# GPV Signatures (Hash-then-Sign Signatures)

## Advanced Topics in ~~Cybersecurity~~Cryptography (7CCSMATC)

Martin R. Albrecht

# Falcon (signature scheme)



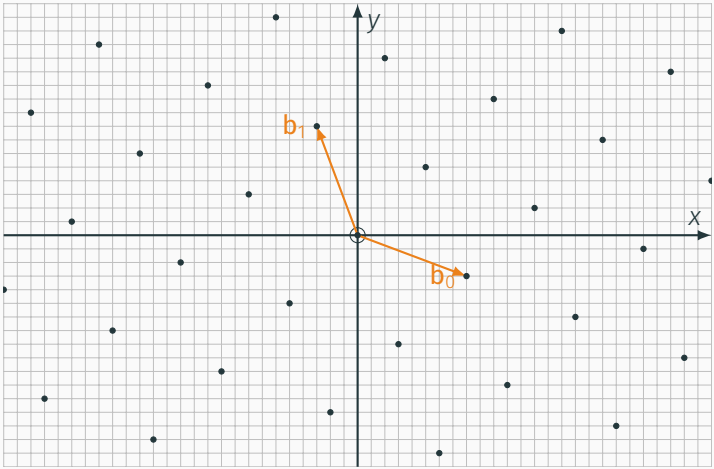
From Wikipedia, the free encyclopedia

**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 ***Fa**st Fourier **l**attice-based **co**mpact signatures over **N**TRU*.

# Outline

# Lattices

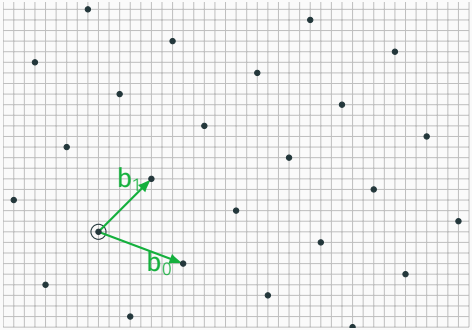
# Lattices

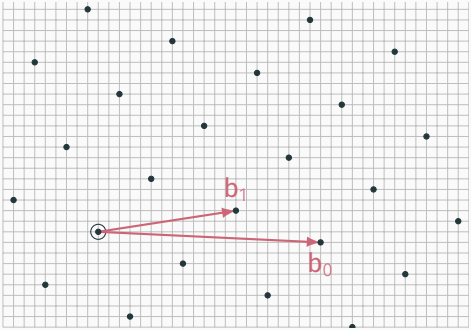



A lattice $\Lambda$ is a discrete subgroup of $\mathbb{R}^d$.

Picture credit: Léo Ducas

# Lattice Bases



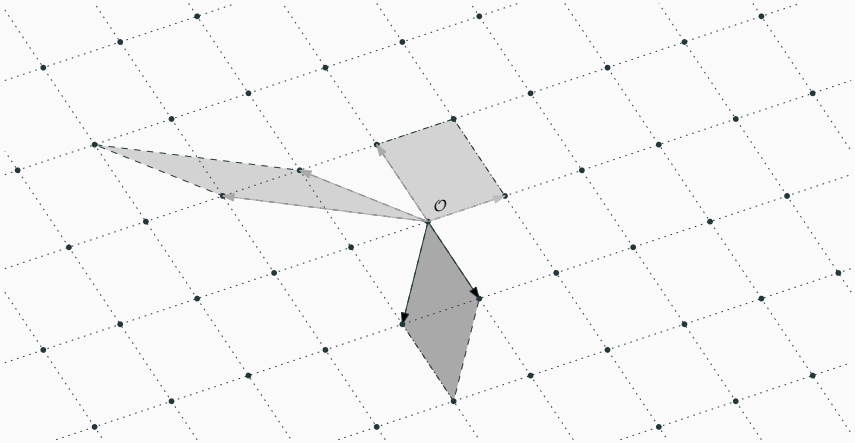




| | | | | |
|---|---|---|---|---|
| G | $\rightarrow$ | B | : easy | (compute Hermite Normal form); |
| B | $\rightarrow$ | G | : hard | (BKZ, lattice sieve ...). |

Picture credit: Léo Ducas

# Lattice Volume I

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




Picture credit: Joop van de Pol

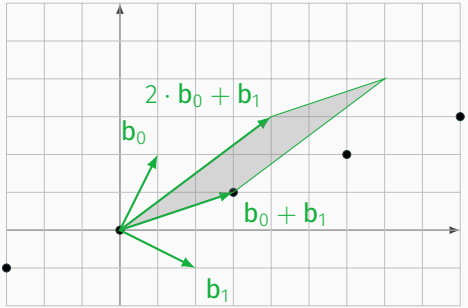
# Lattice Volume II

For any two bases $G$, $B$ of the same lattice $\Lambda$:

