Building and Attacking Lattice-Based Signatures

Part II: Security Evaluation and Physical Attacks

Mehdi Tibouchi

NTT Secure Platform Laboratories

Crypto-CO, Medellín, 2019–06–13

©2019 NTT Secure Platform Laboratories

Outline

Evaluating the security of lattice-based schemes

- A primer on lattices
- Example: the hardness of SIS
- Methodology for setting parameters

Physical attacks against BLISS

- The BLISS signature scheme
- Fault attack on the Gaussian sampling
- SCA on the rejection sampling

©2019 NTT Secure Platform Laboratories

Outline

Evaluating the security of lattice-based schemes
- A primer on lattices
- Example: the hardness of SIS
- Methodology for setting parameters

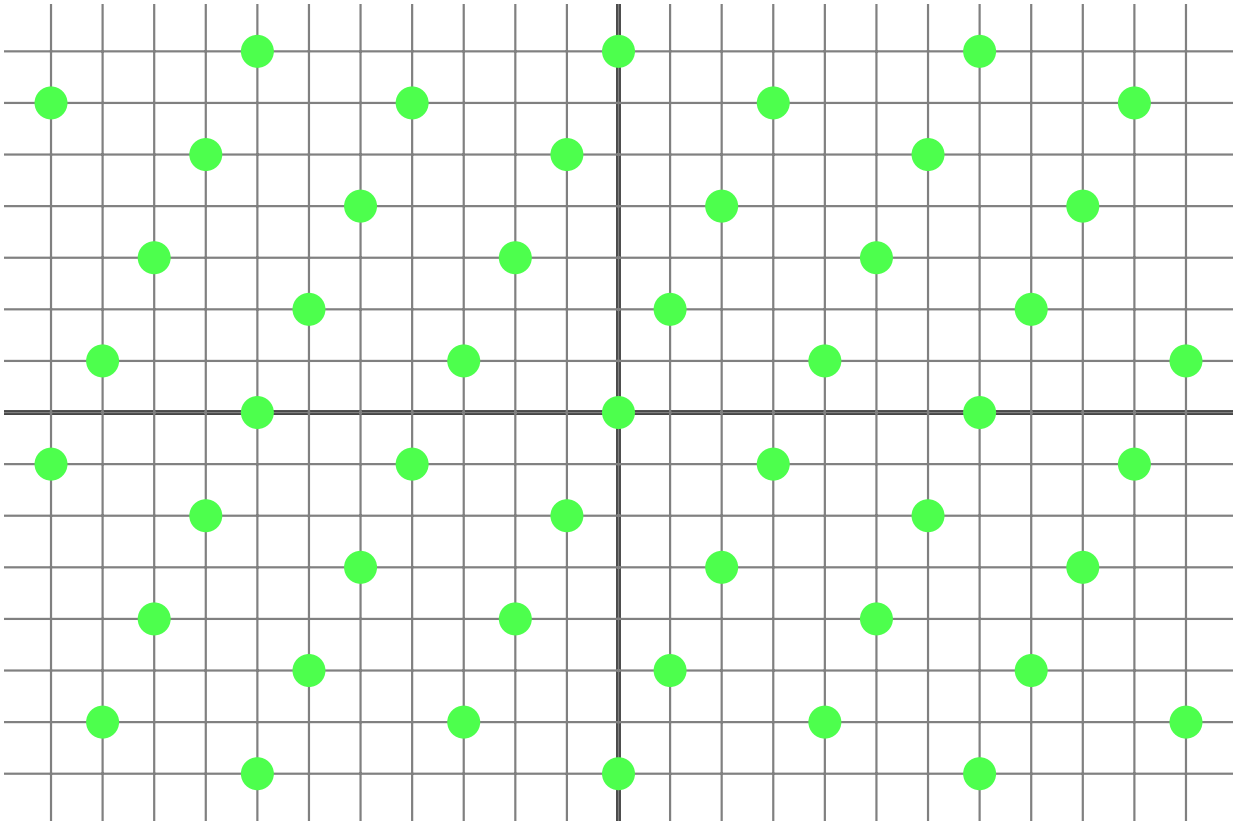
Physical attacks against BLISS
- The BLISS signature scheme
- Fault attack on the Gaussian sampling
- SCA on the rejection sampling

©2019 NTT Secure Platform Laboratories

Outline

©2019 NTT Secure Platform Laboratories

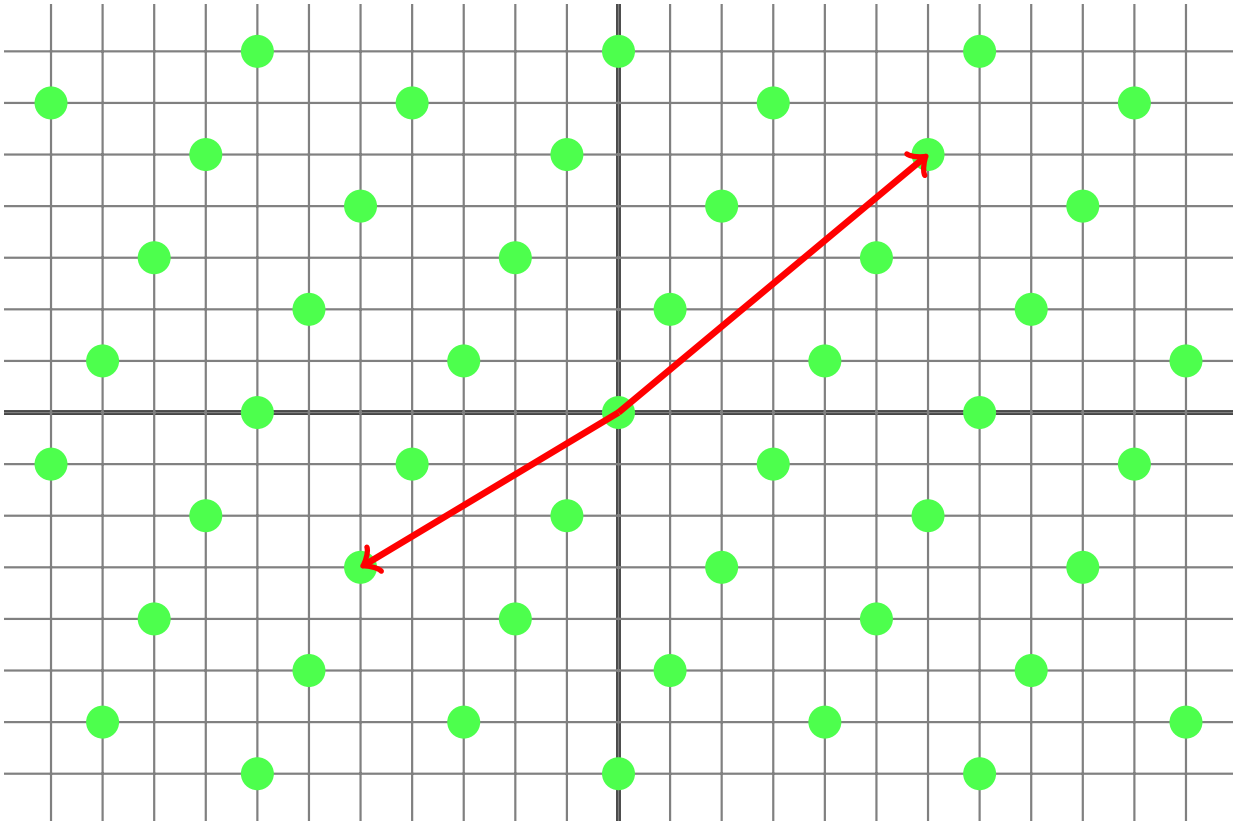
A primer on lattices



A lattice L is a subgroup of $\mathbb{Z}^n$ for some n:
a regular arrangement of points in $\mathbb{R}^n$.

©2019 NTT Secure Platform Laboratories

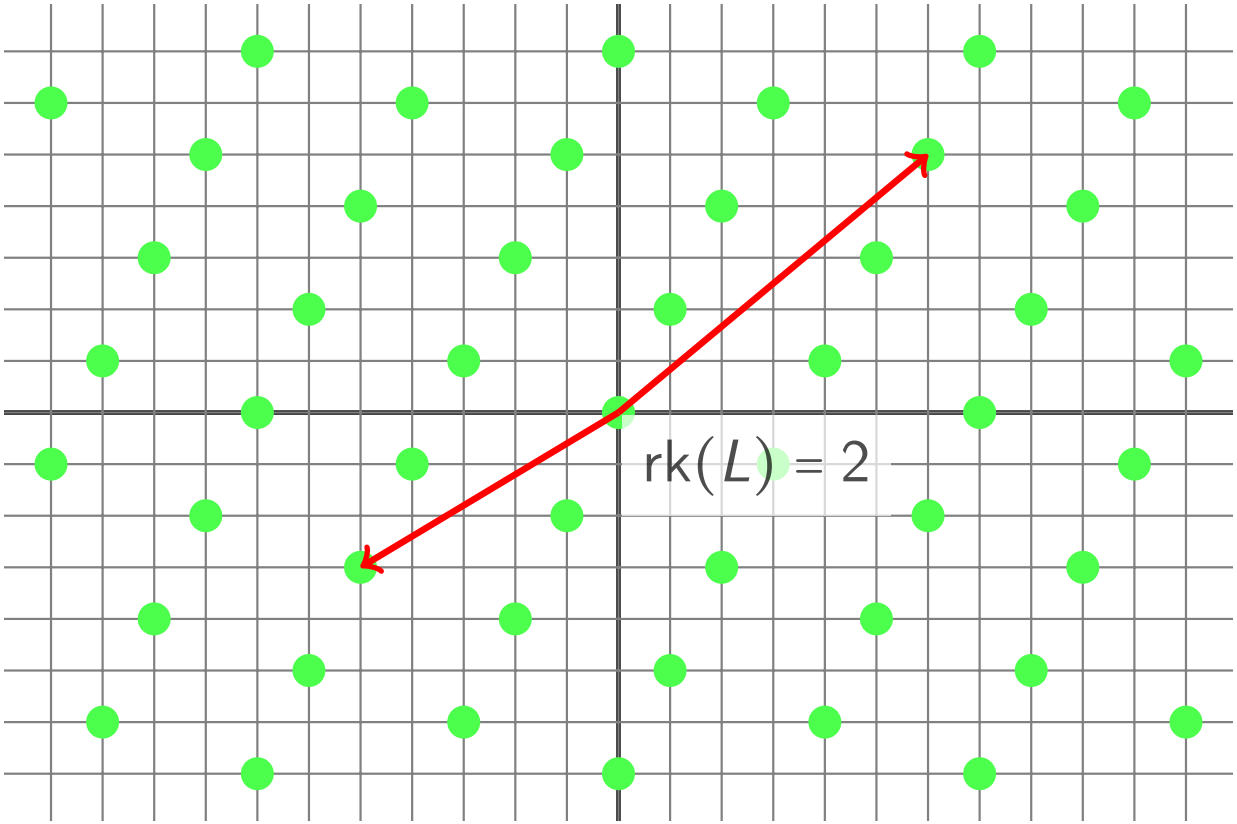
A primer on lattices



Often represented by a basis
(minimal generating set of vectors in L).

©2019 NTT Secure Platform Laboratories

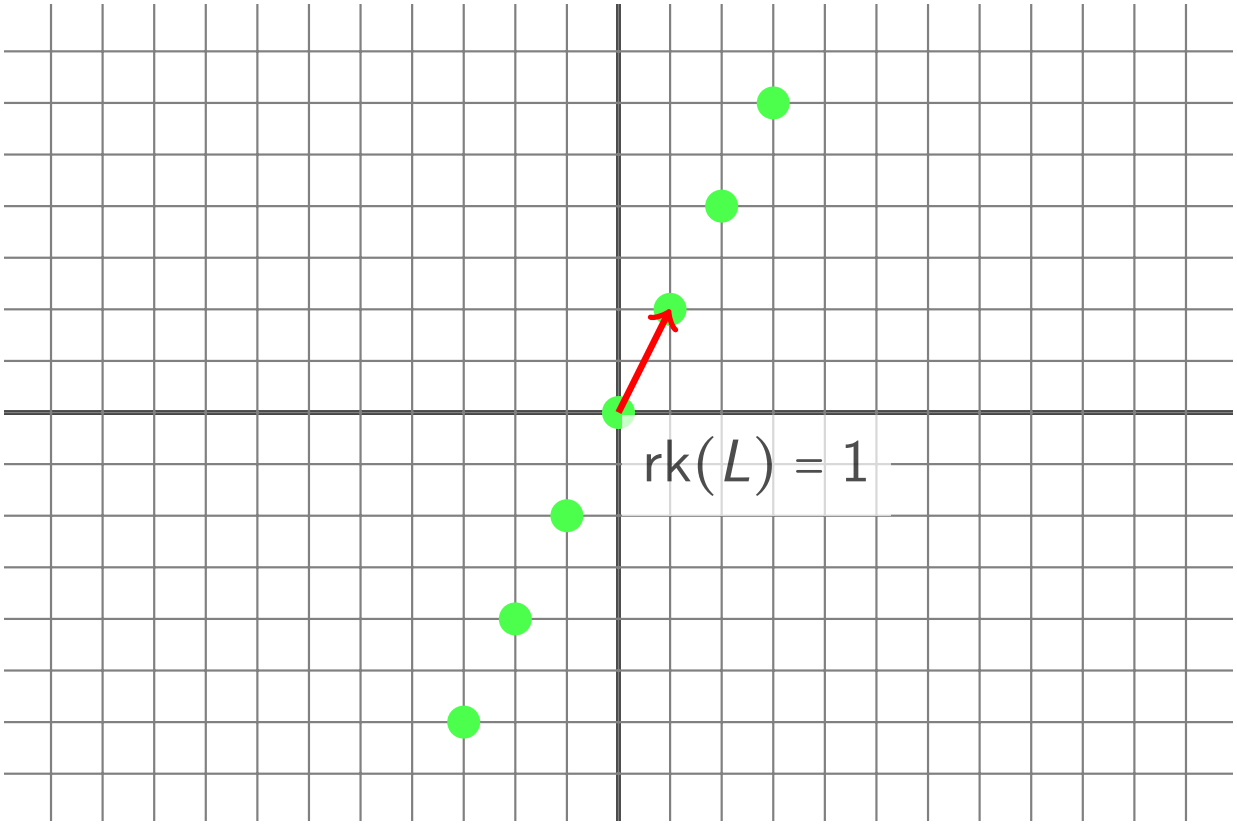
A primer on lattices



The number of vectors in a basis is called the rank or dimension $\mathrm{rk}(L)$. It is well-defined.

©2019 NTT Secure Platform Laboratories

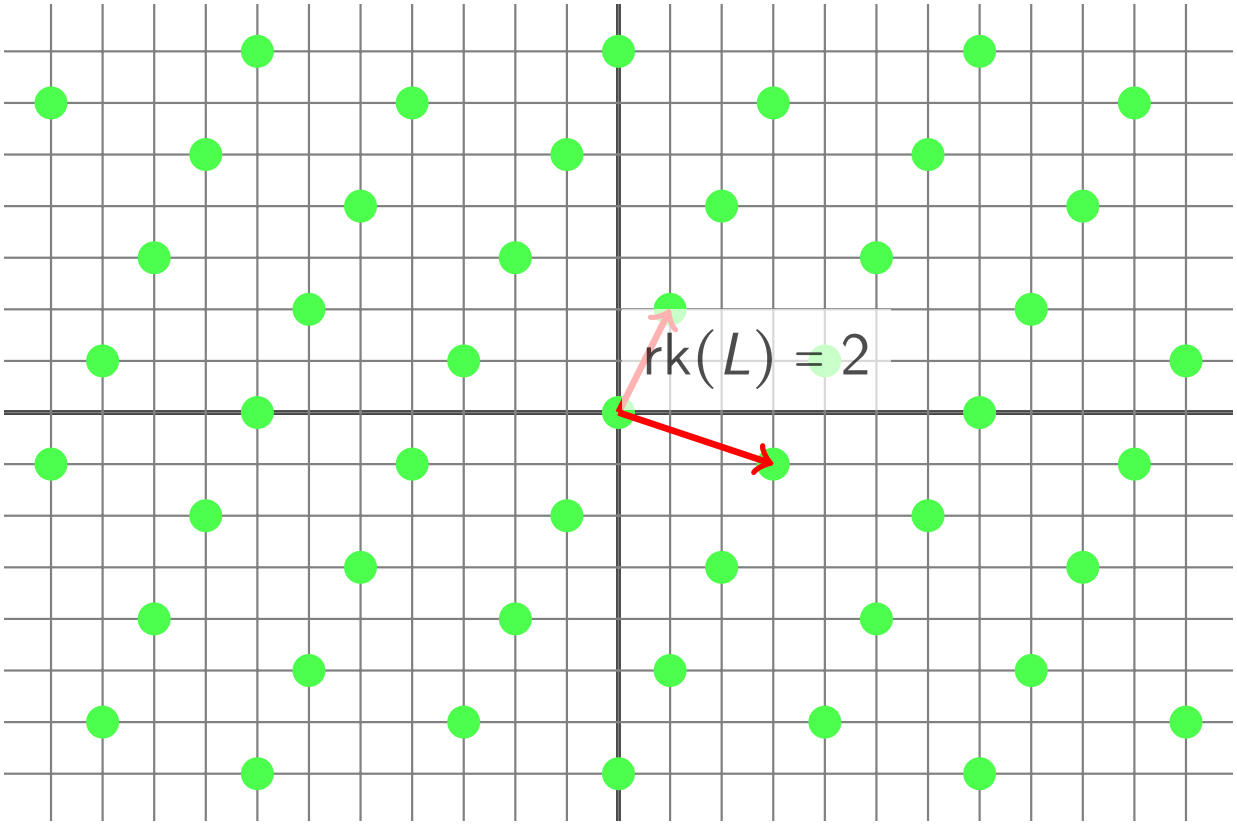
A primer on lattices



The number of vectors in a basis is called the rank or dimension $\mathrm{rk}(L)$. It is well-defined.

