History-Free Sequential Aggregate Signatures from Generic Trapdoor Functions

Alessio Meneghetti[1][0000−0002−5159−7252] and Edoardo Signorini[2][0000−0002−1224−6732]

[1] University of Trento, Trento, Italy
alessio.meneghetti@unitn.it
[2] Telsy, Turin, Italy
edoardo.signorini@telsy.it

Abstract. A sequential aggregate signature (SAS) scheme allows multiple users to sequentially combine their respective signatures in order to reduce communication costs. Historically, early proposals required the use of trapdoor permutation (e.g., RSA). In recent years, a number of attempts have been made to extend SAS schemes to post-quantum assumptions. Many post-quantum signatures have been proposed in the hash-and-sign paradigm, which requires the use of trapdoor functions and appears to be an ideal candidate for sequential aggregation attempts. However, the hardness in achieving post-quantum one-way permutations makes it difficult to obtain similarly general constructions. Direct attempts at generalizing permutation-based schemes have been proposed, but they either lack formal security or require additional properties on the trapdoor function, which are typically not available for multivariate or code-based functions. In this paper, we propose a history-free sequential aggregate signature based on generic trapdoor functions, generalizing existing techniques. We prove the security of our scheme in the random oracle model by adopting the probabilistic hash-and-sign with retry paradigm, and we instantiate our construction with three post-quantum schemes, comparing their compression capabilities. Finally, we discuss how direct extensions of permutation-based SAS schemes are not possible without additional properties, showing the insecurity of two existing multivariate schemes when instantiated with Unbalanced Oil and Vinegar.

Keywords: sequential aggregate signature · post-quantum cryptography · hash-and-sign

1 Introduction

An Aggregate Signature (AS) scheme allows n users to combine their individual signatures on separate messages to produce a single, directly verifiable aggregate signature. This approach aims to achieve shorter signature lengths compared to trivial concatenation of individual signatures. Aggregate signatures have interesting applications in various network scenarios with high communication costs, such as PKI certificate chains or secure routing protocols authentication. The concept of aggregate signatures was initially introduced in a seminal paper by Boneh, Gentry, Lynn, and Shacham [8]. They proposed a method that allows any participant to aggregate signatures from distinct users using a public aggregation algorithm. Although this general aggregation approach is efficient and valuable, it is limited to the use of bilinear pairings. Another variant of aggregate signatures, known as Sequential Aggregated Signature (SAS), was introduced by Lysyanskaya, Micali, Reyzin, and Shacham [24]. In SAS schemes, signatures are aggregated in a specific sequence, starting from the so-far aggregated signature and possibly from the public key and message information of previous users. The sequential structure is still beneficial in many applications, and numerous works have been pursued in this direction, proposing constructions based on trapdoor permutations [24,25,10,19] or the use of bilinear pairings [23,2,18].

Aggregate Signatures from Post-Quantum Assumptions In recent years, there has been a growing interest in post-quantum signatures, leading researchers to explore AS schemes in this field. Lattice-based assumptions have proven quite successful in this direction, with generic solutions based on non-interactive arguments [15,1] and sequential aggregations both in the Fiat-Shamir [9] and Hash-and-Sign paradigms [16,28]. Focusing on the latter, both [16,28] have extended previous trapdoor permutation-based approaches [25,19], but their security relies on the collision-resistance property of lattice trapdoor Preimage Sampleable Functions (PSF) [20]. These additional properties are not available for generic trapdoor functions employed for instance in multivariate-quadratic-based (MQ-based) or code-based signature schemes. Currently, SAS schemes based on these assumptions are very limited, with only two existing MQ-based schemes [17,12], which follow the construction of [24]. Unfortunately, both [17,12] lack formal security and there are instances of the underlying function for which they are insecure, as outlined below.

Our Contribution In this work, we address the extension of permutation-based SAS schemes to generic trapdoor functions, making them applicable to a wider range of post-quantum signatures. In Section 3, we present a history-free sequential aggregate signature scheme based on generic trapdoor functions. In a history-free SAS, signers receive only the so-far aggregated signature without requiring previous users' public keys and messages. Our approach builds upon the work of Brogle, Goldberg, and Reyzin [10] for trapdoor permutations, adapting the encoding technique of [25,16] to include trapdoor functions beyond permutations. We adopt the probabilistic hash-and-sign with retry paradigm, which is common in post-quantum signature constructions and crucial for security proofs.

The security of our scheme is analyzed within the partial-signature history-free security model [13], reducing to the one-wayness of the trapdoor function and to an additional notion of Preimage Sampling (PS) indistinguishability introduced in [22]. We acknowledge that our construction is not fully black-box, as it requires proving or assuming the PS notion for each specific trapdoor function. Nevertheless, this requirement is reasonable for post-quantum trapdoor functions due to existing results related to their use in signature schemes.

In Section 4, we demonstrate the applicability of our scheme to MQ-based signature schemes, specifically UOV [21] and MAYO [5], and the code-based scheme Wave [14]. For each scheme, we evaluate its compression capabilities and review its PS security so that it can be covered in our security proof. Lattice-based schemes, such as the NIST PQC finalist Falcon [26], can also benefit from history-free aggregation. However, we already noted how different design choices become feasible due to the additional properties of trapdoor PSF.

Finally, in Section 5, we argue that the simpler approaches of [24,25] are not viable for generic trapdoor functions. As evidence, we show how two existing MQ-based aggregate signature schemes [17,12] are universally forgeable when instantiated with UOV and discuss their lack of provable security.

2 Notation and Preliminaries

For $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, \ldots, n\}$. For a finite set X, we write $|X|$ for the cardinality of X and $\mathrm{len}(X)$ for the bit size of an element in X. By $x \leftarrow\!\!\$\ X$ we denote the sample of the element x from $U(X)$, the uniform distribution over X. For an algorithm A, we write $x \leftarrow A(y)$ to denote the assignment of x to the output of A on input y. For an adversary $\mathcal{A}$ and a function F, we write $x \leftarrow \mathcal{A}^{\mathsf{OF}}$ the assignment of x of the output of $\mathcal{A}$ with oracle access to F. For two bit strings $x, y \in \{0,1\}^*$, we denote by $x \,\|\, y$ the bit string obtained by their concatenation. We write $\mathbb{F}_q$ for a finite field of q elements. We denote by $\mathbb{F}_q^{m\times n}$ the set of matrices over $\mathbb{F}_q$ with m rows and n columns. $\mathbf{I}_{n\times n}$ is the identity matrix of size n. $\mathbf{0}_{m\times n}$ is the $m \times n$ zero matrix and $\mathbf{0}_n$ is the zero vector in $\mathbb{F}_q^n$. In the remainder of this section, we introduce standard definitions and notions related to digital signature schemes based on trapdoor functions.

2.1 Trapdoor Functions

Definition 1. *A trapdoor function (TDF)* T *is a tuple of four algorithms* $(\mathsf{TrapGen}, \mathsf{F}, \mathsf{I}, \mathsf{SampDom})$*:*

- $\mathsf{TrapGen}(1^\lambda)$*: takes as input a security parameter* 1^λ *and generates an efficiently computable function* $\mathsf{F}\colon \mathcal{X} \to \mathcal{Y}$ *and a trapdoor* I *that allow to invert* F.
- $\mathsf{F}(x)$*: takes as input* $x \in \mathcal{X}$ *and outputs* $\mathsf{F}(x) \in \mathcal{Y}$.
- $\mathsf{I}(y)$*: takes as input* $y \in \mathcal{Y}$ *and outputs* $x \in \mathcal{X}$ *such that* $\mathsf{F}(x) = y$ *or it fails by returning* $\perp$.
- $\mathsf{SampDom}(\mathsf{F})$ *takes as input a function* $\mathsf{F}\colon \mathcal{X} \to \mathcal{Y}$ *and outputs* $x \in \mathcal{X}$.

We define the standard notion of one-wayness (OW) for a trapdoor function.

Algorithm 1: Hash-and-sign with retry

```
KGen($1^\lambda$):
 1: $(\mathsf{F}, \mathsf{I}) \leftarrow \mathsf{TrapGen}(1^\lambda)$
 2: return $(\mathsf{F}, \mathsf{I})$

Vrfy($\mathsf{F}$, $m$, $(r, x)$):
 1: return $\mathsf{F}(x) = \mathsf{H}(r, m)$

Sign($\mathsf{I}$, $m$):
 1: repeat
 2:     $r \leftarrow_\$ \{0,1\}^\lambda$
 3:     $x \leftarrow \mathsf{I}(\mathsf{H}(r, m))$
 4: until $x \neq \bot$
 5: return $(r, x)$
```

Definition 2. *Let* $\mathsf{T} = (\mathsf{TrapGen}, \mathsf{F}, \mathsf{I}, \mathsf{SampDom})$ *be a TDF and let* $\mathcal{A}$ *be an adversary. We define the advantage of* $\mathcal{A}$ *playing the* OW *game against* T *as*

$$\mathsf{Adv}_{\mathsf{T}}^{\mathrm{OW}}(\mathcal{A}) = \Pr\left[\mathsf{F}(x) = y \;\middle|\; \begin{array}{c} (\mathsf{F}, \mathsf{I}) \leftarrow \mathsf{TrapGen}(1^\lambda) \\ y \leftarrow_\$ \mathcal{Y} \\ x \leftarrow \mathcal{A}(\mathsf{F}, y) \end{array} \right]$$

Definition 3 (Trapdoor Permutation (TDP)). *A TDF* $\mathsf{T} = (\mathsf{TrapGen}, \mathsf{F}, \mathsf{I}, \mathsf{SampDom})$ *is said to be a TDP if* F *and* I *are permutations.*

2.2 Digital Signatures

A digital signature scheme Sig is a tuple of three algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$:

- $\mathsf{KGen}(1^\lambda)$: takes as input a security parameter 1^λ and generates a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: takes as input a signing key sk and a message m and returns a signature σ.
- $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$: takes as input a verification key pk, a message m and a signature σ and returns $\top$ for acceptance or $\bot$ for rejection.

We define the standard notion of existential unforgeability against chosen-message attack (EUF-CMA).

Definition 4 (EUF-CMA security). *Let* O *be a random oracle, let* $\mathsf{Sig} = (\mathsf{KGen}, \mathsf{OSign}, \mathsf{OVrfy})$ *be a signature scheme, let* $\mathcal{A}$ *be an adversary. We define the advantage of* $\mathcal{A}$ *playing the* EUF-CMA *game against* Sig *in the random oracle model as:*

$$\mathsf{Adv}_{\mathsf{Sig}}^{\text{EUF-CMA}}(\mathcal{A}) = \Pr\left[\begin{array}{c} \mathsf{OVrfy}(\mathsf{pk}, m, \sigma) = \top \\ \mathsf{OSign}(\mathsf{sk}, \cdot) \textit{ not queried on } m \end{array} \;\middle|\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda) \\ (m, \sigma) \leftarrow \mathcal{A}^{\mathsf{O}, \mathsf{OSign}(\mathsf{sk}, \cdot)}(\mathsf{pk}) \end{array} \right]$$

2.3 Hash-and-Sign Schemes

The (probabilistic) hash-and-sign (HaS) paradigm is a standard approach to build digital signature schemes in the random oracle model from a trapdoor function T and a hash function $\mathsf{H} \colon \{0,1\}^* \to \mathcal{Y}$. To sign a message m, a signer with secret key $\mathsf{sk} = \mathsf{I}$ applies the hash function, modeled as a random oracle,

Game 1: PS_b

```
1: $(\mathsf{F}, \mathsf{I}) \leftarrow \mathsf{TrapGen}(1^\lambda)$
2: $b^\star \leftarrow \mathcal{A}^{\mathsf{Sample}_b}(\mathsf{F})$
3: return $b^\star \in \{0,1\}$

$\mathsf{Sample}_1$:
1: $r_i \leftarrow_\$ \{0,1\}^\lambda$
2: $x_i \leftarrow \mathsf{SampDom}(\mathsf{F})$
3: return $(r_i, x_i)$

$\mathsf{Sample}_0$:
1: repeat
2:     $r_i \leftarrow_\$ \{0,1\}^\lambda$
3:     $y_i \leftarrow_\$ \mathcal{Y}$
4:     $x_i \leftarrow \mathsf{I}(y_i)$
5: until $x_i \neq \perp$
6: return $(r_i, x_i)$
```

to the message $y \leftarrow \mathsf{H}(m)$ and computes its inverse $x \leftarrow \mathsf{I}(y)$ through the secret trapdoor. In some scenarios, the HaS paradigm requires the use of a random string r, which acts as a salt for the hash function, i.e. $y \leftarrow \mathsf{H}(m \parallel r)$. The resulting signature is the couple $\sigma = (x, r)$. A verifier uses the corresponding public key $\mathsf{pk} = \mathsf{F}$ to verify whether $\mathsf{F}(x) = \mathsf{H}(m \parallel r)$.

When T is a trapdoor permutation, this construction is known as Full Domain Hash and the EUF-CMA security of the signature scheme can be proved from the one-wayness assumption of T [3]. For generic TDF a black-box security proof is not known and custom reductions are needed for different constructions. This becomes particularly significant in the post-quantum era where no constructions of one-way permutations are known. In order to achieve a secure signature, it is possible to consider trapdoor functions with additional properties. For instance, Preimage Sampleable Functions (PSF) [20], which can be constructed from lattices [26], or Average Trapdoor PSF (ATPSF) [11], which can be constructed from code-based assumptions [14]. More generally, a slightly different paradigm known as probabilistic hash-and-sign with retry (Algorithm 1), is used to prove the EUF-CMA security. With this approach, a random string r is sampled until a preimage for $\mathsf{H}(m \parallel r)$ is found. The security is based on the one-wayness of the trapdoor function and on the additional condition that the output of the signing algorithm (r, x) is indistinguishable from a couple (r', x') with $r' \leftarrow_\$ \{0,1\}^\lambda$ and $x' \leftarrow_\$ \mathsf{SampDom}(\mathsf{F})$. Ad-hoc versions of this paradigm are commonly employed for MQ-based signatures, and have been utilized to prove the security of Unbalanced Oil and Vinegar (UOV), Hidden-Field Equation (HFE) [27], and MAYO [5] signature schemes.

In this work, we build a history-free sequential aggregate signature $\mathsf{HaS\text{-}HF\text{-}SAS}$ from generic trapdoor functions and the probabilistic hash-and-sign with retry approach. The security of $\mathsf{HaS\text{-}HF\text{-}SAS}$ requires the indistinguishability condition on preimages, that we formalize adopting the following notion from [22].

Definition 5 (Preimage Sampling [22]). *Let* $\mathsf{T} = (\mathsf{TrapGen}, \mathsf{F}, \mathsf{I}, \mathsf{SampDom})$ *be a TDF, let* $\mathcal{A}$ *be an adversary. We define the advantage of* $\mathcal{A}$ *playing the* PS *game (Game 1) against* T *as:*

$$\mathsf{Adv}^{\mathrm{PS}}_{\mathsf{T}}(\mathcal{A}) = |\Pr[\mathrm{PS}_0(\mathcal{A}) = 1] - \Pr[\mathrm{PS}_1(\mathcal{A}) = 1]|$$

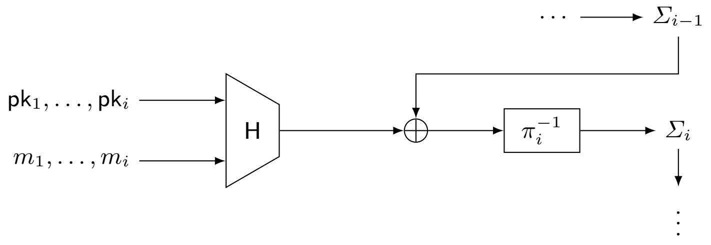

Fig. 1. High level description of SAS scheme from [24]

2.4 History-Free Sequential Aggregate Signature

History-Free Sequential Aggregate Signatures (HF-SAS) were first introduced in [10,18] as a variant of the original construction of [24] that does not require knowledge of previous messages and public keys in the aggregation step.

