Lattice-Based Trapdoors and Digital Signatures

Dr. Essam Ghadafi

CyBOK © Crown Copyright, The National Cyber Security Centre 2025, licensed under the Open Government Licence http://www.nationalarchives.gov.uk/doc/open-government-licence/

CyBOK Mapping

The lecture maps to the following CyBOK Knowledge Areas:

- Systems Security → Cryptography
- Infrastructure Security → Applied Cryptography

Outline

- Trapdoor Functions: Constructions and their applications in lattice-based cryptography
- Hash Functions from Lattice Assumptions: Their role in post-quantum cryptography
- Falcon Signature Schem: Construction and security
- Dilithium Signature Scheme: Construction and security

Digital Signatures – Definition

Consists of three algorithms:

- $\mathsf{KeyGen}(1^\kappa)$: Given a security parameter, it generates a key pair $(\mathsf{VK}, \mathsf{SK})$
- $\mathsf{Sign}(\mathsf{SK}, m)$: Produces a signature σ on message m
- $\mathsf{Verify}(\mathsf{VK}, m, \sigma)$: Checks validity of σ on m, outputs 1 (accept) or 0 (reject)

Correctness: For all security parameters κ, for all key pairs $(\mathsf{VK}, \mathsf{SK})$ from KeyGen and all messages m:

$$\mathsf{Verify}(\mathsf{VK}, m, \mathsf{Sign}(\mathsf{SK}, m)) = 1$$

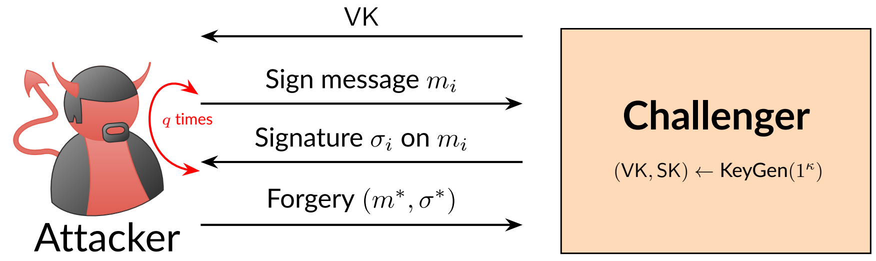
Existential Unforgeability – Intuition

Existential Unforgeability under Chosen-Message Attack (EUF-CMA)

- Attacker can query a signing oracle to get signatures on messages of their choice
- Attacker's goal is to produce a valid signature (forgery) on a new message never queried
- A scheme is EUF-CMA secure if no efficient attacker can succeed with more than negligible probability

Existential Unforgeability

Existential Unforgeabilitiy under Chosen-Message Attack (EUF-CMA)



It is infeasible for an attacker playing the above game to output a valid forgery (m^*, σ^*) where:

- σ^* is a valid signature on m^*, and m^* was not submitted for signing during the game

Signatures from Trapdoor Functions

It is well-known that trapdoor functions yield digital signatures

Intuition: The steps are:

- KeyGen: Create a public key VK and a corresponding private trapdoor td. td is part of SK
- Sign: Use the secret key (trapdoor) to generate a signature
- Verify: Use the public key to verify the signature

Security: Without the secret key (trapdoor), forging a valid signature is infeasible

Lattice-Based Trapdoor Function

Intuition: A Trapdoor function is a function that is easy to compute in one direction but hard to reverse without a secret trapdoor

- Reversing is easy if one has the trapdoor

One can get a trapdoor one-way function for ISIS

- Sample a random matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and a trapdoor $\mathsf{td}_{\mathbf{A}}$
- To compute $f_{\mathbf{A}}(\mathbf{x})$ for $\mathbf{x} \in \mathbb{Z}_q^n$, compute $\mathbf{y} = \mathbf{A}\mathbf{x} \pmod q$ and return $\mathbf{y}$
- To compute $f_{\mathbf{A}}^{-1}(\mathsf{td}_{\mathbf{A}}, \mathbf{y})$ for $\mathbf{y} \in \mathbb{Z}_q^m$, use $\mathsf{td}_{\mathbf{A}}$ to find $\mathbf{x} \in \mathbb{Z}_q^n$ such that $\mathbf{A}\mathbf{x} = \mathbf{y} \pmod q$ and $\|\mathbf{x}\| \leq \gamma$