©2019 NTT Secure Platform Laboratories

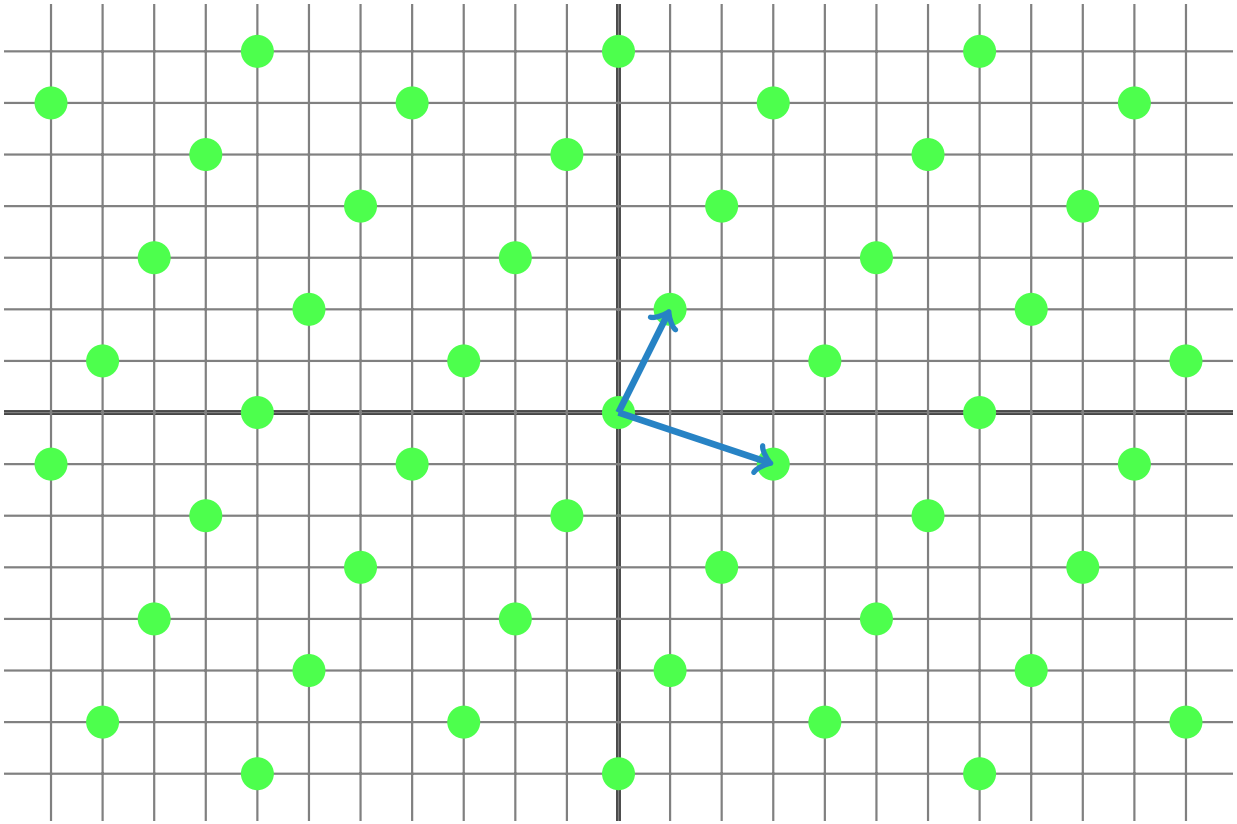
A primer on lattices



The number of vectors in a basis is called the rank or dimension $\mathrm{rk}(L)$. It is well-defined.

©2019 NTT Secure Platform Laboratories

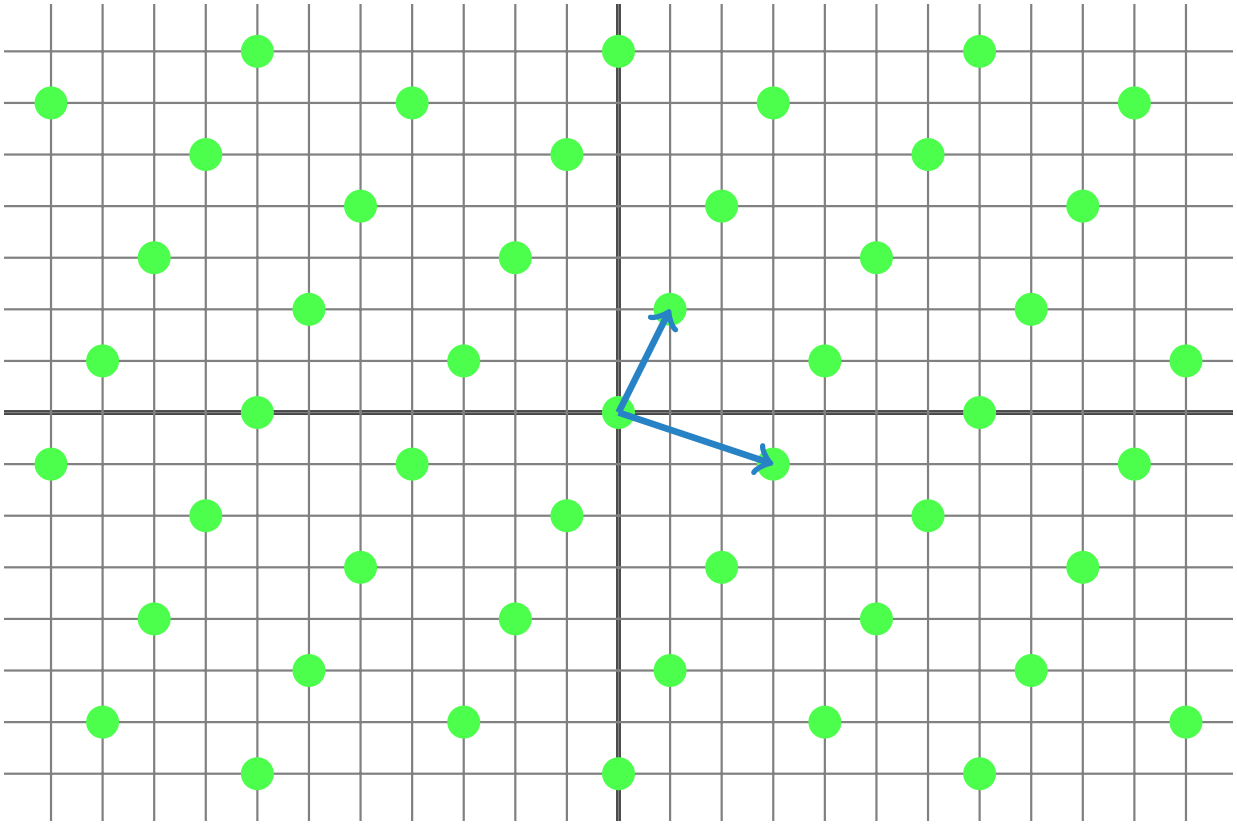
A primer on lattices



Some bases are better than others: with shorter, almost orthogonal vectors. We call them reduced basis.

©2019 NTT Secure Platform Laboratories

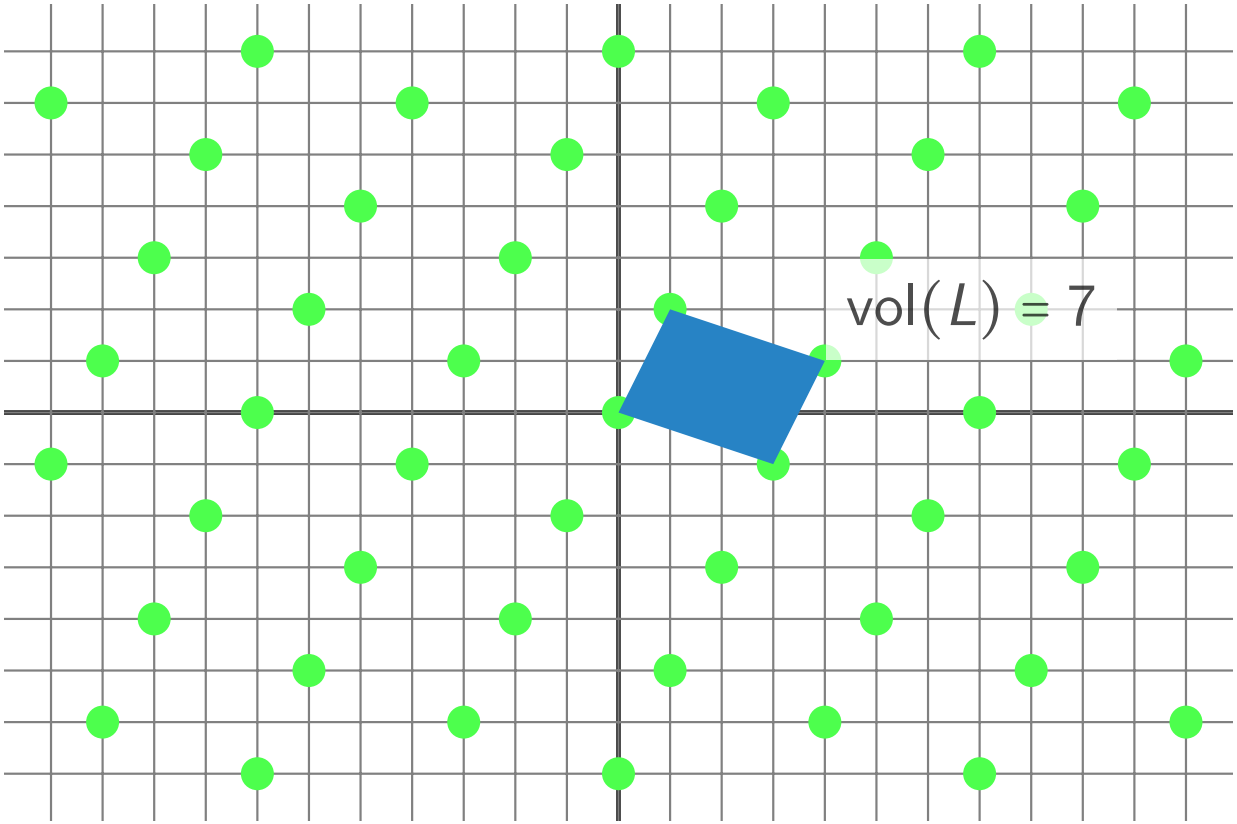
A primer on lattices



We have algorithms, such as LLL, to compute reduced bases. In low dimension (say $\lesssim 100$), we can obtain "optimal" lattice reduction in practice.

©2019 NTT Secure Platform Laboratories

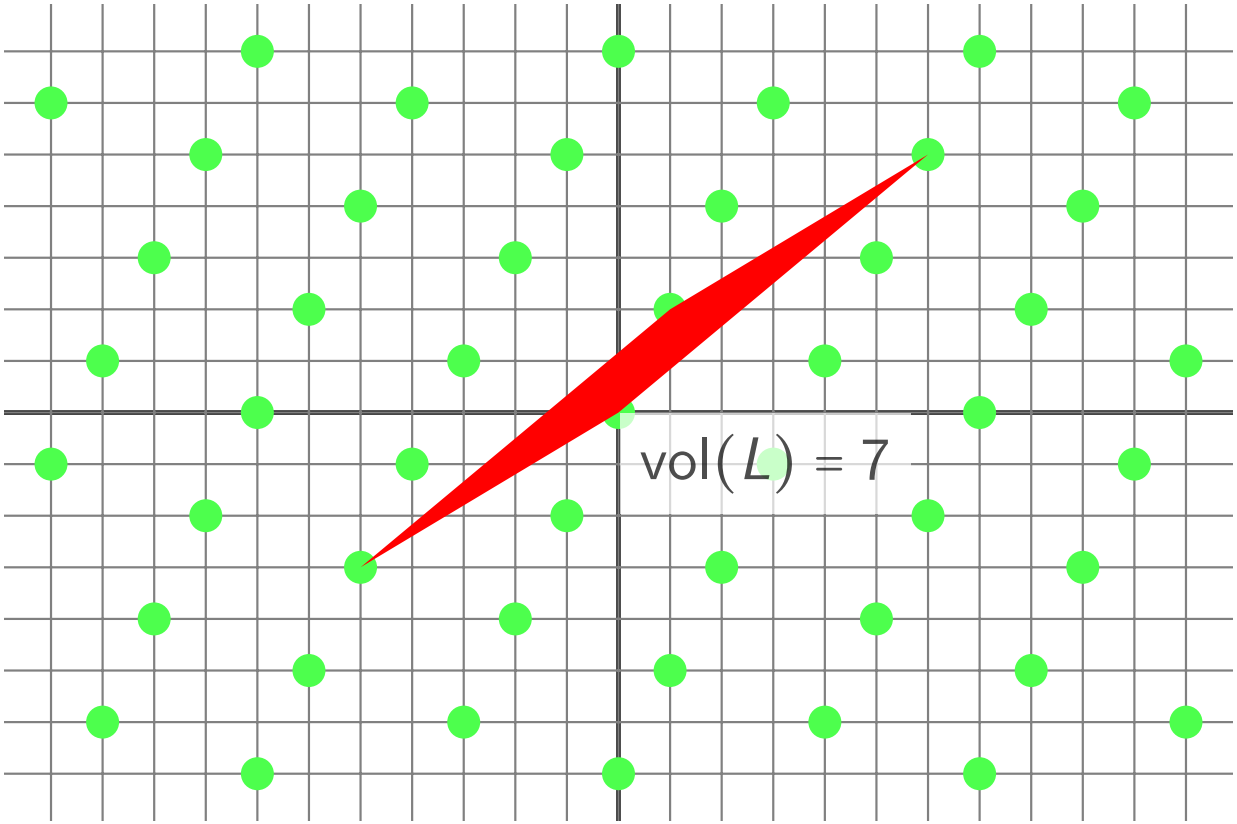
A primer on lattices



Another important invariant: lattice volume; d-dimensional volume of the parallelipiped defined by a basis. Independent of the basis.

©2019 NTT Secure Platform Laboratories

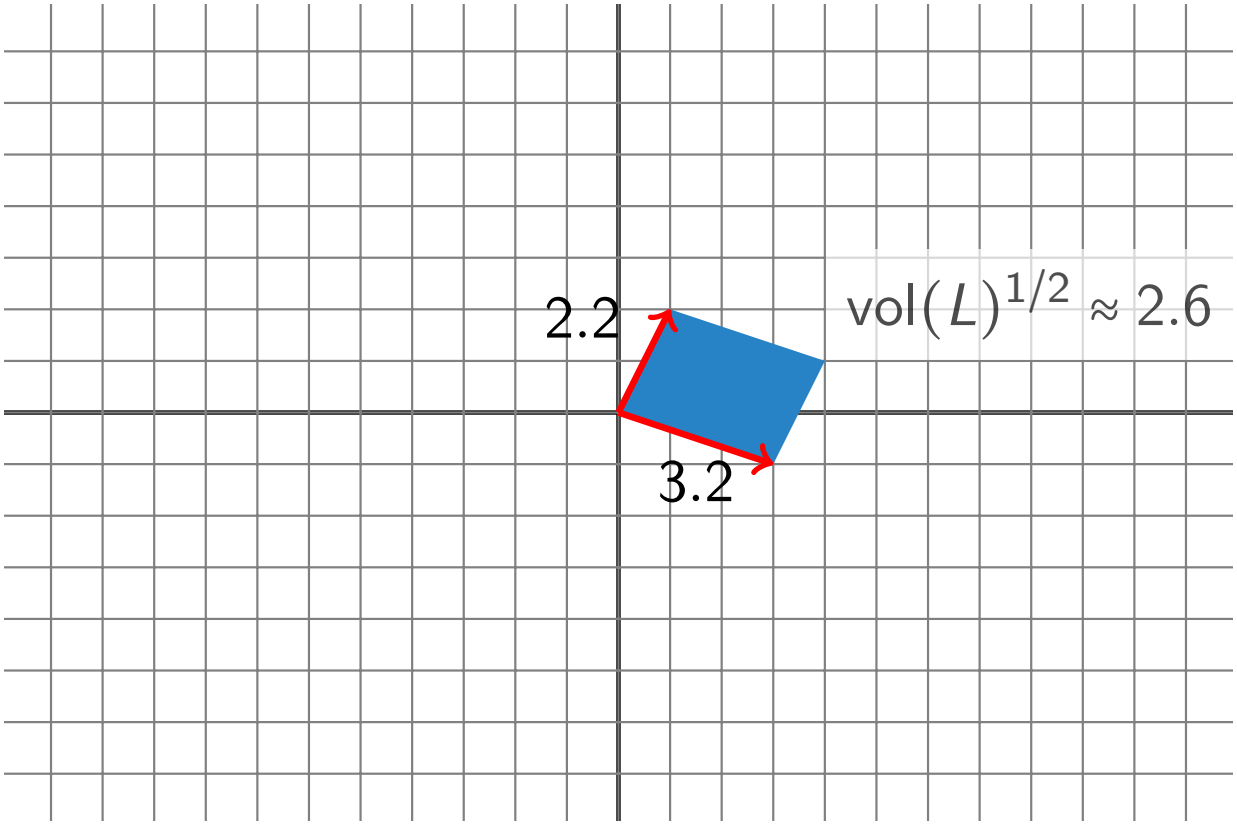
A primer on lattices



Another important invariant: lattice volume; d-dimensional volume of the parallelipiped defined by a basis. Independent of the basis.

©2019 NTT Secure Platform Laboratories

A primer on lattices



For "typical" (e.g. random) full-rank lattices, vectors in a short basis are all roughly of the same length $\approx \mathrm{vol}(L)^{1/\mathrm{rk}(L)}$.

©2019 NTT Secure Platform Laboratories

Problems related to lattices

- Given a (non-zero) lattice $L \subset \mathbb{Z}^n$, we define its first minimum:

$$\lambda_1(L) = \min_{\mathbf{x} \in L \setminus \{0\}} \|\mathbf{x}\|$$

  ($\|\cdot\|$ usually Euclidean norm). More generally, we can define the i-th minimum $\lambda_i(L)$ for $1 \leq i \leq \operatorname{rk} L$.

