Lattice-based Cryptography and Cryptanalysis

Brice Minaud

email: brice.minaud@ens.fr

MPRI, 2025-2026

A Hash-and-Sign lattice signature

FALCON

Fast-Fourier Lattice-based
Compact Signatures over NTRU

Trapdoor permutation

Trapdoor permutation

Sample: outputs key **k** and *trapdoor* **t**.

Forward: given key **k** and input x, can compute y = F(**k**,x) in PPT.

Inverse: given trapdoor **t** and target y, can compute x such that y = F(**k**,x) in PPT.

Security: given target y, **cannot** compute x such that y = F(**k**,x) in PPT (except with negligible probability of success).

Example: RSA is a trapdoor permutation, with key **N** = pq, trapdoor **d** = $e^{-1} \bmod \phi(N)$, F(**N**,x) = $x^e \bmod N$, F^{-1}(**N**,**d**,y) = $y^d \bmod N$.

Hash-and-Sign Signatures

Hash-and-Sign signature — Given: 'hash' = hash function.

Public key: **pk** = key for trapdoor permutation F (allows to compute F).

Secret key: trapdoor **t** for F (allows to compute F^{-1}).

- **Sign(*m*):** $\sigma = F^{-1}(\text{hash}(m))$, computed using **t**.
- **Verify(*m*,σ):** check $F(\sigma) = \text{hash}(m)$, computed using **pk**.

This blueprint* transforms a *trapdoor permutation* into a *signature scheme*.

⇒ all we need is a lattice-based trapdoor permutation.

Remark: we need the hash function to map into the range of F.

*only a “blueprint” because it does not necessarily yield a secure signature scheme (cf. later).

Short Integer Solution (SIS)

Ajtai ’96 (the foundational article of Lattice-based crypto).

Say I have $m > n$ vectors a_i in $\mathbb{Z}_q^n$.

Problem: find **short** $x = (x_1,\ldots,x_m)$ in $\mathbb{Z}_q^m$ such that $\sum x_i a_i = 0$.
Here, **short** means of small norm: $\|x\| \leq \beta$.

- The crucial point is the norm constraint β. Otherwise this is just a linear system.
- Typically, Euclidian norm, with representatives in $[-q/2, q/2]$.
- Solution must exist as long as there are at least q^n vectors of norm $\leq \beta/\sqrt{2}$, due to collisions. E.g. $\beta > \sqrt{n \log q}$ and $m \geq n \log q$.

Inhomogeneous SIS problem

ISIS problem. Given a uniform matrix $A \in \mathbb{Z}_q^{n \times m}$, and $t \in \mathbb{Z}_q^n$, find $x \in \mathbb{Z}_q^m$ with and $||x|| \leq \beta$ such that $Ax = t$.

ISIS ⇔ finding a preimage for the hash function x ↦ Ax.

⚠️ x is required to be *short*.

⇒ ISIS implies preimage resistance for x ↦ Ax, while SIS implies collision resistance.

Adding a trapdoor

ISIS problem. Given a uniform matrix $A \in \mathbb{Z}_q^{n \times m}$, and $t \in \mathbb{Z}_q^n$, find $x \in \mathbb{Z}_q^m$ with and $||x|| \leq \beta$ such that $Ax = t$.

SIS = finding a *short* vector in $\mathcal{L}^\perp(A)$.

ISIS = finding a *close* vector in $\mathcal{L}^\perp(A)$.

⇒ a good basis of $\mathcal{L}^\perp(A)$ yields a “trapdoor” for SIS and ISIS.

⇒ it gives a trapdoor for x ↦ Ax, turning it into a trapdoor permutation.

GPV/Falcon-style Hash-and-Sign lattice-based signature

Falcon-style signature

Public key: **A** = matrix for which $F_{\mathbf{A}}$: x ↦ **A**x is preimage-resistant.*

Secret key: trapdoor **t** for $F_{\mathbf{A}}$ = good basis of $\mathcal{L}^{\perp}(A)$.*

- **Sign(*m*):** short $\sigma = F_{\mathbf{A}}^{-1}(\text{hash}(m))$, computed using **t**.
- **Verify(*m*,σ):** check $F_{\mathbf{A}}(\sigma) = \text{hash}(m)$, *and* check σ is short.

