NTRU and Lattice-Based Crypto: Past, Present, and Future

Joseph H. Silverman

Brown University

The Mathematics of Post-Quantum Cryptography
DIMACS Center, Rutgers University

January 12–16, 2015

Some Definitions, Some Notation, and Some Theory

Lattices

A **lattice** L is a (maximal) discrete subgroup of $\mathbb{R}^n$, or equivalently,

$$L = \{a_1\boldsymbol{v}_1 + \cdots + a_n\boldsymbol{v}_n : a_1, \ldots, a_n \in \mathbb{Z}\}$$

for some $\mathbb{R}$-basis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ of $\mathbb{R}^n$. If $L \subset \mathbb{Z}^n$, it is called an **integral lattice**.

The **discriminant** of L is the volume of a fundamental domain

$$\mathrm{Disc}(L) = \mathrm{Vol}\{t_1\boldsymbol{v}_1 + t_2\boldsymbol{v}_2 + \cdots + t_n\boldsymbol{v}_n : 0 \le t_i < 1\}.$$

Lattices have been extensively studied since (at least) the 19th century and have applications througout mathematics, physics, and computer science.

For many applications, both theoretical and practical, one is interested in finding short non-zero vectors in L.

Short Vectors — Theory

A famous theorem of Hermite (1870s) says that a lattice L contains a non-zero vector $\boldsymbol{v} \in L$ satisfying

$$\|\boldsymbol{v}\| \le \gamma_n \operatorname{Disc}(L)^{1/n}.$$

The optimal value for γ_n, called Hermite's constant, is known only for $n \le 8$, but for large n we have

$$\sqrt{n/2\pi e} \;\lesssim\; \gamma_n \;\lesssim\; \sqrt{n/\pi e}.$$

The **shortest vector problem** (**SVP**) is that of determining the shortest non-zero vector in L. Hermite's theorem suggests that in a "random" lattice,

$$\min\{\|\boldsymbol{v}\| : \mathbf{0} \ne \boldsymbol{v} \in L\} \asymp \sqrt{n} \cdot \operatorname{Disc}(L)^{1/n}.$$

The **closest vector problem** (**CVP**) is that of determining the vector in L that is closest to a given non-lattice vector $\boldsymbol{w}$.

Short Vectors — Practice

In low dimension it is not too hard to find short(est) vectors. But as the dimension increases, it becomes very hard. A computational breakthrough is the

LLL Algorithm 1982. Let $n = \dim(L)$ and let $\lambda(L)$ denote the length of shortest non-zero vector in L. Then there is a polynomial time algorithm to find a non-zero vector $\boldsymbol{v} \in L$ satisfying

$$\|\boldsymbol{v}\| \le 2^{n/2}\lambda(L).$$

Many improvements have been made, but there is currently no algorithm that finds a vector satisfying

$$0 \ne \|\boldsymbol{v}\| \le \mathrm{Poly}(n)\lambda(L)$$

faster than $O(1)^n$. This suggests using SVP and CVP as the basis for cryptographic algorithms.

Lattice-Based Crypto
Early History

Lattice-Based Crypto

- Ajtai and Dwork (1995) described a lattice-based public key cryptosystem whose security relies on the difficulty of solving CVP in a certain set of lattices $\mathcal{L}_{\mathrm{AD}}$.
- They proved that breaking their system for a a randomly chosen lattice of dimension m in $\mathcal{L}_{\mathrm{AD}}$ is as difficult as solving SVP for all lattices of dimension n, where n depends on m.
- This *average case-worst case equivalence* is a theoretical cryptographic milestone, but unfortunately the Ajtai-Dwork cryptosystem is quite impractical.
- More practical lattice-based cryptosystem were proposed in 1996 by Goldreich, Goldwasser, and Halevi (GGH, inspired by AD), and independently by Hoffstein, Pipher, and Silverman (NTRU).

Why Use Lattices for Crypto?

- A primary initial motivation was efficiency. Lattice-based systems can be 10 to 100 times faster than RSA or ECC systems at equivalent security levels.
- Of course, all of these systems have gotten faster over the years due to implementation "tricks".
- And as CPU speeds increased and memory costs decreased, speed differences became less relevant on many (but not all) devices.
- Recently, there has been renewed interest in lattice systems because, at present, there are no quantum algorithms that solve general cases of SVP or CVP in polynomial (or even subexponential) time.
- And this is not through lack of trying. Shor's original article specifically mentions SVP as an interesting problem for quantum algorithm analysis.

Good Bases, Bad Bases, and CVP

Solving CVP Using a Good Basis

It actually easy to solve (appr)CVP if one has a "good" basis $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ for L, where a basis is **good** if the vectors are pairwise "reasonably orthogonal."

To find a $\boldsymbol{v} \in L$ that is close to $\boldsymbol{w}$, first use linear algebra to write

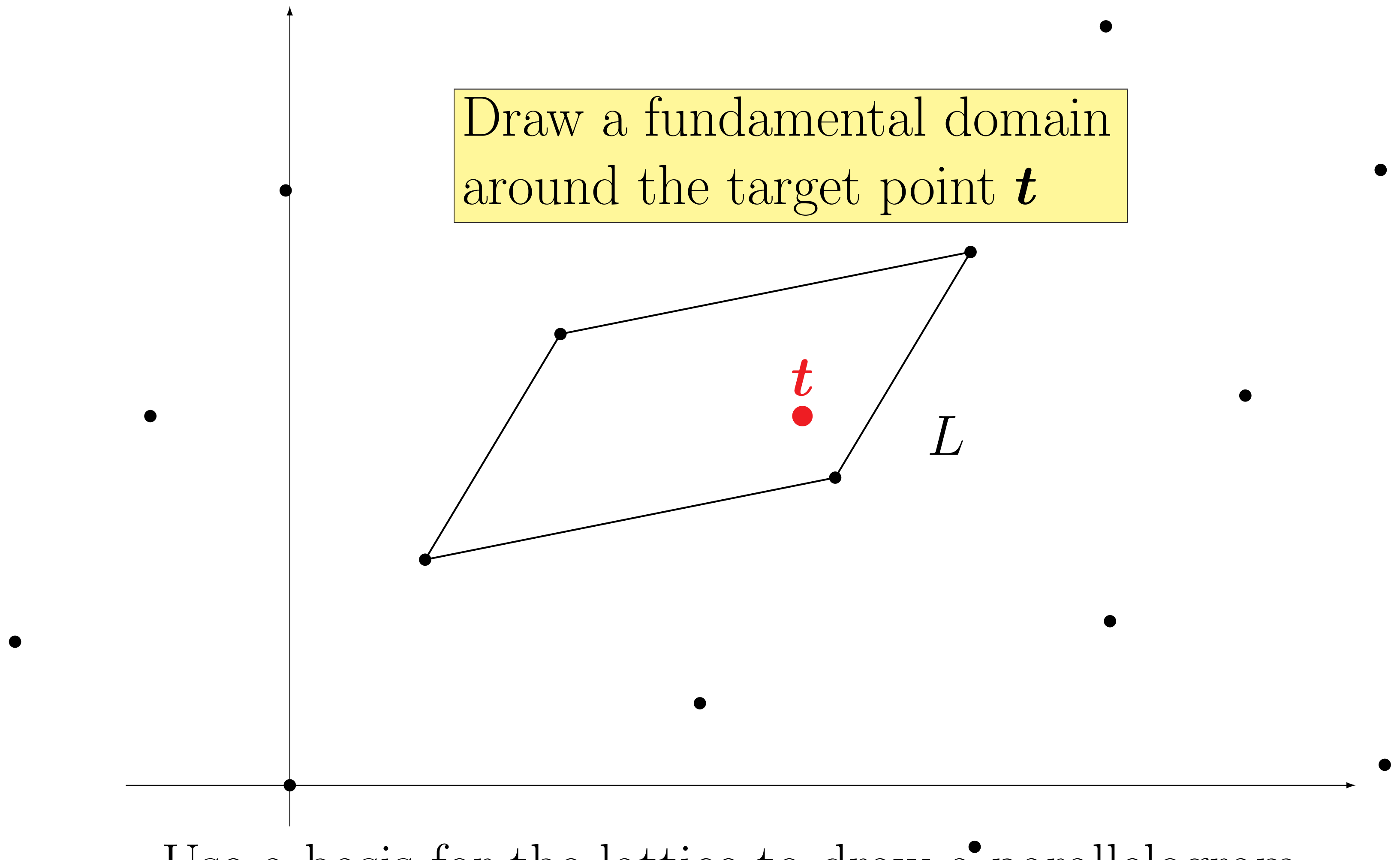
$$\boldsymbol{w} = \alpha_1 \boldsymbol{v}_1 + \cdots + \alpha_n \boldsymbol{v}_n \quad \text{with } \alpha_i \in \mathbb{R},$$

and then round the α_i to get a lattice vector

$$\boldsymbol{v} = \lfloor \alpha_1 \rceil \boldsymbol{v}_1 + \cdots + \lfloor \alpha_n \rceil \boldsymbol{v}_n \in L$$

