Tight and Optimal Reductions for Signatures based on Average Trapdoor Preimage Sampleable Functions and Applications to Code-Based Signatures *

André Chailloux[2] and Thomas Debris-Alazard[1,2]

[1] Information Security Group, Royal Holloway, University of London
[2] Inria de Paris, EPI COSMIQ
andre.chailloux@inria.fr,
thomas.debris@rhul.ac.uk

Abstract. The GPV construction [18] presents a generic construction of signature schemes in the Hash and Sign paradigm. This construction requires a family $\mathcal{F}$ of trapdoor preimage sampleable functions (TPSF). In this work we extend this notion to the weaker Average TPSF (ATPSF) and show that the GPV construction also holds for ATPSF in the Random Oracle Model. We also introduce the problem of finding a Claw with a random function (Claw(RF)) and present a tight security reduction to the Claw(RF) problem. Our reduction is also optimal meaning that an algorithm that solves the Claw(RF) problem breaks the scheme. We extend these results to the quantum setting and prove this same tight and optimal reduction in the QROM. Finally, we apply these results to code-based signatures, notably the Wave signature scheme and prove tight and optimal reductions for it in the ROM and the QROM improving and extending the original analysis of [12].

1 Introduction

Signature schemes are an important element of many cryptographic applications and are one of the schemes standardized by the post-quantum NIST competition [23]. Assessing the exact security (and therefore efficiency) of these schemes is therefore a very important task, both against classical and quantum computers. The GPV construction [18] presents a generic construction of signature schemes in the Hash and Sign paradigm. This construction requires a family $\mathcal{F}$ of trapdoor preimage sampleable functions (TPSF), which informally is a collection of functions that are hard to invert but which can be easily inverted with some trapdoor. There are two specific properties of their construction:

1. The inversion algorithm that uses the trapdoor should have good repartition properties for each image y. This means that for each image y, the inversion algorithm should output a preimage x according to a certain distribution.
2. The security of the resulting signature scheme is tightly based on the collision resistance of the family $\mathcal{F}$ and not on the one-wayness.

These properties were tailored for lattice based schemes where these 2 properties hold. For example, the lattice-based FALCON signature scheme [17] is based on the GPV construction. Notice that it is also possible to base the security on one-wayness but the security reduction is not tight.

In this paper, we extend the notion of TPSF where the property (1) above should hold only on average for y, defining the notion of Average TPSF (ATPSF). A direct use of the leftover hash lemma shows that we can go from an ATPSF to a TPSF with a quadratic loss in the security of $\mathcal{F}$. What we show is the following:

– We show that this quadratic loss is not necessary and that we can use ATPSF instead of TPSF without any loss.

* The work of Debris-Alazard was supported by the grant EPSRC EP/S02087X/1.

- Applying the GPV construction of signature schemes from a family $\mathcal{F}$ of ATPSF, we show that the security of the signature scheme ie equivalent to solving the Claw with Random Function (Claw(RF)) problem for $\mathcal{F}$. Informally, in the Claw(RF) problem, we are given a random function $\mathcal{H}$ and a random f from $\mathcal{F}$ and we want to find (x, y) such that $f(x) = \mathcal{H}(y)$.
- We extend this to the quantum setting and show that our tight and optimal results also hold in the QROM.
- We apply these results to the Wave signature scheme [12] and show more formally its classical and quantum security.

One of the implications of these results is that:

$$\text{Collision} \preccurlyeq \begin{array}{c}\text{Claw(RF)}\\ \Updownarrow \\ \text{Signature}\end{array} \preccurlyeq \text{One way}.$$

This means that the collision problem is easier than the Claw(RF) problem which itself is easier than the preimage problem. Moreover, attacking the signature scheme is equivalently hard to solving the Claw(RF) problem in the ROM. In the case of lattices, we have:

$$\textit{Collision} \approx \text{SIS}^3 \preccurlyeq \text{ Signature } \preccurlyeq \text{One way} = \text{ISIS}^4.$$

In the context of code-based cryptography, things are very different. In many regimes used for signatures, the collision problem is actually easy to solve. Therefore, we can only use the non-tight reduction to one wayness. From there, there are two possibilities: (1) lose the factor associated to non-tightness and have a big loss in parameters or (2) ignore the non-tightness and assume it won't have a practical impact. Solution (2) is of course very risky as the security proof of the actual scheme becomes incomplete. On the other hand, those that decide on (1) have a loss in parameters which could be unnecessary.

What we also advocate through our result is that if we want to study the concrete security of these signature schemes in the ROM, the Claw(RF) problem is the actual problem we should be looking at. Others that would want to construct a family of ATPSF for which the collision problem is easier than the preimage problem can use our results to study the real security of their schemes. Because we also prove this optimal security in the quantum ROM, our result is especially adapted for post-quantum cryptography.

In order to prove our results, we use fairly standard techniques in the ROM based on reprogramming the hashing and signing oracles. In order to do this formally, we need to keep track of the internal memory of the reprogrammed oracles - which is often a problem that is discarded in game based reductions - and not only look at their output distributions. We take the approach of constructing more explicitly an algorithm for the Claw(RF) problem from an attacker that attacks the signature scheme instead of using the game formalism, even though we strongly inspire ourselves from this formalism. An interesting aspect of our proof is that we manage to reprogram only the signing oracle, and not the hashing oracle, which reduces the requirement on the family of ATPSF functions.

For the quantum case, our proofs mainly use a result by Zhandry [29] on the indistinguishability of close quantum oracles. Here, we need to reprogram the hash function as well since we cannot work on the internal memory of the quantum oracles.

2 Preliminaries

Probabilistic Notation. Let $\mathcal{D}$ be a distribution, and X be a random variable. The notation $X \xleftarrow{\$} \mathcal{D}$ denotes that X is distributed according to $\mathcal{D}$. Furthermore, for a set S, we will denote by $\mathcal{U}(S)$ the uniform distribution over S. We use the same notations for picking elements: $y \xleftarrow{\$} \mathcal{D}$ means that y is picked according to $\mathcal{D}$ while $y \xleftarrow{\$} S$ denotes that y is uniformly distributed over S.

[3] SIS: Short Integer Solution problem commonly used in lattice-based cryptography
[4] ISIS: Inhomogeneous Short Integer Solution

Sometimes when we wish to emphasize on which probability space the probabilities or the expectations are taken, we note on the right of a symbol ":" the random variable specifying the associated probability space over which the probabilities or expectations are taken. For instance the probability $\mathbb{P}(\mathcal{E} : X)$ of the event $\mathcal{E}$ is taken over the random variable X.

The statistical distance between two discrete probability distributions $\mathcal{D}_1, \mathcal{D}_2$ over a same space $\mathcal{E}$ is defined as:

$$\Delta(\mathcal{D}_0, \mathcal{D}_1) \stackrel{\triangle}{=} \frac{1}{2} \sum_{x \in \mathcal{E}} |\mathcal{D}_0(x) - \mathcal{D}_1(x)|.$$

The statistical distance Δ satisfies the triangle inequality.

A function $f(\lambda)$ is said to be negligible, and we denote this by $f \in \text{negl}(\lambda)$, if for all polynomials $p(\lambda)$, $|f(\lambda)| < p(\lambda)^{-1}$ for all sufficiently large λ.

For any 2 sets D, R, we denote by $\mathfrak{F}_R^D$ the set of functions from D to R.

Query Algorithms and Oracles. For any algorithm $\mathcal{A}$, we denote by $|\mathcal{A}|$ it's total running time. We will also consider query algorithms $\mathcal{A}^{\mathcal{O}}$ that will make a certain amount of calls to an oracle $\mathcal{O}$. For us, an oracle $\mathcal{O}$ will be a deterministic or probabilistic function for which we have only a black box access. When we write $\mathcal{A}^{\mathcal{O}}$, it will mean that the oracle is non specified and we can replace $\mathcal{O}$ with any oracle.

For a query algorithm $\mathcal{A}^{\mathcal{O}}$, we write $|\mathcal{A}^{\mathcal{O}}| = (t, q)$ indicating that its running time is t and that it performs q queries to $\mathcal{O}$. An algorithm can also query different oracles, which we indicate as $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2}$ and $|\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2}| = (t, q_1, q_2)$ indicates that it runs in time and perform q_1 queries to $\mathcal{O}_1$ and q_2 queries to $\mathcal{O}_2$.

For any (deterministic or probabilistic) function f, we denote by $\mathcal{O}_f$ its associated oracle, and we will write it:

$\underline{\mathcal{O}_f(x)}$
`return` $f(x)$.

An important concept in this paper will be the concept of oracles with internal memory. We will denote by $\mathcal{O}(x; \mathcal{L})$ a query x to oracle $\mathcal{O}$ which has internal memory $\mathcal{L}$. If the result to this query is y and the internal memory is changed to $\mathcal{L}'$, we will write `return` $(y; \mathcal{L}')$. This internal memory is never revealed.

One oracle of interest will be the random oracle. It mimics a uniformly chosen random function from $\mathfrak{F}_R^D$. We will denote this oracle $\mathcal{O}_{RO}$ (the sets D and R are implicit).

$\underline{\mathcal{O}_{RO}(x; \mathcal{L})}$
`if` $\exists! y : (x, y) \in \mathcal{L}$, `return` $(y; \mathcal{L})$
`otherwise, pick` $y \stackrel{\$}{\hookleftarrow} R_\lambda$, `return` $(y; \mathcal{L} \cup \{(x, y)\})$

This oracle mimics a call to a random function. Each time x is queried, a random image y is constructed. If the same x is called afterwards, the same output y should be given. Therefore, we have a list $\mathcal{L}$ that stores values (x, y) already specified by the function. If $\mathcal{L}$ is initialized with $\emptyset$, we should never have $x, y, y' \neq y$ such that $(x, y) \in \mathcal{L}$ and $(x, y') \in \mathcal{L}$. For any algorithm $\mathcal{A}^{\mathcal{O}}$, we have:

$$\mathbb{P}\left(\mathcal{A}^{\mathcal{O}_g} \text{ outputs } 0 \mid g \stackrel{\$}{\hookleftarrow} \mathfrak{F}_R^D\right) = \mathbb{P}\left(\mathcal{A}^{\mathcal{O}_{RO}} \text{ outputs } 0 \mid \mathcal{O}_{RO} \text{ is initialized with } \mathcal{L} = \emptyset\right).$$

Another important aspect of query algorithms is that if we consider an algorithm $\mathcal{A}^{\mathcal{O}}$ and 2 close oracles $\mathcal{O}_1, \mathcal{O}_2$, $\mathcal{A}_{\mathcal{O}_1}$ and $\mathcal{A}_{\mathcal{O}_2}$ will be close. This is at the core of the game formalism presented in [27]. More formally

Proposition 1. *Let $\mathcal{A}^{\mathcal{O}}$ be a query algorithm with $|\mathcal{A}^{\mathcal{O}}| = (t, q)$. Let $\mathcal{O}_1, \mathcal{O}_2$ be 2 oracles such that:*

$$\forall x, \mathcal{L}, \quad \Delta(\mathcal{O}_1(x; \mathcal{L}), \mathcal{O}_2(x; \mathcal{L})) \leq \delta.$$

Then we have:

$$\mathbb{P}\left(\mathcal{A}^{\mathcal{O}_1} \textit{ outputs } 0\right) - \mathbb{P}\left(\mathcal{A}^{\mathcal{O}_2} \textit{ outputs } 0\right) \leq q\delta.$$

3 Digital Signatures and EUF-CMA Security Model in a Classical/Quantum Setting

A signature scheme S consists of three algorithms $(\text{S.KEYGEN}, \text{S.SIGN}, \text{S.VERIFY})$:

- $\text{S.KEYGEN}(1^\lambda) \to (\text{pk}, \text{sk})$ is the generation of the public key pk and the secret key sk from the security parameter λ.
- $\text{S.SIGN}(m, \text{pk}, \text{sk}) \to \sigma_m$: generates the signature σ_m of a message m from m, pk, sk.
- $\text{S.VERIFY}(m, \sigma, \text{pk}) \to \{0, 1\}$ verifies that σ is a valid signature of m using m, σ, pk. The output 1 corresponds to a valid signature.