Definition 6 (HF-SAS)**.** *A History-Free Sequential Aggregate Signatures is a tuple of three algorithms* $(\mathsf{KGen}, \mathsf{AggSign}, \mathsf{AggVrfy})$*:*

- $\mathsf{KGen}(1^\lambda)$*: takes as input a security parameter* 1^λ *and generates a key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{AggSign}(\mathsf{sk}_i, m_i, \Sigma_{i-1})$*: takes as input the secret key* sk_i *and the message* m_i *of the* i*th user and the previous aggregate signature* Σ_{i-1}*. Returns an aggregate signature* Σ_i*.*
- $\mathsf{AggVrfy}(L_n, \Sigma_n)$*: takes as input the full history* $L_n = (\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n)$ *of public key, message pairs and an aggregate signature* Σ_n*. Returns* $\top$ *if* Σ_n *is a valid aggregate signature and* $\perp$ *otherwise.*

Every signer has a key pair $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KGen}(1^\lambda)$. The signature aggregation process is done iteratively: the first signer with keys $(\mathsf{pk}_1, \mathsf{sk}_1)$ generates a signature Σ_1 for message m_1 with $\Sigma_1 \leftarrow \mathsf{AggSign}(\mathsf{sk}_1, m_1, \varepsilon)$, where ε represents the empty string to indicate that this is the first signature in the sequence. The ith signer with keys $(\mathsf{pk}_i, \mathsf{sk}_i)$ receives an aggregate signature Σ_{i-1} from the $(i-1)$th signer and aggregate his signature on message m_i to obtain the aggregate signature $\Sigma_i \leftarrow \mathsf{AggSign}(\mathsf{sk}_i, m_i, \Sigma_{i-1})$. Note that the aggregate signature algorithm $\mathsf{AggSign}$ does not require the public keys and messages from the previous signers. Finally, the verifier can check the validity of the aggregate signature by running $\mathsf{AggVrfy}(L_n, \Sigma_n)$.

SAS schemes were originally introduced by [24] for generic trapdoor permutation with the FDH approach. The history-free variant of [10] still requires trapdoor permutation, while [18] relies on bilinear pairing. The main intuition behind the aggregation process in TDP schemes is to "embed" the previous aggregate signature into the new message to be signed. This ensures that the aggregate signature can be retrieved during the verification process, as depicted in Figure 1.

Game 2: PS-HF-UF-CMA$_\mathsf{S}$

```
1: $(\mathsf{pk}^\star, \mathsf{sk}^\star) \leftarrow \mathsf{KGen}(1^\lambda)$
2: $\mathcal{Q} \leftarrow \emptyset$
3: $(L_n, \bar{\Sigma}_n) \leftarrow \mathcal{A}^{\mathsf{O},\mathsf{OAggSign}}(\mathsf{pk}^\star)$
4: $(\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n) \leftarrow L_n$
5: if $\nexists i^\star : (\mathsf{pk}_{i^\star} = \mathsf{pk}^\star \wedge m_{i^\star} \notin \mathcal{Q})$ then
6:     return $\perp$
7: return $\mathsf{AggVrfy}(L_n, \bar{\Sigma}_n)$

$\mathsf{OAggSign}(m, \mathsf{Part}(\Sigma))$:
1: $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
2: $(\mathsf{Part}(\Sigma'), \mathsf{Comp}(\Sigma')) \leftarrow \mathsf{AggSign}(\mathsf{sk}^\star, m, \mathsf{Part}(\Sigma))$
3: return $\mathsf{Part}(\Sigma'), \mathsf{Comp}(\Sigma')$
```

The main challenge in extending previous schemes to trapdoor functions that are not permutations lies in their lack of injectivity. This issue was addressed in [16] within the context of lattice-based signatures by employing an encoding technique derived from [25]. The proposed solution is to use a suitable encoding function, which splits the signature into two components. The first component can be injected into the codomain of the trapdoor function and subsequently made part of the computation of the aggregate signature, similar to the approach used in [24]. The second component is transmitted to the next signer and becomes part of the final aggregate signature. During the verification phase, this component is used to recover the partial aggregate signature through a corresponding decoding function. Subsequently, this idea was applied to MQ-based schemes instantiated with HFEv- [17] and UOV [12].

In the following, we define a slight modification of HF-SAS, as formalized in [9]. In this variant, the aggregation step requires only partial knowledge about the so-far aggregated signature. This description better captures the intuition behind the use of the encoding function and is better suited to our proposed scheme. During each aggregation step, the signer produces a partial signature information, which will be sent to the next signer, along with a complementary component. At the end of the aggregation sequence, an additional $\mathsf{Combine}$ step is performed, potentially by a third party. This step combines all the complementary information and the last signature of the sequence, resulting in the complete aggregated signature.

Definition 7 (PS-HF-SAS). *A Partial-Signature History-Free Sequential Aggregate Signature is a tuple of four algorithms* $(\mathsf{KGen}, \mathsf{AggSign}, \mathsf{AggVrfy}, \mathsf{Combine})$*:*

- KGen *and* $\mathsf{AggVrfy}$ *as described in Definition 6.*
- $\mathsf{AggSign}(\mathsf{sk}_i, m_i, \mathsf{Part}(\Sigma_{i-1}))$*: takes as input the secret key* sk_i *and the message* m_i *of the ith user and a partial description* $\mathsf{Part}(\Sigma_{i-1})$ *of the previous aggregate signature* Σ_{i-1}*. Computes an updated aggregate signature* Σ_i *and returns a partial description* $\mathsf{Part}(\Sigma_i)$ *and some complementary information* $\mathsf{Comp}(\Sigma_i)$*.*
- $\mathsf{Combine}(\mathsf{Comp}(\Sigma_1), \ldots, \mathsf{Comp}(\Sigma_n), \mathsf{Part}(\Sigma_n))$*: takes as input the complementary information* $\mathsf{Comp}(\Sigma_i)$ *of the first* $n-1$ *signatures and the full description of the last signature* Σ_n*. Returns the complete description of the aggregate signature* $\bar{\Sigma}_n$*.*

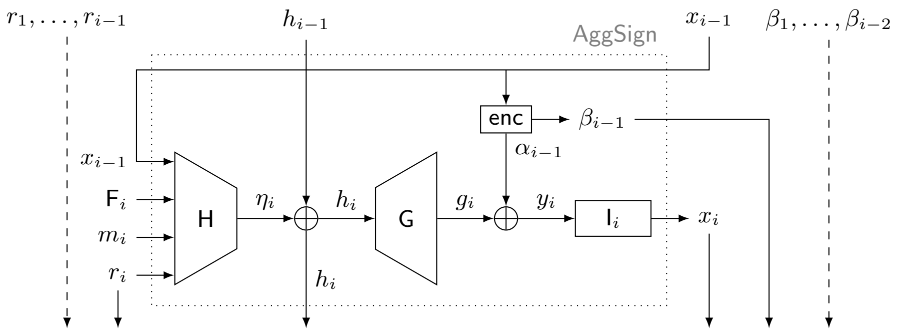

Fig. 2. High level description of our HaS-HF-SAS scheme

Below we show the definition of partial-signature history-free unforgeability under adaptive chosen message (PS-HF-UF-CMA). In this model, the forger controls all signers' private keys except for at least one honest signer. The forger can choose the keys of the rogue signers and adaptively query an aggregate signature oracle. Finally, to win the experiment, the forger must produce a valid, non-trivial aggregate signature involving the public key of the honest signer.

Definition 8 (PS-HF-UF-CMA **Security).** *Let* O *be a random oracle, let* $\mathsf{S} = (\mathsf{KGen}, \mathsf{OAggSign}, \mathsf{OAggVrfy}, \mathsf{OCombine})$ *be a* PS-HF-SAS *scheme, let* $\mathcal{A}$ *be an adversary. We define the advantage of* $\mathcal{A}$ *playing the* PS-HF-UF-CMA *game (Game 2) against* S *as follows:*

$$\mathsf{Adv}_{\mathsf{S}}^{\text{PS-HF-UF-CMA}}(\mathcal{A}) = \Pr[\text{PS-HF-UF-CMA}_{\mathsf{S}}(\mathcal{A}) = 1].$$

3 Sequential Aggregation of Generic Trapdoor Signatures

We present a history-free sequential aggregate signature HaS-HF-SAS from generic trapdoor functions. The scheme is obtained by combining the construction of [10] and the probabilistic hash-and-sign with retry approach of Algorithm 1 with the encoding technique from [25].

A high-level description of the scheme is shown in Figure 2 while a detailed description is given in Algorithm 2. The scheme uses a generic trapdoor function as described in Section 2.1.

3.1 Security Proof

In the following, we prove the PS-HF-UF-CMA security of Algorithm 2.

Theorem 1 (PS-HF-UF-CMA **Security).** *Let* T *be a multivariate trapdoor function. Let* $\mathcal{A}$ *be a* PS-HF-UF-CMA *adversary against the* HaS-HF-SAS *scheme on* T *in the random oracle model, which runs in time* t *and makes* $\mathsf{q_S}$ *signing queries,* $\mathsf{q_H}$ *queries to the random oracle* H *and* $\mathsf{q_G}$ *queries to the random*

Algorithm 2: $\mathsf{HaS\text{-}HF\text{-}SAS}$

Let $h_0 = \varepsilon, x_0 = \varepsilon$. The random oracles are $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$ and $\mathsf{G}\colon \{0,1\}^{2\lambda} \to \mathcal{Y}$. The encoding function is $\mathsf{enc}\colon \mathcal{X} \to \mathcal{Y} \times \mathcal{X}'$ and the corresponding decoding function is $\mathsf{dec}\colon \mathcal{Y} \times \mathcal{X}' \to \mathcal{X}$ such that $\mathsf{dec}(\mathsf{enc}(x)) = x$.

```
$\mathsf{KGen}(1^\lambda)$:
 1: $(\mathsf{F}, \mathsf{I}) \leftarrow \mathsf{TrapGen}(1^\lambda)$
 2: **return** $\mathsf{pk} \leftarrow \mathsf{F}, \mathsf{sk} \leftarrow (\mathsf{F}, \mathsf{I})$

$\mathsf{AggSign}((\mathsf{F}_i, \mathsf{I}_i),\ m_i,\ \mathsf{Part}(\Sigma_{i-1}))$:
 1: $(h_{i-1}, x_{i-1}) \leftarrow \mathsf{Part}(\Sigma_{i-1})$
 2: $(\alpha_{i-1}, \beta_{i-1}) \leftarrow \mathsf{enc}(x_{i-1})$
 3: **repeat**
 4:     $r_i \leftarrow_\$ \{0,1\}^\lambda$
 5:     $\eta_i \leftarrow \mathsf{H}(\mathsf{F}_i, m_i, r_i, x_{i-1})$
 6:     $h_i \leftarrow h_{i-1} \oplus \eta_i$
 7:     $g_i \leftarrow \mathsf{G}(h_i)$
 8:     $y_i \leftarrow g_i \oplus \alpha_{i-1}$
 9:     $x_i \leftarrow \mathsf{I}_i(y_i)$
10: **until** $x_i \neq \perp$
11: $\mathsf{Part}(\Sigma_i) \leftarrow (h_i, x_i)$
12: $\mathsf{Comp}(\Sigma_i) \leftarrow (r_i, \beta_{i-1})$
13: **return** $\mathsf{Part}(\Sigma_i), \mathsf{Comp}(\Sigma_i)$

$\mathsf{AggVrfy}(L_n,\ \bar{\Sigma}_n)$:
 1: $(\mathsf{F}_1, m_1), \ldots, (\mathsf{F}_n, m_n) \leftarrow L_n$
 2: $(\vec{r}_n, \vec{\beta}_{n-1}, h_n, x_n) \leftarrow \bar{\Sigma}_n$
 3: **for** $i \leftarrow n, \ldots 2$ **do**
 4:     $y_i \leftarrow \mathsf{F}_i(x_i)$
 5:     $g_i \leftarrow \mathsf{G}(h_i)$
 6:     $\alpha_{i-1} \leftarrow g_i \oplus y_i$
 7:     $x_{i-1} \leftarrow \mathsf{dec}(\alpha_{i-1}, \beta_{i-1})$
 8:     $\eta_i \leftarrow \mathsf{H}(\mathsf{F}_i, m_i, r_i, x_{i-1})$
 9:     $h_{i-1} \leftarrow h_i \oplus \eta_i$
10: **return** $h_1 = \mathsf{H}(\mathsf{F}_1, r_1, m_1, \varepsilon) \wedge$
      $\mathsf{F}_1(x_1) = \mathsf{G}(h_1)$

$\mathsf{Combine}(\mathsf{Comp}(\Sigma_1), \ldots, \mathsf{Comp}(\Sigma_{n-1}),$
    $\Sigma_n)$:
 1: $(r_i, \beta_{i-1}) \leftarrow \mathsf{Comp}(\Sigma_i)$
 2: $(r_n, \beta_{n-1}, h_n, x_n) \leftarrow \mathsf{Part}(\Sigma_n)$
 3: $\vec{r}_n \leftarrow (r_1, \ldots, r_n)$
 4: $\vec{\beta}_{n-1} \leftarrow (\beta_1, \ldots, \beta_{n-1})$
 5: **return** $\bar{\Sigma}_n \leftarrow (\vec{r}_n, \vec{\beta}_{n-1}, h_n, x_n)$
```

oracle G. *Then, there exist a* OW *adversary* $\mathcal{B}$ *against* T *that runs in time* $t + \mathcal{O}((\mathsf{q_H} + \mathsf{q_S} + 1) \cdot \mathsf{poly}(\mathrm{len}(\mathcal{X}), \mathrm{len}(\mathcal{Y})))$*, and a* PS *adversary* $\mathcal{D}$ *against* T *issuing* $\mathsf{q_S}$ *sampling queries that runs in time* $t + \mathcal{O}(\mathsf{q_S} \cdot \mathsf{poly}(\mathrm{len}(\mathcal{X}), \mathrm{len}(\mathcal{Y})))$*, such that*

$$\mathsf{Adv}^{\text{PS-HF-UF-CMA}}_{\mathsf{HaS\text{-}HF\text{-}SAS}}(\mathcal{A}) \leq (\tau \mathsf{q_H}) \cdot \mathsf{Adv}^{\text{OW}}_{\mathsf{T}}(\mathcal{B}) + \mathsf{Adv}^{\text{PS}}_{\mathsf{T}}(\mathcal{D}) + \frac{(\mathsf{q_S} + \mathsf{q_H})(\mathsf{q_S} + \mathsf{q_H} + \mathsf{q_G})}{2^{2\lambda}}$$
$$+ \frac{\mathsf{q_S}(\mathsf{q_S} + \mathsf{q_H})}{2^\lambda} + \frac{\tau \mathsf{q_H^2}}{2|\mathcal{Y}|} + \frac{(\tau \mathsf{q_H})^{\tau+1} |\mathcal{X}|}{(\tau + 1)! \cdot |\mathcal{Y}|^{\tau+1}},$$

where $\tau \geq \lceil \mathrm{len}(\mathcal{X}) / \mathrm{len}(\mathcal{Y}) \rceil$.

