Peregrine: Toward Fastest FALCON Based on GPV Framework

Feb. 23, 2023

Young-Sik Kim	Chosun University
Eun-young Seo	Chosun University
Joon-Woo Lee	Chung-ang University
Jong-Seon No	Seoul National University

Lattice Based Signatures

- **FALCON**
  - Selected as a standard signature by NIST in July 2022
  - Hash-and-Sign Signature
  - GPV framework + NTRU lattices + fast Fourier sampling

Algorithms	Category	Security	Public Key	Private Key	Signature Size
Dilithium	MLWE	128	1,312B	2,528B	2,420B
		256	2,592B	4,864B	4,595B
Falcon	NTRU	128	**897B**	**1,281B**	**666B**
		256	**1,793B**	**2,305B**	**1,330B**
SPHINCS+	Hash	128	32B	64B	17,088B
		256	64B	128B	49,856B

GPV Framework

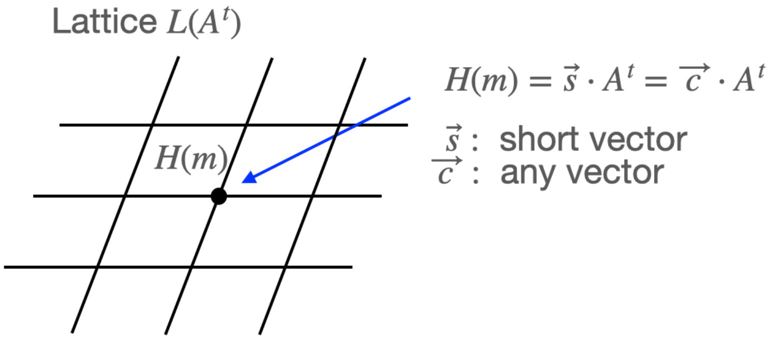
- Gentry, Peikert and Vaikuntanathan (GPV) framework
  - For construction hash-and-sign lattice-based signature scheme

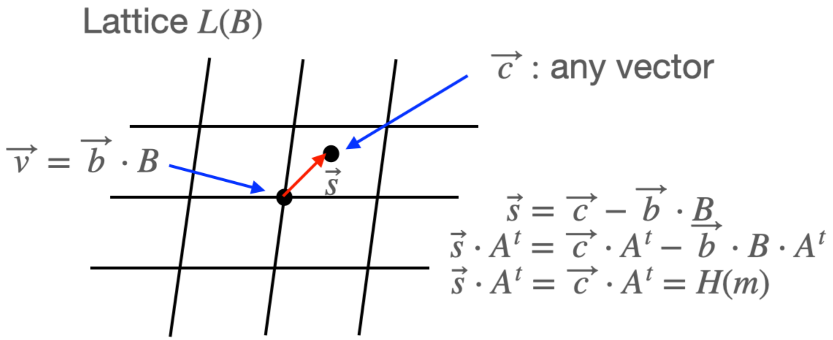
Public Key	A full-rank matrix $A \in Z_q^{n\times m}(m > n)$ generating a q-ary lattice $L_q = L(A^t)$
Private Key	A trapdoor matrix $B \in Z_q^{m\times m}$ generating the lattice $L_q^{\perp} = L(B)$, orthogonal to L_q, such as $B \times A^t = 0$
Signature	a short value $\vec{s} \in Z_q^m$ such that $\vec{s} \cdot A^t = H(m)$
	A message $\boldsymbol{m}$ a hash function $H: \{0, 1\}^* \rightarrow Z_q^n$ a random salt r, $H(m \| \| r)$

GPV Framework

How to find the short s

- Find any $\vec{c}$, satisfying $\vec{c} \cdot A^t = H(m)$
- Find the lattice point $\vec{v} \in L_q^{\perp}$, close to $\vec{c}$
- Then, $\vec{s}A^t = (\vec{c} - \vec{v})A^t = \vec{c}A^t - \vec{b}BA^t = H(m)$