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

We can therefore define:

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



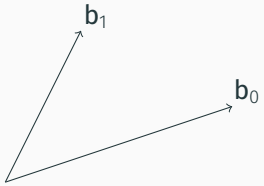

Picture credit: https://tex.stackexchange.com/a/42559

# 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 $\mathbf{b}_0, \ldots, \mathbf{b}_{i-1}$.

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

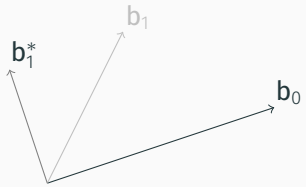


# Length of Gram–Schmidt Vectors

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 $\mathbf{b}_0, \ldots, \mathbf{b}_{i-1}$.

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



# Gram–Schmidt Vectors and Lattice Volumes

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

$$\mathrm{Vol}(\Lambda) = \sqrt{\det(\mathbf{B}^\star \cdot \mathbf{B}^{\star T})} = \prod \|\mathbf{b}_i^\star\| \,.$$

# "Good" Bases

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

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

Therefore, it is somehow equivalent that

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

**Good Basis**

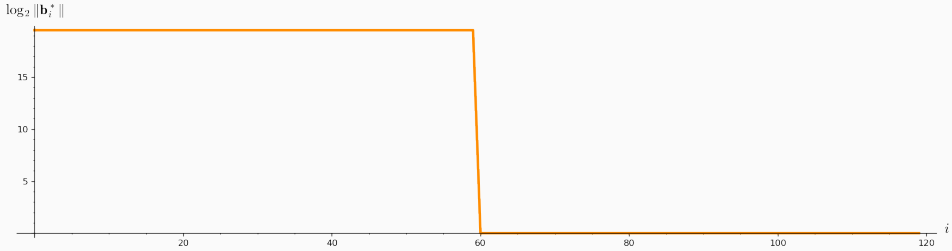
$$\kappa(\mathbf{G}) = \mathrm{poly}(d), \qquad \forall\, i : \|\mathbf{g}_i^\star\| = \mathrm{poly}(d) \cdot \mathrm{Vol}(\Lambda)^{1/d}.$$

**LLL-reduced Basis**

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

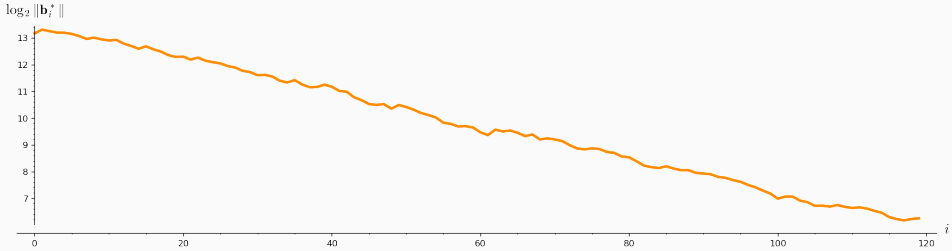
# Example Gram–Schmidt “Shape”

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



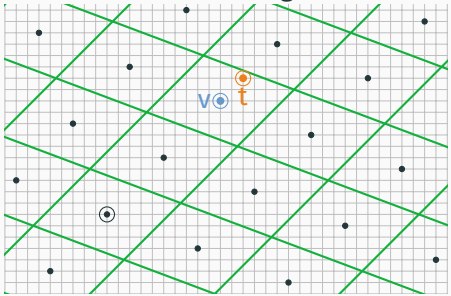
# Example Gram–Schmidt "Shape" after LLL

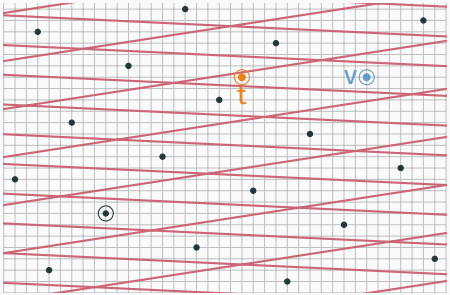
```
A = LLL.reduction(A)
M = GSO.Mat(A, update=True)
line([(i,log(r_, 2)/2) for i, r_ in enumerate(M.r())], **plot_kwds)
```



# Bases and Fundamental Domains

Each basis defines a **tiling**.