Given only **A**, forging a signature

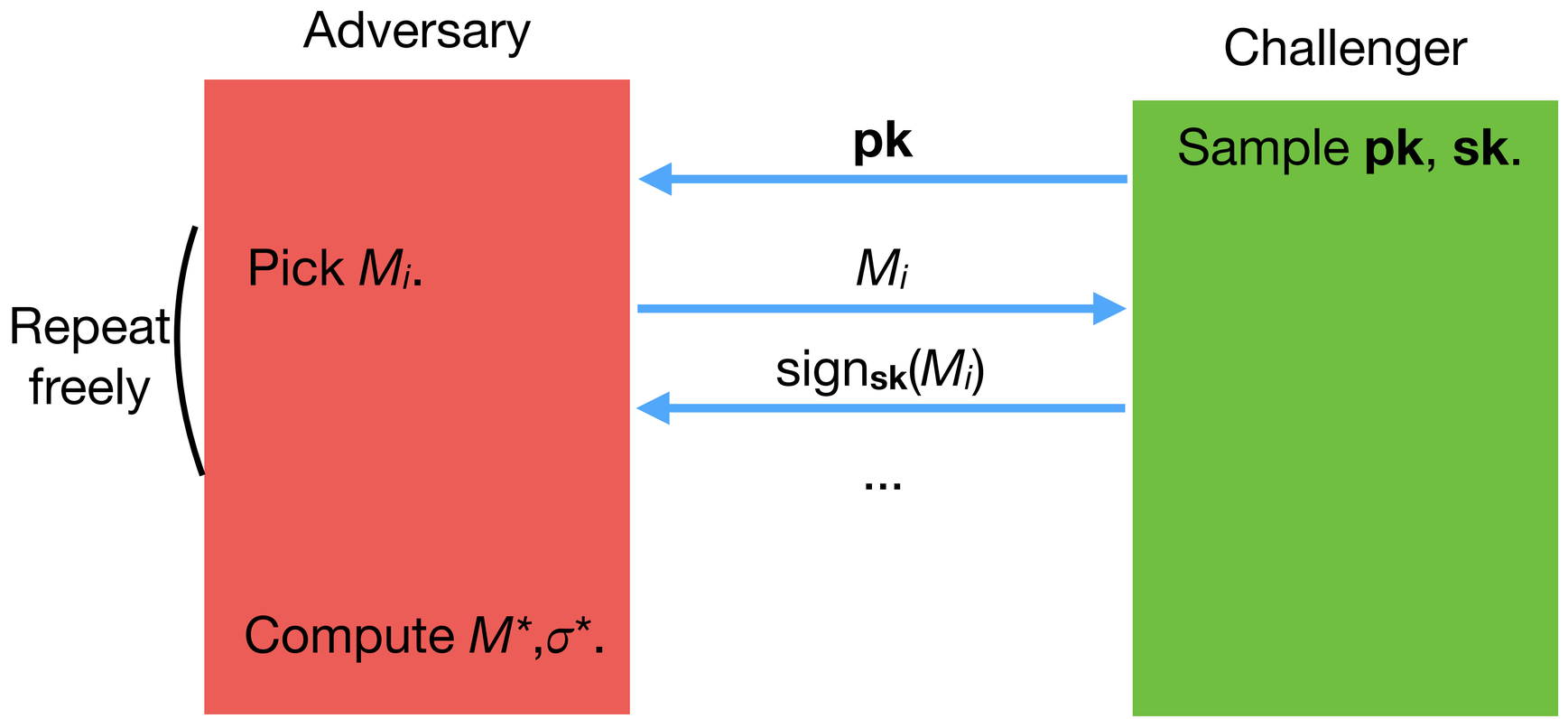
= finding a short preimage of hash(*m*) for the function x ↦ **A**x

= solving ISIS for random target.

But this is not enough for security. *(Why? More to come...)*

*Falcon uses NTRU lattices.