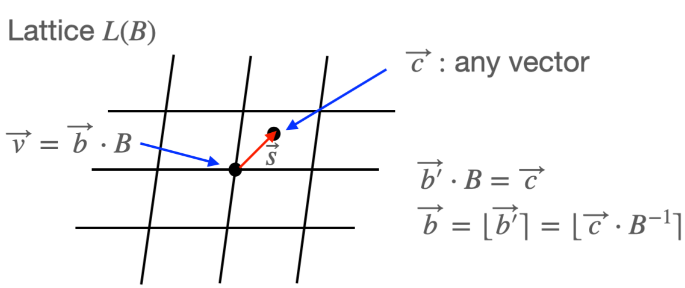


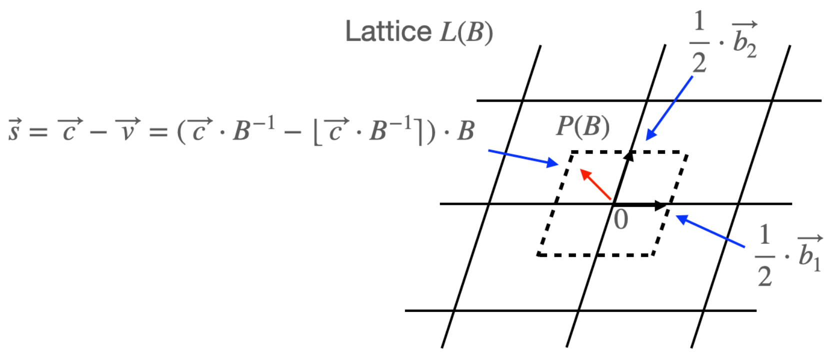


GPV Framework

Babai's round-off algorithm

- Lattice $L(B) = \{\sum_{i=1}^{m} x_i \cdot \overrightarrow{b_i} \,|\, x_i \in Z\}$, row vectors of $B = \{\overrightarrow{b_1}, \overrightarrow{b_2}, \cdots, \overrightarrow{b_m}\}$
- Half-open Parallelipiped $P(B) = \{\sum_{i=1}^{m} x_i \cdot \overrightarrow{b_i} \,|\, -\frac{1}{2} \le x_i < \frac{1}{2}\}$
- Step 1 : find $\overrightarrow{b'}$ satisfying $\overrightarrow{b'} \cdot B = \vec{c}$ such as $\overrightarrow{b'} = \vec{c} \cdot B^{-1}$
- Step 2 : $\vec{b} = \lfloor \overrightarrow{b'} \rceil = \lfloor \vec{c} \cdot B^{-1} \rceil$
- Step 3 : $\vec{v} = \vec{b} \cdot B = \lfloor \vec{c} \cdot B^{-1} \rceil \cdot B$
- Short vector : $\vec{s} = \vec{c} - \vec{v} = (\vec{c} \cdot B^{-1} - \lfloor \vec{c} \cdot B^{-1} \rceil) \cdot B \in P(B)$
- Parallel operation possible, but longer vector $\vec{s}$ than the Babai's nearest plane algorithm





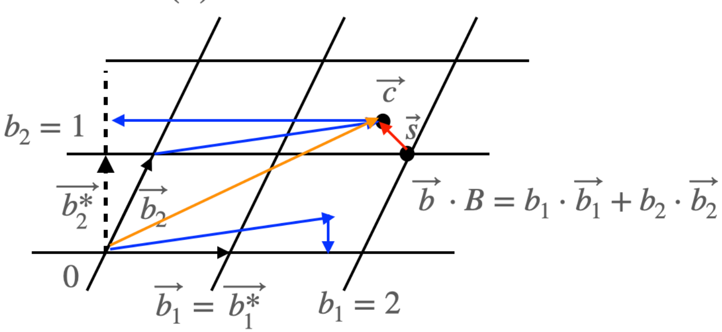
GPV Framework

Babai's nearest plane algorithm

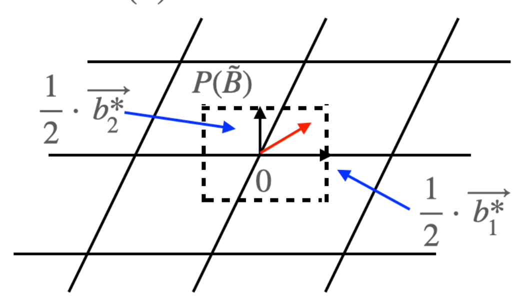
- $\tilde{B}$: Gram-Schmidt orthogonalization of B
- $\tilde{B} = L \cdot B$, row vectors of $\tilde{B} = \{\overrightarrow{b_1^*}, \overrightarrow{b_2^*}, \cdots, \overrightarrow{b_m^*}\}$, $\overrightarrow{b_i^*} = \overrightarrow{b_i} - \sum_{j=1}^{i-1} \frac{<\overrightarrow{b_i}, \overrightarrow{b_j^*}>}{<\overrightarrow{b_j^*}, \overrightarrow{b_j^*}>} \overrightarrow{b_j^*}$, $i > 1$, $\overrightarrow{b_1^*} = \overrightarrow{b_1}$
- Half-open Parallelipiped $P(\tilde{B}) = \{ \sum_{i=1}^{m} x_i \cdot \overrightarrow{b_i^*} \mid -\frac{1}{2} \leq x_i < \frac{1}{2} \}$
- Step 0 : $i = m$
- Step 1 : $b_i = \lfloor \frac{<\vec{c}, \overrightarrow{b_i^*}>}{<\overrightarrow{b_i^*}, \overrightarrow{b_i^*}>} \rceil$, $\vec{b} = (b_1, b_2, \cdots, b_m)$
- Step 2 : $\vec{c} = \vec{c} - b_i \cdot \overrightarrow{b_i}$
- Step 3 : $i = i - 1$, if $i \geq 1$ go to Step 1
- Sequential operation, but shorter $\vec{s}$ than the Babai's round-off algorithm