Correctness. A signature scheme is defined as correct if when we sample $(\text{pk}, \text{sk}) \leftarrow \text{S.KEYGEN}(1^\lambda)$, we have for each m:

$$\text{S.VERIFY}(m, \text{S.SIGN}(m, \text{pk}, \text{sk}), \text{pk}) = 1.$$

Security definitions We consider the EUF-CMA (Existential Universal Forgery for Chosen Message Attack) security for signature schemes. A key pair $(\text{pk}, \text{sk}) \leftarrow \text{S.KEYGEN}(1^\lambda)$ is generated. The goal of the adversary $\mathcal{A}$ is, knowing only pk, to construct a pair (m, σ_m) such that σ_m is a valid signature for m but we give him some additional power. He can query a signing oracle $\mathcal{O}_{\mathsf{Sign}}$, that does the following:

```
proc Sign(m)
σ_m ← S.SIGN(m, pk, sk)
return σ_m
```

Notice here that the signing oracle has access to pk and sk. The goal of the adversary is then in this case to output a valid signature σ_{m^*} for a message m^* *that has not been queried* to the signing oracle.

Definition 1. *Let $\mathcal{A}^{\mathcal{O}}$ be a query algorithm, we define*

$$Adv_{\mathcal{S}}^{\text{EUF-CMA}}(\mathcal{A}^{\mathcal{O}}) = \mathbb{P}\Big(\text{S.VERIFY}(m^*, \sigma^*, \text{pk}) = 1 \text{ AND } m^* \textit{ has not been queried in } \mathcal{O}_{\mathsf{Sign}} : (\text{pk}, \text{sk}) \leftarrow \text{S.KEYGEN}(1^\lambda), (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\text{pk})\Big).$$

For any time t and number of queries q_{sign}, we define:

$$Adv_{\mathcal{S}}^{\text{EUF-CMA}}(t, q_{sign}) = \max_{\mathcal{A}^{\mathcal{O}} : |\mathcal{A}^{\mathcal{O}}| = (t, q_{sign})} Adv_{\mathcal{S}}^{\text{EUF-CMA}}(\mathcal{A}^{\mathcal{O}}).$$

For a quantum adversary, we define similarly the quantum EUF-CMA advantage as:

$$QAdv_{\mathcal{S}}^{\text{EUF-CMA}}(t, q_{sign}) = \max_{\mathcal{A}^{\mathcal{O}} : |\mathcal{A}^{\mathcal{O}}| = (t, q_{sign})} Adv_{\mathcal{S}}^{\text{EUF-CMA}}(\mathcal{A}^{\mathcal{O}}).$$

where the maximum is over quantum query algorithms that perform classical queries.

It is actually standard, even if the algorithm is quantum, to consider classical queries to the signing oracle. This is because in the real life scenario that motivates this security definition, signing queries are done to an external party that can force you to perform classical queries.

Remark 1. We could also require that the EUF-CMA condition holds for almost all key pairs (pk, sk) and not only on average. We can also get this stronger notion, and we discuss this in Section 6.2.

4 Family of ATPSF

In this work we will use the Full Domain Hash (FDH) paradigm of signature schemes [4, 9]. The key ingredient of this kind of constructions is a trapdoor one-way function $f : D \to R$ and a cryptographic hash function $\mathcal{H}$. The corresponding FDH scheme to sign a message m uses the trapdoor to choose a signature $x \in f^{-1}(\mathcal{H}(m))$. The verification step simply consists in computing $\mathcal{H}(m)$ and $f(x)$ to ensure that $f(x) = \mathcal{H}(m)$. The difficulty for designing such primitives is the fact that each time a message is signed, the signature is made public while the secret trapdoor has been used to produce it. Therefore, we must ensure that no information of the trapdoor leaks after the inversion. However, in the nice case where f is a permutation this does not matter. Indeed, the hash of the message $\mathcal{H}(m)$ is classically considered as random and thus the inverse $x = f^{-1}(\mathcal{H}(m))$ will be random too and in this way distributed independently of the trapdoor. This is typically the case for signatures schemes like RSA. Nevertheless, building one-way permutations in the post-quantum world like in code/lattice-based cryptography is a hard condition to meet. Usually [18, 13] functions are many-to-one and then it is non-trivial to build trapdoor candidates with an inversion algorithm which is oblivious to the used trapdoor. Building a secure FDH signature in this situation can be achieved by imposing additional properties [18] to the one-way function. This is mostly captured by the notion of Trapdoor Preimage Sampleable Functions (TPSF) [18, Definition 5.3.1]. We express below this concept in a slightly relaxed way dropping the domain sampleability condition and only assuming that the preimage sampleable property holds *on average* and not for any possible element in the function range. This will be sufficient for proving the security of the associated FDH scheme.

Definition 2. *An ε-*ATPSF *(for Average Trapdoor Preimage Sampleable Functions (or Function Family)) is an efficient triplet of probabilistic algorithms (*$\textsc{TrapGen}$, $\textsc{SampDom}$, $\textsc{SampPre}$*) where:*

- $\textsc{TrapGen}(1^\lambda) \to (f, \tau)$. *Takes the security parameter λ and outputs $f : D_\lambda \to R_\lambda$, an efficiently computable function with an efficient description, and τ, the trapdoor that will allow to invert f.*
- $\textsc{SampDom}(f) \to x$. *Takes a function $f : D_\lambda \to R_\lambda$ (with an efficient description) as an input and outputs some $x \in D_\lambda$.*
- $\textsc{SampPre}(f, \tau, y) \to x$. *Takes a function f with associated trapdoor τ, an element $y \in R_\lambda$ and outputs $x \in D_\lambda$ s.t. $f(x) = y$.*

We define

$$\varepsilon_{f,\tau} \stackrel{\triangle}{=} \Delta\Big(\textsc{SampDom}(f), \textsc{SampPre}(f, \tau, U(R_\lambda))\Big),$$

where $\textsc{SampPre}(f, \tau, U(R_\lambda))$ *is sampled as follows: pick $y \stackrel{\$}{\hookleftarrow} R_\lambda$, return* $\textsc{SampPre}(f, \tau, y)$. *We require that our triplet of algorithms satisfies*

$$\mathbb{E}_{(f,\tau)\leftarrow\textsc{TrapGen}(1^\lambda)}\left(\varepsilon_{f,\tau}\right) \leq \varepsilon. \tag{1}$$

The main difference with the definition of TPSF as defined [18, Definition 5.3.1] is that we consider an average $y \stackrel{\$}{\hookleftarrow} R_\lambda$ instead of wanting the property for almost all y. Furthermore, it is also asked for TPSF to verify that $\mathbb{E}_{(f,t)\leftarrow\textsc{TrapGen}(1^\lambda)}\left(\Delta(f(\textsc{SampDom}(f)), U(R_\lambda))\right) \leq \varepsilon'$ (domain sampleability condition) for some ε' whereas we, a priori, don't request anything of this kind for ATPSF.

As we will show now in the following propositions, (i) ε-ATPSF family verifies the domain sampleability condition of [18] with ε and (ii) Equation 1 holds for almost all y but with a loss of a square factor.

Proposition 2. *Let $\mathcal{F} = (\textsc{TrapGen}, \textsc{SampDom}, \textsc{SampPre})$ be a collection of ε-ATPSF. We have for any f, τ*

$$\Delta(f(\textsc{SampDom}(f)), U(R_\lambda)) \leq \varepsilon_{f,\tau}$$

where for a fixed f, $f(\textsc{SampDom}(f))$ is the distribution which is sampled as follows: $x \leftarrow \textsc{SampDom}(f)$, return $f(x)$. Furthermore,

$$\mathbb{E}_{(f,t)\leftarrow \textsc{TrapGen}(1^\lambda)} \left[\Delta(f(\textsc{SampDom}(f)), U(R_\lambda))\right] \leq \varepsilon$$

Proof. We write

$$\begin{aligned}
\varepsilon_{f,\tau} &= \Delta\Big(\textsc{SampDom}(f), \textsc{SampPre}(f, \tau, U(R_\lambda))\Big) \\
&\geq \Delta\Big(f(\textsc{SampDom}(f)), f(\textsc{SampPre}(f, \tau, U(R_\lambda)))\Big) \\
&= \Delta(f(\textsc{SampDom}(f)), U(R_\lambda))
\end{aligned}$$

where the first inequality uses the fact that for any deterministic function f and random variables X and Y (see [19] for a proof), $\Delta(f(X), f(Y)) \leq \Delta(X, Y)$. We conclude the proof by using the definition of ε. □

Proposition 3 ([1]). *Let $\mathcal{F} = (\textsc{TrapGen}, \textsc{SampDom}, \textsc{SampPre})$ be an ε-ATPSF and for (f, τ) output by $\textsc{TrapGen}(1^\lambda)$. Then we have:*

$$2\varepsilon_{f,\tau} \geq \frac{1}{|R_\lambda|} \sum_{y \in R_\lambda} \Delta\left(\textsc{SampPre}(f, y), X_y\right)$$

where X_y denotes the distribution of $X_y \overset{\$}{\hookleftarrow} \textsc{SampDom}(f)$ given $f(X_y) = y$, meaning

$$\forall x \in D_\lambda, \quad \mathbb{P}\left(X_y = x\right) \overset{\triangle}{=} \mathbb{P}\left(\textsc{SampDom}(f) = x \mid f(\textsc{SampDom}(f)) = y\right). \tag{2}$$

From there,

$$\#\left\{y \in R_\lambda \ : \ \Delta\left(\textsc{SampPre}(f, y), X_y\right) > \sqrt{\varepsilon_{f,\tau}}\right\} \leq 2\sqrt{\varepsilon_{f,\tau}},$$

Proof. Let us denote for all $y \in R_\lambda$,

$$p_y \overset{\triangle}{=} \mathbb{P}\left(\textsc{SampDom}(f) = x \mid f(\textsc{SampDom}(f)) = y\right).$$

We have the following computation,

$$\begin{aligned}
&\Delta(\textsc{SampPre}(f, t, \mathcal{U}(R_\lambda)), \textsc{SampDom}(f)) \\
&= \frac{1}{2} \sum_y \sum_{x \in f^{-1}(y)} \left| \mathbb{P}\left(\textsc{SampDom}(f) = x\right) - \frac{1}{|R_\lambda|} \mathbb{P}(\textsc{SampPre}(f, \tau, y) = x) \right| \\
&= \frac{1}{2} \sum_y \sum_{x \in f^{-1}(y)} \left| \mathbb{P}\left(\textsc{SampDom}(f) = x\right) - \frac{p_y}{|R_\lambda|} + \frac{p_y}{|R_\lambda|} - \frac{1}{|R_\lambda|} \mathbb{P}(\textsc{SampPre}(f, \tau, y) = x) \right| \\
&\geq \frac{1}{2} \sum_y \frac{1}{|R_\lambda|} \sum_{x \in f^{-1}(y)} |p_y - \mathbb{P}(\textsc{SampPre}(f, \tau, y) = x)| - \frac{1}{2} \sum_y \sum_{x \in f^{-1}(y)} \left| \frac{\mathbb{P}\left(\textsc{SampDom}(f) = x\right)}{p_y} - \frac{1}{|R_\lambda|} \right| \\
&= \sum_{y \in R_\lambda} \frac{1}{|R_\lambda|} \Delta\left(\textsc{SampPre}(f, \tau, y), X_y\right) - \frac{1}{2} \sum_y \sum_{x \in f^{-1}(y)} \left| \frac{\mathbb{P}\left(\textsc{SampDom}(f) = x\right)}{p_y} - \frac{1}{|R_\lambda|} \right|
\end{aligned} \tag{3}$$

