Part I
More Preliminaries and Security Proof

A Additional Preliminaries

A.1 Probability and Regularity

The Rényi divergence (RD) defines a measure of distribution closeness. We follow [BLR+18] and set the RD as the exponential of the classical definition.

Definition A.1 (Rényi Divergence). *Let P and Q be two discrete probability distributions such that $\mathsf{Supp}(P) \subseteq \mathsf{Supp}(Q)$. The* Rényi divergence *(of order 2) is defined by*

$$\mathrm{RD}_2(P\|Q) = \sum_{x \in \mathsf{Supp}(P)} \frac{P(x)^2}{Q(x)}.$$

The RD fulfills the following properties, as proved in [vEH14].

Lemma A.2. *Let P, Q be two discrete probability distributions with $\mathsf{Supp}(P) \subseteq \mathsf{Supp}(Q)$.*

Data Processing Inequality: $\mathrm{RD}_2(P^g\|Q^g) \leq \mathrm{RD}_2(P\|Q)$ *for any function g, where P^g (resp. Q^g) denotes the distribution of $g(y)$ induced by sampling $y \leftarrow P$ (resp. $y \leftarrow Q$),*

Probability Preservation: *Let $E \subset \mathsf{Supp}(Q)$ be an event, then*

$$P(E) \leq \sqrt{Q(E) \cdot \mathrm{RD}_2(P\|Q)}.$$

In Section 3, we need the following regularity result.

Lemma A.3 ([BJRW22, Lem. 2.7] Simplified). *Let $k, \ell, q, \gamma \in \mathbb{N}$ such that q is prime. Further, let $R = \mathbb{Z}[X]/\langle X^n + 1\rangle$, where n is a power of 2. Then,*

$$\mathrm{RD}_2((\mathbf{A}, \mathbf{A}\mathbf{y}')\|(\mathbf{A}, \mathbf{v})) \leq \left(1 + \frac{q^k}{(2\gamma+1)^\ell}\right)^n,$$

where $\mathbf{A} \xleftarrow{\$} R_q^{k\times\ell}$, $\mathbf{y}' \xleftarrow{\$} \mathscr{D} := U(S_\gamma^\ell)$ and $\mathbf{v} \xleftarrow{\$} R_q^k$.

In order to obtain a constant Rényi divergence, we require

$$\ell \geq k \cdot \frac{\log_2 q}{\log_2(2\gamma+1)} + O\left(\frac{\log_2 q}{\log_2(2\gamma+1)}\right). \tag{3}$$

Remark A.4. Alternatively, if one prefers the discrete Gaussian distribution for $\mathscr{D}$, one can use the regularity result [LPR13, Cor. 7.5]. It comes with the advantage that it holds for any $q \geq 2$, not necessarily prime, and that the parameter ℓ can be arbitrarily close to k, in particular $\ell = k$ is a possible choice, which is common in practice. However, the Gaussian distribution is with respect to the canonical embedding, which is not implementation friendly and requires to switch between canonical and coefficient embedding throughout the scheme. Further, the resulting Gaussian width is much larger than parameters used in practice.

A.2 Module Lattice Problems

We also recall two lattice problems and refer to [LS15] for more details. We state them in their respective discrete, primal and HNF form.

Definition A.5 (M-LWE). *Let $k, \ell, \eta \in \mathbb{N}$. The* Module Learning With Errors problem $\mathsf{M\text{-}LWE}_{k,\ell,\eta}$ *is defined as follows. Given $\mathbf{A} \xleftarrow{\$} R_q^{k\times\ell}$ and $\mathbf{t} \in R_q^k$. Decide whether $\mathbf{t} \xleftarrow{\$} R_q^k$ or if $\mathbf{t} = [\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{s}$, where $\mathbf{s} \xleftarrow{\$} S_\eta^{\ell+k}$.*

The M-LWE assumption states that no PPT algorithm can distinguish between the two distributions with non-negligible advantage. Worst-case to average-case reductions guarantee that M-LWE is quantumly [LS15] and classically [BJRW20] at least as hard as the approximate shortest vector problem over module lattices.

Definition A.6 (M-SIS). *Let $k, \ell, b \in \mathbb{N}$. The* Module Short Integer Solution problem $\mathsf{M\text{-}SIS}_{k,\ell,b}$ *is as follows. Given a uniformly random matrix $\mathbf{A} \xleftarrow{\$} R_q^{k\times\ell}$. Find a non-zero vector $\mathbf{s} \in R_q^{k+\ell}$ such that $\|\mathbf{s}\|_2 \leq b$ and $[\mathbf{A}|\mathbf{I}_k] \cdot \mathbf{s} = \mathbf{0} \in R_q^k$.*

The M-SIS assumption states that no PPT adversary can solve this problem with non-negligible probability. Worst-case to average-case reductions guarantee that M-SIS is classically [LS15] at least as hard as the approximate shortest independent vector problem over module lattices.

B Proof for Theorem 3.4

Proof. We first sketch the high level ideas of the reduction $\mathcal{B}$. The complete description of $\mathcal{B}$ is found in Alg. 4. The random oracle and the signing oracle in the FH-UF-CMA game (resp. UF-CMA game) are denoted by H and $\mathsf{OSeqSign}$ (resp. H' and OSign). On receiving the public parameter $\mathbf{A}$ and the challenge public key $\mathbf{t}^*$, $\mathcal{B}$ checks that $\mathbf{t}^* \in R_q^k$ contains at least one invertible element. If so, $\mathcal{B}$ forwards $(\mathbf{A}, \mathbf{t}^*)$ to $\mathcal{A}$.

$\mathsf{OSeqSign}$ replies to queries by asking OSign for a signature on uniformly chosen m and programs H such that it outputs c returned by the outer random oracle H'. Here we cannot just forward m_i to OSign, because it might be that a forgery submitted by $\mathcal{A}$ later reuses the same m_i. Then submitting a forgery w.r.t. m_i is not valid in the UF-CMA game, causing $\mathcal{B}$ to lose.

At the core of reduction is simulation of responses to H queries. Suppose the key list L_N as part of the forgery tuple contains $(\mathbf{t}_i, m_i)$ such that $\mathbf{t}_i = \mathbf{t}^*$. Then $\mathcal{B}$ must have extracted the corresponding $\mathbf{u}_i$ and forwarded $\mathbf{u}_i$ to H' together with a random message m, so that $(m, (\mathbf{u}_i, \mathbf{z}_i))$ qualifies as a valid forgery in the UF-CMA game. This extraction operation crucially makes use of $\mathbf{z}_{i-1}$ when $(\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1})$ is queried to H. Intuitively, $\mathbf{z}_{i-1}$ serves as a look-up key to obtain the previous aggregated $\tilde{\mathbf{u}}_{i-1}$, which allows $\mathcal{B}$ to extract $\mathbf{u}_i = \tilde{\mathbf{u}}_i - \tilde{\mathbf{u}}_{i-1}$.

In more detail, starting from the original FH-UF-CMA game, we construct several hybrid games towards the one used by the final reduction $\mathcal{B}$. We denote by $\Pr[\mathrm{G}_i(\mathcal{A})]$ the probability that $\mathrm{G}_i(\mathcal{A})$ halts with output 1.

G_0 This game is identical to the FH-UF-CMA game. At the beginning, the game initializes an empty key-value look-up table HT. Upon receiving a query to the random oracle H with input X, it returns $\mathrm{HT}[X]$ if the table entry is non-empty; otherwise, it samples uniform $c \in \mathsf{Ch}$, sets $\mathrm{HT}[X] := c$, and returns c. It holds that $\Pr[\mathrm{G}_0(\mathcal{A})] = \mathbf{Adv}^{\mathsf{FH\text{-}UF\text{-}CMA}}_{\mathsf{FSwA\text{-}SAS}}(\mathcal{A})$.

G_1 This game is identical to G_0, except that $\mathsf{OSeqSign}$ samples uniform $c_i \in \mathsf{Ch}$ instead of calling $c_i = \mathsf{H}(\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1})$ after $\tilde{\mathbf{u}}_i$ is computed, and that it programs the RO table $\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}] := c_i$ as soon as the rejection sampling step succeeds; if $\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}]$ is already set, the game aborts by setting $\mathsf{bad}_{\mathsf{ucol}} = \mathsf{true}$. It holds that $|\Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})]| \leq \Pr[\mathsf{bad}_{\mathsf{ucol}}]$.

