Cryptanalysis of NTRU with two Public Keys

Abderrahmane Nitaj

Laboratoire de Mathématiques Nicolas Oresme
Université de Caen Basse Normandie, France

Caen, 24 novembre 2011

CONTENU

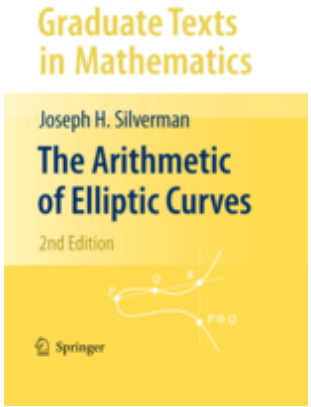
CONTENU

NTRU

NTRU

- Inventé en 1996 par Hoffstein, Pipher et Silverman.



- Problème difficile

Hard problem: Convolutional Factorization

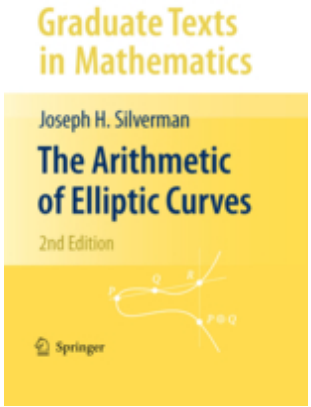
Let $h \in \mathbb{Z}_q[X]/\left(X^N - 1\right)$ be a polynomial. Find two small polynomials $f, g \in \mathbb{Z}_q[X]/\left(X^N - 1\right)$ such that $f * g = h$.

NTRU

NTRU

- Inventé en 1996 par Hoffstein, Pipher et Silverman.



- Problème difficile

Hard problem: Convolutional Factorization

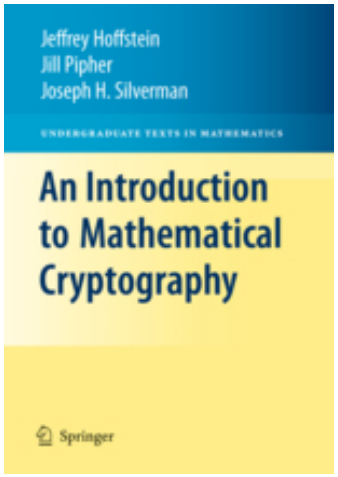
Let $h \in \mathbb{Z}_q[X]/\left(X^N-1\right)$ *be a polynomial. Find two small polynomials* $f, g \in \mathbb{Z}_q[X]/\left(X^N-1\right)$ *such that* $f * g = h$.

NTRU

NTRU

- Inventé en 1996 par Hoffstein, Pipher et Silverman.



- Problème difficile

Hard problem: Shortest Vector Problem (SVP)

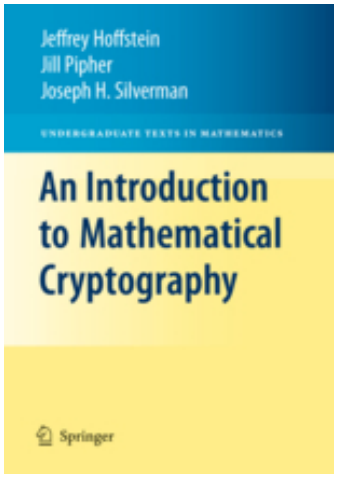
Let $\mathcal{L}$ be a lattice. Find a shortest nonzero vector in $\mathcal{L}$.

NTRU

NTRU

- Inventé en 1996 par Hoffstein, Pipher et Silverman.



- Problème difficile

Hard problem: Shortest Vector Problem (SVP)

Let $\mathcal{L}$ be a lattice. Find a shortest nonzero vector in $\mathcal{L}$.

Past, Present and Future of NTRU

Past

- **NTRUEncrypt**: public key cryptography. Invented in 1996 by Joseph H. Silverman, Jeffrey Hoffstein, Jill Pipher (+ Daniel Lieman).
- **NSS**: digital signature scheme. Invented in 2000 by the same team.
- **NSS** has been broken several times.
- **NTRUSign**: digital signature scheme. Invented in 2000 by Joseph H. Silverman, Jeffrey Hoffstein, Jill Pipher.