Now we have for all $x \in f^{-1}(y)$ and by definition of p_y,

$$\begin{aligned}\frac{\mathbb{P}\left(\textsc{SampDom}(f)=x\right)}{p_y} &= \frac{\mathbb{P}\left(\textsc{SampDom}(f)=x\right)}{\mathbb{P}\left(\textsc{SampDom}(f)=x \mid f(\textsc{SampDom}(f))=y\right)} \\ &= \frac{\mathbb{P}\left(\textsc{SampDom}(f)=x\right)}{\mathbb{P}\left(f(\textsc{SampDom}(f))=y \mid \textsc{SampDom}(f)=x\right)\frac{\mathbb{P}(\textsc{SampDom}(f)=x)}{\mathbb{P}(f(\textsc{SampDom}(f))=y)}} \\ &= \frac{\mathbb{P}(f(\textsc{SampDom}(f))=y)}{\mathbb{P}\left(f(\textsc{SampDom}(f))=y \mid \textsc{SampDom}(f)=x\right)} \\ &= \mathbb{P}(f(\textsc{SampDom}(f))=y) \qquad (4)\end{aligned}$$

where in the last line we used the fact that $f(x)=y$. Therefore, by putting (4) in (3) we get,

$$\begin{aligned}\Delta(\textsc{SampPre}(f,t,\mathcal{U}(R_\lambda)),\textsc{SampDom}(f)) \geq \sum_{y\in R_\lambda}\frac{1}{|R_\lambda|}\Delta\left(\textsc{SampPre}(f,\tau,y),X_y\right) \\ -\Delta(f(\textsc{SampDom}(f)),\mathcal{U}(R_\lambda)).\end{aligned}$$

which easily concludes the proof with proposition 2. □

4.1 Constructing a signature scheme from ATPSF

As pointed out in [1], the fact that a collection of ATPSF verifies the preimage property for almost all inputs is enough to build a signature scheme as in [18] and to use the security reduction given in [18, Proposition 6.1]. Nevertheless, by doing this we loose a square factor. We propose here to generalize the construction of [18] by adding a random salt in the signing algorithm. More precisely, given a collection an ATPSF $\mathcal{F} = (\textsc{TrapGen}, \textsc{SampDom}, \textsc{SampPre})$ we define the following Full Domain Hash signature scheme $\mathrm{S}^{\mathcal{F}}$: select a cryptographic hash function $\mathcal{H} : \{0,1\}^* \to R_\lambda$ and a random salt r of size λ_0 (λ_0 will be precised later). Consider the following 3 algorithms of the signature $\mathrm{S}^{\mathcal{F}}$:

$\mathrm{S}^{\mathcal{F}}.\textsc{keygen}(1^\lambda)$	$\mathrm{S}^{\mathcal{F}}.\textsc{sign}(m,\mathrm{pk},\mathrm{sk})$	$\mathrm{S}^{\mathcal{F}}.\textsc{verify}(m,(x,r),\mathrm{pk})$
$(f,\tau) \leftarrow \textsc{TrapGen}(1^\lambda)$	$r \xleftarrow{\$} \{0,1\}^{\lambda_0}$	$y \leftarrow \mathcal{H}(m,r)$
`return` $(\mathrm{pk},\mathrm{sk}) = (f,\tau)$	$y \leftarrow \mathcal{H}(m,r)$	`if` $f(x) = y$ `return` 1
	$x \leftarrow \textsc{SampPre}(y,\mathrm{sk})$	`else return` 0
	`return`(x,r)	

Our aim in what follows is to give a tight security reduction of this scheme using directly the average property of ATPS. In order to do so, we must first define different computational problems to reduce and in particular introduce our Claw(RF) problem.

The Random Oracle Model (ROM) in this construction. In the random oracle model, we replace the function $\mathcal{H}$ with a random function $h : \{0,1\}^* \times \{0,1\}^{\lambda_0} \to R_\lambda$ to which we only give black box access. Recall the EUF-CMA advantage of $\mathrm{S}^{\mathcal{F}}$:

$$\begin{aligned}Adv^{\text{EUF-CMA}}_{\mathrm{S}^{\mathcal{F}}}(\mathcal{A}^{\mathcal{O}_{\mathrm{S}^{\mathcal{F}}.\textsc{sign}}}) = \mathbb{P}\Big(\mathcal{H}(m^*,r^*) = f(x^*) \text{ AND } m^* \text{ has not been} \\ \text{queried in } \mathcal{O}_{\mathsf{Sign}} : (\mathrm{pk},\mathrm{sk}) \leftarrow \mathrm{S}.\textsc{keygen}(1^\lambda), (m^*,r^*,x^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}^{\mathcal{F}}.\textsc{sign}}}(\mathrm{pk})\Big).\end{aligned}$$

The ROM assumption says that any algorithm can only use $\mathcal{H}$ in a black box fashion and that it behaves as a random function. This translates to the fact that $\mathcal{A}$ can be seen as query algorithm not only to the signing oracle but also to the $\mathcal{H}$ function and that the EUF-CMA advantage is equal to the following one:

$$\mathbb{P}\Big(h(m^*,r^*)=f(x^*) \text{ AND } m^* \text{ has not been queried in } \mathcal{O}_{\mathsf{Sign}} : h \xleftarrow{\$} \mathfrak{F}_{R_\lambda}^{\{0,1\}^*\times\{0,1\}^{\lambda_0}} \\ (\mathrm{pk},\mathrm{sk}) \leftarrow \text{S.KEYGEN}(1^\lambda), (m^*,r^*,x^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{S}^{\mathcal{F}}.\text{SIGN}},\mathcal{O}_h}(\mathrm{pk})\Big).$$

5 One-wayness, Collision Resistance and the Claw with Random Function problem

The interest in using trapdoor functions for signatures is that these functions should be hard to invert without the trapdoor τ. Ideally, we want to reduce the security of the signature scheme to the hardness of inverting the function. However, this is not always possible and we have to reduce to other problems. We first present the notion of advantage related to one-wayness and collision finding. We then define our Claw(RF) problem and the associated advantage.

Definition 3. *Let* $\mathcal{F} = (\text{TrapGen}, \text{SampDom}, \text{SampPre})$ *be an* ATPRF. *For any algorithm* $\mathcal{A}$*, we define*

$$Adv_{\mathcal{F}}^{OW}(\mathcal{A}) \triangleq \mathbb{P}\left(f(x)=y : (f,\tau) \leftarrow \text{TrapGen}(1^\lambda),\ y \xleftarrow{\$} R_\lambda,\ x \leftarrow \mathcal{A}(f,y)\right),$$

$$Adv_{\mathcal{F}}^{Coll}(\mathcal{A}) \triangleq \mathbb{P}\left(f(x_1)=f(x_2) \wedge x_1 \neq x_2 : (f,\tau) \leftarrow \text{TrapGen}(1^\lambda),\ (x_1,x_2) \leftarrow \mathcal{A}(f)\right).$$

For any time t*, we also define*

$$Adv_{\mathcal{F}}^{OW}(t) \triangleq \max_{\mathcal{A}:|\mathcal{A}|=t} Adv_{\mathcal{F}}^{OW}(\mathcal{A}),$$

$$Adv_{\mathcal{F}}^{Coll}(t) \triangleq \max_{\mathcal{A}:|\mathcal{A}|=t} Adv_{\mathcal{F}}^{Coll}(\mathcal{A}).$$

Now, we define the Claw(RF) problem.

Problem 1 (Claw with Random Function - Claw(RF)*).*

- Instance: a function f and a random function h to which we only have black box access.
- Goal: find x, y such that $f(x) = h(y)$.

From there, we define the Claw(RF) advantage for any query algorithm $\mathcal{A}^{\mathcal{O}}$.

Definition 4. *Let* $\mathcal{F} = (\text{TrapGen}, \text{SampDom}, \text{SampPre})$ *be an* ATPRF.

$$Adv_{\mathcal{F}}^{Claw(RF)}(\mathcal{A}^{\mathcal{O}}) \triangleq \mathbb{P}\left(f(x)=h(y) : h \xleftarrow{\$} \mathfrak{F}_R^D,\ (f,\tau) \leftarrow \text{TrapGen}(1^\lambda),\ (x,y) \leftarrow \mathcal{A}^{\mathcal{O}_h}(f)\right) \\ = \mathbb{P}\left(f(x)=\mathcal{O}_{RO}(y) : (f,\tau) \leftarrow \text{TrapGen}(1^\lambda),\ (x,y) \leftarrow \mathcal{A}^{\mathcal{O}_{RO}}(f)\right)$$

For any time t *and any number of queries* q*, we also define*

$$Adv_{\mathcal{F}}^{Claw(RF)}(t,q) \triangleq \max_{\mathcal{A}^{\mathcal{O}}:|\mathcal{A}^{\mathcal{O}}|=(t,q)} Adv_{\mathcal{F}}^{Claw(RF)}(\mathcal{A}^{\mathcal{O}}).$$

Similarly, if we consider quantum algorithms, we can $QAdv_{\mathcal{F}}^{OW}(t)$, $QAdv_{\mathcal{F}}^{Coll}(t)$ and $QAdv_{\mathcal{F}}^{Claw(RF)}(t,q)$ where we maximize over quantum algorithms. In the case of $QAdv_{\mathcal{F}}^{Claw(RF)}(t,q)$, we allow *quantum queries* to $\mathcal{O}_h$.

Proposition 4. *Let $\mathcal{F}$ be an ε-ATPRF For any time t, we have*

$$Adv_{\mathcal{F}}^{OW}(t) \leq Adv_{\mathcal{F}}^{Claw(RF)}(t,1)$$

$$Adv_{\mathcal{F}}^{Claw(RF)}(t,q) \leq q \cdot Adv_{\mathcal{F}}^{OW}(t)$$
$$Adv_{\mathcal{F}}^{Claw(RF)}(t,q) \leq Adv_{\mathcal{F}}^{Coll}(t+\widetilde{O}(q)) + q\varepsilon + \mathbb{E}_{(f,t)\leftarrow \textsc{TrapGen}}(MNP(f))$$

where for $(f,\tau) \leftarrow \textsc{TrapGen}(1^\lambda)$, the minimal number of preimages $MNP(f)$ is

$$MNP(f) \stackrel{\triangle}{=} \min_y \left(|\{x : f(x) = y|\}\right).$$

Proof. We prove each inequality separately.

1. $\mathbf{Adv}_{\mathcal{F}}^{\mathbf{OW}}(\mathbf{t}) \leq \mathbf{Adv}_{\mathcal{F}}^{\mathbf{Claw(RF)}}(\mathbf{t},\mathbf{1})$. Let $\mathcal{A}$ be an algorithm running in time t with one-way advantage $Adv_{\mathcal{F}}^{OW}(t)$. We consider the following algorithm $\mathcal{B}^{\mathcal{O}_g}(f)$: $x_2 \stackrel{\$}{\hookleftarrow} D$, $y \stackrel{\triangle}{=} g(x_2)$, $x_1 \leftarrow \mathcal{A}(f,y)$, return (x_1,x_2). For a random g, y is a uniform element in R_λ. Moreover, since $f(x_1) = g(x_2)$ is equivalent to $f(x_1) = y$, we have $Adv_{\mathcal{F}}^{OW}(\mathcal{A}) \leq Adv_{\mathcal{F}}^{Claw(RF)}(\mathcal{B}^{\mathcal{O}_g})$. Finally notice that $\mathcal{B}^{\mathcal{O}_g}$ makes a single call to g and runs in the same time as $\mathcal{A}$, which concludes the proof.

2. $\mathbf{Adv}_{\mathcal{F}}^{\mathbf{Claw(RF)}}(\mathbf{t},\mathbf{q}) \leq \mathbf{q} \cdot \mathbf{Adv}_{\mathcal{F}}^{\mathbf{OW}}(\mathbf{t})$. Let $\mathcal{A}^{\mathcal{O}}$ be a query algorithm running in time t, performing q queries to $\mathcal{O}$ with Claw(RF) advantage $Adv_{\mathcal{F}}^{Claw(RF)}(t,q)$. Let $\mathcal{O}_{RO}(x;\mathcal{L})$ be the random oracle. We construct a new procedure $\mathcal{O''}_{j,y}$ which is equivalent to $\mathcal{O}_{RO}$ except the j^{th} call that outputs y. In the internal memory of $\mathcal{O''}_{j,y}$, we will keep track in a index i that corresponds to the number of times the oracle was queried +1.