$\vec{c} - \vec{b} \cdot B \in P(\tilde{B})$

Lattice $L(B)$



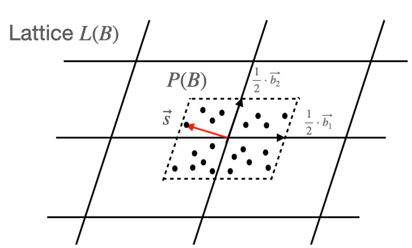
Lattice $L(B)$



GPV Framework

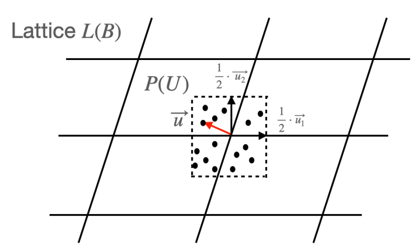
Problem of deterministic algorithms

- Solving Hidden Parallelepiped Problem(Learing a Parallelepiped:Cryptanalysis of GGH and NTRU Signatures)
- **Input** : A polynomial number of samples uniformly distributed over $P(B)$
- **Output** : Approximation of rows of $B = \{\overrightarrow{b_1}, \overrightarrow{b_2}, \cdots, \overrightarrow{b_m}\}$
- 1. **Gram Leakage :** $\vec{s} = \vec{x}B$, $\vec{x} = (x_1, x_2, \cdots, x_m)$, x_i has uniform distribution over $[-\frac{1}{2}, \frac{1}{2}] \rightarrow E[\vec{s}^t\vec{s}] = B^tB/12$
- 2. **Hypercube Transformation :** $G = B^tB$ is the symmetric positive definite matrix. There exists the unique lower-triangular matrix L, such that $G^{-1} = LL^t$. Then, $U = BL$ is an orthogonal matrix and $P(U)$ is a unit hypercube.
- 3. **Learning** U : Define a function $m_{\vec{u},k}(\vec{w}) = E[<\vec{u}, \vec{w}>^k]$, $\vec{u}$ is uniformly distributed over $P(U)$,
  $\vec{u} = \vec{x}U = \sum_{i=1}^{m} x_i\overrightarrow{u_i}$, $\vec{w} \in R^n$. When $k = 4$, $m_{\vec{u},4}(\vec{w}) = \frac{||\vec{w}||^4}{3} - \frac{2}{15}\sum_{i=1}^{m} <\overrightarrow{u_i}, \vec{w}>^4$ has the global minimum $\frac{1}{5}$ and the minimum is obtained at $\pm\overrightarrow{u_1}, \cdots, \pm\overrightarrow{u_m}$ over the unit sphere of R^n. Gradient descent algorithm can be used to find the minimum point.
- 4. **Approximation of** B : $B = UL^{-1}$
- B and $\tilde{B}$ can be revealed from uniform samples over $P(B)$ and $P(\tilde{B})$.



Assumption :$\vec{s}$ are independent and uniformly distributed over $P(B)$

$$(B^tB)^{-1} = LL^t$$
$$U = BL$$



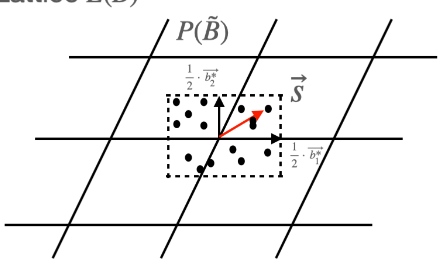
Hypercube transformation

GPV Framework to FALCON

Add randomness with discrete Gaussian R.V.

- Hide the private key from addtion of randomness
- $m-$ dimensional Gaussian function $\rho : R^n \rightarrow (0,1]$ is defined as $\rho(\vec{x}) = e^{-\pi \cdot <\vec{x},\vec{x}>}$, and $\rho_B(\vec{x}) = \rho(B^{-1}\vec{x}) = \rho_{\sqrt{\Sigma}}(\vec{x})$, $BB^t = \Sigma$
- Discrete Gaussian distribution over a lattice L : for all $\vec{x} \in L + \vec{c}$, $D_{L+\vec{c},\sqrt{\Sigma}} = \dfrac{\rho_{\sqrt{\Sigma}}(\vec{x})}{\rho_{\sqrt{\Sigma}}(L+\vec{c})}$
- Babai's nearest plane algorithm + discrete Gaussian r.v.

