Réseaux Euclidiens : Algorithmes et Cryptographie

Soutenance d'HDR

Damien Stehlé

http://perso.ens-lyon.fr/damien.stehle

CNRS/LIP, ENS de Lyon

Lyon, 14 Octobre 2011

Goals of the talk

- To present facets of my field of research.
- To focus on some specific results.
- To discuss future directions.

1. Euclidean lattices: definitions and algorithmic problems.
2. Reducing lattice bases efficiently.
3. Paying more to get nicer bases.
4. Fast lattice-based cryptography.
5. Open problems.

Euclidean lattices

$$\text{Lattice} \equiv \left\{ \sum_{i \leq n} x_i \mathbf{b}_i : x_i \in \mathbb{Z} \right\},$$

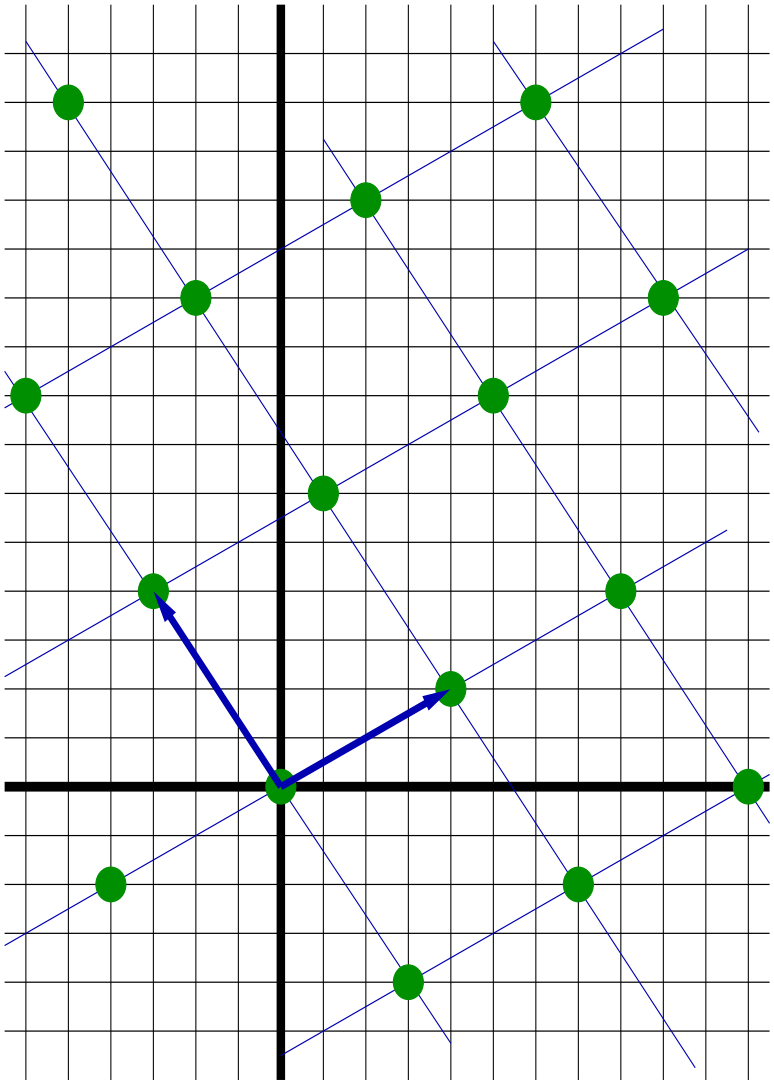
where the n linearly independent $\mathbf{b}_i$'s are called a basis.

Bases are not unique, but can be obtained from each other by integer transforms of determinant ± 1:

$$\begin{bmatrix} -2 & 1 \\ 10 & 6 \end{bmatrix} = \begin{bmatrix} 4 & -3 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}.$$

Lattice reduction:

find a nice basis, given an arbitrary one.



Euclidean lattices

Lattice $\equiv \left\{ \sum_{i \leq n} x_i \mathbf{b}_i : x_i \in \mathbb{Z} \right\}$,

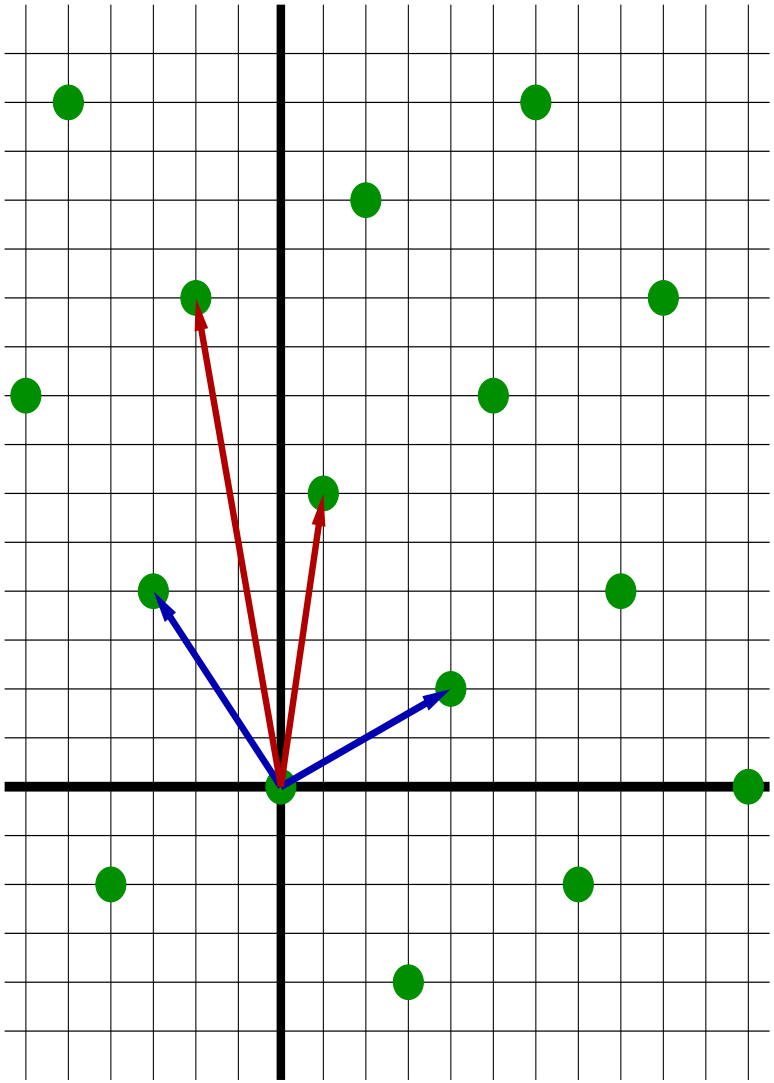
where the n linearly independent $\mathbf{b}_i$'s are called a basis.

Bases are not unique, but can be obtained from each other by integer transforms of determinant ± 1:

$$\begin{bmatrix} -2 & 1 \\ 10 & 6 \end{bmatrix} = \begin{bmatrix} 4 & -3 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}.$$

Lattice reduction:

find a nice basis, given an arbitrary one.