Past, Present and Future of NTRU

Present

- Approved for standardization by the Institute of Electrical and Electronics Engineers (IEEE) in 2009.
- Adopted as X9 Standard for Data Protection in 2011.
- Acquired by Security Innovation in 2009.

Past, Present and Future of NTRU

Future

- Based on the shortest vector problem in a lattice which is NP-hard.
- Alternative to RSA and ECC.
- A promising candidate for being quantum computer resistant.

CONTENU

Ring of Convolution $\mathcal{R} = \mathbb{Z}[X]/(X^N - 1)$

Polynomials

$f = \sum_{i=0}^{N-1} f_i X^i, \qquad g = \sum_{i=0}^{N-1} g_i X^i,$

Sum

$f + g = (f_0 + g_0, f_1 + g_1, \cdots, f_{N-1} + g_{N-1}).$

Product

$f * g = h = (h_0, h_1, \cdots, h_{N-1})$ with

$$h_k = \sum_{i+j \equiv k \pmod N} f_i g_j.$$

Ring of Convolution $\mathcal{R} = \mathbb{Z}[X]/(X^N - 1)$

Polynomials

$f = \sum_{i=0}^{N-1} f_i X^i, \qquad g = \sum_{i=0}^{N-1} g_i X^i,$

Sum

$f + g = (f_0 + g_0, f_1 + g_1, \cdots, f_{N-1} + g_{N-1}).$

Product

$f * g = h = (h_0, h_1, \cdots, h_{N-1})$ with

$$h_k = \sum_{i+j \equiv k \pmod N} f_i g_j.$$

Ring of Convolution $\mathcal{R} = \mathbb{Z}[X]/(X^N - 1)$

Polynomials

$f = \sum_{i=0}^{N-1} f_i X^i, \qquad g = \sum_{i=0}^{N-1} g_i X^i,$

Sum

$f + g = (f_0 + g_0, f_1 + g_1, \cdots, f_{N-1} + g_{N-1}).$

Product

$f * g = h = (h_0, h_1, \cdots, h_{N-1})$ with

$$h_k = \sum_{i+j \equiv k \pmod N} f_i g_j.$$

Ring of Convolution $\mathcal{R} = \mathbb{Z}[X]/(X^N - 1)$

Convolution

$$\underbrace{f = (f_0, f_1, \cdots, f_{N-1}), \qquad g = (g_0, g_1, \cdots, g_{N-1})}_{f * g = h = (h_0, h_1, \cdots, h_{N-1})}.$$

	1	X	$\cdots$	X^k	$\cdots$	X^{N-1}
	f_0g_0	f_0g_1	$\cdots$	f_0g_k	$\cdots$	f_0g_{N-1}
$+$	f_1g_{N-1}	f_1g_0	$\cdots$	f_1g_{k-1}	$\cdots$	f_1g_{N-2}
$+$	f_2g_{N-2}	f_2g_{N-1}	$\cdots$	f_2g_{k-2}	$\cdots$	f_2g_{N-3}
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\cdots$	$\vdots$	$\vdots$
$+$	$f_{N-2}g_2$	$f_{N-2}g_3$	$\cdots$	$f_{N-2}g_{k+2}$	$\cdots$	$f_{N-2}g_1$
$+$	$f_{N-1}g_1$	$f_{N-1}g_2$	$\cdots$	$f_{N-1}g_{k+1}$	$\cdots$	$f_{N-1}g_0$
$h =$	h_0	h_1	$\cdots$	h_k	$\cdots$	h_{N-1}

NTRU Parameters

- $N =$ a prime number (e.g. $N = 167,\ 251,\ 347,\ 503$).
- $q =$ a large modulus (e.g. $q = 128,\ 256$).
- $p =$ a small modulus (e.g. $p = 3$).

Key Generation:

- Randomly choose two private polynomials f and g.
- Compute the inverse of f modulo q: $f * f_q = 1 \pmod{q}$.
- Compute the inverse of f modulo p: $f * f_p = 1 \pmod{p}$.
- Compute the public key $h = f_q * g \pmod{q}$.

Encryption:

- m is a plaintext in the form of a polynomial mod q.
- Randomly choose a private polynomial r.
- Compute the encrypted message $e = m + pr * h \pmod{q}$.

Decryption:

- Compute $a = f * e = f * (m + pr * h) = f * m + pr * g \pmod{q}$.
- Compute $a * f_p = (f * m + pr * g) * f_p = m \pmod{p}$.

