

GPV SIGNATURES (HASH-THEN-SIGN SIGNATURES)

ADVANCED TOPICS IN ~~CYBERSECURITY~~ CRYPTOGRAPHY (7CCSMATC)

Martin R. Albrecht



WIKIPEDIA
The Free Encyclopedia



[Create account](#) [Log in](#)



≡ Falcon (signature scheme)

🌐 1 language ▾

Article [Talk](#)

[Read](#) [Edit](#) [View history](#) [Tools](#) ▾

From Wikipedia, the free encyclopedia

Falcon is a [post-quantum signature scheme](#) selected by the [NIST](#) at the fourth round of the [post-quantum standardisation](#) process. It was designed by Thomas Prest, Pierre-Alain Fouque, [Jeffrey Hoffstein](#), Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang.^{[1][2][3]} It relies on the hash-and-sign technique over the [Gentry](#), Peikert, and [Vaikuntanathan](#) framework^[4] over [NTRU lattices](#). The name *Falcon* is an [acronym](#) for ***F**ast **F**ourier **L**attice-based **c**ompact signatures over **N**TRU*.

OUTLINE

Lattices

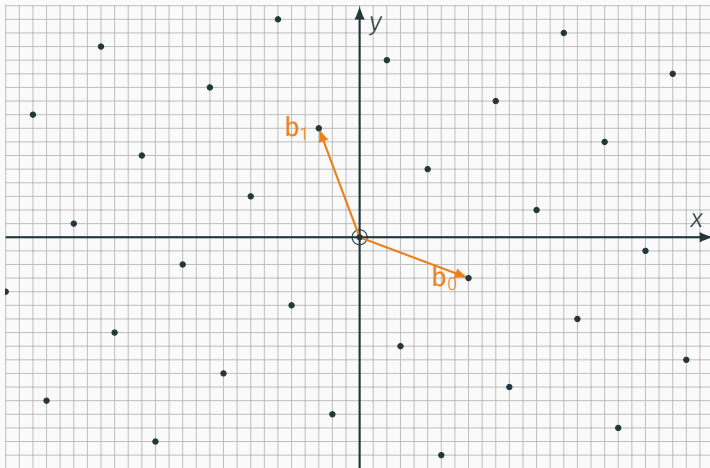
q-ary Lattices

GPV Signatures

Security Proof

LATTICES

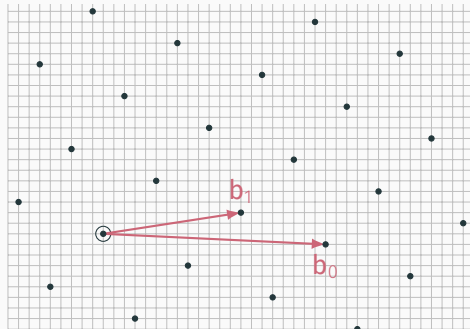
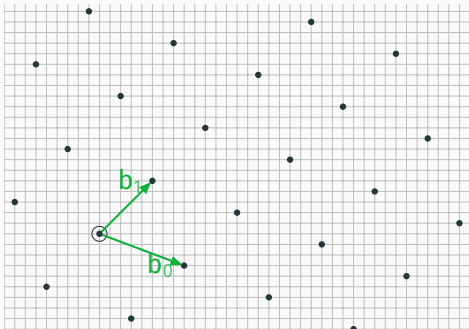
LATTICES



A lattice Λ is a discrete subgroup of \mathbb{R}^d .

Picture credit: Léo Lucas

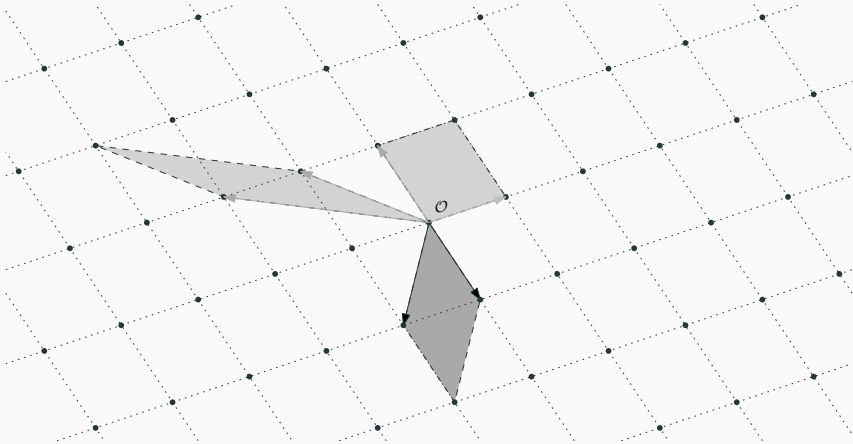
LATTICE BASES



G	\rightarrow	B	: easy	(compute Hermite Normal form);
B	\rightarrow	G	: hard	(BKZ, lattice sieve ...).

LATTICE VOLUME I

The volume of a lattice is the volume of its fundamental domain.