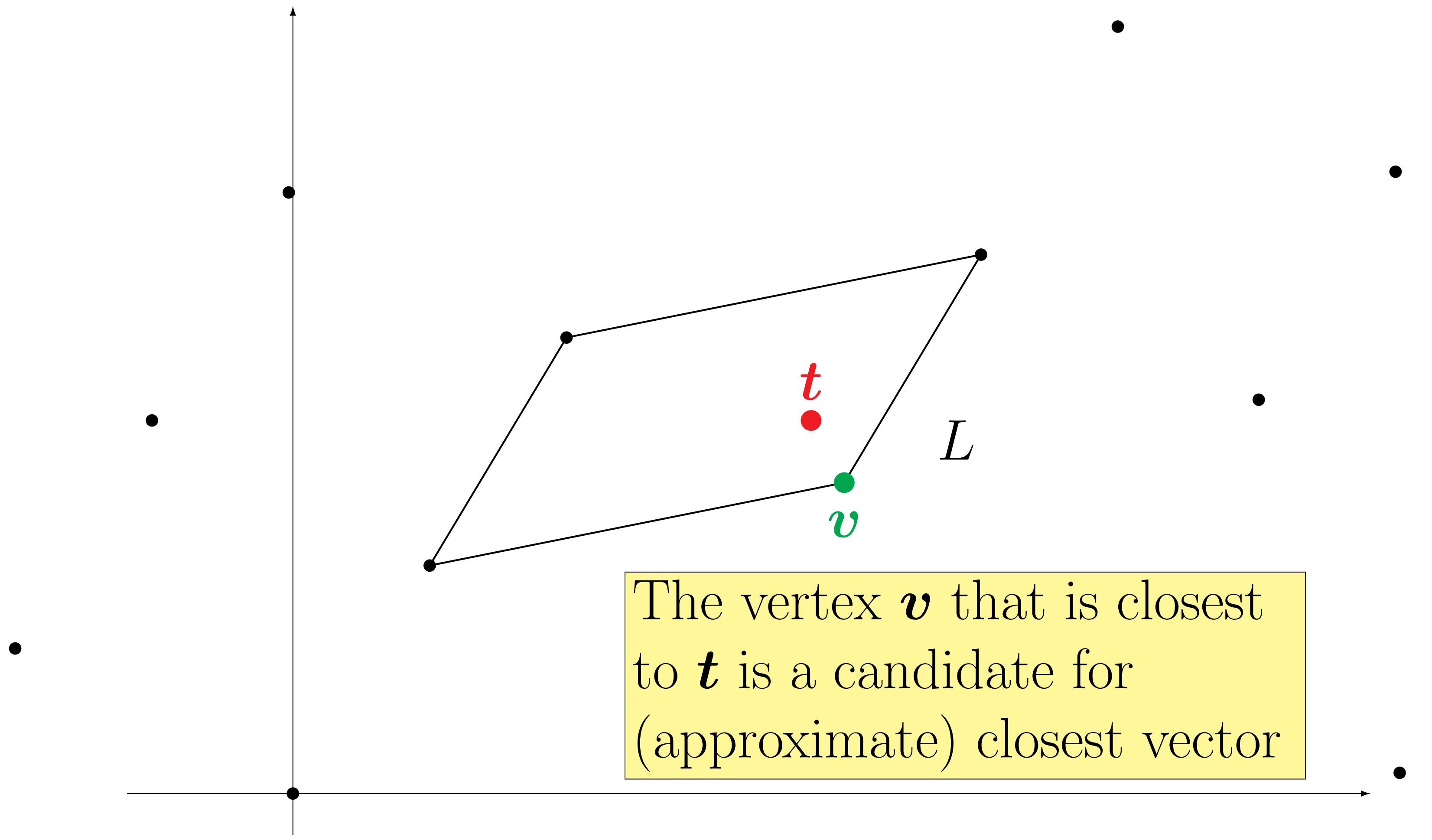
that is "close" to $\boldsymbol{w}$.

Using a Basis to Try to Solve the Closest Vector Problem


Draw a fundamental domain around the target point $\boldsymbol{t}$
$\boldsymbol{t}$
L

Use a basis for the lattice to draw a parallelogram around the target point.

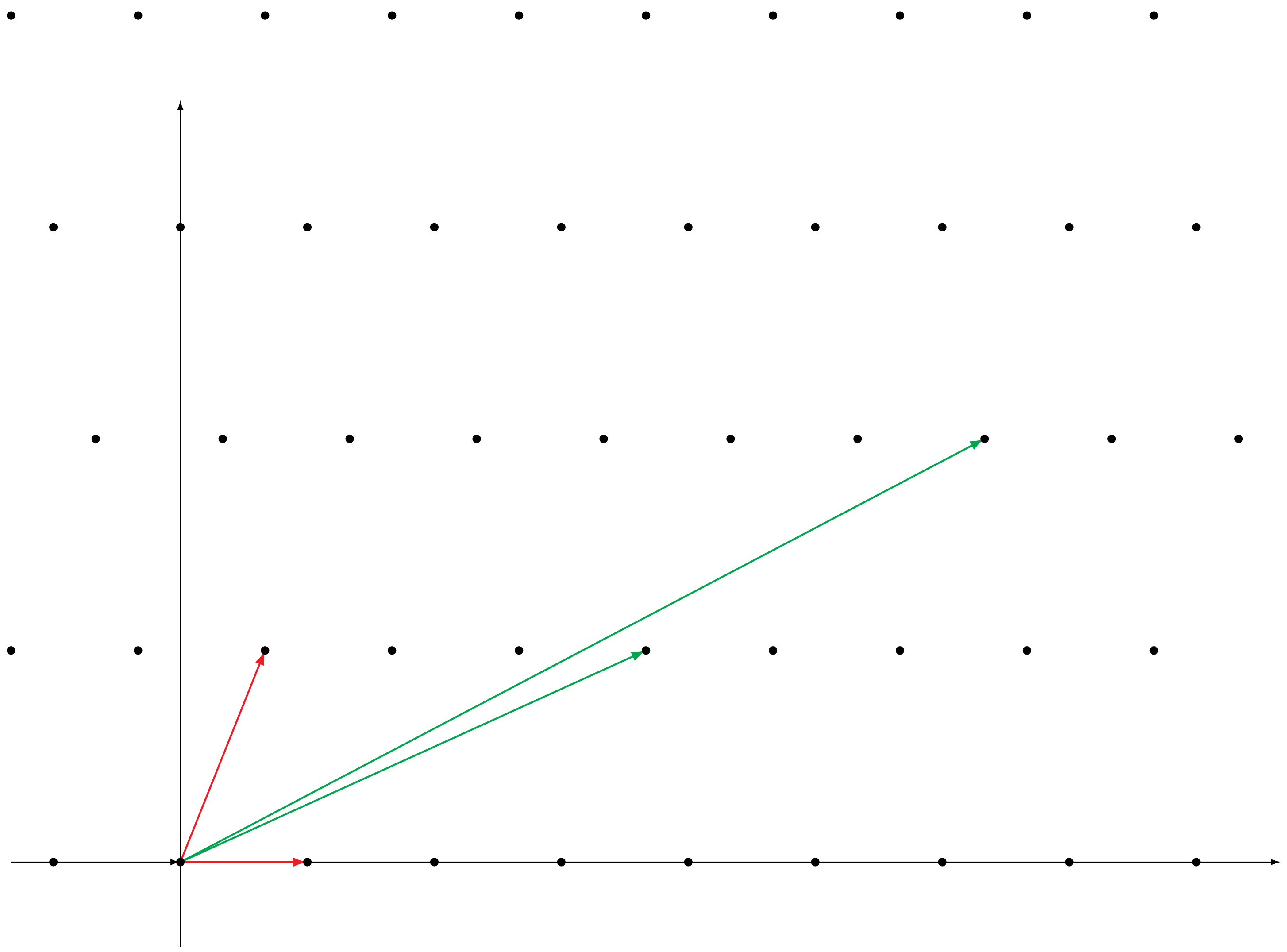
Using a Basis to Try to Solve the Closest Vector Problem