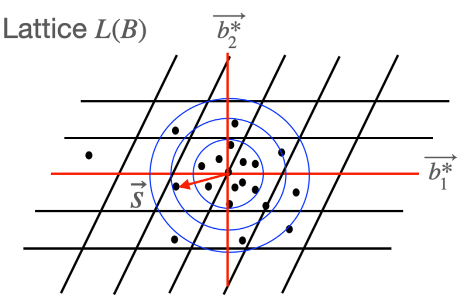
Lattice $L(B)$



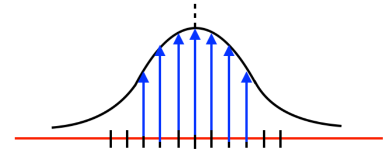
$\vec{s}$: uniformly distributed over $P(\tilde{B})$

Add Randomness

Lattice $L(B)$



$\vec{s}$: discrete Gaussian distribution

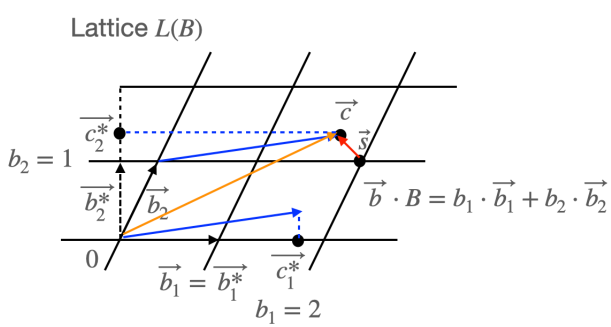


GPV Framework to FALCON

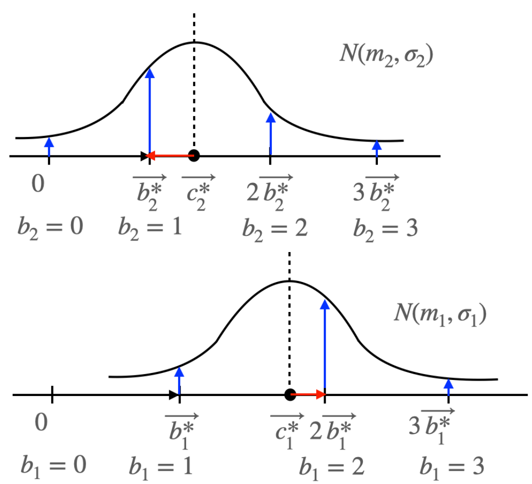
Add randomness with discrete Gaussian R.V.

- Hide the private key from addtion of randomness
- $m-$ dimensional Gaussian function $\rho : R^n \rightarrow (0,1]$ is defined as $\rho(\vec{x}) = e^{-\pi \cdot <\vec{x},\vec{x}>}$, and $\rho_B(\vec{x}) = \rho(B^{-1}\vec{x}) = \rho_{\sqrt{\Sigma}}(\vec{x})$, $BB^t = \Sigma$
- Discrete Gaussian distribution over a lattice L : for all $\vec{x} \in L + \vec{c}$, $D_{L+\vec{c},\sqrt{\Sigma}} = \dfrac{\rho_{\sqrt{\Sigma}}(\vec{x})}{\rho_{\sqrt{\Sigma}}(L + \vec{c})}$

  Babai's nearest plane algorithm + discrete Gaussian r.v.



Babai's nearest plane algorithm

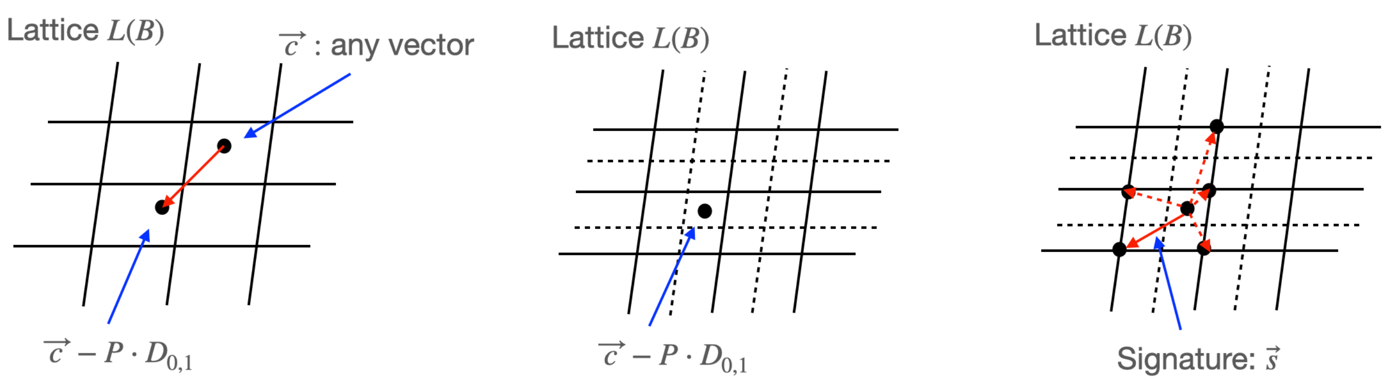


Randomized sampling

GPV Framework to Peikert's algorithm

Add randomness with discrete Gaussian R.V.