Key Generation:

- Randomly choose two private polynomials f and g.
- Compute the inverse of f modulo q: $f * f_q = 1 \pmod{q}$.
- Compute the inverse of f modulo p: $f * f_p = 1 \pmod{p}$.
- Compute the public key $h = f_q * g \pmod{q}$.

Encryption:

- m is a plaintext in the form of a polynomial mod q.
- Randomly choose a private polynomial r.
- Compute the encrypted message $e = m + pr * h \pmod{q}$.

Decryption:

- Compute $a = f * e = f * (m + pr * h) = f * m + pr * g \pmod{q}$.
- Compute $a * f_p = (f * m + pr * g) * f_p = m \pmod{p}$.

Key Generation:

- Randomly choose two private polynomials f and g.
- Compute the inverse of f modulo q: $f * f_q = 1 \pmod{q}$.
- Compute the inverse of f modulo p: $f * f_p = 1 \pmod{p}$.
- Compute the public key $h = f_q * g \pmod{q}$.

Encryption:

- m is a plaintext in the form of a polynomial mod q.
- Randomly choose a private polynomial r.
- Compute the encrypted message $e = m + pr * h \pmod{q}$.

Decryption:

- Compute $a = f * e = f * (m + pr * h) = f * m + pr * g \pmod{q}$.
- Compute $a * f_p = (f * m + pr * g) * f_p = m \pmod{p}$.

CONTENU

Lattice

Definition

1. $b_1, b_2, \cdots, b_n \in \mathbb{R}^n$, n linearly independant vectors.
2. $\mathcal{B} = \{b_1, b_2, \cdots b_n\}$.
3. The lattice spanned by $\mathcal{B}$ is

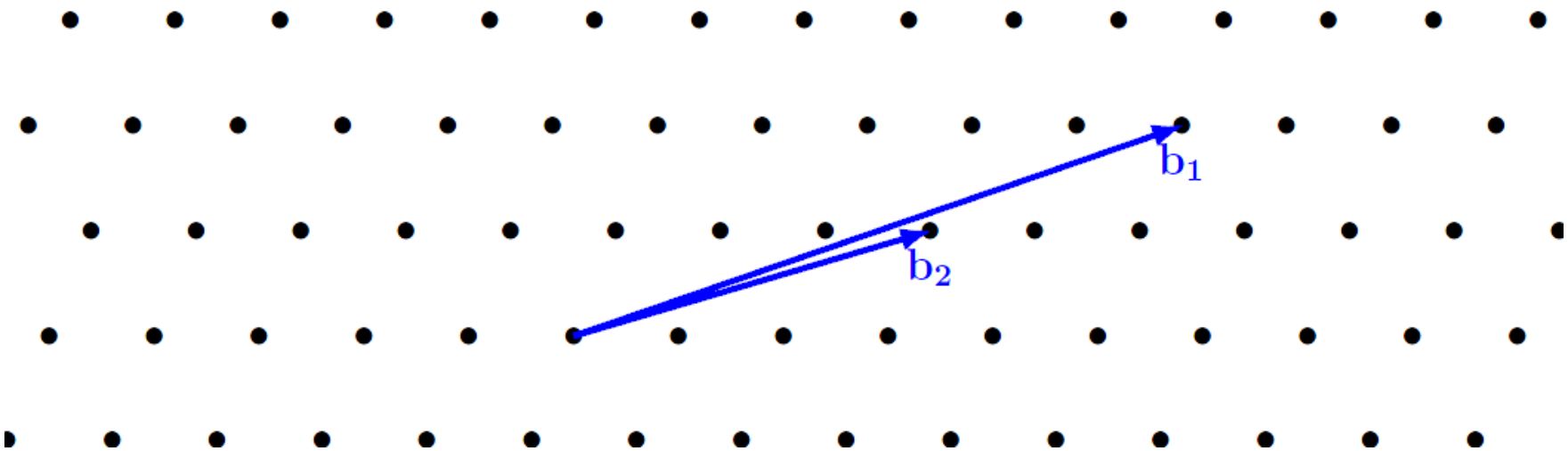
$$\mathcal{L} = \left\{ \sum_{i=1}^{n} \lambda_i b_i, \qquad \lambda_i \in \mathbb{Z} \right\}.$$

Invariants

1. $\dim(\mathcal{L}) = n$.
2. $\det(\mathcal{L}) = \text{volume}\left\{\sum_{i=1}^{n} \alpha_i b_i, \quad 0 \leq \alpha_i < 1\right\} = \sqrt{\det\left(BB^t\right)}$.

