An Introduction to Lattices, Lattice Reduction, and Lattice-Based Cryptography

Joseph H. Silverman

Brown University

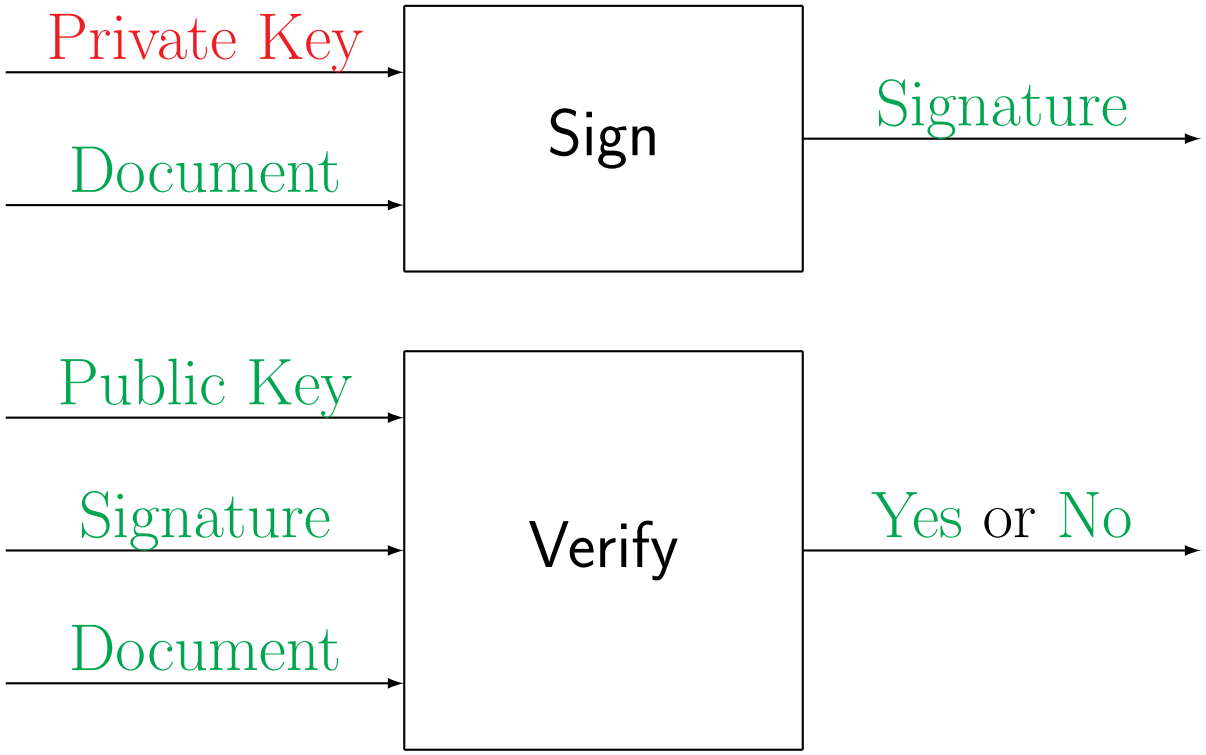
PCMI Lecture Series

July 6–10, 2020

Lecture 5. Lattice-Based Digital Signatures and Rejection Sampling

Digital Signatures

We recall that a **Digital Signature Scheme** consists of a signing function and a verification function:



For a valid (PubKey,PrivKey) public/private key pair, a document Doc, and a purported signature Sig:

$$\mathsf{Verify}(\mathrm{PubKey},\mathrm{Sig},\mathrm{Doc}) = \mathrm{Yes} \iff \mathrm{Sig} = \mathsf{Sign}(\mathrm{PrivKey},\mathrm{Doc}).$$

CVP Digital Signatures — GGH

We already briefly discussed digital signature schemes based on the IFP, DLP, and ECDLP. Today we'll discuss lattice-based digital signature schemes whose security relies on SVP and/or CVP.