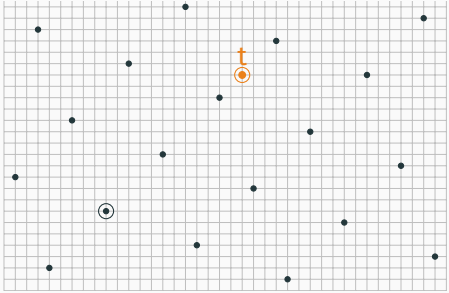



**Round'off Algorithm [Lenstra, Babai]:**

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

# Round Off Algorithm

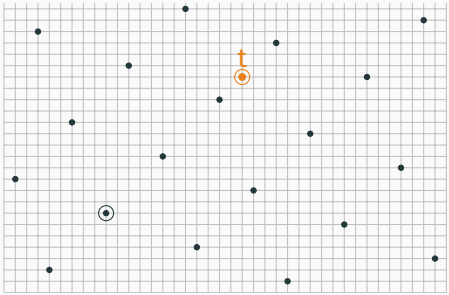





Round Off Algorithm
[Lenstra,Babai]:

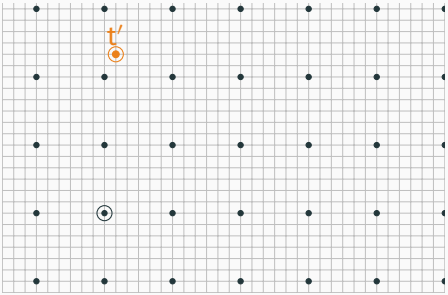
Credit: Léo Ducas

# Round Off Algorithm




$\cdot\, \mathbf{B}^{-1}$

$\longrightarrow$

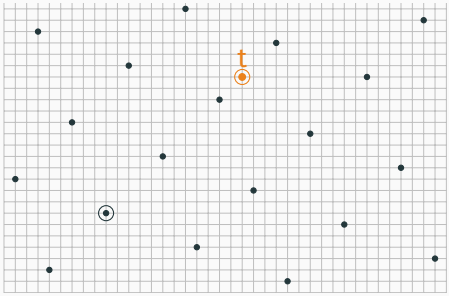



Round Off Algorithm
[Lenstra,Babai]:

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

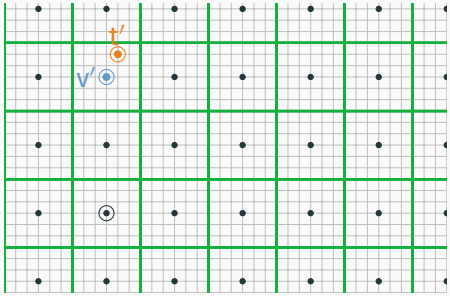
Credit: Léo Ducas

# Round Off Algorithm




$\cdot B^{-1}$

$\longrightarrow$

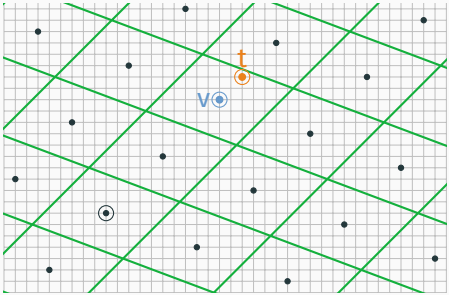



Round Off Algorithm [Lenstra,Babai]:

Credit: Léo Ducas

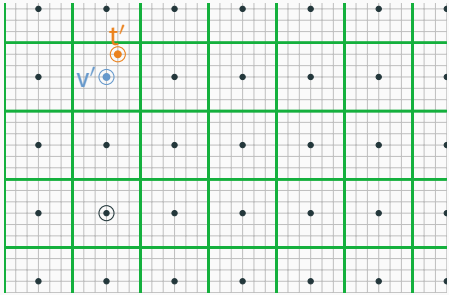
- Use $B$ to switch to the lattice $\mathbb{Z}^d$ $\quad t' = B^{-1} \cdot t$
- Round each coordinate $\quad v' = \lfloor t' \rceil$

# Round Off Algorithm




$\cdot B^{-1}$ $\longrightarrow$

$\longleftarrow$ $\cdot B$



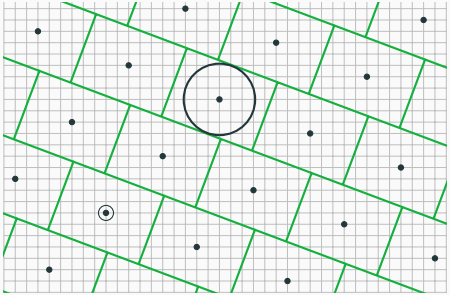

Round Off Algorithm [Lenstra,Babai]:

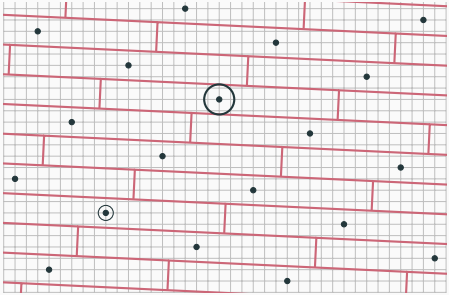
Credit: Léo Ducas

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

# Nearest-Plane Algorithm

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



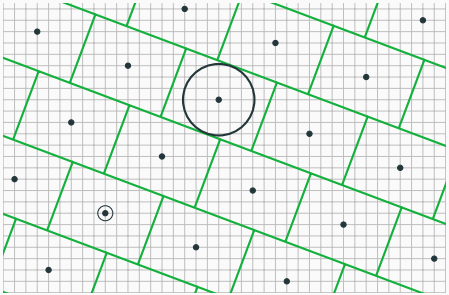


- Worst-case distance: $\frac{1}{2}\sqrt{\sum \left\|\mathbf{b}_i^\star\right\|^2}$ (Approx-CVP)
- Correct decoding of $\mathbf{t} = \mathbf{v} + \mathbf{e}$ for $\mathbf{v} \in \Lambda$ if $\|\mathbf{e}\| \leq \min \left\|\mathbf{b}_i^\star\right\|$ (BDD)

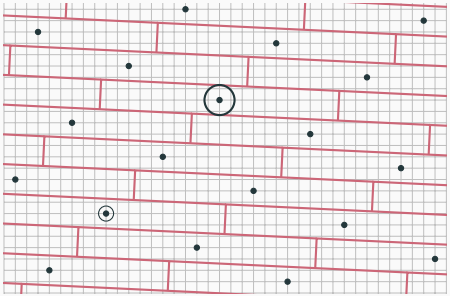
Credit: Léo Ducas

# Trapdoors from Lattices ?

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



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

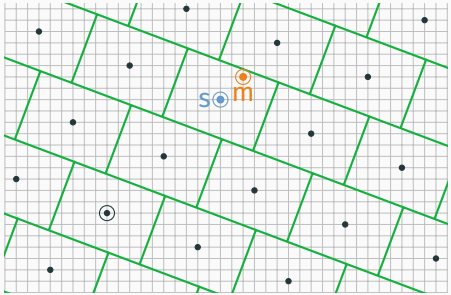


Can we use this as a trapdoor?

# Signatures

**Sign:**

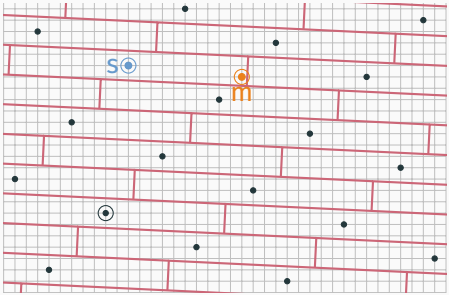
1. Hash message to a random vector $\mathbf{m}$.
2. Apply Nearest Plane with a good basis $\mathbf{G}$: find $\mathbf{s} \in \Lambda$ close to $\mathbf{m}$.




Credit: Léo Ducas

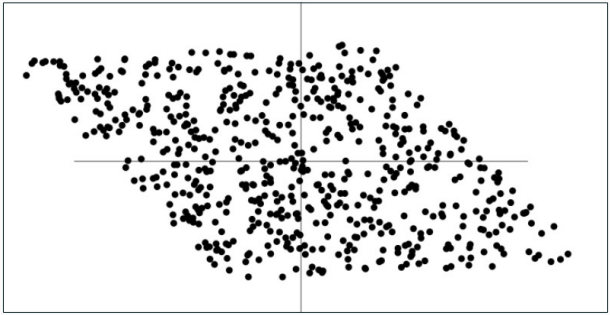
**Verify:**

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 $\mathbf{G}^\star$:



- Each signatures $(\mathbf{s}, \mathbf{m})$ leaks information about $\mathbf{G}$.
- Learning a parallepiped from few signatures [EC:NguReg06]
- Total break of original GGH and NTRUSign schemes

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

| Seminal Work |
| --- |
| STOC:GenPeiVai08 |

# q-ary Lattices

# Construction of $q$-ary Lattices (Primal / Construction A)

- Let $q$ be a prime integer[1] and $n < m$ two positive integers.
- The matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ spans the $q$-ary lattice:

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

**Lattice Parameters**

Assuming $\mathbf{A}$ is full-rank:

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

[1] for simplicity

# Example

```
q = 19; n = 3; m = 7; A = random_matrix(GF(q), n, m)
B = A.lift().stack(q * identity_matrix(m)); B.T
```

```
[1      4 19                ]
[ 3  5      19              ]
[15  3 17       19          ]
[ 2  7             19       ]
[14  7                19     ]
[ 9  8 15                19  ]
[15  1 11                   19]
```

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

```
[ 1                 ]
[    1              ]
[       1           ]
[18 14 17 19        ]
[ 1  4  2    19     ]
[ 2 17  6       19  ]
[ 5  7 14          19]
```

**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. I transposed the output to make it consistent with the slide before.