EUF-CMA: existential unforgeability under chosen message attacks



The adversary wins iff $\text{verify}_{\mathbf{pk}}(M^*,\sigma^*)$ = True, and $M^* \notin \{M_i\}$.

The signature scheme is **secure** if no PPT adversary wins, except with negligible probability.

Arguing security for Hash-and-Sign signatures: overview

General idea:

- Without signature oracle, trapdoor permutation security ⇒ unforgeability.

  So all we need is to argue the signature oracle does not leak information.

- To do that, argue (message,signature) pairs can be simulated *without* **sk**.

  Common strategy: show that (x,F(x)) is indistinguishable from $(F^{-1}(y),y)$.

Roughly, in our lattice-based scheme, this means showing:*

"(x,Ax) where x ← short"

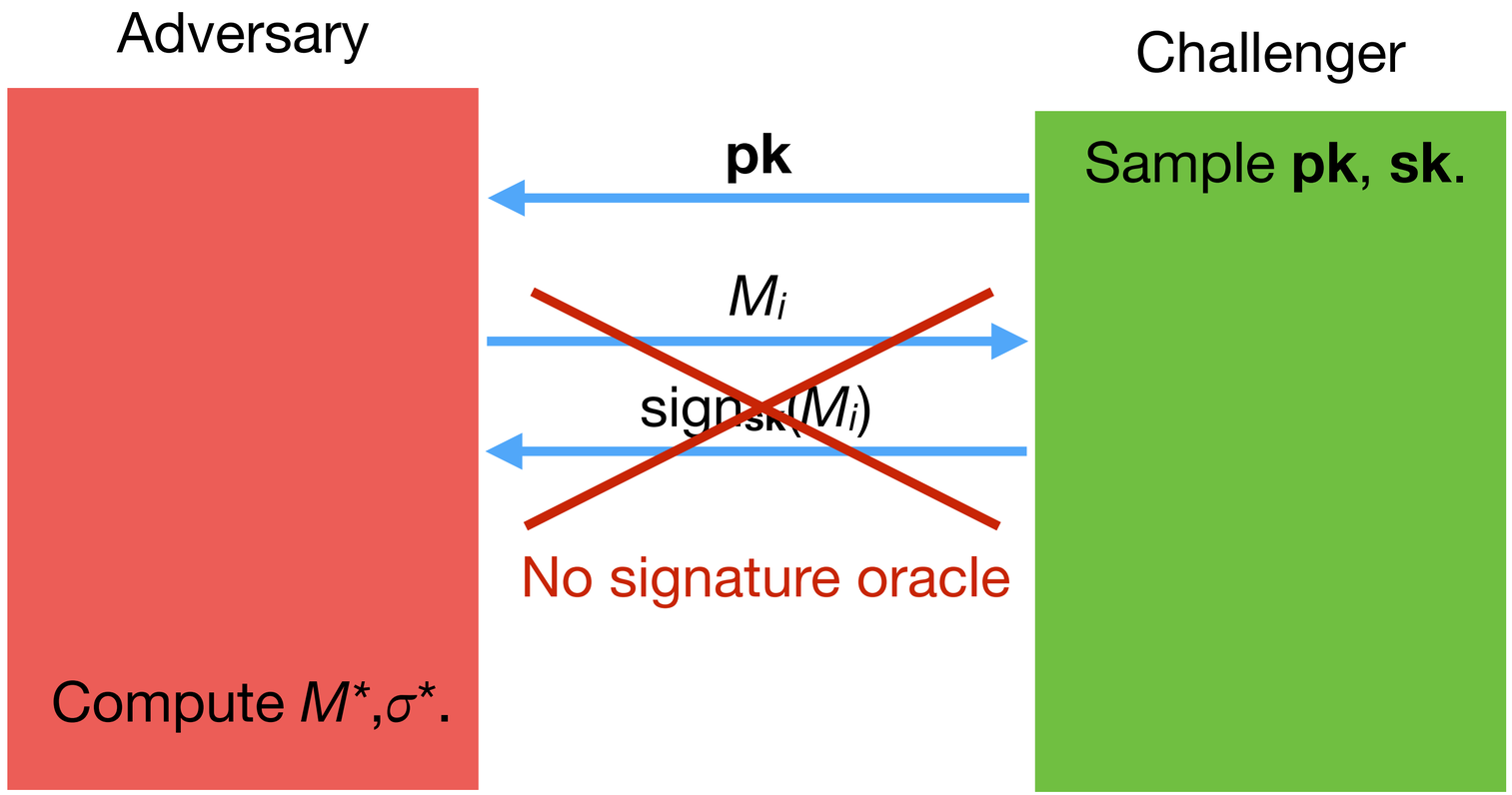
is indistinguishable from:

"$(F^{-1}{}_A(y),y)$ where y ← uniform"

*Falcon uses a form of rejection sampling to enforce this.

Proof structure for Hash-and-Sign signatures

EUF-KO: existential unforgeability with key only



The adversary wins iff $\text{verify}_{\mathbf{pk}}(M^*,\sigma^*)$ = True.

The signature scheme is **secure** if no PPT adversary wins, except with negligible probability. Same as EUF-CMA, without the signature oracle.

Proof structure

Proof outline for Hash-and-Sign signatures: known (loosely) as “Full Domain Hash” security proof.

Roadmap

- *Reduction 1:* intuitively, argue signature oracle reveals no information. Like in ZK, achieved by showing it can be simulated without secret knowledge.
- *Reduction 2:* typically a straightforward reduction to (a multi-target version of) the hard problem underpinning the signature scheme.

Example: recalling RSA signature (for e=3)

RSA signature in Hash-and-Sign framework

Public key: pk = N = pq for p, q large primes. Define F: $x \mapsto x^3 \bmod N$.

Secret key: trapdoor **sk** = d = $3^{-1} \bmod \phi(N)$. Allows to compute $F^{-1} : x \mapsto x^d$.

- **Sign(*m*):** $\sigma = F^{-1}(\text{hash}(m)) = \text{hash}(m)^d \bmod N$, computed using **sk**.
- **Verify(*m*,σ):** check $F(\sigma) = \text{hash}(m)$, computed using **pk**.

Importantly, RSA is built on top of a true trapdoor permutation:

F is a permutation (over $\mathbb{Z}_N^*$, the invertible elements modulo N).

Underlying hard problem: given $x \leftarrow_\$ \mathbb{Z}_N^*$, compute:

$y = F^{-1}(x)$, i.e., find y such that $y^3 = x \bmod N$.

Reduction 1: EUF-CMA → EUF-KO (when F is a permutation)

Intuition first, we'll see formal arguments afterwards.

Intuition 1: the choice of messages M_i queried by the adversary in the EUF-CMA game is irrelevant.

> *Why:* M_i's are immediately processed by the hash function, which maps them to "uniformly random" values.

Corollary: what the adversary really observes from a signature oracle query is $F^{-1}(x)$ for a "uniformly random" x = hash(*m*).

Intuition 2: these observations are useless; they reveal no information.

> *Why:* the adversary can perfectly simulate pairs $(x,F^{-1}(x))$ themselves, by sampling (F(y),y) for uniform y. (Because F is a permutation, these two distributions are the same.)

EUF-CMA → EUF-KO (continued)

Intuitive conclusion:

Learning $(x,F^{-1}(x))$ for uniform x does not help the adversary

⇒ EUF-CMA is the same as EUF-KO.

Now we want to make this intuition precise.

Problem: how can we formalize "hash(*m*) is uniformly random" for a given *m*?

That statement is intuitively true (more than that: in some sense hash functions are designed in order for it to be "true"), but blatantly false in a formal sense, because *hash* is a deterministic map.

Philosophical remarks

Problem: how can we formalize “hash(*m*) is uniformly random” for a given *m*?

- Can argue we are trying to formalize a false statement.
- Very related to the impossibility of formalizing collision resistance for a fixed hash function (like SHA3).

  Hardness is defined by “¬∃ algorithm with such-and-such properties”, but must argue whether the algorithm itself is hard to find ⇒ infinite loop.