The prototypical example is the **GGH digital signature scheme**, whose basic set-up is the same as that of the GGH lattice-based cryptosystem:

$$\begin{aligned} \text{Private Key} &= \text{a good basis } \mathcal{B}^{\text{good}} \text{ for a lattice } \mathcal{L} \\ &= \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}. \\ \text{Public Key} &= \text{a bad basis } \mathcal{B}^{\text{bad}} \text{ for the lattice } \mathcal{L}. \\ &= \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n\}. \\ \text{Document} &= \text{a vector } \boldsymbol{d} \text{ that is } \underline{\text{not}} \text{ in } \mathcal{L}. \end{aligned}$$

- N.B. In practice, the "document" $\boldsymbol{d}$ that Alice signs is the output of a hash function applied to her actual (long) document, plus some random bits.

CVP Digital Signatures — GGH

- Alice uses Babai with $\mathcal{B}^{\text{good}}$ to find a lattice vector close to $\boldsymbol{d}$. Thus she writes

$$\boldsymbol{d} = \delta_1 \boldsymbol{v}_1 + \cdots + \delta_n \boldsymbol{v}_n \quad \text{with } \delta_1, \ldots, \delta_n \in \mathbb{R}.$$

  She rounds the coefficients to get a lattice vector

$$\boldsymbol{s} = \lfloor \delta_1 \rceil \boldsymbol{v}_1 + \cdots + \lfloor \delta_n \rceil \boldsymbol{v}_n \in \mathcal{L}$$

  that is close to $\boldsymbol{d}$.
- Alice writes $\boldsymbol{s}$ using the bad basis $\mathcal{B}^{\text{bad}}$,

$$\boldsymbol{s} = s_1 \boldsymbol{w}_1 + \cdots + s_n \boldsymbol{w}_n. \quad (*)$$

  Her signature on $\boldsymbol{d}$ is the n-tuple $(s_1, \ldots, s_n)$.
- Bob uses the n-tuple and bad public basis $\mathcal{B}^{\text{bad}}$ to reconstruct $\boldsymbol{s}$ using the formula $(*)$. Then $\boldsymbol{s}$ is automatically in $\mathcal{L}$, and Bob verifies that the signature is valid by checking that:

$\boldsymbol{s}$ is sufficiently close to $\boldsymbol{d}$.

Security of GGH and other CVP-based Digital Signatures

Some security issues for CVP-based digital signatures:

- *Cominatorial Security*: The spaces of keys and signature must to be large enough so that an attacker cannot check all their elements. Indeed, collision search algorithms often run in time $O(\sqrt{K})$ on sets of size K.
- *Lattice Security*: Just as with CVP-based public key cryptosystems, one needs to check that latttice reduction algorithms such as LLL and its variants cannot solve apprCVP well enough to forge signatures.
- *Practicality*: The GGH digital signature scheme has impractically large public keys. One might try to instead use an NTRU-type construction, although it is not entirely clear how to do that, since an NTRU lattice only provides a good basis for a half-dimensional sublattice. (It can be done.)

Transcript Attacks on Signature Schemes

Yet Another Thing to Worry About!

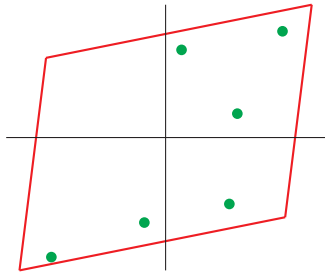
- Each signature potentially reveals some information about the private key.
- So a long transcript of signatures might compromise security.

This is not an idle threat. A number of people, including Gentry–Szydlo (2002) and Nguyen–Regev (2009), developed practical transcript attacks on GGH and other early CVP-based signature schemes.

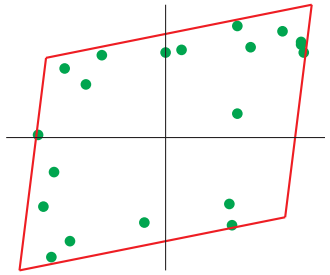
How might this work for GGH? A signature $\boldsymbol{s}$ on a document $\boldsymbol{d}$ reveals a vector in the centered fundamental domain spanned by the good basis:

$$\boldsymbol{s} - \boldsymbol{d} \in \mathcal{F}^{\text{good}} := \left\{ t_1 \boldsymbol{v}_1 + \cdots + t_n \boldsymbol{v}_n : -\frac{1}{2} \le t_i \le \frac{1}{2} \right\}.$$

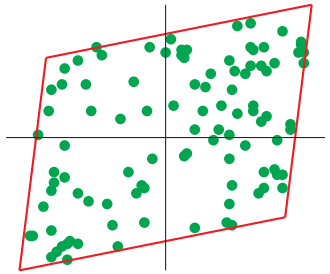
Illustrating a Transcript Attack



A few signatures aren't much help in reconstructing $\mathcal{F}^{\text{good}}$.