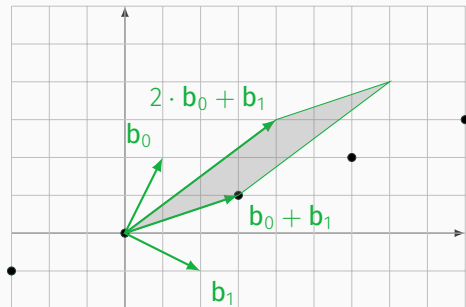
LATTICE VOLUME II

For any two bases \mathbf{G}, \mathbf{B} of the same lattice Λ :

$$\det(\mathbf{G} \cdot \mathbf{G}^T) = \det(\mathbf{B} \cdot \mathbf{B}^T).$$

We can therefore define:

$$\text{Vol}(\Lambda) = \sqrt{\det(\mathbf{B} \cdot \mathbf{B}^T)}.$$



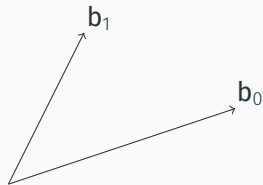
Picture credit: <https://tex.stackexchange.com/a/42559>

LENGTH OF GRAM-SCHMIDT VECTORS

It will be useful to consider the lengths of the Gram-Schmidt vectors.

The vector \mathbf{b}_i^* is the orthogonal projection of \mathbf{b}_i to the space spanned by the vectors $\mathbf{b}_0, \dots, \mathbf{b}_{i-1}$.

Informally, this means taking out the contributions in the directions of previous vectors $\mathbf{b}_0, \dots, \mathbf{b}_{i-1}$.

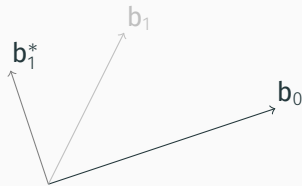


LENGTH OF GRAM-SCHMIDT VECTORS

It will be useful to consider the lengths of the Gram-Schmidt vectors.

The vector \mathbf{b}_i^* is the orthogonal projection of \mathbf{b}_i to the space spanned by the vectors $\mathbf{b}_0, \dots, \mathbf{b}_{i-1}$.

Informally, this means taking out the contributions in the directions of previous vectors $\mathbf{b}_0, \dots, \mathbf{b}_{i-1}$.



GRAM-SCHMIDT VECTORS AND LATTICE VOLUMES

Let \mathbf{B}^* be the Gram-Schmidt Orthogonalisation of \mathbf{B} . The matrix \mathbf{B}^* is **not** a basis for Λ , but

$$\text{Vol}(\Lambda) = \sqrt{\det(\mathbf{B}^* \cdot \mathbf{B}^{*\top})} = \prod \|\mathbf{b}_i^*\|.$$

“GOOD” BASES

Recall that, independently of the basis \mathbf{B} it hold that:

$$\text{vol}(\Lambda) = \prod \|\mathbf{b}_i^*\|.$$

Therefore, it is somehow equivalent that

- $\max_i \|\mathbf{b}_i^*\|$ is small
- $\min_i \|\mathbf{b}_i^*\|$ is large
- $\kappa(\mathbf{B}) = \max_i \|\mathbf{b}_i^*\| / \min_i \|\mathbf{b}_i^*\|$ is small

Good Basis

$$\kappa(\mathbf{G}) = \text{poly}(d), \quad \forall i : \|\mathbf{g}_i^*\| = \text{poly}(d) \cdot \text{Vol}(\Lambda)^{1/d}.$$

LLL-reduced Basis

$$\kappa(\mathbf{G}) \approx (1.04)^d, \quad \max_i \|\mathbf{g}_i^*\| \approx (1.02)^d \cdot \text{Vol}(\Lambda)^{1/d}.$$

EXAMPLE GRAM-SCHMIDT “SHAPE”

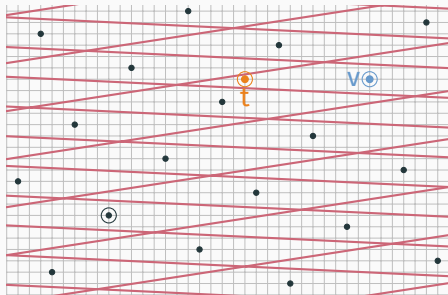
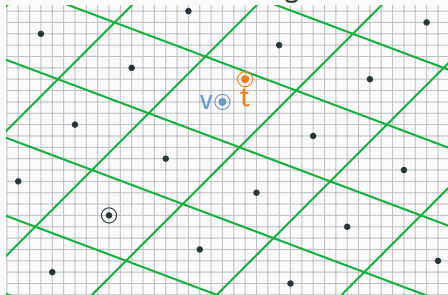
```
A = IntegerMatrix.random(120, "qary", k=60, bits=20)[::-1]
M = GS0.Mat(A, update=True)
line([(i, log(r_, 2)/2) for i, r_ in enumerate(M.r())], **plot_kwds)
```

EXAMPLE GRAM-SCHMIDT “SHAPE” AFTER LLL

```
A = LLL.reduction(A)
M = GS0.Mat(A, update=True)
line([(i, log(r_, 2)/2) for i, r_ in enumerate(M.r())], **plot_kwds)
```

BASES AND FUNDAMENTAL DOMAINS

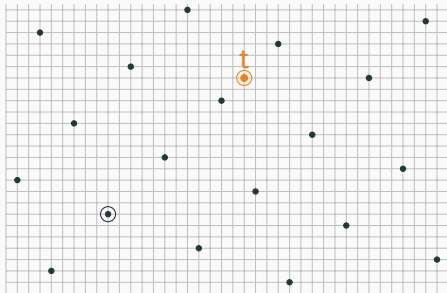
Each basis defines a **tiling**.