The vertex $\boldsymbol{v}$ that is closest to $\boldsymbol{t}$ is a candidate for (approximate) closest vector

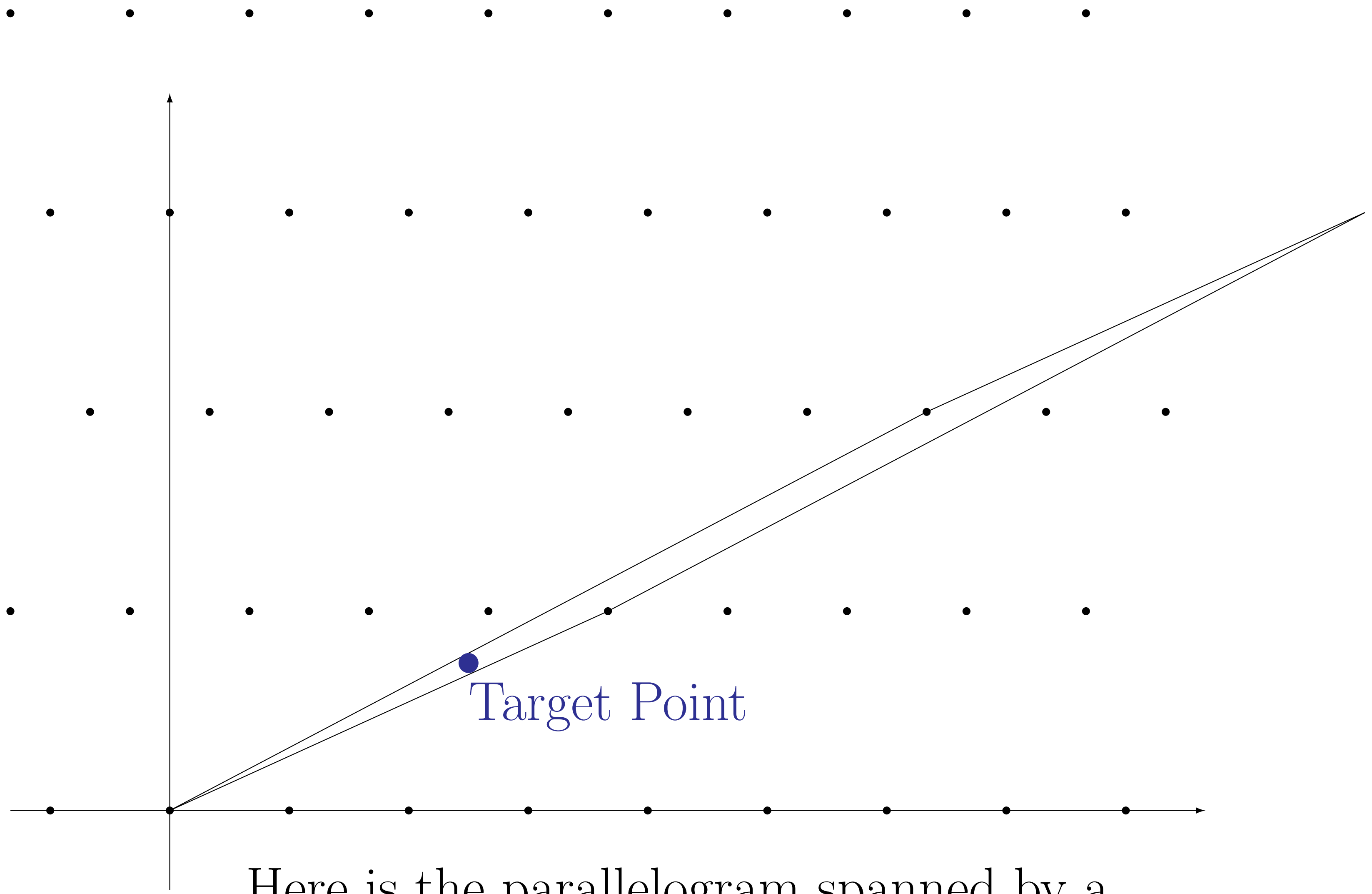
The vertex $\boldsymbol{v}$ of the fundamental domain that is closest to $\boldsymbol{t}$ will be a close lattice point if the basis is “good”, meaning if the basis consists of short vectors that are reasonably orthogonal to one another.