Hash Functions from Lattice

SIS can be used to construct a collision-resistant hash function $\mathcal{H}$ as shown by Ajtai [1]

To construct $\mathcal{H} : \{0,1\}^m \to \mathbb{Z}_q^n$, set $\mathcal{H}(x) = \mathbf{A}\mathbf{x} \bmod q$ for $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$

Security: The scheme is collision-resistant if $\mathsf{SIS}_{m,n,q,\gamma}$ (for $\gamma = \sqrt{m}$) is hard

Digital Signature from Trapdoors

All that remains is to find a safe way to construct the trapdoor $\mathsf{td}_{\mathbf{A}}$

- To be secure for a signing, $f_{\mathbf{A}}$ must not leak information about the trapdoor td used in the signing, i.e. the secret key

Trapdoor construction

Possible Ways to generate $\mathsf{td}_{\mathbf{A}}$ is to choose $\mathsf{td}_{\mathbf{A}}$ as a good (short) basis of the lattice

$$L^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^n : \mathbf{A}\mathbf{x} = \mathbf{0} \pmod q\}$$

Finding short vectors in $L^{\perp}(\mathbf{A})$ is hard, especially with a random $\mathbf{A}$. Pre-selecting a basis without knowing $\mathbf{A}$ is also difficult

Some methods for generating a matrix $\mathbf{A}$ with a trapdoor $\mathsf{td}_{\mathbf{A}}$

- Ajtai in 1996 introduced a method to construct a random-looking matrix $\mathbf{A}$ with a hidden trapdoor

Leftover Hash Lemma

For a distribution D over $\mathbb{Z}^n$ with min-entropy $H_\infty(D)$, for $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$chosen uniformly at random, for $\mathbf{e} \in \mathbb{Z}_q^n$ sampled from D it holds that

$$\Delta\left((\mathbf{A}, \mathbf{A}\mathbf{e}), (\mathbf{A}, U_m)\right) \leq \epsilon.$$

if

$$H_\infty(D) \geq m \log q + 2 \log \left(\frac{1}{\epsilon}\right),$$

The statistical distance between the 2 distributions is at most ϵ where U_m is the uniform distribution over $\mathbb{Z}_q^m$

The lemma implies that under the above conditions, the output of $\mathbf{A}\mathbf{e}$ appears indistinguishable from a uniformly random vector in $\mathbb{Z}_q^m$

Ajtai Trapdoor

Ajtai gave a construction that generates a close to uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ along with a short trapdoor vector $\mathbf{t}$ such that $\mathbf{t} \in \mathbb{Z}^n$ and $\mathbf{At} = \mathbf{0} \pmod q$

Steps:

- Sample a uniformly random matrix $\mathbf{A}' \in \mathbb{Z}_q^{m\times(n-1)}$
- Sample a uniformly random vector $\mathbf{t}' \in \{0,1\}^{n-1}$

Define:

$$\mathbf{A} = \left[\mathbf{A}' \,||\, -\mathbf{A}'\mathbf{t}'\right] \in \mathbb{Z}_q^{m\times n}$$

$$\text{and} \quad \mathbf{t} = (\mathbf{t}', 1).$$

By the Leftover Hash Lemma, $\mathbf{A}'\mathbf{t}'$ is close to uniform, which also means $\mathbf{A}$ is close to uniform

FALCON Signature Scheme

Falcon: Fast Fourier lattice-based compact signatures over NTRU

- Selected by NIST for standardization in 2022 (to appear as FN-DSA in FIPS 206); as of October 2025, the standard draft is still pending and has not yet been released
- Based on:
  - The GPV [2] lattice trapdoor hash-and-sign framework
  - NTRU lattices [3] for efficiency
    - Works over the polynomial ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$

FALCON Signature Scheme

Let $R = \mathbb{Z}[X]/(X^n + 1)$ and $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ **Key Generation:**