Round'off Algorithm [Lenstra, Babai]:

Given a target \mathbf{t} , find $\mathbf{v} \in \Lambda$ at the center the tile.

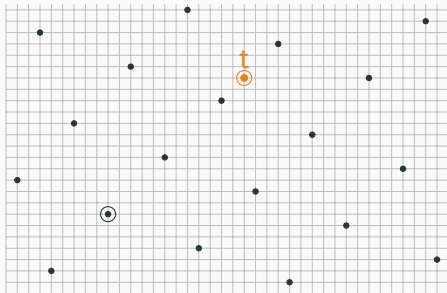
ROUND OFF ALGORITHM



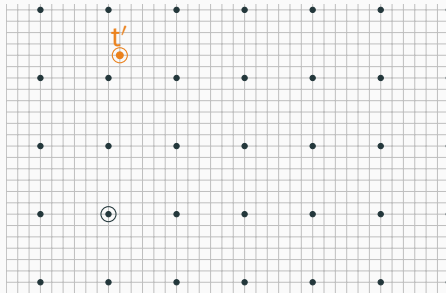
Round Off Algorithm
[Lenstra,Babai]:

Credit: Léo Ducas

ROUND OFF ALGORITHM



$\cdot B^{-1}$

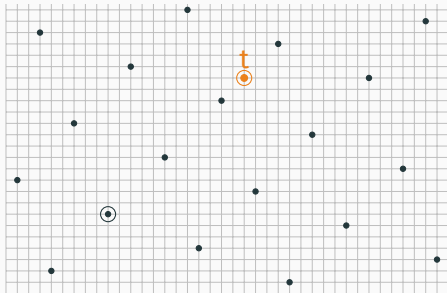


Round Off Algorithm
[Lenstra,Babai]:

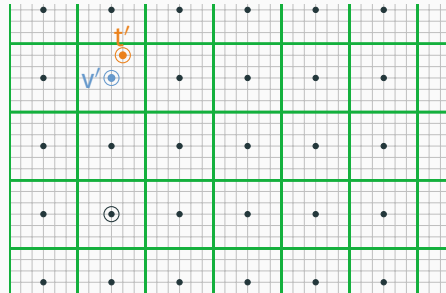
Credit: Léo Ducas

- Use B to switch to the lattice \mathbb{Z}^d $t' = B^{-1} \cdot t$

ROUND OFF ALGORITHM



$\cdot B^{-1}$

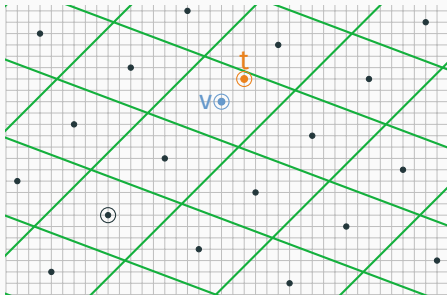


Round Off Algorithm
[Lenstra,Babai]:

Credit: Léo Ducas

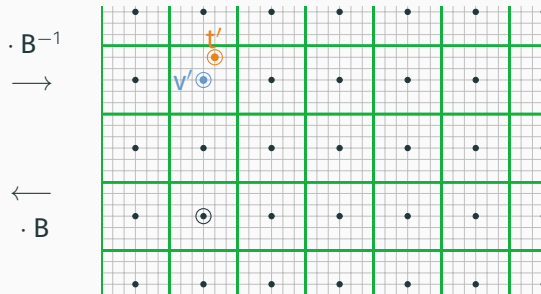
- Use B to switch to the lattice \mathbb{Z}^d $t' = B^{-1} \cdot t$
- Round each coordinate $v' = \lfloor t' \rfloor$

ROUND OFF ALGORITHM



Round Off Algorithm
[Lenstra,Babai]:

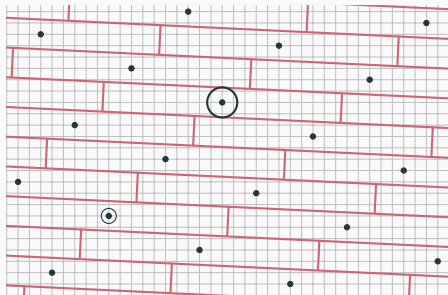
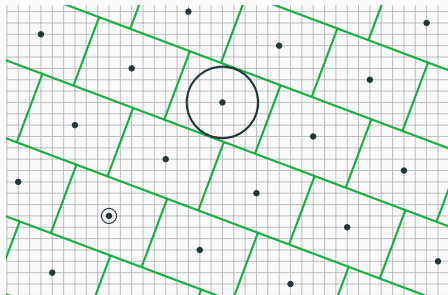
Credit: Léo Ducas



- Use \mathbf{B} to switch to the lattice \mathbb{Z}^d $\mathbf{t}' = \mathbf{B}^{-1} \cdot \mathbf{t}$
- Round each coordinate $\mathbf{v}' = \lfloor \mathbf{t}' \rfloor$
- Switch back to Λ $\mathbf{v} = \mathbf{B} \cdot \mathbf{v}'$