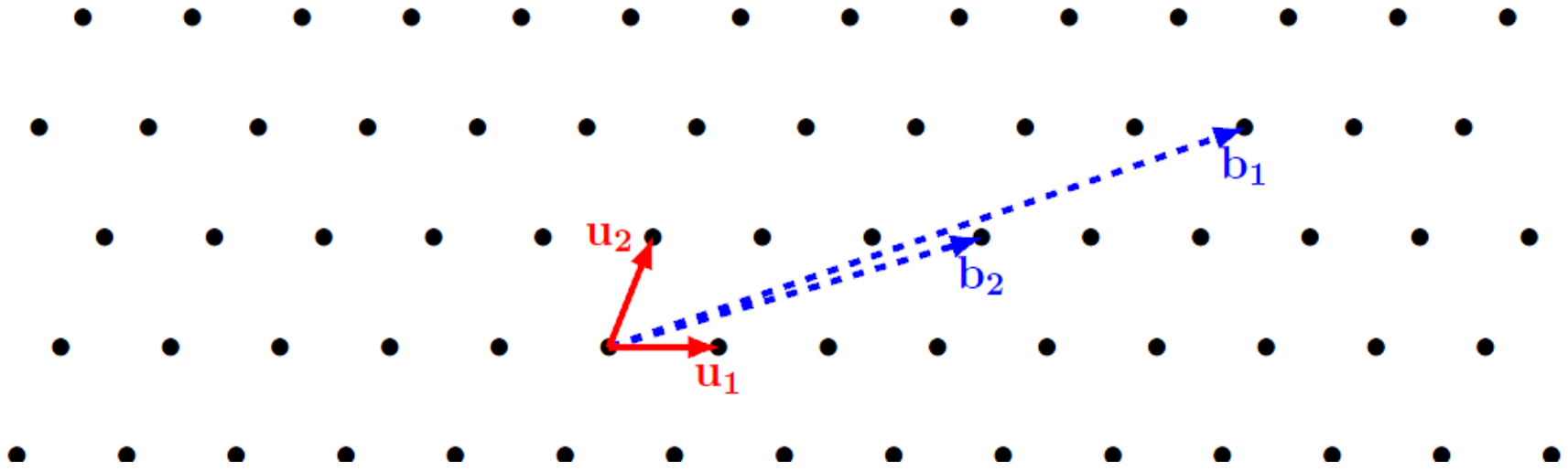
Bad Basis vs Good Basis

A lattice with a bad basis



Bad Basis vs Good Basis

A lattice with a good basis



The LLL algorithm

1. **LLL**=Lenstra-Lenstra- Lovász, 1982.
2. Polynomial time algorithm.

Theorem

- *Let $\mathcal{L}$ be a lattice. The LLL algorithm finds a basis $(b_1, b_2, \cdots, b_n)$ with*
  1. a short vector:
  $$||b_1|| \leq 2^{\frac{n-1}{4}} \det(\mathcal{L})^{1/n}.$$
  2. shortness:
  $$\det(\mathcal{L}) \leq \prod_{i=1}^{n} ||b_i| \leq \left(\frac{4}{3}\right)^{\frac{n(n-1)}{4}} \det(\mathcal{L}).$$

The Gaussian Heuristic

Gauß, 1777-1855

- If $\mathcal{L} \subset \mathbb{R}^n$ is a lattice, how long would we expect its shortest vector to be?
- The Gaussian Heuristic: The shortest nonzero vector in a lattice $\mathcal{L} \subset \mathbb{R}^n$ has length approximately

$$\sigma(\mathcal{L}) = \sqrt{\frac{\dim(L)}{2\pi e}} (\det \mathcal{L})^{1/\dim(\mathcal{L})}.$$