Good and Bad Bases



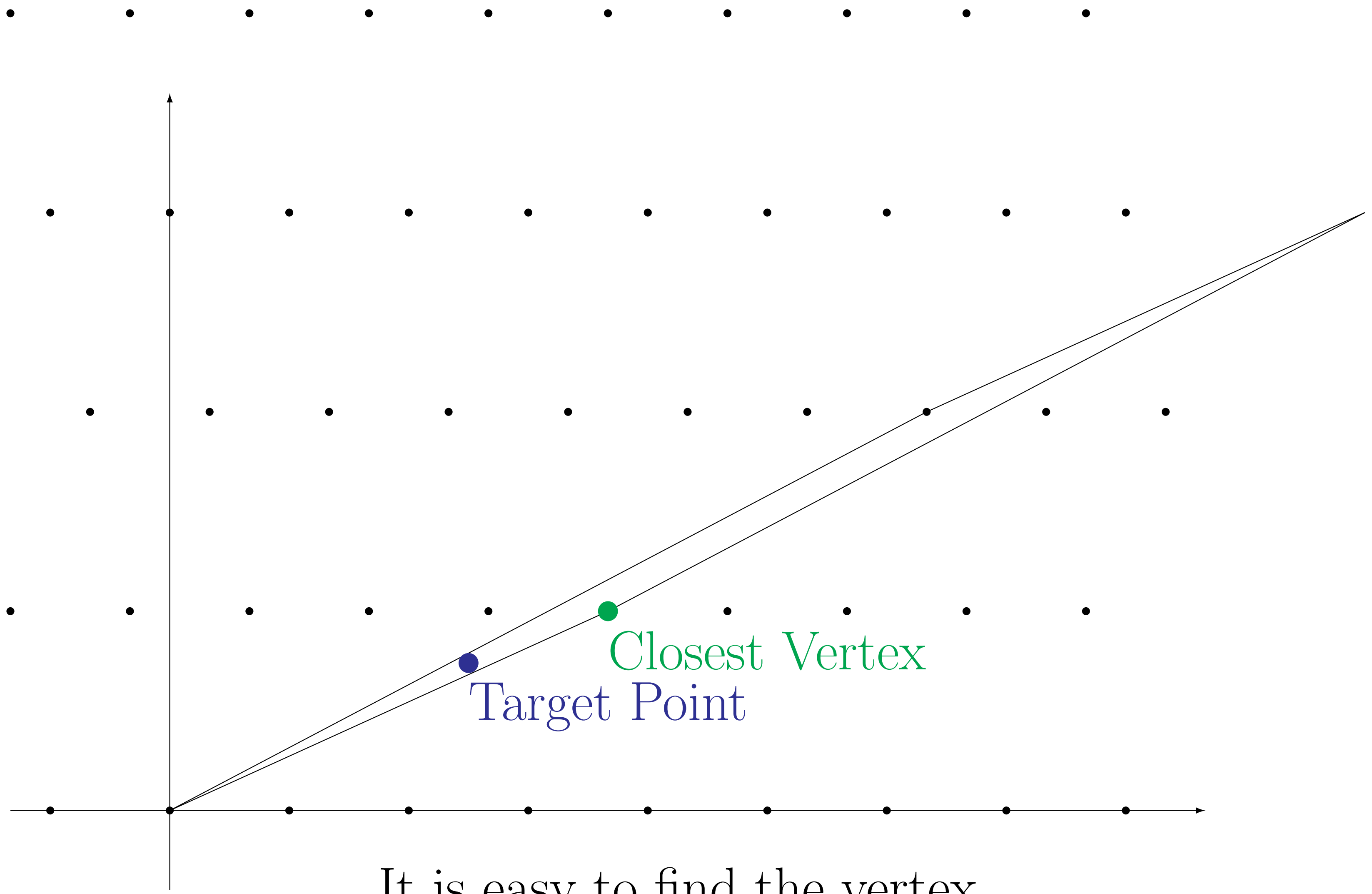
A "good" basis and a "bad" basis

Closest Vertex Method Using Bad Basis



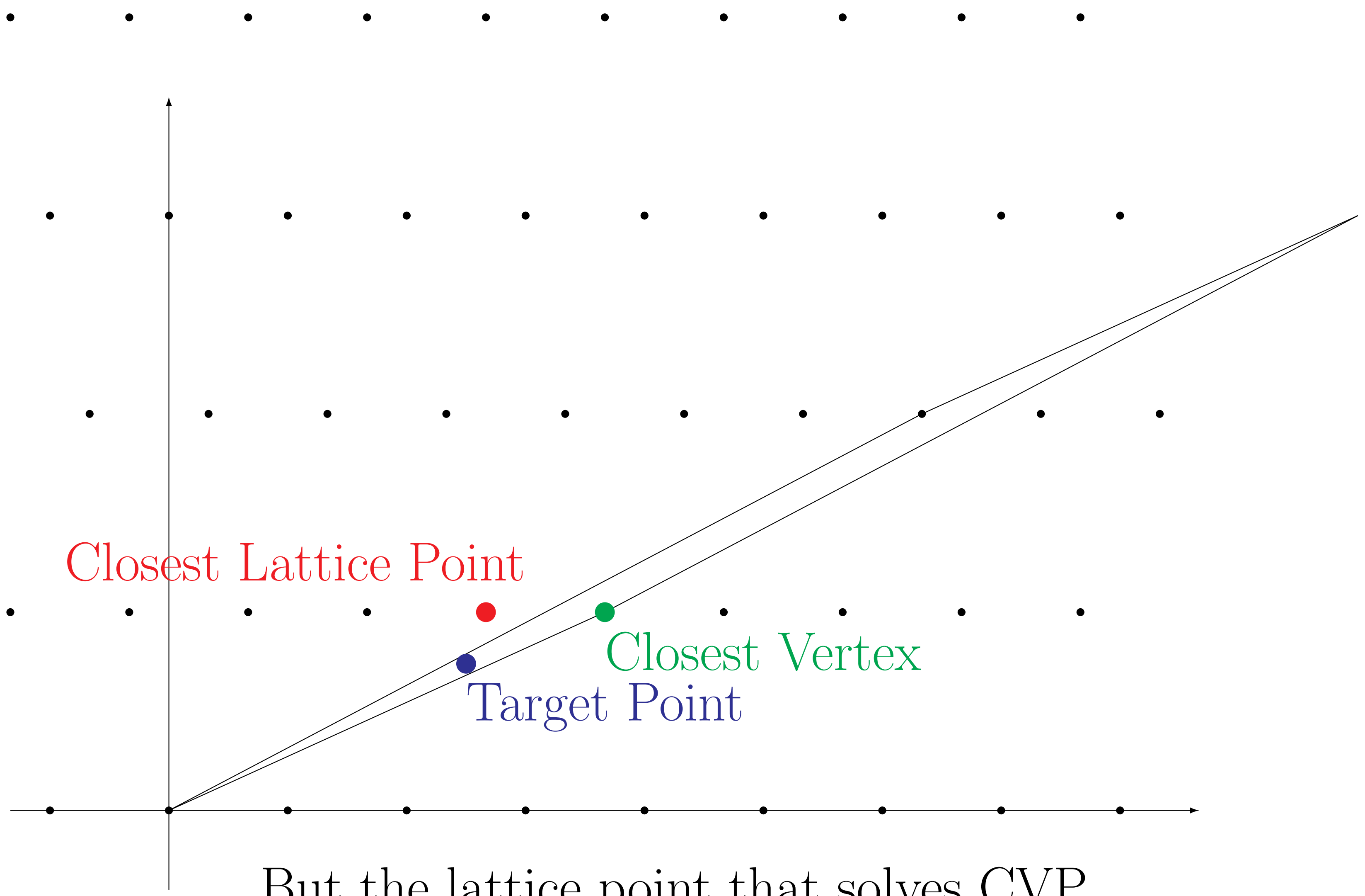
Here is the parallelogram spanned by a "bad" basis and a CVP target point.

Closest Vertex Method Using Bad Basis



It is easy to find the vertex
that is closest to the target point.

Closest Vertex Method Using Bad Basis



But the lattice point that solves CVP
is much closer to the target.

The GGH Cryptosystem — An Outline

The private key is a "good basis"

$$\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$$

for L, and the public key is a "bad basis"

$$\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n\}.$$

To encrypt a plaintext $\boldsymbol{m}$ (a small vector), form

$$\boldsymbol{e} = r_1\boldsymbol{w}_1 + \cdots + r_n\boldsymbol{w}_n + \boldsymbol{m} \quad \text{for random } r_i\text{'s.}$$

To decrypt, express $\boldsymbol{e}$ in terms of the good basis

$$\boldsymbol{e} = \alpha_1\boldsymbol{v}_1 + \cdots + \alpha_n\boldsymbol{v}_n \quad \text{with } \alpha_i \in \mathbb{R},$$

and then round the α_i's to recover

$$\boldsymbol{m} = \boldsymbol{e} - \lfloor\alpha_1\rceil\boldsymbol{v}_1 - \cdots - \lfloor\alpha_n\rceil\boldsymbol{v}_n.$$

GGH versus LLL

The LLL algorithm takes a “bad” basis $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n\}$ and outputs a basis $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ that is “moderately good.”

If n is not too large, say $n < 100$, then LLL can be used to find a basis that will decrypt GGH.

On the other hand, if $n > 400$, then the GGH public key, which consists of n vectors in $\mathbb{Z}^n$ with (say) 6-digit entries, is around 400KB. So practicality is an issue.

The problem is that key size is $O(n^2)$, and LLL is quite effective for $n < 100$ and usable for $n < 300$.

RSA analogy: Factorization of 256 bit products pq is easy, while factorization of 2560 bit products pq is infeasible. But this is okay, because RSA keys are linear in bit-size, not quadratic.

NTRUEncrypt

NTRUEncrypt

NTRUEncrypt is a lattice-based public key cryptosystem invented by Jeff Hoffstein around 1995 and further developed by Jeff, Jill Pipher, and me over the next few years. It was the first practical lattice-based system, where

Practical = Secure + Fast + Small Key Size.

The basic algebraic operation used by NTRU may be described in two equivalent ways:

- Polynomial multiplication in the quotient ring $\frac{\mathbb{Z}[X]}{(X^N-1)}$.
- Convolution product in the group $\mathbb{Z}^N$.

We identify $f(X) = a_1 0 + \cdots + a_{N-1}X^{N-1}$ with its vector of coefficients $\boldsymbol{a} = (a_0, \ldots, a_{N-1})$. We denote the product by $\star$. In terms of convolutions,

$$\boldsymbol{c} = \boldsymbol{a} \star \boldsymbol{b} \quad \text{with} \quad c_k = \sum_{i+j \equiv k \pmod{N}} a_i b_j.$$

NTRUEncrypt — How It Works

Here is a version of NTRUEncrypt (fitting on one slide).

Public Parameters	N	a prime $(250 < N < 2500)$
	q	large modulus $(250 < q < 2500)$
	p	small modulus (say $p = 3$, $p \nmid q$)
Private Key	$\boldsymbol{F}, \boldsymbol{G}$	random $\in \{-1, 0, 1\}^N$
	$\boldsymbol{f}, \boldsymbol{g}$	set $\boldsymbol{f} = 1 + p\boldsymbol{F}$ and $\boldsymbol{g} = p\boldsymbol{G}$
Public Key	$\boldsymbol{h}$	$\equiv \boldsymbol{f}^{-1} \star \boldsymbol{g} \pmod{q}$
Encryption	$\boldsymbol{m}$	plaintext $\in \{-1, 0, 1\}^N$
	$\boldsymbol{r}$	random $\in \{-1, 0, 1\}^N$
	$\boldsymbol{e}$	$\equiv \boldsymbol{r} \star \boldsymbol{h} + \boldsymbol{m} \pmod{q}$, ciphertext
Decryption	$\boldsymbol{a}$	$\equiv \boldsymbol{f} \star \boldsymbol{e} \pmod{q}$
	Lift $\boldsymbol{a}$ to $\mathbb{Z}^N$ with coefficients $\lvert a_i \rvert \le \frac{1}{2}q$	
	$\boldsymbol{a}$	$\pmod{p}$ is equal to $\boldsymbol{m}$.