- Babai's rounding-off algorithm + discrete Gaussian r.v. → skewed(elliptical) Gaussian : the skew mirrors the geometry of the basis
- Perturbation(offline precomputation) + Babai's round-off algorithm + discrete Gaussian r.v.
- Main concept : summation of two discrete Gaussian r.v. → discrete Gaussian r.v.
- Define : matrix P, its covariance matrix $\Sigma_2 = PP^t$, private key matrix B, its covariance matrix $\Sigma_1 = r^2 \cdot BB^t$
- Offline phase : find the matrix P such as $\Sigma_1 + \Sigma_2 = s^2 I$
- Online phase : sample $\vec{x} \leftarrow P \cdot D_{0,1}$ and move $\vec{c}$ to $\vec{c} - \vec{x}$ and apply Babai's round-off algorithm + discrete Gaussian randomization such as $\lfloor B^{-1}(\vec{c} - \vec{x}) \rceil_r$, where $\lfloor v \rceil_r$ has distribution $v + D_{Z-v,r}$



Perturbation of $\vec{c}$ with $\rho_{\sqrt{\Sigma_2}}(\vec{x})$

Babai's round-off algorithm

randomize with $\rho_{\sqrt{\Sigma_1}}(\vec{x})$

FALCON : NTRU Lattices

Polynomial ring	$Z_q[x]/\phi(x)$, where $\phi(x) = x^n + 1$ for $n = 2^k$
NTRU Equation	$fG - gF = q \; mod \; \phi(x) f, g, F, G \in Z_q[x]/\phi(x)$,
Public Key	$A = (1 \;\; h)$, where $h = f^{-1} \cdot g \; mod \; q$
Private Key	$B = \begin{pmatrix} g & -f \\ G & -F \end{pmatrix} \rightarrow \det B = q \rightarrow \sqrt{q} \leq \|\|B\|\|_{GS}$

f and g are generated randomly by discrete Gaussian distribution $D_{Z,\sigma}$ with $\sigma = 1.17\sqrt{q/2n}$
F and G are calculated by solving the NTRU equation

$$B_{2n\times 2n} = \left(\begin{array}{cccc|cccc} g_0, & g_1, & \cdots, & g_{n-1}, & -f_0, & -f_1, & \cdots, & -f_{n-1} \\ -g_{n-1}, & g_0, & \cdots, & g_{n-2}, & f_{n-1}, & -f_1, & \cdots, & -f_{n-2} \\ & \vdots & & & & \vdots & & \\ -g_1, & -g_2, & \cdots, & g_0, & f_1, & f_2, & \cdots, & -f_0 \\ \hline G_0, & G_1, & \cdots, & G_{n-1}, & -F_0, & -F_1, & \cdots, & -F_{n-1} \\ -G_{n-1}, & G_0, & \cdots, & G_{n-2}, & F_{n-1}, & -F_1, & \cdots, & -F_{n-2} \\ & \vdots & & & & \vdots & & \\ -G_1, & -G_2, & \cdots, & G_0, & F_1, & F_2, & \cdots, & -F_0 \end{array}\right)$$

- Signature norm proportional to the Gram-Schmidt norm of B, $||B||_{GS} = \max_{\tilde{b}_i \in \tilde{B}} ||\tilde{B}_i||$
- $||B||_{GS}$ is minimized for $||(f, g)|| \approx 1.17\sqrt{q}$ (experimental result)
- Discard f and g if their norm is more than $1.17\sqrt{q} \rightarrow ||B||_{GS} \leq 1.17\sqrt{q}$

FALCON : Fast Fourier Sampling

How to find the lattice point v close to c

FALCON

Babai's nearest plane + Klein's algorithm (discrete Gaussian randomization) in FFT domain

