement $P$, boole$(P)$ denotes the truth value of $P$. DPT and QPT stand for deterministic polynomial time and quantum polynomial time, respectively.

Let $X$ and $H$ be two finite sets. Func$(X, H)$ denotes a set of all functions whose domain is $X$ and codomain is $H$.

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 $S$, $U(S)$ denotes the uniform distribution over $S$. We often write "$x \leftarrow S$" instead of "$x \leftarrow U(S)$." If $\text{inp}$ is a string, then "$\text{out} \leftarrow \mathcal{A}(\text{inp})$" denotes the output of algorithm $\mathcal{A}$ running on input $\text{inp}$ with an access to a set of oracles $\mathcal{O}$. If $\mathcal{A}$ and oracles are deterministic, then $\text{out}$ is a fixed value and we write "$\text{out} := \mathcal{A}(\text{inp})$." We also use the notation "$\text{out} := \mathcal{A}(\text{inp}, r)$" to make the randomness $r$ of $\mathcal{A}$ explicit.

For a function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$, a quantum access to $f$ is modeled as oracle access to unitary $O_f : \ket{x} \ket{y} \mapsto \ket{x} \ket{f(x)}$ by convention, we will use the notation $A^{f,x}$ to stress $\mathcal{A}$’s quantum and classical access to $f$ and $g$. For a function $f : X \rightarrow H$, 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 $H : X \rightarrow Y$ be a random function. Then any algorithm that makes $q$ quantum queries to $H$ outputs a collision for $H$ with probability at most $632(q + 1)^2/|Y|$.\footnote{The constant $632 > 24 \pi^2 / 3$ is taken from $C = 24 \pi^2$ in the proof of [Zha15, Thm.3.1] for general $X$ and $Y$ with $\#X > \#Y$ and $C' = \pi^2 / 3$ in [Zha12, Cor.7.5].}

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 $X_1$, $X_2$, and $H$ be finite sets. Let $A$ be an adversary that makes $R$ queries to REPROGRAM and $q$ quantum queries to $O_H$. Then, the distinguishing advantage of $A$ is bounded by

$$\left| \Pr[\text{Repro}_0 = 1] - \Pr[\text{Repro}_1 = 1] \right| \leq \frac{3R}{2} \sqrt{\frac{q}{|X_1|}}.$$

where Repro$_0$ and Reprogram is defined in Figure 1.
A digital signature scheme \( DS \) consists of the following triple of PPT algorithms \((\text{Gen}, \text{Sign}, \text{Vrfy})\):

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

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

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

for some negligible function \( \delta \).

**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 \((\mu', \sigma')\) is not equal to the pairs returned by \( \text{Sign} \). The formal definition follows:

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

\[
\text{Adv}_{DS,A}^{\text{cma}}(\kappa) := \Pr[\text{Exp}_{DS,A}^{\text{cma}}(1^s) = 1],
\]

where \( \text{Exp}_{DS,A}^{\text{cma}}(1^s) \) is an experiment described in Figure 2. For GOAL \( \in \{\text{EUF}, \text{sEUF}\} \), we say that \( DS \) is GOAL-CMA-secure if \( \text{Adv}_{DS,A}^{\text{cma}}(\kappa) \) is negligible for any QPT adversary \( A \).

For any \( A \), we define its euf-nma advantage against \( DS \) as \( \text{Adv}_{DS,A}^{\text{nma}}(\kappa) := \Pr[\text{Exp}_{DS,A}^{\text{nma}}(1^s) = 1] \), where \( \text{Exp}_{DS,A}^{\text{nma}}(1^s) \) is the game \( \text{Exp}_{DS,A}^{\text{nma}}(1^s) \) without the signing oracle \( \text{Sign} \). We say that \( DS \) is EUF-NMA-secure if \( \text{Adv}_{DS,A}^{\text{nma}}(\kappa) \) is negligible for any QPT adversary \( A \).
As one of the advanced security notions, Cremers et al. [CDF+21] defined 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, DHF+24].

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

$$Adv_{DS,A}^{goal}(k) := \Pr[Exp_{DS,A}^{goal}(1^k) = 1],$$

where $Exp_{DS,A}^{goal}(1^k)$ 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 $Adv_{DS,A}^{goal}(k)$ is negligible for any QFT adversary $A$.

### 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 $C$. The syntax follows:

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

- $Gen(1^k) \rightarrow (sk, sk)$: a key-generation algorithm that takes $1^k$ as input, where $k$ is the security parameter, and outputs a pair of keys $(sk, sk)$.
- $P(sk, c_{i-1}, state) \rightarrow (a_i, state)$: a prover algorithm that, in the $i$-th round $(i = 1, \ldots, n + 1)$, takes signing key $sk$, the $(i-1)$-th challenge $c_{i-1}$, and state as input, and outputs the $(2i-1)$-th message $a_i$ and state.
- $V(\tilde{a}_0, a_1, \ldots, a_{n}, c_{n, a_{n+1}}) \rightarrow true/false$: a verification algorithm that takes verification key $\tilde{a}_0$ and the transcript $a_1, c_1, \ldots, a_n, c_n, a_{n+1}$ as input and outputs its decision true or false.

We assume that a verification key $\tilde{a}_0$ defines the challenge spaces $C_1, \ldots, C_n$. 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:
Definition 2.5 (Commitment entropy [KLS18, Def. 2.6], adapted). We say that \(\mathbf{2}\) (n + 1)-pass ID scheme ID has a-commitment entropy if for any \((ok, sk)\) generated by Gen\((1^n)\), \(\text{adv}(a, a_1) \sim \mathcal{P}(sk)\) satisfies

\[
\Pr[\mathcal{E}_{ID, A}(1^n) = 1] - \Pr[\mathcal{E}_{ID, A}(1^n) = 1] = \Omega(\epsilon)
\]

where \(\mathcal{E}_{ID, A}(1^n)\) is defined as in Figure 4. We say that ID is q-HVZK if \(\mathcal{E}_{ID, A}(1^n)\) is negligible for any QPT adversary \(A\).

3 Unique Response and Non-Divegency

We say that three-pass ID scheme ID has unique responses if for all \(a_i\), there exists at most one \(a_2\) satisfying \(\forall (ok, a_i, a_1) = \text{true}\). 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 = (Gen, P, V) has the computational unique response (CUR) property if for any QPT adversary \(A\), its advantage defined below is negligible in \(\kappa\) :

\[
\mathcal{Adv}_{ID, A}^{\text{cur}}(\kappa) := \Pr[(ok, sk) \sim \text{Gen}(1^n), (a_i, c_1, a_2, a_1) \sim A(\mathfrak{o})] - \Pr[a_1 \neq a_2 \land V(ok, a_i, c_1, a_2) \land V(ok, a_i, c_1, a_1) \sim A(\mathfrak{o})] = \Omega(\kappa).
\]

We can consider that the two transcripts \((a_1, c_1, a_2)\) and \((a_2, c_1, a_1)\) 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 = (Gen, P, V) has the computational unique response (CUR) property if for any QPT adversary \(A\), its advantage defined below is negligible in \(\kappa\) :

\[
\mathcal{Adv}_{ID, A}^{\text{cur}}(\kappa) := \Pr[(ok, sk) \sim \text{Gen}(1^n), (a_i, c_1, ... a_k) \sim A(\mathfrak{o})] - \Pr[\text{BranchCheck}_{DFM}(trans, trans^{'}) \land V(\mathfrak{ok}, trans^{'}) \land V(\mathfrak{ok}, trans)] = \Omega(\kappa),
\]

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

1. Parse trans = \((a_1, c_1, ... a_k, c_k)\) and trans\(\)' = \((a_k', c_k', ... a_1', c_1')\).
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 BranchCheck\((trans, trans')\) is defined as follows:

```plaintext
1. Expt_{ID, A}^{hvzk, 1}(1^n)
2. (ok, sk) \sim \text{Gen}(1^n)
3. for i \in [q] do
4. trans \sim \mathcal{P}(ok, sk), V(\mathfrak{ok})
5. b' \sim A(\mathfrak{ok}, (\text{trans}_1, ..., \text{trans}_n))
6. return b'
```

Fig. 4. The experiments for computational multi-HVZK.
Definition 3.6 (Strong Branch-checking algorithm).
Fix a branch-checking algorithm and strong non-divergency as follows:

For easiness, we define the stronger version of non-divergency by relaxing the conditions. We define another
Definition 3.7 (Strong non-divergency for BranchCheck algorithm)
where
Remark 3.2. We remark that the new condition (a) covers the conditions (a) and (c) in BranchCheck.

Definition 3.8 (Computational unique response for BranchCheck, trans).
By using this branch-checking algorithm, we can extend the CUR property for BranchCheck can choose two transcripts
where

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 \( F(x) = (f_1(x), \ldots, f_n(x)) \) be an \( m \)-quadratic functions in \( \mathbb{F}_q[x_1, \ldots, x_n] \), where \( f_j(x) = \sum a_{ij}^{(j)} x_i + \sum b_{ij}^{(j)} \) with \( a_{ij}^{(j)}, b_{ij}^{(j)} \in \mathbb{F}_q \) for \( l \in [m] \). A prover and a verifier have \((F, \psi)\) and the prover has a witness \( s \in \mathbb{F}_q^n \) satisfying \( F(s) = \psi \). Let \( \text{Com} \) be a commitment scheme. In the protocol, \( G(x, y) \) denotes \( F \)'s polar form, which is defined as \( G(x, y) := F(x+y) - F(x) - F(y) \). The protocol is defined as follows:

1. The prover chooses \( r_0, t_0 \leftarrow \mathbb{F}_q^m \) and \( e_0 \leftarrow \mathbb{F}_q^m \) uniformly at random. It computes \( r_1 := s - r_0 \). It sends \( \text{com}_0 := \text{Com}(r_0, t_0, e_0) \) and \( \text{com}_1 := \text{Com}(r_1, G(t_0, r_1) + e_0) \).
2. The verifier picks \( a \leftarrow \mathbb{F}_q \) and sends it.
3. The prover sends \( s := ar_0 - t_0 \) and \( e_1 := aF(r_0) - e_0 \).
4. The verifier picks \( b \leftarrow \{0, 1\} \) and sends it.
5. The prover sends \( r_s \).
6. The verifier outputs the result of the check defined as follows:
   - If \( b = 0 \), then check if \( \text{com}_0 = \text{Com}(r_0, ar_0 - t_0, aF(r_0) - e_1) \).
   - If \( b = 1 \), then check if \( \text{com}_1 = \text{Com}(r_1, aF(r_0)) - G(t_1, r_1) - e_1 \).

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

In \( \Pi_{\text{SSH}} \), 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 \( \Pi_{\text{SSH}} \) has computationally unique response.

Unfortunately, this argument is incorrect, and we can falsify it as follows: Let us construct the following adversary who knows \( \alpha \) corresponding to \( F \) and \( \psi \), because Definition 3.2 allows an adversary to produce \( \alpha \):

1. Set \( b = 1 \) and pick \( a \leftarrow \mathbb{F}_q \).
2. Pick \( r_0, t_0 \leftarrow \mathbb{F}_q^m \) and \( e_0 \leftarrow \mathbb{F}_q^m \). Compute \( r_1 := s - r_0 \). Compute \( \text{com}_0 := \text{Com}(r_0, t_0, e_0) \) and \( \text{com}_1 := \text{Com}(r_1, G(t_0, r_1) + e_0) \).
3. Compute \( t_1 := ar_0 - t_0 \) and \( e_1 := aF(r_0) - e_0 \). Compute \( t_2 \) and \( e_2 \) such that \( G(t_1, r_1) + e_1 = G(t_2, r_1) + e_2 \) by choosing \( t_2 \neq t_1 \) and setting \( e_2 = G(t_1, r_1) + e_1 - G(t_2, r_1) \).
4. Output \( \text{trans}_1 := ((\text{com}_0, \text{com}_1), \alpha, (t_1, e_1), 1, r_1) \).

The two transcripts are valid since the verifier checks if

\[
\text{com}_1 = \text{Com}(r_1, \alpha(v - F(r_1)) - G(t_1, r_1) - e_1) = \text{Com}(r_1, \alpha(v - F(r_1)) - G(t_2, r_1) - e_2).
\]

Since \( a_2 \neq a_1 \), they satisfy the criteria of \( \text{Check}_{\text{DSM}} \) in Definition 3.2 and this adversary breaks the CUR property. Even if we modify the definition so that \( \alpha \) 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 \( r_1 \) uniformly at random instead of setting \( r_1 := s - r_0 \).