Proof

- Minkowski's Theorem: *Let $\mathcal{L}$ be a lattice of dimension n. If $S \subset \mathbb{R}^n$ is a symmetric convex set whose volume satisfies $Vol(S) > 2^n \det(L)$, then S contains a nonzero lattice vector.*
- $S = B(r)$ is a n-dimensional sphere of radius r.
- The volume of $B(r)$ is $\mathrm{Vol}(B(r)) = \dfrac{\pi^{n/2} r^n}{\Gamma(1 + n/2)} \approx \left(\dfrac{2\pi e}{n}\right)^{n/2} r^n.$
- Choose r to satisfy $\mathrm{Vol}(S) > 2^n \det(L)$.
- The shortest nonzero vector in a lattice $\mathcal{L} \subset \mathbb{R}^n$ has length approximately

$$\sigma(\mathcal{L}) = \sqrt{\frac{\dim(\mathcal{L})}{2\pi e}} (\det \mathcal{L})^{1/\dim(\mathcal{L})}.$$

CONTENU

Attack of Coppersmith and Shamir

Lattice based attack, Coppersmith and Shamir, 1998.

- Principle: Since $h = f_q * g \pmod q$, then $f * h = g \pmod q$ where f and g are short polynomials.
- Lattice: $\mathcal{L}_{CS} = \{(a,b) \in \mathbb{Z}^{2N}, \quad h * a = b \pmod q\}$.
- Then $h * f = g \pmod q$ transforms to $h * f + qu = g$:

$$\left[\begin{array}{c|c} I_N & 0_N \\ \hline \begin{matrix} h_0 & h_{N-1} & \cdots & h_1 \\ h_1 & h_0 & \cdots & h_2 \\ \vdots & \vdots & \ddots & \vdots \\ h_{N-1} & h_{N-2} & \cdots & h_0 \end{matrix} & qI_N \end{array}\right] \left[\begin{array}{c} f_0 \\ \vdots \\ f_{N-1} \\ \hline u_0 \\ \vdots \\ u_{N-1} \end{array}\right] = \left[\begin{array}{c} f_0 \\ \vdots \\ f_{N-1} \\ \hline g_0 \\ \vdots \\ g_{N-1} \end{array}\right]$$

- Apply the LLL algorithm to $\mathcal{L}_{CS}$ to find (f,g).

Attack of Coppersmith and Shamir

Lattice based attack, Coppersmith and Shamir, 1998.

- Principle: Since $h = f_q * g \pmod q$, then $f * h = g \pmod q$ where f and g are short polynomials.
- Lattice: $\mathcal{L}_{CS} = \{(a,b) \in \mathbb{Z}^{2N}, \quad h * a = b \pmod q\}$.
- Then $h * f = g \pmod q$ transforms to $h * f + qu = g$:

$$\left[\begin{array}{ccccc|c} & & I_N & & & 0_N \\ \hline h_0 & h_{N-1} & \cdots & h_1 & & \\ h_1 & h_0 & \cdots & h_2 & & qI_N \\ \vdots & \vdots & \ddots & \vdots & & \\ h_{N-1} & h_{N-2} & \cdots & h_0 & & \end{array}\right] \left[\begin{array}{c} f_0 \\ \vdots \\ f_{N-1} \\ \hline u_0 \\ \vdots \\ u_{N-1} \end{array}\right] = \left[\begin{array}{c} f_0 \\ \vdots \\ f_{N-1} \\ \hline g_0 \\ \vdots \\ g_{N-1} \end{array}\right]$$

- Apply the LLL algorithm to $\mathcal{L}_{CS}$ to find (f,g).

Attack of Coppersmith and Shamir

Lattice based attack, Coppersmith and Shamir, 1998.

- Principle: Since $h = f_q * g \pmod q$, then $f * h = g \pmod q$ where f and g are short polynomials.
- Lattice: $\mathcal{L}_{CS} = \{(a,b) \in \mathbb{Z}^{2N}, \quad h * a = b \pmod q\}$.
- Then $h * f = g \pmod q$ transforms to $h * f + qu = g$:

$$\left[\begin{array}{cccc|c} & & I_N & & 0_N \\ \hline h_0 & h_{N-1} & \cdots & h_1 & \\ h_1 & h_0 & \cdots & h_2 & \\ \vdots & \vdots & \ddots & \vdots & qI_N \\ h_{N-1} & h_{N-2} & \cdots & h_0 & \end{array}\right] \left[\begin{array}{c} f_0 \\ \vdots \\ f_{N-1} \\ - - - \\ u_0 \\ \vdots \\ u_{N-1} \end{array}\right] = \left[\begin{array}{c} f_0 \\ \vdots \\ f_{N-1} \\ - - - \\ g_0 \\ \vdots \\ g_{N-1} \end{array}\right]$$

- Apply the LLL algorithm to $\mathcal{L}_{CS}$ to find (f,g).