Euclidean lattices

Lattice $\equiv \left\{ \sum_{i \leq n} x_i \mathbf{b}_i : x_i \in \mathbb{Z} \right\}$,

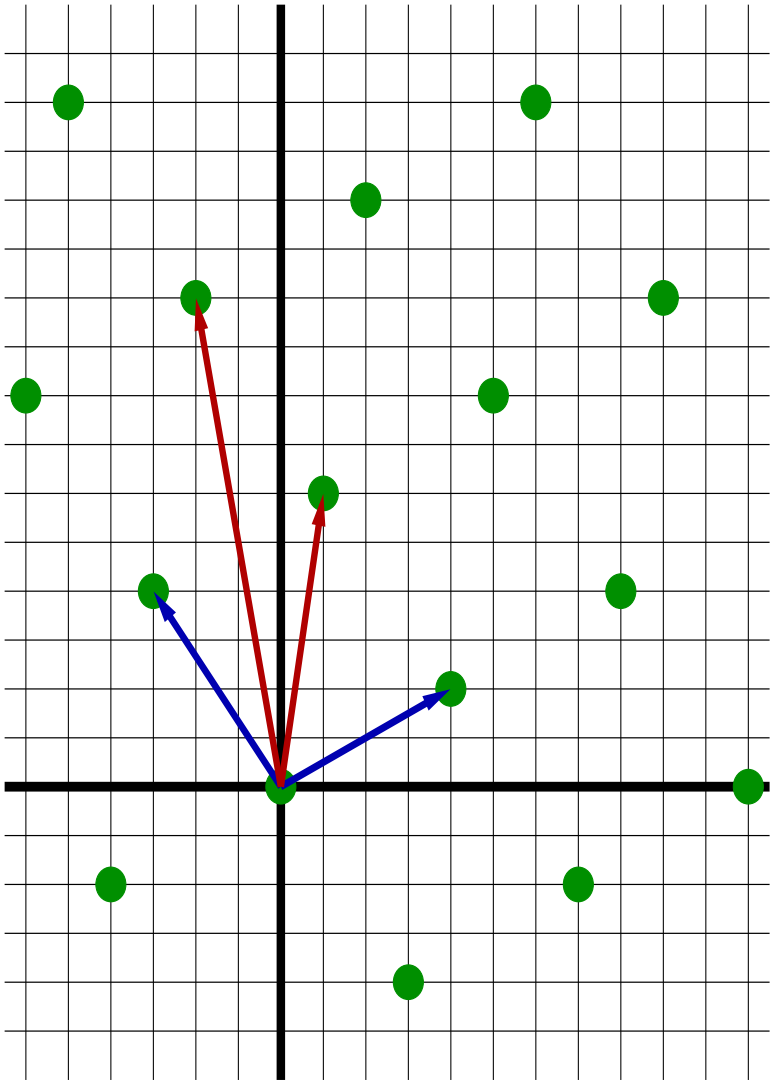
where the n linearly independent $\mathbf{b}_i$'s are called a basis.

Bases are not unique, but can be obtained from each other by integer transforms of determinant ± 1:

$$\begin{bmatrix} -2 & 1 \\ 10 & 6 \end{bmatrix} = \begin{bmatrix} 4 & -3 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}.$$

Lattice reduction:

find a nice basis, given an arbitrary one.



Lattice invariants and lattice reduction

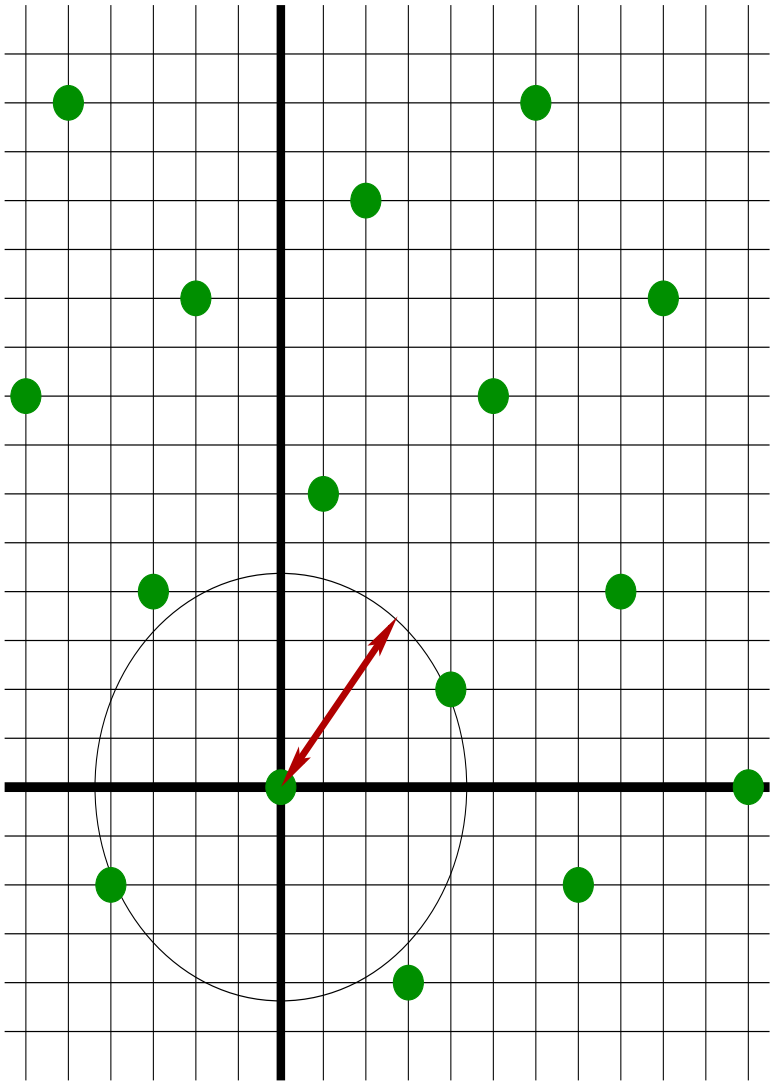
Minimum:
$\lambda(L) = \min(\|\mathbf{b}\| : \mathbf{b} \in L \setminus \mathbf{0})$.

Lattice determinant:
$\det L = |\det(\mathbf{b}_i)_i|$, for any basis.

Minkowski theorem:
$\lambda(L) \leq \sqrt{n} \cdot (\det L)^{1/n}$.