NEAREST-PLANE ALGORITHM

There is a better algorithm (Nearest Plane) based on Gram-Schmidt Orthogonalisation:



- Worst-case distance: $\frac{1}{2}\sqrt{\sum \|b_i^*\|^2}$

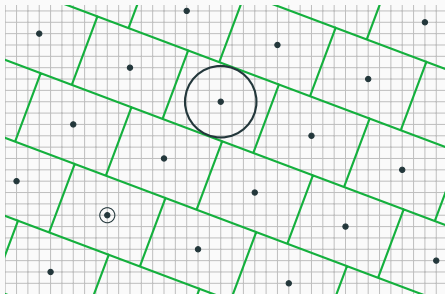
(Approx-CVP)

- Correct decoding of $\mathbf{t} = \mathbf{v} + \mathbf{e}$ for $\mathbf{v} \in \Lambda$ if $\|\mathbf{e}\| \leq \min \|b_i^*\|$

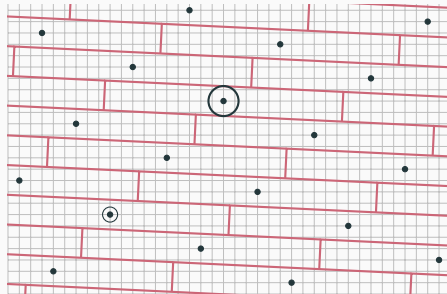
(BDD)

TRAPDOORS FROM LATTICES ?

Good basis **G**: can solve Approx-CVP / BDD.



Bad basis **B**: solving CVP is **hard**.

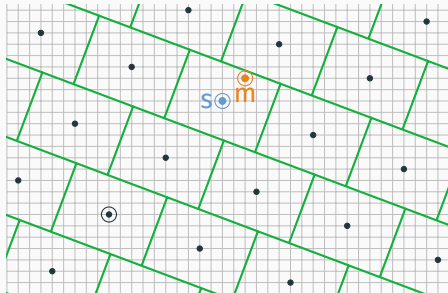


Can we use this as a trapdoor?

SIGNATURES

Sign:

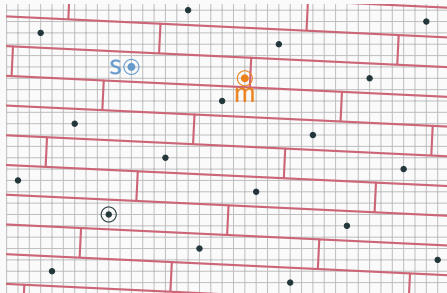
1. Hash the message to a random vector \mathbf{m} .
2. Apply Nearest Plane with a good basis \mathbf{G} : find $\mathbf{s} \in \Lambda$ close to \mathbf{m} .



Credit: Léo Ducas

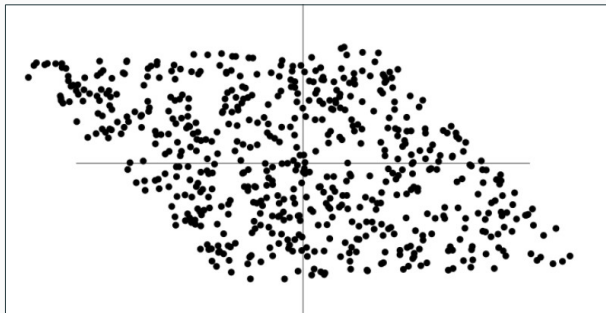
Verify:

1. Check that $\mathbf{s} \in \Lambda$ using the bad basis \mathbf{B}
2. Check that \mathbf{m} is close to \mathbf{s} .



A STATISTICAL ATTACK

The difference $\mathbf{s} - \mathbf{m}$ is always inside the parallelepiped spanned by the good basis \mathbf{G} or its Gram-Schmidt Orthogonalisation \mathbf{G}^* :



- Each signatures (\mathbf{s}, \mathbf{m}) leaks information about \mathbf{G} .
- Learning a parallelepiped from few signatures [NR06]
- Total break of original GGH and NTRUSign schemes

The distribution $\mathbf{s} - \mathbf{m}$ can be made independent of \mathbf{G} by randomising the above algorithms:

Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. **Trapdoors for hard lattices and new cryptographic constructions**. In: *40th ACM STOC*. ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 197–206. DOI: [10.1145/1374376.1374407](https://doi.org/10.1145/1374376.1374407)

Q-ARY LATTICES

CONSTRUCTION OF q -ARY LATTICES (PRIMAL / CONSTRUCTION A)

- Let q be a prime integer¹ and $n < m$ two positive integers.
- The matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ spans the q -ary lattice:

$$\Lambda_q(\mathbf{A}) := \{\mathbf{x} \in \mathbb{Z}^m \mid \exists \mathbf{y} \in \mathbb{Z}_q^n, \mathbf{x} \equiv \mathbf{A} \cdot \mathbf{y} \bmod q\} = \mathbf{A} \cdot \mathbb{Z}_q^n + q\mathbb{Z}^m$$