`proc` $\mathcal{O''}_{j,y_0}(x;\mathcal{L},i)$
`if` $\exists y : (x,y) \in \mathcal{L}$, `return` $(y;\mathcal{L},(i+1))$
`otherwise if` $i = j$ `return` $(y_0;\mathcal{L} \cup \{(x,y_0)\},(i+1)$
`else take` $y \stackrel{\$}{\hookleftarrow} R_\lambda$, `return` $(y;\mathcal{L} \cup \{(x,y)\},(i+1))$

Notice that if j and y_0 are chosen at random then this doesn't change the behavior of the oracle. We consider the following algorithm $\mathcal{B}(f,y_0)$: pick $j \stackrel{\$}{\hookleftarrow} \{1,\ldots,q\}$. Run $\mathcal{A}^{\mathcal{O}_{RO}}$ but replace calls to $\mathcal{O}_{RO}$ with calls to $\mathcal{O}''_{j,y_0}$. Let (x_1,x_2) be the output of $\mathcal{A}^{\mathcal{O}_{RO}}$. $\mathcal{B}$ returns x_1. We write

$$\begin{aligned}
&Adv_{\mathcal{F}}^{Claw(RF)}(\mathcal{A}^{\mathcal{O}}) \\
&= \mathbb{P}\left(f(x_1) = \mathcal{O}_{RO}(x_2) : (f,\tau) \leftarrow \textsc{TrapGen}(1^\lambda),\ (x_1,x_2) \leftarrow \mathcal{A}^{\mathcal{O}_{RO}}(f)\right) \\
&= \mathbb{P}\Big[f(x_1) = \mathcal{O''}_{j,y}(x_2) : (f,\tau) \leftarrow \textsc{TrapGen}(1^\lambda), \\
&\qquad\qquad j \stackrel{\$}{\hookleftarrow} \{1,\ldots,q\},\ y \stackrel{\$}{\hookleftarrow} R_\lambda,\ (x_1,x_2) \leftarrow \mathcal{A}^{\mathcal{O''}_{j,y}}(f)\Big] \\
&\geq \frac{1}{q}\mathbb{P}\Big[f(x_1) = y : (f,\tau) \leftarrow \textsc{TrapGen}(1^\lambda),\ j \stackrel{\$}{\hookleftarrow} \{1,\ldots,q\},\ y \stackrel{\$}{\hookleftarrow} R_\lambda,\ (x_1,x_2) \leftarrow \mathcal{A}^{\mathcal{O''}_{j,y}}(f)\Big] \\
&= \frac{1}{q}\mathbb{P}\Big[f(x_1) = y : (f,\tau) \leftarrow \textsc{TrapGen}(1^\lambda),\ y \stackrel{\$}{\hookleftarrow} R_\lambda,\ x_1 \leftarrow \mathcal{B}(f,y)\Big]
\end{aligned}$$

where the inequality comes from the fact that when x_2 is queried in $\mathcal{O''}_{j,y}(x_2)$, there is a probability of $\frac{1}{q}$ that this corresponds to the j^{th} query on average on j which corresponds to $\mathcal{O''}_{j,y}(x_2) = y$.

3. $\mathbf{Adv}_{\mathcal{F}}^{\mathbf{Claw(RF)}}(\mathbf{t},\mathbf{q}) \leq \mathbf{Adv}_{\mathcal{F}}^{\mathbf{Coll}}(\mathbf{t}+\widetilde{\mathbf{O}}(\mathbf{q})) + \mathbf{q}\varepsilon + \mathbb{E}_{(\mathbf{f},\mathbf{t})\leftarrow \textsc{TrapGen}}\mathbf{MNP}(\mathbf{f})$. Let $\mathcal{A}^{\mathcal{O}}$ be a query algorithm running in time t, performing q queries to $\mathcal{O}$ with Claw(RF) advantage $Adv_{\mathcal{F}}^{Claw(RF)}(t,q)$. Here, $\mathcal{A}^{\mathcal{O}_g}$ makes calls to a procedure $\mathcal{O}_g$ for a random function g. We write

$$\boxed{\begin{array}{l}\texttt{proc } O_g(x)\\ \texttt{return } g(x)\end{array}}$$

Instead of calling $\mathcal{O}_g$ for a randomly chosen g, we use oracle $\mathcal{O}_{RO}$. We have

$$Adv_{\mathcal{F}}^{Claw(RF)}(\mathcal{A}^{\mathcal{O}}) = \mathbb{P}\Big(f(x_1) = \mathcal{O}_{RO}(x_2) : (f,\tau) \leftarrow \textsc{TrapGen}(1^\lambda),\ (x_1,x_2) \leftarrow \mathcal{A}^{\mathcal{O}_{RO}}(f)\Big) \quad (5)$$

We now define another procedure $\mathcal{O}'_f$ that is similar to $\mathcal{O}_{RO}$ but we change the way y is sampled.

$$\boxed{\begin{array}{l}\texttt{proc } \mathcal{O}'_f(x;\mathcal{L})\\ \texttt{if } \exists y : (x,y) \in \mathcal{L}\texttt{, return } (y;\mathcal{L})\\ \texttt{otherwise compute } z \leftarrow \textsc{SampDom}(f)\texttt{, } y \stackrel{\triangle}{=} f(z)\texttt{, return } (y;\mathcal{L}\cup\{(x,y)\})\end{array}}$$

Therefore we have:

$$\forall x,\mathcal{L}, \qquad \Delta\left(\mathcal{O}_{RO}(x;\mathcal{L}),\mathcal{O}'_f(x;\mathcal{L})\right) \leq \varepsilon \quad (6)$$

from Proposition 2.

We now consider the following algorithm $\mathcal{B}$: run $\mathcal{A}^{\mathcal{O}}$. Each time $\mathcal{O}$ is called, run $\mathcal{O}'_f$ and keep track efficiently of the internal memory $\mathcal{L}$, with a sorted list. Initialize $\mathcal{L} = \emptyset$. The list $\mathcal{L}$ is of size at most q so each membership query to $\mathcal{L}$ can be done in time at most $O(\log(q))$, so $\mathcal{B}$ runs in time $t + \widetilde{O}(q)$. Moreover, since $\mathcal{O}'_f$ is called q times, using Equations (5),(6) and Proposition 1, we have

$$Adv_{\mathcal{F}}^{Claw(RF)}(\mathcal{A}^{\mathcal{O}}) \leq \mathbb{P}\Big(f(x_1) = \mathcal{O}'_f(x_2) : (f,\tau) \leftarrow \textsc{TrapGen}(1^\lambda),\ (x_1,x_2) \leftarrow \mathcal{B}(f)\Big) + q\varepsilon.$$

Now, we construct the following algorithm $\mathcal{C}$: run $\mathcal{B}$. Each time $\mathcal{O}'_f(x;\mathcal{L})$ is called, keep track of the value z such that $f(z) = \mathcal{O}'_f(x)$. Let x_1, x_2 be the output of $\mathcal{B}(f)$. Let z such that $\mathcal{O}'_f(x_2) = f(z)$. Output (x_1, z). Again, $\mathcal{C}$ runs in time $t + \widetilde{O}(q)$. We have

$$Adv_{\mathcal{F}}^{Claw(RF)}(\mathcal{A}^{\mathcal{O}}) \leq \mathbb{P}\Big(f(x_1) = f(z) : (f,\tau) \leftarrow \textsc{TrapGen}(1^\lambda),\ (x_1,z) \leftarrow \mathcal{C}(f)\Big) + q\varepsilon.$$

In order to relate this to the collision advantage, we just need to find the probability that $x_1 \neq z$ in the above. From the construction of $\mathcal{C}$ and $\mathcal{O}_{f'}$, we have that z is a random preimage of $f(x_1)$. Therefore, $x_1 \neq z$ with probability $1 - MNP(f)$[5]. From there, we can conclude

$$Adv_{\mathcal{F}}^{Claw(RF)}(\mathcal{A}^{\mathcal{O}}) \leq Adv_{\mathcal{F}}^{Coll}(\mathcal{C}) + q\varepsilon + \mathbb{E}_{(f,t)\leftarrow \textsc{TrapGen}}\left(MNP(f)\right).$$

6 Tight reduction to the claw problem, with ATPSF

6.1 Proof of our main theorem

Theorem 1. *Let $\mathcal{F} = (\textsc{TrapGen}, \textsc{SampDom}, \textsc{SampPre})$ be a collection of ε-ATPSF with security parameter λ. Let $\mathrm{S}^{\mathcal{F}}$ be the associated Hash and Sign signature scheme with salt size λ_0. For any $t, q_{\text{hash}}, q_{\text{sign}}$, we have*

$$Adv_{\mathrm{S}^{\mathcal{F}}}^{\text{EUF-CMA}}(t, q_{\text{hash}}, q_{\text{sign}}) \leq Adv_{\mathcal{F}}^{Claw(RF)}(\widetilde{O}(t), q_{\text{hash}}) + q_{\text{sign}}\left(\varepsilon + \frac{(q_{sign} + q_{hash})}{2^{\lambda_0}}\right)$$

by taking $\lambda_0 = \lambda + 2\log(q_{\text{sign}}) + \log(q_{\text{hash}})$, we have

$$Adv_{\mathrm{S}^{\mathcal{F}}}^{\text{EUF-CMA}}(t, q_{\text{hash}}, q_{\text{sign}}) \leq Adv_{\mathcal{F}}^{Claw(RF)}(\widetilde{O}(t), q_{\text{hash}}) + q_{\text{sign}}\varepsilon + \frac{1}{2^\lambda}.$$

[5] A similar argument was already implicitly used in [18]

Proof. Let $\mathcal{A}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}}}$ be an attacker with $|\mathcal{A}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}}}| = (t, q_{\text{hash}}, q_{\text{sign}})$ such that $Adv^{\text{EUF-CMA}}_{\mathrm{S}^{\mathcal{F}}}(t, q_{\text{hash}}, q_{\text{sign}}) = Adv^{\text{EUF-CMA}}_{\mathrm{S}^{\mathcal{F}}}(\mathcal{A}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}}})$. We show how to construct a query algorithm $\mathcal{C}^{\mathcal{O}}$ to attack the claw with random function property of $\mathcal{F}$. In the signature scheme $\mathrm{S}^{\mathcal{F}}$, we have the following hash and sign procedures, where the Hash procedure is the Random Oracle.

$\underline{\texttt{proc Hash}}(x; \mathcal{L})$
$\texttt{if } \exists! y : (x, y) \in \mathcal{L}\texttt{, return } (y; \mathcal{L})$
$\texttt{otherwise, pick } y \xleftarrow{\$} R_\lambda\texttt{, return } (y; \mathcal{L} \cup \{(x, y)\})$

$\underline{\texttt{proc Sign}}(m; \mathcal{L})$
$r \xleftarrow{\$} \{0,1\}^{\lambda_0}$
$(y; \mathcal{L}') \leftarrow \texttt{Hash}(m||r; \mathcal{L})$
$x \leftarrow \textsc{SampPre}(f, \tau, y)$
$\texttt{return } (x, r; \mathcal{L}')$

Recall that $\mathcal{L}$ corresponds to the list of input/output pairs already queried to the Hash function. Here, both procedures use the same $\mathcal{L}$ and each time is it updated, this update happens for both procedures at the same time. We first explicitly replace the Hash procedure by its code in Sign:

$\underline{\texttt{proc Sign}}(m; \mathcal{L})$
$r \xleftarrow{\$} \{0,1\}^{\lambda_0}$
$(y; \mathcal{L}') \leftarrow \texttt{Hash}(m||r; \mathcal{L})$
$x \leftarrow \textsc{SampPre}(f, \tau, y)$
$\texttt{return } (x, r; \mathcal{L}')$

=

$\underline{\texttt{proc Sign}}(m; \mathcal{L})$
$r \xleftarrow{\$} \{0,1\}^{\lambda_0}$
$\texttt{if } \exists! y_0 : (m||r, y_0) \in \mathcal{L}$
$\texttt{then}$
$x_0 \xleftarrow{\$} \textsc{SampPre}(f, \tau, y_0)$
$\texttt{return } (x_0, r; \mathcal{L})$
$\texttt{else}$
$y \xleftarrow{\$} R_\lambda$
$x \xleftarrow{\$} \textsc{SampPre}(f, \tau, y)$
$\mathcal{L}' \triangleq \mathcal{L} \cup \{(m||r, y)\}$
$\texttt{return } (x, r; \mathcal{L}')$

Now, we present a new signature procedure Sign', that will be close to Sign but doesn't use τ.