G_2 This game is identical to G_1, except that responses to random oracle queries $\mathsf{H}(\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1})$ are simulated as follows. Initialize an empty key-value look-up table ZT. If $i = 1$ or there exists some $X := (\tilde{\mathbf{u}}_{i-1}, L_{i-1}, \mathbf{z}_{i-2})$ such that $\mathrm{ZT}[X] = \bar{\mathbf{A}}\mathbf{z}_{i-1} \bmod q$, then extract $\mathbf{u}_i := \tilde{\mathbf{u}}_i - \tilde{\mathbf{u}}_{i-1}$, sample uniform $c_i \in \mathsf{Ch}$, and set $\mathrm{ZT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}] := \mathbf{u}_i + c_i\mathbf{t}_i$. If there already exists some entry $X' := (\tilde{\mathbf{u}}_{i-1}, L_{i-1}, \mathbf{z}_{i-2})$ such that $\mathrm{ZT}[X'] = \mathbf{u}_i + c_i\mathbf{t}_i$, the game aborts by setting $\mathsf{bad}_{\mathsf{zcol}} = \mathsf{true}$. It holds that $|\Pr[\mathrm{G}_1(\mathcal{A})] - \Pr[\mathrm{G}_2(\mathcal{A})]| \leq \Pr[\mathsf{bad}_{\mathsf{zcol}}]$.

G_3 This game is identical to G_2, except that $\mathsf{OSeqSign}$ and H proceed as follows. The game initializes an empty set $\mathcal{M}$ and key-value look-up table MT. Whenever $\mathsf{OSeqSign}$ receives a query, it internally samples a uniform message $m \in M$ and adds m to $\mathcal{M}$. Whenever H receives a query with input $(\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1})$ and manages to extract $\mathbf{u}_i$ as above, it samples a uniform message $m \in M$ and aborts by setting $\mathsf{bad}_{\mathsf{mcol}} = \mathsf{true}$ if $m \in \mathcal{M}$. Else, it sets $\mathrm{MT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}] = m$. It holds that $|\Pr[\mathrm{G}_2(\mathcal{A})] - \Pr[\mathrm{G}_3(\mathcal{A})]| \leq \Pr[\mathsf{bad}_{\mathsf{mcol}}]$.

G_4 This game is identical to G_3, except that it performs the following checks against the ZT entries after the adversary outputs a valid signature-history pair $(L_N, (\tilde{\mathbf{u}}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N))$ as follows. Let $\tilde{\mathbf{u}}_{N-1}, \ldots, \tilde{\mathbf{u}}_1$ be as derived during the execution of $\mathsf{SeqVerify}$. If for some $i \in [N]$ the entry $\mathrm{ZT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}]$ is undefined, the game halts by setting $\mathsf{bad}_{\mathsf{ord}} = \mathsf{true}$. It holds that $|\Pr[\mathrm{G}_3(\mathcal{A})] - \Pr[\mathrm{G}_4(\mathcal{A})]| \leq \Pr[\mathsf{bad}_{\mathsf{ord}}]$.

$\mathcal{B}$ Given an adversary $\mathcal{A}$ winning G_4, the reduction $\mathcal{B}$ described in Alg. 4 is obtained as follows. Upon receiving a query to $\mathsf{OSeqSign}$, $\mathcal{B}$ makes a query to OSign of the UF-CMA game with a uniform message $m \in M$, receives $\mathbf{u}_i$ and $\mathbf{z}_i$, and programs HT using challenge c_i output by the outer random oracle $\mathsf{H}'(\mathbf{u}_i, \mathbf{t}^*, m)$. Moreover, H obtains fresh challenge c_i for $\mathbf{t}_i = \mathbf{t}^*$ by querying the outer random oracle $\mathsf{H}'(\mathbf{u}_i, \mathbf{t}^*, m)$ if it succeeds in extracting $\mathbf{u}_i = \tilde{\mathbf{u}}_i - \tilde{\mathbf{u}}_{i-1}$. Since $\mathcal{A}$ is guaranteed to receive an invertible challenge public key in $\mathcal{B}$, the view of $\mathcal{A}$ is identical to that of G_4.

We now show that, as long as none of the bad events happen, $\mathcal{B}$ is guaranteed to output a message-signature pair $(m, (\mathbf{u}_{i^*}, \mathbf{z}_{i^*}))$ that gets accepted in the UF-CMA game, i.e., $\|\mathbf{z}_{i^*}\|_\infty \leq B$ and $\mathbf{u}_{i^*} = \bar{\mathbf{A}}\mathbf{z}_{i^*} - c\mathbf{t}^* \bmod q$ where $c = \mathsf{H}'(\mathbf{u}_{i^*}, \mathbf{t}^*, m)$. The former condition is immediate from the verification condition of $\mathsf{SeqVerify}$. To argue the latter, notice that we have $c = \mathsf{H}'(\mathbf{u}_{i^*}, \mathbf{t}^*, m) = \mathrm{HT}[\tilde{\mathbf{u}}_{i^*}, L_{i^*}, \mathbf{z}_{i^*-1}] = c_{i^*}$ as long as the RO entries $\mathrm{HT}[\tilde{\mathbf{u}}_1, L_1, \mathbf{z}_0], \ldots, \mathrm{HT}[\tilde{\mathbf{u}}_N, L_N, \mathbf{z}_{N-1}]$ have been set in

the right order and thus $\mathbf{u}_{i^*} = \tilde{\mathbf{u}}_{i^*} - \tilde{\mathbf{u}}_{i^*-1}$ is extracted during the invocation of $\mathsf{H}(\tilde{\mathbf{u}}_{i^*}, L_{i^*}, \mathbf{z}_{i^*-1})$. The following lemma indeed assures that such queries have been made in the right order as long as $\mathsf{bad}_{\mathsf{zcol}} = \mathsf{bad}_{\mathsf{ord}} = \mathsf{false}$.

Lemma B.1. *Let $\sigma_N = (\tilde{\mathbf{u}}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N)$ and $L_N = (\mathbf{t}_1, m_1)||\ldots||(\mathbf{t}_N, m_N)$ a valid signature-history pair that $\mathcal{B}$ received from $\mathcal{A}$. Let $\tilde{\mathbf{u}}_1, \ldots, \tilde{\mathbf{u}}_{N-1}$ be as derived in* SeqVerify *run by $\mathcal{B}$. Suppose $\mathsf{bad}_{\mathsf{zcol}} = \mathsf{false}$. Then for $i \in [N-1]$, the random oracle entry $\mathrm{HT}[\tilde{\mathbf{u}}_{i+1}, L_{i+1}, \mathbf{z}_i]$ had been set after $\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}]$ was set if and only if $\mathsf{bad}_{\mathsf{ord}} = \mathsf{false}$.*

Proof. "Only if" We first argue that if the oracle entries $\mathrm{HT}[\tilde{\mathbf{u}}_1, L_1, \mathbf{z}_0], \ldots, \mathrm{HT}[\tilde{\mathbf{u}}_N, L_N, \mathbf{z}_{N-1}]$ have been set in this order and $\mathsf{bad}_{\mathsf{zcol}} = \mathsf{false}$, the corresponding ZT entries are all non-empty and thus $\mathsf{bad}_{\mathsf{ord}}$ must be false. Suppose this statement holds for $1 \le i \le j$, i.e., $\mathrm{ZT}[\tilde{\mathbf{u}}_1, L_1, \mathbf{z}_0], \ldots, \mathrm{ZT}[\tilde{\mathbf{u}}_j, L_j, \mathbf{z}_{j-1}]$ are non-empty. Due to the verification condition it must be that $\tilde{\mathbf{u}}_{j-1} = \tilde{\mathbf{u}}_j - \bar{\mathbf{A}}\mathbf{z}_j + c_j\mathbf{t}_j$ and thus $\bar{\mathbf{A}}\mathbf{z}_j = \tilde{\mathbf{u}}_j - \tilde{\mathbf{u}}_{j-1} + c_j + \mathbf{t}_j$. Because we assumed that $\mathrm{HT}[\tilde{\mathbf{u}}_{j-1}, L_{j-1}, \mathbf{z}_{j-2}]$ and $\mathrm{HT}[\tilde{\mathbf{u}}_j, L_j, \mathbf{z}_{j-1}]$ are set in this order, the invocation of $\mathsf{H}(\tilde{\mathbf{u}}_j, L_j, \mathbf{z}_{j-1})$ must have extracted $\mathbf{u}_j = \tilde{\mathbf{u}}_j - \tilde{\mathbf{u}}_{j-1}$ and have set $\mathrm{ZT}[\tilde{\mathbf{u}}_j, L_j, \mathbf{z}_{j-1}] = \mathbf{u}_j + c_j\mathbf{t}_j$. Note that, since $\mathsf{bad}_{\mathsf{zcol}} = \mathsf{false}$, there is no other entry in ZT that records the same value as $\mathbf{u}_j + c_j\mathbf{t}_j$. Thus, when $(\tilde{\mathbf{u}}_{j+1}, L_{j+1}, \mathbf{z}_j)$ is queried, H can uniquely find a tuple $(\tilde{\mathbf{u}}_j, L_j, \mathbf{z}_{j-1})$ such that $\mathrm{ZT}[\tilde{\mathbf{u}}_j, L_j, \mathbf{z}_{j-1}] = \bar{\mathbf{A}}\mathbf{z}_j$ and then set $\mathrm{ZT}[\tilde{\mathbf{u}}_{j+1}, L_{j+1}, \mathbf{z}_j] = \tilde{\mathbf{u}}_{j+1} - \tilde{\mathbf{u}}_j + c_{j+1}\mathbf{t}_{j+1}$. It is easy to see that the base case $j = 1$ is true: when $(\tilde{\mathbf{u}}_1, L_1, \mathbf{z}_0)$ is queried, the invocation of H always sets $\mathrm{ZT}[\tilde{\mathbf{u}}_1, L_1, \mathbf{z}_0] = \mathbf{u}_1 + c_1\mathbf{t}_1$ where $\mathbf{u}_1 = \tilde{\mathbf{u}}_1$.