- Example lattice problems:
  - **SVP**: given L, find $\mathbf{x} \in L$ with $\|\mathbf{x}\| = \lambda_1(L)$
  - **SIVP**: given L of full rank n, find $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ basis of L with $\|\mathbf{b}_i\| = \lambda_i(L)$
  - **CVP**: given L and $\mathbf{y} \in \mathbb{Z}^n$, find $\mathbf{x} \in L$ minimizing $\|\mathbf{x} - \mathbf{y}\|$
  - there are more (**uSVP**, **GapSVP**, etc.)
- Also important, approximate variants of these problems:
  - **SVP**$_\gamma$: given L, find $\mathbf{x} \in L \setminus \{0\}$ with $\|\mathbf{x}\| \leq \gamma \lambda_1(L)$
  - **CVP**$_\gamma$: given L and $\mathbf{y} \in \mathbb{Z}^n$, find $\mathbf{x} \in L$ such that $\|\mathbf{x} - \mathbf{y}\| \leq \gamma \cdot d(\mathbf{y}, L)$

©2019 NTT Secure Platform Laboratories

Problems related to lattices

- Given a (non-zero) lattice $L \subset \mathbb{Z}^n$, we define its first minimum:

$$\lambda_1(L) = \min_{\mathbf{x} \in L \setminus \{0\}} \|\mathbf{x}\|$$

  ($\|\cdot\|$ usually Euclidean norm). More generally, we can define the i-th minimum $\lambda_i(L)$ for $1 \leq i \leq \operatorname{rk} L$.
- Example lattice problems:
  - **SVP**: given L, find $\mathbf{x} \in L$ with $\|\mathbf{x}\| = \lambda_1(L)$
  - **SIVP**: given L of full rank n, find $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ basis of L with $\|\mathbf{b}_i\| = \lambda_i(L)$
  - **CVP**: given L and $\mathbf{y} \in \mathbb{Z}^n$, find $\mathbf{x} \in L$ minimizing $\|\mathbf{x} - \mathbf{y}\|$
  - there are more (**uSVP**, **GapSVP**, etc.)
- Also important, approximate variants of these problems:
  - $\mathbf{SVP}_\gamma$: given L, find $\mathbf{x} \in L \setminus \{0\}$ with $\|\mathbf{x}\| \leq \gamma \lambda_1(L)$
  - $\mathbf{CVP}_\gamma$: given L and $\mathbf{y} \in \mathbb{Z}^n$, find $\mathbf{x} \in L$ such that $\|\mathbf{x} - \mathbf{y}\| \leq \gamma \cdot d(\mathbf{y}, L)$

©2019 NTT Secure Platform Laboratories

Problems related to lattices

- Given a (non-zero) lattice $L \subset \mathbb{Z}^n$, we define its first minimum:

$$\lambda_1(L) = \min_{\mathbf{x} \in L \setminus \{0\}} \|\mathbf{x}\|$$

  ($\|\cdot\|$ usually Euclidean norm). More generally, we can define the i-th minimum $\lambda_i(L)$ for $1 \leq i \leq \operatorname{rk} L$.
- Example lattice problems:
  - **SVP**: given L, find $\mathbf{x} \in L$ with $\|\mathbf{x}\| = \lambda_1(L)$
  - **SIVP**: given L of full rank n, find $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ basis of L with $\|\mathbf{b}_i\| = \lambda_i(L)$
  - **CVP**: given L and $\mathbf{y} \in \mathbb{Z}^n$, find $\mathbf{x} \in L$ minimizing $\|\mathbf{x} - \mathbf{y}\|$
  - there are more (**uSVP**, **GapSVP**, etc.)
- Also important, approximate variants of these problems:
  - **SVP**$_\gamma$: given L, find $\mathbf{x} \in L \setminus \{0\}$ with $\|\mathbf{x}\| \leq \gamma \lambda_1(L)$
  - **CVP**$_\gamma$: given L and $\mathbf{y} \in \mathbb{Z}^n$, find $\mathbf{x} \in L$ such that $\|\mathbf{x} - \mathbf{y}\| \leq \gamma \cdot d(\mathbf{y}, L)$

©2019 NTT Secure Platform Laboratories

Hardness of lattice problems

- For a full-rank lattice in $\mathbb{Z}^n$, all of these problems are considered to be essentially equally hard for a given approximation factor
- Best algorithm for **SVP** (etc.) is in $2^{\Theta(n)}$
- $\mathbf{SVP}_\gamma$ is NP-hard for $\gamma = n^{O(1/\log\log n)}$. Probably not NP-hard for $\gamma = \mathrm{poly}(n)$, but best known algorithm still exponential
- LLL solves $\mathbf{SVP}_\gamma$ in polynomial time for $\gamma = 2^{n/4}$ (and in practice, for $\gamma \approx 1.1^n$)
- Best known algorithm for $\mathbf{SVP}_\gamma$ is BKZ (see later)

©2019 NTT Secure Platform Laboratories

Hardness of lattice problems

- For a full-rank lattice in $\mathbb{Z}^n$, all of these problems are considered to be essentially equally hard for a given approximation factor
- Best algorithm for **SVP** (etc.) is in $2^{\Theta(n)}$
- $\mathbf{SVP}_\gamma$ is NP-hard for $\gamma = n^{O(1/\log\log n)}$. Probably not NP-hard for $\gamma = \mathrm{poly}(n)$, but best known algorithm still exponential
- LLL solves $\mathbf{SVP}_\gamma$ in polynomial time for $\gamma = 2^{n/4}$ (and in practice, for $\gamma \approx 1.1^n$)
- Best known algorithm for $\mathbf{SVP}_\gamma$ is BKZ (see later)

©2019 NTT Secure Platform Laboratories

Hardness of lattice problems

- For a full-rank lattice in $\mathbb{Z}^n$, all of these problems are considered to be essentially equally hard for a given approximation factor
- Best algorithm for **SVP** (etc.) is in $2^{\Theta(n)}$
- $\mathbf{SVP}_\gamma$ is NP-hard for $\gamma = n^{O(1/\log\log n)}$. Probably not NP-hard for $\gamma = \mathrm{poly}(n)$, but best known algorithm still exponential
- LLL solves $\mathbf{SVP}_\gamma$ in polynomial time for $\gamma = 2^{n/4}$ (and in practice, for $\gamma \approx 1.1^n$)
- Best known algorithm for $\mathbf{SVP}_\gamma$ is BKZ (see later)

©2019 NTT Secure Platform Laboratories

Hardness of lattice problems

- For a full-rank lattice in $\mathbb{Z}^n$, all of these problems are considered to be essentially equally hard for a given approximation factor
- Best algorithm for **SVP** (etc.) is in $2^{\Theta(n)}$
- $\mathbf{SVP}_\gamma$ is NP-hard for $\gamma = n^{O(1/\log\log n)}$. Probably not NP-hard for $\gamma = \mathrm{poly}(n)$, but best known algorithm still exponential
- LLL solves $\mathbf{SVP}_\gamma$ in polynomial time for $\gamma = 2^{n/4}$ (and in practice, for $\gamma \approx 1.1^n$)
- Best known algorithm for $\mathbf{SVP}_\gamma$ is BKZ (see later)

©2019 NTT Secure Platform Laboratories

Hardness of lattice problems

- For a full-rank lattice in $\mathbb{Z}^n$, all of these problems are considered to be essentially equally hard for a given approximation factor
- Best algorithm for **SVP** (etc.) is in $2^{\Theta(n)}$
- $\mathbf{SVP}_\gamma$ is NP-hard for $\gamma = n^{O(1/\log\log n)}$. Probably not NP-hard for $\gamma = \mathrm{poly}(n)$, but best known algorithm still exponential
- LLL solves $\mathbf{SVP}_\gamma$ in polynomial time for $\gamma = 2^{n/4}$ (and in practice, for $\gamma \approx 1.1^n$)
- Best known algorithm for $\mathbf{SVP}_\gamma$ is BKZ (see later)

©2019 NTT Secure Platform Laboratories

Outline

Evaluating the security of lattice-based schemes
- A primer on lattices
- Example: the hardness of SIS
- Methodology for setting parameters

Physical attacks against BLISS
- The BLISS signature scheme
- Fault attack on the Gaussian sampling
- SCA on the rejection sampling

©2019 NTT Secure Platform Laboratories

The SIS problem

- The "integer" version of the lattice problem we discussed last time
- Given a uniformly random $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, find $\mathbf{u} \neq 0$ with small entries such that $\mathbf{Au} \equiv 0 \pmod{q}$
- Clearly related to **Approx-SVP** in the lattice $L = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{Ax} \equiv 0 \pmod{q}\}$
- Proved to be as hard as solving $\mathbf{SIVP}_\gamma$ in any lattice of suitable dimension (worst case to average case reduction)

©2019 NTT Secure Platform Laboratories

The SIS problem

- The "integer" version of the lattice problem we discussed last time
- Given a uniformly random $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, find $\mathbf{u} \neq 0$ with small entries such that $\mathbf{Au} \equiv 0 \pmod{q}$
- Clearly related to **Approx-SVP** in the lattice $L = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{Ax} \equiv 0 \pmod{q}\}$
- Proved to be as hard as solving $\mathbf{SIVP}_\gamma$ in any lattice of suitable dimension (worst case to average case reduction)

©2019 NTT Secure Platform Laboratories

The SIS problem

- The "integer" version of the lattice problem we discussed last time
- Given a uniformly random $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, find $\mathbf{u} \neq 0$ with small entries such that $\mathbf{Au} \equiv 0 \pmod{q}$
- Clearly related to **Approx-SVP** in the lattice $L = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{Ax} \equiv 0 \pmod{q}\}$
- Proved to be as hard as solving $\mathbf{SIVP}_\gamma$ in any lattice of suitable dimension (worst case to average case reduction)

©2019 NTT Secure Platform Laboratories

The SIS problem

- The "integer" version of the lattice problem we discussed last time
- Given a uniformly random $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, find $\mathbf{u} \neq 0$ with small entries such that $\mathbf{Au} \equiv 0 \pmod{q}$
- Clearly related to **Approx-SVP** in the lattice $L = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{Ax} \equiv 0 \pmod{q}\}$
- Proved to be as hard as solving $\mathbf{SIVP}_\gamma$ in any lattice of suitable dimension (worst case to average case reduction)

©2019 NTT Secure Platform Laboratories

How hard is SIS?

- Two approaches to understand the hardness of SIS: use the reduction, or use the best attack
- Using the reduction is the conservative approach (to the extent that we understand the security of $\mathbf{SIVP}_\gamma$), but arguably too conservative (leads to unnecessarily large parameters)
- Using the best attack is normally considered pragmatic (although it can in principle expose to weak keys/unknown attacks)
- In this case, the best attack is using lattice reduction to find a vector $\mathbf{u}$ of length $\leq \beta$ in the lattice $L = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{A}\mathbf{x} \equiv 0 \pmod{q}\}$

©2019 NTT Secure Platform Laboratories

How hard is SIS?

- Two approaches to understand the hardness of SIS: use the reduction, or use the best attack
- Using the reduction is the conservative approach (to the extent that we understand the security of $\mathbf{SIVP}_\gamma$), but arguably too conservative (leads to unnecessarily large parameters)
- Using the best attack is normally considered pragmatic (although it can in principle expose to weak keys/unknown attacks)
- In this case, the best attack is using lattice reduction to find a vector $\mathbf{u}$ of length $\leq \beta$ in the lattice $L = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{A}\mathbf{x} \equiv 0 \pmod{q}\}$

©2019 NTT Secure Platform Laboratories

How hard is SIS?

- Two approaches to understand the hardness of SIS: use the reduction, or use the best attack
- Using the reduction is the conservative approach (to the extent that we understand the security of $\mathbf{SIVP}_\gamma$), but arguably too conservative (leads to unnecessarily large parameters)
- Using the best attack is normally considered pragmatic (although it can in principle expose to weak keys/unknown attacks)
- In this case, the best attack is using lattice reduction to find a vector $\mathbf{u}$ of length $\leq \beta$ in the lattice $L = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{A}\mathbf{x} \equiv 0 \pmod{q}\}$

©2019 NTT Secure Platform Laboratories

How hard is SIS?

- Two approaches to understand the hardness of SIS: use the reduction, or use the best attack
- Using the reduction is the conservative approach (to the extent that we understand the security of $\mathbf{SIVP}_\gamma$), but arguably too conservative (leads to unnecessarily large parameters)
- Using the best attack is normally considered pragmatic (although it can in principle expose to weak keys/unknown attacks)
- In this case, the best attack is using lattice reduction to find a vector $\mathbf{u}$ of length $\leq \beta$ in the lattice $L = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{A}\mathbf{x} \equiv 0 \pmod{q}\}$

©2019 NTT Secure Platform Laboratories

Finding short vectors

- The best known algorithm to find relatively short vectors in a lattice is BKZ
- Roughly speaking, BKZ relies on a subroutine that solves exact-**SVP** in smaller dimension b (the block size), and applies that subroutine poly-many times to recursively reduce a given basis of the lattice
- Ultimately finds vectors of length $\approx \delta_b^n \,\mathrm{vol}(L)^{1/n}$, where the root Hermite factor δ_b depends on the block size b:

$$\delta_b \approx (b/2\pi e)^{1/2b}$$

- The cost of BKZ-b is $\mathrm{poly}(n)$ times the cost of the underlying algorithm for exact-**SVP**. Two choices:
  - enumeration: time $2^{O(b \log b)}$ but polynomial space
  - sieving: time $2^{O(b)}$ but exponential space
- Common conservative estimate: lower bound the cost of BKZ by the cost of one sieving step $\approx 2^{0.292b}$
  - for $\lambda = 128$-bit security, this gives $b \approx 438$ and $\delta_b \approx 1.0037$

©2019 NTT Secure Platform Laboratories

Finding short vectors

- The best known algorithm to find relatively short vectors in a lattice is BKZ
- Roughly speaking, BKZ relies on a subroutine that solves exact-**SVP** in smaller dimension b (the block size), and applies that subroutine poly-many times to recursively reduce a given basis of the lattice
- Ultimately finds vectors of length $\approx \delta_b^n \operatorname{vol}(L)^{1/n}$, where the root Hermite factor δ_b depends on the block size b:

$$\delta_b \approx (b/2\pi e)^{1/2b}$$

- The cost of BKZ-b is poly(n) times the cost of the underlying algorithm for exact-**SVP**. Two choices:
  - enumeration: time $2^{O(b \log b)}$ but polynomial space
  - sieving: time $2^{O(b)}$ but exponential space
- Common conservative estimate: lower bound the cost of BKZ by the cost of one sieving step $\approx 2^{0.292b}$
  - for $\lambda = 128$-bit security, this gives $b \approx 438$ and $\delta_b \approx 1.0037$

©2019 NTT Secure Platform Laboratories

Finding short vectors

- The best known algorithm to find relatively short vectors in a lattice is BKZ
- Roughly speaking, BKZ relies on a subroutine that solves exact-**SVP** in smaller dimension b (the block size), and applies that subroutine poly-many times to recursively reduce a given basis of the lattice
- Ultimately finds vectors of length $\approx \delta_b^n \,\mathrm{vol}(L)^{1/n}$, where the root Hermite factor δ_b depends on the block size b:

$$\delta_b \approx (b/2\pi e)^{1/2b}$$

- The cost of BKZ-b is poly(n) times the cost of the underlying algorithm for exact-**SVP**. Two choices:
  - enumeration: time $2^{O(b \log b)}$ but polynomial space
  - sieving: time $2^{O(b)}$ but exponential space
- Common conservative estimate: lower bound the cost of BKZ by the cost of one sieving step $\approx 2^{0.292b}$
  - for $\lambda = 128$-bit security, this gives $b \approx 438$ and $\delta_b \approx 1.0037$

©2019 NTT Secure Platform Laboratories

Finding short vectors

- The best known algorithm to find relatively short vectors in a lattice is BKZ
- Roughly speaking, BKZ relies on a subroutine that solves exact-**SVP** in smaller dimension b (the block size), and applies that subroutine poly-many times to recursively reduce a given basis of the lattice
- Ultimately finds vectors of length $\approx \delta_b^n \,\mathrm{vol}(L)^{1/n}$, where the root Hermite factor δ_b depends on the block size b:

$$\delta_b \approx (b/2\pi e)^{1/2b}$$

- The cost of BKZ-b is poly(n) times the cost of the underlying algorithm for exact-**SVP**. Two choices:
  - enumeration: time $2^{O(b \log b)}$ but polynomial space
  - sieving: time $2^{O(b)}$ but exponential space
- Common conservative estimate: lower bound the cost of BKZ by the cost of one sieving step $\approx 2^{0.292b}$
  - for $\lambda = 128$-bit security, this gives $b \approx 438$ and $\delta_b \approx 1.0037$

©2019 NTT Secure Platform Laboratories

Finding short vectors

- The best known algorithm to find relatively short vectors in a lattice is BKZ
- Roughly speaking, BKZ relies on a subroutine that solves exact-**SVP** in smaller dimension b (the block size), and applies that subroutine poly-many times to recursively reduce a given basis of the lattice
- Ultimately finds vectors of length $\approx \delta_b^n \operatorname{vol}(L)^{1/n}$, where the root Hermite factor δ_b depends on the block size b:

$$\delta_b \approx (b/2\pi e)^{1/2b}$$

- The cost of BKZ-b is poly(n) times the cost of the underlying algorithm for exact-**SVP**. Two choices:
  - enumeration: time $2^{O(b \log b)}$ but polynomial space
  - sieving: time $2^{O(b)}$ but exponential space
- Common conservative estimate: lower bound the cost of BKZ by the cost of one sieving step $\approx 2^{0.292b}$
  - for $\lambda = 128$-bit security, this gives $b \approx 438$ and $\delta_b \approx 1.0037$

©2019 NTT Secure Platform Laboratories

Condition for SIS security

- Recall that the SIS problem is to find $\mathbf{u}$ of norm $\leq \beta$ in the lattice $L = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{A}\mathbf{x} \equiv 0 \pmod{q}\}$
- Feasible with BKZ-b if $\beta \gtrsim \delta_b^n \cdot \mathrm{vol}(L)^{1/n}$
- Now $\mathrm{vol}(L) = [\mathbb{Z}^n : L] = q^m$ (if $\mathbf{A}$ is of maximum rank m, which happens w.h.p.)
- Hence, for security, q, m, n and β are linked by the relation:

$$\beta \ll \delta_b^n \cdot q^{m/n}$$

with δ_b set by the security level

- Note that the expected size of the shortest vector is $\approx q^{m/n}$, and β can only be larger by a factor δ_b^n. The dimension n needs to be large for this factor to leave sufficient slack
  - e.g. for $\delta_b \approx 1.0037$, we have $\delta_b^{1024} \approx 44$

©2019 NTT Secure Platform Laboratories

Condition for SIS security

- Recall that the SIS problem is to find $\mathbf{u}$ of norm $\leq \beta$ in the lattice $L = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{A}\mathbf{x} \equiv 0 \pmod{q}\}$
- Feasible with BKZ-b if $\beta \gtrsim \delta_b^n \cdot \mathrm{vol}(L)^{1/n}$
- Now $\mathrm{vol}(L) = [\mathbb{Z}^n : L] = q^m$ (if $\mathbf{A}$ is of maximum rank m, which happens w.h.p.)
- Hence, for security, q, m, n and β are linked by the relation:

$$\beta \ll \delta_b^n \cdot q^{m/n}$$

  with δ_b set by the security level
- Note that the expected size of the shortest vector is $\approx q^{m/n}$, and β can only be larger by a factor δ_b^n. The dimension n needs to be large for this factor to leave sufficient slack
  - e.g. for $\delta_b \approx 1.0037$, we have $\delta_b^{1024} \approx 44$

©2019 NTT Secure Platform Laboratories

Condition for SIS security

- Recall that the SIS problem is to find $\mathbf{u}$ of norm $\leq \beta$ in the lattice $L = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{A}\mathbf{x} \equiv 0 \pmod{q}\}$
- Feasible with BKZ-b if $\beta \gtrsim \delta_b^n \cdot \mathrm{vol}(L)^{1/n}$
- Now $\mathrm{vol}(L) = [\mathbb{Z}^n : L] = q^m$ (if $\mathbf{A}$ is of maximum rank m, which happens w.h.p.)
- Hence, for security, q, m, n and β are linked by the relation:

$$\beta \ll \delta_b^n \cdot q^{m/n}$$

with δ_b set by the security level

- Note that the expected size of the shortest vector is $\approx q^{m/n}$, and β can only be larger by a factor δ_b^n. The dimension n needs to be large for this factor to leave sufficient slack
  - e.g. for $\delta_b \approx 1.0037$, we have $\delta_b^{1024} \approx 44$

©2019 NTT Secure Platform Laboratories

Condition for SIS security

- Recall that the SIS problem is to find $\mathbf{u}$ of norm $\leq \beta$ in the lattice $L = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{A}\mathbf{x} \equiv 0 \pmod{q}\}$
- Feasible with BKZ-b if $\beta \gtrsim \delta_b^n \cdot \mathrm{vol}(L)^{1/n}$
- Now $\mathrm{vol}(L) = [\mathbb{Z}^n : L] = q^m$ (if $\mathbf{A}$ is of maximum rank m, which happens w.h.p.)
- Hence, for security, q, m, n and β are linked by the relation:

$$\beta \ll \delta_b^n \cdot q^{m/n}$$

with δ_b set by the security level

- Note that the expected size of the shortest vector is $\approx q^{m/n}$, and β can only be larger by a factor δ_b^n. The dimension n needs to be large for this factor to leave sufficient slack
  - e.g. for $\delta_b \approx 1.0037$, we have $\delta_b^{1024} \approx 44$

©2019 NTT Secure Platform Laboratories

Condition for SIS security

- Recall that the SIS problem is to find $\mathbf{u}$ of norm $\leq \beta$ in the lattice $L = \{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{A}\mathbf{x} \equiv 0 \pmod{q}\}$
- Feasible with BKZ-b if $\beta \gtrsim \delta_b^n \cdot \mathrm{vol}(L)^{1/n}$
- Now $\mathrm{vol}(L) = [\mathbb{Z}^n : L] = q^m$ (if $\mathbf{A}$ is of maximum rank m, which happens w.h.p.)
- Hence, for security, q, m, n and β are linked by the relation:

$$\beta \ll \delta_b^n \cdot q^{m/n}$$

  with δ_b set by the security level
- Note that the expected size of the shortest vector is $\approx q^{m/n}$, and β can only be larger by a factor δ_b^n. The dimension n needs to be large for this factor to leave sufficient slack
  - e.g. for $\delta_b \approx 1.0037$, we have $\delta_b^{1024} \approx 44$

©2019 NTT Secure Platform Laboratories

Outline

Evaluating the security of lattice-based schemes
- A primer on lattices
- Example: the hardness of SIS
- Methodology for setting parameters

Physical attacks against BLISS
- The BLISS signature scheme
- Fault attack on the Gaussian sampling
- SCA on the rejection sampling

©2019 NTT Secure Platform Laboratories

Applying Fiat–Shamir

Recall last time's scheme based on Module-SIS (below). How would we go about setting secure parameters?