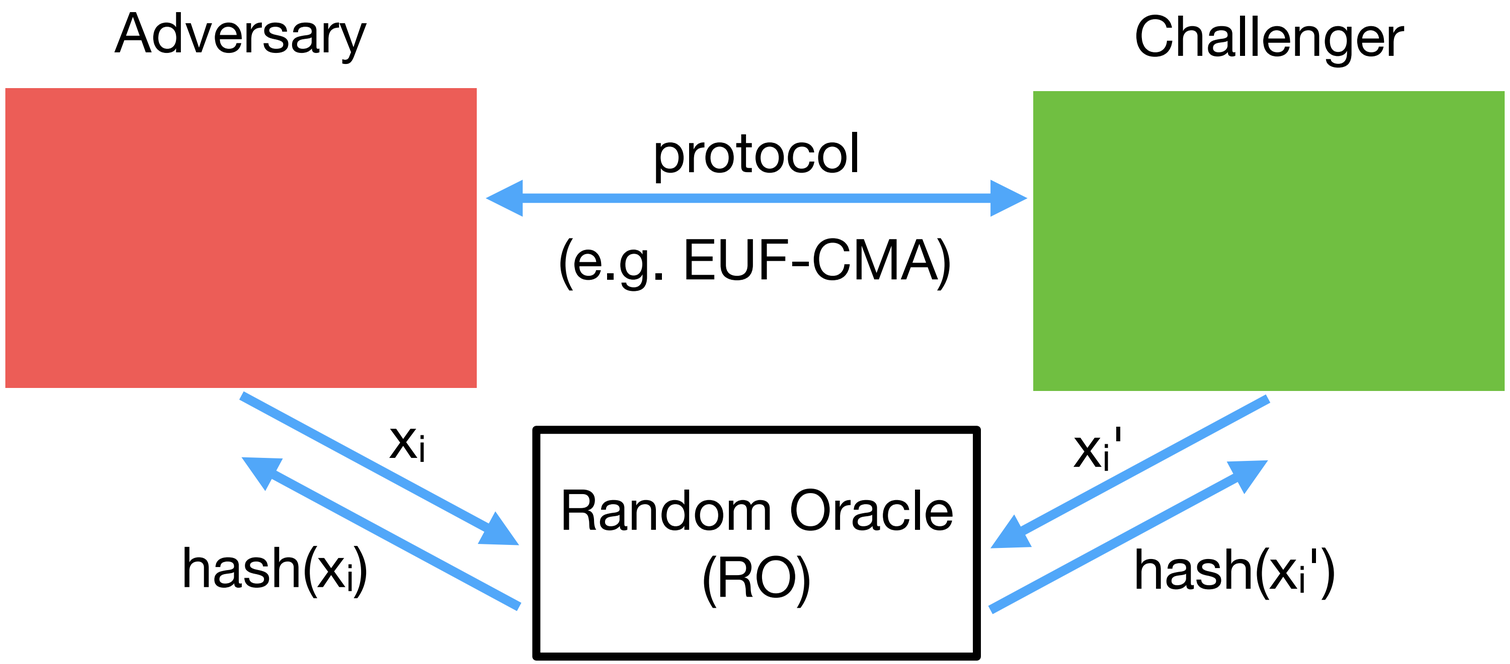
  Can argue the problem lies with the non-algorithmic nature of “∃”.

- If we forgo trying to formalize the above, we miss out on very efficient and desirable cryptographic protocols, that are widely believed to be secure.

  In that sense, we are dealing with an incomplete axiomatic system: there exist “true” statements, such as “FDH is secure”, with no valid proof. The Random Oracle Model can be loosely interpreted as an additional axiom that makes those statements true. In fact, it is too strong: it makes the system inconsistent (can prove false statements).

~~Philosophical~~ pragmatic remarks

In practice, the Random Oracle Model (ROM) is a well-accepted answer to the problem*. (Well-accepted by practitioners.)



RO is a new party. Every party can interact at will with RO by sending x. RO answers with hash(x), where hash(x) is sampled uniformly and independently (except repeated queries).

*In some cases using a PRF in place of the hash function is enough. (Adding entropy fixes the definitional problem.)

Reduction 1: formal version (still with F a permutation)

Equipped with RO, we can transform the earlier intuitive reasoning into a proof.