"If" We give a proof by induction. As an induction hypothesis, we assume that for $i = 1, \ldots, j-1$ the random oracle entry $\mathrm{HT}[\tilde{\mathbf{u}}_{i+1}, L_{i+1}, \mathbf{z}_i]$ had been set after $\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}]$ was set whenever $\mathsf{bad}_{\mathsf{zcol}} = \mathsf{bad}_{\mathsf{ord}} = \mathsf{false}$. Now suppose, for a contradiction, that $\mathrm{HT}[\tilde{\mathbf{u}}_{j+1}, L_{j+1}, \mathbf{z}_j]$ was set *before* $\mathrm{HT}[\tilde{\mathbf{u}}_j, L_j, \mathbf{z}_{j-1}]$ while $\mathsf{bad}_{\mathsf{zcol}} = \mathsf{bad}_{\mathsf{ord}} = \mathsf{false}$. Because $\mathsf{bad}_{\mathsf{ord}} = \mathsf{false}$, the entry $\mathrm{ZT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}]$ is non-empty for all $i \in [N]$. When H is queried with input $(\tilde{\mathbf{u}}_{j+1}, L_{j+1}, \mathbf{z}_j)$, since the corresponding entry in ZT is non-empty, it must be that there exists some $X' := (\tilde{\mathbf{u}}'_j, L_j, \mathbf{z}'_{j-1}) \neq (\tilde{\mathbf{u}}_j, L_j, \mathbf{z}_{j-1})$ such that $c'_j := \mathrm{HT}[X']$ and $\mathrm{ZT}[X'] = \bar{\mathbf{A}}\mathbf{z}_j$ are already set. This implies that X' has been queried to H before and that $\bar{\mathbf{A}}\mathbf{z}_j = \mathbf{u}'_j + c'_j\mathbf{t}_j \bmod q$, where $\mathbf{u}'_j$ is some value extracted inside $\mathsf{H}(X')$. On the other hand, due to the verification condition it also holds that $\bar{\mathbf{A}}\mathbf{z}_j = \mathbf{u}_j + c_j\mathbf{t}_j \bmod q$, where $c_j = \mathrm{HT}[\tilde{\mathbf{u}}_j, L_j, \mathbf{z}_{j-1}]$ and $\mathbf{u}_j = \tilde{\mathbf{u}}_j - \tilde{\mathbf{u}}_{j-1}$. Here, $\mathbf{u}_j$ is the value extracted when $(\mathbf{u}_j, L_j, \mathbf{z}_{j-1})$ is queried to H for the first time, because due to the induction hypothesis $\mathrm{HT}[\tilde{\mathbf{u}}_{j-1}, L_{j-1}, \mathbf{z}_{j-2}]$ had been already set at this point. However, this implies that $\mathsf{bad}_{\mathsf{zcol}}$ is set when $(\tilde{\mathbf{u}}_j, L_j, \mathbf{z}_{j-1})$ is queried to H, contradicting the assumption that $\mathsf{bad}_{\mathsf{zcol}} = \mathsf{false}$.

Let us prove the base case $j = 2$ in a similar manner. Suppose $\mathrm{HT}[\tilde{\mathbf{u}}_2, L_2, \mathbf{z}_1]$ was set *before* $\mathrm{HT}[\tilde{\mathbf{u}}_1, L_1, \mathbf{z}_0]$ while $\mathsf{bad}_{\mathsf{zcol}} = \mathsf{bad}_{\mathsf{ord}} = \mathsf{false}$. When H is queried with input $(\tilde{\mathbf{u}}_2, L_2, \mathbf{z}_1)$, since the corresponding entry in ZT is non-empty, it must be that there exists some $X' := (\tilde{\mathbf{u}}'_1, L_1, \mathbf{z}_0) \neq (\tilde{\mathbf{u}}_1, L_1, \mathbf{z}_0)$ such that $c'_1 := \mathrm{HT}[X']$ and $\mathrm{ZT}[X'] = \bar{\mathbf{A}}\mathbf{z}_1$ are already set. This implies that X' has been queried to H before and that $\bar{\mathbf{A}}\mathbf{z}_1 = \tilde{\mathbf{u}}'_1 + c'_1\mathbf{t}_1 \bmod q$. On the other hand, due to the

verification condition it also holds that $\bar{\mathbf{A}}\mathbf{z}_1 = \tilde{\mathbf{u}}_1 + c_1\mathbf{t}_1 \bmod q$, where $c_1 = \mathrm{HT}[\tilde{\mathbf{u}}_1, L_1, \mathbf{z}_0]$. However, this implies that $\mathsf{bad}_{\mathsf{zcol}}$ is set when $(\tilde{\mathbf{u}}_1, L_1, \mathbf{z}_0)$ is queried to H, contradicting the assumption that $\mathsf{bad}_{\mathsf{zcol}} = \mathsf{false}$. □

All in all, unless $\mathcal{B}$ sets $\mathsf{bad}_{\mathsf{inv}} = \mathsf{true}$, $\mathcal{B}$ wins the UF-CMA game if and only if G_4 outputs 1. In other words,

$$\begin{aligned}\mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathsf{FSwA\text{-}S}}(\mathcal{B}) &= (1 - \Pr[\mathsf{bad}_{\mathsf{inv}}]) \cdot \Pr[\mathrm{G}_4] \\ &\geq p_{\mathsf{inv}} \cdot \left(\mathbf{Adv}^{\mathsf{FH\text{-}UF\text{-}CMA}}_{\mathsf{FSwA\text{-}SAS}}(\mathcal{A}) - \Pr[\mathsf{bad}_{\mathsf{ucol}}] - \Pr[\mathsf{bad}_{\mathsf{zcol}}] - \Pr[\mathsf{bad}_{\mathsf{mcol}}] - \Pr[\mathsf{bad}_{\mathsf{ord}}]\right).\end{aligned}$$

The running time of $\mathcal{B}$ is at most the running time of $\mathcal{A}$ plus the time it takes for running verification operations and handling random oracle queries. The former takes $O(Nk\ell t_{\mathsf{pmul}})$ because each iteration of the for-loop involves matrix-vector multiplication $\bar{\mathbf{A}}\mathbf{z}_i$ (ignoring the run-time for polynomial addition as it's much smaller than multiplication). The latter takes $O(Q_h k\ell t_{\mathsf{pmul}})$ because $\mathcal{B}$ carries out matrix-vector multiplication $\bar{\mathbf{A}}\mathbf{z}_{i-1}$ for each query to the RO H.

In the following, we provide a concrete bound for each bad event.

