**3.2 Security Proof**

We now demonstrate Algorithm 2's FH-UF-CMA security (as per Definition 2.9). We additionally outline the security of our plan in Appendix A using a novel model that we refer to as the *partial-signature history-free* security model, which was presented in [CZ22], for the sake of completeness.

**Theorem 3.4 (security of FH-UF-CMA).** Assume that the n is a power of two, q is prime, and Eq. 1 is satisfied for every k, ℓ, n, q, η, γ, l ∈ N. If s is uniformly sampled from Sℓ+kη and ¯A = [A|Ik] with A is uniformly sampled from Rk×ℓq, respectively, then let *p*inv be the likelihood that ¯As has a minimum of one invertible coefficient over Rq. If Algorithm 1's FSwA-S signature scheme having message space M = {0, 1}l is UF-CMA secure, then Algorithm 2's sequential aggregated signature FSwA-SAS, or FH-UF-CMA secure as well. In specific terms, for every adversary A against FH-UF-CMA security that queries the random oracle H at most Qh times, the OSeqSign oracle at most Qs times, and produces a forgery having a history of length N, there is an adversary B against UF-CMA security such that

Equation

*and* Time(B) = Time(A)+*O*((*N* +*Qh*)*kℓt*pmul)*, where t*pmul *is the time of polynomial multiplication in Rq.*

Proof. The high-level ideas of reduction B are first sketched. The entire description of B can be found in Alg 3. For a FH-UF-CMA game (resp. UF-CMA game), H and also OSeqSign (resp. H′ and OSign) stand for the random oracle as well as the signing oracle. After obtaining the challenge public key t∗ and the public parameter A, B verifies that t∗ ∈ Rk q comprises a minimum of one invertible element. In that case, B transfers (A, t∗) to A.

In response to an inquiry, OSeqSign asks OSign to sign a uniformly selected m and programs H to produce the value c that is returned by an external random oracle H′. Since a forgery filed by A may subsequently utilize the same mi, we are unable to simply pass mi to OSign in this case. Afterward, B loses the UF-CMA game since their submission of the forgery w.r.t. mi will be invalid.

The simulation of response to H queries is the core element of reduction. The forgery tuple's key list LN includes (ti,mi) so that ti = t∗. Afterward, in order for (m, (ui, zi)) to be considered a legitimate forgery within the UF-CMA game, B had to have extracted the matching ui and sent ui to H′ as well as a random message m. Importantly, zi−1 is utilized in this extraction process whenever (˜ui,Li, zi−1) is questioned to H. B can extract ui = ˜ui − ˜ui−1 by using zi−1 as an intuitive look-up key to retrieve the previous aggregated ˴ui−1.

To be more precise, we build a number of hybrid games from the initial FH-UF-CMA game in order to arrive at the one that is employed in that final reduction B. The probability of which Gi(A) stops with output 1 is indicated by Pr[Gi(A)].

G0 This game and the FH-UF-CMA game are identical. First, a blank key-value look-up table (HT) is initialized by the game. When the table's entry is not empty, a random oracle H returns HT[X] upon receiving the query with input X; If not, it returns c, assigns HT[X] := c, sample uniform c ∈ Ch. It states that AdvFH-UF-CMA FSwA-SAS (A) = Pr[G0(A)].

G1 This game is precisely the same as G0, with the exception that OSeqSign implements a RO table HT[˜ui,Li, zi−1]:= ci as soon that the rejection sampling phase is successful, and samples uniform ci ∈ Ch rather than calling ci = H(˜ui,Li, zi−1) afterwards ˜ui is computed; The game terminates by declaring baducol = true if HT[˜ui,Li, zi−1] has been set. According to this, Pr[baducol] ≤ | Pr[G0(A)] − Pr[G1(A)]|.

G2 With the exception of simulating the answers to random oracle questions H(˜ui,Li, zi−1), this game appears identical to G1. Create a new, empty ZT key-value lookup table. In the event that i = 1 or X := (˜ui−1,Li−1, zi−2) exists and ZT[X] = ¯Azi−1 mod q, extract ui := ˜ui − ˜ui−1, a sample uniform ci ∈ Ch, and then assign ZT[˜ui,Li, zi−1] := ui + citi. In the event that an entry X′ := (˜ui−1,Li−1, zi−2) already exists and ZT[X′] = ui + citi, the game terminates by setting badzcol = true. It holds that | Pr[G1(A)] − Pr[G2(A)]| ≤ Pr[badzcol].

G3 With the exception of OSeqSign and H, which progress as follows, this game resembles to G2. A key-value look-up table (MT) and an empty set M have been initialized by the game. Upon receiving a query, OSeqSign internally samples the uniform message m ∈ M then appends m to M. After extracting ui as previously stated, H sample a uniform message m ∈ M and aborts by declaring badmcol = true if m ∈ M whenever it receives a query with input (˜ui,Li, zi−1). . It holds

that | Pr[G2(A)] − Pr[G3(A)]| ≤ Pr[badmcol].

G4 When the adversary delivers an appropriate signature-history pair (LN, (˴uN, z1,..., zN)), this game is the same as G3, with the exception that it validates the ZT entries in the following ways. Let ˴uN−1,..., ˴u1 be the values obtained during the SeqVerify run. The game ends by setting badord = true if for some i ∈ [N] the entry for ZT[˜ui,Li, zi−1] is undefined. It holds

that | Pr[G3(A)] − Pr[G4(A)]| ≤ Pr[badord].

B Assuming that adversary A wins G4, a reduction B mentioned in Alg. 3 can be acquired in this way. After receiving an OSeqSign question, B queries the UF-CMA game's OSign with the uniformity message m ∈ M. B then retrieves ui and zi, programs HT using the challenge ci returned by the external arbitrary oracle H′(ui, t∗,m). If H is successful in extracting ui = ˜ui − ˵ui−1, it also receives new challenge ci for ti = t∗ through querying the outermost random oracle H′(ui, t∗,m). As A is assured of receiving an invertible challenging public key in B, A's perspective is the same as G4's.

We now demonstrate that, provided that none of the negative events occur, B is guaranteed to provide a message-signature pair (m, (ui∗, zi∗ )) that is accepted in the UF-CMA game; that is, ∥zi∗∥∞ ≤ B and ui∗ = ￣Azi∗ − ct∗ mod q, where c = H′(ui∗ , t∗,m). The former requirement is instantaneous based on SeqVerify's verification condition. To counter the latter, observe that c = H′(ui∗, t∗,m) = HT[˜ui∗,Li∗, zi∗−1] = ci∗ as soon that RO entries HT[˜u1,L1, z0],...,HT[˜uN,LN, zN−1] have been set in the correct sequence. As a result, ui∗ = ˜ui∗ − ˜ui∗−1 gets extracted through the invocation of H(˜ui∗,Li∗, zi∗−1). As it remains that badzcol = badord = false, the following lemma does, in fact, guarantee that such questions have been asked in the correct order.