Lattice reduction:
Find basis with small Hermite Factor:

$$\mathrm{HF}(B) := \frac{\|\mathbf{b}_1\|}{(\det L)^{1/n}}.$$



Lattice invariants and lattice reduction

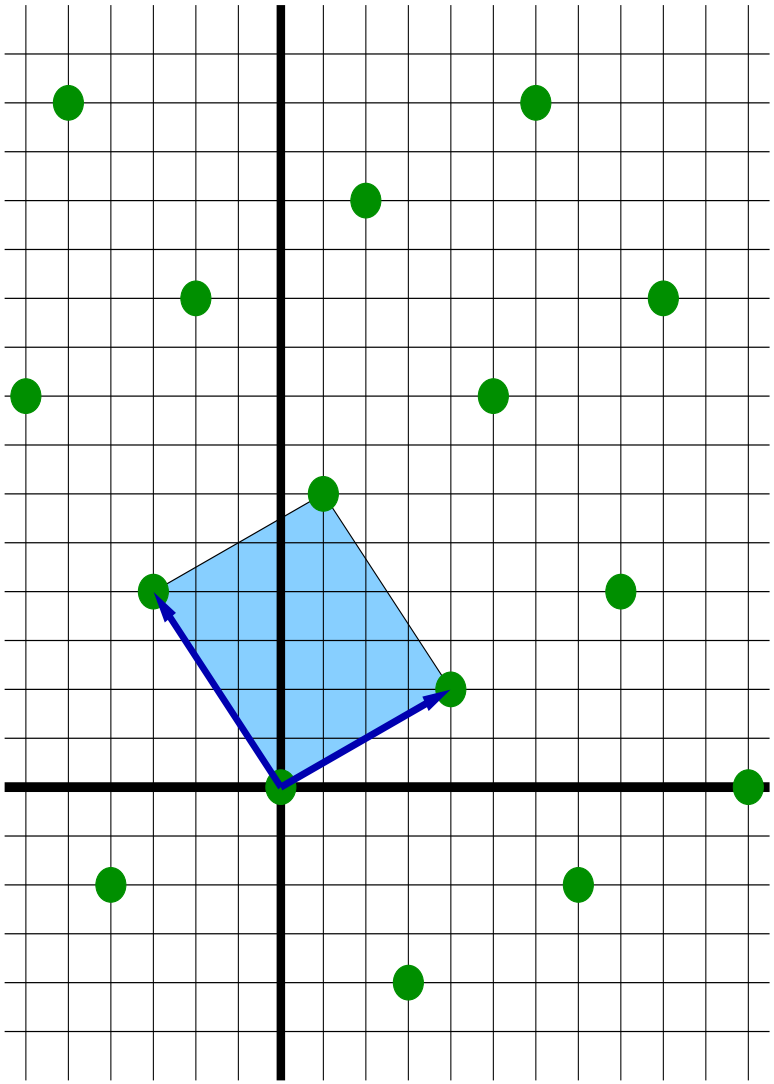
Minimum:
$\lambda(L) = \min(\|\mathbf{b}\| : \mathbf{b} \in L \setminus \mathbf{0})$.

Lattice determinant:
$\det L = |\det(\mathbf{b}_i)_i|$, for any basis.

Minkowski theorem:
$\lambda(L) \leq \sqrt{n} \cdot (\det L)^{1/n}$.

Lattice reduction:
Find basis with small Hermite Factor:

$$\mathrm{HF}(B) := \frac{\|\mathbf{b}_1\|}{(\det L)^{1/n}}.$$



Lattice invariants and lattice reduction

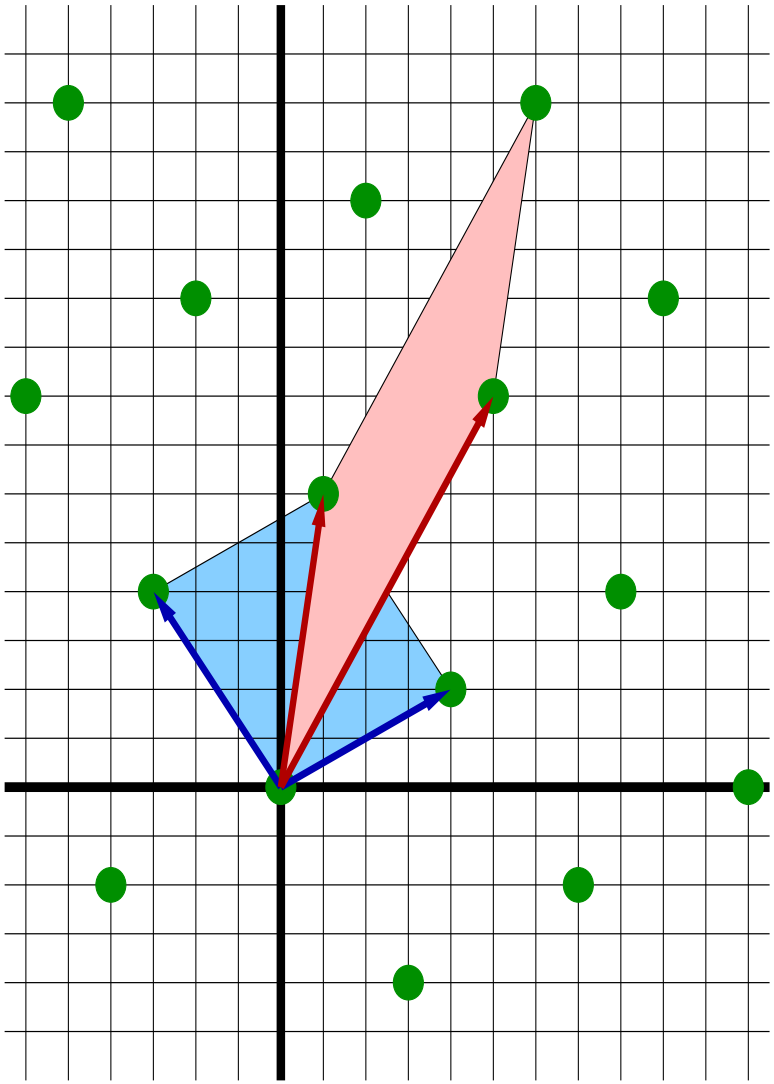
Minimum:
$\lambda(L) = \min(\|\mathbf{b}\| : \mathbf{b} \in L \setminus \mathbf{0})$.

Lattice determinant:
$\det L = |\det(\mathbf{b}_i)_i|$, for any basis.

Minkowski theorem:
$\lambda(L) \leq \sqrt{n} \cdot (\det L)^{1/n}$.

Lattice reduction:
Find basis with small Hermite Factor:

$$\mathrm{HF}(B) := \frac{\|\mathbf{b}_1\|}{(\det L)^{1/n}}.$$



Lattice invariants and lattice reduction

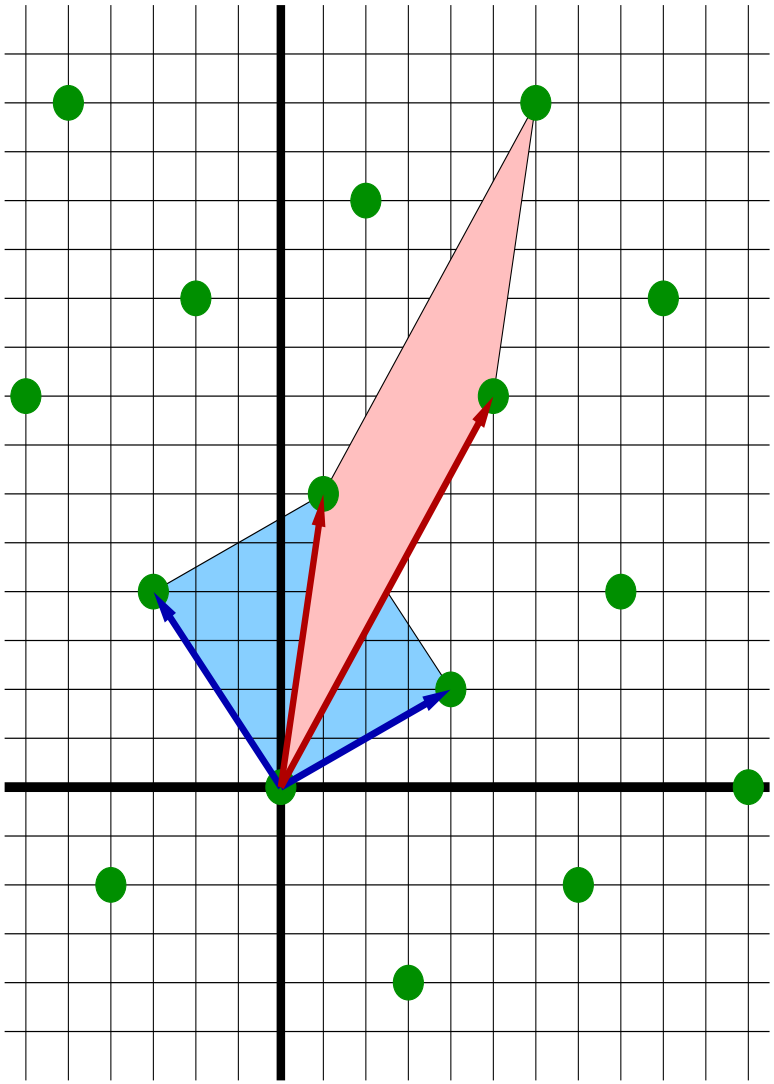
Minimum:
$\lambda(L) = \min(\|\mathbf{b}\| : \mathbf{b} \in L \setminus \mathbf{0})$.

Lattice determinant:
$\det L = |\det(\mathbf{b}_i)_i|$, for any basis.

Minkowski theorem:
$\lambda(L) \leq \sqrt{n} \cdot (\det L)^{1/n}$.

Lattice reduction:
Find basis with small Hermite Factor:

$$\mathrm{HF}(B) := \frac{\|\mathbf{b}_1\|}{(\det L)^{1/n}}.$$



Lattice invariants and lattice reduction

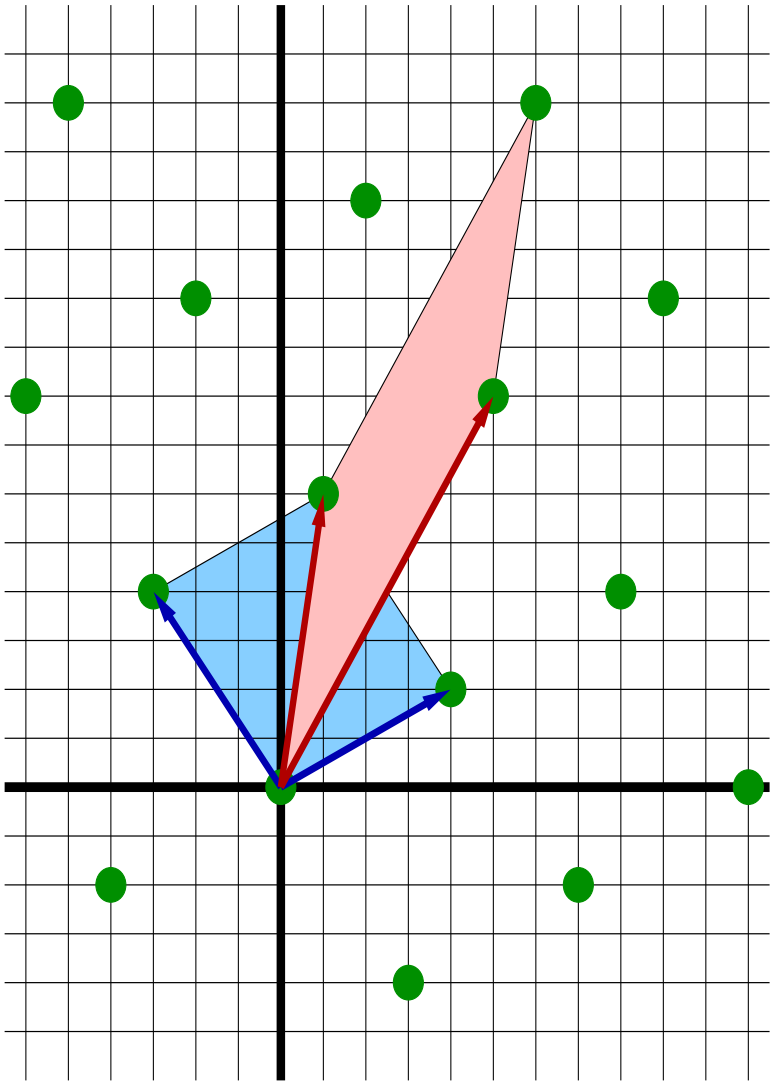
Minimum:
$\lambda(L) = \min(\|\mathbf{b}\| : \mathbf{b} \in L \setminus \mathbf{0})$.

Lattice determinant:
$\det L = |\det(\mathbf{b}_i)_i|$, for any basis.

Minkowski theorem:
$\lambda(L) \leq \sqrt{n} \cdot (\det L)^{1/n}$.

Lattice reduction:
Find basis with small Hermite Factor:

$$\mathrm{HF}(B) := \frac{\|\mathbf{b}_1\|}{(\det L)^{1/n}}.$$



Why do we care about lattice reduction?

Finding a basis with small HF allows one to solve:

SVP_γ

Given a basis of L, find $\mathbf{b} \in L$ with

$$0 < \|\mathbf{b}\| \leq \gamma \cdot \lambda(L).$$

- For small γ and large n:
  Cryptanalysis of lattice-based cryptosystems.
- For large γ but huge bit-size:
  Cryptanalyses of variants of RSA, factorisation of rational polynomials, integer relation detection, etc.

Why do we care about lattice reduction?

Finding a basis with small HF allows one to solve:

SVP_γ

Given a basis of L, find $\mathbf{b} \in L$ with

$$0 < \|\mathbf{b}\| \leq \gamma \cdot \lambda(L).$$

- For small γ and large n:
  Cryptanalysis of lattice-based cryptosystems.
- For large γ but huge bit-size:
  Cryptanalyses of variants of RSA, factorisation of rational polynomials, integer relation detection, etc.

Why do we care about lattice reduction?

Finding a basis with small HF allows one to solve:

SVP_γ

Given a basis of L, find $\mathbf{b} \in L$ with

$$0 < \|\mathbf{b}\| \leq \gamma \cdot \lambda(L).$$

- For small γ and large n:
  Cryptanalysis of lattice-based cryptosystems.
- For large γ but huge bit-size:
  Cryptanalyses of variants of RSA, factorisation of rational polynomials, integer relation detection, etc.