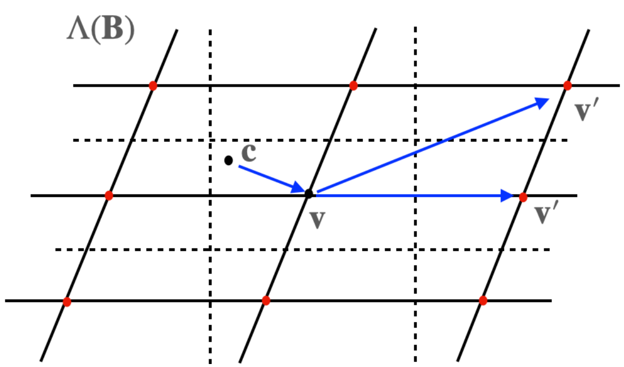
Sequential sampling

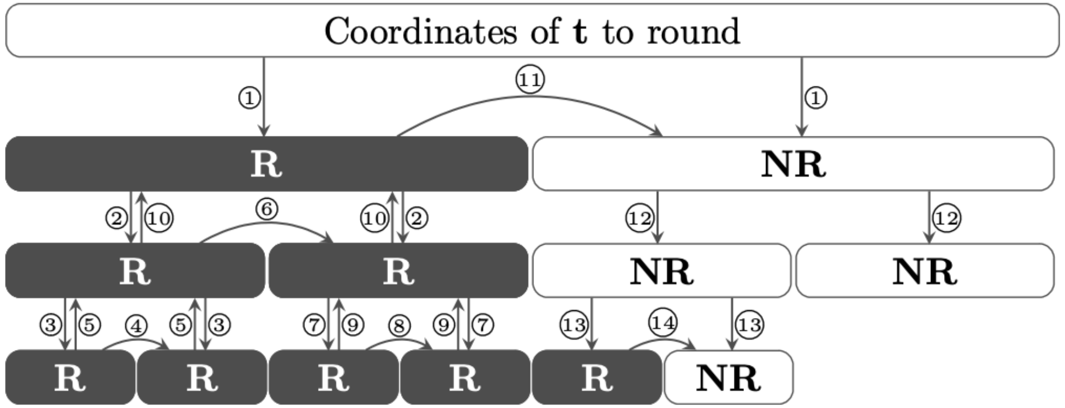
$$\left(t_0, t_1\right) \cdot B - \left(z_0\ z_1\right) \cdot B = P(\widetilde{B})$$

Parallelepiped of orthogonal basis $\widetilde{B}$

$z_1 : [t_1]$ + Gaussian random variable

$z_0 : \left[t_0 + \left(t_1 - z_1\right) \cdot l\right]$ + Gaussian random variable





FALCON : Verification

- **Verification**
  - Check that the received signature is short enough vector

Algorithm 7 verify(M, **sig**, **pk**, $\lfloor\beta^2\rfloor$)

Require: A message M, a signature **sig** $= (r, s_1)$, a public key **pk** $= h \in \mathbb{Z}_q[x]/\phi$, a bound $\lfloor\beta^2\rfloor$

Ensure: Accept or reject

1: $c \leftarrow H(M||r)$
2: $s_1 \leftarrow c - s_2 \cdot h \mod (\phi, q)$
3: **if** $||(s_1, s_2)|| \leq \lfloor\beta^2\rfloor$ **then**
4:     accept
5: **else**
6:     reject
7: **end if**

Complexity of FALCON

- Secure implementation of discrete Gaussian r.v.
- Floating point operation in the FFT and discrete Gaussian r.v.
- Recursive sequential sampling

- To overcome sequentail sampling : hybrid Peikert's algorithm and Falcon→MITAKA
- We tried in Peregrine : Babai's round-off algorithm + centered Binomial r.v. + NTT only in the signature process

Peregrine

- **Key Generation**

Public Key

$h(x) = f^{-1}(x) \cdot g(x) \in Z_q[x]/\phi(x)$

Random polynomials $f(x), g(x) \in Z_q[x]/\phi(x)$ generated by random variable with centered binomial distribution

$$P[X = x] = \frac{\mu!}{((\mu/2) + x)! \cdot ((\mu/2) - x)!} \cdot 2^{-\mu},$$
$$-\mu/2 \leq x \leq \mu/2$$

- The centered Binomial distribution with μ=24 or 26 is very similar with the discrete Gaussian distribution which is used for random public key generation in FALCON.
- Simulation result in Peregrine shows that the proposed centered binomial distribution generates the public key and secret key, which are derived by solving the NTRU equation, successfully in similar running time with FALCON.

Peregrine

- **Key Generation**
  - Implementation of the centered binomial random variable is much simpler than the discrete Gaussian random variable and can cope with the side channel attack.