NTRUEncrypt — Why It Works

First we compute

$$\begin{aligned} \boldsymbol{a} &\equiv \boldsymbol{f} \star \boldsymbol{e} \pmod{q} \\ &\equiv \boldsymbol{f} \star (\boldsymbol{r} \star \boldsymbol{h} + \boldsymbol{m}) \pmod{q} \\ &\equiv \boldsymbol{f} \star (\boldsymbol{r} \star \boldsymbol{f}^{-1} \star \boldsymbol{g} + \boldsymbol{m} \pmod{q} \\ &\equiv \boldsymbol{r} \star \boldsymbol{g} + \boldsymbol{f} \star \boldsymbol{m} \pmod{q}. \end{aligned}$$

Since $\boldsymbol{r}, \boldsymbol{g}, \boldsymbol{f}, \boldsymbol{m}$ have small coefficients, when we lift $\boldsymbol{a}$, we get an *exact* equality

$$\boldsymbol{a} = \boldsymbol{r} \star \boldsymbol{g} + \boldsymbol{f} \star \boldsymbol{m} \quad \text{in } \mathbb{Z}^N.$$

Then reducing modulo p gives

$$\begin{aligned} \boldsymbol{a} &\equiv \boldsymbol{r} \star \boldsymbol{g} + \boldsymbol{f} \star \boldsymbol{m} \pmod{p} \\ &\equiv \boldsymbol{r} \star (p\boldsymbol{G}) + (1 + p\boldsymbol{F}) \star \boldsymbol{m} \pmod{p} \\ &\equiv \boldsymbol{m} \pmod{p}. \end{aligned}$$

NTRU as a Lattice-Based Cryptosystem

The **Convolution Modular Lattice** $L_{\boldsymbol{h}}$ associated to the vector $\boldsymbol{h}$ and modulus q is the $2N$ dimensional lattice with basis given by the rows of the matrix:

$$L_{\boldsymbol{h}} = \text{RowSpan}\left(\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{N-1} \\ 0 & 1 & \cdots & 0 & h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & h_1 & h_2 & \cdots & h_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array}\right)$$

Another way to describe $L_{\boldsymbol{h}}$ is the set of vectors

$$L_{\boldsymbol{h}} = \left\{(\boldsymbol{a},\boldsymbol{b}) \in \mathbb{Z}^{2N} : \boldsymbol{a} \star \boldsymbol{h} \equiv \boldsymbol{b} \pmod{q}\right\}.$$

Small Vectors in NTRU Lattices

NTRU public/private key pairs are constructed via

$$\boldsymbol{f} \star \boldsymbol{h} \equiv \boldsymbol{g} \pmod{q} \quad \text{with “small” } f \text{ and } g.$$

This convolution relation implies that the NTRU lattice $L_{\boldsymbol{h}}$ contains the short vector

$$[\boldsymbol{f}, \boldsymbol{g}] = [f_0, f_1, \ldots, f_{N-1}, g_0, g_1, \ldots, g_{N-1}].$$

To see that $[\boldsymbol{f}, \boldsymbol{g}]$ is in $L_{\boldsymbol{h}}$, write

$$\boldsymbol{f} \star \boldsymbol{h} - \boldsymbol{g} = -q\boldsymbol{u} \quad \text{with} \quad \boldsymbol{u} \in \mathbb{Z}^N, \quad \text{and then}$$

$$[\boldsymbol{f}, \boldsymbol{g}] = [\boldsymbol{f}, \boldsymbol{u}] \begin{pmatrix} 1 & \cdots & 0 & h_0 & \cdots & h_{N-1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & h_1 & \cdots & h_0 \\ 0 & \cdots & 0 & q & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & q \end{pmatrix} \in L_{\boldsymbol{h}}.$$

- Can also search for $[\boldsymbol{F}, \boldsymbol{G}]$ via a CVP.

NTRU Decryption as a CVP Problem

Recall that the ciphertext $\boldsymbol{e}$ has the form

$$\boldsymbol{e} = \boldsymbol{r} \star \boldsymbol{h} + \boldsymbol{m} \pmod{q}.$$

We can rewrite this relation in vector form as

$$\begin{aligned}[0, \boldsymbol{e}] &= [0, \boldsymbol{r} \star \boldsymbol{h} + \boldsymbol{m} \pmod q] \\ &\equiv [\boldsymbol{r}, \boldsymbol{r} \star \boldsymbol{h} \pmod q] + [-\boldsymbol{r}, \boldsymbol{m}].\end{aligned}$$

The vector $[\boldsymbol{r}, \boldsymbol{r} \star \boldsymbol{h} \pmod{q}]$ is in the lattice $L_{\boldsymbol{h}}$, while, the vector $[-\boldsymbol{r}, \boldsymbol{m}]$ is quite short.

Conclusion. For appropriate parameters, recovery of the private key $\boldsymbol{f}$ from the public key $\boldsymbol{h}$ is equivalent to finding a shortest vector in $L_{\boldsymbol{h}}$, and recovery of the plaintext $\boldsymbol{m}$ from $\boldsymbol{h}$ and the ciphertext $\boldsymbol{e}$ is equivalent to finding the vector in $L_{\boldsymbol{h}}$ that is closest to the vector $[0, \boldsymbol{e}]$.

Lattice-Based Digital Signatures

Digital Signatures

A digital signature scheme consists of:

- A set of (hashes of) digital documents $\mathcal{D}$.
- A set of signatures $\mathcal{S}$.
- A set of randomization elements $\mathcal{R}$.
- A set $\mathcal{K}$ of pairs $(K_{\text{sign}}, K_{\text{verify}})$ consisting of linked signing and verification keys.

A signing key is a map

$$K_{\text{sign}} : \mathcal{D} \times \mathcal{R} \to \mathcal{S},$$

and a verification key is a map

$$K_{\text{verify}} : \mathcal{D} \times \mathcal{S} \to \{\text{Yes}, \text{No}\}.$$

Sign and verify keys satisfy

$$K_{\text{verify}}(d, s) = \text{Yes} \quad \Longleftrightarrow \quad s = K_{\text{sign}}(d, r) \quad \text{for some } r \in \mathcal{R}.$$

Digital Signatures Based on Lattice Problems

It is easy to create a CVP-based digital signature scheme using good and bad bases.

A GGH Digital Signature Scheme

- **Key Creation**:

$$\begin{aligned}\text{Private Key} &= \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_n\} = \text{a good basis}\\ \text{Public Key} &= \{\boldsymbol{w}_1, \dots, \boldsymbol{w}_n\} = \text{a bad basis}\end{aligned}$$

- **Signing**: To sign $\boldsymbol{d} \in \mathbb{R}^n$, use the good basis and rounding to find an

$$\boldsymbol{s} = a_1\boldsymbol{v}_1 + \cdots + a_n\boldsymbol{v}_n \in L$$

that is close to $\boldsymbol{d}$. Publish the signature

$$\boldsymbol{s} = b_1\boldsymbol{w}_1 + \cdots + b_n\boldsymbol{w}_n$$

expressed in terms of the bad basis.

- **Verification**: Reconstruct $\boldsymbol{s}$ from the bad basis and the b_i's and check that it is close to $\boldsymbol{d}$.

Adapting NTRU for Digital Signatures

GGH signatures are unwieldy because keys are at least $O(n^2)$ bits and LLL forces (say) $n > 300$.

NTRU lattices are specified by only $O(N \log N)$ bits, but how do we find a good basis? The NTRU lattice $L_{\boldsymbol{h}}$ contains N independent short vector by rotating $(\boldsymbol{f}, \boldsymbol{g})$,

$$(\boldsymbol{e}_i \star \boldsymbol{f}, \boldsymbol{e}_i \star \boldsymbol{g}) \in L_{\boldsymbol{h}} \quad \text{for } 0 \le i < N.$$

But $L_{\boldsymbol{h}}$ has dimension $2N$.