$\underline{\texttt{proc Sign}'}(m; \mathcal{L})$
$r \xleftarrow{\$} \{0,1\}^{\lambda_0}$
$\texttt{if } \exists! y_0 : (m||r, y_0) \in \mathcal{L}$
$\texttt{then}$
$\texttt{return } \perp$
$\texttt{else}$
$x \xleftarrow{\$} \textsc{SampDom}(f)$
$y \triangleq f(x)$
$\mathcal{L}' \triangleq \mathcal{L} \cup \{(m||r, y)\}$
$\texttt{return } (x, r; \mathcal{L}')$

We made two changes from Sign to Sign'. In the case where $\exists y_0, : (m||r, y_0) \in \mathcal{L}$, Sign' outputs $\perp$. In the other case, Sign' also has a different way of sampling x and y. We show that these 2 changes do not change a lot the output distribution of the sign procedure:

Lemma 1. *For any f, τ as well as m and $\mathcal{L}$, we have $\Delta\left(\mathsf{Sign}(m; \mathcal{L}), \mathsf{Sign}'(m; \mathcal{L})\right) \leq \varepsilon_{f,\tau} + \frac{|\mathcal{L}|}{2^{\lambda_0}}$.*

Proof. We consider the following intermediate procedure Sign_{int}

$\underline{\texttt{proc Sign}_{int}}(m; \mathcal{L})$
$r \xleftarrow{\$} \{0,1\}^{\lambda_0}$
$\texttt{if } \exists! y_0 : (m||r, y_0) \in \mathcal{L}$
$\texttt{then}$
$\texttt{return } \perp$
$\texttt{else}$
$y \xleftarrow{\$} R_\lambda$
$x \xleftarrow{\$} \textsc{SampPre}(f, \tau, y)$
$\mathcal{L}' \triangleq \mathcal{L} \cup \{(m||r, y)\}$
$\texttt{return } (x, r; \mathcal{L}')$

$\mathsf{Sign}(m;\mathcal{L})$ and $\mathsf{Sign}_{int}(m;\mathcal{L})$ only differ when for the random choice $r \xleftarrow{\$} \{0,1\}^{\lambda_0}$, $\exists! y_0 : (m||r, y_0) \in \mathcal{L}$. This event happens with probability at most $\frac{|\mathcal{L}|}{2^{\lambda_0}}$ hence $\Delta(\mathsf{Sign}(m;\mathcal{L}), \mathsf{Sign}_{int}(m;\mathcal{L})) \leq \frac{|\mathcal{L}|}{2^{\lambda_0}}$.

Now, let's look at the distance between $\mathsf{Sign}_{int}(m;\mathcal{L})$ and $\mathsf{Sign}'(m;\mathcal{L})$. The only difference in those distributions comes from the way x and y are sampled. Since both in Sign_{int} and Sign', we have $y = f(x)$ (and f is deterministic), the only difference comes from the way x is sampled. Therefore,

$$\Delta\left(\mathsf{Sign}_{int}(m;\mathcal{L}), \mathsf{Sign}'(m;\mathcal{L})\right) = \Delta\left(\textsc{SampPre}(f,\tau,U(R_\lambda)), \textsc{SampDom}(f)\right) = \varepsilon_{f,\tau}$$

and we can therefore conclude the proof using the triangle inequality. □

We are now ready to finish the proof of Theorem 1. From an adversary $\mathcal{A}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}}}(f)$, we construct an algorithm $\mathcal{B}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}'}}(f)$ which corresponds to running $\mathcal{A}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}}}$ but calls to $\mathcal{O}_{\mathsf{Sign}}$ are replaced with calls to $\mathcal{O}_{\mathsf{Sign}'}$. We also ask $\mathcal{B}$ to emulate by himself the oracles $\mathcal{O}_{\mathsf{Hash}}, \mathcal{O}'_{\mathsf{Sign}}$. To do this, it initializes $\mathcal{L} = \emptyset$ and runs these algorithms by himself by updating $\mathcal{L}$ efficiently via a sorted list. Notice that this was not possible with $\mathcal{O}_{\mathsf{Sign}}$ because it required τ that $\mathcal{B}$ does not have access to. Let us define $Adv'(\cdot)$ as:

$$Adv'(\mathcal{B}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}'}}) \triangleq \mathbb{P}\Big(\mathsf{Hash}(m^*||r^*) = f(e^*) \ \wedge \ (m^*,e^*,r^*) \text{ wasn't answered} \\ \text{by } \mathcal{O}_{\mathsf{Sign}'} \text{ in } \mathcal{B} : (f,\tau) \leftarrow \textsc{TrapGen}(1^\lambda), (m^*,r^*,e^*) \leftarrow \mathcal{B}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}'}}(f)\Big).$$

On average on f, the outputs of $\mathcal{B}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}'}}$ differ from those of $\mathcal{A}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}}}(f)$ only because we replaced calls to $\mathcal{O}_{\mathsf{Sign}}$ with calls to $\mathcal{O}_{\mathsf{Sign}'}$. There are q_{sign} such calls and using Lemma 1, we have:

$$Adv^{\text{SEUF-CMA}}_{\text{S}^{\mathcal{F}}}(\mathcal{A}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}}}) \leq Adv'(\mathcal{B}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}'}}) + q_{sign}\left(\varepsilon + \frac{(q_{sign}+q_{hash})}{2^{\lambda_0}}\right)$$

where we here also averaged over $(f,\tau) \leftarrow \textsc{TrapGen}(1^\lambda)$.

When we first discussed the random oracle model, we showed how when calling an oracle $\mathcal{O}_g$ for a random g, we could "internalize" the random function into each call of $\mathcal{O}_{RO}$. In order to arrive at the quantity $Adv^{Claw(RF)}_{\mathcal{F}}$, we have to undo this step and externalize the random function, but we want to keep the internal memory $\mathcal{L}$ since it can also be modified by $\mathcal{O}_{\mathsf{Sign}'}$. More precisely, for any function g, we define

$$\boxed{\begin{array}{l}\texttt{proc } \mathsf{Hash}_g(x;\mathcal{L}) \\ \texttt{if } \exists! y : (x,y) \in \mathcal{L}\texttt{, return } (y;\mathcal{L}) \\ \texttt{otherwise, return } (g(x); \mathcal{L} \cup \{(x,g(x))\})\end{array}}.$$

When, we run Hash, each time a fresh x is queried - meaning $\forall y, (x,y) \notin \mathcal{L}$ - we pick a random value y as its output. Equivalently, we can compute all those possible values y at the beginning, characterized by values $g(x)$ for a random function g. Therefore, we have

$$Adv'(\mathcal{B}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}'}}) = \mathbb{P}\Big(\mathsf{Hash}_g(m^*||r^*) = f(x^*) \ \wedge \ m^* \text{ wasn't queried to} \\ \mathcal{O}_{\mathsf{Sign}'} \text{ in } \mathcal{B} : g \xleftarrow{\$} \mathcal{RF}, \ (f,\tau) \leftarrow \textsc{TrapGen}(1^\lambda), (m^*,r^*,x^*) \leftarrow \mathcal{B}^{\mathcal{O}_{\mathsf{Hash}_g},\mathcal{O}_{\mathsf{Sign}'}}(f)\Big).$$

Now, for a fixed g, let's try to characterize $\mathsf{Hash}_g(m||r)$ for any m, r. If $\forall y, (m||r,y) \notin \mathcal{L}$ then $\mathsf{Hash}_g(m||r) = g(m||r)$. Otherwise, let y such that $(m||r,y) \in \mathcal{L}$ and we distinguish 2 cases:

1. $(m||r,y)$ was put in $\mathcal{L}$ after a call to Hash, then $\mathsf{Hash}_g(m||r) = g(m||r)$.
2. $(m||r,y)$ was put in $\mathcal{L}$ after a call to Sign', then m was queried to $\mathcal{O}_{\mathsf{Sign}'}$.

Therefore, for any triplet $(m^*, r^*, x^*) \leftarrow \mathcal{B}^{\mathcal{O}_{\mathsf{Hash}_g}, \mathcal{O}_{\mathsf{Sign}'}}$, we have

$$m^* \text{ is not queried to } \mathcal{O}_{\mathsf{Sign}'} \text{ or } m^* \text{ is queried and } (x^*, r^*) \text{ is not answered by } \mathcal{O}_{\mathsf{Sign}'} \\ \Leftrightarrow \mathsf{Hash}_g(m^*||r^*) = g(m^*||r^*).$$

From there, we have

$$Adv'(\mathcal{B}^{\mathcal{O}_{\mathsf{Hash}}, \mathcal{O}_{\mathsf{Sign}'}}) = \mathbb{P}\Big(g(m^*||r^*) = f(x^*) : g \leftarrow \mathfrak{F}_{R_\lambda}^{\{0,1\}^* \times \{0,1\}^{\lambda_0}}, \\ (f, \tau) \leftarrow \textsc{TrapGen}(1^\lambda), (m^*, r^*, x^*) \leftarrow \mathcal{B}^{\mathcal{O}_{\mathsf{Hash}_g}, \mathcal{O}_{\mathsf{Sign}'}(f)}\Big).$$

In order to conclude, notice that the algorithm $\mathcal{B}^{\mathcal{O}_{\mathsf{Hash}_g}, \mathcal{O}_{\mathsf{Sign}'}}$ can be seen as an algorithm $\mathcal{C}^{\mathcal{O}_g}$ that runs in time $\widetilde{O}(t)$ and performs q_{hash} queries to O_g so

$$\begin{aligned} Adv'(\mathcal{B}^{\mathcal{O}_{\mathsf{Hash}}, \mathcal{O}_{\mathsf{Sign}'}}) &= \mathbb{P}\Big(g(m^*||r^*) = f(x^*) : g \leftarrow \mathfrak{F}_{R_\lambda}^{\{0,1\}^* \times \{0,1\}^{\lambda_0}}, \\ &\qquad (f, \tau) \leftarrow \textsc{TrapGen}(1^\lambda), (m^*, r^*, x^*) \leftarrow \mathcal{C}^{\mathcal{O}_g}(f)\Big) \\ &= Adv_{\mathcal{F}}^{Claw(RF)}(\mathcal{C}^{\mathcal{O}_g}) \end{aligned}$$

Putting everything together, we get $Adv_{\mathcal{F}}^{Claw(RF)}(\mathcal{C}^{\mathcal{O}_g}) = Adv'(\mathcal{B}^{\mathcal{O}_{\mathsf{Hash}}, \mathcal{O}_{\mathsf{Sign}'}})$ and

$$Adv_{\mathrm{S}^{\mathcal{F}}}^{\text{EUF-CMA}}(\mathcal{A}^{\mathcal{O}_{\mathsf{Hash}}, \mathcal{O}_{\mathsf{Sign}}}) \leq Adv_{\mathcal{F}}^{Claw(RF)}(\mathcal{C}^{\mathcal{O}_g}) + q_{sign}\left(\varepsilon + \frac{(q_{sign} + q_{hash})}{2^{\lambda_0}}\right)$$

which concludes the proof. □

6.2 Changing the security notion to avoid weak key attacks

The way we defined EUF-CMA security for a signature scheme $\mathcal{S}$ was that an attacker in time t can't break the signature scheme with probability greater than $Adv_{\mathcal{S}}^{\text{EUF-CMA}}(t)$. However, this could lead weak key attacks in the following way: $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{S.KEYGEN}$ could generate some keys such that with probability 2^{-40}, one can recover sk from pk with probability 2^{-40}. However, the way $Adv_{\mathcal{S}}^{\text{EUF-CMA}}$ is defined (on average on $(\mathrm{pk}, \mathrm{sk})$) would seem to imply that this scenario has 80 bits of security. This is probably something known in the community but is often overlooked in the definitions.