Bounding $\Pr[\mathsf{bad}_{\mathsf{mcol}}]$ Assuming that the adversary makes at most Q_s queries to $\mathsf{OSeqSign}$, there are at most Q_s distinct values in $\mathcal{M}$ and Q_h distinct values in MT, respectively. The $\mathsf{bad}_{\mathsf{mcol}}$ flag is potentially set due to two different causes: (1) H internally samples m that is already recorded in $\mathcal{M}$ and thus the corresponding entry is not stored in MT, or (2) $\mathsf{OSeqSign}$ internally samples m that is already recorded in MT and thus the corresponding entry gets removed. The probability that a randomly sampled m inside H collides with one of the values in $\mathcal{M}$ is at most $Q_s/|M| = Q_s/2^l$. Since at most Q_h queries to H are made by $\mathcal{A}$, the probability that case (1) occurs during such queries is at most $Q_s \cdot Q_h/2^l$. If for some $i \in I$ a tuple $(\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1})$ is queried to H during the invocation of $\mathsf{SeqVerify}$ for the first time, in order to cause $\mathsf{bad}_{\mathsf{mcol}} = \mathsf{true}$, *for all such* i independently sampled m must be in $\mathcal{M}$. Thus, the probability that case (1) occurs during such queries is at most $Q_s/2^l$. The probability that a randomly sample m inside $\mathsf{OSeqSign}$ collides with one of the values in MT is at most $Q_h/2^l$. Since at most Q_s queries to $\mathsf{OSeqSign}$ are made, the probability that case (2) occurs is at most $Q_s \cdot Q_h/2^l$. Overall, we get $\Pr[\mathsf{bad}_{\mathsf{mcol}}] \leq Q_s \cdot (2Q_h + 1)/2^l$.

Bounding $\Pr[\mathsf{bad}_{\mathsf{ucol}}]$ Since there are at most $Q_h + Q_s$ values in HT and $\mathbf{u}$ is generated by the signing algorithm Sign from $\mathsf{FSwA\text{-}S}$, for each query to $\mathsf{OSeqSign}$ the probability that the flag $\mathsf{bad}_{\mathsf{ucol}}$ is set is at most

$$\max_{\mathbf{u}} \Pr\left[\mathbf{u} = \bar{\mathbf{A}}\mathbf{y} \bmod q \;:\; \mathbf{y} \xleftarrow{\$} \mathscr{D}^{\ell+k}\right]. \tag{4}$$

Equation 4 can be upper bounded by Lemma A.2, using the probability preservation property of the Rényi divergence. It yields

$$\begin{aligned}&\max_{\mathbf{u}} \Pr\left[\mathbf{u} = \bar{\mathbf{A}}\mathbf{y} \bmod q \;:\; \mathbf{y} \xleftarrow{\$} \mathscr{D}^{\ell+k}\right] \\ &\leq \sqrt{\max_{\mathbf{u}} \Pr\left[\mathbf{u} = \mathbf{v} \bmod q \;:\; \mathbf{v} \xleftarrow{\$} R_q^k\right]} \cdot \mathrm{RD}_2((\bar{\mathbf{A}}, \bar{\mathbf{A}}\mathbf{y}) \| (\bar{\mathbf{A}}, \mathbf{v})).\end{aligned}$$

Algorithm 4: Reduction to UF-CMA security of FSwA-S

The random oracle in the UF-CMA game is denoted by H'. The sign oracle in the UF-CMA game is denoted by OSign. Let $\tilde{\mathbf{u}}_0 = 0$. Without loss of generality, $\mathcal{A}$ queries H with input public keys $\mathbf{t}_1, \ldots, \mathbf{t}_i$ all of which contain at least one invertible element, because otherwise such keys will be rejected by the verification algorithm anyway. All flags are initially set to false.

$\mathcal{B}^{\mathsf{OSign},\mathsf{H}'}(\bar{\mathbf{A}}, \mathbf{t}^*)$

$\mathcal{Q} := \varnothing$; $\mathcal{M} := \varnothing$
if $\mathbf{t}^*$ has no invertible element **then**
  $\mathsf{bad}_{\mathsf{inv}} := \mathsf{true}$
$(\sigma_N, L_N) \leftarrow \mathcal{A}^{\mathsf{OSeqSign},\mathsf{H}}(\bar{\mathbf{A}}, \mathbf{t}^*)$
$(\mathbf{t}_1, m_1)||\ldots||(\mathbf{t}_N, m_N) := L_N$
$(\tilde{\mathbf{u}}_N, \mathbf{z}_1 \ldots, \mathbf{z}_N) := \sigma_N$
$I := \{i \in [N] : \mathbf{t}_i = \mathbf{t}^* \wedge (m_i, L_{i-1}) \notin \mathcal{Q}\}$
if $\mathsf{SeqVerify}(\sigma_N, L_N) = 1 \wedge |I| \neq 0$ **then**
  Derive $\tilde{\mathbf{u}}_{N-1}, \ldots, \tilde{\mathbf{u}}_1$ as in $\mathsf{SeqVerify}$
  if $\exists i \in [N]$ such that $\mathrm{ZT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}] = \perp$ **then**
    $\mathsf{bad}_{\mathsf{ord}} := \mathsf{true}$
  if $\exists i^* \in I$ such that $m := \mathrm{MT}[\tilde{\mathbf{u}}_{i^*}, L_{i^*}, \mathbf{z}_{i^*-1}] \neq \perp$ **then**
    $\mathbf{u}_{i^*} := \tilde{\mathbf{u}}_{i^*} - \tilde{\mathbf{u}}_{i^*-1}$
    return $(m, \mathbf{u}_{i^*}, \mathbf{z}_{i^*})$
  else
    $\mathsf{bad}_{\mathsf{mcol}} := \mathsf{true}$

$\mathsf{OSeqSign}(m_i, L_{i-1}, \sigma_{i-1})$

$\mathcal{Q} := \mathcal{Q} \cup \{(m_i, L_{i-1})\}$
$(\tilde{\mathbf{u}}_{i-1}, \mathbf{z}_1, \ldots, \mathbf{z}_{i-1}) := \sigma_{i-1}$
$m \xleftarrow{\$} M$
if $\exists X$ such that $\mathrm{MT}[X] = m$ **then**
  $\mathrm{MT}[X] := \perp$
$\mathcal{M} := \mathcal{M} \cup \{m\}$
$(\mathbf{u}, \mathbf{z}) \leftarrow \mathsf{OSign}(m)$
$c := \mathsf{H}'(\mathbf{u}, m, \mathbf{t}^*)$
$L_i := L_{i-1}||(\mathbf{t}^*, m_i)$
$\tilde{\mathbf{u}}_i := \tilde{\mathbf{u}}_{i-1} + \mathbf{u} \bmod q$
$\mathbf{z}_i := \mathbf{z}$
if $\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}] \neq \perp$ **then**
  $\mathsf{bad}_{\mathsf{ucol}} := \mathsf{true}$
else
  $\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}] := c$
$\sigma_i := (\tilde{\mathbf{u}}_i, \mathbf{z}_1, \ldots, \mathbf{z}_i)$
return σ_i

$\mathsf{H}(\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1})$

if $\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}] \neq \perp$ **then**
  return $\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}]$
$(\mathbf{t}_1, m_1)||\ldots||(\mathbf{t}_i, m_i) := L_i$
if $i = 1$ or $\exists X := (\tilde{\mathbf{u}}_{i-1}, L_{i-1}, \mathbf{z}_{i-2})$ such that $\mathrm{HT}[X] \neq \perp \wedge \mathrm{ZT}[X] = \bar{\mathbf{A}}\mathbf{z}_{i-1} \bmod q$ **then**
  $\mathbf{u}_i := \tilde{\mathbf{u}}_i - \tilde{\mathbf{u}}_{i-1} \bmod q$
  if $\mathbf{t}_i = \mathbf{t}^*$ **then**
    $m \xleftarrow{\$} M$
    $c_i := \mathsf{H}'(\mathbf{u}_i, m, \mathbf{t}^*)$
    if $m \notin \mathcal{M}$ **then**
      $\mathrm{MT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}] := m$
  else
    $c_i \xleftarrow{\$} \mathsf{Ch}$
  if $\exists X' := (\tilde{\mathbf{u}}'_i, L_i, \mathbf{z}'_{i-1})$ such that $\mathrm{ZT}[X'] = \mathbf{u}_i + c_i\mathbf{t}_i \bmod q$ **then**
    $\mathsf{bad}_{\mathsf{zcol}} := \mathsf{true}$
  $\mathrm{ZT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}] := \mathbf{u}_i + c_i\mathbf{t}_i \bmod q$
else
  $c_i \xleftarrow{\$} \mathsf{Ch}$
$\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}] := c_i$
return c_i

The probability in the second inequality is given by $\sqrt{1/|R_q^k|} = q^{-nk/2}$. By Lem. A.2, using the data processing inequality, it holds $\mathrm{RD}_2((\bar{\mathbf{A}}, \bar{\mathbf{A}}\mathbf{y})\|(\bar{\mathbf{A}}, \mathbf{v})) \leq \mathrm{RD}_2((\mathbf{A}, \mathbf{A}\mathbf{y}')\|(\mathbf{A}, \mathbf{v}))$, where $\mathbf{y}' \xleftarrow{\$} \mathscr{D}^\ell$. By Lem. A.3, the latter is bounded above by a constant if Eq. 3 is fulfilled. Since $\mathsf{OSeqSign}$ receives at most Q_s queries, overall, we obtain $\Pr[\mathsf{bad}_{\mathsf{ucol}}] \leq O(Q_s(Q_h + Q_s)/q^{nk/2})$.