Several types of lattice reductions

	HKZ	BKZ_k	LLL
Hermite factor	$\sqrt{n}$	$\simeq k^{n/(2k)}$	$\simeq 2^n$
Run-time*	$2^{\mathcal{O}(n)}$	$2^{\mathcal{O}(k)} \times \mathrm{Poly}(n)$	$\mathrm{Poly}(n)$

*Neglecting arithmetic costs

- HKZ = Hermite-Korkine-Zolotareff (19th c.).
- LLL = Lenstra-Lenstra-Lovász ('82).
- BKZ = Block Korkine-Zolotareff (Schnorr'87)

Two main contradicting goals:

- Decrease the complexity bounds.
- Exploit hardness to devise cryptographic primitives.

Several types of lattice reductions

	HKZ	BKZ_k	LLL
Hermite factor	$\sqrt{n}$	$\simeq k^{n/(2k)}$	$\simeq 2^n$
Run-time*	$2^{\mathcal{O}(n)}$	$2^{\mathcal{O}(k)} \times \mathrm{Poly}(n)$	$\mathrm{Poly}(n)$

*Neglecting arithmetic costs

- HKZ = Hermite-Korkine-Zolotareff (19th c.).
- LLL = Lenstra-Lenstra-Lovász ('82).
- BKZ = Block Korkine-Zolotareff (Schnorr'87)

Two main contradicting goals:

- Decrease the complexity bounds.
- Exploit hardness to devise cryptographic primitives.

Several types of lattice reductions

	HKZ	BKZ_k	LLL
Hermite factor	$\sqrt{n}$	$\simeq k^{n/(2k)}$	$\simeq 2^n$
Run-time*	$2^{\mathcal{O}(n)}$	$2^{\mathcal{O}(k)} \times \mathrm{Poly}(n)$	$\mathrm{Poly}(n)$

*Neglecting arithmetic costs

- HKZ = Hermite-Korkine-Zolotareff (19th c.).
- LLL = Lenstra-Lenstra-Lovász ('82).
- BKZ = Block Korkine-Zolotareff (Schnorr'87)

Two main contradicting goals:

- Decrease the complexity bounds.
- Exploit hardness to devise cryptographic primitives.

Several types of lattice reductions

	HKZ	BKZ_k	LLL
Hermite factor	$\sqrt{n}$	$\simeq k^{n/(2k)}$	$\simeq 2^n$
Run-time*	$2^{\mathcal{O}(n)}$	$2^{\mathcal{O}(k)} \times \mathrm{Poly}(n)$	$\mathrm{Poly}(n)$

*Neglecting arithmetic costs

- HKZ = Hermite-Korkine-Zolotareff (19th c.).
- LLL = Lenstra-Lenstra-Lovász ('82).
- BKZ = Block Korkine-Zolotareff (Schnorr'87)

Two main contradicting goals:

- Decrease the complexity bounds.
- Exploit hardness to devise cryptographic primitives.

Road-map

1. Euclidean lattices: definitions and algorithmic problems.
2. **Reducing lattice bases efficiently.**
3. Paying more to get nicer bases.
4. Fast lattice-based cryptography.
5. Future directions.

Fast and decent reduction: LLL

- LLL is the main (only?) algorithm for finding lattice bases of decent quality: $\mathrm{HF} \leq 2^n$.
- But text-book LLL is amazingly slow.

Fast and decent reduction: LLL

- LLL is the main (only?) algorithm for finding lattice bases of decent quality: $\mathrm{HF} \leq 2^n$.
- But text-book LLL is amazingly slow.

Fast and decent reduction: LLL

- LLL is the main (only?) algorithm for finding lattice bases of decent quality: $\mathrm{HF} \leq 2^n$.
- But text-book LLL is amazingly slow.

Using MAGMA V2.16:

```
> n := 25; B := RMatrixSpace(Integers(),n,n)!0;
> beta := 2000; for i:=1 to 25 do
>    B[i][i]:=1; B[i][1]:=RandomBits(beta);
> end for;
> time C := LLL(B:Method:=``Integral'');
```

Time: 11.700

```
> time C := LLL(B);
```

Time: 0.240

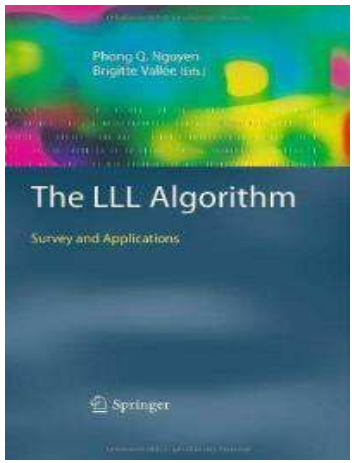
Our contributions to fast LLL reduction

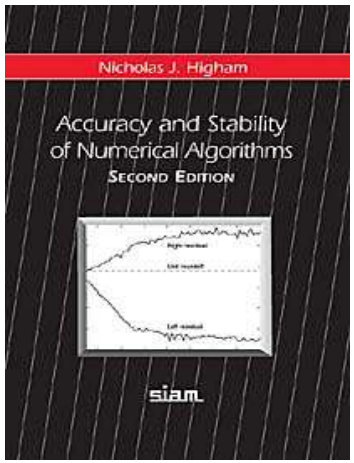
- Costly component: underlying QR/Gram-Schmidt.
- Floating-point arithmetic is well-suited for these [Odlyzko'82].

Our contributions to fast LLL reduction

- Costly component: underlying QR/Gram-Schmidt.
- Floating-point arithmetic is well-suited for these [Odlyzko'82].

$\Rightarrow$ Disclose and exploit links between lattice reduction algorithms and numerical linear algebra.



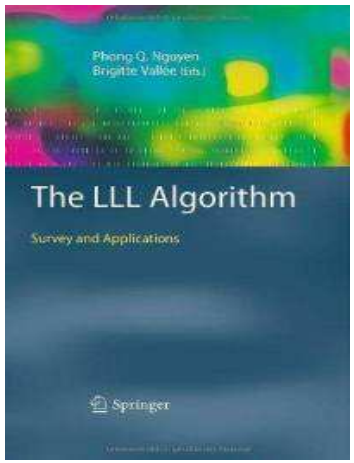


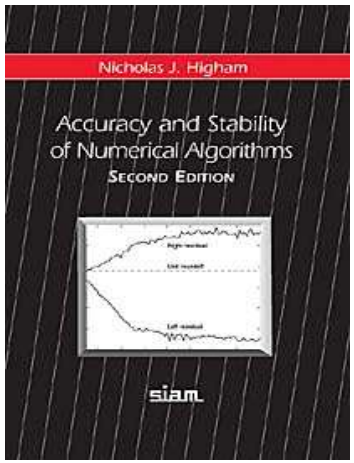
- Chang, S. and Villard. *Perturbation Analysis of the QR factor R in the Context of LLL Lattice Basis Reduction*. Math. Comp.'11.
- Morel, S. and Villard. *H-LLL: Using Householder inside LLL*. ISSAC'09.
- Novocin, S. and Villard. *An LLL-reduction algorithm with quasi-linear time complexity*. STOC'11.

Our contributions to fast LLL reduction

- Costly component: underlying QR/Gram-Schmidt.
- Floating-point arithmetic is well-suited for these [Odlyzko'82].

$\Rightarrow$ Disclose and exploit links between lattice reduction algorithms and numerical linear algebra.