Every call to *hash* in our Hash-and-Sign definition is now treated as a call to RO.

Sketch of formal reduction EUF-CMA → EUF-KO.

Let A be an EUF-CMA adversary. We will build B an EUF-KO adversary.

How to build B: execute A. In EUF-CMA, A first asks for the public key pk. Forward the public key pk received by B in the EUF-KO game.

A can then query the signature oracle. Wlog (increasing #RO calls if needed), A always queries RO on *m* when querying Sign(*m*). A expects to receive x = hash(*m*) from RO, and $F^{-1}(x)$ from Sign oracle. We simulate this by sampling y u.i.d. and setting hash(*m*) := F(y) (called programming the RO), and answering y to the query Sign(*m*). From A's perspective, this is perfectly indistinguishable from an interaction with the real RO and real Sign oracle. This is because the distribution $(x,F^{-1}(x))$ is identical to the distribution (F(y),y), for uniform x and y, because F is a permutation. Hence A's probability of success does not change.

Finally, A outputs a forged signature in its EUF-CMA game. B outputs the same forged signature for its own EUF-KO game.

Conclusion: probability of success of B = probability of success of A. Running time of B = running time of A + running time of simulating the Sign queries. This last running time must be polynomial because A is PPT (indeed, we need to show that every PPT adversary has a negligible probability of success, hence we need only consider PPT adversaries), so the number of Sign queries must be polynomial, and computing F is polynomial by definition of a trapdoor permutation.

Reduction 1: Falcon version

So far, F was a permutation. Crucial to argue $(F^{-1}(x),x) = (y,F(y))$ as distributions.

Falcon: $F_{\mathbf{A}}: x \mapsto \mathbf{A}x$ maps short vectors in $\mathbb{Z}_q^m$ to (arbitrary) vectors in $\mathbb{Z}_q^n$, $m > n$.

It is surjective (whp) but not injective.

What we need for the previous proof to still work (~ means indistinguishable):

$$(F^{-1}(x),x) \sim (y,F(y))$$

where "F^{-1}" is *some* way to sample a preimage, and x and y come from *some* publicly sampleable distributions.

Formalized more precisely in oft-cited "GPV framework".
[Gentry, Peikert, Vaikuntanathan, STOC 2008]

Sampling for Falcon signature simulatibility

Falcon is built so that its signatures are sampleable in the previous sense.

- **For F^{-1}:** use the Klein sampler*. *(Uses secret key.)*

  Essentially Babai's nearest plane algorithm, where rounding is replaced by one-dimensional discrete Gaussian sampling. Works because

  $$\text{Gaussian(dim = n)} = \text{Gaussian(dim = 1)}^{\otimes n}$$

  (See Phong's slides for more details.)

- **For sampling x:** sample uniformly. *(Publicly sampleable.)*

  No choice here: x is supposed to be a hash output.

- **For sampling y:** sample along a Gaussian in ambient space $\mathbb{Z}_q^m$. *(Publicly sampleable.)*

  With suitable parameters, F(y) is uniform (consequence of famous lemma called “Leftover Hash Lemma”). Moreover, same distribution as $F^{-1}(x)$ for uniform x; this is the point of the Klein sampler.

*The actual Falcon scheme uses an FFO “Fast Fourier” sampler for efficiency, but same behavior.

Reduction 1 for Falcon: conclusion

With all that, we have:

$$(F^{-1}(x),x) \sim (y,F(y))$$

$\Rightarrow$ can simulate real signature queries $(F^{-1}(x),x)$ using simulation $(y,F(y))$.

We still need to prove that replacing $(F^{-1}(x),x)$ by $(y,F(y))$ does not affect the probability of success of the EUF-CMA adversary too much.

Use data processing inequality (see intro slides for second half of the course): suffices to argue that Statistical Distance between $(F^{-1}(x),x)$ and $(y,F(y))$ is negligible.

(We can improve on that using Rényi Divergences, omitted here, but idea is the same: still a data processing inequality.)

Reduction 2: EUF-KO → hard problem

Moving on to the second reduction:

For Falcon, target hard problem = SIS.