Bounding $\Pr[\mathsf{bad}_{\mathsf{zcol}}]$ Fix an existing entry in ZT of the form $\tilde{\mathbf{u}}_i' + c_i'\mathbf{t}_i \bmod q$. Then the probability that $\mathbf{u}_i + c_i\mathbf{t}_i \bmod q$ hits such an entry is

$$\Pr_{c_i \xleftarrow{\$} \mathsf{Ch}}\left[\mathbf{u}_i + c_i\mathbf{t}_i = \tilde{\mathbf{u}}_i' + c_i'\mathbf{t}_i \bmod q\right] = \Pr_{c_i \xleftarrow{\$} \mathsf{Ch}}\left[c_i\mathbf{t}_i = \tilde{\mathbf{u}}_i' + c_i'\mathbf{t}_i - \mathbf{u}_i \bmod q\right].$$

Since at least one coefficient of $\mathbf{t}_i$ is invertible, the above probability is bounded by $1/|\mathsf{Ch}|$. Let Q_i be the number of entries in HT indexed by a tuple containing a history L of size i. Then we have $Q_h + Q_s = \sum_{i=1}^{N} Q_i$. Because H receives at most $Q_i + 1$ queries for each (where "+1" comes from the fact that an additional query is made from inside $\mathsf{SeqVerify}$), by the union bound, we have that $\Pr[\mathsf{bad}_{\mathsf{zcol}}] \leq \sum_{i=1}^{N} Q_i(Q_i + 1)/(2|\mathsf{Ch}|)$.

Bounding $\Pr[\mathsf{bad}_{\mathsf{ord}}]$ Due to Lemma B.1, "$\mathsf{bad}_{\mathsf{ord}} = \mathsf{true}$ while $\mathsf{bad}_{\mathsf{zcol}} = \mathsf{false}$" implies that there exists some $i \in [N-1]$, such that the entry $\mathrm{HT}[\tilde{\mathbf{u}}_{i+1}, L_{i+1}, \mathbf{z}_i]$ was set before $\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}]$, where $(\tilde{\mathbf{u}}_N, (\mathbf{z}_1, \ldots, \mathbf{z}_N))$ and L_N are signature-history pair output by $\mathcal{A}$ at the end of the game. We argue that this event occurs with negligible probability if the verification condition is satisfied. The event potentially occurs in two ways: (1) for some $i \in [N-1]$, $\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}]$ is set for the first time *during the invocation of* $\mathsf{SeqVerify}$, and (2) for some $i \in [N-1]$, $\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}]$ was set for the first time *before the invocation of* $\mathsf{SeqVerify}$, *but after* $\mathrm{HT}[\tilde{\mathbf{u}}_{i+1}, L_{i+1}, \mathbf{z}_i]$ *was set*.

To bound case (1), it is sufficient to prove the following statement inductively: if none of $\mathrm{HT}[\tilde{\mathbf{u}}_1, L_1, \mathbf{z}_0], \ldots, \mathrm{HT}[\tilde{\mathbf{u}}_{i-1}, L_{i-1}, \mathbf{z}_{i-2}]$ have been set for the first time during the invocation of $\mathsf{SeqVerify}$, the probability that $\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}]$ is set for the first time while running $\mathsf{SeqVerify}$ *and* that the verification condition is satisfied, is at most $Q_{i-1}/|\mathsf{Ch}|$, where Q_i's are defined as above and let $Q_0 = 1$ for convenience. Since $\tilde{\mathbf{u}}_{i-1}, \tilde{\mathbf{u}}_i, \mathbf{z}_i, \mathbf{t}_i$ have been already fixed at the moment when $(\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1})$ is queried to H inside $\mathsf{SeqVerify}$, the probability that the signature gets accepted is

$$\Pr_{c_i \xleftarrow{\$} \mathsf{Ch}}\left[\tilde{\mathbf{u}}_{i-1} = \tilde{\mathbf{u}}_i - (\bar{\mathbf{A}}\mathbf{z}_i - c_i\mathbf{t}_i) \bmod q\right],$$

which is at most $1/|\mathsf{Ch}|$. Because there are at most Q_{i-1} entries HT indexed by a tuple containing L_i and thus at most Q_{i-1} different values for $\tilde{\mathbf{u}}_{i-1}$ exist, by the union bound, the probability that case (1) happens is at most $Q_{i-1}/|\mathsf{Ch}|$. The base case is true: if $i = 1$, since $\tilde{\mathbf{u}}_1, \mathbf{z}_1, \mathbf{t}_1$ have been already fixed at the moment when $(\tilde{\mathbf{u}}_1, L_1, \mathbf{z}_0)$ is queried to H inside $\mathsf{SeqVerify}$, the probability that the signature gets accepted is

$$\Pr_{c_1 \xleftarrow{\$} \mathsf{Ch}}\left[\tilde{\mathbf{u}}_1 = \bar{\mathbf{A}}\mathbf{z}_1 - c_1\mathbf{t}_1 \bmod q\right],$$

which is at most $1/|\mathsf{Ch}|$.

To bound case (2), it is sufficient to prove the following statement inductively: if $\mathrm{HT}[\tilde{\mathbf{u}}_1, L_1, \mathbf{z}_0], \ldots, \mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}]$ have been set in this order before the invocation of $\mathsf{SeqVerify}$, the probability that $\mathrm{HT}[\tilde{\mathbf{u}}_{i+1}, L_{i+1}, \mathbf{z}_i]$ was set before $\mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}]$ *and* that the verification condition is satisfied, is at most $Q_iQ_{i+1}/|\mathsf{Ch}|$, where Q_i's are defined as above. By the definition of the verification procedure, it holds that $\tilde{\mathbf{u}}_{i-1} = \tilde{\mathbf{u}}_i - (\bar{\mathbf{A}}\mathbf{z}_i - c_i\mathbf{t}_i) \bmod q$. However, because both $\mathrm{HT}[\tilde{\mathbf{u}}_{i-1}, L_{i-1}, \mathbf{z}_{i-2}]$ and $\mathrm{HT}[\tilde{\mathbf{u}}_{i+1}, L_{i+1}, \mathbf{z}_i]$ have been already set *before* $c_i = \mathrm{HT}[\tilde{\mathbf{u}}_i, L_i, \mathbf{z}_{i-1}]$ is sampled, for fixed $\tilde{\mathbf{u}}_i, \tilde{\mathbf{u}}_{i-1}, \mathbf{z}_i, \mathbf{t}_i, \bar{\mathbf{A}}$, the probability that fresh c_i meets the verification condition is

$$\Pr_{c_i \xleftarrow{\$} \mathsf{Ch}} \left[\tilde{\mathbf{u}}_{i-1} = \tilde{\mathbf{u}}_i - (\bar{\mathbf{A}}\mathbf{z}_i - c_i\mathbf{t}_i) \bmod q\right],$$