So we expand the list of N very short vectors and include N additional moderately short vectors to form a full basis. More precisely, we find one moderately short vector $(\boldsymbol{f}', \boldsymbol{g}')$ and use its N rotations to fill out the basis. This can be done and leads to a reasonably practical digital signature scheme.

However, these GGH and NTRU schemes both have a potential weakness!

Lattices Signature Schemes and Transcript Attacks

Digital signature schemes differ from public key cryptosystems in that each document/signature pair $(\boldsymbol{d}, \boldsymbol{s})$ potentially reveals information about the private key. A **Transcript Attack** is a method for recovering the private key from a long list (transcipt) of signatures:

$$(\boldsymbol{d}_1, \boldsymbol{s}_1), (\boldsymbol{d}_2, \boldsymbol{s}_2), \ldots, (\boldsymbol{d}_t, \boldsymbol{s}_t).$$

Each GGH or NTRU signature reveals a lattice vector of the form $\boldsymbol{s} = a_1\boldsymbol{v}_1 + \cdots + a_n\boldsymbol{v}_n$

The attacker does not know the a_i or the $\boldsymbol{v}_i$, but taking an appropriate weighted average over a transcript, he can build up a picture of the fundamental domain

$$\{t_1\boldsymbol{v}_1 + \cdots + t_n\boldsymbol{v}_n : 0 \le t_i < 1\}.$$

(This is a simplification, but conveys the underlying idea.) Using this picture, he can then forge signatures.

Naive NTRU Signatures and Transcript Attacks

Various sorts of transcript attacks were developed, both for general lattices and specifically for NTRU lattices, by a number of people including Gentry, Nguyen, Regev and Szydlo.

In particular, an early proposal for an NTRU-like signature scheme was destroyed by Gentry and Szydlo by averaging over a transcript to recover the product $\boldsymbol{f} \star \tilde{\boldsymbol{f}}$, and more recently (2006) Nguyen and Regev devised a very clever and very efficient algorithm for recovering the secret key parallelopiped from a small number of signatures.

As my colleague Jeff Hoffstein so aptly describes it:

A Signature Scheme Disaster

"Luckily the crypto community was pretty forgiving about this mishap."

A Signature Scheme Disaster

“Luckily the crypto community was pretty forgiving about this mishap.”

Rejection Sampling and Transcript Security

Various ad hoc perturbation methods were proposed to make it harder for the attacker to build up a picture of the good fundamental domain, but it was hard to analyze how effective they were.

Lyubashevsky recently described how to use **rejection sampling** to completely(!) eliminate transcript attacks on certain lattice-based digital signature schemes.

- First one includes some randomness in each signature.
- Next one rejects “bad” signatures and only uses “good” signatures.
- If done properly, the probability distribution of the set of good signatures is the same for all private keys. Hence a transcript of signatures contains no information about the private key!

NTRUSign

It is not immediately clear how to adapt rejection sampling to GGH or NTRUSign. In a recent preprint, Hoffstein et al. have proposed a "two-prime" version of NTRUSign that simultaneously:

- Avoids the problem of having only half a short basis.
- Allows transcript security via rejection sampling.

In the next few slides, I will describe how **NTRUSign** works and how rejection sampling achieves transcript security. First one piece of notation:

$$\|\boldsymbol{a}\|_\infty = \big\|(a_1,\ldots,a_n)\big\|_\infty = \max|a_i|.$$

Also, a vector "$\boldsymbol{a} \bmod q$" has coefficients $|a_i| \le \frac{1}{2}q$.

NTRUSign and Rejection Sampling

Public Parameters: Dimension parameter N, odd primes p and q, and a norm bound $B = \lceil p^2 N/4 \rceil$.

Signing Key: A pair of vectors $(\boldsymbol{f}, \boldsymbol{g})$, where $\boldsymbol{f} = p\boldsymbol{F}$ with $\boldsymbol{F}$ random mod 3, and $\boldsymbol{g}$ random mod p.

Verification Key: $\boldsymbol{h} = \boldsymbol{f}^{-1} \star \boldsymbol{g} \pmod{q}$

Digital Documents: A document (hash) is a pair of mod p vectors $(\boldsymbol{s}_p, \boldsymbol{t}_p)$.

Valid Signatures: A signature on $(\boldsymbol{s}_p, \boldsymbol{t}_p)$ for the signing key $\boldsymbol{h}$ is a pair of vectors $(\boldsymbol{s}, \boldsymbol{t})$ satisfying:

- $\boldsymbol{t} \equiv \boldsymbol{s} \star \boldsymbol{h} \pmod{q}$, i.e., $(\boldsymbol{s}, \boldsymbol{t}) \in L_{\boldsymbol{h}}$.
- $(\boldsymbol{s}, \boldsymbol{t}) \equiv (\boldsymbol{s}_p, \boldsymbol{t}_p) \pmod{p}$.
- $\|\boldsymbol{s}\|_\infty$ and $\|\boldsymbol{t}\|_\infty$ are both $\le \frac{1}{2}q - B$.

NTRUSign — Signing Algorithm

This algorithm computes the NTRUSign signature on a document $(\boldsymbol{s}_p, \boldsymbol{t}_p)$ using the signing key $(\boldsymbol{f}, \boldsymbol{g})$.

(1) Choose a random $\boldsymbol{r}$ with $\|\boldsymbol{r}\|_\infty \le \left\lfloor \frac{q}{2p} - \frac{1}{2} \right\rfloor$.

(2) Set $\boldsymbol{s}_0 = \boldsymbol{s}_p + p\boldsymbol{r}$.

(3) Set $\boldsymbol{t}_0 = \boldsymbol{h} \star \boldsymbol{s}_0 \pmod{q}$.

(4) Compute $\boldsymbol{a} = \boldsymbol{g}^{-1} \star (\boldsymbol{t}_p - \boldsymbol{t}_0) \pmod{p}$.

(5) Set $\boldsymbol{s} = \boldsymbol{s}_0 + \boldsymbol{a} \star \boldsymbol{f}$ and $\boldsymbol{t} = \boldsymbol{t}_0 + \boldsymbol{a} \star \boldsymbol{g}$.

(6) If $\|\boldsymbol{s}\|_\infty$ or $\|\boldsymbol{t}\|_\infty$ is $> \frac{1}{2}q - B$, then REJECT. Go to Step (1).

(7) Return the signature $(\boldsymbol{s}, \boldsymbol{t})$.

It is easy to check that the $(\boldsymbol{s}, \boldsymbol{t})$ returned by the algorithm has the three properties needed to be a valid signature for the document $(\boldsymbol{s}_p, \boldsymbol{t}_p)$.

Transcript Security of NTRUSign

NTRUSign is secure against transcript attacks due to:

Theorem. Fix a private key $(\boldsymbol{f}, \boldsymbol{g})$ and a document $(\boldsymbol{s}_p, \boldsymbol{t}_p)$ to be signed. Then among vectors $(\boldsymbol{s}, \boldsymbol{t})$ with

$$\|\boldsymbol{s}\| \le \frac{1}{2}q - B \quad \text{and} \quad \|\boldsymbol{t}\| \le \frac{1}{2}q - B$$
$$\text{and} \quad (\boldsymbol{s}, \boldsymbol{t}) \equiv (\boldsymbol{s}_p, \boldsymbol{t}_p) \pmod{p},$$

the probability that $(\boldsymbol{s}, \boldsymbol{t})$ is chosen to be the signature on $(\boldsymbol{s}_p, \boldsymbol{t}_p)$ is

$$\text{Prob}\begin{pmatrix}\text{signature} \\ \text{is } (\boldsymbol{s}, \boldsymbol{t})\end{pmatrix} = \left(\frac{p}{2\lfloor q/2p - 1/2 \rfloor}\right)^N.$$

Conclusion: The probability does not depend on the private key $(\boldsymbol{f}, \boldsymbol{g})$. Hence a transcript contains no information about the key.

Probability of Accepting a Signature

In order for rejection sampling to be practical, there must be a reasonable probability that $(\boldsymbol{s}, \boldsymbol{t})$ will be accepted.