Proof. We prove the reduction by showing that the PS-HF-UF-CMA game can be simulated by the OW adversary $\mathcal{B}$. The high level idea is to modify the PS-HF-UF-CMA game such that in $\mathsf{OAggSign}$ the salt r is chosen uniformly at random in $\{0,1\}^\lambda$ and the preimage is generated by $x \leftarrow \mathsf{SampDom}(\mathsf{F}^\star)$ instead of iterating until $\mathsf{I}^\star(y) \neq \perp$. The PS adversary $\mathcal{D}$ can simulate the two games by either playing PS_0 or PS_1 and the advantage in distinguishing the two games can therefore be estimated with $\mathsf{Adv}^{\text{PS}}_{\mathsf{T}}(\mathcal{D})$. Once the preimages are produced by $x \leftarrow \mathsf{SampDom}(\mathsf{F}^\star)$ without retry, we can apply the techniques adapted from

[10] to complete the reduction. In particular, we will use a labeled tree HTree whose nodes will be populated by some of the queries to the random oracle H. The HTree is initialized with a root node with a single value $h_0 = \varepsilon$. Each subsequent node N_i is added following a query to the random oracle H with input $Q_i = (\mathsf{F}_i, m_i, r_i, x_{i-1})$ and will store the following values:

– a reference to its parent node N_{i-1};
– the query Q_i to the random oracle H;
– the hash response to the query $\eta_i \leftarrow \mathsf{H}(Q)$;
– the hash state $h_i \leftarrow h_{i-1} \oplus \eta_i$, where h_{i-1} is the hash state stored in the parent node N_{i-1};
– an additional value $y_i \leftarrow \mathsf{G}(h_i) \oplus \alpha_{i-1}$ (where α_{i-1} is computed from $\mathsf{enc}(x_{i-1})$) that will be used to establish if future nodes can be added as children of N_i.

A node N_i can be added as child of a node N_{i-1} if it satisfies the relation $\mathsf{F}_{i-1}(x_{i-1}) = y_{i-1}$, where F_{i-1} and y_{i-1} are stored in N_{i-1} while x_{i-1} is stored in N_i. This relationship establishes that the query Q_i can be properly used by the signer with key F_i to aggregate its signature on message m_i with previous signature x_{i-1}, produced by key F_{i-1} and hash state h_{i-1}, which are stored in N_{i-1}. Whenever a query $Q_i = (\mathsf{F}_i, m_i, r_i, x_{i-1})$, with $x_{i-1} \neq \varepsilon$ satisfies this relation with a node N_{i-1} of the HTree we say that Q_i can be *tethered* to N_{i-1}. If $x_{i-1} = \varepsilon$, then Q_i can always be tethered to the root of the HTree.

We now prove the reduction by presenting a sequence of hybrid games, modifying the PS-HF-UF-CMA game (Game 2) until it can be simulated by the OW adversary $\mathcal{B}$. In the following use the notation $\Pr[\mathsf{Game_n}(\mathcal{A}) = 1]$ to denote the probability that $\mathcal{A}$ returns 1 by playing $\mathsf{Game_n}$. The game sequence $\mathsf{Game_0}$-$\mathsf{Game_3}$ for $\mathsf{OAggSign}$ is detailed in Game 3. The game sequence $\mathsf{Game_3}$-$\mathsf{Game_5}$ for H is detailed in Game 4.

$\mathsf{Game_0}$ This is the original PS-HF-UF-CMA game against the $\mathsf{HaS\text{-}HF\text{-}SAS}$ scheme except that it uses programmable random oracles. At the start of the game, the challenger initializes two tables, HT for H and GT for G. When a query Q for H is received, if $\mathsf{HT}[Q] = \perp$ it uniformly samples $\eta \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$ and stores $\mathsf{HT}[Q] \leftarrow \eta$, finally it returns $\mathsf{HT}[Q]$ (similarly for G). It follows that $\Pr[\mathsf{Game_0}(\mathcal{A}) = 1] = \mathsf{Adv}^{\text{PS-HF-UF-CMA}}_{\mathsf{HaS\text{-}HF\text{-}SAS}}(\mathcal{A})$.

$\mathsf{Game_1}$ This game is identical to $\mathsf{Game_0}$ except that $\mathsf{OAggSign}$ aborts by raising $\mathsf{bad_{hcol}}$ if on query $(m, \mathsf{Part}(\Sigma) = (h, x))$ it samples a salt r such that the random oracle H was already queried at input $Q = (\mathsf{F}^\star, m, r, x)$, i.e. $\mathsf{HT}[Q] \neq \perp$. Otherwise it samples $\eta \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$ and programs $\mathsf{HT}[Q] \leftarrow \eta$. It follows that $|\Pr[\mathsf{Game_0}(\mathcal{A}) = 1] - \Pr[\mathsf{Game_1}(\mathcal{A}) = 1]| \leq \Pr[\mathsf{bad_{hcol}}]$.

$\mathsf{Game_2}$ This game is identical to $\mathsf{Game_1}$ except that $\mathsf{OAggSign}$ aborts by raising $\mathsf{bad_{gcol1}}$ if on query $(m, \mathsf{Part}(\Sigma) = (h, x))$, after sampling $\eta \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$ it computes $h' \leftarrow h \oplus \eta$ such that the random oracle G was already queried at input h', i.e. $\mathsf{GT}[h'] \neq \perp$. Otherwise it samples $y' \leftarrow\!\!\$\ \mathcal{Y}$ and programs $\mathsf{GT}[h'] \leftarrow y' \oplus \alpha$. It follows that $|\Pr[\mathsf{Game_1}(\mathcal{A}) = 1] - \Pr[\mathsf{Game_2}(\mathcal{A}) = 1]| \leq \Pr[\mathsf{bad_{gcol1}}]$.

Game 3: Games for $\mathsf{OAggSign}(m, \mathsf{Part}(\Sigma) = (h, x))$

Game_0:
```
1:  $(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
2:  repeat
3:      $r \leftarrow_\$ \{0,1\}^\lambda$
4:      $\eta \leftarrow \mathsf{H}(\mathsf{F}^\star, m, r, x)$
5:      $h' \leftarrow h \oplus \eta$
6:      $g' \leftarrow \mathsf{G}(h')$
7:      $y' \leftarrow g' \oplus \alpha$
8:      $x' \leftarrow \mathsf{I}^\star(y')$
9:  until $x' \neq \bot$
10: return $(r, \beta), (h', x')$
```

Game_1-Game_2:
```
1:  $(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
2:  repeat
3:      $r \leftarrow_\$ \{0,1\}^\lambda$
4:      if $\mathsf{HT}[\mathsf{F}^\star, m, r, x] \neq \bot$ then
5:          raise $\mathsf{bad}_{\mathsf{hcol}}$
6:      $\eta \leftarrow_\$ \{0,1\}^{2\lambda}$
7:      $\mathsf{HT}[\mathsf{F}^\star, m, r, x] \leftarrow \eta$
8:      $h' \leftarrow h \oplus \eta$
9:      if $\mathsf{GT}[h'] \neq \bot$ then
10:         raise $\mathsf{bad}_{\mathsf{gcol1}}$
11:     $y' \leftarrow_\$ \mathcal{Y}$
12:     $\mathsf{GT}[h'] \leftarrow y' \oplus \alpha$
13:     $x' \leftarrow \mathsf{I}^\star(y')$
14: until $x' \neq \bot$
15: return $(r, \beta), (h', x')$
```

Game_3-Game_5:
```
1:  $(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
2:  $r \leftarrow_\$ \{0,1\}^\lambda$
3:  if $\mathsf{HT}[\mathsf{F}^\star, m, r, x] \neq \bot$ then
4:      raise $\mathsf{bad}_{\mathsf{hcol}}$
5:  $\eta \leftarrow_\$ \{0,1\}^{2\lambda}$
6:  $\mathsf{HT}[\mathsf{F}^\star, m, r, x] \leftarrow \eta$
7:  $h' \leftarrow h \oplus \eta$
8:  if $\mathsf{GT}[h'] \neq \bot$ then
9:      raise $\mathsf{bad}_{\mathsf{gcol1}}$
10: $x' \leftarrow \mathsf{SampDom}(\mathsf{F}^\star)$
11: $y' \leftarrow \mathsf{F}^\star(x')$
12: $\mathsf{GT}[h'] \leftarrow y' \oplus \alpha$
13: return $(r, \beta), (h', x')$
```

Game_3 This game is identical to Game_2 except that $\mathsf{OAggSign}$ directly samples $r \leftarrow_\$ \{0,1\}^\lambda$, $x' \leftarrow \mathsf{SampDom}(\mathsf{F}^\star)$ and computes $y' \leftarrow \mathsf{F}^\star(x')$ instead of computing $x' \leftarrow \mathsf{I}^\star(y')$ after sampling $y' \leftarrow_\$ \mathcal{Y}$. The PS adversary $\mathcal{D}$ can simulate both Game_2 and Game_3, noticing that $y' = \mathsf{F}^\star(x')$ and programming G accordingly. More precisely, on receiving a query $Q = (m, \mathsf{Part}(\Sigma) = (h, x))$ for $\mathsf{OAggSign}$, $\mathcal{D}$ computes $(r, x') \leftarrow \mathsf{Sample}_b$ and programs $\mathsf{GT}[h'] \leftarrow \mathsf{F}^\star(x') \oplus \alpha$. Both Game_2 and Game_3 are equivalently modified by moving the programming step of H and G to the end of the $\mathsf{OAggSign}$. It now follows that when $\mathcal{D}$ is playing PS_0 its simulation coincides with Game_2, while when it is playing PS_1 it coincides with Game_3. Either way, $\mathcal{D}$ simulates the games with at most the same running time of $\mathcal{A}$ plus the time required for answering the queries to the sampling oracle. The latter takes $\mathcal{O}(\mathsf{poly}(\mathrm{len}(\mathcal{X}), \mathrm{len}(\mathcal{Y})))$ and is repeated at most $\mathsf{q_S}$ times. Finally, we have that $|\Pr[\mathsf{Game}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Game}_3(\mathcal{A}) = 1]| \leq \mathsf{Adv}^{\mathrm{PS}}_{\mathsf{T}}(\mathcal{D})$.

Game_4 This game is identical to Game_3 except that the random oracle H is simulated as follows. At the start of the game, the challenger initializes a labeled tree HTree, as described at the beginning of the proof. When H receives a query $Q = (\mathsf{F}, m, r, x)$, if $\mathsf{HT}[Q] \neq \bot$ it returns it. Otherwise, it samples a uniformly random $\eta \leftarrow_\$ \{0,1\}^{2\lambda}$ and programs $\mathsf{HT}[Q] \leftarrow \eta$. Then, it checks if Q can be added as a child node of existing nodes in HTree. To determine whether this is the case, it uses the Lookup function (see Algorithm 3) on input x that checks if it can be tethered to existing nodes, i.e. there exists a node $N_i \in \mathsf{HTree}$ such that $\mathsf{F}_i(x) = y_i$. If Q can be tethered to more than τ nodes, the game aborts by raising $\mathsf{bad}_{\mathsf{tcol}}$. Otherwise, H add a new node N_i' with parent N_i for each node $N_i \in \mathsf{HTree}$ returned by $\mathsf{Lookup}(x)$. N_i' contains the original query Q, the hash response η, the hash state $h_i' \leftarrow h_i \oplus \eta$ (where h_i is stored in N_i) and an additional value $y_i' \leftarrow \mathsf{G}(h_i') \oplus \alpha$ (where α is computed from $\mathsf{enc}(x)$) that will be used to check if a future node can be tethered via Lookup queries. It holds that $|\Pr[\mathsf{Game}_3(\mathcal{A}) = 1] - \Pr[\mathsf{Game}_4(\mathcal{A}) = 1]| \leq \Pr[\mathsf{bad}_{\mathsf{tcol}}]$.

Game_5 This game is identical to Game_4 except that the random oracle H is simulated as follows. At the beginning of the game, the challenger uniformly chooses an index $c^\star \leftarrow\$ [\mathsf{q_H}]$ among the queries to the random oracle H, initializes a counter $c \leftarrow 0$ and uniformly samples $y^\star \leftarrow\$ \mathcal{Y}$. When H receives a query $Q = (\mathsf{F}, m, r, x)$ it increments $c \leftarrow c + 1$. Then, if $\mathsf{F} = \mathsf{F}^\star$ and $c = c^\star$ it samples a random index $i^\star$ from the number of nodes in NList. If, for any of the new nodes to be added, it computes $h_i' \leftarrow h_i \oplus \eta$ such that the random oracle G was already queried at input h_i', i.e. $\mathsf{GT}[h_i'] \neq \perp$, it aborts by raising $\mathsf{bad}_{\mathsf{gcol2}}$. Otherwise, if $\mathsf{F} = \mathsf{F}^\star$, $c = c^\star$ and $i = i^\star$, it sets $y_i' \leftarrow y^\star$ and programs $\mathsf{GT}[h_i'] \leftarrow y_i' \oplus \alpha$. It holds that $|\Pr[\mathsf{Game}_4(\mathcal{A}) = 1] - \Pr[\mathsf{Game}_5(\mathcal{A}) = 1]| \leq \Pr[\mathsf{bad}_{\mathsf{gcol2}}]$.

We now show that the OW adversary $\mathcal{B}$ can simulate Game_5 as described in Algorithm 3. After the adversary $\mathcal{A}$ outputs a valid aggregate signature $\bar{\Sigma}_n$ for the history $L_n = (\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n)$ the simulator takes $i^\star \in [n]$ such that $\mathsf{pk}_{i^\star} = \mathsf{pk}^\star$ and $m_{i^\star} \notin \mathcal{Q}$ (the index $i^\star$ is guaranteed to exist when $\mathcal{A}$ is winning Game_5). It then recovers $x_{i^\star}$ by iterating the procedure of Lines 3 to 9 in $\mathsf{AggVrfy}$ for $n - i^\star$ steps. Then, the simulator checks if $x_{i^\star}$ is a preimage of a $y_{i^\star}$ in the HTree as a child of the node $N_{i^\star-1}$ storing $Q_{i^\star-1} = (\mathsf{pk}_{i^\star-1}, m_{i^\star-1}, r_{i^\star-1}, x_{i^\star-2})$, which is itself a child of the node $N_{i^\star-2}$, and so on until the node N_1. If this is not the case, the simulator aborts by raising $\mathsf{bad}_{\mathsf{teth}}$. Otherwise, the value $x_{i^\star}$ produced by the forgery will satisfy $\mathsf{F}^\star(x_{i^\star}) = y_{i^\star}$ for some $y_{i^\star}$ produced either on Line 19 or on Line 21 of H and stored in $N_{i^\star}$. With probability $1/(\tau \mathsf{q_H})$, we have that $y_{i^\star}$ was produced on Line 21 of H and it is equal to $y^\star$. Therefore $\mathsf{F}^\star(x_{i^\star}) = y^\star$ and $\mathcal{B}$ wins his OW game by returning $x_{i^\star}$. Moreover, if none of the bad events happen, $\mathcal{B}$ perfectly simulate Game_5 and we have that

$$\begin{aligned}\mathsf{Adv}_{\mathsf{T}}^{\mathrm{OW}}(\mathcal{B}) &= \frac{1}{\tau \mathsf{q_H}} \Pr[\mathsf{Game}_5(\mathcal{A}) = 1] \\ &\geq \frac{1}{\tau \mathsf{q_H}} (\mathsf{Adv}_{\mathsf{HaS\text{-}HF\text{-}SAS}}^{\mathrm{PS\text{-}HF\text{-}UF\text{-}CMA}}(\mathcal{A}) - \Pr[\mathsf{bad}_{\mathsf{hcol}}] - \Pr[\mathsf{bad}_{\mathsf{gcol1}}] \\ &\qquad - \mathsf{Adv}_{\mathsf{T}}^{\mathrm{PS}}(\mathcal{D}) - \Pr[\mathsf{bad}_{\mathsf{tcol}}] - \Pr[\mathsf{bad}_{\mathsf{gcol2}}] - \Pr[\mathsf{bad}_{\mathsf{teth}}]).\end{aligned}$$

$\mathcal{B}$ can simulate Game_5 with at most the same running time of $\mathcal{A}$ plus the time required for running $\mathsf{AggVrfy}$ and answering the queries to the random oracles H, G, and to the signing oracle $\mathsf{OAggSign}$. These operations takes $\mathcal{O}(\mathsf{poly}(\mathrm{len}(\mathcal{X}), \mathrm{len}(\mathcal{Y})))$ and are repeated at most $\mathsf{q_H} + \mathsf{q_S} + 1$ times.