What we should do is to replace the average on $(\mathrm{pk}, \mathrm{sk})$ with a notion that says that the Advantage should be smaller than ε for almost all $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{S.KEYGEN}$[6]. Our tight and optimal security results directly follow to this definitions using the same proofs. In order for this to work, we require the following:

- The definition of ATPSF should also hold for almost all $(f, \tau) \leftarrow \textsc{TrapGen}$ and not on average.
- The Claw(RF) advantage should also be defined for almost all $(f, \tau) \leftarrow \textsc{TrapGen}$ and not on average on (f, τ).

Once we have this, the same proofs give the same reductions in this scenario.

7 Quantum Security Proof in the QROM

In this section, we will prove that in the quantum setting, we can also prove the security of $\mathrm{S}^{\mathcal{F}}$ for a collection $\mathcal{F}$ of ATPSF. We first present the quantum random oracle model.

[6] Actually, this notion is probably too strong. We believe that the actual correct notion is to allow the adversary to have access to a S.KEYGEN oracle but this is a discussion for another time.

7.1 The Quantum Random Oracle Model

The Quantum Random Oracle Model (QROM) is a model where we model a certain function with a random function $\mathcal{H}$ but since we are in the quantum setting, we have a black box access to $\mathcal{H}$ and thus also to the unitary $\mathcal{O}_{\mathcal{H}}(|x\rangle|y\rangle) = |x\rangle|\mathcal{H}(x)+y\rangle$. Unlike the classical setting, when calling $\mathcal{O}_{\mathcal{H}}$ for a randomly chosen $\mathcal{H}$, we will not be able to generate values $\mathcal{H}(x)$ on the fly as we did classically since a quantum query potentially queries all values $\mathcal{H}(x)$ at the same time[7]. Hopefully we will still have tools to reprogram the QROM.

When a function h is drawn uniformly from the set of functions $\mathfrak{F}^D_{\{0,1\}^m}$, we can equivalently, for each input $x \in D$, draw $h(x) \xleftarrow{\$} \{0,1\}^m$, which fully specified the function h. For each distribution $\mathcal{D}$ on $\{0,1\}^m$, let us consider the distribution of functions $\mathrm{Fun}_{\mathcal{D}}$ where $h \leftarrow \mathrm{Fun}_{\mathcal{D}}$ means that for each x, $h(x) \xleftarrow{\$} \mathcal{D}$. In [29], Zhandry showed the following relation.

Proposition 5. *Let $\mathcal{A}^{\mathcal{O}}$ be a quantum query algorithm running in time t and making q queries to the oracle $\mathcal{O}$. Let $\mathcal{D}$ be a probability distribution on $\{0,1\}^m$ such that $\Delta(\mathcal{D},\mathcal{U}(\{0,1\}^m)) \leq \varepsilon$. We have*

$$\left| \mathbb{P}\left(\mathcal{A}^{\mathcal{O}_h} \text{ outputs } 1 : h \leftarrow \mathfrak{F}^D_{\{0,1\}^m}\right) - \mathbb{P}\left(\mathcal{A}^{\mathcal{O}_g} \text{ outputs } 1 : g \leftarrow Fun_{\mathcal{D}}\right) \right| \leq \frac{8\pi}{\sqrt{3}} q^{3/2}\sqrt{\varepsilon}.$$

One can compare this to the classical case, which follows directly from Proposition 1.

Proposition 6. *Let $\mathcal{A}^{\mathcal{O}}$ be a classical query algorithm running in time t and making q queries to the oracle $\mathcal{O}$. Let $\mathcal{D}$ be a probability distribution on $\{0,1\}^m$ such that $\Delta(\mathcal{D},\mathcal{U}(\{0,1\}^m)) \leq \varepsilon$. We have*

$$\left|\mathbb{P}\left(\mathcal{A}^{\mathcal{O}_h} \text{ outputs } 1 : h \leftarrow \mathfrak{F}^D_{\{0,1\}^m}\right) - \mathbb{P}\left(\mathcal{A}^{\mathcal{O}_g} \text{ outputs } 1 : g \leftarrow Fun_D\right)\right| \leq q\varepsilon.$$

With Proposition 5, we will be able to prove the quantum security of $\mathrm{S}^{\mathcal{F}}$.

7.2 Tight quantum security of $\mathrm{S}^{\mathcal{F}}$

The goal of this section is to prove the following theorem

Theorem 2. *Let $\mathcal{F} = (\textsc{TrapGen}, \textsc{SampDom}, \textsc{SampPre})$ be an ε-ATPSF. Let $\mathrm{S}^{\mathcal{F}}$ be the associated Hash and Sign signature scheme. Let $\mathcal{A}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}}}$ an adversary with $|\mathcal{A}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}}}| = (t, q_{\mathsf{Hash}}, q_{\mathsf{Sign}})$ running in time t. There exists an algorithm $\mathcal{C}^{\mathcal{O}}$ running in time $\widetilde{O}(t)$ such that*

$$QADV^{\text{EUF-CMA}}_{\mathrm{S}^{\mathcal{F}}}(t, q_{\text{hash}}, q_{\text{sign}}) \leq \frac{1}{2}\left(QADV^{Claw(RF)}(\widetilde{O}(t), q_{\text{hash}}) + \frac{8\pi}{\sqrt{3}} q^{3/2}\sqrt{\varepsilon} + q_{\text{sign}}\left(\varepsilon + \frac{q_{\text{sign}}}{2^{\lambda_0}}\right)\right)$$

by taking $\lambda_0 = \lambda + 2\log(q_{\text{sign}})$, this gives

$$QADV^{\text{EUF-CMA}}_{\mathrm{S}^{\mathcal{F}}}(t, q_{\text{hash}}, q_{\text{sign}}) \leq \frac{1}{2}\left(QADV^{Claw(RF)}(\widetilde{O}(t), q_{\text{hash}}) + \frac{8\pi}{\sqrt{3}} q^{3/2}\sqrt{\varepsilon} + q_{\text{sign}}\varepsilon + \frac{1}{2^{\lambda}}\right).$$

Before proving this statement, we need to add another definition. Let $\mathcal{F} = (\textsc{TrapGen}, \textsc{SampDom}, \textsc{SampPre})$ be an ε-ATPSF. We said that $\textsc{SampDom}(f)$ was an efficient probabilistic algorithm. Here, we need to explicit this randomness and work with a deterministic algorithm. Let $\textsc{SampDom}_{det}(f, K)$ be the algorithm which corresponds to running $\textsc{SampDom}(f)$ with randomness $K \in \{0,1\}^k$. What this means is that running $\textsc{SampDom}(f)$ is done by choosing $K \xleftarrow{\$} \{0,1\}^k$ and running $\textsc{SampDom}_{det}(f, k)$.

With this new definition, we can go and prove our theorem.

[7] It is actually possible to do this via the quantum lazy sampling routine [10] but we will use much simpler tools here.

Proof of Theorem 2: Fix $\mathcal{F}, \mathrm{S}^{\mathcal{F}}$ and let $\mathcal{A}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}}}$ an adversary in quantum EUF-CMA model with $|\mathcal{A}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}}}| = (t, q_{\mathsf{Hash}}, q_{\mathsf{Sign}})$ running in time t . We first construct a new function Z that we will use as a replacement for Hash. Consider J, a random hash function from $\{0,1\}^* \times \{0,1\}^{\lambda_0}$ to $\{0,1\} \times \{0,1\}^k$. In particular, the first bit of $J(m,r)$ is a random element of $\{0,1\}$ for any m, r. From the functions J and Hash we can build the function $Z : \{0,1\}^* \times \{0,1\}^{\lambda_0} \to R_\lambda$ as follows:

$$Z(m,r) = \begin{cases} \mathsf{Hash}(m,r) & \text{if } J(m,r) = (0,K) \text{ for some } K \in \{0,1\}^k \\ f(\textsc{SampDom}_{det}(f,K)) & \text{if } J(m,r) = (1,K) \end{cases}$$

We can easily construct an efficient quantum circuit for O_Z using $O_{\mathcal{H}}$ and O_J. Notice also that for each m, r since Hash and J are random functions, $Z(m,r)$ follows a distribution which is at most $\varepsilon/2$-close to the uniform distribution. Indeed, $Z(m,r)$ follows the uniform distribution with probability $\frac{1}{2}$ and the distribution $\textsc{SampDom}(f)$ with probability $\frac{1}{2}$. But these two distributions are at most at distance $\varepsilon_{f,\tau}$ from Proposition2.

Our first change is that we replace Hash with Z (we also do this change in the Sign procedure). Then, we change the Sign procedure into a procedure Sign' so that it doesn't use the trapdoor and can be emulated only with the public key.

$\underline{\texttt{proc Sign}(m)}$
$r \xleftarrow{\$} \{0,1\}^{\lambda_0}$
$y \leftarrow Z(m,r)$
$x \leftarrow \textsc{SampPre}(f,\tau,y)$
$\texttt{return } (x,r)$

$\rightarrow$

$\underline{\texttt{proc Sign}'(m)}$
$r \xleftarrow{\$} \{0,1\}^{\lambda_0}$
$(b,K) = J(m,r)$. $\texttt{If } b = 0\texttt{, pick new } r.$
$J(m,r) = (1,K)$ $\texttt{so } Z(m,r) = f(\textsc{SampDom}_{det}(f,K))$.
$x \triangleq \textsc{SampDom}_{det}(f,K)$
$\texttt{return } (x,r).$

Notice that for any m, $\Delta(\mathsf{Sign}(m), \mathsf{Sign}(m')) \leq \varepsilon$. This is because both in Sign and Sign', r is a uniform random string as long as each call to the signing oracle chooses a different r. This happens with probability $\geq 1 - \frac{q_{\text{sign}}}{2^{\lambda_0}}$ (the values r such that $J(m,r) = (1,e)$ are uniform for a random J). We therefore have using Proposition 5:

$$\begin{aligned} QADV^{\text{EUF-CMA}}_{\mathrm{S}^{\mathcal{F}}}(\mathcal{A}^{\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Sign}}}) &\geq QADV^{\text{EUF-CMA}}_{\mathrm{S}^{\mathcal{F}}}(\mathcal{A}^{\mathcal{O}_Z,\mathcal{O}_{\mathsf{Sign}}}) + \frac{8\pi}{\sqrt{3}} q^{3/2}\sqrt{\varepsilon} \\ &\geq QADV^{\text{EUF-CMA}}_{\mathrm{S}^{\mathcal{F}}}(\mathcal{A}^{\mathcal{O}_Z,\mathcal{O}_{\mathsf{Sign}'}}) + \frac{8\pi}{\sqrt{3}} q^{3/2}\sqrt{\varepsilon} + q_{\text{sign}}\left(\varepsilon + \frac{q_{\text{sign}}}{2^{\lambda_0}}\right). \end{aligned}$$

Now, we write:

$$\begin{aligned} QADV^{\text{EUF-CMA}}_{\mathrm{S}^{\mathcal{F}}}(\mathcal{A}^{\mathcal{O}_Z,\mathcal{O}_{\mathsf{Sign}'}}) &= \mathbb{P}\Big(Z(m^*,r^*) = f(x^*) \text{ AND } m^* \text{ has not been queried in } \mathcal{O}_{\mathsf{Sign}'} : \\ &\qquad (f,\tau) \leftarrow \textsc{TrapGen},\ (m^*,x^*,r^*) \leftarrow \mathcal{A}^{\mathcal{O}_Z,\mathcal{O}_{\mathsf{Sign}'}}(f)\Big) \\ &= \frac{1}{2}\mathbb{P}\Big(\mathsf{Hash}(m^*,r^*) = f(x^*) : \\ &\qquad (f,\tau) \leftarrow \textsc{TrapGen},\ (m^*,x^*,r^*) \leftarrow \mathcal{A}^{\mathcal{O}_Z,\mathcal{O}_{\mathsf{Sign}'}}(f)\Big) \\ &= \frac{1}{2} QADV^{Claw(RF)}(\mathcal{A}^{\mathcal{O}_Z,\mathcal{O}'_{\mathsf{Sign}}}). \end{aligned}$$