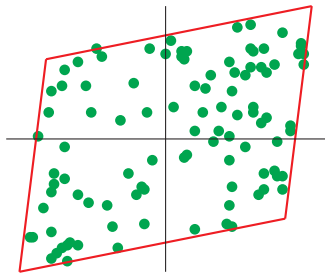
With a moderate number of signatures, the fundamental domain $\mathcal{F}^{\text{good}}$ starts to emerge.



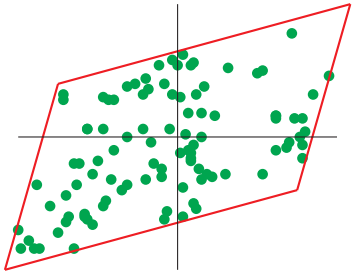
With lots of signatures, one may be able to reconstruct the fundamental domain $\mathcal{F}^{\text{good}}$.

Different Private Good Bases Have Different Transcripts

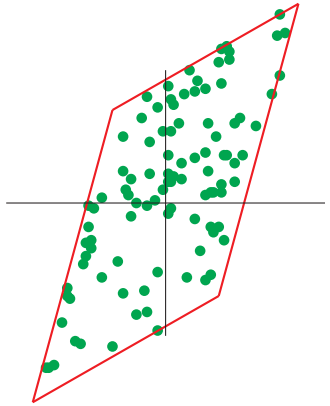
Alice, Bob and Carl have their own private good bases. Each basis has a different fundamental domain, and a transcript uniformly fills out that domain.



A Fundamental Domain $\mathcal{F}^{\text{good}}_{\text{Alice}}$ for Alice's Private Basis



A Fundamental Domain $\mathcal{F}^{\text{good}}_{\text{Bob}}$ for Bob's Private Basis



A Fundamental Domain $\mathcal{F}^{\text{good}}_{\text{Carl}}$ for Carl's Private Basis

Rejection Sampling to the Rescue

Rejection sampling is a technique from statistics in which one starts with a random process whose output has a known distribution, and by judiciously throwing away (rejecting) some of the output values, one creates a new random process whose output is some other desired distribution.

Lyubashevsky constructed a lattice-based identification scheme and digital signature using a technique that he called *aborting*, which is a version of rejection sampling.

The basic idea is that Alice generates a signature on her document and checks if it will leak information about her individual private key. If it leaks, then she rejects that signature and generates another one. She continues to do this until finding a safe signature, which she publishes.

First Issue: How does Alice generate multiple signatures on the same document?

Rejection Sampling for GGH

Remember that Alice isn't really signing her document, she's signing a hash of her document. In fact, for various reasons, it is advisable that she signs something like

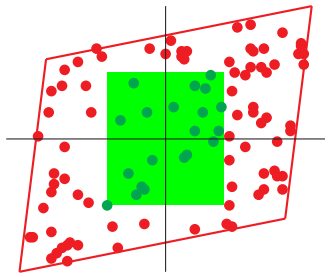
$$\mathsf{Hash}\begin{pmatrix}\text{Alice's actual document concate-}\\ \text{nated with, say, 80 random bits}\end{pmatrix}.$$

So if a signature is rejected, she just selects a new 80 bits and try again. Her signature includes the 80 random bits that she used in her unrejected signature.