Algorithm of Discrete Gaussian

Algorithm 12 BaseSampler()

Require: -

Ensure: An integer $z_0 \in \{0, \ldots, 18\}$ such that $z \sim \chi$ ▷ χ is uniquely defined by (3.33)

- $u \leftarrow$ UniformBits(72) ▷ See (3.32)
- $z_0 \leftarrow 0$
- for $i = 0, \ldots, 17$ do
  - $z_0 \leftarrow z_0 + [\![u < \mathtt{RCDT}[i]]\!]$ ▷ Note that one should use RCDT, not pdt or cdt
- return z_0

Algorithm 13 ApproxExp(x, ccs)

Require: Floating-point values $x \in [0, \ln(2)]$ and $ccs \in [0, 1]$

Ensure: An integral approximation of $2^{63} \cdot ccs \cdot \exp(-x)$

- C = [0x00000004741183A3,0x00000036548CFC06,0x0000024FDCBF140A,0x0000171D939DE045, 0x0000D00CF58F6F84, 0x000680681CF796E3, 0x002D82D8305B0FEA, 0x011111110E066FD0, 0x0555555555070F00, 0x155555555581FF00, 0x400000000002B400, 0x7FFFFFFFFFFF4800, 0x8000000000000000]
- $y \leftarrow C[0]$ ▷ y and z remain in $\{0, \ldots, 2^{63} - 1\}$ the whole algorithm.
- $z \leftarrow \lfloor 2^{63} \cdot x \rfloor$
- for $1 = 1, \ldots, 12$ do
  - $y \leftarrow C[u] - (z \cdot y) >> 63$ ▷ $(z \cdot y)$ fits in 126 bits, but we only need the top 63 bits
- $z \leftarrow \lfloor 2^{63} \cdot ccs \rfloor$
- $y \leftarrow (z \cdot y) >> 63$
- return y

Algorithm 14 BerExp(x, ccs)

Require: Floating point values $x, ccs \geq 0$

Ensure: A single bit, equal to 1 with probability $\approx ccs \cdot \exp(-x)$

- $s \leftarrow \lfloor x / \ln(2) \rfloor$ ▷ Compute the unique decomposition $x = 2^s \cdot r$, with $(r, s) \in [0, \ln 2) \times \mathbb{Z}^+$
- $r \leftarrow x - s \cdot \ln(2)$
- $s \leftarrow \min(s, 63)$
- $z \leftarrow (2 \cdot \text{ApproxExp}(r, ccs) - 1) >> s$ ▷ $z \approx 2^{64-s} \cdot ccs \cdot \exp(-r) = 2^{64} \cdot ccs \cdot \exp(-x)$
- $i \leftarrow 64$
- do
  - $i \leftarrow i - 8$
  - $w \leftarrow \text{UniformBits}(8) - ((z >> i)\ \&\ \text{0xFF})$
- while $((w = 0)$ and $(i > 0))$ ▷ This loop does not need to be done in constant-time
- return $[\![w < 0]\!]$ ▷ Return 1 with probability $2^{-64} \cdot z \approx ccs \cdot \exp(-x)$

Algorithm 15 SamplerZ(μ, σ')

Require: Floating-point values $\mu, \sigma' \in \mathcal{R}$ such that $\sigma' \in [\sigma_{\min}, \sigma_{\max}]$

Ensure: An integer $z \in \mathbb{Z}$ sampled from a distribution very close to $D_{\mathbb{Z},\mu,\sigma'}$

- $r \leftarrow \mu - \lfloor \mu \rfloor$ ▷ r must be in $[0, 1)$
- $ccs \leftarrow \sigma_{\min} / \sigma'$ ▷ ccs helps to make the algorithm running time independent of σ'
- while (1) do
  - $z_0 \leftarrow$ BaseSampler()
  - $b \leftarrow$ UniformBits(8) & 0x1
  - $z \leftarrow b + (2 \cdot b - 1) z_0$
  - $x \leftarrow \frac{(z-r)^2}{2\sigma'^2} - \frac{z_0^2}{2\sigma_{\max}^2}$
  - if (BerExp$(x, ccs) = 1$) then
    - return $z + \lfloor \mu \rfloor$

Algorithm of Centered Binomial

Pseudo random number	0	1	0	1	1	0	0	1	0	0	0	0	……..

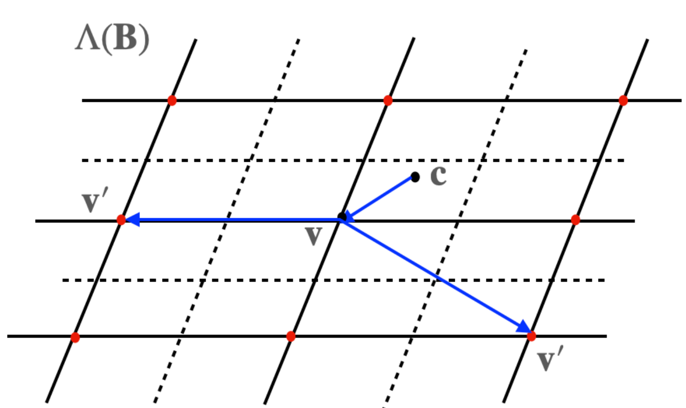
# of 1	1	2	1	0
Output	$1 - 2 = -1$		$1 - 0 = 1$	