We can modify the condition checking algorithm \( \text{Check}_{\text{DSM}} \) with \( \text{BranchCheck} \). If so, an adversary needs to output transcripts such that \( a_2 \neq a_1 \wedge c_2 \neq c_1 \) and the above attack does not work. However, if we consider the \( \tau \)-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 \( c_1 = (b'_1, \ldots, b'_\tau) \) 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 \( \alpha \), then one can use a more malicious simulator: For the index \( \tau \), the simulator first chooses \( a \leftarrow \mathbb{F}_q, r_0, r_1, t_1 \leftarrow \mathbb{F}_q^m \), and \( e_1 \leftarrow \mathbb{F}_q^m \), computes \( \text{com}_0 := \text{Com}(r_0, ar_0 - t_0, aF(r_0) - e_1) \) and \( \text{com}_1 := \text{Com}(r_1, aF(r_0)) - G(t_1, r_1) - e_1 \), and outputs \( a_\tau = (\text{com}_0, \text{com}_1), c_1 = a, a_2 = t_1, c_2 = 0, c'_2 = 1, \) and \( a_3 = r_0, a'_3 = r_1 \),

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 \( \text{ID} = (\text{Gen}, P, V) \) via the FS transform [EDV’12, DGV’16, CHR’16]. Let \( H : \{0, 1\}^* \rightarrow H \) and \( y_i : H \rightarrow C_i \), for \( i \in [n] \) be hash functions modeled as random oracles. The FS transform converts \( \text{ID} \) into a signature scheme \( \text{DS} = \text{FS}[\text{ID}, H, y] \).
Fig. 5. Scheme FS\textsubscript{eq}(ID, H, y) = (Gen, Sign\textsubscript{eq}, Vrfy\textsubscript{eq}), where ID = (Gen, P, V), H : \{0, 1\}^* \rightarrow H is modeled as the random oracle, and \( y : H \rightarrow C \) for \( i \in [n] \) is also modeled as the random oracle. For ease of notation, we let aux = aux\( (i,vk,\mu) \).

by computing \( i \)-th challenge \( c_i \) from a message \( \mu \), 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
\[
c_i := \begin{cases} 
H(1, i, v_k, a_i) & \text{if } i = 1 \\
H(i, c_{i-1}, a_i) & \text{if } i = 2, \ldots, n.
\end{cases}
\]

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:
\[
h_i := \begin{cases} 
H(\text{aux}_i, a_i) & \text{if } i = 1 \\
H(\text{aux}_i, h_{i-1}, a_i) & \text{if } i = 2, \ldots, n.
\end{cases}
\]

\begin{itemize}
\item[\( h_i \)]
\end{itemize}

where \( \text{aux}_i = \text{aux}(i, vk, \mu) \) is a value computed from \( \mu, vk, \) 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 \( \mu = \mu' \) then aux\( (i, vk, \mu) = aux(i, vk, \mu') \) holds (perfectly or computationally). We also use a similar property with respect to \( vk \) to discuss the M-S-UFO 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 \( A \), its advantage
\[
\text{Adv}_{\text{aux},\mu}(\lambda, \kappa) := \Pr \left[ \left( \begin{array}{c} (v_k, v_k', \mu, \mu') \\
\mu \neq \mu' \wedge \exists i \in [\lambda, n], \forall i \in [1, \ell], \text{aux}(i, v_k, \mu) = \text{aux}(i, v_k', \mu')
\end{array} \right) \right]
\]
is negligible in the security parameter \( \kappa \).

We say that aux is collision-resistant with respect to verification key on index \( \lambda \in [n] \) if for any QPT adversary \( A \), its advantage
\[
\text{Adv}_{\text{aux},\lambda}(\kappa) := \Pr \left[ \left( \begin{array}{c} (v_k, v_k', \mu, \mu') \\
v_k \neq v_k' \wedge \exists i \in [\lambda, n], \forall i \in [1, \ell], \text{aux}(i, v_k, \mu) = \text{aux}(i, v_k', \mu')
\end{array} \right) \right]
\]
is negligible in the security parameter \( \kappa \).

4.1 sEUF-CMA Security for FS\textsubscript{eq}

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

Theorem 4.1 (EUF-NMA \( \Rightarrow \) sEUF-CMA for FS\textsubscript{eq} on \((2n + 1)\)-pass ID). Let ID be a \((2n + 1)\)-pass ID scheme that has \( \alpha \)-commitment entropy. Let \( H : \{0, 1\}^* \rightarrow H \) and \( y_i : H \rightarrow C \) for \( i \in [n] \) be random oracles. Suppose that aux is collision-resistant with respect to message on index \( \lambda \). Let DS := FS\textsubscript{eq}(ID, H, y). For any quantum adversary \( A \) against the sEUF-CMA security of DS issuing at most \( q_y \) classical queries to the signing oracle and at most \( q_r \) and \( q_t \) quantum queries to the random oracles H and \( y \), there exist an adversary \( A_{\text{aux}} \) against the EUF-NMA
security of DS, an adversary \( A_{\text{forik}} \) against the \( q_5 \)-HVZK property of ID, an adversary \( A_{\text{cr}} \) against aux’s collision-resistance property with respect to message on index \( \lambda \), and an adversary \( A_{\text{id}} \) against the \( q_4 \)-non-divergency of ID, such that

\[
\text{Adv}_{DS, A}^{\text{seuf-cma}}(1^*) \\
\leq \text{Adv}^{\text{q-wrvk}}_{ID, A_{\text{forik}}}(1^*) + \text{Adv}_{DS, A_{\text{forik}}}^{\text{aux-mag}}(1^*) + \text{Adv}_{ID, A_{\text{id}}}^{\text{nd}}(1^*) \\
+ 632 \cdot (q_4 + n + 1)^{3} \cdot |H|^{-1} + 632 \sum_{i=1}^{S_1} (q_i + q_4 + 2)^{3} \cdot |C_i|^{-1} \\
+ \frac{3M}{2} \sqrt{(q_4 + q_5 + 2) \cdot |H|^{-1}},
\]

where \( \alpha_1 = \alpha \) and \( \alpha_2 = \ldots = \alpha_s = \lg(#H) \). The running times of \( A_{\text{forik}}, A_{\text{cr}}, A_{\text{id}} \), and \( A_{\text{id}} \) are approximately that of \( A \).

We prove this theorem by modifying the proof of [GHHM21, Thm.3]. We define eight games \( G_0, \ldots, G_7 \) in Figure 6. In \( G_1 \), we introduce an algorithm to check a collision of \( H \), denoted by CollCheck. In \( G_2 \), we additionally check a collision of \( y_i \)’s. Those two changes prohibit the adversary from converging 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 \( y_1, \ldots, y_5 \). We next modify the signing oracle to use the simulator instead of the proven algorithms in \( G_3 \). In \( G_5 \), we introduce 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 \( G_6 \), we again introduce 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

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

**Game \( G_1 \):** In this game, the challenger manages the list \( \mathcal{L} \) that contains the hash values and challenges that the signing oracle \( \text{Sign} \) produced. Receiving a message \( \mu \) and a signature \( (a_1, \ldots, a_{n_1}) \), the challenger runs CollCheck for \( G_1 \) (Figure 6) and, if it returns true, then the adversary loses.

If there is a difference between \( G_0 \) and \( G_1 \), CollCheck returns true. According to the definition of CollCheck for \( G_1 \), this means that there exists an entry \((aux, h_{i-1}, c_{i-1}, a_i, h_i, c_i)\) in \( \mathcal{L} \) such that \( (h_{i-1}, a_i) \neq (h_i, a_i) \). This implies a collision for \( H \) since we have \((aux, h_{i-1}, a_i) \neq (aux, h_i, a_i) \) but \( H[aux, h_{i-1}, a_i] = h_i = H[aux, h_i, a_i] \).

Since the number of queries to \( H \) is at most \( q_4 + n + 1 \), we have the following lemma by using Lemma 2.1.

**Lemma 4.1.** We have \( \Pr[W_0] - \Pr[W_1] \leq 632(q_4 + n + 1)^{3} \cdot |H|^{-1} \).

**Game \( G_2 \):** In this game, receiving the forgery \( \mu \) and \( (a_1, \ldots, a_{n_1}) \), the challenger runs CollCheck for \( G_2 - G_0 \) and, if it returns true, then the adversary loses.

Notice that the difference between those two CollChecks, \( \exists (aux, h_{i-1}, a_i, h_i, c_i) \in \mathcal{L} \) in \( G_1 \) and \( \exists (aux, h_{i-1}, a_i, h_i, c_i) \in \mathcal{L} \) in \( G_2 \). Thus, if there is a difference between \( G_0 \) and \( G_2 \), then there exists \((aux, h_{i-1}, a_i, h_i, c_i) \) in \( \mathcal{L} \) such that \( (h_{i-1}, a_i) \neq (h_i, a_i) \) and \( h_i \neq h_i^* \). This implies a collision for \( y \), since we have \( h_i \neq h_i^* \) but \( y(h_i) = c_i^* = y(h_i^*) \).

Since the number of queries to \( y \) is \( q_4 + q_5 + 1 \), applying Lemma 2.1 to \( y_1, \ldots, y_n \), we have the following lemma:

**Lemma 4.2.** We have \( \Pr[W_0] - \Pr[W_1] \leq \sum_{i=1}^{S_1} 632(q_i + q_5 + 2)^{3} \cdot |C_i|^{-1} \).

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

**Lemma 4.3.** We have \( \Pr[W_0] - \Pr[W_1] \leq \sum_{i=1}^{S_1} \frac{3M}{2} \sqrt{(q_i + q_4 + 1) / 2^{n_i}} \).

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 \( y_i \) by choosing \( c_i \leftarrow C_i \) as in L.9–10 of \( \text{Sign} \). Applying Lemma 2.2, we have the following lemma:

**Lemma 4.4.** We have \( \Pr[W_0] - \Pr[W_1] \leq \sum_{i=1}^{S_1} \frac{3M}{2} \sqrt{(q_i + q_4 + 1) / |H|} \).

The proof is the same as that of [GHHM21, Thm.3], and we omit it.
Fig. 6. Games $G_3$–$G_7$ for sEUF-CMA security proof of $FS_{out}$. 
Lemma 4.5. There exists an adversary $A_{\text{brk}}$ against the $q_c$-HVZK property of $ID$ such that
\[ \Pr[W_5] - \Pr[W_6] \leq \text{Adv}_{\text{ID},A_{\text{brk}}}^{\text{brk}}(1^*) . \]

The running time of $A_{\text{brk}}$ is approximately that of $A$.

The proof is the same as that of \cite[Thm.3]{GHMM21}, and we omit it.

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

Lemma 4.6. There exists an adversary $A_\text{aux}$ against $aux$’s collision-resistance property of with respect to message on index $\lambda$ such that
\[ \Pr[W_3] - \Pr[W_4] \leq \text{Adv}_{\text{ID},A_\text{aux}}^{\text{aux}}(1^*) . \]

The running time of $A_\text{aux}$ is approximately that of $A$.

Proof. The difference between $G_3$ and $G_6$ occurs if the adversary submits a valid pair of a message and a signature $(\mu', (a'_1, \ldots, a'_n, a'_n))$ such that $(\mu', (a'_1, \ldots, a'_n, a'_n)) \notin \ mour$, the pair passes the collision checks by $\text{CollCheck}$, and $\text{AuxCheck}^{\mu'} = \text{true}$. The last condition $\text{AuxCheck}^{\mu'} = \text{true}$ implies that we have two messages $\mu \neq \mu'$ such that $a_{ux} = a_{ux}(\mu, \nu, \mu, \mu')$ is equivalent to $a_{ux} = a_{ux}(\lambda, \nu, \mu')$. This breaks the collision resistance property of $aux$ with respect to the message on index $\lambda$, and we can easily construct a reduction.

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

Lemma 4.7. There exists an adversary $A_m$ against the non-divergency of $ID$ such that
\[ \Pr[W_4] - \Pr[W_5] \leq \text{Adv}_{\text{ID},A_m}^{\text{non-divergency}}(1^*) . \]

The running time of $A_m$ is approximately that of $A$.

Proof. The difference between $G_2$ and $G_6$ happens if the adversary submits valid pair of message and signature $(\mu', (a'_1, \ldots, a'_n, a'_n))$ such that $(\mu', (a'_1, \ldots, a'_n, a'_n)) \notin \ mour$, the pair passes the collision checks by $\text{CollCheck}$ and $\text{AuxCheck}$, and $\text{ForkCheck}^{(\mu', a'_1, \ldots, a'_n, a'_n)} = \text{true}$. If the last check by $\text{ForkCheck}$ is true, then we have two valid transcripts $(\mu', a'_1, c'_1, \ldots, a'_n, c'_n) \ldots (\mu', a'_1, c'_1, \ldots, a'_n, c'_n)$ generated by the adversary and
\[ (\mu, a'_1, c'_1, \ldots, a'_n, c'_n, a'_{n+1}) \ldots (\mu, a'_1, c'_1, \ldots, a'_n, c'_n, a'_{n+1}) \]

generated by the signing oracle, where $k \in [2, n+1]$ satisfying $a'_1 = a'_1$; or,
\[ (\mu, a'_1, c'_1, \ldots, a'_n, c'_n, a'_{n+1}) \ldots (\mu, a'_1, c'_1, \ldots, a'_n, c'_n, a'_{n+1}) \]

generated by the signing oracle, where $k \in [2, n]$ satisfying $c'_1 = c'_1$ for all $l \in [k, n]$. Notice that, on the former condition, if $k \leq n$, then the collision checks force $c'_1 = c'_1$ for all $l \in [k, n]$. Hence, the former condition covers the conditions (a) and (b) of $\text{BranchCheck}$ in Definition 3.3. The latter condition is equivalent to the condition (c) of it. Therefore, the two valid transcripts apparently violate $q_c$-non-divergency of $ID$, and we can easily construct a reduction.

Lemma 4.8. There exists an adversary $A_{\text{msg}}$ against the EUF-NMA security of $DS$ such that
\[ \Pr[W_3] \leq \text{Adv}_{\text{ID},A_{\text{msg}}}^{\text{EUF-NMA}}(1^*) . \]

The running time of $A_{\text{msg}}$ is approximately that of $A$.

Proof. We show that to win in $G_1$, the adversary should submit a valid pair of message and signature $(\mu', (a'_1, \ldots, a'_n, a'_n)) \notin \ mour$ such that the values on $(\text{aux}_h, h_{\mu}^*, a'_{\mu})$ for $H$ and $h_{\mu}^*$ for $\mu$ are not reprogrammed by $\text{Sign}$. If not, there exists at least one index $i \in [n]$ such that $H$ is reprogrammed on input $(\text{aux}_h, h_{\mu}^*, a'_{\mu})$ or $\mu$ is reprogrammed on input $h_{\mu}^*$. Let $\ell \in [n]$ be the minimum of the indices of the reprogrammed points.

