1 Security

We construct a proof of security imitating the Banquet scheme. We first prove that adversary having access only to the public key can't forge a signature, except with negligible probability, i.e, we prove EUF-KO (Existential unforgeability-key only). Using this we prove EUF-CMA (Existential unforgeability-chosen message attack), were an adversary has access to a signing oracle.

Theorem 1.1 (Scheme is EUF-KO)**.** *Assuming that $\mathcal{F}$ is a one way function. Then for any adversasy $\mathcal{A}$ probabilistic running in poly(κ) time.*

Then there exists another prob poly(κ), adversary against the one wayness of $\mathcal{F}$ so that:

$$Adv_{\mathcal{A}}^{EUF-KO} \leq Adv_{\mathcal{B}}^{OWF} + \varepsilon(Q_c, Q_1, Q_2)$$

Where ε is a function that we will detail in the proof, Q_c, Q_1, Q_2 are the queries to the Commit oracle, H_1 and H_2.

Proof. $\mathcal{B}$ mantains tables ...

$\mathcal{B}$ receives a challenge y, which it forwards to $\mathcal{A}$. It then runs $\mathcal{A}$ normally, when $\mathcal{A}$ asks for the output of an oracle, $\mathcal{B}$ answers in the following way:

- H_c: It receives an input $q_c = \sigma_1, \mu, salt)$, then it chooses $x \stackrel{unif}{\leftarrow}$.

  In case, $x \in Bad$, $\mathcal{B}$ aborts. Otherwise, it adds x to Bad, adds (q_c, x) to Q_c and outputs x.

- H_1: Here $\mathcal{B}$ checks whether the query of $\mathcal{A}$ corresponds to a query already output by a previous query.

  In the affirmative case, $\mathcal{B}$ reconstructs the views of the parties. Otherwise, it does nothing.

- H_2: Same as in the case of H_c.

When $\mathcal{A}$ terminates, $\mathcal{B}$ checks T_in for the values of sk and checks that $\mathcal{F}(sk) = y$. If he does find one, $\mathcal{B}$ wins. Otherwise, it outputs $\perp$.

Now we observe:

$P[\mathcal{A}wins] = P[\mathcal{A}wins \wedge \mathcal{B}aborts] + P[\mathcal{A}wins \wedge \mathcal{B}outputs\perp] + P[\mathcal{A}wins \wedge \mathcal{B}outputswitness] + \leq P[\mathcal{B}aborts]$

So we only have to analyse: $P[\mathcal{A}wins \wedge \mathcal{B}outputs\perp]$. □

Theorem 1.2 (Scheme is EUF-CMA)**.** *Assuming that $\mathcal{F}$ is a one way function. Then the scheme is EUF-CMA.*

Proof. □

2 Choosing parameters

Again, we imitate the Banquet choice of parameters: Suppose an adversary is trying to attack the scheme, denote the cost of this attack by C. We want that $C > 2^{\kappa}$. The attacker must cheat either in the first challenge or the second challenge of every round. Since there are τ rounds, it must cheats in τ_1 challenges in the first round and τ_2 challenges in the second round, where $\tau_1 + \tau_2 = \tau$.

The probability of cheating τ_1 times in the first challenge is

$$P_1 = \sum_{k=\tau_1}^{\tau} PMF\left(k, \tau, \frac{2}{q^n - 3}\right)$$

$$PMF\left(k, \tau, p\right) = \binom{\tau}{k} p^k (1-p)^{\tau - k}$$

The probability of cheating τ_2 times in the second challenge is

$$P_2 = N^{-\tau_2}$$

Then $C = 1/P_1 + 1/P_2$.

To sum up, the attack wants to find τ_1, τ_2 so that C is maximum. To do this we simply do a brute force of the parameters. We obtain the following results:

N	τ	τ_1	τ_2	security level	signature size
8	43	0	32	128.0	6.45KB
16	32	0	26	128.0	5.31KB
32	26	0	26	130.0	4.73KB
64	22	0	22	132.0	4.36KB
128	19	0	19	133.0	4.07KB
1024	13	0	13	130.0	3.41KB
65536	8	0	8	128.0	2.8KB

There is a good explantion for why the best attack is to always try to guess the second challenge: As discussed above the soundness of the Schwart-Zippel lemma is bounded above by:

$$\frac{2}{2^{144} - 3}$$

So, the probability of guessing at least 1 of τ challenges is at most: $1 - (1-p)^{\tau}$. So for example in the case of $N = 128, \tau = 19$ we have: $1 - (1-p)^{\tau} = 2^{-138.75} < 2^{-\kappa}$.

Algorithm 1 Sign(sk, msg)

Phase 1: Committing to the seeds, the execution views and interpolated polynomials of the parties.