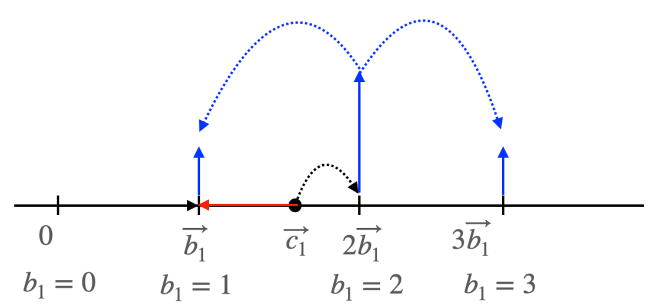
```
for(i = 0; i<10; i++)
{
    r = get_rng_u64(rng);
    printf("%l64u",r);
    d = 0;
    for (j = 0; j < 3; j++)
      d += (r >> j) & 0x249249;

    a[0] = d & 0x7;
    b[0] = (d >> 3) & 0x7;
    a[1] = (d >> 6) & 0x7;
    b[1] = (d >> 9) & 0x7;
    a[2] = (d >> 12) & 0x7;
    b[2] = (d >> 15) & 0x7;
    a[3] = (d >> 18) & 0x7;
    b[3] = (d >> 21);

    s[4 * i + 0] = (int16_t)(a[0] - b[0]);
    s[4 * i + 1] = (int16_t)(a[1] - b[1]);
    s[4 * i + 2] = (int16_t)(a[2] - b[2]);
    s[4 * i + 3] = (uint16_t)(a[3] - b[3]);
}
```

Peregrine

- **Signature**

Based on the GPV framework	Lattice-based hash-and-sign signature
Babai's round-off algorithm + randomization	Integer operation only without floating point FFT operation
RNS + NTT	ModDown algorithm is used to implement the round-off algorithm
Randomization	Centered binomial distribution with $\mu = 6$ is used to insert the randomness





Randomize with centered Binomial r.v.

Peregrine

- **Signature**

Message Poly.	$c = H(M\|\|r) \in Z_q[x]/\phi(x)$
Signature	Short vector $\mathbf{s} = (s_1, s_2) \in (Z_q[x]/\phi(x))^2$ satisfying $\mathbf{s} \cdot \mathbf{A}^{\mathbf{t}} = c$, where $\mathbf{A} = (1, h)$

- Need to find a polynomial vector $\mathbf{t} = \left(t_1, t_2\right) \in \left(Q[x]/\phi\right)^2$ satisfying $\mathbf{t} \cdot \mathbf{B} = \mathbf{c} = (c, 0)$
- $\mathbf{t} = \left(t_1, t_2\right) = \left(\frac{c \cdot f}{q}, -\frac{c \cdot F}{q}\right) = (R_1 + I_1,\ R_2 + I_2)$, where $R_l = \sum_{i=0}^{n-1} r_{lj} \cdot x^j$, $l = 1, 2,\ -0.5 < r_{lj} \leq 0.5$ and $I_i = \sum_{j=0}^{n-1} i_{lj} \cdot x^j, l = 1, 2, i_{lj} \in Z$.

Signature Generation

$$s = (t - z') \cdot B = \left(c - \left(I_1 + J_1\right) \cdot g - \left(I_2 + J_2\right) \cdot G, \left(I_1 + J_1\right) \cdot f + \left(I_2 + J_2\right) \cdot F\right) mod(\phi, q)$$

Problems and Future Works

- Randomization with centered Binomial distribution is not enough to protect the private key information B.
- In order for FALCON to be used on various platforms, further research on reducing the implementation complexity of FALCON is needed.

Reference

- [Babai85]L Babai. On lovasz' lattice reduction and the nearest lattice point problem. In Proceedings on STACS 85 2Nd Annual Symposium on Theoretical Aspects of Computer Science, New York, NY, USA, 1985. Springer-Verlag New York, Inc.
- [Babai86]László Babai. On lovasz' lattice reduction and the nearest lattice point problem. Combinatorica, 6(1), 1986.
- [Pei10]Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, CRYPTO 2010, volume 6223 of LNCS, pages 80–97, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.
- [GPV08]Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new crypto- graphic constructions. In Richard E. Ladner and Cynthia Dwork, editors, 40th ACM STOC, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press.
- [NR06]Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In Serge Vaudenay, editor, EUROCRYPT 2006, volume 4004 of LNCS, pages 271–288, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.
- [MITAKA]Espitau, T. *et al.* (2022). MITAKA: A Simpler, Parallelizable, Maskable Variant of FALCON. In: Dunkelman, O., Dziembowski, S. (eds) Advances in Cryptology – EUROCRYPT 2022. EUROCRYPT 2022. Lecture Notes in Computer Science, vol 13277. Springer, Cham.

Thank You!