In the following we bound the probability of each bad event happening.

Probability of $\mathsf{bad}_{\mathsf{hcol}}$ The event $\mathsf{bad}_{\mathsf{hcol}}$ occurs on Line 5 of $\mathsf{OAggSign}$ on input $(m, \mathsf{Part}(\Sigma) = (h, x))$ when it samples $r \leftarrow\$ \{0,1\}^\lambda$ such that a value for

Game 4: Games for $\mathsf{H}(\mathsf{F}, m, r, x)$

```
Game_0-Game_3:
1: $Q \leftarrow (\mathsf{F}, m, r, x)$
2: if $\mathsf{HT}[Q] = \bot$ then
3:     $\eta \leftarrow_\$ \{0,1\}^{2\lambda}$
4:     $\mathsf{HT}[Q] \leftarrow \eta$
5: return $\mathsf{HT}[Q]$

Game_4:
1: $Q \leftarrow (\mathsf{F}, m, r, x)$
2: if $\mathsf{HT}[Q] = \bot$ then
3:     $\eta \leftarrow_\$ \{0,1\}^{2\lambda}$
4:     $\mathsf{HT}[Q] \leftarrow \eta$
5:     $\mathsf{NList} \leftarrow \mathsf{Lookup}(x)$
6:     for $i \in [|\mathsf{NList}|]$ do
7:         $N_i \leftarrow \mathsf{NList}[i]$
8:         $N'_i \leftarrow$ new node with parent $N_i$
9:         Retrieve $h_i$ from $N_i$
10:        $h'_i \leftarrow h_i \oplus \eta$
11:        $(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
12:        $g'_i \leftarrow \mathsf{G}(h'_i)$
13:        $y'_i \leftarrow g'_i \oplus \alpha$
14:        $N'_i \leftarrow (Q, \eta, h'_i, y'_i)$
15: return $\mathsf{HT}[Q]$

Game_5:
1: $Q \leftarrow (\mathsf{F}, m, r, x)$
2: $c \leftarrow c + 1$
3: if $\mathsf{HT}[Q] = \bot$ then
4:     $\eta \leftarrow_\$ \{0,1\}^{2\lambda}$
5:     $\mathsf{HT}[Q] \leftarrow \eta$
6:     $\mathsf{NList} \leftarrow \mathsf{Lookup}(x)$
7:     if $\mathsf{F} = \mathsf{F}^\star \wedge c = c^\star$ then
8:         $i^\star \leftarrow_\$ [|\mathsf{NList}|]$
9:     for $i \in [|\mathsf{NList}|]$ do
10:        $N_i \leftarrow \mathsf{NList}[i]$
11:        $N'_i \leftarrow$ new node with parent $N_i$
12:        Retrieve $h_i$ from $N_i$
13:        $h'_i \leftarrow h_i \oplus \eta$
14:        $(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
15:        if $\mathsf{GT}[h'_i] \neq \bot$ then
16:            raise $\mathsf{bad}_{\mathsf{gcol2}}$
17:        if $\mathsf{F} \neq \mathsf{F}^\star \vee c \neq c^\star \vee i \neq i^\star$ then
18:            $g'_i \leftarrow \mathsf{G}(h'_i)$
19:            $y'_i \leftarrow g'_i \oplus \alpha$
20:        else
21:            $y'_i \leftarrow y^\star$
22:            $\mathsf{GT}[h'_i] \leftarrow y'_i \oplus \alpha$
23:        $N'_i \leftarrow (Q, \eta, h'_i, y'_i)$
24: return $\mathsf{HT}[Q]$
```

$Q = (\mathsf{F}^\star, m, r, x)$ was already assigned in the HT. The table HT is populated by either $\mathsf{OAggSign}$ or H, so its entries are at most $\mathsf{q_S} + \mathsf{q_H}$. The probability that a uniformly random r produces a collision with one of the entries is then at most $(\mathsf{q_S} + \mathsf{q_H})2^{-\lambda}$. Since at most $\mathsf{q_S}$ are made to $\mathsf{OAggSign}$, then $\Pr[\mathsf{bad_{hcol}}] \leq \mathsf{q_S}(\mathsf{q_S} + \mathsf{q_H})2^{-\lambda}$.

Probability of $\mathsf{bad_{gcol1}}$ The event $\mathsf{bad_{gcol1}}$ occurs on Line 10 of $\mathsf{OAggSign}$ on input $(m, \mathsf{Part}(\Sigma) = (h, x))$ when, after sampling $\eta \leftarrow_\$ \{0,1\}^{2\lambda}$, it computes $h' \leftarrow h \oplus \eta$ such that a value for h' was already assigned in the GT. The table GT is populated by either $\mathsf{OAggSign}$, H or G so its entries are at most $\mathsf{q_S} + \mathsf{q_H} + \mathsf{q_G}$. The probability that a uniformly random η produces a collision with one of the entries is then at most $(\mathsf{q_S} + \mathsf{q_H} + \mathsf{q_G})2^{-2\lambda}$. Since at most $\mathsf{q_S}$ are made to $\mathsf{OAggSign}$, then $\Pr[\mathsf{bad_{gcol1}}] \leq \mathsf{q_S}(\mathsf{q_S} + \mathsf{q_H} + \mathsf{q_G})2^{-2\lambda}$.

Probability of $\mathsf{bad_{tcol}}$ The event $\mathsf{bad_{tcol}}$ occurs on Line 5 of Lookup on input x when the HTree contains $k > \tau$ nodes $N_1, \ldots, N_k$ such that $\mathsf{F}_i(x) = y_i$ for $i = 1, \ldots, k$, where F_i, y_i are stored in their respective nodes N_i. The HTree is populated by the simulation of the random oracle H. There are at most

Algorithm 3: OW $\implies$ PS-HF-UF-CMA

```
$\mathcal{B}(\mathsf{F}^\star, y^\star)$:
 1: $\mathcal{Q} \leftarrow \emptyset; c^\star \leftarrow\!\$\, [\mathsf{q_H}]; c \leftarrow 0$
 2: $(L_n, \bar{\Sigma}_n) \leftarrow \mathcal{A}^{\mathsf{H},\mathsf{G},\mathsf{OAggSign}}(\mathsf{F}^\star)$
 3: $(\mathsf{F}_1, m_1), \ldots, (\mathsf{F}_n, m_n) \leftarrow L_n$
 4: if $\mathsf{AggVrfy}(L_n, \Sigma_n) \wedge \exists i^\star : (\mathsf{F}_{i^\star} = \mathsf{F}^\star \wedge m_{i^\star} \notin \mathcal{Q})$ then
 5:    Recover $x_{i^\star}$ as in $\mathsf{AggVrfy}$
 6:    $\mathsf{NList} \leftarrow \mathsf{Lookup}(x_{i^\star})$
 7:    if $\mathsf{NList} = \perp$ then
 8:       raise $\mathsf{bad_{teth}}$
 9:    for $N_{i^\star} \in \mathsf{NList}$ do
10:       Retrieve $y_{i^\star}$ from $N_{i^\star}$
11:       if $y_{i^\star} = y^\star$ then
12:          return $x_{i^\star}$
13:    raise $\mathsf{bad_{inv}}$

$\mathsf{OAggSign}(m, \mathsf{Part}(\Sigma) = (h, x))$:
 1: $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
 2: $(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
 3: $r \leftarrow\!\$\, \{0,1\}^\lambda$
 4: if $\mathsf{HT}[\mathsf{F}^\star, m, r, x] \neq \perp$ then
 5:    raise $\mathsf{bad_{hcol}}$
 6: $\eta \leftarrow\!\$\, \{0,1\}^{2\lambda}$
 7: $\mathsf{HT}[\mathsf{F}^\star, m, r, x] \leftarrow \eta$
 8: $h' \leftarrow h \oplus \eta$
 9: if $\mathsf{GT}[h'] \neq \perp$ then
10:    raise $\mathsf{bad_{gcol1}}$
11: $x' \leftarrow\!\$\, \mathcal{X}$
12: $y' \leftarrow \mathsf{F}^\star(x')$
13: $\mathsf{GT}[h'] \leftarrow y' \oplus \alpha$
14: return $(r, \beta), (h', x')$

$\mathsf{G}(h)$:
 1: if $\mathsf{GT}[h] = \perp$ then
 2:    $g \leftarrow\!\$\, \mathcal{Y}$
 3:    $\mathsf{GT}[h] \leftarrow g$
 4: return $\mathsf{GT}[h]$

$\mathsf{H}(\mathsf{F}, m, r, x)$:
 1: $Q \leftarrow (\mathsf{F}, m, r, x)$
 2: $c \leftarrow c + 1$
 3: if $\mathsf{HT}[Q] = \perp$ then
 4:    $\eta \leftarrow\!\$\, \{0,1\}^{2\lambda}$
 5:    $\mathsf{HT}[Q] \leftarrow \eta$
 6:    $\mathsf{NList} \leftarrow \mathsf{Lookup}(x)$
 7:    if $\mathsf{F} = \mathsf{F}^\star \wedge c = c^\star$ then
 8:       $i^\star \leftarrow\!\$\, [|\mathsf{NList}|]$
 9:    for $i \in [|\mathsf{NList}|]$ do
10:       $N_i \leftarrow \mathsf{NList}[i]$
11:       $N'_i \leftarrow$ new node with parent $N_i$
12:       Retrieve $h_i$ from $N_i$
13:       $h'_i \leftarrow h_i \oplus \eta$
14:       $(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
15:       if $\mathsf{GT}[h'_i] \neq \perp$ then
16:          raise $\mathsf{bad_{gcol2}}$
17:       if $\mathsf{F} \neq \mathsf{F}^\star \vee c \neq c^\star \vee i \neq i^\star$ then
18:          $g'_i \leftarrow \mathsf{G}(h'_i)$
19:          $y'_i \leftarrow g'_i \oplus \alpha$
20:       else
21:          $y'_i \leftarrow y^\star_i$
22:          $\mathsf{GT}[h'_i] \leftarrow y'_i \oplus \alpha$
23:       Populate node $N'_i$ with $Q, \eta, h'_i, y'_i$
24: return $\mathsf{HT}[Q]$

$\mathsf{Lookup}(x)$:
 1: if $x = \varepsilon$ then
 2:    return Root of $\mathsf{HTree}$
 3: $\mathsf{NList} \leftarrow \{N \in \mathsf{HTree} : (\mathsf{F}, y) \in N \wedge \mathsf{F}(x) = y\}$
 4: if $|\mathsf{NList}| > \tau$ then
 5:    raise $\mathsf{bad_{tcol}}$
 6: else if $|\mathsf{NList}| = 0$ then
 7:    return $\perp$
 8: else
 9:    return $\mathsf{NList}$
```

$\mathsf{q_H}$ queries to H and each query contributes a maximum of τ nodes to the tree. Consequently, the total number of nodes in HTree does not exceed $\tau\mathsf{q_H}$. Therefore, we need to bound the probability that any $(\tau+1)$-tuple of nodes produce a collision on x.

First, observe that when a new node is added to the HTree on Line 23 of H, the value y'_i is chosen uniformly at random from $\mathcal{Y}$ and is independent of the view of $\mathcal{A}$. In fact, whenever the query to H is not the random guess $c^\star$ chosen by the simulator, we have $y'_i \leftarrow \mathsf{G}(h'_i) \oplus \alpha$. Here, $\mathsf{G}(h'_i)$ is guaranteed to be a fresh uniformly random value since, otherwise, H would abort on Line 16 and the node would not be added to the HTree. If, on the other hand, the query $c^\star$ was made to H, then we set $y'_i \leftarrow y^\star$ for one of the new nodes to be added. Since $c^\star$ was chosen randomly among all queries to H, the assignment of $y^\star$ is made independently of the view of $\mathcal{A}$ and previous interactions with H.

To conclude, we prove that for any $(\tau+1)$-tuple (possibly adversarially chosen) of functions $\mathsf{F}_i\colon \mathcal{X} \to \mathcal{Y}$ and uniformly random $y_i \in \mathcal{Y}$, there exists $x \in \mathcal{X}$ such that $\mathsf{F}_i(x) = y_i$, for any $i = 1, \ldots, \tau+1$, with probability at most $|\mathcal{X}|/|\mathcal{Y}|^{\tau+1}$ (Lemma 2). Indeed, the adversary can issue $\tau+1$ queries to H with inputs any functions F_i to be stored in $\tau+1$ nodes N_i in the HTree, but from the previous observation, it would receive $\tau+1$ random, independent values y_i.

Since the number of $(\tau+1)$-tuple of nodes in the HTree are at most $(\tau\mathsf{q_H})^{\tau+1}/(\tau+1)!$, by the union bound, we obtain $\Pr[\mathsf{bad_{tcol}}] \leq (\tau\mathsf{q_H})^{\tau+1}|\mathcal{X}|/((\tau+1)! \cdot |\mathcal{Y}|^{\tau+1})$.

Probability of $\mathsf{bad_{gcol2}}$ The event $\mathsf{bad_{gcol2}}$ occurs on Line 16 of H on input (F, m, r, x) when, after sampling $\eta \leftarrow_\$ \{0,1\}^{2\lambda}$ and retrieving h_{i-1} from the parent node N_{i-1}, it computes $h_i \leftarrow h_{i-1} \oplus \eta$ such that a value for h_i was already assigned in the GT. The same argument from the bound of $\Pr[\mathsf{bad_{gcol}}]$ can be used to prove that $\Pr[\mathsf{bad_{gcol2}}] \leq \mathsf{q_H}(\mathsf{q_S} + \mathsf{q_H} + \mathsf{q_G})2^{-2\lambda}$.

Probability of $\mathsf{bad_{teth}}$ The event $\mathsf{bad_{teth}}$ occurs on Line 8 of the simulation of $\mathcal{B}$ when, after the adversary $\mathcal{A}$ outputs a valid aggregate signature Σ_n for the history $L_n = (\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n)$ the simulator recovers $x_{i^\star}$, with $i^\star \in [n]$ such that $\mathsf{pk}_{i^\star} = \mathsf{pk}^\star$ and $m_{i^\star} \notin \mathcal{Q}$, but $x_{i^\star}$ cannot be tethered to any node in the HTree.