Lattice Parameters

Assuming \mathbf{A} is full-rank:

- $\dim(\Lambda_q(\mathbf{A})) = m$
- $\text{Vol}(\Lambda_q(\mathbf{A})) = q^{m-n}$

¹for simplicity

EXAMPLE

```
q = 7; n = 3; m = 7
A = random_matrix(GF(q), n, m)
B = A.lift().stack(q * identity_matrix(m)); B
```

```
[3 3 2 1 2 6 2]
[2 6 1 2 5 5]
[1 3 6 4 4 3]
[7]
[ 7]
[ 7]
[ 7]
[ 7]
[ 7]
[ 7]
```

```
B.echelon_form()[ :B.rank() ]
```

```
[1 3 4 5 ]
[ 1 2 5 4]
[ 1 3 6 2 2]
[ 7 ]
[ 7 ]
[ 7 ]
[ 7 ]
```

Row-Based

SageMath's convention is “row based”, i.e. it considers combinations of rows not columns. The literature on lattices favours column-based notation, i.e. combinations of columns. You'll need to transpose the above picture to make it consistent with the slide before.

CONSTRUCTION OF q -ARY LATTICES (DUAL / PARITY-CHECK)

- Let q be a prime integer² and $n < m$ two positive integers.
- The matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is the parity-check of the lattice:

$$\Lambda_q^\perp(\mathbf{A}) := \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{A} \cdot \mathbf{x} \equiv \mathbf{0} \bmod q\} = \ker(\mathbf{x} \mapsto \mathbf{A} \cdot \mathbf{x} \bmod q)$$

Lattice parameters

Assuming \mathbf{A} is full-rank:

- $\dim(\Lambda_q^\perp(\mathbf{A})) = m$
- $\text{Vol}(\Lambda_q^\perp(\mathbf{A})) = q^n$

²for simplicity

EXAMPLE

```
q = 7; n = 3; m = 7
A = random_matrix(GF(q), n, m)
K = A.right_kernel().basis_matrix()
B = K.lift().stack(q * identity_matrix(m)); B
```

```
[1      2 6 6]
[ 1      6 3]
[  1  4 1 ]
[    1 4 1 1]
[7      ]
[ 7      ]
[  7      ]
[    7      ]
[      7  ]
[      7  ]
[      7]
```

```
B.echelon_form()[ :B.rank() ]
```

```
[1      2 6 6]
[  1      6 3]
[    1 4 1 ]
[      1 4 1 1]
[      7  ]
[      7  ]
[      7]
```

THE SHORT INTEGER SOLUTION PROBLEM (SIS)

Definition (SIS Assumption)

Given a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, finding a small non-zero $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{A} \cdot \mathbf{x} \equiv \mathbf{0} \pmod{q}$ is hard.³

Lattice Formulation

Solving Approx-SVP in $\Lambda_q^\perp(\mathbf{A})$ is hard.

In [Ajt98], Ajtai established that if solving SIS is easy then solving Approx-SVP for any lattice is also easy.

³Also on a quantum computer!

A SIMPLE APPLICATION OF SIS

Consider the function:

$$f_A : \{0, 1\}^m \rightarrow \mathbb{Z}_q^n, \quad x \mapsto A \cdot x \bmod q$$

SIS \Rightarrow Collision Resistant Hashing and One-Way Function

- Finding collisions is as hard as SIS⁴ (take the difference)

Moreover, if $m \gg n \log q$:

- f_A is highly surjective (many pre-images for any image)
- Finding pre-images is hard (show it!)

⁴Collisions must exist when $m > n \log q$.

GPV SIGNATURES

KeyGen generate a random (looking) matrix \mathbf{A} together with a short basis td for the lattice $\Lambda_q^\perp(\mathbf{A}) := \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{A} \cdot \mathbf{x} \equiv \mathbf{0} \pmod{q}\}$.

Sign Compute $H(m)$.

Find an arbitrary, not-necessarily short, preimage \mathbf{u} : $\mathbf{A} \cdot \mathbf{u} \equiv H(m) \pmod{q}$.

Use the trapdoor td to solve Approx-CVP for \mathbf{u} on $\Lambda_q^\perp(\mathbf{A})$, call it \mathbf{z} . By construction $\mathbf{A} \cdot \mathbf{z} \equiv \mathbf{0} \pmod{q}$.

Output $\mathbf{y} := \mathbf{u} - \mathbf{z}$. By construction $\|\mathbf{y}\|$ is small.

Verify Check that $\mathbf{A} \cdot \mathbf{y} \equiv H(m) \pmod{q}$ and that $\|\mathbf{y}\|$ is small.

SCHEME

KeyGen(1^λ)

$\mathbf{A}, \text{td} \leftarrow \text{TrapGen}(1^n, 1^m, q, \beta)$
return $\text{vk} := \mathbf{A}, \text{sk} := \text{td}$

Sign(m, sk)

$\mathbf{r} \leftarrow \{0, 1\}^\lambda$
 $\mathbf{y} \leftarrow \text{SampPre}(\text{td}, H(m, \mathbf{r}), \beta')$
return $\sigma := (\mathbf{y}, \mathbf{r})$

Verify(vk, σ, m)

return $\|\mathbf{y}\| \stackrel{?}{\leq} \beta' \wedge H(m, \mathbf{r}) \stackrel{?}{=} \mathbf{A} \cdot \mathbf{y}$

- $(\mathbf{A}, \text{td}) \leftarrow \text{TrapGen}(1^n, 1^m, q, \beta)$ takes dimensions $n, m \in \mathbb{N}$, a modulus $q \in \mathbb{N}$ and a norm bound $\beta \in \mathbb{R}$ and generates a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a trapdoor td .