- KeyGen(1^λ): set R and the parameters $m, \alpha, \kappa, \sigma$ according to λ. H is a random oracle $\{0,1\}^* \times R \to D_c$. Sample $\hat{\mathbf{s}}$ as the signing key, h and $\mathbf{S} = h(\hat{\mathbf{s}})$ as the verification key.
- Sign$(\hat{\mathbf{s}}, h, \mu)$:
  - sample $\hat{\mathbf{y}}$ and let $\mathbf{Y} = h(\hat{\mathbf{y}})$
  - let $\mathbf{c} = H(\mu, \mathbf{Y})$ and $\hat{\mathbf{z}} = \hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}}$
  - if $\|\hat{\mathbf{z}}\| > \alpha - \sigma\kappa$ restart, else output sig $= (\mathbf{Y}, \hat{\mathbf{z}})$
- Verify$\big(h, \mathbf{S}, \mu, \text{sig} = (\mathbf{Y}, \hat{\mathbf{z}})\big)$:
  - compute $\mathbf{c} = H(\mu, \mathbf{Y})$
  - accept iff $\|\hat{\mathbf{z}}\| \leq \alpha - \sigma\kappa$ and $h(\hat{\mathbf{z}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$

©2019 NTT Secure Platform Laboratories

Applying Fiat–Shamir

Recall last time's scheme based on Module-SIS (below). How would we go about setting secure parameters?

- KeyGen(1^λ): set R and the parameters $m, \alpha, \kappa, \sigma$ according to λ. H is a random oracle $\{0,1\}^* \times R \to D_c$. Sample $\hat{\mathbf{s}}$ as the signing key, h and $\mathbf{S} = h(\hat{\mathbf{s}})$ as the verification key.
- Sign$(\hat{\mathbf{s}}, h, \mu)$:
  - sample $\hat{\mathbf{y}}$ and let $\mathbf{Y} = h(\hat{\mathbf{y}})$
  - let $\mathbf{c} = H(\mu, \mathbf{Y})$ and $\hat{\mathbf{z}} = \hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}}$
  - if $\|\hat{\mathbf{z}}\| > \alpha - \sigma\kappa$ restart, else output sig $= (\mathbf{Y}, \hat{\mathbf{z}})$
- Verify$\big(h, \mathbf{S}, \mu, \text{sig} = (\mathbf{Y}, \hat{\mathbf{z}})\big)$:
  - compute $\mathbf{c} = H(\mu, \mathbf{Y})$
  - accept iff $\|\hat{\mathbf{z}}\| \leq \alpha - \sigma\kappa$ and $h(\hat{\mathbf{z}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$

©2019 NTT Secure Platform Laboratories

Applying Fiat–Shamir

Recall last time's scheme based on Module-SIS (below). How would we go about setting secure parameters?

- KeyGen(1^λ): set R and the parameters $m, \alpha, \kappa, \sigma$ according to λ. H is a random oracle $\{0,1\}^* \times R \to D_c$. Sample $\hat{\mathbf{s}}$ as the signing key, h and $\mathbf{S} = h(\hat{\mathbf{s}})$ as the verification key.
- Sign$(\hat{\mathbf{s}}, h, \mu)$:
  - sample $\hat{\mathbf{y}}$ and let $\mathbf{Y} = h(\hat{\mathbf{y}})$
  - let $\mathbf{c} = H(\mu, \mathbf{Y})$ and $\hat{\mathbf{z}} = \hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}}$
  - if $\|\hat{\mathbf{z}}\| > \alpha - \sigma\kappa$ restart, else output sig $= (\mathbf{Y}, \hat{\mathbf{z}})$
- Verify$\big(h, \mathbf{S}, \mu, \text{sig} = (\mathbf{Y}, \hat{\mathbf{z}})\big)$:
  - compute $\mathbf{c} = H(\mu, \mathbf{Y})$
  - accept iff $\|\hat{\mathbf{z}}\| \leq \alpha - \sigma\kappa$ and $h(\hat{\mathbf{z}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$

©2019 NTT Secure Platform Laboratories

Constraints on the parameters

- Multiple constraints on the parameters needed for having a secure scheme with an efficient signing algorithm
- The range of the random oracle H should be large
  - $\binom{n}{\kappa} \geq 2^{\lambda}$
- A public key $\mathbf{S}$ needs to have exponentially many associated secret keys $\hat{\mathbf{s}}$: i.e. the set of secret keys should be much larger than the range of h
  - $(2\sigma+1)^{mn} \geq 2^{\lambda} \cdot q^{n}$
- The rejection probability should be constant
  - $(1-\sigma\kappa/\alpha)^{nm} = \Omega(1)$, hence $\alpha = \Omega(nm \cdot \sigma\kappa)$
- Module-SIS should be hard up to infinity norm $2(\alpha+\kappa\sigma)$ to prevent the collisions on h found in the security reduction
  - see next slide

©2019 NTT Secure Platform Laboratories

Constraints on the parameters

- Multiple constraints on the parameters needed for having a secure scheme with an efficient signing algorithm
- The range of the random oracle H should be large
  - $\binom{n}{\kappa} \geq 2^{\lambda}$
- A public key $\mathbf{S}$ needs to have exponentially many associated secret keys $\hat{\mathbf{s}}$: i.e. the set of secret keys should be much larger than the range of h
  - $(2\sigma+1)^{mn} \geq 2^{\lambda} \cdot q^{n}$
- The rejection probability should be constant
  - $(1-\sigma\kappa/\alpha)^{nm} = \Omega(1)$, hence $\alpha = \Omega(nm \cdot \sigma\kappa)$
- Module-SIS should be hard up to infinity norm $2(\alpha+\kappa\sigma)$ to prevent the collisions on h found in the security reduction
  - see next slide

©2019 NTT Secure Platform Laboratories

Constraints on the parameters

- Multiple constraints on the parameters needed for having a secure scheme with an efficient signing algorithm
- The range of the random oracle H should be large
  - $\binom{n}{\kappa} \geq 2^{\lambda}$
- A public key $\mathbf{S}$ needs to have exponentially many associated secret keys $\hat{\mathbf{s}}$: i.e. the set of secret keys should be much larger than the range of h
  - $(2\sigma+1)^{mn} \geq 2^{\lambda} \cdot q^{n}$
- The rejection probability should be constant
  - $(1-\sigma\kappa/\alpha)^{nm} = \Omega(1)$, hence $\alpha = \Omega(nm \cdot \sigma\kappa)$
- Module-SIS should be hard up to infinity norm $2(\alpha+\kappa\sigma)$ to prevent the collisions on h found in the security reduction
  - see next slide

©2019 NTT Secure Platform Laboratories

Constraints on the parameters

- Multiple constraints on the parameters needed for having a secure scheme with an efficient signing algorithm
- The range of the random oracle H should be large
  - $\binom{n}{\kappa} \geq 2^{\lambda}$
- A public key $\mathbf{S}$ needs to have exponentially many associated secret keys $\hat{\mathbf{s}}$: i.e. the set of secret keys should be much larger than the range of h
  - $(2\sigma+1)^{mn} \geq 2^{\lambda} \cdot q^{n}$
- The rejection probability should be constant
  - $(1-\sigma\kappa/\alpha)^{nm} = \Omega(1)$, hence $\alpha = \Omega(nm \cdot \sigma\kappa)$
- Module-SIS should be hard up to infinity norm $2(\alpha+\kappa\sigma)$ to prevent the collisions on h found in the security reduction
  - see next slide

©2019 NTT Secure Platform Laboratories

Constraints on the parameters

- Multiple constraints on the parameters needed for having a secure scheme with an efficient signing algorithm
- The range of the random oracle H should be large
  - $\binom{n}{\kappa} \geq 2^{\lambda}$
- A public key $\mathbf{S}$ needs to have exponentially many associated secret keys $\hat{\mathbf{s}}$: i.e. the set of secret keys should be much larger than the range of h
  - $(2\sigma+1)^{mn} \geq 2^{\lambda} \cdot q^{n}$
- The rejection probability should be constant
  - $(1-\sigma\kappa/\alpha)^{nm} = \Omega(1)$, hence $\alpha = \Omega(nm \cdot \sigma\kappa)$
- Module-SIS should be hard up to infinity norm $2(\alpha+\kappa\sigma)$ to prevent the collisions on h found in the security reduction
  - see next slide

©2019 NTT Secure Platform Laboratories

Module-SIS hardness

- As far as we know, the best attack on Module-SIS does not use the module structure: same attack as regular SIS!
  - matrix dimension is now $n \times (nm)$
  - sorry for the confusing notation!
- By the previous section, security is achieved for a Euclidean norm bound β such that:

$$\beta \ll \delta_b^{nm} \cdot q^{1/m}$$

  with $\delta_b = 1.0037$ for 128-bit security
- In our case, β is $\sqrt{nm}$ times the bound in infinity norm, hence $\beta = 2(\alpha + \kappa\sigma)\sqrt{nm}$

©2019 NTT Secure Platform Laboratories

Module-SIS hardness

- As far as we know, the best attack on Module-SIS does not use the module structure: same attack as regular SIS!
  - matrix dimension is now $n \times (nm)$
  - sorry for the confusing notation!
- By the previous section, security is achieved for a Euclidean norm bound β such that:

$$\beta \ll \delta_b^{nm} \cdot q^{1/m}$$

with $\delta_b = 1.0037$ for 128-bit security

- In our case, β is $\sqrt{nm}$ times the bound in infinity norm, hence $\beta = 2(\alpha + \kappa\sigma)\sqrt{nm}$

©2019 NTT Secure Platform Laboratories

Module-SIS hardness

- As far as we know, the best attack on Module-SIS does not use the module structure: same attack as regular SIS!
  - matrix dimension is now $n \times (nm)$
  - sorry for the confusing notation!
- By the previous section, security is achieved for a Euclidean norm bound β such that:

$$\beta \ll \delta_b^{nm} \cdot q^{1/m}$$

  with $\delta_b = 1.0037$ for 128-bit security
- In our case, β is $\sqrt{nm}$ times the bound in infinity norm, hence $\beta = 2(\alpha + \kappa\sigma)\sqrt{nm}$

©2019 NTT Secure Platform Laboratories

Concrete parameter choice (I)

- To fix idea, set for example $n = 512$, $\sigma = 1$. Let's try to set parameters for $\lambda = 128$-bit security
- To get $\binom{n}{\kappa} \geq 2^{128}$, we pick $\kappa = 23$
- To get an acceptance probability $\approx 1/e$ (2.7 repetitions per sig. on average), we set $\alpha = (nm-1) \cdot (\kappa\sigma) = 23(nm-1)$
- This yields $\beta = 2\kappa\sigma \cdot (nm)^{3/2} = 46(nm)^{3/2}$
- Moreover, to satisfy $(2\alpha+1)^{nm} \geq 2^{\lambda} \cdot q^{n}$, we will pick $q^{1/m}$ slightly smaller than $2\alpha + 1 = 3$
- Use the hardness of Module-SIS to find m

©2019 NTT Secure Platform Laboratories

Concrete parameter choice (I)

- To fix idea, set for example $n = 512$, $\sigma = 1$. Let's try to set parameters for $\lambda = 128$-bit security
- To get $\binom{n}{\kappa} \geq 2^{128}$, we pick $\kappa = 23$
- To get an acceptance probability $\approx 1/e$ (2.7 repetitions per sig. on average), we set $\alpha = (nm - 1) \cdot (\kappa\sigma) = 23(nm - 1)$
- This yields $\beta = 2\kappa\sigma \cdot (nm)^{3/2} = 46(nm)^{3/2}$
- Moreover, to satisfy $(2\alpha + 1)^{nm} \geq 2^{\lambda} \cdot q^{n}$, we will pick $q^{1/m}$ slightly smaller than $2\alpha + 1 = 3$
- Use the hardness of Module-SIS to find m

©2019 NTT Secure Platform Laboratories

Concrete parameter choice (I)

- To fix idea, set for example $n = 512$, $\sigma = 1$. Let's try to set parameters for $\lambda = 128$-bit security
- To get $\binom{n}{\kappa} \geq 2^{128}$, we pick $\kappa = 23$
- To get an acceptance probability $\approx 1/e$ (2.7 repetitions per sig. on average), we set $\alpha = (nm - 1) \cdot (\kappa\sigma) = 23(nm - 1)$
- This yields $\beta = 2\kappa\sigma \cdot (nm)^{3/2} = 46(nm)^{3/2}$
- Moreover, to satisfy $(2\alpha + 1)^{nm} \geq 2^{\lambda} \cdot q^n$, we will pick $q^{1/m}$ slightly smaller than $2\alpha + 1 = 3$
- Use the hardness of Module-SIS to find m

©2019 NTT Secure Platform Laboratories

Concrete parameter choice (I)

- To fix idea, set for example $n = 512$, $\sigma = 1$. Let's try to set parameters for $\lambda = 128$-bit security
- To get $\binom{n}{\kappa} \geq 2^{128}$, we pick $\kappa = 23$
- To get an acceptance probability $\approx 1/e$ (2.7 repetitions per sig. on average), we set $\alpha = (nm - 1) \cdot (\kappa\sigma) = 23(nm - 1)$
- This yields $\beta = 2\kappa\sigma \cdot (nm)^{3/2} = 46(nm)^{3/2}$
- Moreover, to satisfy $(2\alpha + 1)^{nm} \geq 2^{\lambda} \cdot q^{n}$, we will pick $q^{1/m}$ slightly smaller than $2\alpha + 1 = 3$
- Use the hardness of Module-SIS to find m

©2019 NTT Secure Platform Laboratories

Concrete parameter choice (I)

- To fix idea, set for example $n = 512$, $\sigma = 1$. Let's try to set parameters for $\lambda = 128$-bit security
- To get $\binom{n}{\kappa} \geq 2^{128}$, we pick $\kappa = 23$
- To get an acceptance probability $\approx 1/e$ (2.7 repetitions per sig. on average), we set $\alpha = (nm - 1) \cdot (\kappa\sigma) = 23(nm - 1)$
- This yields $\beta = 2\kappa\sigma \cdot (nm)^{3/2} = 46(nm)^{3/2}$
- Moreover, to satisfy $(2\alpha + 1)^{nm} \geq 2^{\lambda} \cdot q^{n}$, we will pick $q^{1/m}$ slightly smaller than $2\alpha + 1 = 3$
- Use the hardness of Module-SIS to find m

©2019 NTT Secure Platform Laboratories

Concrete parameter choice (I)

- To fix idea, set for example $n = 512$, $\sigma = 1$. Let's try to set parameters for $\lambda = 128$-bit security
- To get $\binom{n}{\kappa} \geq 2^{128}$, we pick $\kappa = 23$
- To get an acceptance probability $\approx 1/e$ (2.7 repetitions per sig. on average), we set $\alpha = (nm - 1) \cdot (\kappa\sigma) = 23(nm - 1)$
- This yields $\beta = 2\kappa\sigma \cdot (nm)^{3/2} = 46(nm)^{3/2}$
- Moreover, to satisfy $(2\alpha + 1)^{nm} \geq 2^{\lambda} \cdot q^{n}$, we will pick $q^{1/m}$ slightly smaller than $2\alpha + 1 = 3$
- Use the hardness of Module-SIS to find m

©2019 NTT Secure Platform Laboratories

Concrete parameter choice (II)

- The hardness condition says:

$$\delta_b^{nm} \gg \frac{\beta}{q^{1/m}} \approx \frac{46}{3}(nm)^{3/2}$$

  Letting $d = nm$ and taking logs, we get:

$$d \log \delta_b \gtrsim \frac{3}{2} \log d + \log(46/3)$$

- Numerically solving for d, we get $d \gtrsim 4110$, hence $m \geq 9$
- However, we also need $92nm = 4(\alpha + \kappa\sigma) < q < 3^m \cdot 2^{-\lambda/n}$, which imposes $m \geq 13$
- The choice $q = 3 \cdot 2^{18} + 1$ and $m = 13$ is consistent with these constraints

©2019 NTT Secure Platform Laboratories

Concrete parameter choice (II)

- The hardness condition says:

$$\delta_b^{nm} \gg \frac{\beta}{q^{1/m}} \approx \frac{46}{3}(nm)^{3/2}$$

  Letting $d = nm$ and taking logs, we get:

$$d \log \delta_b \gtrsim \frac{3}{2} \log d + \log(46/3)$$

- Numerically solving for d, we get $d \gtrsim 4110$, hence $m \geq 9$
- However, we also need $92nm = 4(\alpha + \kappa\sigma) < q < 3^m \cdot 2^{-\lambda/n}$, which imposes $m \geq 13$
- The choice $q = 3 \cdot 2^{18} + 1$ and $m = 13$ is consistent with these constraints

©2019 NTT Secure Platform Laboratories

Concrete parameter choice (II)

- The hardness condition says:

$$\delta_b^{nm} \gg \frac{\beta}{q^{1/m}} \approx \frac{46}{3}(nm)^{3/2}$$

  Letting $d = nm$ and taking logs, we get:

$$d \log \delta_b \gtrsim \frac{3}{2} \log d + \log(46/3)$$

- Numerically solving for d, we get $d \gtrsim 4110$, hence $m \geq 9$
- However, we also need $92nm = 4(\alpha + \kappa\sigma) < q < 3^m \cdot 2^{-\lambda/n}$, which imposes $m \geq 13$
- The choice $q = 3 \cdot 2^{18} + 1$ and $m = 13$ is consistent with these constraints

©2019 NTT Secure Platform Laboratories

Concrete parameter choice (II)

- The hardness condition says:

$$\delta_b^{nm} \gg \frac{\beta}{q^{1/m}} \approx \frac{46}{3}(nm)^{3/2}$$

  Letting $d = nm$ and taking logs, we get:

$$d \log \delta_b \gtrsim \frac{3}{2} \log d + \log(46/3)$$

- Numerically solving for d, we get $d \gtrsim 4110$, hence $m \geq 9$
- However, we also need $92nm = 4(\alpha + \kappa\sigma) < q < 3^m \cdot 2^{-\lambda/n}$, which imposes $m \geq 13$
- The choice $q = 3 \cdot 2^{18} + 1$ and $m = 13$ is consistent with these constraints

©2019 NTT Secure Platform Laboratories

Concrete parameter choice (III)

- Finally, we select:

$$n = 512 \qquad \sigma = 1 \qquad \kappa = 23$$
$$m = 13 \qquad q = 3 \cdot 2^{18} + 1 \qquad \alpha = 153065$$

- Resulting sizes:
  - sk size: 0.1 kB
  - vk size: 18 kB
  - sig. size: 15 kB
  - repetition rate: 2.72
- One could choose slightly different trade-offs, but probably no significant improvement possible without changing the scheme

©2019 NTT Secure Platform Laboratories

Concrete parameter choice (III)

- Finally, we select:

$$n = 512 \qquad \sigma = 1 \qquad \kappa = 23$$
$$m = 13 \qquad q = 3 \cdot 2^{18} + 1 \qquad \alpha = 153065$$

- Resulting sizes:
  - sk size: 0.1 kB
  - vk size: 18 kB
  - sig. size: 15 kB
  - repetition rate: 2.72
- One could choose slightly different trade-offs, but probably no significant improvement possible without changing the scheme

©2019 NTT Secure Platform Laboratories

Concrete parameter choice (III)

- Finally, we select:

$$n = 512 \qquad \sigma = 1 \qquad \kappa = 23$$
$$m = 13 \qquad q = 3 \cdot 2^{18} + 1 \qquad \alpha = 153065$$

- Resulting sizes:
  - sk size: 0.1 kB
  - vk size: 18 kB
  - sig. size: 15 kB
  - repetition rate: 2.72
- One could choose slightly different trade-offs, but probably no significant improvement possible without changing the scheme

©2019 NTT Secure Platform Laboratories

Outline

Evaluating the security of lattice-based schemes
- A primer on lattices
- Example: the hardness of SIS
- Methodology for setting parameters

Physical attacks against BLISS
- The BLISS signature scheme
- Fault attack on the Gaussian sampling
- SCA on the rejection sampling

©2019 NTT Secure Platform Laboratories

Black-box vs real-world security

- Recall the security definition of digital signatures
- Traditional, "black-box" view of security:
  - the attacker, Alice, interacts with the signer, Bob
  - Alice sends Bob messages to sign, only gets the results of Bob's computation (no other info about the computation is revealed)
  - based on that, Alice tries to forge new signatures/extract info about Bob's signing key
- Real-world security:
  - Bob is actually a smart card, say
  - Alice can measure all sorts of emanation from the card as it operates, or mess with it in various ways
  - all that extra information can be useful to break things!