When $\mathsf{bad_{teth}}$ happens, the aggregate signature $\bar{\Sigma}_n$ must be valid on L_n. In particular, the inputs $Q_1 = (\mathsf{F}_1, m_1, r_1, \varepsilon), Q_2 = (\mathsf{F}_2, m_2, r_2, x_1), \ldots, Q_{i^\star} = (\mathsf{F}_{i^\star}, m_{i^\star}, r_{i^\star}, x_{i^\star-1})$ have been queried to H in $\mathsf{OAggVrfy}$. Let $\eta_1, \ldots, \eta_{i^\star}$ be the outputs of these queries, so that $\mathsf{HT}[Q_j] = \eta_j$. Each of these entries has been populated by H since we already verified that $m_{i^\star}$ has not been queried to $\mathsf{OAggSign}$. Each step of $\mathsf{OAggVrfy}$ also recovers a value $h_j \leftarrow h_{j+1} \oplus \eta_j$ which is the input of the G query. Since the aggregate signature is correct, we obtain that $h_1 = \eta_1$. Observe that since Q_1 was queried to H, it must be tethered to the root of HTree and was therefore inserted as a node of HTree with additional values $\eta_1, h_1 = \eta_1, y_1 = \mathsf{G}(h_1)$. Then, since $\mathsf{F}(x_1) = y_1$, the query Q_2 is tethered to N_1. Now we prove that either all $Q_1, \ldots, Q_{i^\star}$ are part of a path of nodes in HTree, or there exists an input Q_j that was queried to H, is tethered to a node in HTree and is not itself in a node of HTree. We proceed by induction on $j \leq i^\star$: we have already shown that Q_1 is in HTree; suppose that Q_j is in the HTree, then, since $\mathsf{F}_j(x_j) = y_j$, the query Q_{j+1} is tethered to Q_j and it may or may not be part of HTree. To conclude, we prove that if an input Q has not been entered in the HTree after being queried to H, the probability that it will ever become tethered to a node in HTree is at most $\tau \mathsf{q}'/|\mathcal{Y}|$, where q' is the number of queries made to H after Q (Lemma 3). Since there are at most $\mathsf{q_H}$ queries that add new nodes to HTree, we obtain, by the union bound, that $\Pr[\mathsf{bad_{teth}}] \leq \tau \mathsf{q}_\mathsf{H}^2/(2|\mathcal{Y}|)$.

Combining the previous bound on bad events, we obtain the claimed estimate of $\mathsf{Adv}^{\text{PS-HF-UF-CMA}}_{\mathsf{HaS\text{-}HF\text{-}SAS}}(\mathcal{A})$.

4 Instantiation and Evaluation

In this section, we will provide some concrete applications of the $\mathsf{HaS\text{-}HF\text{-}SAS}$ of Section 3 to MQ-based and code-based HaS signature schemes. In particular, we analyze the compression capabilities of the scheme when instantiated with UOV, MAYO, and Wave. More details on the trapdoor functions and PS security of these schemes are given in Appendix B.

The main measure of efficiency of an aggregate signature scheme is the *compression rate*, i.e. the reduction in the length of the aggregate signature $\bar{\Sigma}_N$ of N users compared to the trivial concatenation of N individual signatures σ. The compression rate of N signatures is defined as $\tau(N) = 1 - \frac{|\bar{\Sigma}_N|}{N \cdot |\sigma|}$. An $\mathsf{HaS\text{-}HF\text{-}SAS}$ signature of N users is the output of the $\mathsf{Combine}$ algorithm on $\mathsf{Comp}(\Sigma_1), \ldots, \mathsf{Comp}(\Sigma_{N-1}), \Sigma_N$ and is given by $\bar{\Sigma}_N = (\overrightarrow{r}_N, \overrightarrow{\beta}_{N-1}, h_N, x)$. An individual signature of a generic HaS scheme as described in Section 2.3 is given by $\sigma = (r, x)$. In the following, we assume that the aggregation scheme is applied on the signature scheme without further possible optimization, so that we have the same size for salts $|r| = \lambda$ and preimages $|x| = \mathrm{len}(\mathcal{X})$, where $\mathrm{len}(X)$ denotes the bit size of an element in X. The compression rate is thus given by

$$\begin{aligned}\tau(N) &= 1 - \frac{N \cdot \lambda + (N-1) \cdot \mathrm{len}(\mathcal{X}') + 2\lambda + \mathrm{len}(\mathcal{X})}{N \cdot (\lambda + \mathrm{len}(\mathcal{X}))} \\ &= 1 - \frac{N \cdot (\lambda + \mathrm{len}(\mathcal{X}) - \mathrm{len}(\mathcal{Y})) + 2\lambda + \mathrm{len}(\mathcal{Y})}{N \cdot (\lambda + \mathrm{len}(\mathcal{X}))} \\ &= \frac{\mathrm{len}(\mathcal{Y})}{\lambda + \mathrm{len}(\mathcal{X})} - \frac{2\lambda + \mathrm{len}(\mathcal{Y})}{N \cdot (\lambda + \mathrm{len}(\mathcal{X}))}.\end{aligned} \tag{1}$$

Notice that the aggregate signature is smaller than the trivial concatenation whenever $N > \frac{2\lambda}{\mathrm{len}(\mathcal{Y})} + 1$, which for typical parameters is as soon as $N > 2$.

UOV (Appendix B.2) We consider the parameters proposed in [7] with respect to NIST security levels I, III and V. For UOV, the domain $\mathcal{X}$ is given by $\mathbb{F}_q^n$ with elements of length $\text{len}(\mathcal{X}) = n\lceil\log_2 q\rceil$. The codomain $\mathcal{Y}$ is $\mathbb{F}_q^m$ with elements of length $\text{len}(\mathcal{Y}) = m\lceil\log_2 q\rceil$. Regardless of the security level, [7] use 128-bit salts. In our comparison we consider salts of length $\lambda = 128, 192$ and 256 bits, respectively.

The size of a sequential aggregate signature instantiated with UOV is given by $|\bar{\Sigma}_N| = N \cdot (\lambda + (n-m)\lceil\log_2 q\rceil) + 2\lambda + m\lceil\log_2 q\rceil$ and the size of a single signature is given by $|\sigma| = n\lceil\log_2 q\rceil + \lambda$.

Concrete numbers for different security parameters and the number of signers are given in Table 1.

MAYO (Appendix B.3) We consider the parameters proposed in [6] with respect to NIST security levels I, III and V. For MAYO, the domain $\mathcal{X}$ is given by $\mathbb{F}_q^{kn}$ with elements of length $\text{len}(\mathcal{X}) = kn\lceil\log_2 q\rceil$. The codomain $\mathcal{Y}$ is $\mathbb{F}_q^m$ with elements of length $\text{len}(\mathcal{Y}) = m\lceil\log_2 q\rceil$. In [6] the salt length $|r|$ is slightly longer than the security parameter for consistency with the security proof. In our comparison we maintain this choice, adjusting the compression rate computation of Equation (1).

The size of a sequential aggregate signature instantiated with MAYO is given by $|\bar{\Sigma}_N| = N \cdot (|r| + (kn-m)\lceil\log_2 q\rceil) + 2\lambda + m\lceil\log_2 q\rceil$ and the size of a single signature is given by $|\sigma| = kn\lceil\log_2 q\rceil + |r|$.

Concrete numbers for different security parameters and the number of signers are given in Table 2.

Wave (Appendix B.4) We consider the parameters proposed in [14] with respect to NIST security level I. For Wave, the domain $\mathcal{X}$ is given by $S_{w,n}$, the subset of $\mathbb{F}_q^n$ with vectors of hamming weight w, with elements of length $\text{len}(\mathcal{X}) = \lceil n\log_2 q\rceil$. The codomain $\mathcal{Y}$ is $\mathbb{F}_q^{n-k}$ with elements of length $\text{len}(\mathcal{Y}) = \lceil(n-k)\log_2 q\rceil$.

The size of a sequential aggregate signature instantiated with Wave is given by $|\bar{\Sigma}_N| = N \cdot (\lambda + \lceil k\log_2 q\rceil) + 2\lambda + \lceil(n-k)\log_2 q\rceil$ and the size of a single signature is given by $|\sigma| = \lceil n\log_2 q\rceil + \lambda$.

Concrete numbers for different security parameters and the number of signers are given in Table 3.

5 Security of Existing Multivariate SAS Schemes

In this section, we show a universal forgery for the sequential aggregate signature schemes of [17,12] when instantiated with the UOV signature scheme. Both schemes are based on the variant with encoding of [24] and require an alternative definition of SAS with the notion of *full-history*: at each aggregation step, the signer needs the so-far aggregated signature and the complete list of messages and public keys of previous signers. Moreover, knowledge of the full description of the aggregate signature is required, as the signer needs to check its validity before adding its own.

Table 1. Aggregate signature sizes and compression rates of HaS-HF-SAS scheme on UOV parameters from [7].

Parameter	`ov-Ip`	`ov-Is`	`ov-III`	`ov-V`
NIST SL	I	I	III	V
(n, m, q)	(112,44,256)	(160,64,16)	(184,72,256)	(244,96,256)
$N \cdot \lvert\sigma\rvert$ (bytes)	$128 \cdot N$	$96 \cdot N$	$208 \cdot N$	$276 \cdot N$
$\lvert\bar{\Sigma}_N\rvert$ (bytes)	$84 \cdot N + 76$	$64 \cdot N + 64$	$136 \cdot N + 120$	$180 \cdot N + 160$
$\tau(5)$	0.23	0.20	0.23	0.23
$\tau(20)$	0.31	0.30	0.32	0.32
$\tau(100)$	0.34	0.33	0.34	0.34
Asym. $\tau(N)$	0.34	0.33	0.35	0.35

Table 2. Aggregate signature sizes and compression rates of HaS-HF-SAS scheme on MAYO parameters from [6].

Parameter	MAYO_1	MAYO_2	MAYO_3	MAYO_5
NIST SL	I	I	III	V
(n, m, o, k, q)	(66,64,8,9,16)	(78,64,18,4,16)	(99,96,10,11,16)	(133,128,12,12,16)
$\lvert r\rvert$ (bytes)	24	24	32	40
$N \cdot \lvert\sigma\rvert$ (bytes)	$321 \cdot N$	$180 \cdot N$	$577 \cdot N$	$838 \cdot N$
$\lvert\bar{\Sigma}_N\rvert$ (bytes)	$289 \cdot N + 64$	$148 \cdot N + 64$	$529 \cdot N + 96$	$774 \cdot N + 128$
$\tau(5)$	0.06	0.11	0.05	0.05
$\tau(20)$	0.09	0.16	0.07	0.07
$\tau(100)$	0.10	0.17	0.08	0.07
Asym. $\tau(N)$	0.10	0.18	0.08	0.08

Table 3. Aggregate signature sizes and compression rates of HaS-HF-SAS scheme on Wave parameters from [14].

Parameter	`128g`
NIST SL	I
(n, k, w, q)	(8492,5605,7980,3)
$N \cdot \lvert\sigma\rvert$ (bytes)	$1699 \cdot N$
$\lvert\bar{\Sigma}_N\rvert$ (bytes)	$1127 \cdot N + 604$
$\tau(5)$	0.27
$\tau(20)$	0.32
$\tau(100)$	0.33
Asym. $\tau(N)$	0.34

Game 5: FH-UF-CMA$_{\mathsf{S}'}$

```
1: $(\mathsf{pk}^\star, \mathsf{sk}^\star) \leftarrow \mathsf{KGen}(1^\lambda)$
2: $\mathcal{Q} \leftarrow \emptyset$
3: $(L_n, \Sigma_n) \leftarrow \mathcal{A}^{\mathsf{O},\mathsf{OAggSign}}(\mathsf{pk}^\star)$
4: $(\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n) \leftarrow L_n$
5: if $\nexists i^\star : (\mathsf{pk}_{i^\star} = \mathsf{pk}^\star \wedge (m_{i^\star}, L_{i^\star}) \notin \mathcal{Q})$ then
6:     return $\perp$
7: return $\mathsf{AggVrfy}(L_n, \Sigma_n)$
```

```
$\mathsf{OAggSign}(m_i, L_{i-1}, \Sigma_{i-1})$:
1: if $\mathsf{AggVrfy}(L_{i-1}, \Sigma_{i-1}) = 0$ then
2:     return $\perp$
3: $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(L_{i-1}, m_i)\}$
4: $\Sigma_i \leftarrow \mathsf{AggSign}(\mathsf{sk}^\star, m_i, L_{i-1}, \Sigma_{i-1})$
5: return $\Sigma_i$
```

Definition 9 (FH-SAS)**.** *A Full-History Sequential Aggregate Signature is a tuple of three algorithms* $(\mathsf{KGen}, \mathsf{AggSign}, \mathsf{AggVrfy})$*:*

- $\mathsf{KGen}(1^\lambda)$ *as described in Definition 6.*
- $\mathsf{AggSign}(\mathsf{sk}_i, m_i, L_{i-1}, \Sigma_{i-1})$*: takes as input the secret key* sk_i *and the message* m_i *of the ith user, a list* $L_{i-1} = (\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_{i-1}, m_{i-1})$ *of public keys, message pairs, and the previous aggregate signature* Σ_{i-1}*. If* $\mathsf{AggVrfy}(L_{i-1}, \Sigma_{i-1}) = 1$*, it returns an updated aggregate signature* Σ_i*.*
- $\mathsf{AggVrfy}(L_n, \Sigma_n)$*: takes as input the full history* $L_n = (\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n)$ *of public key, message pairs, and an aggregate signature* Σ_n*. Returns 1 if* Σ_n *is a valid aggregate signature and 0 otherwise.*

Below we show the definition of full-history unforgeability under adaptive chosen message (FH-UF-CMA). Compared to the notion of PS-HF-UF-CMA (Definition 8), the signing oracle $\mathsf{OAggSign}$ requires sending the list L_{i-1} of public keys and messages along with the aggregate signature Σ_{i-1} and returns the updated signature if and only if Σ_{i-1} is valid.

Definition 10 (FH-UF-CMA **Security**)**.** *Let* O *be a random oracle, let* $\mathsf{S}' = (\mathsf{KGen}, \mathsf{OAggSign}, \mathsf{OAggVrfy})$ *be a* FH-SAS *scheme, let* $\mathcal{A}$ *be an adversary. We define the advantage of* $\mathcal{A}$ *playing the* FH-UF-CMA *game (Game 5) against* S' *as follows:*

$$\mathsf{Adv}_{\mathsf{S}'}^{\text{FH-UF-CMA}}(\mathcal{A}) = \Pr[\text{FH-UF-CMA}_{\mathsf{S}'}(\mathcal{A}) = 1].$$

5.1 Multivariate FH-SAS

The FH-SAS schemes of [17,12] are instantiated with HFEv- and UOV, respectively, but no explicit use is made of unique features of these trapdoor functions. The description of Algorithm 4 refers to a generic multivariate trapdoor function T (as in Section 2.1) and is based on the construction of [17], which is slightly more general.

In Algorithm 4, the random oracle is $\mathsf{H}\colon \{0,1\}^* \to \mathcal{Y}$. The encoding function is $\mathsf{enc}\colon \mathcal{X} \to \mathcal{Y} \times \mathcal{X}'$ that splits an element x_i as $\mathsf{enc}(x_i) = (\alpha_i, \beta_i)$ and the corresponding decoding function is $\mathsf{dec}\colon \mathcal{Y} \times \mathcal{X}' \to \mathcal{X}$ such that $\mathsf{dec}(\mathsf{enc}(x)) = x$. To simplify the description we will also use the notation $\alpha(x_i) = \alpha_i$ and $\beta(x_i) = \beta_i$, where $\alpha(\cdot), \beta(\cdot)$ are implicitly defined by enc.