# Construction of $q$-ary Lattices (Dual / Parity-Check)

- Let $q$ be a prime integer[2] and $n < m$ two positive integers.
- The matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is the parity-check of the lattice:

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

**Lattice parameters**

Assuming $\mathbf{A}$ is full-rank:

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

[2] for simplicity

# Example

```
q = 19; 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.T
```

```
[ 1          19                 ]
[    1          19              ]
[       1          19           ]
[          1          19        ]
[13  1  1 16             19     ]
[12 15  1 11                19   ]
[12 12  4  2                   19]
```

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

```
[ 1                 ]
[    1              ]
[       1           ]
[          1        ]
[13  1  1 16 19     ]
[12 15  1 11    19  ]
[12 12  4  2       19]
```

# 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 $\mathbf{A} \cdot \mathbf{x} \equiv \mathbf{0} \bmod q$ is hard.[3]

**Lattice Formulation**

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

In [STOC:Ajtai98], Ajtai established that if solving SIS is easy then solving Approx-SVP for any lattice is also easy.

[3] Also on a quantum computer!

# A Simple Application of SIS

Consider the function:

$$f_{\mathbf{A}} : \{0,1\}^m \to \mathbb{Z}_q^n, \qquad \mathbf{x} \mapsto \mathbf{A} \cdot \mathbf{x} \bmod q$$

**SIS $\Rightarrow$ Collision Resistant Hashing and One-Way Function**

- Finding collisions is as hard as SIS[4] (take the difference)

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

- $f_{\mathbf{A}}$ is highly surjective (many pre-images for any image)
- Finding pre-images is hard (show it!)

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

# GPV Signatures

# Outline

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

**Sign** Compute $H(m)$.
Find an arbitrary, not-necessarily short, preimage $\mathbf{u}$: $\mathbf{A} \cdot \mathbf{u} \equiv H(m) \bmod q$.
Use the trapdoor td to solve Approx-CVP for $\mathbf{u}$ on $\Lambda_q^\perp(\mathbf{A})$, call it $\mathbf{z}$. By construction $\mathbf{A} \cdot \mathbf{z} \equiv \mathbf{0} \bmod q$.
Output $\mathbf{y} := \mathbf{u} - \mathbf{z}$. By construction $\|\mathbf{y}\|$ is small.

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

# Scheme

KeyGen($1^\lambda$)

$\mathbf{A}, \mathsf{td} \leftarrow\!\$\ \mathsf{TrapGen}(1^n, 1^m, q, \beta)$

**return** $\mathsf{vk} := \mathbf{A}, \mathsf{sk} := \mathsf{td}$

Sign($m, \mathsf{sk}$)

$\mathbf{r} \leftarrow\!\$\ \{0,1\}^\lambda$

$\mathbf{y} \leftarrow\!\$\ \mathsf{SampPre}(\mathsf{td}, H(m, \mathbf{r}), \beta')$

**return** $\sigma := (\mathbf{y}, \mathbf{r})$

Verify($\mathsf{vk}, \sigma, m$)

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

- $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q, \beta)$ 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 $\mathrm{negl}(\lambda)$ statistical distance to the uniform distribution on $\mathbb{Z}_q^{n \times m}$.*
- $\mathbf{u} \leftarrow \mathsf{SampD}(1^n, 1^m, \beta')$ 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* $\mathrm{negl}(\lambda)$ *statistical distance to the uniform distribution on* $\mathbb{Z}_q^n$.
- $\mathbf{u} \leftarrow \mathsf{SampPre}(\mathsf{td}, \mathbf{v}, \beta')$ 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* $\mathrm{negl}(\lambda)$ *statistical distance to* $\mathbf{u} \leftarrow \mathsf{SampD}(1^n, 1^m, \mathcal{R}, \beta')$ *conditioned on* $\mathbf{v} \equiv \mathbf{A} \cdot \mathbf{u} \bmod q$.