Reduction 2_A: Falcon EUF-KO → mISIS

Rougly speaking, Falcon EUF-KO ***is*** multi-target ISIS.

Intuitively: each RO call on message M_i provides one target hash(M_i). Computing F^{-1}(hash(M_i)) for *any* target hash(M_i) yields a forgery.

⇒ an EUF-KO adversary is the same as an mISIS adversary with $r = q_{RO}$ targets,

where q_{RO} is (an upper bound on) the number of RO queries of the EUF-KO adversary.

Formally (sketch): given an EUF-KO adversary, we build an mISIS adversary by programming the RO of the EUF-KO adversary with hash(M_i) = t_i, where the t_i's are the targets of the mISIS adversary, and M_1, ..., M_{qRO}, are the queries to the RO of the EUF-KO adversary. If the EUF-KO adversary succeeds in forging a message, either it was for a message not queried to the RO (probability of success of the forgery = 1/[size of the hash range] = negligible), or it was for a message M_i queried to the RO. The second case yields a valid solution $F^{-1}(M_i)$ to the mISIS adversary. *Conclusion:* the reductions lowers the probability of success by at most a negligible amount (≤ 1/[size of the hash range]).

Reduction 2_A: Falcon EUF-KO → mISIS (cont'd)

We just built a reduction:

$$\text{Falcon EUF-CMA} \rightarrow \text{mISIS}$$

There is also a reduction (exercise, very similar):

$$\text{mISIS} \rightarrow \text{Falcon EUF-CMA}$$

Conclusion: multi-target ISIS captures the difficulty of breaking Falcon *exactly*.

- Both reductions are very efficient and have almost no impact on the probability of success.
- Reduction 1 was information-theoretic, hence it is an “absolute” proof. Unless the parameters are wrong/proof is false, there is no way to exploit the signature oracle.

⇒ In short, Falcon EUF-CMA is information theoretically the same as Falcon EUF-KO, which is the same as multi-target ISIS on the Falcon lattice (= NTRU).

Reduction 2_B: mISIS → ISIS

The following is a standard trick used in Hash-and-Sign proofs, not specific to ISIS.

Given an adversary A against mISIS, we build an adversary B against ISIS.

How to build B (sketch): B forwards the matrix of its ISIS instance to A. B samples r targets for A. One of these targets, chosen uniformly at random, is actually B's target. If A succeeds, A provides an ISIS preimage for one target. With probability *exactly* 1/r, this preimage is for B's target. (Why?)

Conclusion: probability of success of B = (probability of success of A)/r.

Reduction 2_B: mISIS → ISIS (cont'd)

Conclusion: probability of success of B = (probability of success of A)/r.

Does the factor r matter?

Proof perspective: r = number of RO calls = large quantity in practice (albeit polynomial in theory). For NIST candidates, $r = 2^{96}$ ⇒ reduction loses 96 bits of security (prohibitive).

Attack perspective: no known algorithm performs much better for mISIS than for ISIS (there are gains, but fairly small). Heuristically, we assume mISIS has a similar difficulty to ISIS. But this not a proof. There exist problems whose "multi-target" version does actually lose the factor r in real attacks (any problem whose best attack behaves like brute force).

Conclusion: multi-target ISIS captures the hardness of breaking Falcon much more accurately than ISIS (or SIS), at least from a proof perspective.

Reduction 2_C: ISIS → SIS

The security of Falcon is traditionally viewed as relying on SIS.

Given an algorithm A that solves ISIS, we build an algorithm B that solves SIS for the same matrix M.

How to build B (sketch): sample a short input x and query A on the target t = Mx. For information theoretic reasons (there are many preimages of Mx, among which x is not "special"), it is unlikely that A outputs x as its solution. Instead it outputs x' such that Mx' = t = Mx. We get M(x' - x) = 0, thus x'-x is a solution to the SIS problem, for a norm bound roughly twice the norm bound of the ISIS problem.

Remark: the increase in norm bound is not irrelevant. For actual Falcon parameters, it significantly affects the provable security level if we want to reduce all the way to SIS. (In practice, as far as I know, we don't really have better algorithms for ISIS than for SIS; but this is not a proof.)