Algorithm 4: Multivariate $\mathsf{FH\text{-}SAS_T}$

```
Let $\Sigma_0 = (\emptyset, \varepsilon)$.

$\mathsf{KGen}(1^\lambda)$:
 1: $(\mathsf{F}, \mathsf{I}) \leftarrow \mathsf{TrapGen}(1^\lambda)$
 2: return $\mathsf{pk} \leftarrow \mathsf{F}, \mathsf{sk} \leftarrow (\mathsf{F}, \mathsf{I})$

$\mathsf{AggVrfy}(L_n, \Sigma_n)$:
 1: $(\mathsf{F}_1, m_1), \dots, (\mathsf{F}_n, m_n) \leftarrow L_n$
 2: $(\vec{\beta}_{n-1}, x_n) \leftarrow \Sigma_n$
 3: for $i \leftarrow n, \dots 2$ do
 4:     $L_i \leftarrow (\mathsf{F}_1, m_1), \dots, (\mathsf{F}_i, m_i)$
 5:     $h_i \leftarrow \mathsf{H}(L_i)$
 6:     $\alpha_{i-1} \leftarrow \mathsf{F}_i(x_i) \oplus h_i$
 7:     $x_{i-1} \leftarrow \mathsf{dec}(\alpha_{i-1}, \beta_{i-1})$
 8: return $\mathsf{F}_1(x_1) = \mathsf{H}(\mathsf{F}_1, m_1)$

$\mathsf{AggSign}((\mathsf{F}_i, \mathsf{I}_i), m_i, L_{i-1}, \Sigma_{i-1})$:
 1: $(\vec{\beta}_{i-2}, x_{i-1}) \leftarrow \Sigma_{i-1}$
 2: if $\mathsf{AggVrfy}(L_{i-1}, \Sigma_{i-1}) = 0$ then
 3:     return $\perp$
 4: $L_i \leftarrow L_{i-1} \cup \{(\mathsf{F}_i, m_i)\}$
 5: $(\alpha_{i-1}, \beta_{i-1}) \leftarrow \mathsf{enc}(x_i)$
 6: $h_i \leftarrow \mathsf{H}(L_i)$
 7: $x_i \leftarrow \mathsf{I}_i(\alpha_{i-1} \oplus h_i)$
 8: $\vec{\beta}_{i-1} \leftarrow \vec{\beta}_{i-2} \cup \{\beta_{i-1}\}$
 9: return $\Sigma_i \leftarrow (\vec{\beta}_{i-1}, x_i)$
```

Both [17] and [12] provide a similar claim on the formal security of their sequential aggregate signature scheme. In the following we are considering a generic multivariate trapdoor function, since their choice does not influence the security claim.

Theorem 2 ([17]). *Let* T *be a multivariate trapdoor function. Let* $\mathcal{A}$ *be a* FH-UF-CMA *adversary against the* $\mathsf{FH\text{-}SAS}$ *scheme on* T *in the random oracle model, which makes* $\mathsf{q_S}$ *signing queries and* $\mathsf{q_H}$ *queries to the random oracle. Then, there exist a* OW *adversary* $\mathcal{B}$ *against* T *such that*

$$\mathsf{Adv}^{\text{FH-UF-CMA}}_{\mathsf{FH\text{-}SAS_T}}(\mathcal{A}) \leq (\mathsf{q_S q_H} + 1) \cdot \mathsf{Adv}^{\text{OW}}_{\mathsf{T}}(\mathcal{B})$$

and the running time of $\mathcal{B}$ *is about that of* $\mathcal{A}$.

In [12] the authors omit the proof for their security claim, while in [17] the authors provide a sketch of the proof in which they state that almost all the steps of the security proof follow [24] with only some slight modifications taking into account the use of the encoding function.

In the next section we show an explicit universal forgery on $\mathsf{FH\text{-}SAS}$ when instantiated with the UOV signature scheme. Then, in Section 5.3, we provide some insight into why the security proof of [24] cannot be applied to multivariate schemes, and more generally to signature schemes based on trapdoor functions that are not permutations.

5.2 Description of the Forgery

We recall that in UOV, the trapdoor function is a multivariate quadratic map $\mathcal{P} \colon \mathbb{F}_q^n \to \mathbb{F}_q^m$ that vanishes on a secret linear subspace $O \subset \mathbb{F}_q^n$ of dimension m. A more in-depth description can be found in Appendix B.1.

In the following we are assuming that the encoding function $\mathsf{enc}(\boldsymbol{x})$ can be expressed via an appropriate affine map and, accordingly, $\alpha(\boldsymbol{x}) = R(\boldsymbol{x}) = \mathbf{A}\boldsymbol{x}+\boldsymbol{b}$, where $\mathbf{A} \in \mathbb{F}_q^{m\times n}, \boldsymbol{b} \in \mathbb{F}_q^m$. In [17,12], $\mathsf{enc}(\boldsymbol{x})$ is always the projection in the first m and the last $n-m$ components[3] of $\boldsymbol{x}$. This is a slight generalization that captures the intuition that there must be a corresponding efficient decoding function.

Lemma 1. *The multivariate* FH-SAS *scheme of Section 5.1, instantiated with UOV, is not* FH-UF-CMA.

Proof. Let $\mathsf{pk}_i = \mathcal{P}_i$ be the target public key and assume that the forger $\mathcal{F}$ knows a valid aggregate signature $\Sigma_i = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{i-1}, \boldsymbol{x}_i)$ for a honest history $L_i = (\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_i, m_i)$. This is a typical attack environment, much weaker than the notion of FH-UF-CMA that we introduced in Definition 10. Then, $\mathcal{F}$ select a message $m_i^\star$ on which it will produce a forged signature for the target user.

The forger $\mathcal{F}$ computes a forged signature by replacing the $(i-1)$th honest signer, as follows:

1. First, $\mathcal{F}$ appropriately generates a UOV key pair $(\mathsf{pk}_\mathcal{F}, \mathsf{sk}_\mathcal{F}) = (\mathcal{P}_\mathcal{F}, O_\mathcal{F})$ by randomly choosing an m-dimensional linear subspace $O_\mathcal{F} \subset \ker \mathbf{A}$ and use the same procedure of $\mathsf{TrapGen}_{\mathsf{uov}}$ (Algorithm 5) to sample $\mathcal{P}_\mathcal{F}$ that vanishes on $O_\mathcal{F}$.
2. Then, $\mathcal{F}$ arbitrarily chooses a message $m_\mathcal{F}$, computes a corresponding forged history $L^\star = L_{i-2} \cup \{(\mathsf{pk}_\mathcal{F}, m_\mathcal{F}), (\mathsf{pk}_i, m_i)\}$ and computes $\boldsymbol{\alpha}^\star \leftarrow \mathcal{P}_i(\boldsymbol{x}_i) \oplus \mathsf{H}(L^\star)$.
3. Finally, $\mathcal{F}$ finds a preimage $\boldsymbol{x}_\mathcal{F}$ under $\mathcal{P}_\mathcal{F}$ for $L_\mathcal{F} = L_{i-2} \cup \{(\mathsf{pk}_\mathcal{F}, m_\mathcal{F})\}$ such that
$$\mathcal{P}_\mathcal{F}(\boldsymbol{x}_\mathcal{F}) = \boldsymbol{\alpha}_{i-2} \oplus \mathsf{H}(L_\mathcal{F}) \qquad \text{and} \qquad \alpha(\boldsymbol{x}_\mathcal{F}) = \boldsymbol{\alpha}^\star. \tag{2}$$
Then $\Sigma^\star = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{i-2}, \beta(\boldsymbol{x}_\mathcal{F}), \boldsymbol{x}_i)$ is a valid aggregate signature for the forged history $L^\star$.

Finding a preimage $\boldsymbol{x}_\mathcal{F}$ that satisfies Equation (2) is equivalent to finding partially fixed preimage for $\mathcal{P}_\mathcal{F}$ under the map R. In particular, the forger can use the appropriately generated secret key $O_\mathcal{F}$ to restrict the preimage search to an appropriate affine subspace and guarantee the condition $R(\boldsymbol{x}_\mathcal{F}) = \boldsymbol{\alpha}^\star$. The forger searches for a preimage of $\boldsymbol{t} = \boldsymbol{\alpha}_{i-2} \oplus \mathsf{H}(L_\mathcal{F})$ by using a procedure similar to the Sign procedure described in Algorithm 5: on Line 1, instead of randomly sampling the vectors $\boldsymbol{v}$ from $\mathbb{F}_q^n$, samples $\boldsymbol{v}$ from $\ker R'$, with $R'(\boldsymbol{x}) = \mathbf{A}\boldsymbol{x} + (\boldsymbol{b} - \boldsymbol{\alpha}^\star)$. Then when a preimage $\boldsymbol{x}_\mathcal{F} \in \mathbb{F}_q^n$ of $\boldsymbol{t}$ is found, the forger would have $\boldsymbol{x}_\mathcal{F} \in \ker R'$, since $\boldsymbol{x}_\mathcal{F} = \boldsymbol{v} + \boldsymbol{o}$ with $\boldsymbol{v} \in \ker R'$ and $\boldsymbol{o} \in \ker \mathbf{A}$. Therefore $\alpha(\boldsymbol{x}_\mathcal{F}) = R(\boldsymbol{x}_\mathcal{F}) = \boldsymbol{\alpha}^\star$.

We then show that $\Sigma^\star$ passes the verification correctly for the forged history $L^\star$:

[3] In this case we would have that $\alpha(\boldsymbol{x}) = \mathbf{A}\boldsymbol{x}$ with $\mathbf{A} = [\mathbf{I}_m \mid \mathbf{0}_{m,n-m}]$.

1. The verifier applies the first iteration of $\mathsf{AggVrfy}$ (Algorithm 4) to recover the previous signature $\boldsymbol{x}_{\mathcal{F}}$ from $\boldsymbol{x}_i$ as follows:

$$\alpha(\boldsymbol{x}_{\mathcal{F}}) \leftarrow \mathcal{P}_i(\boldsymbol{x}_i) \oplus \mathsf{H}(L^\star) = \alpha^\star, \qquad \boldsymbol{x}_{\mathcal{F}} \leftarrow \mathsf{dec}(\alpha(\boldsymbol{x}_{\mathcal{F}}), \beta(\boldsymbol{x}_{\mathcal{F}}))$$

2. Since $\boldsymbol{x}_{\mathcal{F}}$ is a preimage of $\boldsymbol{\alpha}_{i-2} \oplus \mathsf{H}(L_{\mathcal{F}})$, the verifier correctly obtains $\boldsymbol{x}_{i-2}$ proceeding in the iterations of $\mathsf{AggVrfy}$:

$$\boldsymbol{\alpha}_{i-2} \leftarrow \mathcal{P}_{\mathcal{F}}(\boldsymbol{x}_{\mathcal{F}}) \oplus \mathsf{H}(L_{\mathcal{F}}), \qquad \boldsymbol{x}_{i-2} \leftarrow \mathsf{dec}(\boldsymbol{\alpha}_{i-2}, \boldsymbol{\beta}_{i-2}).$$

3. The $(i-2)$th signer was not tampered and, hence, the intermediate signature $\Sigma_{i-2} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{i-3}, \boldsymbol{x}_{i-2})$ can be correctly verified with $\mathsf{AggVrfy}$ on honest history L_{i-2}.

Therefore, the verifier determines the forged signature $\Sigma^\star$ as valid.

5.3 Discussion

The previous forging procedure can be directly applied to constructions derived from [24] and instantiated via UOV, such as [17,12]. In particular, we have shown how the existential unforgeability claims of [17,12] are incorrect when the schemes are instantiated with UOV. The attack essentially involves finding a partially fixed preimage, following an adversarial key generation based on the public parameters of the aggregate signature scheme, specifically the encoding function. Although this attack may have applicability beyond UOV, it is not a universal forgery for generic trapdoor functions. However, this result aligns with the analysis of critical issues encountered when attempting to extend the security proof of [24] to generic trapdoor functions, as outlined in the following.

Programming the Random Oracle In [24], the random oracle can be simulated to determine preimages for any permutation $\pi : \mathcal{X} \to \mathcal{X}$. This is typically achieved by sampling a uniformly random $x \leftarrow_\$ \mathcal{X}$ and returning $\pi(x)$, which is uniformly distributed in $\mathcal{Y} = \mathcal{X}$. However, in the case of a generic trapdoor function, we cannot assume that the image of F is uniform. Consequently, to provide an accurate simulation, we must sample and return a value that is uniformly random in $\mathcal{Y}$.

Uniqueness of the Aggregate Signature If we relax the previous condition and assume a uniformity property of the trapdoor functions[4] we may attempt to replicate the process described in [24] to answer the signing oracle. Indeed, on input $Q = (m^\star, L_{n-1}, \Sigma_{n-1})$ the simulator can use the knowledge of appropriate preimages for F_i to craft a valid aggregate signature on $L_{n-1} \cup \{ (\mathsf{F}^\star, m^\star) \}$. However, we argue that this property alone is not sufficient for a correct simulation, which is instead based on the following fact of TDP-based constructions: for a

[4] For instance, this is the case for Trapdoor Preimage Sampleable Function [20].

fixed input $L_n = (\mathsf{F}_1, m_1), \ldots, (\mathsf{F}_n, m_n)$ there exists a unique aggregate signature on L_n. Otherwise, if the aggregate signature is not unique, the simulator would be unable to provide a valid response to the aggregate signature query. In fact, on input Q, the simulator would take the preimage $x^\star$ for $\mathsf{F}^\star$ on $\alpha(x_{n-1}) \oplus \mathsf{H}(L_n)$ associated to the random oracle query on input $L_n = L_{n-1} \cup \{ (\mathsf{F}^\star, m^\star) \}$. But, without the knowledge that x_{n-1} is equal to the preimage computed by the adversary on input L_{n-1}, the aggregate signature produced by the simulator may not be properly verified, resulting in an invalid response.

Reduction to OW Eventually, the adversary will produce a valid non-trivial aggregate signature Σ_n on input $L_n = (\mathsf{F}_1, m_1), \ldots, (\mathsf{F}_n, m_n)$, where we assume, for simplicity, that $\mathsf{F}_n = \mathsf{F}^\star$ is the target public key. Since the aggregate signature is correct, it follows that $\mathsf{F}^\star(x_n) = \alpha(x_{n-1}) \oplus \mathsf{H}(L_n) = y$. In the context of TDPs, [24] shows that y is equal to the target $y^\star$ of the OW game, with probability $(\mathsf{q_H} + \mathsf{q_S} + 1)$.

When we consider generic TDFs, the previously mentioned approach is not valid, and it is necessary to modify the simulation of H by returning a fresh random value for each query. Moreover, we claim that in this setting it is not possible to correctly simulate the response to the oracles in order to obtain a preimage of $y^\star$. In fact, to obtain a valid preimage, the simulator would require $\mathsf{F}^\star(x_n) = \alpha(x_{n-1}) \oplus \mathsf{H}(L_n) = y^\star$ and therefore $\mathsf{H}(L_n) = y^\star \oplus \alpha(x_{n-1})$. It should then have been able to simulate the random oracle to return $y^\star \oplus \alpha(x_{n-1})$ when given the input L_n. However, it is not possible to provide this answer, as x_{n-1} is not part of the query input and is not uniquely determined.