# Security Proof

# EUF-CMA in the ROM

| EUF-CMA | S($m$) | H($m$) |
|---|---|---|
| $\mathcal{Q}, \mathcal{H} \leftarrow \emptyset, \emptyset$; | $\sigma \leftarrow\$ \Sigma.\mathsf{Sign}(\mathsf{sk}, m)$ | **if** $m \notin \mathcal{H}$ **then** |
| $\mathsf{vk}, \mathsf{sk} \leftarrow\$ \Sigma.\mathsf{KeyGen}(1^\lambda)$; | $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$ | $\mathcal{H}[m] \leftarrow\$ \{0,1\}^\lambda$ |
| $(m^\star, \sigma^\star) \leftarrow\$ \mathcal{A}^{\mathsf{S},\mathsf{H}}(\mathsf{vk})$; | **return** $\sigma$ | **return** $\mathcal{H}[m]$ |
| $b_0 \coloneqq (m^\star, \cdot) \notin \mathcal{Q}$ | | |
| $b_1 \coloneqq \Sigma.\mathsf{Verify}(\mathsf{vk}, \sigma^\star, m^\star) = 1$ | | |
| **return** $b_0 \wedge b_1$ | | |

$$\mathsf{Adv}^{\text{euf-cma}}_{\mathcal{A},\Sigma}(\lambda) \coloneqq \Pr[\text{EUF-CMA}^{\mathcal{A}}_{\Sigma}(\lambda) \Rightarrow 1]$$

# Inlining GPV Signatures

| $\mathrm{Game}_0$ | $\mathsf{S}(m)$ | $\mathsf{H}(m, \mathbf{r})$ |
|---|---|---|
| $\mathcal{Q}, \mathcal{H} \leftarrow \emptyset, \emptyset;$ | $\mathbf{r} \leftarrow\!\$\ \{0,1\}^\lambda$ | if $(m, \mathbf{r}) \notin \mathcal{H}$ then |
| $\mathbf{A}, \mathsf{td} \leftarrow\!\$\ \mathsf{TrapGen}(\cdots)$ | $\mathbf{y} \leftarrow\!\$\ \mathsf{SampPre}(\mathsf{td}, H(m, \mathbf{r}), \beta')$ | $\mathcal{H}[m, \mathbf{r}] \leftarrow\!\$\ \mathbb{Z}_q^n$ |
| $(m^\star, (\mathbf{y}^\star, \mathbf{r}^\star)) \leftarrow \mathcal{A}^{\mathsf{S},\mathsf{H}}(\mathsf{vk});$ | $\sigma := (\mathbf{y}, \mathbf{r})$ | return $\mathcal{H}[m, \mathbf{r}]$ |
| $b_0 := (m^\star, \cdot) \notin \mathcal{Q}$ | $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$ | |
| $b_1 := \lVert \mathbf{y}^\star \rVert \leq \beta'$ | return $\sigma$ | |
| $b_2 := H(m^\star, \mathbf{r}^\star) \equiv \mathbf{A} \cdot \mathbf{y}^\star$ | | |
| return $b_0 \wedge b_1 \wedge b_2$ | | |

# Changing the RO

| $\mathsf{Game}_1$ | $\mathsf{S}(m)$ | $\mathsf{H}(m, \mathsf{r})$ |
|---|---|---|
| $\mathcal{Q}, \mathcal{H}, \mathcal{P} \leftarrow \emptyset, \emptyset, \emptyset;$ | $\mathsf{r} \leftarrow\!\$\ \{0,1\}^\lambda$ | **if** $(m, \mathsf{r}) \notin \mathcal{H}$ **then** |
| $\mathsf{A}, \mathsf{td} \leftarrow\!\$\ \mathsf{TrapGen}(\cdots)$ | $\mathsf{y} \leftarrow\!\$\ \mathsf{SampPre}(\mathsf{td}, H(m, \mathsf{r}), \beta')$ | $\mathsf{y} \leftarrow\!\$\ \mathsf{SampD}(\cdots, \beta')$ |
| $(m^\star, (\mathsf{y}^\star, \mathsf{r}^\star)) \leftarrow \mathcal{A}^{\mathsf{S},\mathsf{H}}(\mathsf{vk});$ | $\sigma := (\mathsf{y}, \mathsf{r})$ | $\mathsf{t} := \mathsf{A} \cdot \mathsf{y} \bmod q$ |
| $b_0 := (m^\star, \cdot) \notin \mathcal{Q}$ | $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$ | $\mathcal{H}[m, \mathsf{r}] \leftarrow \mathsf{t}$ |
| $b_1 := \lVert \mathsf{y}^\star \rVert \leq \beta'$ | **return** $\sigma$ | $\mathcal{P}[m, \mathsf{r}] \leftarrow \mathsf{y}$ |
| $b_2 := H(m^\star, \mathsf{r}^\star) \equiv \mathsf{A} \cdot \mathsf{y}^\star$ | | **return** $\mathcal{H}[m, \mathsf{r}]$ |
| $b_0 \wedge b_1 \wedge b_2$ | | |

By the Leftover Hash Lemma the distributions of $\mathsf{Game}_0$ and $\mathsf{Game}_1$ are statistically close.