If $H$ is reprogrammed on input $(\text{aux}_h, h_{\mu}^*, a'_{\mu})$, then the simulator generates a transcript $(a_1, c_1, a_2, c_2, \ldots, a_n, c_n)$ in the computation of $\text{Sign}(\mu)$ for some $\mu$ satisfying $(\text{aux}_h, h_{\mu}^*, a'_{\mu})$ and $h_{\mu}^*$. Due to the collision check, $h_{\mu}^* - h_{\mu}^* \implies (\text{aux}_h, h_{\mu}^*, a'_{\mu}, a_{\mu}) = (\text{aux}_h, h_{\mu}^*, a'_{\mu}, a_{\mu})$, and so on. Thus, we have
\[ (\text{aux}_h, a_1, h_{\mu}^*, c_1) = (\text{aux}_h, a_1, h_{\mu}^*, c_1) \]

for $j = 1, \ldots, \ell$.

which implies that $H$ is reprogrammed for the indices $1, \ldots, \ell$. Since the index $\ell$ is the minimum of the indices of the reprogrammed points, Equation 1 implies that $\ell = 1$. We also note that, since $\text{ForkCheck}^{(\mu', a'_1, \ldots, a'_n, a'_n)}$ returns false, $a_1 = a'_1$ implies that $(a_{\mu+1} = a'_{\mu+1})$. Since we have $a_i = a'_i$ for all $i \in [n+1]$, $\mu$ must be distinct from $\mu^*$ due to the check in L.5 of $G_7$.

We then consider two subcases of $\lambda$:

8 and more by the second condition of $\text{ForkCheck}$. 

15
Fig. 7. Scheme $FS_h$ [ID, $H, y] = (Gen, Sign, Vrfy)$, where $ID = (Gen, P, V), H : \{0,1\}^* \rightarrow H$ is modeled as the random oracle, and $y_i : H \rightarrow C$ for $i \in [n]$ is also modeled as the random oracle. For ease of notation, we let $aux_i = aux(i, sk, \mu)$.

- Suppose that $\lambda = 1$. Since $\mu \neq \mu'$, we have $aux_i = aux'_i$ due to $AuxCheck$. However, this contradicts $aux_i = aux'_i$ from Equation 1 with $\ell = 1$.
- Next, suppose that $1 < \lambda \leq n$. Then, we have $aux_i = aux'_i$ due to $AuxCheck$ and $aux_0 = aux'_0$ for $l = 1, \ldots, \lambda - 1$. Since $(aux_i, h_{i-l}, a_i) = (aux'_i, h'_{i-l}, a'_i)$ implies $(h_i, c_i) = (h'_i, c'_i)$ for $j = 1, \ldots, \lambda - 1$ by induction, we have $(h_{i-1}, c_{i-1}) = (h'_{i-1}, c'_{i-1})$.
- Now, in the computation of $h_i$ and $h'_i$, $aux_i \neq aux'_i$ while $h_{i-l} = h'_{i-l}$ and $a_i = a'_i$. Due to the collision check, we have $h_{i-l} = h'_i$ and $c_i \neq c'_i$, which yields $c_i \neq c'_i$ for all $l > \lambda$. Thus, we have two different transcripts $(a_i, c_i, \ldots, a_n, c_n)$ and $(a'_i, c'_i, \ldots, a'_n, c'_n)$ satisfying $a_i = a'_i$ for all $i \in [n+1]$, $c_i = c'_i$ for all $i \in [\lambda, n]$. But, this case is already eliminated by L.5 of $ForkCheck$, and this leads to the contradiction.
- If $y_i$ is reprogrammed on input $h_i$, then the simulator generates a transcript $(a_i, c_i, \ldots, a_n, c_n)$ such that $h_i = h_i$ and $c_i = c'_i$. Due to the collision check, $h_i = h_i$ implies $(aux_i, h_{i-1}, a_i) = (aux'_i, h'_{i-1}, a'_i)$, 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 $(\mu, (a'_1, \ldots, a'_n)) \not\in Q$ that causes no reprogramming, we can easily construct $B$ against the EUF-NMA security of $DS$. $\Box$

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

5 $FS_h$ for Multi-Pass ID

If one can reproduce $a_1, \ldots, a_n$ from the challenges $c_1, \ldots, c_n$ and last message $a_{n+1}$, then we have a chance to replace $a_1, \ldots, a_n$ in the signature with the hash values $h_1, \ldots, h_n$. This replacement drastically shortens a signature because the prover’s messages $a_1, \ldots, a_n$ are much longer than the hash values $h_1, \ldots, h_n$. We call this variant of the FS transform as $FS_h$. In this section, we adopt the notion 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 algorithm $Rep$ is a DPT algorithm that takes $(ok, c_1, \ldots, c_n, a_{n+1})$ and messages $(a_1, \ldots, a_n)$, which might be $\perp$. We require completeness defined as follows: for honestly generated keys $(ok, sk)$ by $Gen$ and transcript $(a_1, c_1, \ldots, a_n, c_n, a_{n+1})$, if the transcript is valid, then $(a_1, \ldots, a_n) = Rep(ok, c_1, \ldots, c_n, a_{n+1})$.

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 $(c_1, c_2, \ldots, c_n, a_{n+1})$ such that if we reproduce non-$\perp$ messages $(a_1, \ldots, a_n)$ by $Rep$, then the transcript $(a_1, c_1, \ldots, c_n, a_{n+1})$ is valid. Since we want to consider $FS_h$, we replace $c_1, \ldots, c_n$ with $h_1, \ldots, h_n$ 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 $A$, its advantage is negligible in the security parameter, where the advantage is defined as

$$Adv^{\text{sound}}_{ID,A}(1^\lambda) := \Pr[\text{Exp}_{ID,A}^{\text{sound}}(1^\lambda) = \text{true}],$$

and $\text{Exp}_{ID,A}^{\text{sound}}(1^\lambda)$ is defined in Figure 8. If the advantage is 0 for any unbounded adversary, we say that the scheme is perfectly sound.
Assume that $H$ is resistant with respect to $\lambda$ as $H$ is also collision-resistant with respect to the verification key on index $\lambda$. Furthermore, we can show $wNR$ key on index $\lambda$.

**Lemma 5.1** (Special verifier means perfect soundness, extended for $FS_h$). Suppose that, on input $(\sigma, a_i, c_i, \ldots, c_n, a_{n+1})$, the verification algorithm $V$ outputs $\text{bool}(\text{Rep}(\sigma, c_i, \ldots, c_n, a_{n+1}))$. Then, the identification scheme $ID$ is perfectly sound.

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

**Theorem 5.1** ($FS_{inl} \Rightarrow ID, H, y$) is $\text{EUF-CMA/sEUF-CMA}$-secure, then $FS_{inl}[ID, H, y]$ 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 $h_i$ as $H(0x01, \text{salt}, \mu, \sigma, a_i)$, which means that $aux(1, \sigma, \mu) = (0x01, \text{salt}, \mu, \sigma)$ and $aux$ is perfectly collision-resistant with respect to $\lambda = 1$. PERK’s signature is of the form $(h_1, h_2, a_i)$, where $a_i$ includes salt.

If the adversary breaks the MBS security by outputting $\sigma, a_i$ as $\sigma, a_i$, then we have a collision of $aux$, for some $i$ or a collision of $H$. In addition, if the adversary breaks the M-S-UEO security by outputting $\sigma, a_i$ instead of $\sigma, a_i, \mu$, $\sigma$, and $\sigma = (h_1, \ldots, h_n, a_{n+1})$, then such values yield a collision with respect to the verification key of $aux$, 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 $H : \{0, 1\}^* \rightarrow H$ be a hash function. Let $DS := FS_h[ID, H, y]$. Assume that $H$ is collision-resistant.

- If $aux$ is collision-resistant with respect to message on index $\lambda$, then $DS$ satisfies S-DEO and MBS.

- If $aux$ is also collision-resistant with respect to the verification key on index $\lambda'$, 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 $\lambda$ and there exists index $\xi \in [\lambda, n]$ such that $aux_\xi$ can be written as $(\mu, \eta_\xi)$ for some $\eta_\xi$. Then $DS = FS_h[ID, H, y]$ satisfies wNR in the (Q)ROM.

The proof is in subsection A.3.

**Remark 5.1.** See Table 2. AllMer, MQOM, and PERK satisfy the condition of Lemma 5.3. The hash values in MIRA and RYDE involve $H(\mu)$ instead of $\mu$. The proof is obtained similarly by inserting one game. We will require an additional argument to show wNR security of FAEST and SDIH because their signature consists of $h_x$ 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, 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 $s \leftarrow F_q^*$. The verification key consists of $\text{seedF} \in \{0, 1\}^\gamma$ and $t \in F_q^n$: $\text{seedF}$ produces a sequence of random elements in $F_q$ and generates $f = (f_0, \ldots, f_n) \in F_q[x_1, \ldots, x_n]^n$ with $f_i = A_x + A_{x_i} - A_{x_i}$ for $i \in [n]$, where $A_x = x_1, \ldots, x_n = a_i^{(k,i)} + \sum_{l \neq i} a_l^{(k,l)} x_l \in F_q[x_1, \ldots, x_n]$ is a random Affine form; and $t$ is $f(s)$. For two vectors $a, b \in F_q^n$, $a \circ b$ is defined as component-wise multiplication. For a vector $a \in F_q^n$, we denote shares of $a$ via an $(N, N)$-additive secret share as $[a] = ([a]_1, \ldots, [a]_N) \in (F_q^n)^N$.

In nutshell, the signer will show the relation that $z = t - A_x(s) = x \circ y$, where $x = A_x(s)$ and $y = A_x(s)$ via an MPCItH protocol.

We modify the underlying MPCItH protocol $\text{ID}_{\text{Biscuit}} = (P_1, P_2, P_3)$ and $V$ with $\text{Repr}$ as depicted in Figure 9 to fit their scheme in our framework. The algorithms in the protocol are summarized as follows:

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

Notice that $\text{Repr}$ computes $[u] = -\sum_{i \in [N]} [u_i]$. Thus, the verifier $V$ checks if $\sum_{i} [u_i] = 0$ as the MPC’s result.

For the details, see the original specification [BKPV23].

The signature scheme Biscuit = $\text{FS}_{\text{h}}[\text{ID}_{\text{Biscuit}}, H, y]$ is defined by $\text{aux}_1 = (0x01, \text{salt}, \mu)$ and $\text{aux}_2 = (0x02, \text{salt})$.

### 6.1 Security

**sEUF-CMA security**: To show the sEUF-CMA security, we discuss the protocol’s HVZK property and non-divergence. For the definitions of primitives, see Section A. The HVZK property of $\text{ID}_{\text{Biscuit}}$ 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 (q₆-HVZK).** Suppose that PRF is secure, TreePRG and MakeShares are pseudorandom, and Com is hiding. Let $q_6$ be a polynomial of $1^\gamma$. Then, $\text{ID}_{\text{Biscuit}}$ with simulator $\text{Sim}_{\text{Biscuit}}$ in Figure 10 is $q_6$-HVZK.

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

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

**Proof.** For simplicity, we ignore parallelism $r$. Suppose that the adversary declines a valid transcript $\text{trans}_i = (a_1, c_1, a_2, c_2, a_3)$ generated by the simulator and outputs a valid transcript $\text{trans}' = (a_1, c_1', a_2', c_2')$. We parse them as $a_i = (\text{com}_i, \ldots, \text{com}_N, A_x, A_c)$ and $c_i = \epsilon$.

If condition (a) of BranchCheck’ in Definition 3.6 is met, then we have $c_2 = b' = c_2'$. We parse $c_2 = c_1' = c_3'$ and $a_2' = (\text{salt}', \text{path}', A_x', A_c', \text{com}', [\text{ar}]', \ldots)$. In this case, the adversary opens $\text{com}_i$ in $a_i$ as $(\text{salt}', \text{path}', \text{seed}', \text{rho}')$ computed from path’ and $i'$ since $i' = i'$ and the transcript $(a_1, a_2', c_2', a_3')$ is valid. Thus, we have $\text{com}_i = \text{Com}_{\text{salt}', \text{path}', \text{seed}', \text{rho}'}$. Since $\text{com}_i$ is chosen uniformly at random in L.11 of the simulator $\text{Sim}_{\text{Biscuit}}$ in Figure 10, this violates the non-invertibility of Com.

If condition (b) of BranchCheck’ is met, then we have $(a_2, c_2) = (a_3, c_3)$ and $a_3 \neq a_1$. Then we parse $a_2 = ((\text{ar}), [\text{ar}]), [\text{ar}])$, $c_2 = c_1' = c_3'$, and $a_3' = (\text{salt}', \text{path}', A_x', A_c', \text{com}', [\text{ar}])$. We have the following cases:

- If $\text{salt} = \text{salt}'$, then we have a collision for Com.
- If $\text{path} = \text{path}'$:
  - If $(\text{state}, \text{rho})_{\text{res}} = (\text{state}', \text{rho}')_{\text{res}}$, then it implies the collision for Reconst.
  - If $(\text{state}, \text{rho})_{\text{res}} \neq (\text{state}', \text{rho}')_{\text{res}}$, then we have at least one index $i$ satisfying $(\text{state}_i, \text{rho}_i) \neq (\text{state}', \text{rho}')$. Since the two transcripts are valid, we have a collision as $\text{com} = \text{Com}_{\text{salt, state, \rho}} = \text{Com}_{\text{salt, state, \rho}}$. If $(\text{Ar}, A_c, \text{com}_i, [\text{ar}]) = (\text{Ar}', A_c', \text{com}_i', [\text{ar}])$, then at least one of two transcripts is invalid because of inconsistency with $a_1$ and $a_2$, and this never happens.

Using those observations, we can construct reductions easily. \(\square\)

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

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

**S-DEO and MBS security**: Biscuit employs $\text{FS}_{\text{h}}$ with $\text{aux}_1 = (0x01, \text{salt}, \mu)$ and $\text{aux}_2 = (0x02, \text{salt})$. Therefore, $h_i$ in the signature includes the information of $\mu$. Since $\text{aux}$ is perfectly collision-resistant with respect to message on index 1, according to Lemma 5.2, Biscuit satisfies S-DEO and MBS if $H$ is collision-resistant.
Fig. 9. Prover, reconstruction, and verification algorithms of \texttt{ID}_{\text{Biscuit}}.
6.2 S-CEO and wNR Insecurity

Since \( \text{aux}_p \) and \( \text{aux}_q \) have no information on \( ok \), 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 \( ok' \) on which a message \( \mu \) and a signature \( \sigma \) is valid, while the adversary obtains \( \langle \mu, \sigma \rangle \) from the signing oracle \( \text{Sign} (ok, \cdot) \) many times as in the CMA setting.