©2019 NTT Secure Platform Laboratories

Black-box vs real-world security

- Recall the security definition of digital signatures
- Traditional, "black-box" view of security:
  - the attacker, Alice, interacts with the signer, Bob
  - Alice sends Bob messages to sign, only gets the results of Bob's computation (no other info about the computation is revealed)
  - based on that, Alice tries to forge new signatures/extract info about Bob's signing key
- Real-world security:
  - Bob is actually a smart card, say
  - Alice can measure all sorts of emanation from the card as it operates, or mess with it in various ways
  - all that extra information can be useful to break things!

©2019 NTT Secure Platform Laboratories

Black-box vs real-world security

- Recall the security definition of digital signatures
- Traditional, "black-box" view of security:
  - the attacker, Alice, interacts with the signer, Bob
  - Alice sends Bob messages to sign, only gets the results of Bob's computation (no other info about the computation is revealed)
  - based on that, Alice tries to forge new signatures/extract info about Bob's signing key
- Real-world security:
  - Bob is actually a smart card, say
  - Alice can measure all sorts of emanation from the card as it operates, or mess with it in various ways
  - all that extra information can be useful to break things!

©2019 NTT Secure Platform Laboratories

Implementation attacks

- The security guarantees offered by "security proofs" for lattice-based crypto are in the black-box model
- But to break a real-world crypto implementation, no need to play by the rules of that model
- Passive physical attack: measure the side-channel leakage of an implementation of lattice-based signatures, and use it together with a little bit of number theory to recover the entire key
  - specifically, electromagnetic emanations
  - it would also work with power consumption, etc.
- Active physical attack: inject faults that tamper with the device during the computation in order to make key recovery possible
  - voltage spikes, clock glitches, etc.

©2019 NTT Secure Platform Laboratories

Implementation attacks

- The security guarantees offered by "security proofs" for lattice-based crypto are in the black-box model
- But to break a real-world crypto implementation, no need to play by the rules of that model
- Passive physical attack: measure the side-channel leakage of an implementation of lattice-based signatures, and use it together with a little bit of number theory to recover the entire key
  - specifically, electromagnetic emanations
  - it would also work with power consumption, etc.
- Active physical attack: inject faults that tamper with the device during the computation in order to make key recovery possible
  - voltage spikes, clock glitches, etc.

©2019 NTT Secure Platform Laboratories

Implementation attacks

- The security guarantees offered by "security proofs" for lattice-based crypto are in the black-box model
- But to break a real-world crypto implementation, no need to play by the rules of that model
- Passive physical attack: measure the side-channel leakage of an implementation of lattice-based signatures, and use it together with a little bit of number theory to recover the entire key
  - specifically, electromagnetic emanations
  - it would also work with power consumption, etc.
- Active physical attack: inject faults that tamper with the device during the computation in order to make key recovery possible
  - voltage spikes, clock glitches, etc.

©2019 NTT Secure Platform Laboratories

Implementation attacks

- The security guarantees offered by "security proofs" for lattice-based crypto are in the black-box model
- But to break a real-world crypto implementation, no need to play by the rules of that model
- Passive physical attack: measure the side-channel leakage of an implementation of lattice-based signatures, and use it together with a little bit of number theory to recover the entire key
  - specifically, electromagnetic emanations
  - it would also work with power consumption, etc.
- Active physical attack: inject faults that tamper with the device during the computation in order to make key recovery possible
  - voltage spikes, clock glitches, etc.

©2019 NTT Secure Platform Laboratories

Outline

©2019 NTT Secure Platform Laboratories

BLISS: the basics

- Possibly the most efficient lattice-based signature proposed so far
- Introduced by Ducas, Durmus, Lepoint and Lyubashevsky at CRYPTO'13
- Implementations on various platforms: desktop computers, microcontrollers/smartcards, FPGAs
- Deployed in the VPN library strongSwan
- Not a NIST candidate
  - authors cite concerns regarding side-channel attacks
  - parameter choice less conservative than most NIST lattice-based candidates

©2019 NTT Secure Platform Laboratories

BLISS: the basics

- Possibly the most efficient lattice-based signature proposed so far
- Introduced by Ducas, Durmus, Lepoint and Lyubashevsky at CRYPTO'13
- Implementations on various platforms: desktop computers, microcontrollers/smartcards, FPGAs
- Deployed in the VPN library strongSwan
- Not a NIST candidate
  - authors cite concerns regarding side-channel attacks
  - parameter choice less conservative than most NIST lattice-based candidates

©2019 NTT Secure Platform Laboratories

BLISS: the basics

- Possibly the most efficient lattice-based signature proposed so far
- Introduced by Ducas, Durmus, Lepoint and Lyubashevsky at CRYPTO'13
- Implementations on various platforms: desktop computers, microcontrollers/smartcards, FPGAs
- Deployed in the VPN library strongSwan
- Not a NIST candidate
  - authors cite concerns regarding side-channel attacks
  - parameter choice less conservative than most NIST lattice-based candidates

©2019 NTT Secure Platform Laboratories

BLISS: the basics

- Possibly the most efficient lattice-based signature proposed so far
- Introduced by Ducas, Durmus, Lepoint and Lyubashevsky at CRYPTO'13
- Implementations on various platforms: desktop computers, microcontrollers/smartcards, FPGAs
- Deployed in the VPN library strongSwan
- Not a NIST candidate
  - authors cite concerns regarding side-channel attacks
  - parameter choice less conservative than most NIST lattice-based candidates

©2019 NTT Secure Platform Laboratories

BLISS: the basics

- Possibly the most efficient lattice-based signature proposed so far
- Introduced by Ducas, Durmus, Lepoint and Lyubashevsky at CRYPTO'13
- Implementations on various platforms: desktop computers, microcontrollers/smartcards, FPGAs
- Deployed in the VPN library strongSwan
- Not a NIST candidate
  - authors cite concerns regarding side-channel attacks
  - parameter choice less conservative than most NIST lattice-based candidates

©2019 NTT Secure Platform Laboratories

BLISS: signing and verification keys

- Works in the cyclotomic ring $R = \mathbb{Z}[\mathbf{x}]/(x^n + 1)$, $n = 512$
- Computations modulo the prime $q = 12289$
- Secret key: random sparse $\mathbf{s}_1, \mathbf{s}_2 \in R$ with coefficients in $\{-1, 0, 1\}$
- Verification key: $\mathbf{a} = -\mathbf{s}_2/\mathbf{s}_1 \bmod q$
  - restart if $\mathbf{s}_1$ not invertible

©2019 NTT Secure Platform Laboratories

BLISS: signing and verification keys

- Works in the cyclotomic ring $R = \mathbb{Z}[\mathbf{x}]/(x^n + 1)$, $n = 512$
- Computations modulo the prime $q = 12289$
- Secret key: random sparse $\mathbf{s}_1, \mathbf{s}_2 \in R$ with coefficients in $\{-1, 0, 1\}$
- Verification key: $\mathbf{a} = -\mathbf{s}_2/\mathbf{s}_1 \bmod q$
  - restart if $\mathbf{s}_1$ not invertible

©2019 NTT Secure Platform Laboratories

BLISS: signing and verification keys

- Works in the cyclotomic ring $R = \mathbb{Z}[\mathbf{x}]/(x^n + 1)$, $n = 512$
- Computations modulo the prime $q = 12289$
- Secret key: random sparse $\mathbf{s}_1, \mathbf{s}_2 \in R$ with coefficients in $\{-1, 0, 1\}$
- Verification key: $\mathbf{a} = -\mathbf{s}_2/\mathbf{s}_1 \bmod q$
  - restart if $\mathbf{s}_1$ not invertible

©2019 NTT Secure Platform Laboratories

BLISS: signing and verification keys

- Works in the cyclotomic ring $R = \mathbb{Z}[\mathbf{x}]/(x^n + 1)$, $n = 512$
- Computations modulo the prime $q = 12289$
- Secret key: random sparse $\mathbf{s}_1, \mathbf{s}_2 \in R$ with coefficients in $\{-1, 0, 1\}$
- Verification key: $\mathbf{a} = -\mathbf{s}_2/\mathbf{s}_1 \bmod q$
  - restart if $\mathbf{s}_1$ not invertible

©2019 NTT Secure Platform Laboratories

BLISS: signature (simplified)

1: **function** $\textsc{Sign}(\mu, pk = \mathbf{a},\ sk = \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2))$
2: $\quad \mathbf{y}_1, \mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$ ▷ Gaussian sampling
3: $\quad \mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, \mu)$ ▷ special hashing
4: $\quad$ choose a random bit b
5: $\quad \mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
6: $\quad \mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$
7: $\quad$ **continue** with probability
$1/\left(M \exp(-\|\mathbf{Sc}\|^2/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc} \rangle/\sigma^2\right)$ otherwise **restart**
8: $\quad \mathbf{z}_2^\dagger \leftarrow \textsc{Compress}(\mathbf{z}_2)$
9: $\quad$ **return** $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$
10: **end function**

©2019 NTT Secure Platform Laboratories

BLISS: signature (simplified)

1: **function** $\textsc{Sign}(\mu, pk = \mathbf{a},\ sk = \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2))$
2: $\quad \mathbf{y}_1, \mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$ ▷ Gaussian sampling
3: $\quad \mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, \mu)$ ▷ special hashing
4: $\quad$ choose a random bit b
5: $\quad \mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
6: $\quad \mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$
7: $\quad$ **continue** with probability
$1/\left(M \exp(-\|\mathbf{Sc}\|^2/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc} \rangle/\sigma^2\right)$ otherwise **restart**
8: $\quad \mathbf{z}_2^\dagger \leftarrow \textsc{Compress}(\mathbf{z}_2)$
9: $\quad$ **return** $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$
10: **end function**

©2019 NTT Secure Platform Laboratories

BLISS: signature (simplified)

1: **function** $\textsc{Sign}(\mu, pk = \mathbf{a},\ sk = \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2))$
2: $\quad \mathbf{y}_1, \mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$ ▷ Gaussian sampling
3: $\quad \mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, \mu)$ ▷ special hashing
4: $\quad$ choose a random bit b
5: $\quad \mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
6: $\quad \mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$
7: $\quad$ **continue** with probability $1/\left(M \exp(-\|\mathbf{Sc}\|^2/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc} \rangle/\sigma^2\right)$ otherwise **restart**
8: $\quad \mathbf{z}_2^\dagger \leftarrow \textsc{Compress}(\mathbf{z}_2)$
9: $\quad$ **return** $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$
10: **end function**

©2019 NTT Secure Platform Laboratories

BLISS: signature (simplified)

1: **function** $\textsc{Sign}(\mu, pk = \mathbf{a},\ sk = \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2))$
2: $\quad \mathbf{y}_1, \mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$ ▷ Gaussian sampling
3: $\quad \mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, \mu)$ ▷ special hashing
4: $\quad$ choose a random bit b
5: $\quad \mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
6: $\quad \mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$
7: $\quad$ **continue** with probability
$1/\left(M \exp(-\|\mathbf{Sc}\|^2/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc} \rangle/\sigma^2\right)$ otherwise **restart**
8: $\quad \mathbf{z}_2^\dagger \leftarrow \textsc{Compress}(\mathbf{z}_2)$
9: $\quad$ **return** $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$
10: **end function**

©2019 NTT Secure Platform Laboratories

BLISS: signature (simplified)

1: **function** $\textsc{Sign}(\mu, pk = \mathbf{a},\ sk = \mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2))$
2: $\quad \mathbf{y}_1, \mathbf{y}_2 \leftarrow D^n_{\mathbb{Z},\sigma}$ ▷ Gaussian sampling
3: $\quad \mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, \mu)$ ▷ special hashing
4: $\quad$ choose a random bit b
5: $\quad \mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
6: $\quad \mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$
7: $\quad$ **continue** with probability
$1/\left(M \exp(-\|\mathbf{Sc}\|^2/(2\sigma^2)) \cosh(\langle \mathbf{z}, \mathbf{Sc} \rangle/\sigma^2\right)$ otherwise **restart**
8: $\quad \mathbf{z}_2^\dagger \leftarrow \textsc{Compress}(\mathbf{z}_2)$
9: $\quad$ **return** $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$
10: **end function**

©2019 NTT Secure Platform Laboratories

BLISS: verification

To check if $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$ is a valid signature:

1. Uncompress $\mathbf{z}_2^\dagger$ to essentially get $\mathbf{z}_2$
2. Check if $\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}$ are small enough
3. Compute $\mathbf{u} = \mathbf{a} \cdot \mathbf{z}_1 + \mathbf{z}_2$; it satisfies:

$$\begin{aligned}\mathbf{u} &= \mathbf{a} \cdot (\mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}) + (\mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}) \\ &= (\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2) + (-1)^b (\mathbf{a}\mathbf{s}_1 + \mathbf{s}_2) \qquad = \mathbf{a}\mathbf{y}_1 + \mathbf{y}_2 \pmod q\end{aligned}$$

since $\mathbf{a} = -\mathbf{s}_2/\mathbf{s}_1 \bmod q$

4. Check whether $H(\mathbf{u}) \stackrel{?}{=} \mathbf{c}$

Works even with approximate decompression, because H depends only on the most significant bits of its input

©2019 NTT Secure Platform Laboratories

BLISS: parameters

- Parameters proposed by Ducas et al. for 128-bit security (BLISS–I)
  - $n = 512$, $q = 12289$
  - $\delta = 0.3$ (density of $\mathbf{s}_1, \mathbf{s}_2$)
  - $\sigma = 215$ (std. dev. of $\mathbf{y}_1, \mathbf{y}_2$)
  - $\kappa = 23$ (number of 1's in $\mathbf{c}$)
- Correspond to the following sizes:
  - sk size: 0.25 kB
  - vk size: 0.9 kB
  - sig. size: 0.7 kB
  - repetition rate: 1.6
- Main sources of improvement compared to the naive scheme:
  - security of the signing key based on LWE instead of SIS
  - use of NTRU lattices
  - randomness sampled from a discrete Gaussian distribution (instead of uniform)
  - bimodal Gaussians to reduce the repetition rate

©2019 NTT Secure Platform Laboratories

BLISS: parameters

- Parameters proposed by Ducas et al. for 128-bit security (BLISS–I)
  - $n = 512$, $q = 12289$
  - $\delta = 0.3$ (density of $\mathbf{s}_1, \mathbf{s}_2$)
  - $\sigma = 215$ (std. dev. of $\mathbf{y}_1, \mathbf{y}_2$)
  - $\kappa = 23$ (number of 1's in $\mathbf{c}$)
- Correspond to the following sizes:
  - sk size: 0.25 kB
  - vk size: 0.9 kB
  - sig. size: 0.7 kB
  - repetition rate: 1.6
- Main sources of improvement compared to the naive scheme:
  - security of the signing key based on LWE instead of SIS
  - use of NTRU lattices
  - randomness sampled from a discrete Gaussian distribution (instead of uniform)
  - bimodal Gaussians to reduce the repetition rate

©2019 NTT Secure Platform Laboratories

BLISS: parameters

- Parameters proposed by Ducas et al. for 128-bit security (BLISS–I)
  - $n = 512$, $q = 12289$
  - $\delta = 0.3$ (density of $\mathbf{s}_1, \mathbf{s}_2$)
  - $\sigma = 215$ (std. dev. of $\mathbf{y}_1, \mathbf{y}_2$)
  - $\kappa = 23$ (number of 1's in $\mathbf{c}$)
- Correspond to the following sizes:
  - sk size: 0.25 kB
  - vk size: 0.9 kB
  - sig. size: 0.7 kB
  - repetition rate: 1.6
- Main sources of improvement compared to the naive scheme:
  - security of the signing key based on LWE instead of SIS
  - use of NTRU lattices
  - randomness sampled from a discrete Gaussian distribution (instead of uniform)
  - bimodal Gaussians to reduce the repetition rate

©2019 NTT Secure Platform Laboratories

Outline

©2019 NTT Secure Platform Laboratories

Attacking y

- The ring element $\mathbf{y}_1$, which acts as additive mask in the relation:

$$\mathbf{z}_1 \equiv \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c} \pmod{q}$$

  is sampled according to a discrete Gaussian
- Sampling carried out coefficient by coefficient
- Idea of the attack: use fault injection to abort the sampling early, so that a faulty signature will be generated with a low-degree $\mathbf{y}_1$
- Can be done by attacking the branching test of the loop (voltage spike, clock variation...), or the contents of the loop counter (lasers, x-rays...)