# Bookkeeping

| $\mathrm{Game}_2$ | $\mathsf{S}(m)$ | $\mathsf{H}(m, \mathsf{r})$ |
|---|---|---|
| $\mathcal{Q}, \mathcal{H}, \mathcal{P} \leftarrow \emptyset, \emptyset, \emptyset;$ | $\mathsf{r} \leftarrow\!\!\$\ \{0,1\}^\lambda$ | **if** $(m, \mathsf{r}) \notin \mathcal{H}$ **then** |
| $\mathsf{bad} \leftarrow \mathbf{false}$ | **if** $(m, \mathsf{r}) \in \mathcal{H}$ **then** | $\mathsf{y} \leftarrow\!\!\$\ \mathrm{SampD}(\cdots, \beta')$ |
| $\mathsf{A}, \mathrm{td} \leftarrow\!\!\$\ \mathrm{TrapGen}(\cdots)$ | $\mathsf{bad} \leftarrow \mathbf{true}$ | $\mathsf{t} \coloneqq \mathsf{A} \cdot \mathsf{y} \bmod q$ |
| $(m^\star, (\mathsf{y}^\star, \mathsf{r}^\star)) \leftarrow \mathcal{A}^{\mathsf{S},\mathsf{H}}(\mathrm{vk});$ | $\mathsf{y} \leftarrow\!\!\$\ \mathrm{SampPre}(\mathrm{td}, H(m, \mathsf{r}), \beta')$ | $\mathcal{H}[m, \mathsf{r}] \leftarrow \mathsf{t}$ |
| $b_0 \coloneqq (m^\star, \cdot) \notin \mathcal{Q}$ | $\sigma \coloneqq (\mathsf{y}, \mathsf{r})$ | $\mathcal{P}[m, \mathsf{r}] \leftarrow \mathsf{y}$ |
| $b_1 \coloneqq \lVert \mathsf{y}^\star \rVert \leq \beta'$ | $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$ | **return** $\mathcal{H}[m, \mathsf{r}]$ |
| $b_2 \coloneqq H(m^\star, \mathsf{r}^\star) \equiv \mathsf{A} \cdot \mathsf{y}^\star$ | **return** $\sigma$ | |
| $b_0 \wedge b_1 \wedge b_2$ | | |

No change in behaviour.

# Avoid Collisions

| $\mathsf{Game}_3$ | $\mathsf{S}(m)$ | $\mathsf{H}(m, \mathsf{r})$ |
|---|---|---|
| $\mathcal{Q}, \mathcal{H}, \mathcal{P} \leftarrow \emptyset, \emptyset, \emptyset$; | $\mathsf{r} \leftarrow\!\$\ \{0,1\}^\lambda$ | **if** $(m, \mathsf{r}) \notin \mathcal{H}$ **then** |
| $\mathsf{bad} \leftarrow$ **false** | **if** $(m, \mathsf{r}) \in \mathcal{H}$ **then** | $\mathsf{y} \leftarrow\!\$\ \mathsf{SampD}(\cdots, \beta')$ |
| $\mathsf{A}, \mathsf{td} \leftarrow\!\$\ \mathsf{TrapGen}(\cdots)$ | $\mathsf{bad} \leftarrow$ **true** | $\mathsf{t} \coloneqq \mathsf{A} \cdot \mathsf{y} \bmod q$ |
| $(m^\star, (\mathsf{y}^\star, \mathsf{r}^\star)) \leftarrow \mathcal{A}^{\mathsf{S},\mathsf{H}}(\mathsf{vk})$; | **abort** | $\mathcal{H}[m, \mathsf{r}] \leftarrow \mathsf{t}$ |
| $b_0 \coloneqq (m^\star, \cdot) \notin \mathcal{Q}$ | $\mathsf{y} \leftarrow\!\$\ \mathsf{SampPre}(\mathsf{td}, H(m, \mathsf{r}), \beta')$ | $\mathcal{P}[m, \mathsf{r}] \leftarrow \mathsf{y}$ |
| $b_1 \coloneqq \lVert \mathsf{y}^\star \rVert \leq \beta'$ | $\sigma \coloneqq (\mathsf{y}, \mathsf{r})$ | **return** $\mathcal{H}[m, \mathsf{r}]$ |
| $b_2 \coloneqq H(m^\star, \mathsf{r}^\star) \equiv \mathsf{A} \cdot \mathsf{y}^\star$ | $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$ | |
| $b_0 \wedge b_1 \wedge b_2$ | **return** $\sigma$ | |