We notice that, in the verification procedure, \( t \) appears only \( [z_j] := t - (A_{ij}([s])) \) for \( i = 1 \) of LinearCircuit.\(^{10}\) In addition, the direct computation involving \( [z_j] \) is \( \{[x], \otimes [a] \} \) for any \( a \in [N] \). Thus, the verification is on \( \mu, \sigma \), and the shifted verification key \( \langle \text{Sign}(ok, \cdot), t + e_j \rangle \).

This attack is justified as follows: When we consider the computation of \( [z'_j] \) on LinearCircuit on this modified \( t' \), we have \( [z'_j] = [z_j] + e_j \) for \( i = 1 \) and \( [z'_j] = [z_j] + e_j \) for \( i = 2, \ldots, N \). Thus, the verification is on \( \mu, \sigma \), and the shifted verification key \( \langle \text{Sign}(ok, \cdot), t + e_j \rangle \).

This attack succeeds if we have a signature and an index \( j \in [m] \) such that \( e_j \) is the random oracle, each signature satisfies this condition with probability \( p_j \) defined as \( p_j := 1 - (1 - q^{-m})^m \). After \( Q \) signing queries, there is at least one signature satisfying the condition with probability \( p_Q := 1 - (1 - p_j)^Q = 1 - (1 - q^{-m})^{mQ} \).

Table 3 summarizes the parameter sets of Biscuit and the success probability of the above S-CEO attack with \( Q = 2^{44} \) where, for small \( a \) and large \( b \), we use approximations \( 1 - (1 - a)^b \approx ab \) for \( ab \ll 1 \) and \( ab \lesssim 1 - \exp(-ab) \) otherwise. Since \( p_Q \) is larger that \( 2^{-8} \), 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 \( ok \) and \( \sigma \) on \( \mu \) and need to produce \( ok' \) and \( \sigma' \) such that \( \langle ok', \mu, \sigma' \rangle \) is valid while we cannot see \( \mu \). Notice that the above attack does not use the information of \( \mu \) 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 \( p_i \) in Table 3. Since \( p_i \) for parameter sets end with \( s \) is larger than \( 2^{-8} \), 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 \( t \) 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 \( 2^{44} \) chosen messages” [NIS22, 4.B.2].
Table 3. Parameter sets in Biscuit’s specification v1.1 and success probability $p_Q$ of S-CEO attack with $Q = 2^{64}$ signing queries.

<table>
<thead>
<tr>
<th>name</th>
<th>$q$</th>
<th>$n$</th>
<th>$m$</th>
<th>$\tau$</th>
<th>$N$</th>
<th>$p_1$</th>
<th>$p_Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>biscuit128f</td>
<td>16</td>
<td>64</td>
<td>67</td>
<td>16</td>
<td></td>
<td></td>
<td>$2^{-129.934}$ $≈ 2^{-65.934}$</td>
</tr>
<tr>
<td>biscuit128s</td>
<td>16</td>
<td>64</td>
<td>18</td>
<td>256</td>
<td></td>
<td></td>
<td>$2^{-65.934}$ $≈ 2^{-0.230}$</td>
</tr>
<tr>
<td>biscuit192f</td>
<td>16</td>
<td>87</td>
<td>55</td>
<td>16</td>
<td></td>
<td></td>
<td>$2^{-213.508}$ $≈ 2^{-149.508}$</td>
</tr>
<tr>
<td>biscuit192s</td>
<td>16</td>
<td>87</td>
<td>90</td>
<td>31</td>
<td>256</td>
<td></td>
<td>$2^{-33.508}$ $≈ 0.230$</td>
</tr>
<tr>
<td>biscuit256f</td>
<td>16</td>
<td>118</td>
<td>121</td>
<td>74</td>
<td></td>
<td></td>
<td>$2^{-289.081}$ $≈ 2^{-225.081}$</td>
</tr>
<tr>
<td>biscuit256s</td>
<td>16</td>
<td>118</td>
<td>121</td>
<td>42</td>
<td>256</td>
<td></td>
<td>$2^{-161.081}$ $≈ 2^{-97.081}$</td>
</tr>
</tbody>
</table>

References


BBd’23b. Carsten Baum, Lennart Braun, Cyprien Delpêch de Saint Guilhem, Michael Kloß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum


A Missing Definitions and Proofs

A.1 Missing Definitions

Definition A.1 (Pseudorandom Functions (PRF)). We say that \( \text{PRF} : \{0,1\}^s \times \{0,1\}^* \rightarrow \{0,1\}^{\kappa(s)} \) is secure if for any QPT adversary \( A \), its advantage

\[
\Pr\left[ A(\text{RF}(s)) = 1 \right] - \Pr_{\text{seed} \leftarrow \{0,1\}^s}\left[ A(\text{PRF}($seed$)) = 1 \right]
\]

is negligible in the security parameter, where \( \text{RF} : \{0,1\}^s \rightarrow \{0,1\}^{\kappa(s)} \) is a random function.

Definition A.2 (Pseudorandom Generator (PRG)). We say that pseudo-random generator \( \text{PRG} : \{0,1\}^s \rightarrow \{0,1\}^{\kappa(s)} \) is secure if for any QPT adversary \( A \), its advantage

\[
\Pr_{\text{seed} \leftarrow \{0,1\}^s}\left[ A(s) = 1 \right] - \Pr_{\text{seed} \leftarrow \{0,1\}^s}\left[ A(\text{PRG}($seed$)) = 1 \right]
\]

is negligible in the security parameter.

Definition A.3 (Tree PRG). A tree PRG scheme consists of the following three DPT algorithms, where an auxiliary information \( \text{aux} \) can be placed:

- \( \text{TreePRG}($seed$, aux) \rightarrow (r_1, \ldots, r_N) \): the tree-PRG algorithm takes \( \text{seed} \in \{0,1\}^s \) as input and outputs \( (r_1, \ldots, r_N) \in \{0,1\}^{\kappa(s)} \).
- \( \text{GetPath}(i^*, \text{seed}, \text{aux}) \rightarrow \text{path} \): the path finding algorithm takes index \( i^* \in [N] \) and seed \( \in \{0,1\}^s \) as input and outputs path information \( \text{path} \).
- \( \text{Reconstruct}(i^*, \text{path}, \text{aux}) \rightarrow (r_{i^*})_{i^*} \): the reconstruction algorithm takes path and index \( i^* \) as input and outputs \( (r_{i^*})_{i^*} \in \{0,1\}^{\kappa(s)} \).

For correctness, we require that for any seed \( \in \{0,1\}^s \), \( i^* \in [N] \), and aux \( \in \{0,1\}^s \), we have \( r_i = r_{i'} \) for all \( i \neq i' \), where \( r_{i^*} := \text{TreePRG}($seed$, aux) \), \( \text{path} := \text{GetPath}(i^*, \text{seed}, \text{aux}) \), and \( (r_{i^*})_{i^*} := \text{Reconstruct}(i^*, \text{path}, \text{aux}) \).

We say that a tree PRG scheme is secure if for any QPT adversary \( A \), for any \( i^* \in [N] \), (and for any aux \( \in \{0,1\}^s \)) its advantage

\[
\Pr_{\text{seed} \leftarrow \{0,1\}^s, (r_{i^*})_{i^*} := \text{TreePRG}($seed$, aux)}\left[ A((r_{i^*})_{i^*}, r_i, \text{GetPath}(i, \text{seed}, \text{aux})) = 1 \right] - \Pr_{\text{seed} \leftarrow \{0,1\}^s, (r_{i^*})_{i^*} := \text{TreePRG}($seed$, aux), s \leftarrow \{0,1\}^{\kappa(s)}}\left[ A((r_{i^*})_{i^*}, s, \text{GetPath}(i, \text{seed}, \text{aux})) = 1 \right]
\]

is negligible in the security parameter.

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

\[
\Pr\left[ (\text{path}, \text{path}') \leftarrow A(1^s) : \text{path} \neq \text{path}' \land \text{Reconstruct}(i^*, \text{path}, \text{aux}) = \text{Reconstruct}(i^*, \text{path}', \text{aux}) \right]
\]

is negligible in \( \kappa \).
Definition A.5 (Commitment). We say that a commitment scheme \( \text{Com} : \{0,1\}^* \times \{0,1\}^s \rightarrow \{0,1\}^s \) is
- non-invertible if for any QPT adversary \( A \), its advantage
  \[
  \Pr[\text{com} \leftarrow \{0,1\}^s, (x, \rho) \leftarrow A(\text{com}) : \text{Com}(x; \rho) = \text{com}]
  \]
  is negligible in the security parameter;
- binding if for any QPT adversary \( A \), its advantage
  \[
  \Pr[(x, \rho, x', \rho') \leftarrow A(1^s) : x \neq x' \land \text{Com}(x; \rho) = \text{Com}(x'; \rho')]
  \]
  is negligible in the security parameter;
- collision-resistant if for any QPT adversary \( A \), its advantage
  \[
  \Pr[(x, \rho, x', \rho') \leftarrow A(1^s) : (x, \rho) \neq (x', \rho') \land \text{Com}(x; \rho) = \text{Com}(x'; \rho')]
  \]
  is negligible in the security parameter;
- hiding if for any QPT adversary \( A \) and for any \( x \in \{0,1\}^* \), its advantage
  \[
  \left| \Pr_{\text{com} \leftarrow \{0,1\}^s}[A(\text{com}) = 1] - \Pr_{\rho \leftarrow \{0,1\}^s}[A(\text{Com}(x; \rho)) = 1] \right|
  \]
  is negligible in the security parameter.

A.2 (Strong) Existential Unforgeability of \( \text{FS}_h \)

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).
- \( G_2 \): In this game, the challenger checks if \( h_i' = \bar{h}_i \) for all \( i \in [n] \) and \( V(\text{sk}, h, c'_1, ..., c'_{s''}) = \text{true} \) (See L.13).
- \( G_7 \): In this game, the challenger checks if \( h_i' = \bar{h}_i \) for all \( i \in [n] \) and \( V(\text{sk}, h, c'_1, ..., c'_{s''}) = \text{false} \) (See L.14).

![Figure 11](image-url)

**Fig. 11.** Games \( G_0, G_7 \), and \( G_7 \) and an adversary \( A_{\text{FS}_h} \) for sEUF-CMA security proof of \( \text{FS}_h \).
Apparently, we have \(\text{Adv}_{\text{fs-h}}^{\text{conf-cma}}(1^*) = \Pr[W_0] \leq \Pr[W_1] + \Pr[W_2].\)

On \(G_1\), we can construct an adversary \(A_{\text{fs-h}}\) against \(\text{FS}_{\text{h}}[\text{ID}, H, y]\) that simulates \(\text{Sign}\) as in Figure 11.

We argue that if \(A\)'s output \((\mu', (h_1', \ldots, h_{n'}', a_{n'+1}'))\) is fresh then \(A_{\text{fs-h}}'\)'s output \((\mu, (a_1', \ldots, a_{n'+1}'))\) is also fresh. Suppose that \(A_{\text{fs-h}}'\)'s output \((\mu', (a_1', \ldots, a_{n'+1}'))\), which is produced from \(A\)'s output \((\mu', (h_1', \ldots, h_{n'}', a_{n'+1}'))\), is in the list. This means that \(\mu'\) is queried by \(A\). \(A_{\text{fs-h}}\) receives \((a_1', \ldots, a_{n'+1}')\) from its signing oracle \(\text{Sign}'\), \(A_{\text{fs-h}}\) computes \(h_i' := H(\text{aux}, h_{i'}, a_{i'})\) for \(i = 1, \ldots, n\), and returns \((h_1', \ldots, h_n', a_{n+1}')\) to \(A\). Thus, \(A\)'s output \((\mu', (h_1', \ldots, h_{n'}', a_{n'+1}'))\) also should be in the list.

Hence, if \(A\) wins, then \(A_{\text{fs-h}}\) also wins. We have
\[
\Pr[W_1] \leq \text{Adv}_{\text{fs-h}}^{\text{conf-cma}}[\text{ID}, y, A_{\text{fs-h}}](1^*).
\]

On \(G_2\), if non-\(\bot (a_1', \ldots, a_{n'+1}')\) is produced by \(\text{Rep}\) in L7, then \(V(\text{aux}, a_1', \ldots, a_{n'+1}')\) should be true since \(\text{ID}\) is computationally sound. In other words, if it is violated, we can construct an adversary \(A_{\text{fs-h}}\) against \(\text{ID}\) by using \(\text{A}\) against \(\text{FS}_h\) such that
\[
\Pr[W_2] \leq \text{Adv}_{\text{ID, y, A}_{\text{fs-h}}}^{\text{conf-cma}}(1^*).
\]

This completes the proof. \(\square\)

### A.3 Weak Non-Resignability of \(\text{FS}_h\)

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 \(S \subseteq \mathcal{X}\) be random. Let \(G, H : \mathcal{X} \rightarrow \mathcal{Y}\) be random functions satisfying \(\forall x \notin S, G(x) = H(x)\). Let \(z\) be a random string. Note that \(S, G, H, z\) may have arbitrary joint distribution.