When $\ell > 2m \log q$, the distribution of \mathbf{A} is within $\text{negl}(\lambda)$ statistical distance to the uniform distribution on $\mathbb{Z}_q^{n \times m}$.

- $\mathbf{u} \leftarrow \text{SampD}(1^n, 1^m, \beta')$ outputs an element in $\mathbf{u} \in \mathbb{Z}^m$ with norm bound $\beta' \geq \beta$.

$\mathbf{v} := \mathbf{A} \cdot \mathbf{u} \bmod q$ is within $\text{negl}(\lambda)$ statistical distance to the uniform distribution on \mathbb{Z}_q^n .

- $\mathbf{u} \leftarrow \text{SampPre}(\text{td}, \mathbf{v}, \beta')$ takes a trapdoor td , a vector $\mathbf{v} \in \mathbb{Z}_q^n$ and a norm bound $\beta' \geq \beta$ and samples $\mathbf{u} \in \mathbb{Z}^\ell$ satisfying $\mathbf{A} \cdot \mathbf{u} \equiv \mathbf{v} \bmod q$ and $\|\mathbf{u}\| \leq \beta'$.

\mathbf{u} is within $\text{negl}(\lambda)$ statistical distance to

$\mathbf{u} \leftarrow \text{SampD}(1^n, 1^m, \mathcal{R}, \beta')$ conditioned on $\mathbf{v} \equiv \mathbf{A} \cdot \mathbf{u} \bmod q$.

SECURITY PROOF

EUFCMA IN THE ROM

EUFCMA	$S(m)$	$H(m)$
$\mathcal{Q}, \mathcal{H} \leftarrow \emptyset, \emptyset;$ $vk, sk \leftarrow \Sigma.\text{KeyGen}(1^\lambda);$ $(m^*, \sigma^*) \leftarrow \mathcal{A}^{S, H}(vk);$ $b_0 := (m^*, \cdot) \notin \mathcal{Q}$ $b_1 := \Sigma.\text{Verify}(vk, \sigma^*, m^*) = 1$ return $b_0 \wedge b_1$	$\sigma \leftarrow \Sigma.\text{Sign}(sk, m)$ $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$ return σ	if $m \notin \mathcal{H}$ then $\mathcal{H}[m] \leftarrow \{0, 1\}^\lambda$ return $\mathcal{H}[m]$

$$\text{Adv}_{\mathcal{A}, \Sigma}^{\text{euf-cma}}(\lambda) := \Pr[\text{EUFCMA}_{\Sigma}^{\mathcal{A}}(\lambda) \Rightarrow 1]$$

INLINING GPV SIGNATURES

Game ₀	S(m)	H(m, r)
$\mathcal{Q}, \mathcal{H} \leftarrow \emptyset, \emptyset;$	$\mathbf{r} \leftarrow \$ \{0, 1\}^\lambda$	if $(m, \mathbf{r}) \notin \mathcal{H}$ then
$\mathbf{A}, \text{td} \leftarrow \$ \text{TrapGen}(\dots)$	$\mathbf{y} \leftarrow \$ \text{SampPre}(\text{td}, H(m, \mathbf{r}), \beta')$	$\mathcal{H}[m, \mathbf{r}] \leftarrow \$ \mathbb{Z}_q^n$
$(m^*, (\mathbf{y}^*, \mathbf{r}^*)) \leftarrow \mathcal{A}^{\text{S}, \text{H}}(\text{vk});$	$\sigma := (\mathbf{y}, \mathbf{r})$	return $\mathcal{H}[m, \mathbf{r}]$
$b_0 := (m^*, \cdot) \notin \mathcal{Q}$	$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$	
$b_1 := \ \mathbf{y}^*\ \leq \beta'$	return σ	
$b_2 := H(m^*, \mathbf{r}^*) \equiv \mathbf{A} \cdot \mathbf{y}^*$		
return $b_0 \wedge b_1 \wedge b_2$		

CHANGING THE RO

Game ₁	$S(m)$	$H(m, r)$
$\mathcal{Q}, \mathcal{H}, \mathcal{P} \leftarrow \emptyset, \emptyset, \emptyset;$	$r \leftarrow \$ \{0, 1\}^\lambda$	if $(m, r) \notin \mathcal{H}$ then
$A, td \leftarrow \$ \text{TrapGen}(\dots)$	$y \leftarrow \$ \text{SampPre}(td, H(m, r), \beta')$	$y \leftarrow \$ \text{SampD}(\dots, \beta')$
$(m^*, (y^*, r^*)) \leftarrow \mathcal{A}^{S, H}(vk);$	$\sigma := (y, r)$	$t := A \cdot y \bmod q$
$b_0 := (m^*, \cdot) \notin \mathcal{Q}$	$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$	$\mathcal{H}[m, r] \leftarrow t$
$b_1 := \ y^*\ \leq \beta'$	return σ	$\mathcal{P}[m, r] \leftarrow y$
$b_2 := H(m^*, r^*) \equiv A \cdot y^*$		return $\mathcal{H}[m, r]$
$b_0 \wedge b_1 \wedge b_2$		

By the Leftover Hash Lemma the distributions of Game₀ and Game₁ are statistically close.