©2019 NTT Secure Platform Laboratories

Attacking y

- The ring element $\mathbf{y}_1$, which acts as additive mask in the relation:

$$\mathbf{z}_1 \equiv \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c} \pmod{q}$$

  is sampled according to a discrete Gaussian
- Sampling carried out coefficient by coefficient
- Idea of the attack: use fault injection to abort the sampling early, so that a faulty signature will be generated with a low-degree $\mathbf{y}_1$
- Can be done by attacking the branching test of the loop (voltage spike, clock variation...), or the contents of the loop counter (lasers, x-rays...)

©2019 NTT Secure Platform Laboratories

Attacking y

- The ring element $\mathbf{y}_1$, which acts as additive mask in the relation:

$$\mathbf{z}_1 \equiv \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c} \pmod{q}$$

  is sampled according to a discrete Gaussian
- Sampling carried out coefficient by coefficient
- Idea of the attack: use fault injection to abort the sampling early, so that a faulty signature will be generated with a low-degree $\mathbf{y}_1$
- Can be done by attacking the branching test of the loop (voltage spike, clock variation...), or the contents of the loop counter (lasers, x-rays...)

©2019 NTT Secure Platform Laboratories

Attacking y

- The ring element $\mathbf{y}_1$, which acts as additive mask in the relation:

$$\mathbf{z}_1 \equiv \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c} \pmod{q}$$

  is sampled according to a discrete Gaussian
- Sampling carried out coefficient by coefficient
- Idea of the attack: use fault injection to abort the sampling early, so that a faulty signature will be generated with a low-degree $\mathbf{y}_1$
- Can be done by attacking the branching test of the loop (voltage spike, clock variation...), or the contents of the loop counter (lasers, x-rays...)

©2019 NTT Secure Platform Laboratories

Attack details (I)

- So let's say we get a signature generated with $\mathbf{y}_1$ of degree $m \ll n$
- If $\mathbf{c}$ is invertible (probability around $(1-1/q)^n \approx 96\%$), we can compute:

$$\mathbf{v} = \mathbf{c}^{-1}\mathbf{z}_1 \equiv \mathbf{c}^{-1}\mathbf{y}_1 + (-1)^b\mathbf{s}_1 \pmod{q}$$

- WLOG, $b = 0$ (equivalent keys)
- Since $\mathbf{s}_1$ is very short, $\mathbf{v}$ very close to the lattice L generated by $q\mathbb{Z}^n$ and $\mathbf{w}_i = \mathbf{c}^{-1}\mathbf{x}^i$, $i = 0, \ldots, m$
- L of dimension n: too large to apply lattice reduction
- However, we have the same relation on arbitrary subset of coefficients: we can reduce the dimension

©2019 NTT Secure Platform Laboratories

Attack details (I)

- So let's say we get a signature generated with $\mathbf{y}_1$ of degree $m \ll n$
- If $\mathbf{c}$ is invertible (probability around $(1 - 1/q)^n \approx 96\%$), we can compute:

$$\mathbf{v} = \mathbf{c}^{-1}\mathbf{z}_1 \equiv \mathbf{c}^{-1}\mathbf{y}_1 + (-1)^b\mathbf{s}_1 \pmod{q}$$

- WLOG, $b = 0$ (equivalent keys)
- Since $\mathbf{s}_1$ is very short, $\mathbf{v}$ very close to the lattice L generated by $q\mathbb{Z}^n$ and $\mathbf{w}_i = \mathbf{c}^{-1}\mathbf{x}^i$, $i = 0, \ldots, m$
- L of dimension n: too large to apply lattice reduction
- However, we have the same relation on arbitrary subset of coefficients: we can reduce the dimension

©2019 NTT Secure Platform Laboratories

Attack details (I)

- So let's say we get a signature generated with $\mathbf{y}_1$ of degree $m \ll n$
- If $\mathbf{c}$ is invertible (probability around $(1 - 1/q)^n \approx 96\%$), we can compute:

$$\mathbf{v} = \mathbf{c}^{-1}\mathbf{z}_1 \equiv \mathbf{c}^{-1}\mathbf{y}_1 + (-1)^b\mathbf{s}_1 \pmod{q}$$

- WLOG, $b = 0$ (equivalent keys)
- Since $\mathbf{s}_1$ is very short, $\mathbf{v}$ very close to the lattice L generated by $q\mathbb{Z}^n$ and $\mathbf{w}_i = \mathbf{c}^{-1}\mathbf{x}^i$, $i = 0, \ldots, m$
- L of dimension n: too large to apply lattice reduction
- However, we have the same relation on arbitrary subset of coefficients: we can reduce the dimension

©2019 NTT Secure Platform Laboratories

Attack details (I)

- So let's say we get a signature generated with $\mathbf{y}_1$ of degree $m \ll n$
- If $\mathbf{c}$ is invertible (probability around $(1 - 1/q)^n \approx 96\%$), we can compute:

$$\mathbf{v} = \mathbf{c}^{-1}\mathbf{z}_1 \equiv \mathbf{c}^{-1}\mathbf{y}_1 + (-1)^b\mathbf{s}_1 \pmod{q}$$

- WLOG, $b = 0$ (equivalent keys)
- Since $\mathbf{s}_1$ is very short, $\mathbf{v}$ very close to the lattice L generated by $q\mathbb{Z}^n$ and $\mathbf{w}_i = \mathbf{c}^{-1}\mathbf{x}^i$, $i = 0, \ldots, m$
- L of dimension n: too large to apply lattice reduction
- However, we have the same relation on arbitrary subset of coefficients: we can reduce the dimension

©2019 NTT Secure Platform Laboratories

Attack details (I)

- So let's say we get a signature generated with $\mathbf{y}_1$ of degree $m \ll n$
- If $\mathbf{c}$ is invertible (probability around $(1-1/q)^n \approx 96\%$), we can compute:

$$\mathbf{v} = \mathbf{c}^{-1}\mathbf{z}_1 \equiv \mathbf{c}^{-1}\mathbf{y}_1 + (-1)^b\mathbf{s}_1 \pmod{q}$$

- WLOG, $b = 0$ (equivalent keys)
- Since $\mathbf{s}_1$ is very short, $\mathbf{v}$ very close to the lattice L generated by $q\mathbb{Z}^n$ and $\mathbf{w}_i = \mathbf{c}^{-1}\mathbf{x}^i$, $i = 0, \ldots, m$
- L of dimension n: too large to apply lattice reduction
- However, we have the same relation on arbitrary subset of coefficients: we can reduce the dimension

©2019 NTT Secure Platform Laboratories

Attack details (I)

- So let's say we get a signature generated with $\mathbf{y}_1$ of degree $m \ll n$
- If $\mathbf{c}$ is invertible (probability around $(1-1/q)^n \approx 96\%$), we can compute:

$$\mathbf{v} = \mathbf{c}^{-1}\mathbf{z}_1 \equiv \mathbf{c}^{-1}\mathbf{y}_1 + (-1)^b\mathbf{s}_1 \pmod{q}$$

- WLOG, $b = 0$ (equivalent keys)
- Since $\mathbf{s}_1$ is very short, $\mathbf{v}$ very close to the lattice L generated by $q\mathbb{Z}^n$ and $\mathbf{w}_i = \mathbf{c}^{-1}\mathbf{x}^i$, $i = 0, \ldots, m$
- L of dimension n: too large to apply lattice reduction
- However, we have the same relation on arbitrary subset of coefficients: we can reduce the dimension

©2019 NTT Secure Platform Laboratories

Attack details (II)

- More precisely, fix a subset $I \subset \{0, \ldots, n-1\}$ of ℓ indices, and let $\varphi_I : \mathbb{Z}^n \to \mathbb{Z}^I$ be the obvious projection
- $\varphi_I(\mathbf{v})$ is close to the lattice generated by $\varphi_I(\mathbf{w}_i)$ and $q\mathbb{Z}^I$, and if ℓ is large enough, the difference should be $\varphi_I(\mathbf{s}_1)$.
- Solve this close vector problem using Babai nearest plane algorithm. Condition on ℓ to recover $\varphi_I(\mathbf{s}_1)$:

$$\ell + 1 \gtrsim \frac{m + 2 + \frac{\log \sqrt{\delta_1 + 4\delta_2}}{\log q}}{1 - \frac{\log \sqrt{2\pi e(\delta_1 + 4\delta_2)}}{\log q}}$$

- For BLISS–I and BLISS–II, this says $\ell \approx 1.09 \cdot m$
- In practice: works fine with LLL for $m \lesssim 60$ and with BKZ with $m \lesssim 100$
- Just apply the attack for several choices of I to recover all of $\mathbf{s}_1$, and subsequently $\mathbf{s}_2$: full key recovery with one fauly sig.!

©2019 NTT Secure Platform Laboratories

Attack details (II)

- More precisely, fix a subset $I \subset \{0, \ldots, n-1\}$ of ℓ indices, and let $\varphi_I : \mathbb{Z}^n \to \mathbb{Z}^I$ be the obvious projection
- $\varphi_I(\mathbf{v})$ is close to the lattice generated by $\varphi_I(\mathbf{w}_i)$ and $q\mathbb{Z}^I$, and if ℓ is large enough, the difference should be $\varphi_I(\mathbf{s}_1)$.
- Solve this close vector problem using Babai nearest plane algorithm. Condition on ℓ to recover $\varphi_I(\mathbf{s}_1)$:

$$\ell + 1 \gtrsim \frac{m + 2 + \frac{\log \sqrt{\delta_1 + 4\delta_2}}{\log q}}{1 - \frac{\log \sqrt{2\pi e(\delta_1 + 4\delta_2)}}{\log q}}$$

- For BLISS–I and BLISS–II, this says $\ell \approx 1.09 \cdot m$
- In practice: works fine with LLL for $m \lesssim 60$ and with BKZ with $m \lesssim 100$
- Just apply the attack for several choices of I to recover all of $\mathbf{s}_1$, and subsequently $\mathbf{s}_2$: full key recovery with one fauly sig.!

©2019 NTT Secure Platform Laboratories

Attack details (II)

- More precisely, fix a subset $I \subset \{0, \ldots, n-1\}$ of ℓ indices, and let $\varphi_I : \mathbb{Z}^n \to \mathbb{Z}^I$ be the obvious projection
- $\varphi_I(\mathbf{v})$ is close to the lattice generated by $\varphi_I(\mathbf{w}_i)$ and $q\mathbb{Z}^I$, and if ℓ is large enough, the difference should be $\varphi_I(\mathbf{s}_1)$.
- Solve this close vector problem using Babai nearest plane algorithm. Condition on ℓ to recover $\varphi_I(\mathbf{s}_1)$:

$$\ell + 1 \gtrsim \frac{m + 2 + \frac{\log \sqrt{\delta_1 + 4\delta_2}}{\log q}}{1 - \frac{\log \sqrt{2\pi e(\delta_1 + 4\delta_2)}}{\log q}}$$

- For BLISS–I and BLISS–II, this says $\ell \approx 1.09 \cdot m$
- In practice: works fine with LLL for $m \lesssim 60$ and with BKZ with $m \lesssim 100$
- Just apply the attack for several choices of I to recover all of $\mathbf{s}_1$, and subsequently $\mathbf{s}_2$: full key recovery with one fauly sig.!

©2019 NTT Secure Platform Laboratories

Attack details (II)

- More precisely, fix a subset $I \subset \{0, \ldots, n-1\}$ of ℓ indices, and let $\varphi_I : \mathbb{Z}^n \to \mathbb{Z}^I$ be the obvious projection
- $\varphi_I(\mathbf{v})$ is close to the lattice generated by $\varphi_I(\mathbf{w}_i)$ and $q\mathbb{Z}^I$, and if ℓ is large enough, the difference should be $\varphi_I(\mathbf{s}_1)$.
- Solve this close vector problem using Babai nearest plane algorithm. Condition on ℓ to recover $\varphi_I(\mathbf{s}_1)$:

$$\ell + 1 \gtrsim \frac{m + 2 + \frac{\log \sqrt{\delta_1 + 4\delta_2}}{\log q}}{1 - \frac{\log \sqrt{2\pi e(\delta_1 + 4\delta_2)}}{\log q}}$$

- For BLISS–I and BLISS–II, this says $\ell \approx 1.09 \cdot m$
- In practice: works fine with LLL for $m \lesssim 60$ and with BKZ with $m \lesssim 100$
- Just apply the attack for several choices of I to recover all of $\mathbf{s}_1$, and subsequently $\mathbf{s}_2$: full key recovery with one fauly sig.!

©2019 NTT Secure Platform Laboratories

Attack details (II)

- More precisely, fix a subset $I \subset \{0, \ldots, n-1\}$ of ℓ indices, and let $\varphi_I : \mathbb{Z}^n \to \mathbb{Z}^I$ be the obvious projection
- $\varphi_I(\mathbf{v})$ is close to the lattice generated by $\varphi_I(\mathbf{w}_i)$ and $q\mathbb{Z}^I$, and if ℓ is large enough, the difference should be $\varphi_I(\mathbf{s}_1)$.
- Solve this close vector problem using Babai nearest plane algorithm. Condition on ℓ to recover $\varphi_I(\mathbf{s}_1)$:

$$\ell + 1 \gtrsim \frac{m + 2 + \frac{\log \sqrt{\delta_1 + 4\delta_2}}{\log q}}{1 - \frac{\log \sqrt{2\pi e(\delta_1 + 4\delta_2)}}{\log q}}$$

- For BLISS–I and BLISS–II, this says $\ell \approx 1.09 \cdot m$
- In practice: works fine with LLL for $m \lesssim 60$ and with BKZ with $m \lesssim 100$
- Just apply the attack for several choices of I to recover all of $\mathbf{s}_1$, and subsequently $\mathbf{s}_2$: full key recovery with one fauly sig.!

©2019 NTT Secure Platform Laboratories

Attack details (II)

- More precisely, fix a subset $I \subset \{0, \ldots, n-1\}$ of ℓ indices, and let $\varphi_I : \mathbb{Z}^n \to \mathbb{Z}^I$ be the obvious projection
- $\varphi_I(\mathbf{v})$ is close to the lattice generated by $\varphi_I(\mathbf{w}_i)$ and $q\mathbb{Z}^I$, and if ℓ is large enough, the difference should be $\varphi_I(\mathbf{s}_1)$.
- Solve this close vector problem using Babai nearest plane algorithm. Condition on ℓ to recover $\varphi_I(\mathbf{s}_1)$:

$$\ell + 1 \gtrsim \frac{m + 2 + \frac{\log \sqrt{\delta_1 + 4\delta_2}}{\log q}}{1 - \frac{\log \sqrt{2\pi e(\delta_1 + 4\delta_2)}}{\log q}}$$

- For BLISS–I and BLISS–II, this says $\ell \approx 1.09 \cdot m$
- In practice: works fine with LLL for $m \lesssim 60$ and with BKZ with $m \lesssim 100$
- Just apply the attack for several choices of I to recover all of $\mathbf{s}_1$, and subsequently $\mathbf{s}_2$: full key recovery with one fauly sig.!

©2019 NTT Secure Platform Laboratories

Implementation results

Fault after iteration number m =	5	10	20	40	80	100
Theoretical minimum dimension ℓ_{min}	6	11	22	44	88	110
Dimension ℓ in our experiment	6	12	24	50	110	150
Lattice reduction algorithm	LLL	LLL	LLL	BKZ–20	BKZ–25	BKZ–25
Success probability (%)	99	100	100	100	100	98
Avg. CPU time to recover ℓ coeffs. (s)	0.005	0.022	0.23	7.3	941	33655
Avg. CPU time for full key recovery	0.5 s	1 s	5 s	80 s	80 min	38 h

©2019 NTT Secure Platform Laboratories

Outline

©2019 NTT Secure Platform Laboratories

Attack overview

- Attack on the rejection sampling
  - cornerstone of BLISS security/efficiency
- Straightforward implementation of rejection sampling would be inefficient: use optimized rejection algorithm
- Idea of the optimization: iterated Bernoulli trials on the bits of $\|\mathbf{Sc}\|^2$
- Side-channel leakage: can read off $\|\mathbf{Sc}\|^2$ on SPA/SEMA trace!
- From a few of these: recover $\mathbf{s_1} \cdot \bar{\mathbf{s}_1}$ ("relative norm" of the secret key)
- Then, algebraic number theory to retrieve $\mathbf{s}_1$

©2019 NTT Secure Platform Laboratories

Attack overview

- Attack on the rejection sampling
  - cornerstone of BLISS security/efficiency
- Straightforward implementation of rejection sampling would be inefficient: use optimized rejection algorithm
- Idea of the optimization: iterated Bernoulli trials on the bits of $\|\mathbf{Sc}\|^2$
- Side-channel leakage: can read off $\|\mathbf{Sc}\|^2$ on SPA/SEMA trace!
- From a few of these: recover $\mathbf{s_1} \cdot \bar{\mathbf{s}_1}$ ("relative norm" of the secret key)
- Then, algebraic number theory to retrieve $\mathbf{s}_1$

©2019 NTT Secure Platform Laboratories

Attack overview

- Attack on the rejection sampling
  - cornerstone of BLISS security/efficiency
- Straightforward implementation of rejection sampling would be inefficient: use optimized rejection algorithm
- Idea of the optimization: iterated Bernoulli trials on the bits of $\|\mathbf{Sc}\|^2$
- Side-channel leakage: can read off $\|\mathbf{Sc}\|^2$ on SPA/SEMA trace!
- From a few of these: recover $\mathbf{s_1} \cdot \bar{\mathbf{s}_1}$ ("relative norm" of the secret key)
- Then, algebraic number theory to retrieve $\mathbf{s}_1$

©2019 NTT Secure Platform Laboratories

Attack overview

- Attack on the rejection sampling
  - cornerstone of BLISS security/efficiency
- Straightforward implementation of rejection sampling would be inefficient: use optimized rejection algorithm
- Idea of the optimization: iterated Bernoulli trials on the bits of $\|\mathbf{Sc}\|^2$
- Side-channel leakage: can read off $\|\mathbf{Sc}\|^2$ on SPA/SEMA trace!
- From a few of these: recover $\mathbf{s_1} \cdot \bar{\mathbf{s}_1}$ ("relative norm" of the secret key)
- Then, algebraic number theory to retrieve $\mathbf{s}_1$

©2019 NTT Secure Platform Laboratories

Attack overview

- Attack on the rejection sampling
  - cornerstone of BLISS security/efficiency
- Straightforward implementation of rejection sampling would be inefficient: use optimized rejection algorithm
- Idea of the optimization: iterated Bernoulli trials on the bits of $\|\mathbf{Sc}\|^2$
- Side-channel leakage: can read off $\|\mathbf{Sc}\|^2$ on SPA/SEMA trace!
- From a few of these: recover $\mathbf{s_1} \cdot \bar{\mathbf{s}_1}$ ("relative norm" of the secret key)
- Then, algebraic number theory to retrieve $\mathbf{s}_1$

©2019 NTT Secure Platform Laboratories

Attack overview

- Attack on the rejection sampling
  - cornerstone of BLISS security/efficiency
- Straightforward implementation of rejection sampling would be inefficient: use optimized rejection algorithm
- Idea of the optimization: iterated Bernoulli trials on the bits of $\|\mathbf{Sc}\|^2$
- Side-channel leakage: can read off $\|\mathbf{Sc}\|^2$ on SPA/SEMA trace!
- From a few of these: recover $\mathbf{s_1} \cdot \bar{\mathbf{s}_1}$ ("relative norm" of the secret key)
- Then, algebraic number theory to retrieve $\mathbf{s}_1$