Sample a random salt: salt $\xleftarrow{\$} \{0,1\}^{2\kappa}$.
for each parallel repetition e **do**
  Sample a random master seed sd_e.
  Derive $seed_e^{(i)}$ from sd_e using a merkle tree and give it to party i.
  Commit to seed: $com_e^{(i)} \leftarrow Commit(salt, e, i, seed_e^{(i)})$
  Expand random tape: $tape_e^{(i)} \leftarrow ExpandTape(salt, e, i, seed_e^{(i)})$
  Sample witness shares: $sk_e^{(i)} \leftarrow Sample(tape_e^{(i)})$
  Compute witness and outupt offsets: $\Delta sk_e \leftarrow sk - \sum_i sk_e^{(i)}$;
  Update shares from Party 1: $sk_e^{(1)} \leftarrow sk_e^{(1)} + \Delta sk_e$
  Set $pk_e^{(1)} \leftarrow pk$ and $pk_e^{(i)} \leftarrow 0, \forall i \in [n] \backslash \{0\}$.
  for each party i **do**: We compute the value of the checking polynomials
    Define $U_e^{(i)}(0) = sk_e^{(i)}$ and $V_e^{(i)}(0) = pk_e^{(i)}$ as elements of $\mathbb{F}_p$.
    Sample $U_e^{(i)}(1)$ from $tape_e^{(i)}$.
    Parties compute $U_e^{(i)}$ and $V_e^{(i)}$ (which have degree 1 and 0).
  end for
  Prover defines $P_e = V_e - F(U_e)$. (Degree 2).
  Prover computes $\Delta P(1)$ and $\Delta P(2)$.
end for
Define $\sigma_1 := ((com_e^{(i)})_{0 \le i \le N}, \Delta P_e(1), \Delta P_e(2))_{0 \le e \le \tau}$

Phase 2: Challenging the checking polynomials.

Compute challenge hash: $h_1 = H(\sigma_1, \mu, salt)$ (μ is the message we want to sign).
Derive R_e from h_1 for every $e \in [\tau]$.

Phase 3: Commiting to the answer of the challenge.

for each parallel repetition e and every party i: **do**
  Party i computes locally: $U_e^{(i)}(R_e), V_e^{(i)}(R_e)$ and $P_e^{(i)}(R_e)$.
end for
Prover commits to $\sigma_2 = (U_e^{(i)}(R_e), V_e^{(i)}(R_e),\ P_e^{(i)}(R_e))$.

Phase 4: Challenging the execution of the protocol.

Compute challenge hash: $h_2 = H(\sigma_2, h_1)$
for each parallel repetition e **do**
  Verifier derives $\bar{i}_e \leftarrow [N]$ from h_2.
end for

Phase 5: Prover reveals the views of N-1 parties.

Prover gets $seeds$ = seeds necesary to reveal $\{seed_{e,i} : i \neq \bar{i}_e; 1 \le e \le M\}$.
Prover outupts: $\left(salt, h_1, h_2, seeds, (com_e^{(\bar{i}_e)}, \Delta sk_e, (\Delta P_e(k))_{k=1,2}, U_e(R_e))_{0 \le e \le \tau}\right)$

Algorithm 2 Verify(pk, σ, msg)

Computation phase

Parse $\sigma \leftarrow \left(salt, h_1, h_2, seeds, (com_e^{(\bar{i}_e)}, \Delta sk_e, (\Delta P_e(k))_{k=1,2}, U_e(R_e))_{0\le e\le\tau}\right)$

Set $\sigma_1 \leftarrow ((com_e^{(i)})_{0\le i\le N}, \Delta P_e(1), \Delta P_e(2))_{0\le e\le\tau}$

for each execution e **do**

  Derive R_e from h_1. Derive i_e from h_2.

  Recompute $\{\tilde{com}_e^{(i)}, U_e^{(i)}(R_e), P_e^{(i)}(R_e)\}_{i\neq\bar{i}_e}$ from σ.

  Recompute $U_e^{(\bar{i}_e)}(R_e) = U_e(R_e) - \sum_{i\neq\bar{i}_e} U_e^{(i)}(R_e)$

  Recompute $P_e^{(\bar{i}_e)}(R_e) = P_e(R_e) - \sum_{i\neq\bar{i}_e} P_e^{(i)}(R_e)$

  Compute $P(R_e) = V_e(R_e) - F \circ U_e(R_e)$.

end for

Set $\sigma_2 \leftarrow (U_e(R_e), V_e(R_e),\ P_e(R_e))_{0\le e\le\tau}$.

Compute $h_1^{(?)} \leftarrow H(\sigma_1, \mu, salt)$, using $com_e^{(\bar{i}_e)}$.

Compute $h_2^{(?)} \leftarrow H(\sigma_2, h_1)$.

Check $h_1 = h?_1$ and $h_2 = h?_2$.

3 Communication

We can upper bound the communication by the following formula:

$$4\kappa + \tau(\kappa\lceil\log N\rceil + 2\kappa + \lambda(2m+n)\log q)$$

Then best value for λ is 1 actually, because the witness is an element of $\mathbb{F}_p$ where $p > 2^{144}$, depending on the choice of parameters. Other choices are $p = 4^{80}, p = 8^{64}, p = 64^{51}$. But these have clearly higher signature size.

k