Let $\mathcal{B}^{\mathcal{O}_{\mathsf{Hash}}} = \mathcal{A}^{\mathcal{O}_Z,\mathcal{O}'_{\mathsf{Sign}}}$. We have

$$QADV^{\text{EUF-CMA}}_{\mathrm{S}^{\mathcal{F}}}(\mathcal{A}^{\mathcal{O}_Z,\mathcal{O}_{\mathsf{Sign}'}}) \leq \frac{1}{2}\left(QADV^{Claw(RF)}(\mathcal{B}^{\mathcal{O}_{\mathsf{Hash}}}) + \frac{8\pi}{\sqrt{3}} q^{3/2}\sqrt{\varepsilon} + q_{\text{sign}}\left(\varepsilon + \frac{q_{\text{sign}}}{2^{\lambda_0}}\right)\right).$$

□

8 Codes: Syndrome Decoding vs. Decoding One Out of Many vs. Collision on codes

In this section we will apply our security reduction to the collection of ATPSF Wave [12] in the paradigm of error correcting codes. It gives a first application of our work when compared to the reduction of [18]. First, Wave is only proved to be "preimage sampleable on average" in [12] and applying directly the reduction of [18] with Proposition 3 would lead to a lost of a square factor. Secondly, as we will see the collision problem for the proposed parameters of Wave can be solved in polynomial time while the $Claw(RF)$ problem, which is in this case roughly equivalent to the Decoding One Out of Many (DOOM) problem [20, 25], is well-known. Furthermore, solving DOOM is in the state-of-the-art for Wave's parameters [7] of the same exponential complexity as the generic decoding problem upon which relies the code-based cryptography which gives an extremely "tight algorithmic reduction".

Notations. We will denote by $\mathbb{F}_q$ the finite field with elements. Vectors will be written with bold letters (such as $\mathbf{e}$) and uppercase bold letters are used to denote matrices (such as $\mathbf{H}$). Vectors are in row notation. We will denote for a vector $\mathbf{x}$ by $\mathbf{x}(i)$ its i-th entry. Furthermore, the Hamming of a vector $\mathbf{x} \in \mathbb{F}_q^n$, that we denote by $|\mathbf{x}|$, is the number of its non-zero components, namely

$$|\mathbf{x}| \stackrel{\triangle}{=} \#\left\{1 \leq i \leq n : \mathbf{x}(i) \neq 0\right\}.$$

8.1 Ony-Way, Claw(RF) and Collision Problems in Code-Based Cryptography

Let us start with some basic definitions and by translating "One-Way, Collision and Claw(RF)" into the code-based setting which namely correspond to problems Syndrome Decoding (SD), Collision and DOOM.

A q-ary linear code $\mathcal{C}$ of length n and dimension k is a subspace of $\mathbb{F}_q^n$ of dimension k and is often defined by a *parity-check matrix* $\mathbf{H}$ over $\mathbb{F}_q$ of size $(n-k) \times n$ as

$$\mathcal{C} = \left\{\mathbf{x} \in \mathbb{F}_q^n : \mathbf{x}\mathbf{H}^\intercal = \mathbf{0}\right\}.$$

The problem of decoding a q-ary linear code with this formalism can be expressed as follows.

Problem 2 (Syndrome Decoding - $\mathrm{SD}(n, q, R, \omega)$*).*

- Instance: a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ of rank $n-k$, a syndrome $\mathbf{s} \in \mathbb{F}_q^{n-k}$,
- Output: $\mathbf{e} \in \mathbb{F}_q^n$ of Hamming weight w such that $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$

where $k \stackrel{\triangle}{=} \lfloor Rn \rfloor$ and $w \stackrel{\triangle}{=} \lfloor \omega n \rfloor$.

The advantage of breaking SD for uniformly distributed outputs is then defined as follows.

Definition 5 (SD-advantage(n, q, R, ω)**).** *Let $k \stackrel{\triangle}{=} \lfloor Rn \rfloor$ and $w \stackrel{\triangle}{=} \lfloor \omega n \rfloor$. For any algorithm $\mathcal{A}$, we define,*

$$Adv^{\mathrm{SD}}_{(n,q,R,\omega)}(\mathcal{A}) \stackrel{\triangle}{=} \mathbb{P}\left(\mathbf{e}\mathbf{H}^\intercal = \mathbf{s} \text{ and } |\mathbf{e}| = w : \mathbf{H} \stackrel{\$}{\hookleftarrow} \mathbb{F}_q^{(n-k)\times n},\ \mathbf{s} \stackrel{\$}{\hookleftarrow} \mathbb{F}_q^{n-k},\ \mathbf{e} \leftarrow \mathcal{A}(\mathbf{H}, \mathbf{s})\right),$$

and for any time t, we also define,

$$Adv^{\mathrm{SD}}_{(n,q,R,\omega)}(t) \stackrel{\triangle}{=} \max_{\mathcal{A}:|\mathcal{A}|=t} Adv^{\mathrm{SD}}_{(n,q,R,\omega)}(\mathcal{A}).$$

SD is the problem upon which all code-based cryptography relies. It is known to be NP-complete [5] and is conjectured to be hard on average for a large set of parameters. This problem has been studied for a long time [24, 28, 15, 2, 16, 21, 3, 8, 22, 14, 6, 7] and the best solutions for solving it on average are off complexity (up to a polynomial factor) $2^{n\alpha(q,R,\omega)}$ where for a constant ω,

$$\begin{cases} \alpha(q,R,\omega) = 0 & \text{if } \omega \in \left(\frac{q-1}{q}(1-R), R + \frac{q-1}{q}(1-R)\right) \\ \alpha(q,R,\omega) \in \mathbb{R}_+^* & \text{otherwise} \end{cases}$$

while for $\omega = o(1)$ we have $\alpha(q,R,\omega) = -\omega \log_2(1-R)$. In this way we can easily build a one-way function from SD for carefully chosen parameters. However, the classical way to introduce a trapdoor in such functions is to choose parity-check matrices with a special structure (and thus define structured codes) that is hidden and which enables to solve SD for parameters where the problem is generically hard.

The second generic problem that we will consider relates to the, so called, Decoding One Out of Many (DOOM) problem. This problem was first considered in [20] and later analyzed in [25]. It corresponds to an instance of SD where for a matrix $\mathbf{H}$ we have various syndromes $\mathbf{s}$ and we are interested in solving only one of them, namely,

---

Problem 3 (Decoding One Out of Many - $\mathrm{DOOM}(n,q,R,\omega,N)$*).*

- Instance: a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ of rank $n-k$ and N syndromes $\mathbf{s}_1, \cdots, \mathbf{s}_N$,
- Output: $\mathbf{e} \in \mathbb{F}_q^n$ of Hamming weight w and $i \in [\![1, N]\!]$ such that $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}_i$

where $k \stackrel{\triangle}{=} \lfloor Rn \rfloor$ and $w \stackrel{\triangle}{=} \lfloor \omega n \rfloor$.

The advantage of breaking DOOM for uniformly distributed outputs is then defined as follows.

Definition 6 (DOOM-advantage(n,q,R,ω,N)**).** *Let* $k \stackrel{\triangle}{=} \lfloor Rn \rfloor$ *and* $w \stackrel{\triangle}{=} \lfloor \omega n \rfloor$*. For any algorithm* $\mathcal{A}$*, we define,*

$$Adv^{\mathrm{DOOM}}_{(n,q,R,\omega,N)}(\mathcal{A}) \stackrel{\triangle}{=} \mathbb{P}\Big(\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}_i \text{ and } |\mathbf{e}| = w : \mathbf{H} \stackrel{\$}{\hookleftarrow} \mathbb{F}_q^{(n-k)\times n}, \\ \mathbf{s}_1, \cdots, \mathbf{s}_N \stackrel{\$}{\hookleftarrow} \mathbb{F}_q^{n-k},\ (\mathbf{e}, i) \leftarrow \mathcal{A}(\mathbf{H}, \mathbf{s})\Big),$$

and for any time t*, we also define,*

$$Adv^{\mathrm{DOOM}}_{(n,q,R,\omega,N)}(t) \stackrel{\triangle}{=} \max_{\mathcal{A}:|\mathcal{A}|=t} Adv^{\mathrm{DOOM}}_{(n,q,R,\omega,N)}(\mathcal{A}).$$

This problem is in the algorithmic state-of-the art of exponential complexity for the same parameters as SD. However, we know how to get some exponential gains when ω is constant. The best achievement is for relative weights ω close to the Gilbert-Varshamov bound where we typically expect one solution in an instance of SD while this improvement vanishes when the number of solution that we expect is growing. For instance in code-based signatures like Surf [11] or Wave [12], where parameters are such that we typically expect an exponential number of solutions, there are no advantage to solve DOOM instead of SD [11, 7].

Let us finish now by giving the collision problem in the paradigm of code-based cryptography.

Problem 4 (Collision Decoding - $\mathrm{CD}(n,q,R,\omega)$*).*

- Instance: a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ of rank $n-k$,
- Output: $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{F}_q^n$ of Hamming weight w such that $\mathbf{e}_1\mathbf{H}^\intercal = \mathbf{e}_2\mathbf{H}^\intercal$

where $k \triangleq \lfloor Rn \rfloor$ and $w \triangleq \lfloor \omega n \rfloor$.

While SD and DOOM are hard for a large class of parameters, this is not the case for CD as shown by the following proposition.

Proposition 7. *Let q, R, ω which verify*

$$\omega \geq \frac{q-1}{q}(1-R) \quad \textit{and} \quad q \geq 3. \tag{7}$$

Then, it exists a polynomial-time in n algorithm which solves on average (over $\mathbf{H}$) $\mathrm{CD}(n, q, R, \omega)$.

Proof. The idea is to first compute a codeword $\mathbf{c} \in \mathbb{F}_q^n$, *i.e*: $\mathbf{c}\mathbf{H}^\intercal = \mathbf{0}$ where $k \triangleq \lfloor Rn \rfloor$. Then we can easily build $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{F}_q^n$ s.t $\mathbf{e}_1\mathbf{H}^\intercal = \mathbf{e}_2\mathbf{H}^\intercal$ and of Hamming weight $w \geq |\mathbf{c}|$. Indeed, it is always possible to choose for i with $\mathbf{c}(i) \neq 0$, $\mathbf{e}_1(1) - \mathbf{e}_2(i) = \mathbf{c}(i)$ and $\mathbf{e}_1(i), \mathbf{e}_2(i) \neq 0$ as $q \geq 3$. Then for other $w - |\mathbf{c}|$ positions we complete as $\mathbf{e}_1(i), \mathbf{e}_2(i) \neq 0$ and $\mathbf{e}_1(i) - \mathbf{e}_2(i) = 0$ and for the last positions we put 0's.

Let us show that we can reach (asymptotically with n) relative weight of codewords $|\mathbf{c}|/n = (q-1)/q(1-R)$ in polynomial time. Let $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ be uniformly distributed. For almost all matrices $\mathbf{H}$ we can make a Gaussian elimination in polynomial-time to put them into the form (up to a permutation)

$$\left(\mathbf{1}_{n-k}\ \mathbf{H}'\right) \quad \mathbf{H}' \in \mathbb{F}_q^{(n-k)\times k} \quad \text{and} \quad \mathbf{1}_{n-k} \text{ the identity with } n-k \text{ one.}$$

It is easily verified that very rows of the matrix $\left(\mathbf{1}_k\ \mathbf{H}'^\intercal\right)$ forms a codeword of typical weight $(q-1)/q(n-k)$ as the matrix $\mathbf{H}'^\intercal$ is random which concludes the proof. □

8.2 The Wave-ATPSF Family

In this context authors of [12] gave a collection of ATPSF which is called Wave. To accomplish this they introduced a family of codes which forms their trapdoor, namely the permuted generalized $(U, U+V)$-codes. The presentation of this trapdoor is out of the scope of this paper. What we only need to know here is the following fact.