Attack of Coppersmith and Shamir

Lattice based attack, Coppersmith and Shamir, 1998.

- Principle: Since $h = f_q * g \pmod q$, then $f * h = g \pmod q$ where f and g are short polynomials.
- Lattice: $\mathcal{L}_{CS} = \{(a,b) \in \mathbb{Z}^{2N}, \quad h * a = b \pmod q\}$.
- Then $h * f = g \pmod q$ transforms to $h * f + qu = g$:

$$\left[\begin{array}{cccc|c} & & I_N & & 0_N \\ \hline h_0 & h_{N-1} & \cdots & h_1 & \\ h_1 & h_0 & \cdots & h_2 & \\ \vdots & \vdots & \ddots & \vdots & qI_N \\ h_{N-1} & h_{N-2} & \cdots & h_0 & \end{array}\right] \left[\begin{array}{c} f_0 \\ \vdots \\ f_{N-1} \\ - - - \\ u_0 \\ \vdots \\ u_{N-1} \end{array}\right] = \left[\begin{array}{c} f_0 \\ \vdots \\ f_{N-1} \\ - - - \\ g_0 \\ \vdots \\ g_{N-1} \end{array}\right]$$

- Apply the LLL algorithm to $\mathcal{L}_{CS}$ to find (f,g).

The Gaussian Heuristics

The Gaussian Heuristics in $\mathcal{L}_{CS}$

- The shortest nonzero vector in a lattice $\mathcal{L} \subset \mathbb{R}^n$ has length approximately

$$\sigma(\mathcal{L}) = \sqrt{\frac{\dim(\mathcal{L})}{2\pi e}} (\det \mathcal{L})^{1/\dim(\mathcal{L})}.$$

- In $\mathcal{L}_{CS}$, $\dim(\mathcal{L}) = 2N$, $\det(\mathcal{L}) = q^N$.
- The shortest nonzero vector in $\mathcal{L}_{CS}$ has length approximately

$$\sigma = \sqrt{\frac{qN}{\pi e}}.$$

CONTENU

The New Attack

Equations

- Suppose $h = f_q * g \pmod{q}$ and $h' = F'_q * G' \pmod{q}$.
- Since f is invertible modulo q, then $h' = f_q * g' \pmod{q}$.
- Using $f * h = g \pmod{q}$ and $f * h' = g' \pmod{q}$, we get

$$f * (h - h') = g - g' \pmod{q}.$$

- Lattice: $\mathcal{L}(h, h') = \{(a, b) \in \mathbb{Z}^{2N}, \quad a * (h - h') = b \pmod{q}\}$.
- $f * (h - h') = g - g' \pmod{q}$ transforms to

$$f * (h - h') + qu = g - g'.$$

The New Attack

Equations

- Suppose $h = f_q * g \pmod q$ and $h' = F'_q * G' \pmod q$.
- Since f is invertible modulo q, then $h' = f_q * g' \pmod q$.
- Using $f * h = g \pmod q$ and $f * h' = g' \pmod q$, we get

$$f * (h - h') = g - g' \pmod q.$$

- Lattice: $\mathcal{L}(h, h') = \{(a, b) \in \mathbb{Z}^{2N}, \quad a * (h - h') = b \pmod q\}$.
- $f * (h - h') = g - g' \pmod q$ transforms to

$$f * (h - h') + qu = g - g'.$$

The New Attack

Equations

- Suppose $h = f_q * g \pmod q$ and $h' = F'_q * G' \pmod q$.
- Since f is invertible modulo q, then $h' = f_q * g' \pmod q$.
- Using $f * h = g \pmod q$ and $f * h' = g' \pmod q$, we get

$$f * (h - h') = g - g' \pmod q.$$

- Lattice: $\mathcal{L}(h, h') = \{(a, b) \in \mathbb{Z}^{2N}, \quad a * (h - h') = b \pmod q\}$.
- $f * (h - h') = g - g' \pmod q$ transforms to