- Chang, S. and Villard. *Perturbation Analysis of the QR factor R in the Context of LLL Lattice Basis Reduction*. Math. Comp.'11.
- Morel, S. and Villard. *H-LLL: Using Householder inside LLL*. ISSAC'09.
- Novocin, S. and Villard. *An LLL-reduction algorithm with quasi-linear time complexity*. STOC'11.

LLL-reduction is not perturbation-friendly

Algorithmic principle: Only the top-most digits contain information, so compute using only these!

Difficulty: LLL-reducedness is not stable under truncations.

$$\begin{bmatrix} 1 & 2^{100}+2^{40} \\ -1 & 2^{100}-2^{40} \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 2^{100} \\ -1 & 2^{100} \end{bmatrix}$$

Not reduced Reduced

LLL-reduction is not perturbation-friendly

Algorithmic principle: Only the top-most digits contain information, so compute using only these!

Difficulty: LLL-reducedness is not stable under truncations.

$$\underbrace{\begin{bmatrix} 1 & 2^{100}+2^{40} \\ -1 & 2^{100}-2^{40} \end{bmatrix}}_{\text{Not reduced}} \Rightarrow \underbrace{\begin{bmatrix} 1 & 2^{100} \\ -1 & 2^{100} \end{bmatrix}}_{\text{Reduced}}$$

LLL-reduction is not perturbation-friendly

Algorithmic principle: Only the top-most digits contain information, so compute using only these!

Difficulty: LLL-reducedness is not stable under truncations.

$$\underbrace{\begin{bmatrix} 1 & 2^{100}+2^{40} \\ -1 & 2^{100}-2^{40} \end{bmatrix}}_{\text{Not reduced}} \Rightarrow \underbrace{\begin{bmatrix} 1 & 2^{100} \\ -1 & 2^{100} \end{bmatrix}}_{\text{Reduced}}$$

Tool: Sensitivity analysis of the R-factor.

$$\underset{\text{non-singular}}{B} = \underset{\text{orthogonal}}{Q} \cdot \underset{\text{up-triangular}}{R} \quad \text{and} \quad \Delta B \text{ small}$$

$$\Downarrow$$

$$\underset{\text{non-singular}}{B+\Delta B} = \underset{\text{orthogonal}}{(Q+\Delta Q)} \cdot \underset{\text{up-triangular}}{(R+\Delta R)} \quad \text{and} \quad \Delta Q, \Delta R \text{ small?}$$

A perturbation-friendly LLL-reduction

Let $\mathrm{cond}(R) = \||R||R^{-1}|\|$. We have:

$$\max \frac{\|\Delta \mathbf{r}_i\|}{\|\mathbf{r}_i\|} \lesssim \mathrm{cond}(R) \cdot \max \frac{\|\Delta \mathbf{b}_i\|}{\|\mathbf{b}_i\|}.$$

- To get meaningful reasults, use precision $> \log_2 \mathrm{cond}(R)$.
- B is LLL-reduced $\Rightarrow \mathrm{cond}(R) = 2^{\mathcal{O}(n)}$.
- Perturb-friendly? allow for columnwise inaccuracy in R.

A perturbation-friendly LLL-reduction

Let $\mathrm{cond}(R) = \||R||R^{-1}|\|$. We have:

$$\max \frac{\|\Delta \mathbf{r}_i\|}{\|\mathbf{r}_i\|} \lesssim \mathrm{cond}(R) \cdot \max \frac{\|\Delta \mathbf{b}_i\|}{\|\mathbf{b}_i\|}.$$

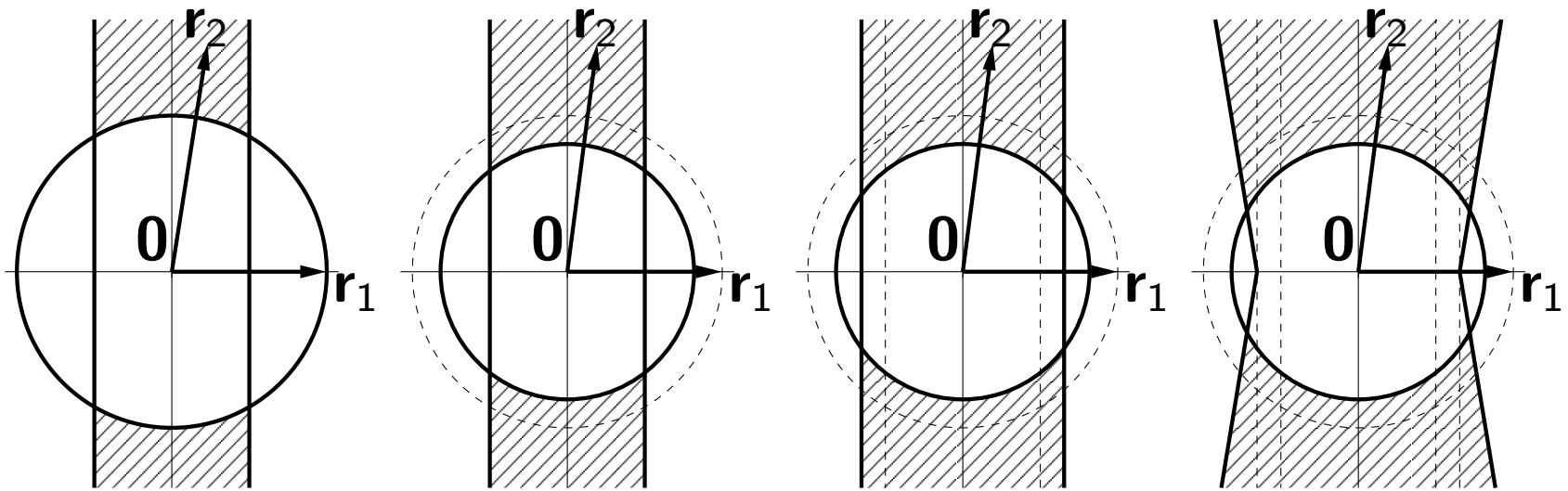
- To get meaningful reasults, use precision $> \log_2 \mathrm{cond}(R)$.
- B is LLL-reduced $\Rightarrow \mathrm{cond}(R) = 2^{\mathcal{O}(n)}$.
- Perturb-friendly? allow for columnwise inaccuracy in R.

A perturbation-friendly LLL-reduction

Let $\mathrm{cond}(R) = \||R||R^{-1}|\|$. We have:

$$\max \frac{\|\Delta \mathbf{r}_i\|}{\|\mathbf{r}_i\|} \lesssim \mathrm{cond}(R) \cdot \max \frac{\|\Delta \mathbf{b}_i\|}{\|\mathbf{b}_i\|}.$$

- To get meaningful reasults, use precision $> \log_2 \mathrm{cond}(R)$.
- B is LLL-reduced $\Rightarrow \mathrm{cond}(R) = 2^{\mathcal{O}(n)}$.
- Perturb-friendly? allow for columnwise inaccuracy in R.



Lowering the bit-complexities

Hybrid approach (L^2 & H-LLL):

- Exact basis B and approximate R-factor.

Totally numeric approach ($\widetilde{L}^1$):

- Approximate basis B and approximate transforms.
- Control granted by gradual feeding [Belabas'04]:
  Move from (almost) reduced to reduced.