Let \(A\) be a \(q\)-query oracle algorithm. Let \(B^{(z)}\) be an algorithm that on input \(z\) chooses \(i \leftarrow [q]\), runs \(A^{(z)}(z)\) 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
\[
\begin{align*}
P_f := & \Pr[b \leftarrow A^{(z)}(z) : b = 1], \\
P_s := & \Pr[b \leftarrow B^{(z)}(z) : b = 1], \\
P_z := & \Pr[s \leftarrow B^{(z)}(z) : s \in S].
\end{align*}
\]

Then, we have
\[
|P_f - P_s| \leq 2q \sqrt{P_f} \quad \text{and} \quad \sqrt{P_f} - \sqrt{P_z} \leq 2q \sqrt{P_z}.
\]

If \(z\) and \(S\) are independent, the bound can be \(4q \cdot \max_{x \in S} \Pr[x \in S]\). But, in our context, \(z\) and \(S\) are correlated. Don, Fehr, Huang, and Struck [DFH23] showed that the BUFF conversion with \(\text{salt (S-BUFF)}\),\(^{12}\) satisfies their revised non-resignability in the (Q)ROM, where the adversary is given auxiliary information \(\text{AUX}(\mu, vk)\) 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 \(\text{FS}_h\) without \(\text{salt}\). Very recently, Don, Fehr, Huang, Liao, and Struck [DFH124] showed that the standard BUFF conversion is enough in the QROM for somewhat stronger non-resignability where the adversary can get \(\text{AUX}(\mu, vk)\) 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. \(A\) is given \(vk\) and \(\sigma\), which is produced on a message \(\mu \leftarrow \mathcal{M}\), and outputs \(vk' \neq vk\) and \(\sigma'\). If \(\text{Ver}_{vk'}(vk', \mu, \sigma') = \text{true}\), then the adversary wins.

- **G_1:** In this game, we introduce a collision-check procedure for \(\text{aux}\) as follows: Receiving \(vk' \neq vk\) and \(\sigma'\), the challenger computes \(\text{aux}' := \text{aux}(\lambda, vk', \mu)\). If \(\text{aux}_{\sigma} = \text{aux}'\), then the adversary loses. This modification is justified by the collision-resistance property of \(\text{aux}\) with respect to the verification key on index \(\lambda\).

- **G_2:** In this game, we introduce a collision-check procedure for \(H\) as follows: Receiving \(vk' \neq vk\) and \(\sigma'\), the challenger checks if \(H(\text{aux}_{\sigma}, h_{j-1}, a_j) = H(\text{aux}_{\sigma}, h_{j-1}', a_j')\) while \((\text{aux}_{\sigma}, h_{j-1}, a_j) \neq (\text{aux}_{\sigma}, h_{j-1}', a_j')\) for some \(j \in [1, n]\), where \( auxiliary values in the verification of \(vk', \mu, \sigma'\). 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 \(\sigma'\) such that \((\text{aux}_{\sigma}, h_{j-1}) \neq (\text{aux}_{\sigma}, h_{j-1}')\) for all \(j \in [1, n]\). Let \(\zeta\) be a minimum index in \([1, n]\) such that \(\text{aux}_{\zeta} = (\mu, \eta_{\zeta})\). Now, \(H\) should be asked at least one point \((\mu, \eta_{\zeta}, h_{j-1}, a_j')\) to compute \(h_{j-1}'\) in the verification of \((vk', \mu, \sigma')\), while this point is not asked in the signing/verification of \((vk, \mu, \sigma)\).

\(^{12}\) The signer first chooses salt \(\text{salt}\), computes \(y = F(vk, \mu, \text{salt})\), and generates a signature \(\sigma \leftarrow \text{Sign}(vk, \mu)\), and outputs \((\sigma, y, \text{salt})\), where \(F\) is the random oracle.
1: \( G_0 \) and \( G_2 \)
2: \((vk, sk) \leftarrow \text{Gen}(1^s)\)
3: \( \mu \leftarrow MS \)
4: \( \sigma \leftarrow \text{Sign}^{h_y}_{H}(sk, \mu) \)
5: \((\sigma', vk') \leftarrow A^{\text{proj}}(vk, \sigma)\)
6: if \( sk = vk' \) then return false
7: return \( \text{Vrfy}^{h_y}_{H}(vk', \sigma') \)

1: \( G_1 \)
2: \((vk, sk) \leftarrow \text{Gen}(1^s)\)
3: \( \mu \leftarrow MS \)
4: \( \sigma \leftarrow \text{Sign}^{h_y}_{H}(sk, \mu) \)
5: \((\sigma', vk') \leftarrow A^{\text{proj}}(vk, \sigma)\)
6: if \( sk = vk' \) then return false
7: parse \( \sigma' = (h_1', ..., h_n', a_{n+1}') \)
8: for \( i \in [n] \) do \( c_i' := y(h_i') \)
9: \((a_1', ..., a_n') := \text{Rep}(sk, c_1', ..., c_n', a_{n+1}')\)
10: if \( (a_1', ..., a_n') = \perp \) then return false
11: if \( \text{aux}_i = \text{aux}_i' \) then return false //\( G_1^- \)
12: \( h_i' := \perp \)
13: for \( i \in [n] \) do \( h_i := H(\text{aux}_i, h_{i-1}', a_i') \)
14: for \( i \in [1, n] \) do //\( G_2^- \)
15: if \( (\text{aux}_i, h_{i-1}', a_i') \neq (\text{aux}_i, h_{i-1}', a_i') \) and \( h_i = \tilde{h}_i \) then return false //\( G_2^- \)
16: return \text{bool}(\forall i \in [n] : h_i' = \tilde{h}_i) \)

Fig. 12. Games \( G_0, G_1, \) and \( G_2 \) for the wNR security proof of \( FS_h \).
Fig. 13. Games $G_3$ and $G_4$ for the wNR security proof of $FS_n$. 

\begin{align*}
G_3 & : H \leftarrow \text{Func}([0,1]^*, \mathcal{H}) \\
G & : = H \\
(s, k) & \leftarrow \text{Gen}(1^*) \\
\mu & \leftarrow \mathcal{MS} \\
\sigma & \leftarrow \text{Sign}_{H}(sk, \mu) \\
S & := \{ (aux_i, h_i, a_i) : aux_i \text{ contains } \mu \} \\
\text{for} (aux_i, h_i, a_i) \in S \text{ do} \\
\hat{h}_i & \leftarrow H \\
G & := G((aux_i, h_i, a_i) \mapsto \hat{h}_i) \\
(\sigma', s_k) & \leftarrow A^G(\sigma') \\
\text{if } \sigma = \sigma' \text{ then return false} \\
\text{parse } \sigma' = (h_{i}', h_{i}', a_{i}') \\
\text{for } i \in [n] : c_i := \gamma(h_i') \\
(a_i', \ldots, a_n') & := \text{Rep}(ok, c_1', \ldots, c_n', a_{i-1}') \\
\text{if } (a_{i}', \ldots, a_n') = \bot \text{ then return false} \\
\text{if aux}_i = aux_i' \text{ then return false} \\
h_i' & := \varnothing \\
\text{for } i \in [n] : \hat{h}_i := H(\text{aux}_i', h_{i}', a_i') \\
\text{for } i \in [\lambda, n] \text{ do} \\
\text{if } (aux_i, h_{i-1}, a_i) \neq (aux_i', h_{i-1}', a_i') \text{ and } h_i = \hat{h}_i \text{ then return false} \\
\text{return bool}(\forall i \in [n] : h_i' = \hat{h}_i) \\
\end{align*}

\begin{align*}
G_4 & : H \leftarrow \text{Func}([0,1]^*, \mathcal{H}) \\
G & : = H \\
(s, k) & \leftarrow \text{Gen}(1^*) \\
\mu & \leftarrow \mathcal{MS} \\
\sigma & \leftarrow \text{Sign}_{H}(sk, \mu) \\
S & := \{ (\mu, \cdot) \} \\
\text{for } (\mu, x) \in S \text{ do} \\
G & := G((\mu, x) \mapsto \bot) \\
(\sigma', s_k) & \leftarrow A^G(\sigma') \\
\text{if } \sigma = \sigma' \text{ then return false} \\
\text{parse } \sigma' = (h_{i}', h_{i}', a_{i}') \\
\text{for } i \in [n] : c_i := \gamma(h_i') \\
(a_i', \ldots, a_n') & := \text{Rep}(ok, c_1', \ldots, c_n', a_{i-1}') \\
\text{if } (a_{i}', \ldots, a_n') = \bot \text{ then return false} \\
\text{if aux}_i = aux_i' \text{ then return false} \\
h_i' & := \varnothing \\
\text{for } i \in [n] : \hat{h}_i := H(\text{aux}_i', h_{i}', a_i') \\
\text{for } i \in [\lambda, n] \text{ do} \\
\text{if } (aux_i, h_{i-1}, a_i) \neq (aux_i', h_{i-1}', a_i') \text{ and } h_i = \hat{h}_i \text{ then return false} \\
\text{return bool}(\forall i \in [n] : h_i' = \hat{h}_i) \\
\end{align*}
- $G_1$: In this game, after obtaining $\sigma = (h_1, \ldots, h_n, a_{n+1})$, we reprogram the points related to $\mu$ 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 $2q\sqrt{\Pr[G_{3,1} \Rightarrow 1]}$, where $G_{3,1}$ is defined in Figure 13.

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

- $G_2$: Next, the challenger gives a filtered random oracle $H'$, which returns $\bot$ if the input is $(\mu, \cdot)$ to the adversary. Notice that in this game, the adversary has no information of the hash value $h_{\mu, \eta_i, R_{i-1}, c_i}$, while it outputs $h'_i$ in the signature. Therefore, the winning probability in this game is at most $1/|H|$.

The difference between the two games $G_2$ and $G_3$ is bounded by the O2H and we have $2q\sqrt{\Pr[G_{3,1} \Rightarrow 1]}$, where game $G_{3,1}$ is defined in Figure 13. Again, since $B$ is given no information of $G(\mu, \cdot)$ via $z = (\alpha, \sigma)$, the probability $\Pr[G_{3,1} \Rightarrow 1]$ is at most $1/|M|$.

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 $(s, (\text{seed}^o)_{e \in [r]})$ uniformly at random. This modification is justified by the security of PRF.
- $G_3$: Next, the prover chooses $\text{seed}^o_e$ and $\rho^o_e$ for $e \in [r]$ uniformly at random. This modification is justified by the security of TreePRG.
- $G_4$: Next, we make the prover choose $([s]_G^o, [a]_G^o, [c]_G^o)$ uniformly at random. This modification is justified by the security of MakeShares used for those shares.
- $G_5$: In this game, the protocol is modified to choose com$_{i,r}$ uniformly at random. This modification is justified by the hiding property of the commitment scheme Com.
- $G_6$: Next, we make the prover compute $[a]_G^o := \sum_{i \in C} [b]_G^i$. This modification is justified by the correctness of the MPCitH protocol.
- $G_7$: Finally, the prover chooses $\text{As}^o, \text{Ac}^o, \text{Ac}^o$, and $[\alpha]_G^o$ uniformly at random and now the modified protocol 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 $[s]_{i, r}, [c]_{i, r},$ and $[a]_{i, r}$ are hidden from the adversary, they mask the distribution of $\text{As}^o, \text{Ac}^o,$ and $[\alpha]_{i, r}$ 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 $FS_{\text{last}}$ defined later. If $h_1, \ldots, h_{n-1}$ 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 $P_1, \ldots, P_r$ since $h_1, \ldots, h_{n-1}$ are independent of a message and, receiving a message $\mu$ 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 $h_1, \ldots, h_{n-1}$, the commitment-reproducing algorithm $Rep$ should be able to reproduce $a_1, \ldots, a_n$ from the last challenge $c_n = y_\mu(h_n)$ 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 algorithm $Rep$ that takes $(\alpha, c_1, \ldots, c_n, a_{n+1})$ and outputs messages $(a_1, \ldots, a_n)$, which may be $\bot$. We
Fig. 14. Scheme $\text{FS}_{\text{shuf}}[\text{ID}, H, \gamma] = (\text{Gen, Sign}_{\text{shuf}}, \text{Vrfy}_{\text{shuf}})$, where $\text{ID} = (\text{Gen, P, V, H}) : [0, 1]^* \rightarrow H$ is modeled as the random oracle, and $\gamma : H \rightarrow C$, for $i \in [n]$ is also modeled as the random oracle. For ease of notation, we let $\text{aux} = \text{aux}(i, ok, \mu)$.

$$\begin{align*}
1: & \text{Sign}_{\text{shuf}}(sk, \mu) \\
2: & h_0 \leftarrow \emptyset; c_0 \leftarrow \emptyset; \text{state} \leftarrow \emptyset \\
3: & \text{for } i = 1, \ldots, n \text{ do} \\
4: & (a_i, \text{state}) \leftarrow \text{P}(sk, c_{i-1}, \text{state}) \\
5: & h_i \leftarrow \text{H}(\text{aux}, h_{i-1}, a_i) \\
6: & c_i \leftarrow y(h_i) \\
7: & a_{i+1} \leftarrow \text{P}(sk, c_i, \text{state}) \\
8: & \text{return } \sigma := (h_n, a_{n+1})
\end{align*}$$

Fig. 15. An adversary $\mathcal{A}_{\text{shuf}}$ for sEUF-CMA security proof of $\text{FS}_{\text{shuf}}$.

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

$$\begin{align*}
\text{Rep}(ok, c_1, c_2, \ldots, c_n, a_{n+1}) &\quad // \text{Ignore } c_1, \ldots, c_{n-1} \\
2: & h_0 \leftarrow \emptyset; \text{state} \leftarrow \emptyset \\
3: & \text{for } i = 1, \ldots, N \text{ do} \\
4: & (a_i, \text{state}) \leftarrow \text{Rep}(ok, (a_{i-1}, c_{i-1}), a_{i-1}, \text{state}) \\
5: & \text{if } a_i = \bot \text{ then return false} \\
6: & h_i \leftarrow \text{H}(\text{aux}, h_{i-1}, a_i) \\
7: & c_i \leftarrow y(h_i) \\
8: & \text{return } (\mu^*, (h_1, \ldots, h_{i-1}, h_i, a_{n+1}))
\end{align*}$$

If $\text{Rep}$ is decomposable, then we can consider the signature scheme $\text{FS}_{\text{shuf}}$ as the variant of $\text{FS}_h$, defined in Figure 14.