- Choose polynomials $f, F, g, G \in R$ with short coefficients s.t. $fG - gF = q \bmod (X^n + 1)$ (this is the NTRU equation)
  The secret key is SK $= \mathbf{B}$ where:

$$\mathbf{B} = \begin{bmatrix} g & -f \\ G & -F \end{bmatrix}$$

Note: $\mathbf{B}$ is a short basis for $L(\mathbf{B})$
The public key is VK $= \mathbf{A}$ (computed from $\mathbf{B}$):

$$\mathbf{A} = \begin{bmatrix} 1 \\ h \end{bmatrix}$$

where $h \in R_q$ is computed as $h = gf^{-1} \bmod q$
Note: The key satisfies $\mathbf{BA} = \mathbf{0} \bmod q$

FALCON Signature Scheme

Sign: To sign msg using SK $= \mathbf{B}$, derived from polynomials $f, F, g, G \in R$, use a suitable hash function $\mathcal{H}$:

- Compute:

$$c = \mathcal{H}(r||\text{msg}) \in R_q$$

  - r is a random salt, e.g. 320-bit binary string

- Using SK and Fast Fourier Sampling (FFSampling), sample two short polynomials $s_1, s_2 \in R$ s.t.:

$$s_1 + s_2 h = c \pmod q$$

- Output the signature:

$$\sigma = (r, \text{compressed}(s_1, s_2))$$

Note: The pair (s_1, s_2) is compressed to reduce size

FALCON Signature Scheme

Verify:

- To verify a signature $\sigma = (r, (s_1, s_2))$, after decompressing, on msg, using VK $= \mathbf{A}$ (derived from polynomial h):
  - Compute $c = \mathcal{H}(r||\text{msg})$
  - Compute $s_1 = c - s_2 h \bmod q$
  - Verify that s_1 and s_2 are short

FALCON Security & Efficiency

Unforgeability: The scheme is EUF-CMA secure (in the random oracle model) if SIS (over NTRU lattices) is intractable.

Table: Falcon Public Key and Signature Sizes

Variant	PK Size (bytes)	Sig Size (bytes)
Falcon-512 (128-bit security)	897	666
Falcon-1024 (256-bit security)	1,793	1,280

Key Components of Falcon

- Trapdoor Sampling: Uses NTRU lattices instead of general lattices
- Fast Fourier Sampling (FFSampling):
  - Efficient method for discrete Gaussian sampling over lattices
  - Avoids rejection sampling overhead

Dilithium Signature Scheme

Dilithium, standardized by NIST as ML-DSA (FIPS 204) [4], is based on Module-LWE and Module-SIS

Based on the Fiat-Shamir heuristic to transform an interactive ZK proof into a signature scheme

- Signer proves knowledge of SK (LWE secret) and includes the message in the hash used to generate the challenge for the proof of knowledge
- Uses rejection sampling

Secure in the random oracle model

Dilithium Efficiency

Table: Dilithium Public Key and Signature Sizes

Variant	PK Size (bytes)	Sig Size (bytes)
Dilithium-2 (128-bit security)	1,312	2,420
Dilithium-3 (192-bit security)	1,952	3,293
Dilithium-5 (256-bit security)	2,592	4,595

Comparison of Falcon and Dilithium

At 128-bit security:

Scheme	Signature Size	Hard Problem
Falcon	666 bytes	SIS over NTRU Lattice
Dilithium	2420 bytes	M-LWE + M-SIS

In comparison, NIST hash-based SPHINCS+ signature size at the same security level is 17088 bytes

Key Takeaways

- Hash functions from SIS provide post-quantum secure lattice-based hash functions
- Falcon is based on SIS over NTRU lattices, providing fast, compact signatures with post-quantum security
- Dilithium is based on Module-LWE and provides efficient signatures with provable post-quantum security in the random oracle model

References

[1] M. Ajtai. Generating hard instances of lattice problems. In STOC, 1996.

[2] C. Gentry, C. Peikert, V. Vaikuntanathan. How to Use a Short Basis: Trapdoors for Hard Lattices and New Cryptographic Constructions. STOC 2008.

[3] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. Silverman, and W. Whyte. NTRUSIGN: Digital Signatures Using the NTRU Lattice. CT-RSA, 2003.

[4] National Institute of Standards and Technology. Module-Lattice-Based Digital Signature Standard. (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS) NIST FIPS 204, 2024. Available: `https://doi.org/10.6028/NIST.FIPS.204`.