©2019 NTT Secure Platform Laboratories

BLISS rejection sampling

function SAMPLEBERNEXP(x)
  for $i = 0$ to $\ell - 1$ **do**
    if $x_i = 1$ **then**
      Sample $a \leftarrow \mathcal{B}_{c_i}$
      if $a = 0$ **then return** 0
    end if
  end for
  return 1
end function ▷ $x = K - \|\mathbf{Sc}\|^2$

function SAMPLEBERNCOSH(x)
  Sample $a \leftarrow \mathcal{B}_{\exp(-x/f)}$
  if $a = 1$ **then return** 1
  Sample $b \leftarrow \mathcal{B}_{1/2}$
  if $b = 1$ **then restart**
  Sample $c \leftarrow \mathcal{B}_{\exp(-x/f)}$
  if $c = 1$ **then restart**
  return 0
end function ▷ $x = 2 \cdot \langle \mathbf{z}, \mathbf{Sc} \rangle$

Sampling algorithms for the distributions $\mathcal{B}_{\exp(-x/f)}$ and $\mathcal{B}_{1/\cosh(x/f)}$ ($c_i = 2^i/f$ precomputed)

©2019 NTT Secure Platform Laboratories

BLISS rejection sampling

function SAMPLEBERNEXP(x)
  for $i = 0$ to $\ell - 1$ **do**
    if $x_i = 1$ **then**
      Sample $a \leftarrow \mathcal{B}_{c_i}$
      if $a = 0$ **then return** 0
    end if
  end for
  return 1
end function ▷ $x = K - \|\mathbf{Sc}\|^2$

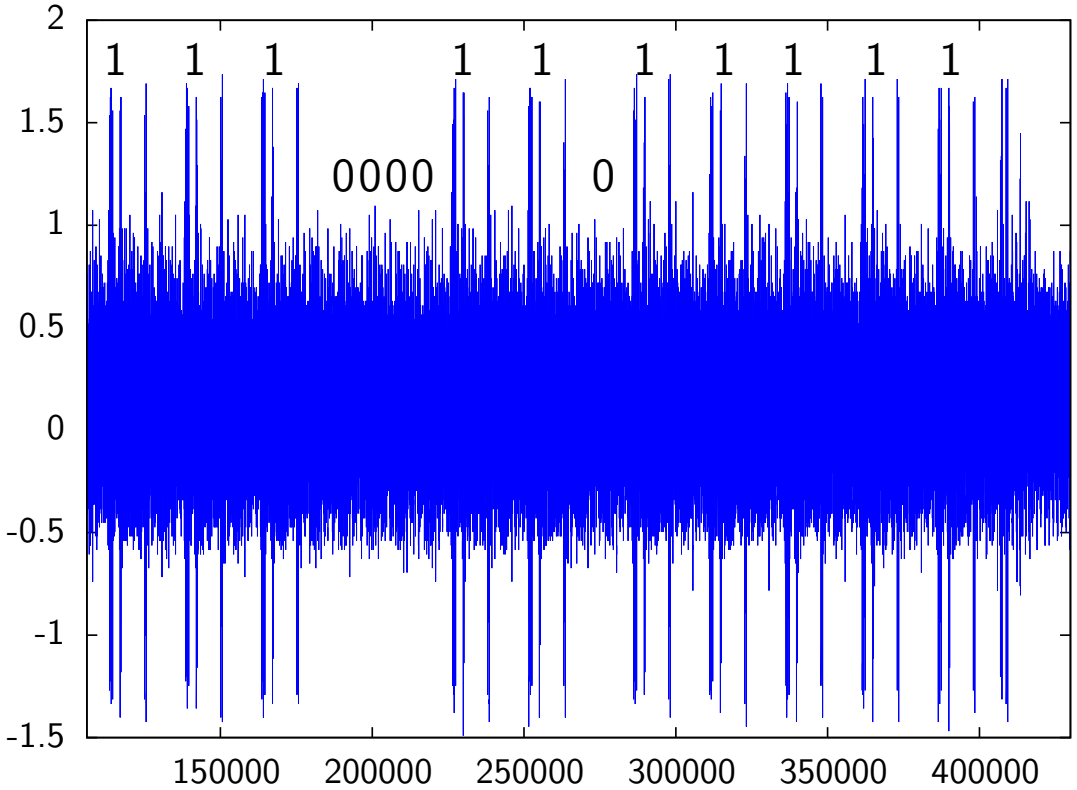
function SAMPLEBERNCOSH(x)
  Sample $a \leftarrow \mathcal{B}_{\exp(-x/f)}$
  if $a = 1$ **then return** 1
  Sample $b \leftarrow \mathcal{B}_{1/2}$
  if $b = 1$ **then restart**
  Sample $c \leftarrow \mathcal{B}_{\exp(-x/f)}$
  if $c = 1$ **then restart**
  return 0
end function ▷ $x = 2 \cdot \langle \mathbf{z}, \mathbf{Sc} \rangle$

Sampling algorithms for the distributions $\mathcal{B}_{\exp(-x/f)}$ and $\mathcal{B}_{1/\cosh(x/f)}$ ($c_i = 2^i/f$ precomputed)

©2019 NTT Secure Platform Laboratories

Experimental leakage

EMA trace of BLISS rejection sampling on 8-bit AVR for norm $\|\mathbf{Sc}\|^2 = 14404$. One reads the value:
$K - \|\mathbf{Sc}\|^2 = 46539 - 14404 = 32135 = \overline{111110110000111}_2$



©2019 NTT Secure Platform Laboratories

Exploiting the leakage

- From each trace, get:

$$\|\mathbf{Sc}\|^2 = \|\mathbf{s}_1 \cdot \mathbf{c}\|^2 + \|\mathbf{s}_2 \cdot \mathbf{c}\|^2$$

  for known $\mathbf{c}$, different each time

- Linear equation on $\mathbf{s}_1 \cdot \bar{\mathbf{s}}_1$ and $\mathbf{s}_2 \cdot \bar{\mathbf{s}}_2$
- Collect $\approx 2 \times 256 = 512$ to recover the relative norms $\mathbf{s}_i \cdot \bar{\mathbf{s}}_i$
  - linear equations are independent w.h.p.
  - very efficient in practice
  - collecting 512 EM traces an easy task
- Going from $\mathbf{s}_1 \cdot \bar{\mathbf{s}}_1$ to $\mathbf{s}_1$: algebraic number theory (Howgrave-Graham–Szydlo)

©2019 NTT Secure Platform Laboratories

Exploiting the leakage

- From each trace, get:

$$\|\mathbf{Sc}\|^2 = \|\mathbf{s}_1 \cdot \mathbf{c}\|^2 + \|\mathbf{s}_2 \cdot \mathbf{c}\|^2$$

  for known $\mathbf{c}$, different each time
- Linear equation on $\mathbf{s}_1 \cdot \bar{\mathbf{s}}_1$ and $\mathbf{s}_2 \cdot \bar{\mathbf{s}}_2$
- Collect $\approx 2 \times 256 = 512$ to recover the relative norms $\mathbf{s}_i \cdot \bar{\mathbf{s}}_i$
  - linear equations are independent w.h.p.
  - very efficient in practice
  - collecting 512 EM traces an easy task
- Going from $\mathbf{s}_1 \cdot \bar{\mathbf{s}}_1$ to $\mathbf{s}_1$: algebraic number theory (Howgrave-Graham–Szydlo)

©2019 NTT Secure Platform Laboratories

Exploiting the leakage

- From each trace, get:

$$\|\mathbf{Sc}\|^2 = \|\mathbf{s}_1 \cdot \mathbf{c}\|^2 + \|\mathbf{s}_2 \cdot \mathbf{c}\|^2$$

  for known $\mathbf{c}$, different each time
- Linear equation on $\mathbf{s}_1 \cdot \bar{\mathbf{s}}_1$ and $\mathbf{s}_2 \cdot \bar{\mathbf{s}}_2$
- Collect $\approx 2 \times 256 = 512$ to recover the relative norms $\mathbf{s}_i \cdot \bar{\mathbf{s}}_i$
  - linear equations are independent w.h.p.
  - very efficient in practice
  - collecting 512 EM traces an easy task
- Going from $\mathbf{s}_1 \cdot \bar{\mathbf{s}}_1$ to $\mathbf{s}_1$: algebraic number theory (Howgrave-Graham–Szydlo)

©2019 NTT Secure Platform Laboratories

Exploiting the leakage

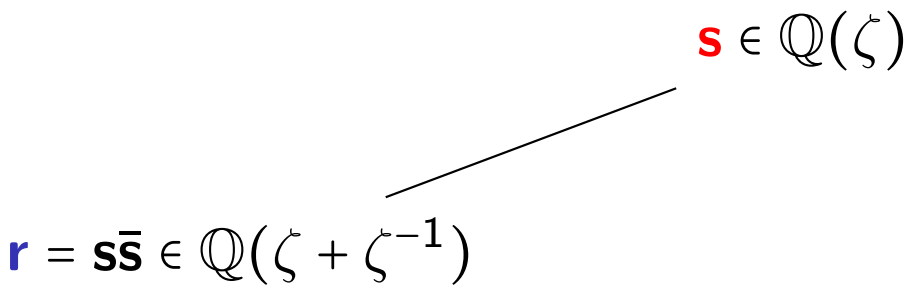
- From each trace, get:

$$\|\mathbf{Sc}\|^2 = \|\mathbf{s}_1 \cdot \mathbf{c}\|^2 + \|\mathbf{s}_2 \cdot \mathbf{c}\|^2$$

  for known $\mathbf{c}$, different each time
- Linear equation on $\mathbf{s}_1 \cdot \bar{\mathbf{s}}_1$ and $\mathbf{s}_2 \cdot \bar{\mathbf{s}}_2$
- Collect $\approx 2 \times 256 = 512$ to recover the relative norms $\mathbf{s}_i \cdot \bar{\mathbf{s}}_i$
  - linear equations are independent w.h.p.
  - very efficient in practice
  - collecting 512 EM traces an easy task
- Going from $\mathbf{s}_1 \cdot \bar{\mathbf{s}}_1$ to $\mathbf{s}_1$: algebraic number theory (Howgrave-Graham–Szydlo)