The coefficients of $\boldsymbol{s}$ and $\boldsymbol{t}$ satisfy

$$\left\|(\boldsymbol{s}, \boldsymbol{t})\right\|_\infty \le \frac{q}{2} + \frac{p^2 N}{4} \approx \frac{q}{2} + B.$$

We fix $2 \le k \le 50$ and take $q \approx kp^2N^2/4 \approx kNB$. Then with the slightly simplifying assumption that the coefficients are uniformly distributed, we find that

$$\begin{aligned}\mathrm{Prob}\begin{pmatrix}(\boldsymbol{s}, \boldsymbol{t}) \text{ is} \\ \text{accepted}\end{pmatrix} &\approx \left(\frac{q/2 - B}{q/2 + B}\right)^{2N} \\ &\approx \left(\frac{1 - 2/kN}{1 + 2/kN}\right)^{2N} \\ &\approx e^{-8/k}.\end{aligned}$$

The Lattice Problem Underlying NTRUSign

In order to forge a signature, the forger must find a vector $(\boldsymbol{s}, \boldsymbol{t})$ satisfying three conditions:

- **Lattice Condition**:

$$(\boldsymbol{s}, \boldsymbol{t}) \in L_{\boldsymbol{h}}, \quad \text{i.e.} \quad \boldsymbol{t} \equiv \boldsymbol{h} \star \boldsymbol{s} \pmod{q}.$$

- **Congruence Condition**:

$$(\boldsymbol{s}, \boldsymbol{t}) \equiv (\boldsymbol{s}_p, \boldsymbol{t}_p) \pmod{p}.$$

- **Norm Condition**:

$$\left\|(\boldsymbol{s}, \boldsymbol{t})\right\|_\infty \le \frac{q}{2} - B.$$

The congruence condition says that the difference

$$(\boldsymbol{s}, \boldsymbol{t}) - (\boldsymbol{s}_p, \boldsymbol{t}_p) \quad \text{is in the lattice } p\mathbb{Z}^{2N}.$$

The Lattice Problem Underlying NTRUSign (continued)

Thus the forger is looking for a short vector in the intersection

$$L_{\boldsymbol{h}} \cap \left(p\mathbb{Z}^{2N} + (\boldsymbol{s}_p, \boldsymbol{t}_p)\right).$$

Using the fact that

$$\mathrm{Disc}(L_{\boldsymbol{h}}) = q^N \quad \text{and} \qquad \mathrm{Disc}(p\mathbb{Z}^{2N}) = p^{2N}$$
$$\text{with} \quad \gcd(p,q) = 1,$$

one can reduce the forgery problem to solving apprCVP in the intersection lattice

$$L_{\boldsymbol{h},p} := L_{\boldsymbol{h}} \cap p\mathbb{Z}^{2N} \quad \text{having} \quad \mathrm{Disc}(L_{\boldsymbol{h},p}) = (p^2q)^N.$$

The difficulty of this problem may then be analyzed in the usual way via BKZ-LLL lattice reduction experiments.

NTRUSign Parameters

Testing is ongoing, but the following should be practical, while providing good security;

$$\begin{aligned} N &= 661 \\ p &= 3 \\ q &= 9829081 \\ B &= 1487 \\ k &= 10 \end{aligned}$$

With these parameters, we have

$$\begin{aligned} \text{Prob(Signature is Accepted)} &\approx 45\% \\ \text{Key and Signature Size} &\approx 15864 \text{ bits} \\ \text{Bit Security} &\approx 192 \text{ to } 256 \end{aligned}$$

$\boxed{\text{\textbf{I want to thank the organizers for the invitation to speak and you for your attention.}}}$

NTRU and Lattice-Based Crypto: Past, Present, and Future

Joseph H. Silverman

Brown University

The Mathematics of Post-Quantum Cryptography
DIMACS Center, Rutgers University

January 12–16, 2015

Addendum: Proof of Transcript Security for NTRUSign

The Preliminary Signing Function

We let

$$\mathcal{R}(k) = \{\boldsymbol{f} : \|\boldsymbol{f}\|_\infty \le k\} \quad \text{and} \quad A = \left\lfloor \frac{q}{2p} - \frac{1}{2} \right\rfloor .$$

If we ignore rejection sampling, signing is a function

$$\sigma'(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \boldsymbol{r}) = (\boldsymbol{s}, \boldsymbol{t})$$

with

$$\begin{array}{rl} (\boldsymbol{f}, \boldsymbol{g}) \in p\mathcal{R}(1) \times \mathcal{R}(p/2) & \text{private key,} \\ (\boldsymbol{s}_p, \boldsymbol{t}_p) \in \mathcal{R}(p/2) \times \mathcal{R}(p/2) & \text{document,} \\ \boldsymbol{r} \in \mathcal{R}(A) & \text{random element.} \end{array}$$

The domain of σ' is the set

$$\Omega' = p\mathcal{R}(1) \times \mathcal{R}\left(\frac{p}{2}\right) \times \mathcal{R}\left(\frac{p}{2}\right) \times \mathcal{R}\left(\frac{p}{2}\right) \times \mathcal{R}(A).$$

The Signing Function with Rejection Sampling

The preliminary signing function is given explicitly by

$$\sigma'(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \boldsymbol{r}) = (\boldsymbol{s}_0 + \boldsymbol{a} \star \boldsymbol{f}, \boldsymbol{t}_0 + \boldsymbol{a} \star \boldsymbol{g}),$$

where

$$\begin{aligned} &\boldsymbol{s}_0 = \boldsymbol{s}_p + p\boldsymbol{r}, \\ &\boldsymbol{t}_0 \equiv \boldsymbol{h} \star \boldsymbol{s}_0 \pmod{q} \quad \text{with } \|\boldsymbol{t}_0\| \le q/2, \\ &\boldsymbol{a} \equiv \boldsymbol{g}^{-1} \star (\boldsymbol{t}_p - \boldsymbol{t}_0) \pmod{p} \quad \text{with } \|\boldsymbol{a}\| \le p/2. \end{aligned}$$

We now introduce rejection sampling by defining

$$\Omega_B = \left\{ (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \boldsymbol{r}) \in \Omega' : \left\| \sigma'(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \boldsymbol{r}) \right\| \le \frac{q}{2} - B \right\}.$$

The restriction of $\boldsymbol{\sigma}'$ to Ω_B, denoted $\boldsymbol{\sigma}$, is a map

$$\sigma : \Omega_B \longrightarrow \mathcal{R}\left(\frac{q}{2} - B\right) \times \mathcal{R}\left(\frac{q}{2} - B\right).$$

Transcript Security Theorem. The rejection signature function σ has the following property: For a given

$$\begin{aligned} \text{private key} \quad & (\boldsymbol{f},\boldsymbol{g}) \in p\mathcal{R}(1) \times \mathcal{R}\left(\frac{p}{2}\right), \\ \text{document} \quad & (\boldsymbol{s}_p,\boldsymbol{t}_p) \in \mathcal{R}\left(\frac{p}{2}\right) \times \mathcal{R}\left(\frac{p}{2}\right), \\ \text{signature} \quad & (\boldsymbol{s},\boldsymbol{t}) \in \mathcal{R}\left(\frac{q}{2}-B\right) \times \mathcal{R}\left(\frac{q}{2}-B\right), \end{aligned}$$

the probability that $(\boldsymbol{s},\boldsymbol{t})$ is the signature on $(\boldsymbol{s}_p,\boldsymbol{t}_p)$ using the key $(\boldsymbol{f},\boldsymbol{g})$ is

$$\text{Prob}\left(\begin{array}{l|l} \text{signature} & \text{private key is } (\boldsymbol{f},\boldsymbol{g}) \text{ and} \\ \text{is } (\boldsymbol{s},\boldsymbol{t}) & \text{document hash is } (\boldsymbol{s}_p,\boldsymbol{t}_p) \end{array}\right)$$
$$= \begin{cases} (p/2A)^N & \text{if } (\boldsymbol{s},\boldsymbol{t}) \equiv (\boldsymbol{s}_p,\boldsymbol{t}_p) \pmod{p}, \\ 0 & \text{if } (\boldsymbol{s},\boldsymbol{t}) \not\equiv (\boldsymbol{s}_p,\boldsymbol{t}_p) \pmod{p}. \end{cases}$$

Proof of the Transcript Security Theorem

We may assume that

$$(\boldsymbol{s},\boldsymbol{t}) \equiv (\boldsymbol{s}_p,\boldsymbol{t}_p) \pmod{p}$$

since otherwise the probability is 0.