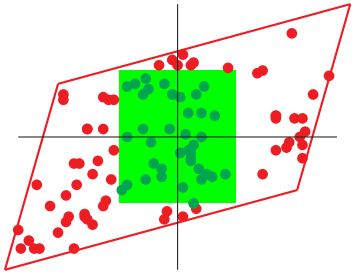
How might that work for GGH? Alice's signatures give a set of points that are uniformly distributed in her fundamental domain $\mathcal{F}^{\text{good}}_{\text{Alice}}$. Similarly, Bob's signatures give points uniformly distributed in $\mathcal{F}^{\text{good}}_{\text{Bob}}$, and the same for Carl. But suppose that they fix a region that's common to $\mathcal{F}^{\text{good}}_{\text{Alice}}$, $\mathcal{F}^{\text{good}}_{\text{Bob}}$, and $\mathcal{F}^{\text{good}}_{\text{Carl}}$, and they reject signatures outside that common region.

Selecting a Region Common to All Fundamental Domains

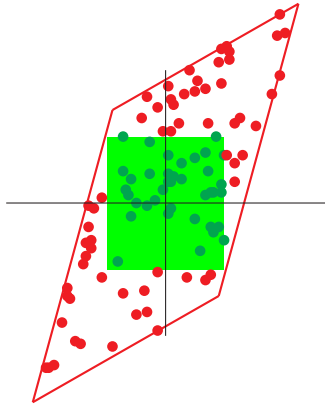
The Green Box is common to everyone's centered fundamental domain. So equidistributed points in the Green Box yield no information about the private basis.



Alice's Private $\mathcal{F}^{\text{good}}_{\text{Alice}}$ with common box



Bob's Private $\mathcal{F}^{\text{good}}_{\text{Bob}}$ with common box



Carl's Private $\mathcal{F}^{\text{good}}_{\text{Carl}}$ with common box

GGH with Rejection Sampling

1. The public parameters include a cut-off value B with the property that everyone's centered fundamental domain $\mathcal{F}^{\text{good}}$ contains the centered box
$$\left\{(x_1, \ldots, x_n) \in \mathbb{R}^n : -\tfrac{1}{2}B \le x_i \le \tfrac{1}{2}B\right\}.$$
2. Alice computes a vector $\boldsymbol{d}$ to sign:
$$\boldsymbol{d} = \mathsf{Hash}(\text{Alice's Doc} \,\|\, \text{Random Bits}).$$
3. Alice uses her good basis $\mathcal{B}^{\text{good}}$ to find a lattice vector $\boldsymbol{s}$ that is close to $\boldsymbol{d}$.
4. If any coordinate of $\boldsymbol{s} - \boldsymbol{d}$ satisfies $|x| > B$, then Alice rejects $\boldsymbol{s}$. She returns to Step 2 and selects new random bits.
5. Otherwise Alice accepts and publishes the signature $\boldsymbol{s}$. (Notice that it is only now that $\boldsymbol{s}$ becomes public.) She will also need to publish the random bits that she used in Step 2.

Transcript Security — At A Cost

- Assuming that the hash function and pseudo-random number generator work as advertised, Alice's and Bob's and Carl's transcripts will have exactly the same distribution. They thus leak no private key information.
- We've neglected the added cost of rejection sampling.

  How many $\boldsymbol{s}$ get rejected before one of them is accepted?

  For a naive implementation of GGH, the volume of the common box will be very small compared to the volume of the fundamental domain. So using rejection sampling is at best inefficient, and at worst, completely impractical.
- Lyubashevsky and others use various methods to make it easier to find non-rejected signatures, while maintaining the property that the distribution of signatures is independent of the private key.

A Prototypical Practical Rejection Sampling Scheme

There are also lattice-based signature schemes for which the simple box rejection method is practical. In the remaining time, I want to describe one such scheme and sketch the proof of transcript security.

In order to concentrate on the transcript aspect, I will not describe the actual signature scheme. You can read the full details in the lecture notes.

The scheme again uses the ring

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

For $\boldsymbol{f}(X) = \sum a_i X^i$, we set the notation

$$\|\boldsymbol{f}\|_\infty := \max_{0 \le i < N} |a_i|,$$

$$\mathcal{R}[b] := \{\boldsymbol{f} \in \mathcal{R} : \|\boldsymbol{f}\|_\infty \le b\}.$$

Example:

$$\mathcal{R}[1] = \{\text{polynomials with coefficients in } \{-1, 0, 1\}\}\ .$$