BOOKKEEPING

Game ₂	$S(m)$	$H(m, r)$
$\mathcal{Q}, \mathcal{H}, \mathcal{P} \leftarrow \emptyset, \emptyset, \emptyset;$	$r \leftarrow \$ \{0, 1\}^\lambda$	if $(m, r) \notin \mathcal{H}$ then
bad \leftarrow false	if $(m, r) \in \mathcal{H}$ then	$y \leftarrow \$ \text{SampD}(\dots, \beta')$
$A, td \leftarrow \$ \text{TrapGen}(\dots)$	bad \leftarrow true	$t := A \cdot y \bmod q$
$(m^*, (y^*, r^*)) \leftarrow \mathcal{A}^{S, H}(vk);$	$y \leftarrow \$ \text{SampPre}(td, H(m, r), \beta')$	$\mathcal{H}[m, r] \leftarrow t$
$b_0 := (m^*, \cdot) \notin \mathcal{Q}$	$\sigma := (y, r)$	$\mathcal{P}[m, r] \leftarrow y$
$b_1 := \ y^*\ \leq \beta'$	$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$	return $\mathcal{H}[m, r]$
$b_2 := H(m^*, r^*) \equiv A \cdot y^*$	return σ	
$b_0 \wedge b_1 \wedge b_2$		

No change in behaviour.

AVOID COLLISIONS

Game ₃	$S(m)$	$H(m, r)$
$\mathcal{Q}, \mathcal{H}, \mathcal{P} \leftarrow \emptyset, \emptyset, \emptyset;$	$r \leftarrow \$ \{0, 1\}^\lambda$	if $(m, r) \notin \mathcal{H}$ then
bad \leftarrow false	if $(m, r) \in \mathcal{H}$ then	$y \leftarrow \$ \text{SampD}(\dots, \beta')$
$A, td \leftarrow \$ \text{TrapGen}(\dots)$	bad \leftarrow true	$t := A \cdot y \bmod q$
$(m^*, (y^*, r^*)) \leftarrow \mathcal{A}^{S, H}(vk);$	abort	$\mathcal{H}[m, r] \leftarrow t$
$b_0 := (m^*, \cdot) \notin \mathcal{Q}$	$y \leftarrow \$ \text{SampPre}(td, H(m, r), \beta')$	$\mathcal{P}[m, r] \leftarrow y$
$b_1 := \ y^*\ \leq \beta'$	$\sigma := (y, r)$	return $\mathcal{H}[m, r]$
$b_2 := H(m^*, r^*) \equiv A \cdot y^*$	$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$	
$b_0 \wedge b_1 \wedge b_2$	return σ	

Apply Fundamental Lemma of Game Playing with $\Pr[\text{bad}] \approx |\mathcal{Q}|/2^{\lambda/2}$.

FAKE SIGNATURES

Game ₄	$S(m)$	$H(m, r)$
$\mathcal{Q}, \mathcal{H}, \mathcal{P} \leftarrow \emptyset, \emptyset, \emptyset;$	$r \leftarrow \$ \{0, 1\}^\lambda$	if $(m, r) \notin \mathcal{H}$ then
bad \leftarrow false	if $(m, r) \in \mathcal{H}$ then	$y \leftarrow \$ \text{SampD}(\dots, \beta')$
$A, \text{td} \leftarrow \$ \text{TrapGen}(\dots)$	bad \leftarrow true	$t := A \cdot y \bmod q$
$(m^*, (y^*, r^*)) \leftarrow \mathcal{A}^{S, H}(\text{vk});$	abort	$\mathcal{H}[m, r] \leftarrow t$
$b_0 := (m^*, \cdot) \notin \mathcal{Q}$	$H(m, r)$	$\mathcal{P}[m, r] \leftarrow y$
$b_1 := \ y^*\ \leq \beta'$	$y \leftarrow \mathcal{P}[m, r]$	return $\mathcal{H}[m, r]$
$b_2 := H(m^*, r^*) \equiv A \cdot y^*$	$\sigma := (y, r)$	
$b_0 \wedge b_1 \wedge b_2$	$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$	
	return σ	

Use $\text{SampPre} \approx \text{SampD}$.