Lowering the bit-complexities

Hybrid approach (L^2 & H-LLL):

- Exact basis B and approximate R-factor.

Totally numeric approach ($\widetilde{L}^1$):

- Approximate basis B and approximate transforms.
- Control granted by gradual feeding [Belabas'04]:
  Move from (almost) reduced to reduced.

Lowering the bit-complexities

Hybrid approach (L^2 & H-LLL):

- Exact basis B and approximate R-factor.

Totally numeric approach ($\widetilde{L}^1$):

- Approximate basis B and approximate transforms.
- Control granted by gradual feeding [Belabas'04]:
  Move from (almost) reduced to reduced.

	[LLL'82]	L^2/H-LLL	$\widetilde{L}^1$
Complexity*	$n^{5+\varepsilon}\beta^{2+\varepsilon}$	$n^{4+\varepsilon}\beta^{2}$	$n^{5+\varepsilon}\beta^{1+\varepsilon}$
Precision	$n\beta$	$1.6n/0.8n$	?

* $n =$dim, $\beta = \log \max \|\mathbf{b}_i\|$, $\varepsilon \approx 0$, with $n = \mathcal{O}(\beta)$.

Road-map

1. Euclidean lattices: definitions and algorithmic problems.
2. Reducing lattice bases efficiently.
3. **Paying more to get nicer bases.**
4. Fast lattice-based cryptography.
5. Future directions.

HKZ and BKZ reductions cost more

```
> n := 62; B := RMatrixSpace(Integers(),n,n)!0;
> beta:=1000; for i:=1 to n do
>    B[i][i]:=1; B[i][1]:=RandomBits(beta);
> end for;
> time C := LLL(B:Delta:=0.999);
```

Time: 1.470

```
> time D := HKZ(C);
```

Time: 3389.650

```
> RealField(3) ! Sqrt( Norm(C[1])/Norm(D[1]) );
```

1.69

HKZ and BKZ reductions cost more

```
> n := 62; B := RMatrixSpace(Integers(),n,n)!0;
> beta:=1000; for i:=1 to n do
>    B[i][i]:=1; B[i][1]:=RandomBits(beta);
> end for;
> time C := LLL(B:Delta:=0.999);
```

Time: 1.470

```
> time D := HKZ(C);
```

Time: 3389.650

```
> RealField(3) ! Sqrt( Norm(C[1])/Norm(D[1]) );
```

1.69

- The time and output norm gaps grow exponentially with respect to the dimension n.
- One can trade quality for time, using BKZ reduction.

Our contributions to strong reductions

- Several known algorithms for HKZ-reduction.
  Most practical one: Kannan-Fincke-Pohst.
- Several known trade-offs between HKZ and LLL.
  Most practical one: Schnorr-Euchner BKZ.

$\Rightarrow$ Measure cost and progress with the R-factor diagonal.

$$(r_{ii})_{i \leq n} = (\|\mathbf{b}_i^*\|)_{i \leq n} \text{ is everything.}$$

- Hanrot and S. *Improved Analysis of Kannan's Shortest Lattice Vector Algorithm.* CRYPTO'07.
- Hanrot, Pujol and S. *Analyzing Blockwise Lattice Algorithms using Dynamical Systems.* CRYPTO'11.

Our contributions to strong reductions

- Several known algorithms for HKZ-reduction.
  Most practical one: Kannan-Fincke-Pohst.
- Several known trade-offs between HKZ and LLL.
  Most practical one: Schnorr-Euchner BKZ.

$\Rightarrow$ Measure cost and progress with the R-factor diagonal.

$$(r_{ii})_{i \leq n} = (\|\mathbf{b}_i^*\|)_{i \leq n} \text{ is everything.}$$

- Hanrot and S. *Improved Analysis of Kannan's Shortest Lattice Vector Algorithm.* CRYPTO'07.
- Hanrot, Pujol and S. *Analyzing Blockwise Lattice Algorithms using Dynamical Systems.* CRYPTO'11.

Our contributions to strong reductions

- Several known algorithms for HKZ-reduction.
  Most practical one: Kannan-Fincke-Pohst.
- Several known trade-offs between HKZ and LLL.
  Most practical one: Schnorr-Euchner BKZ.

$\Rightarrow$ Measure cost and progress with the R-factor diagonal.

$$(r_{ii})_{i \leq n} = (\|\mathbf{b}_i^*\|)_{i \leq n} \text{ is everything.}$$

- Hanrot and S. *Improved Analysis of Kannan's Shortest Lattice Vector Algorithm.* CRYPTO'07.
- Hanrot, Pujol and S. *Analyzing Blockwise Lattice Algorithms using Dynamical Systems.* CRYPTO'11.

A static analysis for HKZ

- Kannan's HKZ algorithm consists in:
  - lower-dimensional HKZ reductions,
  - computations of shortest lattice vectors.
- Shortest vectors via Kannan-Fincke-Pohst: intertwinned enumerations of all short points of projected lattices.

Gaussian heuristic: $|L \cap \mathcal{B}| \approx \mathrm{vol}(\mathcal{B}) \ / \ \det(L)$.

Let $B = QR$. Enumerating all $\mathbf{b} \in L(B)$ with $\|\mathbf{b}\| \leq A$ costs:

$$\leq \ 2^{\mathcal{O}(n)} \cdot \prod_{i \leq n} \max\left(1, \frac{A}{\sqrt{n} \cdot r_{ii}}\right).$$

Within Kannan's HKZ algorithm, this is $\leq n^{n/(2e)+o(1)}$.

A static analysis for HKZ

- Kannan's HKZ algorithm consists in:
  - lower-dimensional HKZ reductions,
  - computations of shortest lattice vectors.
- Shortest vectors via Kannan-Fincke-Pohst: intertwinned enumerations of all short points of projected lattices.

Gaussian heuristic: $|L \cap \mathcal{B}| \approx \mathrm{vol}(\mathcal{B}) \ / \ \det(L)$.

Let $B = QR$. Enumerating all $\mathbf{b} \in L(B)$ with $\|\mathbf{b}\| \leq A$ costs:

$$\leq \ 2^{\mathcal{O}(n)} \cdot \prod_{i \leq n} \max\left(1, \frac{A}{\sqrt{n} \cdot r_{ii}}\right).$$

Within Kannan's HKZ algorithm, this is $\leq n^{n/(2e) + o(1)}$.

A static analysis for HKZ

- Kannan's HKZ algorithm consists in:
  - lower-dimensional HKZ reductions,
  - computations of shortest lattice vectors.
- Shortest vectors via Kannan-Fincke-Pohst: intertwinned enumerations of all short points of projected lattices.

Gaussian heuristic: $|L \cap \mathcal{B}| \approx \mathrm{vol}(\mathcal{B}) \ / \ \det(L)$.

Let $B = QR$. Enumerating all $\mathbf{b} \in L(B)$ with $\|\mathbf{b}\| \leq A$ costs:

$$\leq \ 2^{\mathcal{O}(n)} \ \cdot \ \prod_{i \leq n} \max\left(1, \frac{A}{\sqrt{n} \cdot r_{ii}}\right).$$

Within Kannan's HKZ algorithm, this is $\leq n^{n/(2e) \ + \ o(1)}$.