We have the following theorem:

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

Combined with **Theorem 5.1**, we obtain the following corollary.

**Corollary B.1** ($\text{FS}_{\text{cmt}} \Rightarrow \text{FS}_{\text{shuf}}$). Suppose that ID is computationally sound and $\text{Rep}$ is decomposable. If $\text{FS}_{\text{cmt}}[\text{ID}, H, \gamma]$ is EUF-CMA/sEUF-CMA-secure, then $\text{FS}_{\text{shuf}}[\text{ID}, H, \gamma]$ 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 $A_{FS_k}$ as in Figure 15. Apparently, the simulation of the signing oracle is perfect. We show that if $A$'s output is valid for $FS_{\text{short}}$, then the output of $A_{FS_k}$ is also valid for $FS_k$. Let $(\mu', (h_1, a_{n+1}))$ be $A$'s output and let $(\mu', (h_1, h_2, \ldots, h_{n+1}, a_{n+1}))$ be $A_{FS_k}$'s output. Since $A$'s output is valid, we have $h_1 = h_2^*$. We next check how to compute the hash values in the verification algorithm $Vrfy_k$ (see Figure 7). Let $h_1, \ldots, h_n$ be hash values computed in L.7 of the verification algorithm $Vrfy_k$ on input $ok, \mu'$, and $\sigma' = (h_1, \ldots, h_n, a_{n+1})$. To compute them by Rep, we first compute $c_0 = y(h_0)$; We then compute for $i = 1, \ldots, n$, $(\mu, \text{state}) = (\text{Rep}(ok, (a_i, c_i), t_{i-1}, c_{i-1}, a_{n+1}, \text{state}))$ (and reject if $a_i' = 1$), $h_i = H(\text{aux}_i, h_{i-1}, a_i')$, and $c_i = y(h_i)$; After the recomputation of $a_i', \ldots, a_n'$ by this procedure, we compute $h_i^* = H(\text{aux}_i, h_{i-1}, a_i')$. Thus, we have $h_i^* = h_i^*$ for all $i \in [n]$ and the pair $(\mu', (h_1, \ldots, h_{n+1}, a_{n+1}))$ is also valid for $FS_k$. Finally, if $(\mu', (h_1, a_{n+1}))$ is new, then the converted signature is also new. This completes the proof. \hfill $\square$

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

**Corollary B.2** ($FS_{\text{short}} \Rightarrow FS_{\text{short}}$). Suppose that $\text{Rep}$ is decomposable. If $FS_k[\text{ID}, H, \gamma]$ is wNR-secure, then $FS_{\text{short}}[\text{ID}, H, \gamma]$ is also.

## C MQDSS

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

1. Receive input $ok = (F, v)$, $c_1 = \alpha \in F_q$, and $c_2 = b \in \{0, 1\}$.
2. Compute messages as follows:
   - If $b = 0$, then pick $r_0, t_0 \leftarrow F_q^m$ and $e_0 \leftarrow F_q^m$, compute $com_0 := Com(r_0, t_0, e_0)$, pick a random $com_1 \leftarrow \{0, 1\}^s$, compute $a_0 := (t_0, e_0) = (ar_0 - t_0, aF(r_0) - e_0)$, and set $a_1 = r_0$.
   - If $b = 1$, then pick $r_1, t_1 \leftarrow F_q^m$ and $e_1 \leftarrow F_q^m$, compute $com_1 := Com(r_1, \alpha(v - F(r_1)) - G(t_1, r_1) - e_1)$, pick a random $com_0 \leftarrow \{0, 1\}^s$, set $a_1 := (t_0, e_0)$, and set $a_1 = r_1$.
3. Output $(a_1 = (com_0, com_1), a_2, a_3)$.

It is easy to show that SSH11 is $q$-HVZK, assuming $Com$ is hiding. It is also easy to show that 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 $\alpha \in F_q^k$ and $K \in F_q^{r \times (k-r)}$. The verification key consists of a seed seed$_{\text{sk}}$, which produces $M_1, \ldots, M_r \in F_q^{m \times m}$ via PRG, and a matrix $M_0 \in F_q^{m \times m}$ such that $M_0 \left[ \frac{k \times k} {k \times k} \right] = O$, where $M_n := M_0 + \sum a_i M_i$. The condition means that the rank of the matrix $M_n$ is at most $r$.

We modify the underlying MPCith protocol $\text{ID}_{\text{MPCith}}$, P = $(P_1, P_2, P_3)$ and $V$ with $\text{Rep}$, as depicted in Figure 16.

- The first challenge $R$ is chosen from $F_q^{m \times m}$, where $s < m$.
- For $M \in F_q^{m \times m}$, $M_0 \in F_q^{m \times m}$, and $M_i \in F_q^{m \times (k-r)}$ denotes the matrices consisting of the first $r$ columns of $M$ and the last $(n-r)$ columns of $M$, respectively.
- MakeShares generates pseudorandom shares from the seed and an auxiliary information (salt, $l$).
- The specification sheet just says that “The parties locally compute $[M_{\sigma l}]$ and $[M_{\sigma l}]''$ in $P_i$. In the reference implementation, $M_{\sigma l}$ and $M_{\sigma l}$ 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 GetPath.

For the details, see the original specification [AVR+23]. The signature scheme MiRitH = $FS_k[\text{ID}_{\text{MPCith}}, H, \gamma]$ is defined by aux$_1 = (\text{salt}, \mu)$ and aux$_2 = (\text{salt}, \rho)$. They used implicit domain separation of $H$ for $h_1$ and $h_2$ [AVR+23, Sec.6.5], because the lengths of $a_i$ and $(h_i, a_j)$ differ.

### D.1 Security

#### sEUF-CMA security

**Lemma D.1** ($q_N$-HVZK). Suppose that TreePRG and MakeShares are pseudorandom and Com is hiding. Then, $\text{ID}_{\text{MPCith}}$ is $q_N$-HVZK.

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

1. \[...\]
Fig. 16. Prover, reconstruction, verification, and simulation algorithms of ID_{NIRH}. We run the protocol in $r$-parallel way sharing salt.

1. $P_{\text{in}}(sk, R, state)$ for ID_{NIRH}
   // Simulate MPC
   // The offset follows the reference implementation
2. $M_{a,i}^L := M_{a,i} + \sum_{i \in [\ell]} [\alpha_j]_i M_{a,j}^L$
3.forall $i \in [2, N]$: $[M_{a,i}^R] := \sum_{i \in [\ell]} [\alpha_j]_i M_{a,j}^R$
4. forall $i \in [2, N]$: $[M_{a,i}] := M_{a,i}^L + \sum_{i \in [\ell]} [\alpha_j]_i M_{a,j}^R$
5. forall $i \in [N]$: $[S] := R \cdot [M_{a,i}^L + [A]$
6. forall $i \in [N]$: $[V] := S \cdot [K] - R \cdot [M_{a,i}^L - [C]$
7. $a_2 := ([S], [V])_{\text{in}}$
8. return $a_2$ and state

1. $P_{\text{out}}(sk, i', state)$ for ID_{NIRH}
2. Parse state $= \text{(salt, state}, \rho)_{\text{in}}$
3. $a_3 := \text{(salt, state, \rho)}_{\text{out}}, \text{com}_{\text{in}}, ([S])_c$
4. return $a_3$

1. $\text{Rep}(sk, c_1, c_2, a_i)$ for ID_{NIRH}
2. Parse $c_1 = R$ and $c_2 = i'$
3. Parse $a_i = \text{(salt, state, \rho)}_{\text{out}}, \text{com}_{\text{in}}, ([S])_c$
   // Simulate MPC
4. forall $i \in [N] \setminus \{i'\}$ do
5. if $i \neq N$ then
6. Parse state, $= \text{(seed, \rho)}_{\text{in}}$
7. Compute $[A], [\alpha], [C], [K]$ from salt and seed
8. else
9. Parse state_{N} = \text{(seed, \rho)}_{\text{in}}$
10. Compute $[A]_{\text{in}}$ from salt and seed_{N}
11. Compute com_{i} := Com((i, state), \rho)_i$
12. $a_1 := ([\text{com}_{i}]_{\text{in}}$
   // Run MPC except $i'$
13. forall $i \in [N] \setminus \{i'\}$: Compute $[M_{a,i}^L]$ and $[M_{a,i}^R]$ from $ok$ and $[\alpha]$
14. forall $i \in [N] \setminus \{i'\}$: $[S] := R \cdot [M_{a,i}^L] + [A]$
15. $S := \sum_{i \in [N]} [S]$
16. forall $i \in [N] \setminus \{i'\}$: $[V] := S \cdot [K] - R \cdot [M_{a,i}^L] - [C]$
17. $[V]_{\text{in}} := [\sum_{i \in [N]} [V]]_{\text{in}}$
18. $a_2 := ([S], [V])_{\text{in}}$
19. return $a_1$ and $a_2$

1. $V(\text{ok}, a_1, a_2, a_3)$ for ID_{NIRH}
2. Compute $(a_1, a_2) := \text{Rep}(sk, c_1, c_2, a_i)$
3. return bool$(\text{true, (a_1, a_2)})$

1. Sim_{MPC}(sk, c_1, c_2) for ID_{NIRH}
2. Choose salt at random
3. // Run the following procedure in parallel
4. Parse $c_1 = R$ and $c_2 = i'$
5. // Simulate MPC's setup
6. Choose seed at random
7. $(\text{seed, \rho})_{\text{in}} = \text{TreePRG(\text{seed, salt})}$
8. forall $i \in [N] \setminus \{i'\}$ do
9. if $i \neq N$ then
10. Compute $[A], [\alpha], [C], [K]$ from salt and seed
11. Compute $[V]_{\text{in}}$
12. Compute $[\text{com}_{\text{in}}]:=\text{Com}((i, \text{state}), \rho)_i$
13. else
14. Compute $[A]_{\text{in}}$ from salt and seed_{N}
15. Choose $\text{com}_{\text{in}}$ at random
16. forall $i \in [N] \setminus \{i'\}$: compute $[M_{a,i}^L]$ and $[M_{a,i}^R]$
17. forall $i \in [N] \setminus \{i'\}$: $[S] := R \cdot [M_{a,i}^L] + [A]$
18. Choose $[S]_{\text{in}}$ at random
19. $S := \sum_{i \in [N]} [S]$
20. forall $i \in [N] \setminus \{i'\}$: $[V] := S \cdot [K] - R \cdot [M_{a,i}^L] - [C]$
21. $[V]_{\text{in}} := [\sum_{i \in [N]} [V]]_{\text{in}}$
22. $a_2 := ([S], [V])_{\text{in}}$
   // Simulate response
23. $a_3 := ([\text{salt, state, \rho})_{\text{out}}, \text{com}_{\text{in}}, ([S])_c$
24. return $a_1, a_2, a_3$
Lemma D.2 (Strong non-divergency). Suppose that Com is non-invertible and collision-resistant. Then, ID₉ for MiRitH is strongly non-divergent with respect to Sim₉.

Proof. For simplicity, we ignore parallelism r. Suppose that the adversary declines a valid transcript trans = (a₁, c₁, a₂, c₂, a₃) generated by the simulator and outputs a valid transcript trans' = (a₁, c₁, a₂', c₂', a₃'). Note that they are valid and share a₁ and c₁. We parse them as a₁ = (com₁, ..., comₙ) and c₁ = R. If the condition (a) is met, then we have c₂ = c₂'. We parse a₂ = ([S], [V])₉∈[N], c₂ = i', c₂' = i', and a₃ = (salt', (state', ρ'), ω', com', [S']ᵢ'). Since the adversary opens com₉ as (salt', state', ρ') in the valid transcript (a₁, c₁, a₂', c₂', a₃'), this breaks the non-invertibility of Com.

If the condition (b) is met, then we have (a₂, c₂) = (a₂', c₂') and a₃ ≠ a₃'. We then parse a₂ = ([S], [V])₉∈[N], c₂ = i', a₃ = (salt, (state, ρ), ω, com, [S]ᵢ), and a₃' = (salt', (state', ρ'), ω', com', [S']ᵢ'). We have the following cases:

- If salt ≠ salt', then we have a collision for Com.
- If (state, ρ)ᵢ ≠ (state', ρ')ᵢ, then we have at least one index i satisfying (state, ρ) ≠ (state', ρ'). Since the two transcripts are valid, we have comᵢ = Com(salt, stateᵢ, ρᵢ) = Com(salt, stateᵢ', ρᵢ'). This implies a collision for Com.
- If (comᵢ, [S]ᵢ) ≠ (comᵢ', [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). ID₉ for 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 MiRitH = FS₉[ID₉₉, H, y] 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 EUF-CMA-secure in the (Q)ROM. (If P₁ employs GetPath, then we need the collision-resistance property of Reconst.)