Fixing the Forging Vulnerability The main vulnerability of FH-SAS concerns the overall malleability of the aggregate signature. In the original scheme for TPDs, once the input $L_n = (\mathsf{F}_1, m_1), \ldots, (\mathsf{F}_n, m_n)$ is fixed, it was observed that there is a unique aggregate signature on messages $m_1, \ldots, m_n$ under public keys $\mathsf{F}_1, \ldots, \mathsf{F}_n$. Instead, in the extended version, uniqueness is lost due to the probabilistic nature of the inversion process. Consequently, it is always possible to construct two aggregate signatures on the same input, $\Sigma = (\beta_1, \ldots, \beta_{i-1}, x_n)$ and $\Sigma' = (\beta_1, \ldots, \beta_{i-1}, x'_i)$, which differ only in the aggregation of the last signature. Furthermore, as shown in the forgery presented in Section 5.2, it is possible to have two aggregate signatures on the same input $\Sigma = (\beta_1, \ldots, \beta_{i-1}, x_i)$ and $\Sigma' = (\beta_1, \ldots, \beta'_{i-1}, x_i)$ which differ only in the intermediate partial encodings. While the loss of uniqueness is unavoidable, it is possible to modify the scheme to prevent this additional form of malleability by making partial β encodings part of the random oracle input. We modify the aggregation step of $\mathsf{AggSign}((\mathsf{F}_i, \mathsf{I}_i), m_i, L_{i-1}, \Sigma_{i-1})$ (Algorithm 4): let $\Sigma_{i-1} = (\beta_1, \ldots, \beta_{i-2}, x_{i-1})$ and compute

$$(\alpha_{i-1}, \beta_{i-1}) \leftarrow \mathsf{enc}(x_{i-1}), \qquad x_i \leftarrow \mathsf{I}_i(\alpha_{i-1} \oplus \mathsf{H}(L_i, \overrightarrow{\beta}_{i-1})),$$

where $L_i = L_{i-1} \cup \{ (\mathsf{F}_i, m_i) \}$ and $\overrightarrow{\beta}_{i-1} = (\beta_1, \ldots, \beta_{i-1})$.

Observe that now, once a new signature has been aggregated, it is no longer possible to modify the previous partial encodings while maintaining the validity of the aggregated signature. That is, if $\Sigma = (\beta_1, \ldots, \beta_{i-1}, x_i)$ and $\Sigma' = (\beta_1, \ldots, \beta'_{i-1}, x'_i)$ are valid aggregate signatures on the same input with $\beta_{i-1} \neq \beta'_{i-1}$, then $x_i \neq x'_i$. As a result, the forging procedure described in Section 5.2 is no longer applicable, as the adversary now needs to guess the partial encoding $\beta(x_{\mathcal{F}})$ of its own signature. However, in doing so $\beta(x_{\mathcal{F}})$ becomes fixed and $\alpha^\star$ is not under the adversary's direct control. Once $\alpha^\star$ is computed, the entire signature $x_{\mathcal{F}}$ is fixed, and with high probability, it is not a valid signature.

This minor modification addresses the vulnerability exploited by our attack. However, from a provable security perspective, this construction presents similar problems to the original attempt to generalize [24]. As a result, we are unable to provide a formal proof of security.

6 Conclusions

We proposed a history-free sequential aggregate signature based on generic trapdoor functions and the probabilistic hash-and-sign with retry paradigm. We proved the security of our scheme in the random oracle model, assuming only the one-wayness of the underlying TDF and an additional notion of indistinguishability on preimages. This additional property has been demonstrated for numerous post-quantum TDFs to achieve the security of the related signature schemes. This allowed us to easily instantiate our construction in Section 4 with the UOV, MAYO, and Wave schemes, for which we obtained a compression rate between 5% and 34%. We also pointed out in Section 5 how existing aggregation schemes for multivariate TDFs lack formal security and are insecure for some choices of the underlying function. Therefore, within our knowledge, ours is the first scheme that allows the aggregation of multivariate and code-based HaS signature schemes.

Acknowledgements This publication was created with the co-financing of the European Union FSE-REACT-EU, PON Research and Innovation 2014-2020 DM1062/2021. The first author is a member of the INdAM Research Group GNSAGA. The second author is a member of CrypTO, the group of Cryptography and Number Theory of Politecnico di Torino.

References

1. Albrecht, M.R., Cini, V., Lai, R.W.F., Malavolta, G., Thyagarajan, S.A.K.: Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 102–132. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_4

2. Bellare, M., Namprempre, C., Neven, G.: Unrestricted aggregate signatures. In: Arge, L., Cachin, C., Jurdzinski, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 411–422. Springer, Heidelberg (Jul 2007). https://doi.org/10.1007/978-3-540-73420-8_37
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993). https://doi.org/10.1145/168588.168596
4. Beullens, W.: Improved cryptanalysis of UOV and Rainbow. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 348–373. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_13
5. Beullens, W.: MAYO: Practical post-quantum signatures from oil-and-vinegar maps. In: AlTawy, R., Hülsing, A. (eds.) SAC 2021. LNCS, vol. 13203, pp. 355–376. Springer, Heidelberg (Sep / Oct 2022). https://doi.org/10.1007/978-3-030-99277-4_17
6. Beullens, W., Campos, F., Celi, S., Hess, B., Kannwischer, M.: MAYO - algorithm specifications. MAYO team, latest update: 28/02/2023 (2023), https://pqmayo.org/assets/specs/mayo.pdf, last accessed on 14/04/2023
7. Beullens, W., Chen, M.S., Hung, S.H., Kannwischer, M.J., Peng, B.Y., Shih, C.J., Yang, B.Y.: Oil and vinegar: Modern parameters and implementations. Cryptology ePrint Archive, Report 2023/059 (2023), https://eprint.iacr.org/2023/059
8. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9_26
9. Boudgoust, K., Takahashi, A.: Sequential half-aggregation of lattice-based signatures. Cryptology ePrint Archive, Report 2023/159 (2023), https://eprint.iacr.org/2023/159
10. Brogle, K., Goldberg, S., Reyzin, L.: Sequential aggregate signatures with lazy verification from trapdoor permutations - (extended abstract). In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 644–662. Springer, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34961-4_39
11. Chailloux, A., Debris-Alazard, T.: Tight and optimal reductions for signatures based on average trapdoor preimage sampleable functions and applications to code-based signatures. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 453–479. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45388-6_16
12. Chen, J., Ling, J., Ning, J., Peng, Z., Tan, Y.: MQ aggregate signature schemes with exact security based on UOV signature. In: Liu, Z., Yung, M. (eds.) Information Security and Cryptology - 15th International Conference, Inscrypt 2019, Nanjing, China, December 6-8, 2019, Revised Selected Papers. Lecture Notes in Computer Science, vol. 12020, pp. 443–451. Springer (2019). https://doi.org/10.1007/978-3-030-42921-8_26
13. Chen, Y., Zhao, Y.: Half-aggregation of schnorr signatures with tight reductions. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds.) ESORICS 2022, Part II. LNCS, vol. 13555, pp. 385–404. Springer, Heidelberg (Sep 2022). https://doi.org/10.1007/978-3-031-17146-8_19
14. Debris-Alazard, T., Sendrier, N., Tillich, J.P.: Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 21–51. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34578-5_2

15. Devadas, L., Goyal, R., Kalai, Y., Vaikuntanathan, V.: Rate-1 non-interactive arguments for batch-NP and applications. In: 63rd FOCS. pp. 1057–1068. IEEE Computer Society Press (Oct / Nov 2022). https://doi.org/10.1109/FOCS54457.2022.00103
16. El Bansarkhani, R., Buchmann, J.: Towards lattice based aggregate signatures. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT 14. LNCS, vol. 8469, pp. 336–355. Springer, Heidelberg (May 2014). https://doi.org/10.1007/978-3-319-06734-6_21
17. El Bansarkhani, R., Mohamed, M.S.E., Petzoldt, A.: MQSAS - A multivariate sequential aggregate signature scheme. In: Bishop, M., Nascimento, A.C.A. (eds.) ISC 2016. LNCS, vol. 9866, pp. 426–439. Springer, Heidelberg (Sep 2016). https://doi.org/10.1007/978-3-319-45871-7_25
18. Fischlin, M., Lehmann, A., Schröder, D.: History-free sequential aggregate signatures. In: Visconti, I., Prisco, R.D. (eds.) SCN 12. LNCS, vol. 7485, pp. 113–130. Springer, Heidelberg (Sep 2012). https://doi.org/10.1007/978-3-642-32928-9_7
19. Gentry, C., O'Neill, A., Reyzin, L.: A unified framework for trapdoor-permutation-based sequential aggregate signatures. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 34–57. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76581-5_2
20. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008). https://doi.org/10.1145/1374376.1374407
21. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (May 1999). https://doi.org/10.1007/3-540-48910-X_15
22. Kosuge, H., Xagawa, K.: Probabilistic hash-and-sign with retry in the quantum random oracle model. Cryptology ePrint Archive, Report 2022/1359 (2022), https://eprint.iacr.org/2022/1359
23. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679_28
24. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_5
25. Neven, G.: Efficient sequential aggregate signed data. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 52–69. Springer, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_4
26. Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON. Tech. rep., National Institute of Standards and Technology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
27. Sakumoto, K., Shirai, T., Hiwatari, H.: On provable security of UOV and HFE signature schemes against chosen-message attack. In: Yang, B.Y. (ed.) Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011. pp. 68–82. Springer, Heidelberg (Nov / Dec 2011). https://doi.org/10.1007/978-3-642-25405-5_5
28. Wang, Z., Wu, Q.: A practical lattice-based sequential aggregate signature. In: Steinfeld, R., Yuen, T.H. (eds.) ProvSec 2019. LNCS, vol. 11821, pp. 94–109. Springer, Heidelberg (Oct 2019). https://doi.org/10.1007/978-3-030-31919-9_6

A Missing Proofs

Lemma 2. *For any $k > \tau$ functions $\mathsf{F}_1, \ldots, \mathsf{F}_k \colon \mathcal{X} \to \mathcal{Y}$ and uniformly random $y_1, \ldots, y_k \in \mathcal{Y}$, there exists $x \in \mathcal{X}$ such that $\mathsf{F}_i(x) = y_i$, for every $i = 1, \ldots, k$, with probability at most $|\mathcal{X}|/|\mathcal{Y}|^k$.*

Proof. Let $S_y^{\mathsf{F}} = \{x \in \mathcal{X} : \mathsf{F}(x) = y\}$ be the set of preimages of y under F. For a random choice of y_1 it holds that $|S_{y_1}^{\mathsf{F}_1}| = \alpha$ for some $0 \leq \alpha \leq |\mathcal{X}|$. Then, there are at most α possible values for the tuple $(y_2, \ldots, y_k)$, corresponding to $\{ (\mathsf{F}_2(x), \ldots, \mathsf{F}_k(x)) : x \in S_{y_1}^{\mathsf{F}_1} \}$, such that $\bigcap S_{y_i}^{\mathsf{F}_i} \neq \emptyset$. Since $y_2, \ldots, y_k$ are uniformly chosen in $\mathcal{Y}$, the probability of a non-empty intersection is at most $\alpha|\mathcal{Y}|^{1-k}$. Therefore, the desired probability is bounded by varying over the possible values of α:

$$\sum_{\alpha=0}^{|\mathcal{X}|} \frac{\alpha}{|\mathcal{Y}|^{k-1}} \Pr_{y_1 \leftarrow \$ \mathcal{Y}}\left[|S_{y_1}^{\mathsf{F}_1}| = \alpha\right] = \frac{1}{|\mathcal{Y}|^{k-1}} \sum_{\alpha=0}^{|\mathcal{X}|} \alpha \cdot \frac{|\{ y_1 \in \mathcal{Y} : |S_{y_1}^{\mathsf{F}_1}| = \alpha \}|}{|\mathcal{Y}|} = \frac{|\mathcal{X}|}{|\mathcal{Y}|^k}.$$

Lemma 3. *If an input Q has not been entered in the HTree after being queried to H, the probability that it will ever become tethered to a node in HTree is at most $\tau\mathsf{q}'/|\mathcal{Y}|$, where q' is the number of queries made to H after Q.*

Proof. Suppose that $Q = (\mathsf{F}, m, r, x)$ was queried to H and was not added to the HTree, i.e. $\mathsf{Lookup}(x) = \perp$. Now suppose that a query $Q' = (\mathsf{F}', m', r', x')$ was subsequently sent to H and was added to HTree as part of a node N' with additional value y'. For Q to be tethered to N' it must hold that $\mathsf{F}'(x) = y'$. As noted in the proof of Theorem 1 for the bound on $\Pr[\mathsf{bad}_{\mathsf{tcol}}]$, when a new node is added to the HTree as a result to a call to H, the additional value y is chosen uniformly at random from $\mathcal{Y}$. In particular, y' in random and independent of F' and x. Therefore, the probability of having $\mathsf{F}'(x) = y'$ is $|\mathcal{Y}|^{-1}$. Since there are at most q' queries to H after Q and each query can add at most τ nodes to the HTree, the desired probability follows by the union bound.

B Trapdoor Functions in Hash-and-Sign Signature Schemes

B.1 UOV Trapdoor Function

MQ-based trapdoor function consists of a multivariate quadratic map $\mathcal{P} \colon \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^m$ together with a secret information that allows to efficiently find a preimage. For a random map $\mathcal{P}$, the problem of finding a preimage is called *Multivariate Quadratic (MQ) problem.* The MQ problem is NP-hard over a finite field. Moreover, it is believed to be hard on average if $n \sim m$, both classically and quantumly.

Both UOV and MAYO are based on the same trapdoor function. For the description of the trapdoor function we mainly use the formalism introduced by Beullens in [4].

Algorithm 5: UOV Signature Scheme

```
TrapGen_uov(1^λ):
  1: $O \leftarrow_\$$ $m$-dimensional subspace of $\mathbb{F}_q^n$
  2: $\mathcal{P} \leftarrow_\$$ quadratic map $\mathbb{F}_q^n \to \mathbb{F}_q^m$ that vanishes on $O$
  3: return $\mathsf{F}_{\text{uov}} \leftarrow \mathcal{P}, \mathsf{I}_{\text{uov}} \leftarrow (\mathcal{P}, O)$

$\mathsf{I}^2_{\text{uov}}(\boldsymbol{v}, \boldsymbol{t})$:
  1: if $\{\boldsymbol{o} \in O : \mathcal{P}'(\boldsymbol{v}, \boldsymbol{o}) = \boldsymbol{t} - \mathcal{P}(\boldsymbol{v})\} \neq \emptyset$ then
  2:     return $\perp$
  3: $\boldsymbol{o} \leftarrow_\$ \{\mathcal{P}'(\boldsymbol{v}, \boldsymbol{o}) = \boldsymbol{t} - \mathcal{P}(\boldsymbol{v})\}$
  4: $\boldsymbol{\sigma} \leftarrow \boldsymbol{v} + \boldsymbol{o}$
  5: return $\boldsymbol{\sigma}$

$\mathsf{I}^1_{\text{uov}}()$:
  1: $\boldsymbol{v} \leftarrow_\$ \mathbb{F}_q^n$
  2: return $\boldsymbol{v}$

Sign($\mathsf{I}_{\text{uov}}$, $m$):
  1: $\boldsymbol{v} \leftarrow \mathsf{I}^1_{\text{uov}}()$
  2: repeat
  3:     $r \leftarrow_\$ \{0,1\}^\lambda$
  4:     $\boldsymbol{\sigma} \leftarrow \mathsf{I}^2_{\text{uov}}(\boldsymbol{v}, \mathsf{H}(m, r))$
  5: until $\boldsymbol{\sigma} \neq \perp$
  6: return $(r, \boldsymbol{\sigma})$
```

The trapdoor secret information is a linear subspace $O \subset \mathbb{F}_q^n$ of dimension $\dim(O) = m$. The trapdoor public function is a homogeneous multivariate quadratic map $\mathcal{P}\colon \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^m$ that vanishes on O. For key generation, an m-dimensional subspace $O \subset \mathbb{F}_q^n$ is randomly chosen, then a multivariate quadratic map $\mathcal{P}\colon \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^m$ is randomly chosen such that it vanishes on O. Given a target $\boldsymbol{t} \in \mathbb{F}_q^m$, the secret information O can be used to find a preimage $\boldsymbol{s} \in \mathbb{F}_q^n$, reducing the MQ problem to a linear system. For a map $\mathcal{P}$, we can define its *polar form* as $\mathcal{P}'(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{P}(\boldsymbol{x} + \boldsymbol{y}) - \mathcal{P}(\boldsymbol{x}) - \mathcal{P}(\boldsymbol{y})$. It can be shown that the polar form of a multivariate quadratic map is a symmetric and bilinear map. Now, to find a preimage for $\boldsymbol{t}$, one randomly choose a vector $\boldsymbol{v} \in \mathbb{F}_q^n$ and solves $\mathcal{P}(\boldsymbol{v} + \boldsymbol{o}) = \boldsymbol{t}$ for $\boldsymbol{o} \in O$. Since

$$\boldsymbol{t} = \mathcal{P}(\boldsymbol{v} + \boldsymbol{o}) = \underbrace{\mathcal{P}(\boldsymbol{v})}_{\text{fixed}} + \underbrace{\mathcal{P}(\boldsymbol{o})}_{=0} + \underbrace{\mathcal{P}'(\boldsymbol{v}, \boldsymbol{o})}_{\text{linear in } \boldsymbol{o}},$$

the system reduce to the linear system $\mathcal{P}'(\boldsymbol{v}, \boldsymbol{o}) = \boldsymbol{t} - \mathcal{P}(\boldsymbol{v})$ of m equation and m variables $\boldsymbol{o}$. Notice that whenever the linear map $\mathcal{P}'(\boldsymbol{v}, \cdot)$ is non-singular[5], the system has a unique solution $\boldsymbol{o} \in O$ and the preimage is $\boldsymbol{s} = \boldsymbol{v} + \boldsymbol{o}$.