THROW AWAY TRAPDOOR

Game ₅	$S(m)$	$H(m, r)$
$\mathcal{Q}, \mathcal{H}, \mathcal{P} \leftarrow \emptyset, \emptyset, \emptyset;$	$r \leftarrow \{0, 1\}^\lambda$	if $(m, r) \notin \mathcal{H}$ then
bad \leftarrow false	if $(m, r) \in \mathcal{H}$ then	$y \leftarrow \text{SampD}(\dots, \beta')$
$A \leftarrow \mathbb{Z}_q^{n \times 2n \lceil \log q \rceil}$	bad \leftarrow true	$t := A \cdot y \bmod q$
$(m^*, (y^*, r^*)) \leftarrow \mathcal{A}^{S, H}(\text{vk});$	abort	$\mathcal{H}[m, r] \leftarrow t$
$b_0 := (m^*, \cdot) \notin \mathcal{Q}$	$H(m, r)$	$\mathcal{P}[m, r] \leftarrow y$
$b_1 := \ y^*\ \leq \beta'$	$y \leftarrow \mathcal{P}[m, r]$	return $\mathcal{H}[m, r]$
$b_2 := H(m^*, r^*) \equiv A \cdot y^*$	$\sigma := (y, r)$	
$b_0 \wedge b_1 \wedge b_2$	$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$	
	return σ	

Use that TrapGen produces pseudorandom A .

We turn an adversary \mathcal{A} that wins Game_5 into an adversary that breaks SIS for \mathbf{A} .

- The adversary outputs $(m^*, (\mathbf{y}^*, \mathbf{r}^*))$ s.t. $H(m^*, \mathbf{r}^*) \equiv \mathbf{A} \cdot \mathbf{y}^*$ and $\|\mathbf{y}^*\| \leq \beta'$.
- If \mathcal{A} has not called $H(m^*, \mathbf{r}^*)$ before producing its forgery, $\mathbf{t} := H(m^*, \mathbf{r}^*)$ is unknown and (close to) uniformly random: it succeeds with probability $1/q^n < 2^{-\lambda}$.
- If \mathcal{A} has called $H(m^*, \mathbf{r}^*)$ then $\mathbf{y} \leftarrow \mathcal{P}[m^*, \mathbf{r}^*]$ with $\|\mathbf{y}\| \leq \beta'$ and **with high probability** $\mathbf{y} \neq \mathbf{y}^*$.

$$H(m^*, \mathbf{r}^*) \equiv \mathbf{A} \cdot \mathbf{y}^* \equiv \mathbf{A} \cdot \mathbf{y} \quad \Rightarrow \quad \mathbf{0} \equiv \mathbf{A} \cdot (\mathbf{y}^* - \mathbf{y}) \text{ and } \|\mathbf{y}^* - \mathbf{y}\| \leq 2\beta'$$

THE ROLE OF \mathbf{r}

If the signing algorithm did not sample a fresh \mathbf{r} for each signature, then

- Game_3 would output new, fresh \mathbf{y} on each call, but
- Game_4 would output the same \mathbf{y} on each call.

which is easy to distinguish by just calling the signing algorithm twice.

Corresponding Attack

Without \mathbf{r} , when \mathcal{A} calls the signing oracle twice on the same m , it would get two different preimages $\mathbf{y}_0, \mathbf{y}_1$ for $H(m)$:

$$H(m) \equiv \mathbf{A} \cdot \mathbf{y}_0 \equiv \mathbf{A} \cdot \mathbf{y}_1 \quad \Rightarrow \quad \mathbf{0} \equiv \mathbf{A} \cdot (\mathbf{y}_0 - \mathbf{y}_1) \text{ and } \|\mathbf{y}_0 - \mathbf{y}_1\| \leq 2\beta'.$$

- This solves SIS, which is supposed to be hard for our security proof to work.

THE ROLE OF \mathbf{r}

If the signing algorithm did not sample a fresh \mathbf{r} for each signature, then

- Game_3 would output new, fresh \mathbf{y} on each call, but
- Game_4 would output the same \mathbf{y} on each call.

which is easy to distinguish by just calling the signing algorithm twice.

Corresponding Attack

Without \mathbf{r} , when \mathcal{A} calls the signing oracle twice on the same m , it would get two different preimages $\mathbf{y}_0, \mathbf{y}_1$ for $H(m)$:

$$H(m) \equiv \mathbf{A} \cdot \mathbf{y}_0 \equiv \mathbf{A} \cdot \mathbf{y}_1 \quad \Rightarrow \quad \mathbf{0} \equiv \mathbf{A} \cdot (\mathbf{y}_0 - \mathbf{y}_1) \text{ and } \|\mathbf{y}_0 - \mathbf{y}_1\| \leq 2\beta'.$$

- This solves SIS, which is supposed to be hard for our security proof to work.
- This gives a short vector in $\Lambda_q^\perp(\mathbf{A})$, such short vectors make up our trapdoor!

FIN

IN THE RANDOM ORACLE MODEL WE
CAN CHOOSE WHAT THE HASH
FUNCTION OUTPUTS.

- [Ajt98] Miklós Ajtai. **The Shortest Vector Problem in L2 is NP-hard for Randomized Reductions (Extended Abstract)**. In: *30th ACM STOC*. ACM Press, May 1998, pp. 10–19. DOI: [10.1145/276698.276705](https://doi.org/10.1145/276698.276705).
- [GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. **Trapdoors for hard lattices and new cryptographic constructions**. In: *40th ACM STOC*. Ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 197–206. DOI: [10.1145/1374376.1374407](https://doi.org/10.1145/1374376.1374407).
- [NR06] Phong Q. Nguyen and Oded Regev. **Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures**. In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Berlin, Heidelberg, 2006, pp. 271–288. DOI: [10.1007/11761679_17](https://doi.org/10.1007/11761679_17).