S-DEO and MBS security: MiRitH employs FS₉ with aux₁ = (salt, ρ) and aux₂ = (salt, ρ). Therefore, h₁ and h₂ 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 sk = (seedₑₙ, M₀) and seedₑ produces M₁, ..., Mₙ. As the attack against Biscuit, we keep seedₑ and modify M₀ into M₀'. If the signature is fixed, then on the second message a₂ = ([S], [V])₉∈[N], we have [S] = R · [Mₑₙ] + [A] = R · [Mₑₙ] + [A] and [V] = S · [K] − R · [Mₑₙ] + [C] = S · [K] − R · [Mₑₙ] + [C], for i ∈ [N] \ {i'}, which implies

\[ R \cdot ([Mₑₙ] - [Mₑₙ']) = O, \]

(2)

where [Mₑₙ'] ∈ Fₑₙ is the concatenation of [Mₑₙ] and [Mₑₙ]. Due to the computation of [Mₑₙ], Equation 2 holds for any i ≠ 1. Therefore, if, for e ∈ {r}, i' ≠ 1 and R⁴ · (M₀ - M₀') = O hold, then Equation 2 and the signature is valid for modified ok' = (seedₑₙ, M₀). In other words, if we can find such good (R⁴, ..., R⁴) = γᵣ(h₁) with M₀', we can mount S-CEO and M-S-UEO attacks.
Let us calculate a probability \( p \), that the above holds for random signature. Let \( T \) be the set of indices satisfying \( i'_j = 1 \), that is, \( T = \{ i \in [r] : i'_j = 1 \} \) and let \( r' \) be the number of such indices. We can find \( M'_R \) by taking a non-trivial vector \( a \) from the intersection of kernels \( \bigcap_{i \in [r']} \ker(R^{(i)}) \) and setting \( M'_R = M_0 + [a, 0, \ldots, 0] \) if and only if \( \bigcap_{i \in [r']} \ker(R^{(i)}) \neq \{0\} \). The condition can be written as \( \text{rank}([R^{(i)}; \ldots; R^{(r')}]) < m \), where, for \( A \in \mathbb{F}_q^{m \times m} \) and \( B \in \mathbb{F}_q^{m \times m} \), \([A; B]\) denotes the block matrix \( \begin{pmatrix} A & B \end{pmatrix} \in \mathbb{F}_q^{(m+m) \times m} \). Using this argument, we can compute \( p \) as

\[
\begin{align*}
    p := \sum_{r' \in [r]} p_{\text{num}, r'} \cdot p_{\text{rank}, r'},
\end{align*}
\]

where \( p_{\text{num}, r'} := \Pr_{\left[i_1, \ldots, i_{r'}\right] \in [r]} \left[ i_j = 1 \right] = r' \) and \( p_{\text{rank}, r'} := \Pr_{\left[i_1, \ldots, i_{r'}\right] \in [r]} \left[ \text{rank}([R_{i_1}; \ldots; R_{i_{r'}}]) < m \right] \Pr_{\left[i_1, \ldots, i_{r'}\right] \in [r]} \left[ \text{rank}(R_{i}) < m \right] \). By routine calculation, we have

\[
\begin{align*}
    p_{\text{num}, r'} &= \left( \frac{r'}{r} \right) \frac{(N-1)^{r'} / N^r}{1}, \\
    p_{\text{rank}, r'} &= \begin{cases} 
    1 & \text{if } s(r - r') < m, \\
    1 - \prod_{j=r'}^{r-1} (1 - q^{-j}) & \text{otherwise.}
    \end{cases}
\end{align*}
\]

We note that \( 1 - \prod_{j=r'}^{r-1} (1 - q^{-j}) \leq 2m q^{-((r-r')-m+1)} \). Thus, if \( s(r - r') \) 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 \( p_0 \) defined by

\[
\begin{align*}
    p_0 := 1 - (1 - p_1)^{Q},
\end{align*}
\]

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.

<table>
<thead>
<tr>
<th>name</th>
<th>q</th>
<th>m</th>
<th>k</th>
<th>r</th>
<th>s</th>
<th>N</th>
<th>τ</th>
<th>p1</th>
<th>pQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ia-f</td>
<td>16</td>
<td>15</td>
<td>78</td>
<td>6</td>
<td>5</td>
<td>16</td>
<td>39</td>
<td>2</td>
<td>134920</td>
</tr>
<tr>
<td>Ia-s</td>
<td>16</td>
<td>15</td>
<td>78</td>
<td>6</td>
<td>9</td>
<td>256</td>
<td>19</td>
<td>2</td>
<td>132591</td>
</tr>
<tr>
<td>IIIa-f</td>
<td>16</td>
<td>19</td>
<td>19</td>
<td>142</td>
<td>4</td>
<td>5</td>
<td>16</td>
<td>55</td>
<td>192739</td>
</tr>
<tr>
<td>IIIa-s</td>
<td>16</td>
<td>19</td>
<td>19</td>
<td>142</td>
<td>4</td>
<td>6</td>
<td>256</td>
<td>29</td>
<td>207346</td>
</tr>
<tr>
<td>Va-f</td>
<td>16</td>
<td>21</td>
<td>19</td>
<td>189</td>
<td>7</td>
<td>7</td>
<td>16</td>
<td>74</td>
<td>272148</td>
</tr>
<tr>
<td>Va-s</td>
<td>16</td>
<td>21</td>
<td>21</td>
<td>189</td>
<td>7</td>
<td>10</td>
<td>256</td>
<td>318</td>
<td>278478</td>
</tr>
</tbody>
</table>

Table 4. Parameter sets in MiRitH’s specification v1.0 and success probability with \( Q = 2^{64} \).
1. \( P_s(\text{sk}) \) for PERK
2. Choose salt and mseed uniformly at random
3. \((\text{seed}^{(1)}, \ldots, \text{seed}^{(t)})\) := PRG(salt, mseed)
   \(/\text{Run the following procedure in parallel for \( e \in [t] \)}\)
4. \((\text{seed}_e, \rho_e)_{e \in [N]} := \text{TreePRG}(\text{seed}, \text{salt})\)
5. for \( i = N \) to 2 do
6. \((\pi_i, v_i) := \text{MakeShares}(\text{seed}_i, \text{salt})\)
7. state := seed,
   \(/\text{The second part only for } i = 1\)
8. \( v_1 := \text{MakeShares}(\text{seed}_1, \text{salt})\)
9. \( \pi_1 := \pi_1^{-1} \circ \cdots \circ \pi_1^{-1} \circ \pi_1 \circ \pi_1 \circ \pi_1 \circ \pi_1 \)
10. state := seed,
11. for all \( i \in [N] \): \( \text{com}_{i,j} := \text{Com}((\text{salt}, e, i, \text{state}), \rho_i)\)
12. \( v := v_N + \sum_{i=1}^{N-1} \pi_i^{-1} \cdots \pi_i^{-1} (\theta)\)
13. \( \text{com}_{i} := H_1(\text{state}, e, H_0)\)
14. \( a_i := (\text{com}_{i}, (\text{com}_{i,j})_{j\in[N]}(\pi_i))\)
15. state := (\text{salt}, (\text{state}), \rho_i)_{i\in[N]}(\text{com}_{i,j})_{j\in[N]}(\pi_i)_{i\in[N]}\)
16. return \( a_i \) and state

Fig. 17. Prover, reconstruction, and verification algorithms for ID\textsubscript{PERK}. We run the protocol in \( \tau \)-parallel way sharing salt.

- MakeShares generates pseudorandom shares \((\pi_i^{(j)}, \theta_i^{(j)})\) from the seed \text{seed}^{(j)} with an auxiliary information
  salt, where \( \pi_i^{(j)} \in S_i \) and \( \theta_i^{(j)} \in E_i \).
- In \( P_s \), \( a_3 \) 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 PERK = FS\textsubscript{n}[ID\textsubscript{PERK}, H, y] is defined by aux\textsubscript{1} = (0x01, salt, \( \mu, \text{ok} \)) and aux\textsubscript{2} = (0x02, salt, \( \mu \)).

E.1 Security

\textit{sEUF-CMA security:} Since we modify the protocol, we need to modify the simulator, which is described
in Figure 18. The HVZK property of ID\textsubscript{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 \( \text{Sim} \) yields \( \text{q}\text{-HVZK} \) for polynomial \( q = q^*(1^t) \) 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 (\textit{q}\text{-HVZK}). Suppose that PRG, TreePRG, and MakeShares are pseudorandom and Com is hiding.
Then, ID\textsubscript{PERK} with simulator Sim\textsubscript{PERK} in Figure 18 is \( \textit{q}\text{-HVZK} \).

Lemma E.2 (Strong non-divergency). Suppose that \( H_0 \) is collision-resistant. Com is non-invertible and collision-resistant.
Then, ID\textsubscript{PERK} is \( \textit{q}\text{-}\textit{q}\text{-non-divergent} with respect to Sim\textsubscript{PERK}.

Proof. For simplicity, we ignore parallelness \( \tau \). Suppose that the adversary declines a valid transcript \( \text{trans}_s = (a_1, c_1, d_1, a_2) \) generated by the simulator and outputs a valid transcript \( \text{trans}' = (a_1, c_1, d_1', c_2') \). Note that
they are valid and share \( a_1 \) and \( c_1 \). We parse them as \( a_1 = (\text{com}_0, \text{com}_1, \ldots, \text{com}_{1/2}) \).
Notice that the adversary opens \textit{MBS} aux \( i \), and both \( a \) auxs can be written as \((\mu, \eta i)\) and \((\mu, \eta j)\), respectively. Hence, \textit{PERK} satisfies wNR due to Lemma 5.3.

\textbf{Theorem E.2.} Assume that \( H \) is collision-resistant. Then, \( \text{PERK} = \text{FS} [ \text{ID\_{PERK}} , H, y ] \) satisfies MBS and M-S-UEO.

\textbf{BUFF security:} Recall that aux = \((0x01, \text{salt}, \mu, \eta k)\) and aux2 = \((0x02, \text{salt}, \mu)\) in \( \text{PERK} \). It is obvious that aux is perfectly collision-resistant with respect to the message on index 1. Thus, applying Lemma 5.2, \( \text{PERK} \) satisfies MBS and M-S-UEO. In addition, aux is perfectly collision-resistant with respect to the verification key on index 1, and both aux and aux2 can be written as \((\mu, \eta i)\) and \((\mu, \eta j)\), respectively. Hence, \( \text{PERK} \) satisfies wNR due to Lemma 5.3.

\textbf{Theorem E.1.} (\textit{PERK}'s sEUF-CMA security). Suppose that \( \text{PERK} = \text{FS} [ \text{ID\_{PERK}} , H, y ] \) is EUF-NMA-secure in the \( \text{QROM} \), \( \text{PRG} \), \( \text{TreePRG} \), and \text{MakeShares} are pseudorandom, \( H \) is collision-resistant, \( \text{Com} \) is hiding, non-invertible, and collision-resistant. Then, \( \text{PERK} \) is sEUF-CMA-secure in the \( \text{QROM} \). (If \( \mathcal{P} \) employs GetPath, then we need the collision-resistance property of \textit{Reconst}.)
F AIMer

We briefly review AIMer [KCC+23].

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

1. Let \( a_0 := H(\mu, \eta) \), and output \( \sigma := (h_1, h_2, a_0) \).

2. Compute \( h_1 := H(\mu, \eta) \).

3. Compute \( h_2 := H(\eta, h_1) \).

4. Compute \( a_1 := H\left(\mu, H(h_1)\right) \).

5. Output \( \sigma := (h_1, h_2, a_1) \).

We can consider AIMer as \( F_{\text{AIMer}}[\text{ID}_{\text{AIMer}}, H, y] \) with \( \text{aux}_1 = (\mu, \eta, \text{salt}) \) and \( \text{aux}_2 = (\eta, \text{salt}) \). \( \text{aux} \) 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 \( \varepsilon\text{EUF-CMA-secure in the (Q)ROM if it is EUF-NMA-secure in the (Q)ROM and used primitives are secure.} \) Since \( \text{aux} \) 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, \( \text{aux} \) can be written as \( (\mu, \eta, \text{salt}) \). Hence, AIMer is \( w\text{NR-secure if } H \) 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 \( F_q \) via MPC using polynomials and the extension field \( F_p \) 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, H{"u}lsing, 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 \( q\text{-HVZK by tracing their proof. It is also easy to check the protocol is strongly non-divergent under appropriate assumptions on the primitives used in the protocol.} \)

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 \( \text{ID}_{\text{MIRA}} \) be the underlying 5-pass MPCitH protocol. Let \( DS\_1, DS\_2, DS\_M \in \{0, 1\}^* \) be domain separators for the random oracle \( H \). In MIRA, the signature is computed as follows:

1. Let \( md := H_{\mu}(\mu) \), where \( H_{\mu}(\mu) = H(DS\_M, \mu) \).
2. Compute \( h_1 := H(DS\_1, salt, md, a_0) \), and \( c_1 := y(h_1) \).
3. Compute \( h_2 := H(DS\_2, salt, md, h_1, a_0) \), and \( c_2 := y(h_2) \).
4. Compute \( a_1 \), which includes salt, and output \( \sigma := (h_1, h_2, a_1) \).

The verification algorithm verifies a signature as follows:

1. Let \( md := H_{\mu}(\mu) \).
2. Compute \( c_1 := y(h_1) \) and \( c_2 := y(h_2) \).
3. Reconstruct \( a_1 \) and \( a_2 \) from \( c_1, c_2, a_0 \).
4. Compute \( h_1 := H(DS\_1, salt, md, a_1) \) and \( h_2 := H(DS\_2, salt, md, h_1, a_2) \).
5. Output \( \sigma := (h_1, h_2, a_1) \).

Thus, we can consider MIRA as \( F_{\text{MIRA}}[\text{ID}_{\text{MIRA}}, H, y] \) with \( \text{aux}_1 = (DS\_1, salt, md) \) and \( \text{aux}_2 = (DS\_2, salt, md) \), where \( md := H(DS\_M, \mu) \). \( \text{aux} \) 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 \( \varepsilon\text{EUF-CMA-secure in the (Q)ROM if it is EUF-NMA-secure in the (Q)ROM and used primitives are secure.} \) Since \( \text{aux} \) 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 \( \mu \) with \( md \), we can apply Lemma 5.3 and show that MIRA is \( w\text{NR-secure if } H \) is the random oracle.
G.2 SDitH – SDitH-HC

We briefly review SDitH v.1.1 [AFG’23]. Here, we only consider the hypercubic MPCitH version, which we call SDitH-HC. Let ID_{SDitH-HC} be the underlying 5-pass MPCitH protocol. In SDitH-HC, the signature is computed as follows:

1. Compute \( a_1, h_1 := H(0x01, salt, v_k, a_1) \), and \( c_1 := y(h_1) \).
2. Compute \( a_2, h_2 := H(0x02, salt, \mu, h_1, a_2) \), and \( c_2 := y(h_2) \).
3. Compute \( a_3 \), which includes salt, and output \( \sigma := (h_3, a_3) \).