which is at most $1/|\mathsf{Ch}|$. Because there are at most Q_{i+1} different existing entries of $\mathrm{HT}[\tilde{\mathbf{u}}_{i+1}, L_{i+1}, \mathbf{z}_i]$ and fresh c_i is sampled at most Q_i times, by the union bound, we obtain the overall upper bound $Q_iQ_{i+1}/|\mathsf{Ch}|$. The base case $i = 1$ is clearly true: if the tuple $(\tilde{\mathbf{u}}_1, L_1, \mathbf{z}_0)$ is queried after $\mathrm{HT}[\tilde{\mathbf{u}}_2, L_2, \mathbf{z}_1]$ has been set, for the verification condition to be met fresh c_1 must satisfy $c_1\mathbf{t}_1 = \tilde{\mathbf{u}}_1 - \bar{\mathbf{A}}\mathbf{z}_1 \bmod q$ for fixed $\tilde{\mathbf{u}}_1, \mathbf{z}_1, \mathbf{t}_1$. The overall probability is thus bounded by $Q_1Q_2/|\mathsf{Ch}|$ using the same argument as above. All in all, we have that $\Pr[\mathsf{bad}_{\mathsf{ord}}] \leq (1 + \sum_{i=1}^{N-1}(Q_i + Q_iQ_{i+1}))/|\mathsf{Ch}|$. Note that

$$\begin{aligned}\Pr[\mathsf{bad}_{\mathsf{zcol}}] + \Pr[\mathsf{bad}_{\mathsf{ord}}] &\leq \sum_{i=1}^{N} Q_i(Q_i+1)/(2|\mathsf{Ch}|) + \left(1 + \sum_{i=1}^{N-1}(Q_i + Q_iQ_{i+1})\right)/|\mathsf{Ch}| \\ &< \left(\sum_{i=1}^{N} Q_i + 1\right)^2 /|\mathsf{Ch}| = (Q_h + Q_s + 1)^2/|\mathsf{Ch}|.\end{aligned}$$

Putting all the bounds above together, we obtain the concrete bound in the theorem statement. □

Part II
History-Free SAS and Attack on Interactive Multi-Signature

C More Security Notions for Sequential Aggregate Signatures

In this section we discuss alternative flavors of security notions for SAS. Gentry et al. [GOR18] distinguish between the *full-history* (see Definition 2.3) and the *history-free* case. Additionally, we formalize yet another flavor as described by Chen and Zhao [CZ22], which we call *partial-signature history-free*.

C.1 Histrory-Free Sequential Aggregate Signatures

History-free sequential aggregate signatures (SAS') were first introduced by Brogle et al. [BGR12]. We recall now their syntax, together with the definition of security following Gentry et al. [GOR18]. Note that the only difference to the full-history setting is that now the sequential signing algorithm $\mathsf{SeqSign}$ doesn't take as input the list L_{i-1} of public keys and messages (i.e., the 'history') which have been used to compute the so-far signature σ_{i-1}. Consequently, the winning condition in the security game changes accordingly. In the full-history case, a valid forgery could use an already queried message, as long as the history changed. Now, a valid forgery must be on a non-queried message.

Definition C.1 (SAS'). *A* history-free sequential aggregate signature (SAS') *for a message space M consists of a tuple of* PPT *algorithms* $\mathsf{SAS}' = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{SeqSign}, \mathsf{SeqVerify})$ *defined as follows:*

$\mathsf{Setup}, \mathsf{Gen}$ *and* $\mathsf{SeqVerify}$ *as in Definition 2.1.*

$\mathsf{SeqSign}(\mathsf{sk}_i, m_i, \sigma_{i-1}) \to \sigma_i$*: On input a secret key* sk_i*, a message* $m_i \in M$*, and a so-far signature* σ_{i-1}*, the sequential signing algorithm outputs a new so-far signature* σ_i*.*

Definition C.2 (HF-UF-CMA Security). *A* SAS' *scheme satisfies* history-free unforgeabilty against chosen message attacks*, if for all* PPT *adversaries* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathsf{HF\text{-}UF\text{-}CMA}}_{\mathsf{SAS}'}(\mathcal{A}) := \Pr\left[\mathsf{HF\text{-}UF\text{-}CMA}_{\mathsf{SAS}'}(\mathcal{A}, \lambda) = 1\right] = \mathsf{negl}(\lambda),$$

where the $\mathsf{HF\text{-}UF\text{-}CMA}_{\mathsf{SAS}'}$ *game is described in Game 2.*

C.2 Partial-Signature History-Free Sequential Aggregate Signatures

As already mentioned before, our construction of Section 3 is inspired by the paper on sequential half-aggregation of Schnorr signatures by Chen and Zhao [CZ22]. In their work, they proposed yet another security model, without providing a formal definition (and name) for it. Their model is specifically tailored to Schnorr type signatures. In the following, we formalize the security model by trying to be as general as possible and hence making the notion useful also for other type of signatures.

Game 2: HF-UF-CMA$_{\mathsf{SAS}'}(\mathcal{A}, \lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{pp})$
$\mathcal{Q} := \varnothing$
$(L_N^*, \sigma_N^*) \leftarrow \mathcal{A}^{\mathsf{OSeqSign}}(\mathsf{pp}, \mathsf{pk})$
if $\mathsf{SeqVerify}(L_N^*, \sigma_N^*) \ \wedge \ \exists i^* \in [N]\colon (\mathsf{pk}_{i^*} = \mathsf{pk} \ \wedge \ m_{i^*} \notin \mathcal{Q})$ **then**
    return 1
else
    return 0

$\mathsf{OSign}(m_i, \sigma_{i-1})$
$\sigma_i \leftarrow \mathsf{SeqSign}(\mathsf{sk}, m_i, \sigma_{i-1})$
$\mathcal{Q} := \mathcal{Q} \cup \{m_i\}$
return σ_i

Recall that in both the history-free and the full-history case (Def. 2.3 and Def. C.2), the sequential aggregate signature takes the full description of the so-far signature σ_{i-1} as input to the SeqSign algorithm and then outputs the full description of the next so-far signature σ_i.

We now define a variant of the history-free case (i.e., no list L_{i-1} given to SeqSign) where the amount of information that needs to be send to the next signer is reduced, at the expense of introducing a new role, that we call the Combine algorithm. At each signing step, the signer forwards only a partial description of σ_{i-1} to the next signer, while at the same time sending some complementary description of σ_{i-1} to the combiner. Note that the complementary information can overlap with the partial description. At the end, the combiner takes the partial description of the final signature and all the complementary information they have received so far, to derive the full description of the final signature.

We call it the *partial-signature history-free* case. The winning condition for the adversary stays the same as in the history-free case.

Chen and Zhao [CZ22] actually consider a stronger notion where the winning condition is relaxed to allow for forgeries that are on already queried messages as long as the corresponding complementary part (in their case the response of the underlying Σ-protocol) is new. We suggest calling this the *strong* PS-HF-UF-CMA security.

Definition C.3 (SAS″). *A* partial-signature history-free sequential aggregate signature *scheme* (SAS″) *for a message space* M *consists of a tuple of* PPT *algorithms* $\mathsf{SAS}'' = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{SeqSign}, \mathsf{SeqVerify}, \mathsf{Combine})$ *defined as follows:*

$\mathsf{Setup}, \mathsf{Gen}$ *and* SeqVerify *as in Definition 2.1 and C.1.*

$\mathsf{SeqSign}(\mathsf{sk}_i, m_i, \mathsf{Part}(\sigma_{i-1})) \to (\mathsf{Part}(\sigma_i), \mathsf{Compl}(\sigma_i))$*: On input a secret key* sk_i*, a message* $m_i \in M$*, and a partial description* $\mathsf{Part}(\sigma_{i-1})$ *of the so-far signature* σ_{i-1}*, the sequential signing algorithm outputs a partial descrip-*

Game 3: strong PS-HF-UF-CMA$_{\mathsf{SAS}''}(\mathcal{A}, \lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{pp})$
$\mathcal{Q} := \varnothing$
$(L_N^*, \sigma_N^*) \leftarrow \mathcal{A}^{\mathsf{OSeqSign}}(\mathsf{pp}, \mathsf{pk})$
if $\mathsf{SeqVerify}(L_N^*, \sigma_N^*) \ \wedge \ \exists i^* \in [N]\colon (\mathsf{pk}_{i^*} = \mathsf{pk} \wedge (m_{i^*}, \mathsf{Compl}(\sigma_{i^*})) \notin \mathcal{Q})$ **then**
  return 1
else
  return 0

$\mathsf{OSign}(m_i, \mathsf{Part}(\sigma_{i-1}))$
$(\mathsf{Part}(\sigma_i), \mathsf{Compl}(\sigma_i)) \leftarrow \mathsf{SeqSign}(\mathsf{sk}, m_i, \mathsf{Part}(\sigma_{i-1}))$
$\mathcal{Q} := \mathcal{Q} \cup (\{m_i\}, \mathsf{Compl}(\sigma_{i^*}))$
return $(\mathsf{Part}(\sigma_i), \mathsf{Compl}(\sigma_i))$

tion $\mathsf{Part}(\sigma_i)$ *of a new so-far signature* σ_i *and some complementary information* $\mathsf{Compl}(\sigma_i)$.