B.2 Unbalanced Oil and Vinegar

Let $\mathsf{T}_{\text{uov}} = (\mathsf{TrapGen}_{\text{uov}}, \mathsf{F}_{\text{uov}}, \mathsf{I}_{\text{uov}})$ be the TDF based on the description of the previous section. Unbalanced Oil and Vinegar (UOV) [21] is a HaS signature scheme based on T_{uov}. The key generation and the signing procedure with the modified trapdoor functions are shown in Algorithm 5.

By adopting the probabilistic HaS with retry paradigm, UOV is proven EUF-CMA secure in the random oracle model [27]. To obtain uniform preimages

[5] This happens with probability approximately $1 - 1/q$

Algorithm 6: MAYO Signature Scheme

```
$\mathsf{TrapGen}_{\mathrm{mayo}}(1^\lambda)$:
1: $\mathbf{O} \leftarrow_\$ \mathbb{F}_q^{o\times(n-o)}$
2: $O \leftarrow \mathrm{RowSpace}(\mathbf{OI}_o)$
3: $\mathcal{P} \leftarrow_\$$ quadratic map $\mathbb{F}_q^n \to \mathbb{F}_q^m$ that vanishes on $O$
4: return $\mathsf{F}_{\mathrm{mayo}} \leftarrow \mathcal{P}, \mathsf{I}_{\mathrm{mayo}} \leftarrow (\mathcal{P}, \mathbf{O})$

$\mathsf{I}_{\mathrm{mayo}}(\boldsymbol{t})$:
1: $\mathcal{P}^*(\boldsymbol{x}_1,\dots,\boldsymbol{x}_k) \leftarrow \sum_{i=1}^k \mathbf{E}_{i,i}\,\mathcal{P}(\boldsymbol{x}_i) + \sum_{1\le i<j\le k} \mathbf{E}_{i,j}\,\mathcal{P}'(\boldsymbol{x}_i,\boldsymbol{x}_j)$
2: $\boldsymbol{v}_1,\dots,\boldsymbol{v}_k \leftarrow_\$ (\mathbb{F}_q^n \times \mathbf{0}_m)^k$
3: if $\mathcal{P}^*(\boldsymbol{v}_1+\boldsymbol{o}_1,\dots,\boldsymbol{v}_k+\boldsymbol{o}_k)$ does not have full rank then
4:     return $\perp$
5: $\boldsymbol{o}_1,\dots,\boldsymbol{o}_k \leftarrow_\$ \{\boldsymbol{o}_1,\dots,\boldsymbol{o}_k \in O : \mathcal{P}^*(\boldsymbol{v}_1+\boldsymbol{o}_1,\dots,\boldsymbol{v}_k+\boldsymbol{o}_k) = \boldsymbol{t}\}$
6: $\boldsymbol{\sigma} \leftarrow (\boldsymbol{v}_1+\boldsymbol{o}_1,\dots,\boldsymbol{v}_k+\boldsymbol{o}_k)$
7: return $\boldsymbol{\sigma}$
```

over $\mathbb{F}_q^n$, the UOV signing procedure is slightly different from the generic one described in Algorithm 1. The signer starts by fixing a random $\boldsymbol{v} \leftarrow_\$ \mathbb{F}_q^n$, then it repeatedly samples $r \leftarrow_\$ \{0,1\}^\lambda$ until there is a solution to the linear system $\mathcal{P}'(\boldsymbol{v},\boldsymbol{o}) = \mathsf{H}(m,r) - \mathcal{P}(\boldsymbol{v})$ Equivalently, the trapdoor $\mathsf{I}_{\mathrm{uov}}$ can be split in two distinct functions $\mathsf{I}^1_{\mathrm{uov}}$ and $\mathsf{I}^2_{\mathrm{uov}}$. The former is invoked only once and randomly chooses $\boldsymbol{v} \leftarrow_\$ \mathbb{F}_q^n$. The latter is part of the repeat loop and tries to find a preimage $\boldsymbol{s}$ of the corresponding linear system. With this procedure, the authors of [27] proved that the preimages produced from $\mathsf{Sign}(\mathsf{I}_{\mathrm{uov}},\cdot)$ are indistinguishable from the output of $\mathsf{SampDom}(\mathsf{F}_{\mathrm{uov}})$, so that in Theorem 1 we have $\mathsf{Adv}^{\mathrm{PS}}_{\mathsf{T}_{\mathrm{uov}}}(\mathcal{D}) = 0$.

B.3 MAYO

MAYO [5] is a HaS signature scheme based on the UOV trapdoor function and employs a so-called *whipping* technique to use a smaller secret subspace O of dimension $\dim(O) = o < m$. Let $\mathsf{T}_{\mathrm{mayo}} = (\mathsf{TrapGen}_{\mathrm{mayo}}, \mathsf{F}_{\mathrm{mayo}}, \mathsf{I}_{\mathrm{mayo}})$ be the TDF of the MAYO. The key generation process is the same as for UOV and produces a multivariate quadratic map $\mathcal{P}\colon \mathbb{F}_q^n \to \mathbb{F}_q^m$ that vanishes on O. In the signing procedure, $\mathcal{P}$ is deterministically transformed into a larger (whipped) map $\mathcal{P}^*\colon \mathbb{F}_q^{kn} \to \mathbb{F}_q^m$, for some $k > 1$, which vanishes on $O^k \subset \mathbb{F}_q^{kn}$ of dimension $ko \ge m$. In [5], the whipping transformation is obtained by choosing $k(k+1)/2$ random invertible matrices $\{\,\mathbf{E}_{i,j} \in \mathrm{GL}_m(\mathbb{F}_q)\,\}_{1\le i\le j\le k}$ and defining

$$\mathcal{P}^*(\boldsymbol{x}_1,\dots,\boldsymbol{x}_k) = \sum_{i=1}^{k} \mathbf{E}_{i,i}\,\mathcal{P}(\boldsymbol{x}_i) + \sum_{1\le i<j\le k} \mathbf{E}_{i,j}\,\mathcal{P}'(\boldsymbol{x}_i,\boldsymbol{x}_j).$$

Similarly to UOV, to find a preimage for $\boldsymbol{t} \in \mathbb{F}_q^m$, we randomly choose $\boldsymbol{v}_1,\dots,\boldsymbol{v}_k \in \mathbb{F}_q^{n-m} \times \mathbf{0}_m$. Then, $\mathcal{P}^*(\boldsymbol{v}_1+\boldsymbol{o}_1,\dots,\boldsymbol{v}_k+\boldsymbol{o}_k) = \boldsymbol{t}$ is a system of m linear equation in $ko \ge m$ variables, so it will be solvable with high probability. The key generation and the preimage computation via $\mathsf{I}_{\mathrm{mayo}}$ are shown in Algorithm 6.

Instead of computing $\mathsf{Adv}^{\mathrm{PS}}_{\mathsf{T}_{\mathrm{mayo}}}(\mathcal{D})$, we can use the result of [5, Lemma 2] that bounds the probability B that $\mathcal{P}^*(\boldsymbol{v}_1+\boldsymbol{o}_1,\dots,\boldsymbol{v}_k+\boldsymbol{o}_k)$ does not have full rank.

Algorithm 7: Wave Signature Scheme

```
$\mathsf{TrapGen}_{\mathrm{wave}}(1^\lambda)$:
1: $\mathbf{H}_{\mathsf{sk}} \in \mathbb{F}_q^{(n-k)\times n} \leftarrow_\$$ generalized $(U, U+V)$-code
2: $\mathbf{S} \leftarrow_\$ \mathrm{GL}_{n-k}(\mathbb{F}_q)$
3: $\mathbf{P} \leftarrow_\$ n \times n$ permutation matrix
4: $\mathbf{H}_{\mathsf{pk}} \leftarrow \mathbf{S}\mathbf{H}_{\mathsf{sk}}\mathbf{P}$
5: return $\mathsf{F}_{\mathrm{mayo}} \leftarrow \mathbf{H}_{\mathsf{pk}}, \mathsf{I}_{\mathrm{mayo}} \leftarrow (\mathbf{H}_{\mathsf{sk}}, \mathbf{S}, \mathbf{P})$

$\mathsf{I}_{\mathrm{wave}}(\boldsymbol{y})$:
1: $\boldsymbol{e} \leftarrow D_{\mathbf{H}_{\mathsf{sk}}}(\boldsymbol{y}(\mathbf{S}^{-1})^\intercal)$
2: $\boldsymbol{x} \leftarrow \boldsymbol{e}\mathbf{P}$
3: return $\boldsymbol{x}$
```

It can be shown that if $\mathsf{I}_{\mathrm{mayo}}$ has never output $\perp$, then the preimages produced by $\mathsf{Sign}(\mathsf{I}_{\mathrm{mayo}}, \cdot)$ are indistinguishable from $\mathsf{SampDom}(\mathsf{F}_{\mathrm{mayo}})$. Therefore, we can modify the proof of Theorem 1 by introducing a new intermediate game $\mathsf{Game}_{\mathsf{2b}}$. This game is identical to Game_2 except that $\mathsf{OAggSign}$ aborts if $\mathsf{I}_{\mathrm{mayo}}$ outputs $\perp$. Since there are at most q_S queries are made to $\mathsf{OAggSign}$, the probability that $\mathsf{Game}_{\mathsf{2b}}$ does not abort is at least $1 - \mathsf{q}_\mathsf{S}\mathsf{B}$. It follows that $\Pr[\mathsf{Game}_2(\mathcal{A}) = 1] \leq \frac{1}{1-\mathsf{q}_\mathsf{S}\mathsf{B}} \Pr[\mathsf{Game}_{\mathsf{2b}}(\mathcal{A}) = 1]$. Now, when $\mathsf{Game}_{\mathsf{2b}}$ does not abort, the game is indistinguishable from Game_3, so that $\Pr[\mathsf{Game}_3(\mathcal{A}) = 1] = \Pr[\mathsf{Game}_{\mathsf{2b}}(\mathcal{A}) = 1]$. The remainder of the proof proceeds as the original.

B.4 Wave

Wave [14] is a HaS signature scheme based on the family of the generalized $(U, U+V)$-codes. Let $\mathsf{T}_{\mathrm{wave}} = (\mathsf{TrapGen}_{\mathrm{wave}}, \mathsf{F}_{\mathrm{wave}}, \mathsf{I}_{\mathrm{wave}})$ be the TDF of the Wave. The OW security of $\mathsf{F}_{\mathrm{wave}}$ is based on the indistinguishability of $(U, U+V)$-codes from random codes and the Syndrome Decoding (SD) problem. The indistinguishability problem is NP-complete for large finite fields $\mathbb{F}_q$, while the SD problem is NP-hard for arbitrary finite fields. The trapdoor secret information is a random generalized $(U, U+V)$-code over $\mathbb{F}_q$ of length n and dimension $k = k_U + k_V$, described by its parity check matrix $\mathbf{H}_{\mathsf{sk}} \in \mathbb{F}_q^{(n-k)\times n}$, an invertible matrix $\mathbf{S} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ and a permutation matrix $\mathbf{P} \in \mathbb{F}_q^{n\times n}$. Using the underlying structure of the $(U, U+V)$-code, an efficient decoding algorithm $D_{\mathbf{H}_{\mathsf{sk}}}$ is produced. The public function $\mathsf{F}_{\mathrm{wave}}$ is obtained from the parity check matrix $\mathbf{H}_{\mathsf{pk}} = \mathbf{S}\mathbf{H}_{\mathsf{sk}}\mathbf{P}$. Let $S_{w,n}$ be the subset of vectors in $\mathbb{F}_q^n$ with Hamming weight w. The weight w is chosen such that the public function $\mathsf{F}_{\mathrm{wave}} \colon \boldsymbol{e} \in S_{w,n} \mapsto \boldsymbol{e}\mathbf{H}_{\mathsf{pk}}^\intercal \in \mathbb{F}_q^{n-k}$ is a surjection. To find a preimage for $\boldsymbol{y} \in \mathbb{F}_q^{n-k}$, the signer uses the decoding algorithm $D_{\mathbf{H}_{\mathsf{sk}}}$ on $\boldsymbol{y}(\mathbf{S}^{-1})^\intercal$ to find $\boldsymbol{e} \in S_{w,n}$, and finally returns $\boldsymbol{e}\mathbf{P}$. The key generation and the preimage computation via $\mathsf{I}_{\mathrm{mayo}}$ are shown in Algorithm 7.

Wave can be described in the HaS *without* retry paradigm. In [11], $\mathsf{T}_{\mathrm{wave}}$ is described in the context of ATPSF, a weaker notion of PSF where the uniformity property on preimages is required to hold only on average. In particular, for any $(\mathsf{F}, \mathsf{I}) \leftarrow \mathsf{TrapGen}_{\mathrm{wave}}(1^\lambda)$, consider the statistical distance $\varepsilon_{\mathsf{F},\mathsf{I}} =$

$\Delta(\mathsf{SampDom}(\mathsf{F}), \mathsf{I}(U(\mathcal{Y})))$. Then, it holds that $\mathbb{E}_{(\mathsf{F},\mathsf{I})}[\varepsilon_{\mathsf{F},\mathsf{I}}] \leq \varepsilon$, where ε is negligible in the security parameter λ. In Theorem 1 we can use this condition and [11, Prop. 1] to bound the distinguishing advantage on PS with ε, obtaining $\mathsf{Adv}^{\mathrm{PS}}_{\mathsf{T}_{\mathrm{mayo}}}(\mathcal{D}) \leq \mathsf{q}_{\mathsf{S}}\varepsilon$.