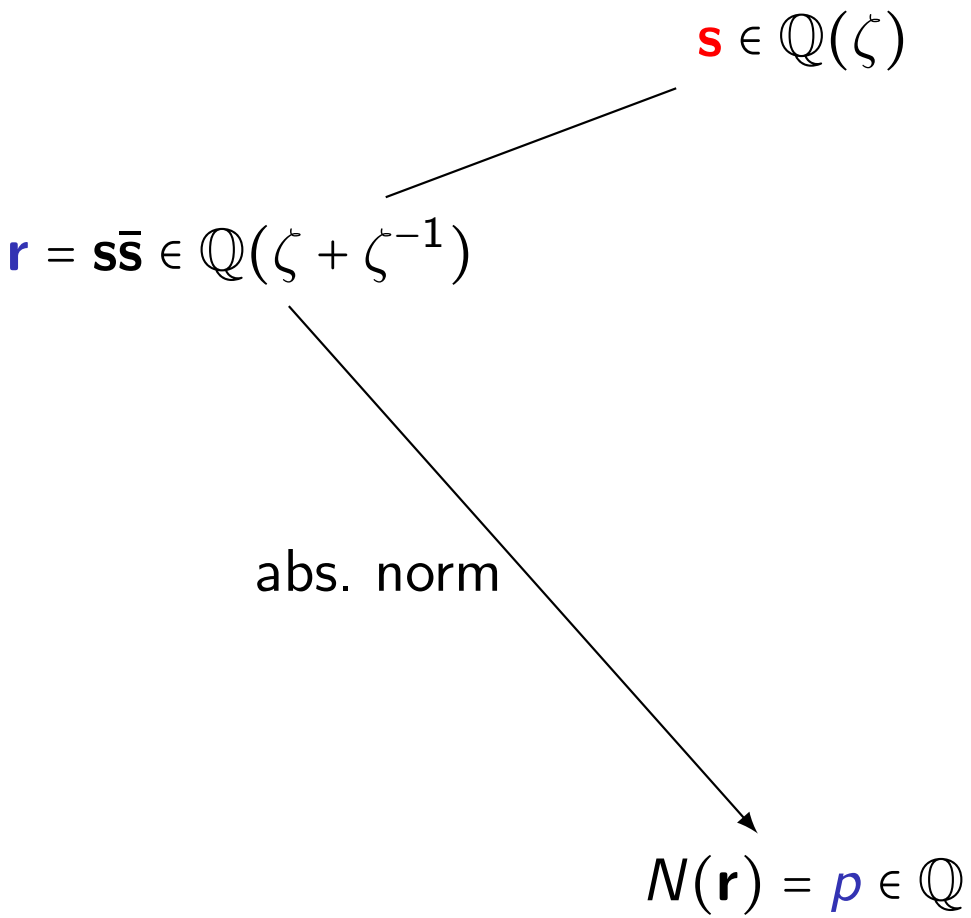
©2019 NTT Secure Platform Laboratories

Howgrave-Graham–Szydlo in a nutshell



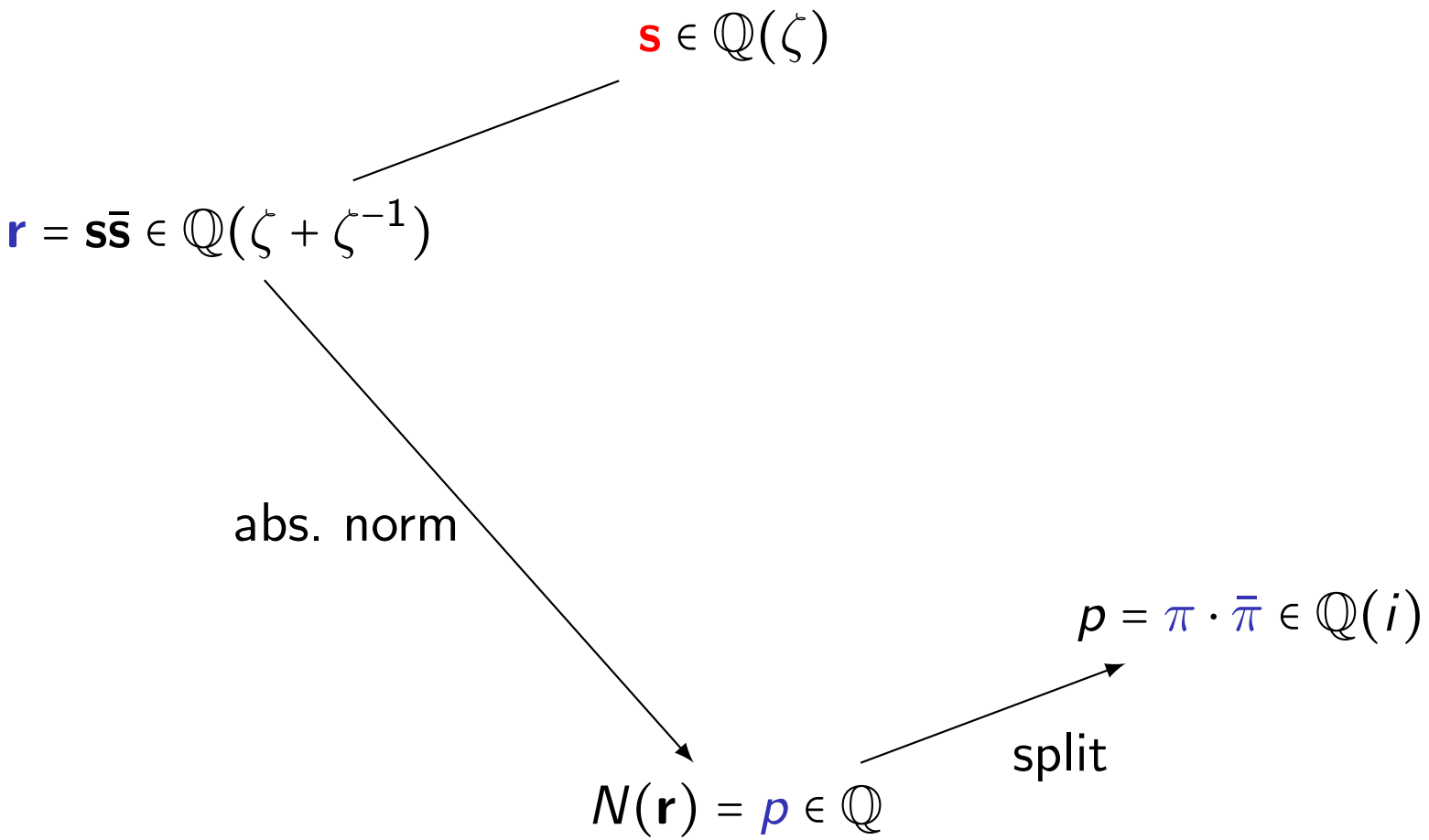
©2019 NTT Secure Platform Laboratories

Howgrave-Graham–Szydlo in a nutshell



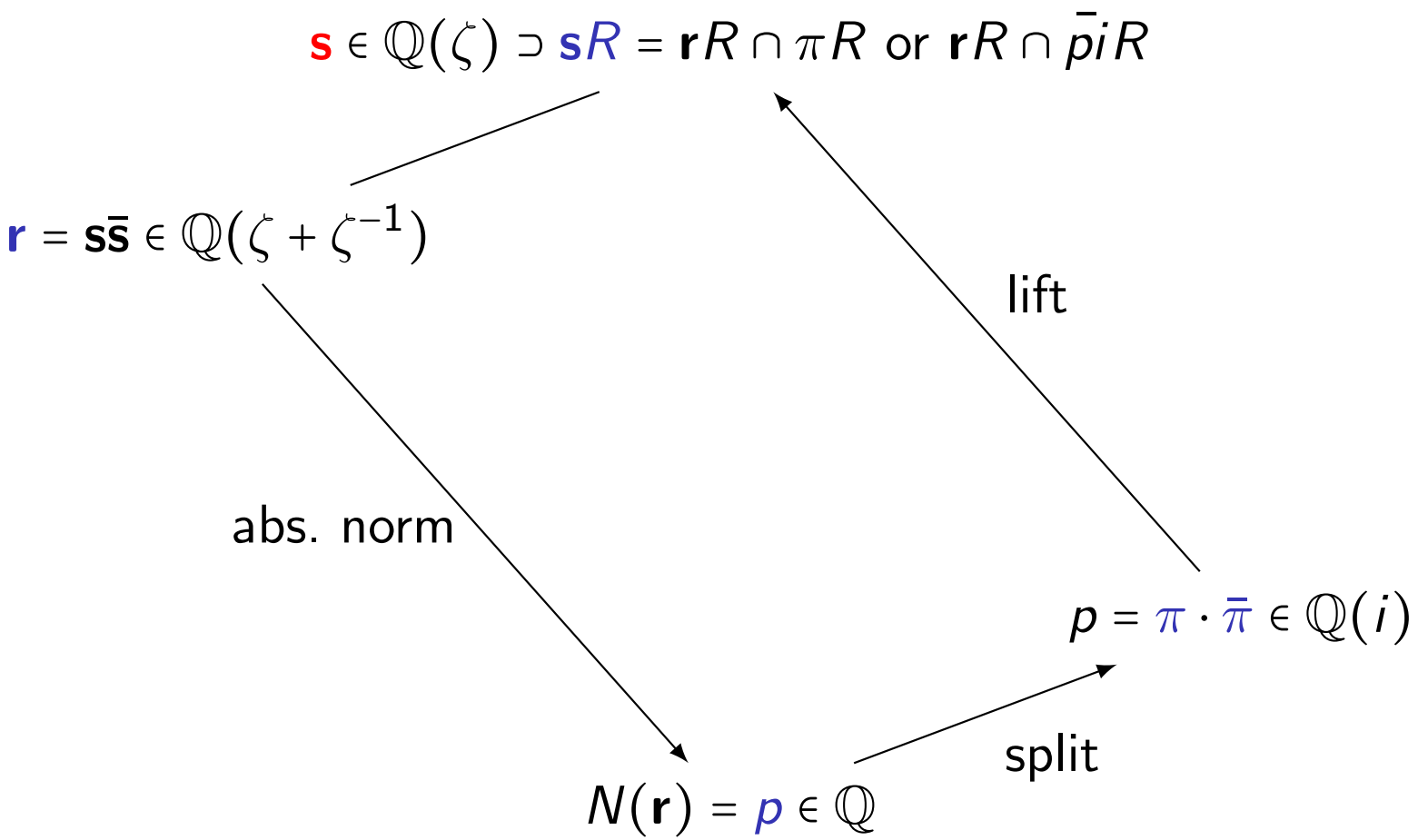
©2019 NTT Secure Platform Laboratories

Howgrave-Graham–Szydlo in a nutshell



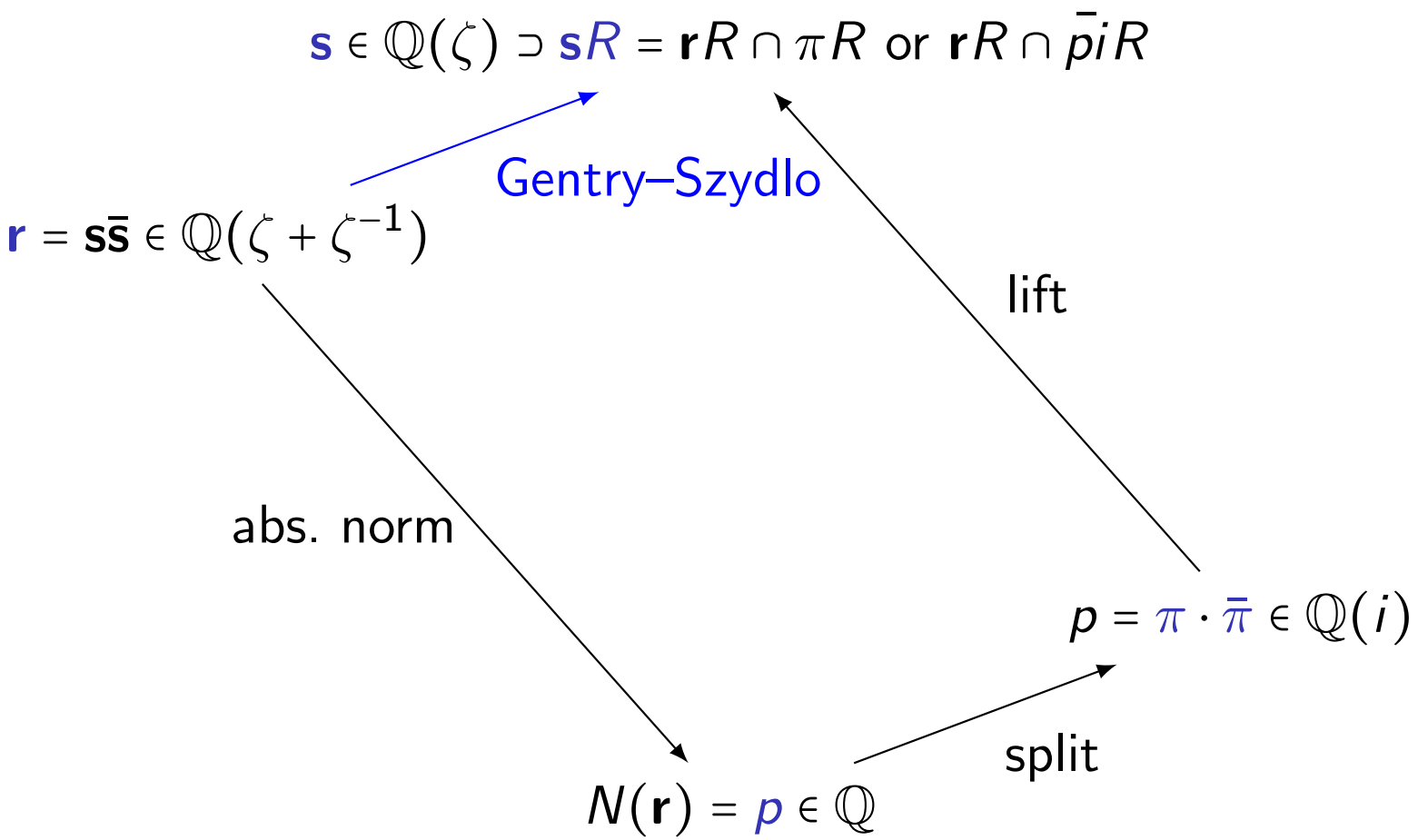
©2019 NTT Secure Platform Laboratories

Howgrave-Graham–Szydlo in a nutshell



©2019 NTT Secure Platform Laboratories

Howgrave-Graham–Szydlo in a nutshell



©2019 NTT Secure Platform Laboratories

Completing the attack

- Previous slide: works if absolute norm is prime
- Generalizes easily if we can factor the absolute norm
- Then: recover a few candidates for $\mathbf{s}_1$, up to a root of unity
- Easy to check correctness
  - compute the corresponding $\mathbf{s}_2$ as $\mathbf{a} \cdot \mathbf{s}_1 \bmod q$
  - should have coefficients in $\{-1, 0, 1\}$ and be sparse
- Multiplying a correct key $(\mathbf{s}_1, \mathbf{s}_2)$ by a root of unity results in an equivalent key, so we are done!
- Attack works for weak keys with factorable absolute norm of $\mathbf{s}_1$ or $\mathbf{s}_2$
  - e.g. norm of the form $N_0 p$ with p prime and N_0 smooth

©2019 NTT Secure Platform Laboratories

Completing the attack

- Previous slide: works if absolute norm is prime
- Generalizes easily if we can factor the absolute norm
- Then: recover a few candidates for $\mathbf{s}_1$, up to a root of unity
- Easy to check correctness
  - compute the corresponding $\mathbf{s}_2$ as $\mathbf{a} \cdot \mathbf{s}_1 \bmod q$
  - should have coefficients in $\{-1, 0, 1\}$ and be sparse
- Multiplying a correct key $(\mathbf{s}_1, \mathbf{s}_2)$ by a root of unity results in an equivalent key, so we are done!
- Attack works for weak keys with factorable absolute norm of $\mathbf{s}_1$ or $\mathbf{s}_2$
  - e.g. norm of the form $N_0 p$ with p prime and N_0 smooth

©2019 NTT Secure Platform Laboratories

Completing the attack

- Previous slide: works if absolute norm is prime
- Generalizes easily if we can factor the absolute norm
- Then: recover a few candidates for $\mathbf{s}_1$, up to a root of unity
- Easy to check correctness
  - compute the corresponding $\mathbf{s}_2$ as $\mathbf{a} \cdot \mathbf{s}_1 \bmod q$
  - should have coefficients in $\{-1, 0, 1\}$ and be sparse
- Multiplying a correct key $(\mathbf{s}_1, \mathbf{s}_2)$ by a root of unity results in an equivalent key, so we are done!
- Attack works for weak keys with factorable absolute norm of $\mathbf{s}_1$ or $\mathbf{s}_2$
  - e.g. norm of the form $N_0 p$ with p prime and N_0 smooth

©2019 NTT Secure Platform Laboratories

Completing the attack

- Previous slide: works if absolute norm is prime
- Generalizes easily if we can factor the absolute norm
- Then: recover a few candidates for $\mathbf{s}_1$, up to a root of unity
- Easy to check correctness
  - compute the corresponding $\mathbf{s}_2$ as $\mathbf{a} \cdot \mathbf{s}_1 \bmod q$
  - should have coefficients in $\{-1, 0, 1\}$ and be sparse
- Multiplying a correct key $(\mathbf{s}_1, \mathbf{s}_2)$ by a root of unity results in an equivalent key, so we are done!
- Attack works for weak keys with factorable absolute norm of $\mathbf{s}_1$ or $\mathbf{s}_2$
  - e.g. norm of the form $N_0 p$ with p prime and N_0 smooth

©2019 NTT Secure Platform Laboratories

Completing the attack

- Previous slide: works if absolute norm is prime
- Generalizes easily if we can factor the absolute norm
- Then: recover a few candidates for $\mathbf{s}_1$, up to a root of unity
- Easy to check correctness
  - compute the corresponding $\mathbf{s}_2$ as $\mathbf{a} \cdot \mathbf{s}_1 \bmod q$
  - should have coefficients in $\{-1, 0, 1\}$ and be sparse
- Multiplying a correct key $(\mathbf{s}_1, \mathbf{s}_2)$ by a root of unity results in an equivalent key, so we are done!
- Attack works for weak keys with factorable absolute norm of $\mathbf{s}_1$ or $\mathbf{s}_2$
  - e.g. norm of the form $N_0 p$ with p prime and N_0 smooth

©2019 NTT Secure Platform Laboratories

Completing the attack

- Previous slide: works if absolute norm is prime
- Generalizes easily if we can factor the absolute norm
- Then: recover a few candidates for $\mathbf{s}_1$, up to a root of unity
- Easy to check correctness
  - compute the corresponding $\mathbf{s}_2$ as $\mathbf{a} \cdot \mathbf{s}_1 \bmod q$
  - should have coefficients in $\{-1, 0, 1\}$ and be sparse
- Multiplying a correct key $(\mathbf{s}_1, \mathbf{s}_2)$ by a root of unity results in an equivalent key, so we are done!
- Attack works for weak keys with factorable absolute norm of $\mathbf{s}_1$ or $\mathbf{s}_2$
  - e.g. norm of the form $N_0 p$ with p prime and N_0 smooth

©2019 NTT Secure Platform Laboratories

Efficiency of the attack

	n	$B = 5$	$B = 65537$	$B = 655373$	$B = 6553733$
BLISS-0	256	3%	3.8%	6%	6.5%
BLISS-I/II	512	1.5%	2%	2.8%	3.7%
BLISS-III/IV	512	1%	1.75%	2%	2.5%

Experimental density of keys with semi-smooth absolute norm ($N = N_0 \cdot p$ with B-smooth N_0) for various BLISS parameters

Field size n	32	64	128	256	512
CPU time	0.6 s	13 s	21 min.	17h 22 min.	38 days
Clock cycles	$\approx 2^{30}$	$\approx 2^{35}$	$\approx 2^{41}$	$\approx 2^{47}$	$\approx 2^{53}$

Average running time of the attack for various field sizes n

©2019 NTT Secure Platform Laboratories

What about the inner product leakage? (I)

- Recall the rejection sampling probability of BLISS signing:

$$1 \Big/ \left(M \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2} \right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc} \rangle}{\sigma^2} \right) \right),$$

- The exp part of the rejection sampling leaks $\|\mathbf{Sc}\|^2$ and ultimately the relative norm of $\mathbf{s}_1$ and $\mathbf{s}_2$: what we have used so far
- Can't we use the cosh part instead? It directly leaks:

$$\langle \mathbf{z}_1, \mathbf{s}_1\mathbf{c} \rangle + \langle \mathbf{z}_2, \mathbf{s}_2\mathbf{c} \rangle$$

- If we know $(\mathbf{c}, \mathbf{z}_1, \mathbf{z}_2)$, this gives a *linear* relation on the secret: recover everything from around 1024 signatures without breaking a sweat!

©2019 NTT Secure Platform Laboratories

What about the inner product leakage? (I)

- Recall the rejection sampling probability of BLISS signing:

$$1 \Big/ \left(M \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2} \right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc} \rangle}{\sigma^2} \right) \right),$$

- The exp part of the rejection sampling leaks $\|\mathbf{Sc}\|^2$ and ultimately the relative norm of $\mathbf{s}_1$ and $\mathbf{s}_2$: what we have used so far
- Can't we use the cosh part instead? It directly leaks:

$$\langle \mathbf{z}_1, \mathbf{s}_1\mathbf{c} \rangle + \langle \mathbf{z}_2, \mathbf{s}_2\mathbf{c} \rangle$$

- If we know $(\mathbf{c}, \mathbf{z}_1, \mathbf{z}_2)$, this gives a *linear* relation on the secret: recover everything from around 1024 signatures without breaking a sweat!

©2019 NTT Secure Platform Laboratories

What about the inner product leakage? (I)

- Recall the rejection sampling probability of BLISS signing:

$$1 \Big/ \left(M \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2} \right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc} \rangle}{\sigma^2} \right) \right),$$

- The exp part of the rejection sampling leaks $\|\mathbf{Sc}\|^2$ and ultimately the relative norm of $\mathbf{s}_1$ and $\mathbf{s}_2$: what we have used so far
- Can't we use the cosh part instead? It directly leaks:

$$\langle \mathbf{z}_1, \mathbf{s}_1\mathbf{c} \rangle + \langle \mathbf{z}_2, \mathbf{s}_2\mathbf{c} \rangle$$

- If we know $(\mathbf{c}, \mathbf{z}_1, \mathbf{z}_2)$, this gives a *linear* relation on the secret: recover everything from around 1024 signatures without breaking a sweat!

©2019 NTT Secure Platform Laboratories

What about the inner product leakage? (I)

- Recall the rejection sampling probability of BLISS signing:

$$1 \Big/ \left(M \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2} \right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc} \rangle}{\sigma^2} \right) \right),$$

- The exp part of the rejection sampling leaks $\|\mathbf{Sc}\|^2$ and ultimately the relative norm of $\mathbf{s}_1$ and $\mathbf{s}_2$: what we have used so far
- Can't we use the cosh part instead? It directly leaks:

$$\langle \mathbf{z}_1, \mathbf{s}_1\mathbf{c} \rangle + \langle \mathbf{z}_2, \mathbf{s}_2\mathbf{c} \rangle$$

- If we know $(\mathbf{c}, \mathbf{z}_1, \mathbf{z}_2)$, this gives a *linear* relation on the secret: recover everything from around 1024 signatures without breaking a sweat!

©2019 NTT Secure Platform Laboratories

What about the inner product leakage? (II)

- Problem: signatures do not contain $\mathbf{z}_2$, but only a compressed variant $\mathbf{z}_2^\dagger$, and the compression is lossy: we only obtain a noisy linear system on the secret
- Our first reaction: this is like Learning With Errors in twice the original dimension, so probably hopeless
- Second thought: not hopeless at all. Since there is no modular reduction, we can simply approach the problem with linear least squares
- Works on 100% of keys, but needs ≈ 30000 signatures vs. ≈ 512 for Howgrave-Graham–Szydlo

©2019 NTT Secure Platform Laboratories

What about the inner product leakage? (II)

- Problem: signatures do not contain $\mathbf{z}_2$, but only a compressed variant $\mathbf{z}_2^\dagger$, and the compression is lossy: we only obtain a noisy linear system on the secret
- Our first reaction: this is like Learning With Errors in twice the original dimension, so probably hopeless
- Second thought: not hopeless at all. Since there is no modular reduction, we can simply approach the problem with linear least squares
- Works on 100% of keys, but needs ≈ 30000 signatures vs. ≈ 512 for Howgrave-Graham–Szydlo

©2019 NTT Secure Platform Laboratories

What about the inner product leakage? (II)

- Problem: signatures do not contain $\mathbf{z}_2$, but only a compressed variant $\mathbf{z}_2^\dagger$, and the compression is lossy: we only obtain a noisy linear system on the secret
- Our first reaction: this is like Learning With Errors in twice the original dimension, so probably hopeless
- Second thought: not hopeless at all. Since there is no modular reduction, we can simply approach the problem with linear least squares
- Works on 100% of keys, but needs ≈ 30000 signatures vs. ≈ 512 for Howgrave-Graham–Szydlo

©2019 NTT Secure Platform Laboratories

What about the inner product leakage? (II)

- Problem: signatures do not contain $\mathbf{z}_2$, but only a compressed variant $\mathbf{z}_2^\dagger$, and the compression is lossy: we only obtain a noisy linear system on the secret
- Our first reaction: this is like Learning With Errors in twice the original dimension, so probably hopeless
- Second thought: not hopeless at all. Since there is no modular reduction, we can simply approach the problem with linear least squares
- Works on 100% of keys, but needs ≈ 30000 signatures vs. ≈ 512 for Howgrave-Graham–Szydlo

©2019 NTT Secure Platform Laboratories

Further side-channel leakages in BLISS

- Several other vulnerable parts in BLISS
- Gaussian sampling: use side-channels to recover $\mathbf{y}_1$, and solve for $\mathbf{s}_1$ in $\mathbf{z}_1 = \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
  - very basic idea of the cache timing attack by Groot Bruinderink et al.
- Bit flip: if the multiplication by $(-1)^b$ is not done in constant-time, recover b with timing side-channels and construct $(-1)^b \mathbf{z}$, which is not independent of the secret key
  - full key recovery using pure statistical techniques!
- Finding good countermeasures is an active area of research!

©2019 NTT Secure Platform Laboratories

Further side-channel leakages in BLISS

- Several other vulnerable parts in BLISS
- Gaussian sampling: use side-channels to recover $\mathbf{y}_1$, and solve for $\mathbf{s}_1$ in $\mathbf{z}_1 = \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
  - very basic idea of the cache timing attack by Groot Bruinderink et al.
- Bit flip: if the multiplication by $(-1)^b$ is not done in constant-time, recover b with timing side-channels and construct $(-1)^b \mathbf{z}$, which is not independent of the secret key
  - full key recovery using pure statistical techniques!
- Finding good countermeasures is an active area of research!

©2019 NTT Secure Platform Laboratories

Further side-channel leakages in BLISS

- Several other vulnerable parts in BLISS
- Gaussian sampling: use side-channels to recover $\mathbf{y}_1$, and solve for $\mathbf{s}_1$ in $\mathbf{z}_1 = \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
  - very basic idea of the cache timing attack by Groot Bruinderink et al.
- Bit flip: if the multiplication by $(-1)^b$ is not done in constant-time, recover b with timing side-channels and construct $(-1)^b \mathbf{z}$, which is not independent of the secret key
  - full key recovery using pure statistical techniques!
- Finding good countermeasures is an active area of research!

©2019 NTT Secure Platform Laboratories

Further side-channel leakages in BLISS

- Several other vulnerable parts in BLISS
- Gaussian sampling: use side-channels to recover $\mathbf{y}_1$, and solve for $\mathbf{s}_1$ in $\mathbf{z}_1 = \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
  - very basic idea of the cache timing attack by Groot Bruinderink et al.
- Bit flip: if the multiplication by $(-1)^b$ is not done in constant-time, recover b with timing side-channels and construct $(-1)^b \mathbf{z}$, which is not independent of the secret key
  - full key recovery using pure statistical techniques!
- Finding good countermeasures is an active area of research!

©2019 NTT Secure Platform Laboratories