$\mathsf{Combine}(\mathsf{Compl}(\sigma_1), \ldots, \mathsf{Compl}(\sigma_{N-1}), \mathsf{Part}(\sigma_N)) \to \sigma_N$.

Definition C.4 (PS-HF-UF-CMA Security). *A* SAS″ *scheme satisfies* partial-signature history-free unforgeabilty against chosen message attacks, *if for all* PPT *adversaries* $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{PS\text{-}HF\text{-}UF\text{-}CMA}}_{\mathsf{SAS}''}(\mathcal{A}) := \Pr\left[\mathsf{PS\text{-}HF\text{-}UF\text{-}CMA}_{\mathsf{SAS}''}(\mathcal{A}, \lambda) = 1\right] = \mathsf{negl}(\lambda),$$

where the PS-HF-UF-CMA$_{\mathsf{SAS}''}$ *game is described in Game 3. It satisfies strong* PS-HF-UF-CMA *security if the modifications in purple apply.*

The interest of this model lies in the fact that when only half-aggregation is possible (as it is the case in our construction and the one of [CZ22]), then a lot of information in every so-far signature is redundant and simply carried over to the next signing step. In our case, this corresponds to the list of responses $\mathbf{z}_1, \ldots, \mathbf{z}_i$ that is attached to every so-far signature σ_i. One possible instantiation of Part and Compl would be to set $\mathsf{Part}(\sigma_i) = (\tilde{\mathbf{u}}_i, \mathbf{z}_i)$ and $\mathsf{Compl}(\sigma_i) = \mathbf{z}_i$. By outsourcing this redundant information to the combiner, one safes significantly in bandwidth.

Algorithm 5 specifies how our FSwA-SAS from Section 3 can easily be modified to FSwA-SAS″ which fulfills PS-HF-UF-CMA security. The main modifications are that in SeqSign and SeqVerify the random oracle H now doesn't take the full history L_i as input, but only the current public key-message pair $(\mathbf{t}_i, m_i)$ together with an index i that fixes the position in the sequence. The combine algorithm simply puts together the relevant information in order to define σ_N.

It is straightforward to adapt the security reduction (as specified in Algorithm 4) to the PS-HF-UF-CMA setting. Again, one only has to index of the different tables $\mathrm{ZT}, \mathrm{HT}, \mathrm{MT}$ by replacing the full history L_i by the 'current' history $(\mathbf{t}_i, m_i, i)$. All bad flags are preserved and occur with the same probability.

Algorithm 5: FSwA-SAS″

The challenge space is $\mathsf{Ch} := \{c \in R : \|c\|_\infty = 1 \wedge \|c\|_1 = \kappa\}$ and the message space is $M' = \{0,1\}^l$. The random oracle is $\mathsf{H} : \{0,1\}^* \to \mathsf{Ch}$. The starting point is $i = 1$. Let $L_0 = \varnothing$ and $\sigma_0 = (\mathbf{0}, \mathbf{0})$. Setup and Gen are as in Algorithm 2.

$\mathsf{SeqSign}(\mathsf{sk}_i, m_i, \mathsf{Part}(\sigma_{i-1}))$

1: $(\tilde{\mathbf{u}}_{i-1}, \mathbf{z}_{i-1}) := \mathsf{Part}(\sigma_{i-1})$
2: $\mathbf{s}_i := \mathsf{sk}_i$
3: $\mathbf{t}_i := \bar{\mathbf{A}}\mathbf{s}_i \bmod q$
4: $\mathbf{z}_i := \perp$
5: **while** $\mathbf{z}_i := \perp$ **do**
6: $\quad \mathbf{y}_i \leftarrow \mathscr{D}^{\ell+k}$
7: $\quad \mathbf{u}_i := \bar{\mathbf{A}}\mathbf{y}_i \bmod q$
8: $\quad \tilde{\mathbf{u}}_i := \tilde{\mathbf{u}}_{i-1} + \mathbf{u}_i \bmod q$
9: $\quad c_i := \mathsf{H}(\tilde{\mathbf{u}}_i, \mathbf{t}_i, m_i, \mathbf{z}_{i-1}, i)$
10: $\quad \mathbf{z}_i := c_i \cdot \mathbf{s}_i + \mathbf{y}_i$
11: $\quad \mathbf{z}_i := \mathsf{RejSamp}(\mathbf{z}_i, c_i \cdot \mathbf{s}_i)$
12: $\mathsf{Part}(\sigma_i) := (\tilde{\mathbf{u}}_i, \mathbf{z}_i)$
13: $\mathsf{Compl}(\sigma_i) := \mathbf{z}_i$
14: **return** $\mathsf{Part}(\sigma_i), \mathsf{Compl}(\sigma_i)$

$\mathsf{SeqVerify}(L_N, \sigma_N)$

1: $(\mathbf{t}_1, m_1) || \ldots || (\mathbf{t}_N, m_N) := L_N$
2: $(\tilde{\mathbf{u}}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N) := \sigma_N$
3: $\mathbf{z}_0 := 1$
4: **if** $\exists i$ such that $\mathbf{t}_i$ has no invertible element **then**
5: $\quad$ **return** 0
6: **for** $i = N, \ldots, 1$ **do**
7: $\quad$ **if** $\|\mathbf{z}_i\|_2 > B$ **then**
8: $\quad\quad$ **return** 0
9: $\quad c_i := \mathsf{H}(\tilde{\mathbf{u}}_i, \mathbf{t}_i, m_i, \mathbf{z}_{i-1}, i)$
10: $\quad \mathbf{u}_i := \bar{\mathbf{A}}\mathbf{z}_i - c_i\mathbf{t}_i \bmod q$
11: $\quad \tilde{\mathbf{u}}_{i-1} := \tilde{\mathbf{u}}_i - \mathbf{u}_i \bmod q$
12: **if** $\tilde{\mathbf{u}}_1 = \mathbf{u}_1$ **then return** 1

$\mathsf{Combine}(\mathsf{Compl}(\sigma_1), \ldots, \mathsf{Compl}(\sigma_{N-1}), \mathsf{Part}(\sigma_N))$

1: $\mathbf{z}_i := \mathsf{Compl}(\sigma_i)$ for $i \in [N-1]$
2: $(\tilde{\mathbf{u}}_N, \mathbf{z}_N) := \mathsf{Part}(\sigma_N)$
3: $\sigma_N = (\tilde{\mathbf{u}}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N))$
4: **return** σ_N

Theorem C.5 (PS-HF-UF-CMA security). *If the signature scheme* FSwA-S *with message space $M = \{0,1\}^l$, as described in Algorithm 1, is* UF-CMA *secure, then is the sequential aggregate signature* FSwA-SAS″*, as described in Algorithm 5,* PS-HF-UF-CMA *secure.*

D Attack on [FH20]

In this section, we describe how to mount a (partial) secret-key recovery attack against the Dilithium-based multi-signature by Fukumitsu and Hasegawa [FH20], published in the proceedings of the PROVSEC conference from 2020. In order to be successful, the adversary only needs one valid and honestly generated signatures. The attack exploits the fact that every party of the multi-signature (including the adversary) obtains the full information of the first signature part $\mathbf{u}$,

which enables them to compute $c \cdot \mathbf{s}_2$, where $(\mathbf{s}_1, \mathbf{s}_2)$ is the secret key corresponding to the provided challenge public key. As it is easier to see the vulnerability in the single signature setting, we first describe an insecure variant of FSwA-S (specified in Algorithm 6) and then show how the vulnerability is carried over to the multi-signature from [FH20].

The main difference between the original (and secure) FSwA-S (Algorithm 1) and the modified (and insecure) version (Algorithm 6) is that we apply a trick due to Bai and Galbraith [BG14], which enables to compress the size of the signature. This technique is also used in Dilithium. The key idea is to compute $\mathbf{u} = \mathbf{A}\mathbf{y}$ (instead of $\mathbf{u} = \mathbf{A}\mathbf{y}_1 + \mathbf{y}_2$ as before) and to only use the high order bits $\mathsf{HighBits}(\mathbf{u})$ of $\mathbf{u}$ to derive the challenge c. The function HighBits takes a module element $\mathbf{r} = (r_1, \ldots, r_k)^t \in R^k$, decomposes each r_i into the higher-order part $r_{i,1}$ and lower-order part $r_{i,0}$, respectively, and extracts $(r_{1,1}, \ldots, r_{k,1})$ (see [LDK+20] for the formal specification). Subsequently, in the verification algorithm only the high order bits of $\mathbf{u}$ and $\mathbf{A}\mathbf{z} - c\mathbf{t}$ are compared to each other. This modification reduces the dimension of $\mathbf{z}$ from $\ell + k$ to ℓ, where $\mathbf{A} \in R_q^{k \times \ell}$.