Since $\boldsymbol{r}$ is chosen uniformly from the set $\mathcal{R}(A)$, there are $(2A)^N$ possible choices for r. Hence the probability is $(2A)^{-N}$ times the number of elements in the set

$$\Sigma(\boldsymbol{f},\boldsymbol{g},\boldsymbol{s},\boldsymbol{t}) = \bigl\{\boldsymbol{r}\in\mathcal{R}(A) : \sigma(\boldsymbol{f},\boldsymbol{g},\boldsymbol{s}_p,\boldsymbol{t}_p,\boldsymbol{r}) = (\boldsymbol{s},\boldsymbol{t})\bigr\}.$$

Claim There is a well-defined bijection of sets

$$\begin{aligned}\phi : \mathcal{R}\left(\frac{p}{2}\right) &\longrightarrow \Sigma(\boldsymbol{f},\boldsymbol{g},\boldsymbol{s},\boldsymbol{t}),\\ \boldsymbol{b} &\longmapsto \frac{\boldsymbol{s}-\boldsymbol{s}_p}{p} - \boldsymbol{b}\star\frac{\boldsymbol{f}}{p}.\end{aligned}$$

Note that the coefficients of $\boldsymbol{s}-\boldsymbol{s}_p$ are multiples of p, and $\boldsymbol{f}\in p\mathcal{R}(1)$.

Proof of the Claim

To show that $\phi(\boldsymbol{b}) \in \Sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}, \boldsymbol{t})$, we check

$$\sigma\bigl(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}_p, \boldsymbol{t}_p, \phi(\boldsymbol{b})\bigr) = (\boldsymbol{s}, \boldsymbol{t}).$$

We first compute

$$\begin{aligned} \boldsymbol{s}_0 &= \boldsymbol{s}_p + p\phi(\boldsymbol{b}) \\ &= \boldsymbol{s}_p + p\left(\frac{\boldsymbol{s} - \boldsymbol{s}_p}{p} - \boldsymbol{b} \star \frac{\boldsymbol{f}}{p}\right) \\ &= \boldsymbol{s} - \boldsymbol{b} \star \boldsymbol{f}, \\ \boldsymbol{t}_0 &\equiv \boldsymbol{h} \star \boldsymbol{s}_0 \pmod q \\ &\equiv \boldsymbol{h} \star (\boldsymbol{s} - \boldsymbol{b} \star \boldsymbol{f}) \pmod q \\ &\equiv \boldsymbol{h} \star \boldsymbol{s} - \boldsymbol{b} \star \boldsymbol{g} \pmod q \quad \text{since } \boldsymbol{h} \equiv \boldsymbol{f}^{-1} \star \boldsymbol{g}, \\ &\equiv \boldsymbol{t} - \boldsymbol{b} \star \boldsymbol{g} \pmod q \quad \text{since } (\boldsymbol{s}, \boldsymbol{t}) \in L_{\boldsymbol{h}}. \end{aligned}$$

The formula for $\boldsymbol{s}_0$ is exact, but the formula for $\boldsymbol{t}_0$ is only a congruence (for now).

Proof of the Claim (continued)

Next we compute

$$\begin{aligned}
&\|\boldsymbol{t} - \boldsymbol{b} \star \boldsymbol{g}\| \\
&\quad \le \|\boldsymbol{t}\| + \|\boldsymbol{b} \star \boldsymbol{g}\| \quad \text{triangle inequality,} \\
&\quad \le \left(\frac{q}{2} - B\right) + B \quad \text{since } \boldsymbol{t} \in \mathcal{R}\left(\frac{q}{2} - B\right), \\
&\quad = \frac{q}{2}.
\end{aligned}$$

Since $\boldsymbol{t}_0$ is determined by the congruence $\boldsymbol{t}_0 \equiv \boldsymbol{t}_p$ and the norm estimate $\|\boldsymbol{t}_0\| \le q/2$, we find that

$$\boldsymbol{t}_0 = \boldsymbol{t} - \boldsymbol{b} \star \boldsymbol{g} \quad \text{exactly.}$$

Next we compute

$$\boldsymbol{a} \equiv \boldsymbol{g}^{-1} \star (\boldsymbol{t}_p - \boldsymbol{t}_0) \equiv \boldsymbol{b} \pmod{p},$$

and since $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{R}(p/2)$, we get $\boldsymbol{a} = \boldsymbol{b}$.

Proof of the Claim (continued)

We now compute the signature

$$\begin{aligned}
&\sigma\big(\boldsymbol{f},\boldsymbol{g},\boldsymbol{s}_p,\boldsymbol{t}_p,\phi(\boldsymbol{b})\big)\\
&\qquad = (\boldsymbol{s}_0+\boldsymbol{a}\star\boldsymbol{f},\boldsymbol{t}_0+\boldsymbol{a}\star\boldsymbol{g}) \quad \text{definition of } \sigma,\\
&\qquad = (\boldsymbol{s}-\boldsymbol{b}\star\boldsymbol{f}+\boldsymbol{a}\star\boldsymbol{f},\boldsymbol{t}-\boldsymbol{b}\star\boldsymbol{g}+\boldsymbol{a}\star\boldsymbol{g})\\
&\qquad\qquad\qquad \text{from formulas for } \boldsymbol{s}_0 \text{ and } bft_0,\\
&\qquad = (\boldsymbol{s},\boldsymbol{t}) \quad \text{since } \boldsymbol{a}=\boldsymbol{b}.
\end{aligned}$$

The definition of $\Sigma(\boldsymbol{f},\boldsymbol{g},\boldsymbol{s},\boldsymbol{t})$ lets us conclude

$$\phi(\boldsymbol{b}) \in \Sigma(\boldsymbol{f},\boldsymbol{g},\boldsymbol{s},\boldsymbol{t}),$$

This shows that ϕ is a well-defined map

$$\phi : \mathcal{R}\left(\frac{p}{2}\right) \longrightarrow \Sigma(\boldsymbol{f},\boldsymbol{g},\boldsymbol{s},\boldsymbol{t}).$$

It remains to show that ϕ is bijective.

Proof of the Claim (continued)

Fix $\boldsymbol{r} \in \Sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}, \boldsymbol{t})$. We will show that $\#\phi^{-1}(\boldsymbol{r}) = 1$.

Every coefficient of $\boldsymbol{s} - \boldsymbol{s}_p$ and $\boldsymbol{f}$ is divisible by p, so let

$$\boldsymbol{s} - \boldsymbol{s}_p = p\boldsymbol{S} \quad \text{and} \quad \boldsymbol{f} = p\boldsymbol{F}.$$

Then

$$\begin{aligned} \phi(\boldsymbol{b}) = \boldsymbol{r} &\iff \boldsymbol{S} - \boldsymbol{b} \star \boldsymbol{F} = \boldsymbol{r} \\ &\iff \boldsymbol{b} \equiv \boldsymbol{F}^{-1} \star (\boldsymbol{S} - \boldsymbol{r}) \pmod{p} \ \text{ and } \ \|\boldsymbol{b}\| \le \frac{p}{2}. \end{aligned}$$

Hence

$$\phi^{-1}(\boldsymbol{r}) = \begin{pmatrix} \text{the unique } \boldsymbol{b} \in \mathcal{R}(p/2) \text{ satisfying} \\ \boldsymbol{b} \equiv \boldsymbol{F}^{-1} \star (\boldsymbol{S} - \boldsymbol{r}) \pmod{p} \end{pmatrix}.$$

This proves the claim that ϕ is bijective. Then

$$\text{Prob} = \frac{\#\Sigma(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{s}, \boldsymbol{t})}{\#\mathcal{R}(A)} = \frac{\#\mathcal{R}(p/2)}{\#\mathcal{R}(A)} = \left(\frac{p}{2A}\right)^N$$

concludes the proof of the theorem.

NTRU and Lattice-Based Crypto: Past, Present, and Future

Joseph H. Silverman

Brown University

The Mathematics of Post-Quantum Cryptography
DIMACS Center, Rutgers University

January 12–16, 2015