$$f * (h - h') + qu = g - g'.$$

The New Attack

Equations

- Suppose $h = f_q * g \pmod q$ and $h' = F'_q * G' \pmod q$.
- Since f is invertible modulo q, then $h' = f_q * g' \pmod q$.
- Using $f * h = g \pmod q$ and $f * h' = g' \pmod q$, we get

$$f * (h - h') = g - g' \pmod q.$$

- Lattice: $\mathcal{L}(h, h') = \{(a, b) \in \mathbb{Z}^{2N}, \quad a * (h - h') = b \pmod q\}$.
- $f * (h - h') = g - g' \pmod q$ transforms to

$$f * (h - h') + qu = g - g'.$$

The New Attack

Equations

- Suppose $h = f_q * g \pmod q$ and $h' = F'_q * G' \pmod q$.
- Since f is invertible modulo q, then $h' = f_q * g' \pmod q$.
- Using $f * h = g \pmod q$ and $f * h' = g' \pmod q$, we get

$$f * (h - h') = g - g' \pmod q.$$

- Lattice: $\mathcal{L}(h, h') = \{(a, b) \in \mathbb{Z}^{2N}, \quad a * (h - h') = b \pmod q\}$.
- $f * (h - h') = g - g' \pmod q$ transforms to

$$f * (h - h') + qu = g - g'.$$

The New Attack

Lattice

- The polynomial equation $f * (h - h') + qu = g - g'$ can be represented as

$$\left[\begin{array}{cccc|c} & & I_N & & 0_N \\ \hline h_0 - h'_0 & h_{N-1} - h'_{N-1} & \cdots & h_1 - h'_1 & \\ h_1 - h'_1 & h_0 - h'_0 & \cdots & h_2 - h'_2 & \\ \vdots & \vdots & \ddots & \vdots & qI_N \\ h_{N-1} - h'_{N-1} & h_{N-2} - h'_{N-2} & \cdots & h_0 - h'_0 & \end{array}\right] \left[\begin{array}{c} f_0 \\ \vdots \\ f_{N-1} \\ \hline u_0 \\ \vdots \\ u_{N-1} \end{array}\right] = \left[\begin{array}{c} f_0 \\ \vdots \\ f_{N-1} \\ \hline g_0 - g'_0 \\ \vdots \\ g_{N-1} - g'_{N-1} \end{array}\right]$$

- Apply the LLL algorithm to $\mathcal{L}(h, h')$ to find $(f, g - g')$.

The Gaussian Heuristics

The Gaussian Heuristics in $\mathcal{L}(h, h')$

- The shortest nonzero vector in a lattice $\mathcal{L} \subset \mathbb{R}^n$ has length approximately

$$\sigma(\mathcal{L}) = \sqrt{\frac{\dim(\mathcal{L})}{2\pi e}} (\det L)^{1/\dim(L)}.$$

- In $\mathcal{L}(h, h')$, $\dim(L) = 2N$, $\det(\mathcal{L}) = q^N$.
- The shortest nonzero vector in $\mathcal{L}(h, h')$ has length approximately

$$\sigma = \sqrt{\frac{qN}{\pi e}}.$$

The New Attack

The Gaussian Heuristics

- **Attack of Coppersmith and Shamir:** The ratio of the vectors $(f,g),(f,g') \in \mathcal{L}_{CS}$ to the expected shortest nonzero vector in the lattice $\mathcal{L}_{CS}$ is

$$c_1 = \frac{\sqrt{\|f\|^2 + \|g\|^2}}{\sqrt{\frac{qN}{\pi e}}}. \qquad c_1' = \frac{\sqrt{\|f\|^2 + \|g'\|^2}}{\sqrt{\frac{qN}{\pi e}}}.$$

- **The new attack:** The ratio of the vector $(f,g) \in \mathcal{L}(h,h')$ to the expected shortest nonzero vector in the lattice $\mathcal{L}(h,h')$ is

$$c_2 = \frac{\sqrt{\|f\|^2 + \|g-g'\|^2}}{\sqrt{\frac{qN}{\pi e}}}.$$

- Then $c_2 < c_1$ if $\|g-g'\| < \min(\|g\|, \|g'\|)$.

Conclusion

- The new attack works for NTRU with two public keys $h = f_q * g \pmod q$ and $h' = f_q * g' \pmod q$.
- The new attack is more effective than Coppermith and Shamir's attack if $\|g - g'\| < \min(\|g\|, \|g'\|)$.

Merci