In contrast to Dilithium and [BG14], our version contains the full commitment $\mathbf{u}$, and not the challenge c. Transmitting the challenge instead of the commitment is a well-known technique in the lattice setting to further shrink the size of the signature. As we see in the following, it is also crucial for security once the trick of by Bai and Galbraith [BG14] has been applied.

Lemma D.1. *Let* $\mathbf{A} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(\mathbf{A})$ *with* $\mathsf{sk} = (\mathbf{s}_1, \mathbf{s}_2)$ *as in Alg. 6. Given* $(\mathbf{u}, \mathbf{z}) = \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ *for the messages* $m \in M$ *together with the public key* $\mathsf{pk} = \mathbf{t}$*, a* PPT *adversary* $\mathcal{A}$ *can recover* $\mathbf{s}_2$*.*

Proof. The adversary re-constructs $c = \mathsf{H}(\mathsf{HighBits}(\mathbf{u}), \mathbf{t}, m)$ and computes

$$\mathbf{b} = \mathbf{A}\mathbf{z} - \mathbf{u} - c\mathbf{t} = \mathbf{A}\mathbf{y} + c\mathbf{A}\mathbf{s}_1 - \mathbf{A}\mathbf{y} - c\mathbf{A}\mathbf{s}_1 - c\mathbf{s}_2 = -c\mathbf{s}_2.$$

Note that the last equation gives $\mathbf{b} = -c\mathbf{s}_2 \bmod q$, but as there is no wrapping around modulo q (as both c and $\mathbf{s}_2$ are short elements) the equation also holds in R. The adversary now embeds the elements in the field $K = \mathbb{Q}[x]/(x^d + 1)$ and uses the fact that every non-zero element is invertible in K. Note that $0 \notin \mathsf{Ch}$. Hence, they multiply the result by c^{-1} (the K-inverse of c) and recover $-\mathbf{s}_2$. □

We can now easily move to the multi-signature in [FH20]. We don't re-state the full protocol of their scheme here, but simply refer to Figure 5 in [FH20]. For simplicity, we use our notations in the following and ignore the optimization to reduce the public key size in [FH20, Fig. 5]. The main issue is that every signer broadcasts untruncated $\mathbf{u}$ in the clear during the interactive signing process. We now explain how an adversary can during the common signing procedure recover half of the secret key of the single honest signer.

Lemma D.2. *Let* $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *be the multi-signature as defined in [FH20, Fig. 5]. Further, let* $\mathcal{A}$ *be a* PPT *adversary who is controlling all-but-one parties. We denote by* $(\mathsf{pk}^*, \mathsf{sk}^*)$ *the key pair of the honest signer, where* $\mathsf{sk}^* = (\mathbf{s}_1^*, \mathbf{s}_2^*)$*. After one successful multi-signature signing process,* $\mathcal{A}$ *can recover* $\mathbf{s}_2^*$*.*

Algorithm 6: Insecure FSwA-S

The challenge space is $\mathsf{Ch} := \{c \in R : \|c\|_\infty = 1 \wedge \|c\|_1 = \kappa\}$ and the message space is $M = \{0,1\}^l$. The random oracle is $\mathsf{H} : \{0,1\}^* \to \mathsf{Ch}$. Let $\mathscr{D}, B$ and $\mathsf{RejSamp}$ be as in Sec. 2.1.

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

1: $\mathbf{A} \xleftarrow{\$} R_q^{k \times \ell}$
2: **return** $\mathbf{A}$

$\mathsf{Gen}(\mathbf{A})$

1: $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{\$} S_\eta^\ell \times S_\eta^k$
2: $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \bmod q$
3: $\mathsf{sk} := (\mathbf{s}_1, \mathbf{s}_2)$
4: $\mathsf{pk} := \mathbf{t}$
5: **return** $(\mathsf{sk}, \mathsf{pk})$

$\mathsf{Ver}(\mathsf{pk}, \sigma, m)$

1: $(\mathbf{u}, \mathbf{z}) := \sigma$
2: $\mathbf{t} := \mathsf{pk}$
3: $c := \mathsf{H}(\mathsf{HighBits}(\mathbf{u}), \mathbf{t}, m)$
4: **if** $\|\mathbf{z}\|_\infty \leq B \wedge \mathsf{HighBits}(\mathbf{A}\mathbf{z} - c \cdot \mathbf{t}) = \mathsf{HighBits}(\mathbf{u})$ **then**
5: **return** 1
6: **else**
7: **return** 0

$\mathsf{Sign}(\mathsf{sk}, m)$

1: $(\mathbf{s}_1, \mathbf{s}_2) := \mathsf{sk}$
2: $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \bmod q$
3: $\mathbf{z} := \perp$
4: **while** $\mathbf{z} := \perp$ **do**
5: $\mathbf{y} \xleftarrow{\$} \mathscr{D}^\ell$
6: $\mathbf{u} := \mathbf{A}\mathbf{y} \bmod q$
7: $c := \mathsf{H}(\mathsf{HighBits}(\mathbf{u}), \mathbf{t}, m)$
8: $\mathbf{z} := c \cdot \mathbf{s}_1 + \mathbf{y}$
9: $\mathbf{z} := \mathsf{RejSamp}(\mathbf{z}, c \cdot \mathbf{s})$
10: $\sigma := (\mathbf{u}, \mathbf{z})$
11: **return** σ

Proof. Without loss of generality, let party 1 be the honest signer and party 2 to party N be the ones controlled by $\mathcal{A}$. Let $\mathbf{t}_i$ denote the public key of party $i \in [N]$. During the signing process, every party computes $\mathbf{u}_i := \mathbf{A}\mathbf{y}_i$ (denoted by $\mathbf{w}_v$ in the original protocol). At the second stage, they broadcasts $\mathbf{u}_i$ to all the co-signers. In particular, $\mathcal{A}$ receives $\mathbf{u}_1$ and computes $\mathbf{u} := \sum_{i \in [N]} \mathbf{u}_i$ (denoted by $\mathbf{w}$ in the original protocol). The multi-signature contains $\mathbf{z} := \sum_{i \in [N]} \mathbf{z}_i$, where $\mathbf{z}_i = \mathbf{y}_i + c_i \mathbf{s}_{i,1}$. The adversary re-constructs all challenges $c_i = \mathsf{H}(\mathsf{HighBits}(\mathbf{u}), \mathbf{t}_i, m)$ and computes

$$\mathbf{b} = \mathbf{A}\mathbf{z} - \mathbf{u} - \sum_{i \in [N]} c_i \mathbf{t}_i = \sum_{i \in [N]} \mathbf{A}\mathbf{y}_i + c_i \mathbf{A}\mathbf{s}_{i,1} - \mathbf{A}\mathbf{y}_i - c_i \mathbf{A}\mathbf{s}_{i,1} - c_i \mathbf{s}_{i,2} = - \sum_{i \in [N]} c_i \mathbf{s}_{i,2}.$$

As they know $\mathbf{s}_{i,2}$ for all $1 < i \leq N$, they can compute $\mathbf{b} + \sum_{i=2}^{N} c_i \mathbf{s}_{i,2}$ and recover $c_1 \mathbf{s}_{1,2}$. With the same reasoning as in the attack above, they can easily recover $\mathbf{s}_{1,2} = \mathbf{s}_2^*$. □

Remark D.3. In the simple signature setting (Algorithm 6) it is easy to fix this attack by only outputting $\sigma := (\mathsf{HighBits}(\mathbf{u}), \mathbf{z})$. However, in the multi-signature setting, this fix doesn't seem to apply in a trivial manner. The problem is that the function $\mathsf{HighBits}$ is not linear, and in general $\mathsf{HighBits}(\mathbf{u}_1) + \mathsf{HighBits}(\mathbf{u}_2) \neq \mathsf{HighBits}(\mathbf{u}_1 + \mathbf{u}_2)$.