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

# Fake Signatures

| $\mathrm{Game}_4$ | $\mathrm{S}(m)$ | $\mathrm{H}(m, \mathbf{r})$ |
|---|---|---|
| $\mathcal{Q}, \mathcal{H}, \mathcal{P} \leftarrow \emptyset, \emptyset, \emptyset;$ | $\mathbf{r} \leftarrow\!\$\ \{0,1\}^\lambda$ | **if** $(m, \mathbf{r}) \notin \mathcal{H}$ **then** |
| bad $\leftarrow$ **false** | **if** $(m, \mathbf{r}) \in \mathcal{H}$ **then** | $\mathbf{y} \leftarrow\!\$\ \mathrm{SampD}(\cdots, \beta')$ |
| $\mathbf{A}, \mathrm{td} \leftarrow\!\$\ \mathrm{TrapGen}(\cdots)$ | bad $\leftarrow$ **true** | $\mathbf{t} \coloneqq \mathbf{A} \cdot \mathbf{y} \bmod q$ |
| $(m^\star, (\mathbf{y}^\star, \mathbf{r}^\star)) \leftarrow \mathcal{A}^{\mathrm{S},\mathrm{H}}(\mathrm{vk});$ | **abort** | $\mathcal{H}[m, \mathbf{r}] \leftarrow \mathbf{t}$ |
| $b_0 \coloneqq (m^\star, \cdot) \notin \mathcal{Q}$ | $H(m, \mathbf{r})$ | $\mathcal{P}[m, \mathbf{r}] \leftarrow \mathbf{y}$ |
| $b_1 \coloneqq \lVert \mathbf{y}^\star \rVert \leq \beta'$ | $\mathbf{y} \leftarrow \mathcal{P}[m, \mathbf{r}]$ | **return** $\mathcal{H}[m, \mathbf{r}]$ |
| $b_2 \coloneqq H(m^\star, \mathbf{r}^\star) \equiv \mathbf{A} \cdot \mathbf{y}^\star$ | $\sigma \coloneqq (\mathbf{y}, \mathbf{r})$ | |
| $b_0 \wedge b_1 \wedge b_2$ | $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$ | |
| | **return** $\sigma$ | |

Use SampPre $\approx$ SampD.

# Throw Away Trapdoor

| $\mathrm{Game}_5$ | $\mathsf{S}(m)$ | $\mathsf{H}(m, \mathbf{r})$ |
|---|---|---|
| $\mathcal{Q}, \mathcal{H}, \mathcal{P} \leftarrow \emptyset, \emptyset, \emptyset$; | $\mathbf{r} \leftarrow\!\$\ \{0,1\}^\lambda$ | **if** $(m, \mathbf{r}) \notin \mathcal{H}$ **then** |
| bad $\leftarrow$ **false** | **if** $(m, \mathbf{r}) \in \mathcal{H}$ **then** | $\mathbf{y} \leftarrow\!\$\ \mathrm{SampD}(\cdots, \beta')$ |
| $\mathbf{A} \leftarrow\!\$\ \mathbb{Z}_q^{n \times 2n\lceil \log q \rceil}$ | bad $\leftarrow$ **true** | $\mathbf{t} := \mathbf{A} \cdot \mathbf{y} \bmod q$ |
| $(m^\star, (\mathbf{y}^\star, \mathbf{r}^\star)) \leftarrow \mathcal{A}^{\mathsf{S},\mathsf{H}}(\mathrm{vk})$; | **abort** | $\mathcal{H}[m, \mathbf{r}] \leftarrow \mathbf{t}$ |
| $b_0 := (m^\star, \cdot) \notin \mathcal{Q}$ | $H(m, \mathbf{r})$ | $\mathcal{P}[m, \mathbf{r}] \leftarrow \mathbf{y}$ |
| $b_1 := \|\mathbf{y}^\star\| \leq \beta'$ | $\mathbf{y} \leftarrow \mathcal{P}[m, \mathbf{r}]$ | **return** $\mathcal{H}[m, \mathbf{r}]$ |
| $b_2 := H(m^\star, \mathbf{r}^\star) \equiv \mathbf{A} \cdot \mathbf{y}^\star$ | $\sigma := (\mathbf{y}, \mathbf{r})$ | |
| $b_0 \wedge b_1 \wedge b_2$ | $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$ | |
| | **return** $\sigma$ | |

Use that TrapGen produces pseudorandom $\mathbf{A}$.

# Invoke SIS

We turn an adversary $\mathcal{A}$ that wins $\text{Game}_5$ into an adversary that breaks SIS for $\mathbf{A}$.

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

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

# The Role of r

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

- $\text{Game}_3$ would output new, fresh $\mathbf{y}$ on each call, but
- $\text{Game}_4$ 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 r

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

- $\text{Game}_3$ would output new, fresh $\mathbf{y}$ on each call, but
- $\text{Game}_4$ 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!

# In the Random Oracle Model we can choose what the hash function outputs.

# References I