A Prototypical Rejection Sampling Algorithm

1. **Parameters**: N and k
2. **Secret**: Alice chooses a secret $\boldsymbol{f} \in \mathcal{R}[1]$.
3. **Randomness**: Alice chooses a random $\boldsymbol{y} \in \mathcal{R}[k]$.
4. **Hash**: Alice creates $\boldsymbol{c} \in \mathcal{R}[1]$ using a hash function. (The value of $\boldsymbol{c}$ is tied to her document, her public key, and her random polynomial in a way that's described in the lecture notes.)
5. **Sign**: Alice computes $\boldsymbol{s} = \boldsymbol{f} \cdot \boldsymbol{c} + \boldsymbol{y}$.
6. **Rejection Sampling**: If $\|\boldsymbol{s}\|_\infty \geq k - N$, Alice goes back to Step 3 and chooses a new value for $\boldsymbol{y}$.
7. **Publication**: Alice publishes her signature $(\boldsymbol{s}, \boldsymbol{c})$.

Transcript Insecurity Without Rejection Sampling

Suppose that Alice publishes a transcript of signatures

$$(\boldsymbol{s}_1, \boldsymbol{c}_1),\ (\boldsymbol{s}_2, \boldsymbol{c}_2),\ (\boldsymbol{s}_3, \boldsymbol{c}_3),\ \ldots\ ,$$

but she ignores the rejection sampling step. Then Eve can compute the following sum:

$$\begin{aligned}
&\frac{1}{T}\sum_{i=1}^{T} s_i(X)\cdot c_i(X^{N-1}) \\
&= \frac{1}{T}\sum_{i=1}^{T}\Big(\boldsymbol{f}(X)\cdot \boldsymbol{c}_i(X) + \boldsymbol{y}_i(X)\Big)\cdot c_i(X^{N-1}) \\
&= \boldsymbol{f}(X)\cdot\left(\frac{1}{T}\sum_{i=1}^{T}\boldsymbol{c}_i(X)\cdot c_i(X^{N-1})\right) + \frac{1}{T}\sum_{i=1}^{T}\boldsymbol{y}_i(X)\cdot c_i(X^{N-1}) \\
&\approx \frac{4N}{3}\boldsymbol{f}(X) \qquad \text{when } T \text{ is large, since the } \boldsymbol{c}_i \text{ and } \boldsymbol{y}_i \\
&\qquad\qquad\qquad\qquad\quad \text{are random and independent.}
\end{aligned}$$

Transcript Security With Rejection Sampling

Suppose that Alice uses rejection sampling when creating

$$(\boldsymbol{s}_1, \boldsymbol{c}_1),\ (\boldsymbol{s}_2, \boldsymbol{c}_2),\ (\boldsymbol{s}_3, \boldsymbol{c}_3),\ \ldots\ ,$$

Claim: The transcript reveals no information about Alice's private key.

What does it mean to say that the transcript reveals no information? We formulate this as a conditional probability statement:

Claim: For all $\boldsymbol{f}_0 \in \mathcal{R}[1]$ and all $\boldsymbol{s}_0 \in \mathcal{R}[k-N]$,

$$\text{Prob}\left(\begin{matrix}\text{a signature } (\boldsymbol{s}, \boldsymbol{c}) \text{ created} \\ \text{using } \boldsymbol{f} \text{ satisfies } \boldsymbol{s} = \boldsymbol{s}_0\end{matrix} \;\middle|\; \boldsymbol{f} = \boldsymbol{f}_0\right)$$

does not depend on $\boldsymbol{f}_0$.

Proof of Transcript Security

$$\operatorname*{Prob}_{\substack{\boldsymbol{c}\in\mathcal{R}[1]\\ \boldsymbol{y}\in\mathcal{R}[k]}}\left(\boldsymbol{s}=\boldsymbol{s}_0 \;\middle|\; \boldsymbol{f}=\boldsymbol{f}_0\right)$$

$$= \frac{\#\left\{(\boldsymbol{c},\boldsymbol{y})\in\mathcal{R}[1]\times\mathcal{R}[k] \;\middle|\; \boldsymbol{s}_0=\boldsymbol{c}\cdot\boldsymbol{f}_0+\boldsymbol{y}\right\}}{\#\mathcal{R}[1]\cdot\#\mathcal{R}[k]}$$

$$= \frac{\#\left\{\boldsymbol{c}\in\mathcal{R}[1] \;\middle|\; \boldsymbol{s}_0-\boldsymbol{c}\cdot\boldsymbol{f}_0\in\mathcal{R}[k]\right\}}{\#\mathcal{R}[1]\cdot\#\mathcal{R}[k]} \qquad (*).$$