Fact 1 *Let $n \in \mathbb{N}$, $R, \omega \in (0,1)$, q a field size, $k \triangleq \lfloor Rn \rfloor$ and $w \triangleq \lfloor \omega n \rfloor$. It exists a polynomial-time application (in n)*

$$\tau \in T \mapsto \mathbf{H}_\tau \in \mathbb{F}_q^{(n-k)\times n}$$

such that: given $(\mathbf{H}_\tau, \tau)$, for every $\mathbf{s} \in \mathbb{F}_q^{n-k}$ we can find in polynomial-time (in n) $\mathbf{e}$ of Hamming weight w verifying,

$$\mathbf{e}\mathbf{H}_\tau^\intercal = \mathbf{s}.$$

In the following we will denote by:

$$\mathrm{Par}_{\mathrm{UV}}(n, q, R, \omega) \triangleq \{(\tau, \mathbf{H}_\tau) : \tau \in T\}$$

Definition 7 (Wave-ATPSF). *Let (n, q, R, ω) be the parameters of Wave which are functions of the security parameters λ [13, §7.3]. Let $k \triangleq \lfloor Rn \rfloor$ and $w \triangleq \lfloor \omega n \rfloor$. We define,*

- $\textsc{TrapGenWave}(1^\lambda)$ *outputs $(f_{w,\mathbf{H}_\tau}, \tau)$ with $(\tau, \mathbf{H}_\tau) \in \mathrm{Par}_{\mathrm{UV}}(n, q, R, \omega)$ and*

$$\begin{array}{rccc} f_{w,\mathbf{H}_\tau} : & \left\{\mathbf{x} \in \mathbb{F}_q^n : |\mathbf{x}| = w\right\} & \longrightarrow & \mathbb{F}_q^{n-k} \\ & \mathbf{e} & \longmapsto & \mathbf{e}\mathbf{H}_\tau^\intercal \end{array}$$

- SampDomWave$(\cdot)$ *denotes the uniform distribution over words of Hamming weight* w *in* $\mathbb{F}_q^n$,
- SampPreWave$(f_{w,\mathbf{H}_\tau}, \tau, \mathbf{s})$ *for* $\mathbf{s} \in \mathbb{F}_q^{n-k}$ *is defined in [12, §5]*

The Wave family is proved to be an ε-ATPSF for $\varepsilon \in \mathrm{negl}(\lambda)$ in [13, Theorem 1].

Now as explained in [13], some parity-check matrices $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ cannot be written as $\mathbf{H}_\tau$ for $\tau \in T$. It is due to the fact codes defined by parity-check matrices $\mathbf{H}_\tau$ form a sub-family of all the linear code. Therefore, inverting $f_{w,\mathbf{H}_\tau}$ doesn't amount to solve $\mathrm{SD}(n,q,R,\omega)$ and the one-wayness property of Wave doesn't reduce directly to the syndrome decoding problem. However, it is classic in code-based cryptography (see for instance [26]), where a trapdoor is used, to split the difficulty of the considered decoding problem to: (i) distinguish between the used code and a random code and (ii) the SD problem. This is summarized for Wave in the proposition which follows. Let us first start by introducing the advantage to distinguish between a random code and a code used in Wave.

Definition 8 (UVtrapdoor-advantage(n,q,R,ω)). *Let* $k \triangleq \lfloor nR \rfloor$. *For any algorithm* $\mathcal{A}$, *we define*

$$Adv^{\mathrm{UVTrap}}_{(n,q,R,\omega)}(\mathcal{A}) \triangleq \mathbb{P}\left(\mathcal{A}(\mathbf{H}) = 1 : \mathbf{H} \xleftarrow{\$} \mathbb{F}_q^{(n-k)\times n}\right) - \mathbb{P}\left(\mathcal{A}(\mathbf{H}_\tau) = 1 : (\tau, \mathbf{H}_\tau) \xleftarrow{\$} \mathrm{Par}_{\mathrm{UV}}(n,q,R,\omega)\right)$$

For any time t, *we also define*

$$Adv^{\mathrm{UVTrap}}_{(n,q,R,\tau)}(t) \triangleq \max_{\mathcal{A}:|\mathcal{A}|=t} Adv^{\mathrm{UVTrap}}_{(n,q,R,\omega)}(\mathcal{A})$$

Proposition 8. *Let* $\mathcal{F}_{\mathrm{Wave}}$ *be the family Wave of* ATPSF. *We have for parameters* $(n,q,R\omega)$ *and any time* t,

$$Adv^{OW}_{\mathcal{F}_{\mathrm{Wave}}}(t) \leq Adv^{\mathrm{UVTrap}}_{(n,q,R,\omega)}(t) + Adv^{\mathrm{SD}}_{(n,q,R,\omega)}(t)$$

$$Adv^{Claw(RF)}_{\mathcal{F}_{\mathrm{Wave}}}(t,N) \leq Adv^{\mathrm{UVTrap}}_{(n,q,R,\omega)}(t) + Adv^{\mathrm{DOOM}}_{(n,q,R,\omega,N)}(t)$$

Proof. It is enough here to apply the triangular inequality and to remark that in $Adv^{\mathrm{DOOM}}_{(n,q,R,\omega,N)}(t)$ syndromes are uniformly distributed, *i.e*: are output by a random function as in $Adv^{Claw(RF)}_{\mathcal{F}_{\mathrm{Wave}}}(t,N)$. □

In this way our security reduction can be stated in the case of Wave as:

Theorem 3. *Let* $\mathcal{F}_{\mathrm{Wave}}$ *be the collection* Wave *of* ε-ATPSF *with security parameter* λ, *parameters* (n,q,R,ω) *and* $k \triangleq \lfloor nR \rfloor$. *Let* $\mathrm{S}^{\mathcal{F}}_{\mathrm{Wave}}$ *be the associated Hash and Sign signature scheme by taking* $\lambda_0 = \lambda + 2\log(q_{\mathrm{sign}}) + \log(q_{\mathrm{hash}})$. *We have in the ROM*

$$Adv^{\text{EUF-CMA}}_{\mathrm{S}^{\mathcal{F}}_{\mathrm{Wave}}}(t, q_{\mathrm{sign}}) \leq Adv^{\mathrm{UVTrap}}_{(n,q,R,\omega)}(\widetilde{O}(t)) + Adv^{\mathrm{DOOM}}_{(n,q,R,\omega,q_{\mathrm{hash}})}(\widetilde{O}(t)) + q_{\mathrm{sign}}\varepsilon + \frac{1}{2^\lambda}.$$

Proof. It is enough here to apply Theorem 1 and Proposition 8. □

Here our choice to reduce the security into $\mathrm{DOOM}(n,q,R,\omega)$ instead of $\mathrm{CD}(n,q,R,\omega)$ at the same distance is crucial. Indeed, let us take a look on the practical Wave parameters. As defined in [13, §7.3] we have,

$$n = 66.34\lambda, \quad \omega = 0.9396, \quad q = 3, \quad R = \frac{R_U + R_V}{2}$$

with,

$$R_U = 0.8379 \quad \text{and} \quad R_V = 0.4821.$$

This gives

$$\frac{q-1}{q}(1-R) \approx 0.23 \ll \omega.$$

Therefore, by applying Proposition 7 ($q = 3$) it would be easy to break the collision for the Wave parameters. However, it could be argued that parameters may be chosen to avoid this. Nevertheless, it is impossible in the case of Wave, as explained, SampPreWave necessary works for high relative weights ω, namely $\omega \geq \frac{(q-1)}{q}(1-R)$ which is exactly the condition of Proposition 7.

9 Conclusion

In this paper, we extended the GPV construction of signature schemes by allowing the use of ATPSF instead of TPSF. We also presented a security reduction of these signature schemes to the Claw(RF) which is not only tight but also optimal, meaning that an algorithm that solves the Claw(RF) also breaks the underlying signature scheme.

Our results allow to extend the GPV construction to non-lattice based schemes. In particular, for code-based cryptography, it is often easy to find collisions for the underlying trapdoor function. What we showed is that with this construction, we cannot have a tight reduction to Syndrome Decodings so we cannot ignore the non-tightness to SD. On the other side, losing a q factor in the security reduction is greatly overkill. The good approach is to consider the Claw(RF) problem which in the code-based setting is the DOOM problem. Because of our optimality results, the Claw(RF) should always be the studied problem in GPV-like construction in order to correctly assess the security of the signature scheme.

More generally, we advocate that all Hash and Sign signatures should follow similar guidelines. This was implicitly done for lattices because SIS and $ISIS$ are considered of same difficulty and the associated Claw(RF) problem lies between them.

References

1. Personal communication with Damien Stehlé.
2. Alexander Barg. Minimum distance decoding algorithms for linear codes. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 12th International Symposium, AAECC-12, Toulouse, France, June 23-27, 1997, Proceedings*, volume 1255 of *LNCS*, pages 1–14. Springer, 1997.
3. Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In *Advances in Cryptology - EUROCRYPT 2012*, LNCS. Springer, 2012.
4. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures-how to sign with rsa and rabin. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *LNCS*, pages 399–416. Springer, 1996.
5. Elwyn Berlekamp, Robert McEliece, and Henk van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Inform. Theory*, 24(3):384–386, May 1978.
6. Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for LPN security. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography 2018*, volume 10786 of *LNCS*, pages 25–46, Fort Lauderdale, FL, USA, April 2018. Springer.
7. Rémi Bricout, André Chailloux, Thomas Debris-Alazard, and Matthieu Lequesne. Ternary syndrome decoding with large weights. preprint, February 2019. arXiv:1903.07464, to appear in the proceedings of SAC 2019.
8. Rodolfo Canto-Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight, 2015. preprint.
9. Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, pages 272–287, 2002.

10. Jan Czajkowski, Christian Majenz, Christian Schaffner, and Sebastian Zur. Quantum lazy sampling and game-playing proofs for quantum indifferentiability. Cryptology ePrint Archive, Report 2019/428, 2019. `https://eprint.iacr.org/2019/428`.
11. T. Debris-Alazard, N. Sendrier, and J.-P. Tillich. The problem with the surf scheme. preprint, November 2017. arXiv:1706.08065.
12. Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In *Advances in Cryptology - ASIACRYPT 2019*, LNCS, Kobe, Japan, December 2019.
13. Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. Cryptology ePrint Archive, Report 2018/996, March 2019. `https://eprint.iacr.org/2018/996`.
14. Thomas Debris-Alazard and Jean-Pierre Tillich. Statistical decoding. preprint, January 2017. arXiv:1701.07416.
15. Ilya Dumer. On minimum distance decoding of linear codes. In *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, pages 50–52, Moscow, 1991.
16. Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In M. Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 88–105. Springer, 2009.
17. Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. First round submission to the NIST post-quantum cryptography call, November 2017.
18. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206. ACM, 2008.
19. Shafi Goldwasser and Daniele Micciancio. *Complexity of Lattice Problems: A Cryptographic Perspective*, volume 671 of *Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, March 2002.
20. Thomas Johansson and Fredrik Jönsson. On the complexity of some cryptographic problems based on the general decoding problem. *IEEE Trans. Inform. Theory*, 48(10):2669–2678, October 2002.
21. Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $O(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 107–124. Springer, 2011.
22. Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 203–228. Springer, 2015.
23. Nist. Post-quantum cryptography standardization, 2017. https://csrc.nist.gov/projects/post-quantum-cryptography.
24. Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.
25. Nicolas Sendrier. Decoding one out of many. In *Post-Quantum Cryptography 2011*, volume 7071 of *LNCS*, pages 51–67, 2011.
26. Nicolas Sendrier. The tightness of security reductions in code-based cryptography. In *Proc. IEEE Inf. Theory Workshop- ITW 2011*, pages 415–419. IEEE, 2011.
27. Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.
28. Jacques Stern. A method for finding codewords of small weight. In G. D. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *LNCS*, pages 106–113. Springer, 1988.
29. Mark Zhandry. How to construct quantum random functions. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 679–687, Washington, DC, USA, 2012. IEEE Computer Society.