The verification algorithm verifies a signature as follows:

1. Compute \( c_3 := y(h_3) \).
2. Reconstruct \( \hat{a}_1 \) from \( c_1 \) and \( a_1 \).
3. Compute \( \hat{h}_1 := H(0x01, salt, v_k, \hat{a}_1) \) and \( \hat{c}_1 := y(\hat{h}_1) \).
4. Reconstruct \( \hat{a}_2 \) from \( c_1 \) and so on. and \( \hat{h}_2 := H(0x02, salt, \mu, h_1, \hat{a}_2) \).
5. Output bool(\( h_2 = \hat{h}_2 \)).

Thus, we can consider SDitH-HC as \( F_{SDitH-HC}(ID_{SDitH-HC}, h, y) \) with \( aux_1 = (0x01, salt, v_k) \) and \( aux_2 = (0x02, salt, \mu) \). aux is collision-resistant with respect to 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 \( \hat{h}_1 \) is not in the signature. If there is an adversary against the M-S-UEO security, then its output contains two different verification keys \( v_k \) and \( v'_k \), two messages \( \mu \) and \( \mu' \), and a signature \( \sigma = (h_2, a_2) \), where \( a_2 \) contains salt. Let \( a_1 \) (or \( a'_1 \), resp.) be the first messages reconstructed from \( v_k \) (or \( v'_k \), resp.), \( a_3 \), and \( c_2 = y(h_2) \). Then we let \( \hat{h}_1 := H(0x01, salt, v_k, a_1) \) and \( \hat{h}_1' := H(0x01, salt, v'_k, a_1) \).

- If \( h_1 = \hat{h}_1 \), then we find a collision of \( h \).
- Otherwise, we let \( h_2 := H(0x02, salt, \mu, h_1, a_2) \) and \( \hat{h}_2 := H(0x02, salt, \mu, \hat{h}_1', a_1') \). Since the signature is valid for both messages and verification keys, we have \( h_2 = \hat{h}_2 \) and find a collision of \( h \).

Thus, if \( h \) is collision-resistant, then SDitH-HC is M-S-UEO-secure. If we consider SDitH-HC’ := \( F_{SDitH-HC}(ID_{SDitH-HC}, H, y) \), 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 \( (\mu, \eta_2) \). Corollary B.2 states that if \( F_{SDitH-HC}(ID, H, y) \) is wNR-secure, then \( F_{SDitH-HC}(ID, H, y) \) 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 \( a_1 \). 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 ID_{MQOM} be the underlying 7-pass MPCitH protocol. In MQOM, the signature is computed as follows:

1. Compute \( a_1, h_1 := H(0x01, salt, v_k, \mu, a_1) \), and \( c_1 := y(h_1) \).
2. Compute \( a_2, h_2 := H(0x02, salt, \mu, h_1, a_2) \), and \( c_2 := y(h_2) \).
3. Compute \( a_3, h_3 := H(0x03, salt, \mu, h_2, a_3) \), and \( c_3 := y(h_3) \).
4. Compute \( a_4 \), which includes salt, and output \( \sigma := (h_4, h_5, h_6, a_4) \).

The verification algorithm verifies a signature as follows:

1. Compute \( c_4 := y(h_4) \), \( c_5 := y(h_5) \), and \( c_6 := y(h_6) \).
2. Reconstruct \( (\hat{a}_1, \hat{a}_2, \hat{a}_3) \) from \( c_1, c_2, c_3, a_4 \).
3. Compute \( h_1 := H(0x01, salt, v_k, \mu, a_1), h_1 := H(0x02, salt, \mu, h_1, \hat{a}_2), h_1 := H(0x03, salt, \mu, h_2, \hat{a}_3), \) and \( h_1 := H(0x04, salt, \mu, h_3, \hat{a}_4) \).
4. Output bool(\( h_1 = \hat{h}_1, h_2 = \hat{h}_2, h_3 = \hat{h}_3 \), \( h_4 \)).

Thus, we can consider MQOM as \( F_{MQOM}(ID_{MQOM}, H, y) \) with \( aux_1 = (0x01, salt, v_k, \mu), aux_2 = (0x02, salt, \mu), \) and \( aux_3 = (0x03, salt, \mu) \). aux is perfectly collision-resistant with respect to message and verification key on index 1.

We can routinely show \( ID_{MQOM} \)’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 can be written as \( (\mu, \eta_3) \). 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 \(x\) and ciphertext \(y\) such that \(y = Enc_d(x)\). Additionally, the prover (signer) and verifier interact with an ideal functionality \(F_{VOLE}\) which generates correlated random values \((u, v, \Delta, q)\) such that \(q = u \cdot \Delta + v\) and sends \((u, v)\) to the prover and \((\Delta, q)\) 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 \([YSWW^{21}]\) with the help of VOLE correlation \(q = w \cdot \Delta + v\), to check that on the input witness \(w\) and verification key \((x, y)\) the AES circuit evaluates to 1.

In order to achieve the desired security level (such as 128-bit security) the above protocol is repeated \(\tau\) 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 \(ID_{FAEST} = (P_1, P_2, P_3, P_4)\) 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 \(F_{VOLE}\) 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 \(\Delta\). 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 \(\Delta\). 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 \(\Delta\) 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 \(u_i\)) instead of \(\tau\) 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 \(\tau\) independent 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 \(\tau\) VOLE instances efficiently.
- **ExtendWitness**: Extends the secret key \(sk\) to VOLE witness \(w\).
- **Lines 5 to 18 of \(P_1\) 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 \(ch\) generated by the verifier after receiving all-but-one leaves of the GGM tree. Specifically, the values \(\Delta\) 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 \(\tau\) 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/
Fig. 19. Prover algorithms for ID_{FAEST}.
Choose salt, mseed at random

Sample (seed)\textsubscript{3} := PRG\textsubscript{3}(salt, mseed)

//Generate VOLE secrets and tags

for \( i = 1 \) to \( \tau \) do

Compute \((\text{com}\textsubscript{i},\text{dec}\textsubscript{i},u,V_i)\) from salt and seed, using length-doubling PRG

\( V := [V_0 V_1 \ldots V_\tau] \)

\( u := u_1 \)

for \( i = 1 \) to \( \tau - 1 \) do

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

//Commit to VOLE secrets, tags, and commitments

\( h_{\text{com}} := H((\text{com}\textsubscript{i},\text{com}\textsubscript{i+1}) \ldots \text{com}_{\tau}) \)

\( a_1 := (h_{\text{com}},c_{i+1}) \)

Choose \( d \) uniform randomly

Set \( w := d \oplus u \)

Compute \( \Delta \) from \( c_{i+1} \)

Set \( V := V + d\Delta \)

//Universal hash for VOLE consistency

\( \hat{u} := \text{UniversalHash}(c_{i+1}, u) \)

\( \hat{V} := \text{UniversalHash}(c_{i+1}, V) \)

\( h_{\text{V}} := H(\hat{V}) \)

\( a_2 := (\hat{u},h_{\text{V}},d) \)

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

\( a_3 := (\hat{a},\hat{b}) \)

//Generate partial decommitments for VOLE

for \( i = 1 \) to \( \tau \) do

\( \text{dec}\textsubscript{i} := \text{PartialOpen}(c_{i+1}, \text{dec}\textsubscript{i}) \)

\( a_4 := (\text{dec}\textsubscript{i},\text{com}\textsubscript{i+1},\hat{u},\hat{d},\hat{a},(\text{dec}\textsubscript{i},\text{com}\textsubscript{i+1}),\text{salt}) \)

return \( a_1, a_2, a_3 \) and \( a_4 \)

\( \text{Sim}_{\text{FAEST}}(sk, ch_1, ch_2, ch_3) \)

\( \text{Parse } a_1 = ((c_{i+1})_{\text{com}\textsubscript{i+1}},\hat{u},\hat{d},\hat{a},(\text{dec}\textsubscript{i},\text{com}\textsubscript{i+1}),\text{salt}) \)

//Reconstruct VOLE correlations

Compute \((\tilde{a},\tilde{b}) := \text{VOLEReconstruct}(ch_3, (\text{dec}\textsubscript{i},\text{com}\textsubscript{i+1}), \text{salt}) \)

//Apply VOLE corrections

\( (Q,D) := \text{VOLECorrect}(\hat{u},c_{i+1}) \)

\( \hat{Q} := \text{UniversalHash}(c_{i+1}, Q) \)

\( \hat{D} := H_2(\hat{Q} \oplus D) \)

//Verify AES relation

\( \tilde{a},\tilde{b},\hat{Q},\hat{D} \)

//Generate partial decommitments for VOLE

for \( i = 1 \) to \( \tau \) do

\( \text{dec}\textsubscript{i} := \text{PartialOpen}(c_{i+1}, \text{dec}\textsubscript{i}) \)

\( a_4 := (\text{dec}\textsubscript{i},\text{com}\textsubscript{i+1},\hat{u},\hat{d},\hat{a},(\text{dec}\textsubscript{i},\text{com}\textsubscript{i+1}),\text{salt}) \)

return \( a_1, a_2, a_3 \) and \( a_4 \)

//Reconstruction, verification, and simulation algorithms for \( \text{ID}_{\text{FAEST}} \).
Security of FAEST Signature

Lemma H.1 \((q_3,{-}\text{HVZK})\). Suppose that PRG is a length-doubling secure PRG and \(H_1\) is a hash function modelled as random oracle. Let \(q_3\) be a polynomial of \(1^*\). Then, \(\text{ID}_{\text{FAEST}}\) with simulator \(\text{Sim}_{\text{FAEST}}\) in Figure 20 is \(q_3,{-}\text{HVZK}\).

Proof. The length-doubling secure PRG PRG is used to implement the ideal functionality \(\mathcal{F}_{\text{VOLER}}\) using the GGM trees. The rest of the proof follows from the proof for the malicious verifier case from SoftSpoken [Roy22] and QuickSilver [YSW21] protocols, as explained in [BBD*23b].

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

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

When \(ch_1 \neq ch_1'\): In this case, let \(\overline{r}_{\text{com}}\) and \(\overline{r}_{\text{aux}}\) be the values recovered by running \(\text{VOLEReconstruct}\) with inputs \(ch_1\) and \(ch_1'\) respectively. Then if \(\overline{r}_{\text{com}} = \overline{r}_{\text{aux}}\) we have found a collision (during internal computation of \(\text{VOLEReconstruct}\)) 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 \((a_1, a_2, a_3, a_4) = (a_1, a_1', a_2', a_2')\) therefore the only possible case is \((\langle \text{pdec}\rangle_{\text{com}(r)}, \text{salt}) \neq (\langle \text{pdec}\rangle_{\text{aux}(a)}, \text{salt})\). If \((\langle \text{pdec}\rangle_{\text{com}(r)} \neq (\langle \text{pdec}\rangle_{\text{aux}(a)}\rangle_{\text{aux}(a)})\) then again during computation of \(\overline{r}_{\text{com}}\) and \(\overline{r}_{\text{aux}}\) from \(\text{VOLEReconstruct}\) with inputs \((\langle \text{pdec}\rangle_{\text{com}(r)}\rangle_{\text{com}(r)}\) and \((\langle \text{pdec}\rangle_{\text{aux}(a)}\rangle_{\text{aux}(a)}\) respectively we can find a collision (during the internal computation of \(\text{VOLEReconstruct}\)) for either the hash function \(H_1\) or \(H_0\). Similarly, when salt \(\neq \text{salt'}\) we can find a collision for the hash function \(H_0\) while computing \(\overline{r}_{\text{com}}\) and \(\overline{r}_{\text{aux}}\) from \(\text{VOLEReconstruct}\) with inputs salt and salt' respectively.

\(\square\)

Since Rep is decomposable, we can obtain signature scheme \(\text{FAEST} = \text{FS}_{\text{FAEST}}[\text{ID}_{\text{FAEST}}, H, y]\) as follows: \(^{18}\)

- Let \(H\) be a random oracle.
- Let \(y := (y_1, y_2, y_3)\), where \(y_1\) is identity function for \(i \in \{1, 2, 3\}\).
- For message \(m\), compute \(M := H(0x01, \sigma, m)\).
- Set \(\text{aux}_1 := (0x02, 0x01, M, \text{salt})\) and \(h_1 := H(\text{aux}_1, a_1)\).
- Set \(\text{aux}_2 := (0x02, 0x02)\) and \(h_2 := H(aux_2, h_1, a_2)\).
- Set \(\text{aux}_3 := (0x02, 0x03)\) and \(h_3 := H(\text{aux}_2, h_3, a_3)\).

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

Theorem H.1. Suppose that \(\text{FAEST} = \text{FS}_{\text{FAEST}}[\text{ID}_{\text{FAEST}}, H, y]\) is EUF-NMA-secure in the (Q)ROM, PRG is length-doubling PRG, UniversalHash, ZKHash are hiding universal hashes Then, \(\text{FAEST}\) 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 \(\text{FAEST}\) 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 \(\text{FAEST}' := \text{FS}_{\text{FAEST}}[\text{ID}_{\text{FAEST}}, H, y]\) is wNR-secure if \(H\) is the random oracle. We can show this by modifying the wNR security proof for \(\text{FS}_{\text{FAEST}}\) in subsection A.3 as follows.

- \(G_0\): This is the original wNR game with \(\text{FAEST}'\).
- \(G_1\): In this game, if the adversary outputs \(\sigma' = \sigma\) such that \(M = H(0x01, \sigma, \mu) = H(0x01, \sigma', \mu)\), then the adversary loses. Since we have a collision \((0x01, \sigma, \mu) \neq (0x01, \sigma', \mu)\) 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 \(\sigma\) and \(\sigma\) to the adversary, we reprogram the point \((0x01, \sigma, \mu)\) 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 \((0x01, \cdot, \mu)\) with \(\perp\). Since the adversary cannot obtain any information of the hash value \(M' = H(0x01, \sigma', \mu)\), 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 \(0x01, 0x02, 0x03\) in the computation of \(h\) values to split domains while the original specification implicitly did it by the length of inputs and outputs.