Our choice of parameters gives

$$\begin{aligned}\left\|\boldsymbol{s}_0-\boldsymbol{c}\cdot\boldsymbol{f}_0\right\|_\infty &\le \left\|\boldsymbol{s}_0\right\|_\infty+\left\|\boldsymbol{c}\cdot\boldsymbol{f}_0\right\|_\infty\\ &\le \left\|\boldsymbol{s}_0\right\|_\infty+N\left\|\boldsymbol{c}\right\|_\infty\cdot\left\|\boldsymbol{f}_0\right\|_\infty\\ &\le (k-N)+N=k.\end{aligned}$$

Thus the condition $\boldsymbol{s}_0-\boldsymbol{c}\cdot\boldsymbol{f}_0\in\mathcal{R}[k]$ in $(*)$ is vacuous!

Proof of Transcript Security (continued)

Hence

$$\begin{aligned}
&\operatorname*{Prob}_{\substack{\boldsymbol{c}\in\mathcal{R}[1]\\ \boldsymbol{y}\in\mathcal{R}[k]}}\left(\boldsymbol{s}=\boldsymbol{s}_0 \;\middle|\; \boldsymbol{f}=\boldsymbol{f}_0\right) \\
&\qquad = \frac{\#\left\{\boldsymbol{c}\in\mathcal{R}[1] \;\middle|\; \boldsymbol{s}_0-\boldsymbol{c}\cdot\boldsymbol{f}_0\in\mathcal{R}[k]\right\}}{\#\mathcal{R}[1]\cdot\#\mathcal{R}[k]} \qquad (*) \\
&\qquad = \frac{\#\left\{\boldsymbol{c}\in\mathcal{R}[1]\right\}}{\#\mathcal{R}[1]\cdot\#\mathcal{R}[k]} \\
&\qquad = \frac{1}{\#\mathcal{R}[k]}.
\end{aligned}$$

This completes the proof that the probability that a given $\boldsymbol{s}_0 \in \mathcal{R}[k-N]$ appears in a signature is independent of the private key $\boldsymbol{f}_0$ used to create it.

Probability for Accept/Reject

It remains to show that it is feasible to create non-rejected signatures. Each coefficient of $\boldsymbol{f} \cdot \boldsymbol{c}$ is a "random" sum of -1, 0, 1, so looks like a random walk whose absolute value is highly unlikely to be larger than, say, $5\sqrt{N}$. This allows us to estimate:

$$\begin{aligned}
&\operatorname*{Prob}_{\substack{\boldsymbol{c}\in\mathcal{R}[1]\\ \boldsymbol{y}\in\mathcal{R}[k]}}\Big(\|\boldsymbol{c}\star\boldsymbol{f}+\boldsymbol{y}\|_\infty \le k-N\Big) \\
&\qquad \gtrapprox \operatorname*{Prob}_{\boldsymbol{y}\in\mathcal{R}[k]}\Big(\|\boldsymbol{y}\|_\infty \le k-N-5\sqrt{N}\Big) \\
&\qquad = \frac{\#\mathcal{R}[k-N-5\sqrt{N}]}{\#\mathcal{R}[k]} \\
&\qquad = \left(1-\frac{N+5\sqrt{N}}{2k+1}\right)^N \\
&\qquad \approx \exp\left(\frac{-N^2}{2k+1}\right) \quad \text{if } k > N^2 \gg 1.
\end{aligned}$$

And the Road Goes Ever On. . .

We've come to the end of this introduction to lattices and lattice-based cryptography.

I want to thank all of you for coming,
and to thank the graduate
summer school organizers,
Jennifer Balakrishnan, Bjorn Poonen,
and Akshay Venkatesh,
for inviting me to speak.

An Introduction to Lattices, Lattice Reduction, and Lattice-Based Cryptography

Joseph H. Silverman

Brown University

PCMI Lecture Series

July 6–10, 2020