A dynamic analysis for BKZ

- BKZ_k proceeds by k-dimensional HKZ reductions, performed circularly on the diagonal of the R-factor.

$\Rightarrow$ Let's look at the evolution of the r_{ii}'s!

- [Madritsch-Vallée'10]: In LLL, the log r_{ii}'s evolve like a sandpile.

A dynamic analysis for BKZ

- BKZ_k proceeds by k-dimensional HKZ reductions, performed circularly on the diagonal of the R-factor.

$\Rightarrow$ Let's look at the evolution of the r_{ii}'s!

- [Madritsch-Vallée'10]: In LLL, the log r_{ii}'s evolve like a sandpile.

A dynamic analysis for BKZ

- BKZ_k proceeds by k-dimensional HKZ reductions, performed circularly on the diagonal of the R-factor.

$\Rightarrow$ Let's look at the evolution of the r_{ii}'s!

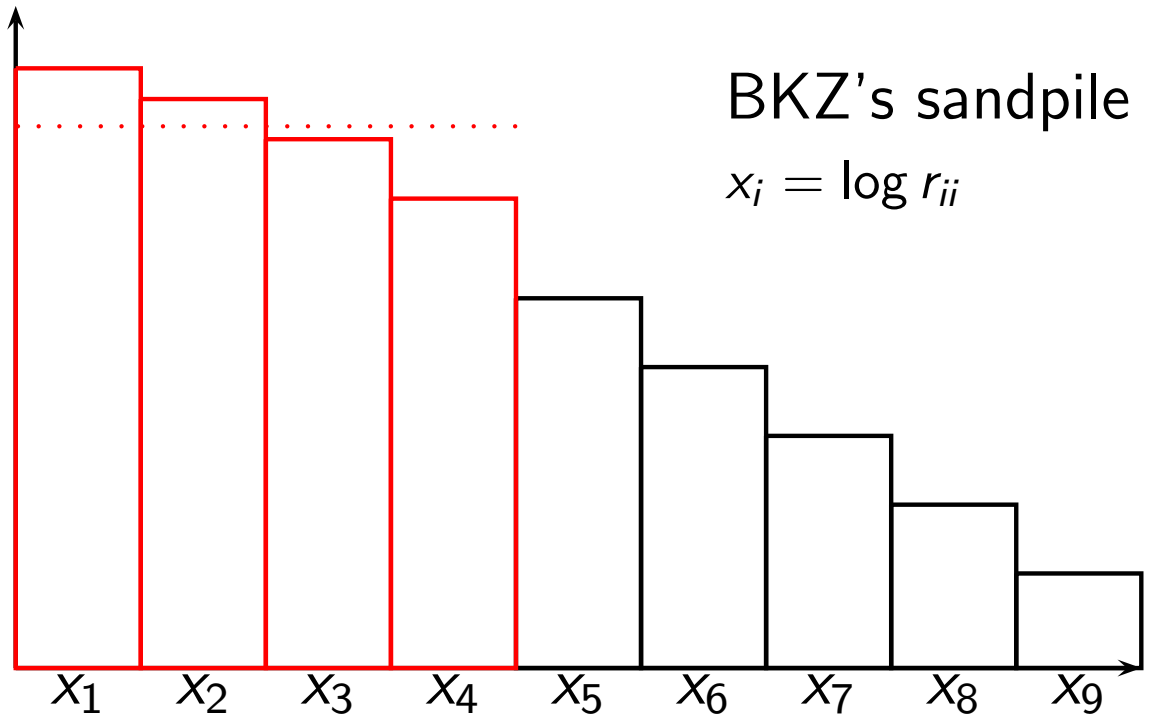
- [Madritsch-Vallée'10]: In LLL, the log r_{ii}'s evolve like a sandpile.

A dynamic analysis for BKZ

- BKZ_k proceeds by k-dimensional HKZ reductions, performed circularly on the diagonal of the R-factor.

$\Rightarrow$ Let's look at the evolution of the r_{ii}'s!

- [Madritsch-Vallée'10]: In LLL, the $\log r_{ii}$'s evolve like a sandpile.

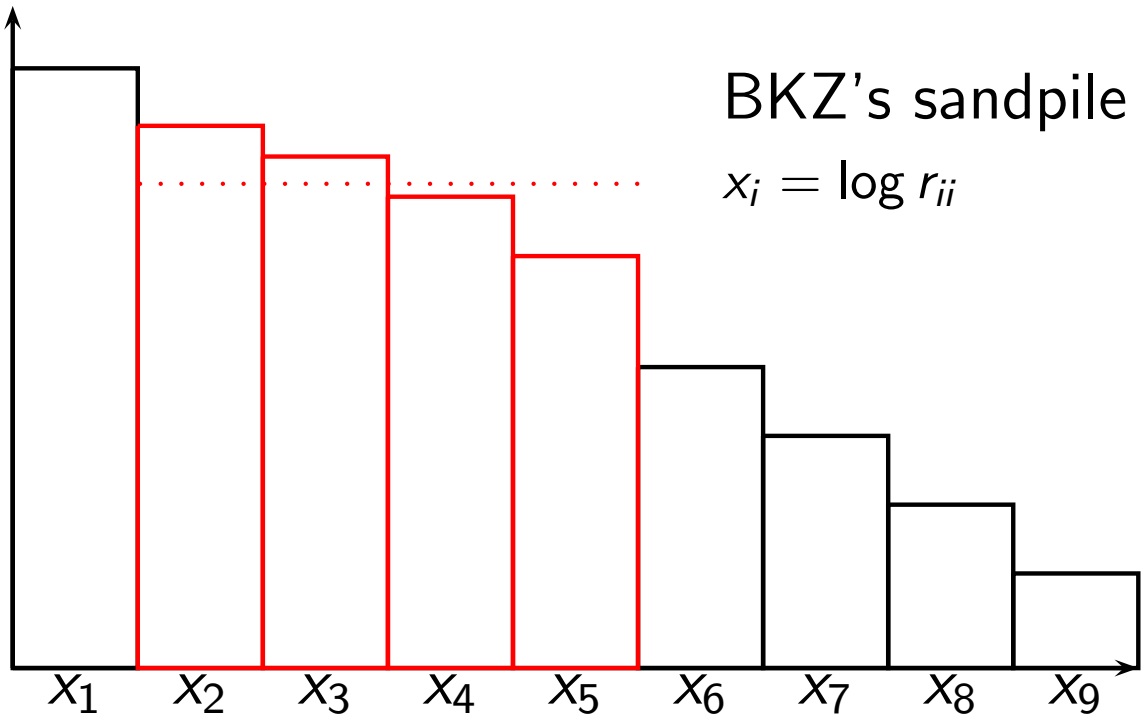


A dynamic analysis for BKZ

- BKZ_k proceeds by k-dimensional HKZ reductions, performed circularly on the diagonal of the R-factor.

$\Rightarrow$ Let's look at the evolution of the r_{ii}'s!

- [Madritsch-Vallée'10]: In LLL, the $\log r_{ii}$'s evolve like a sandpile.

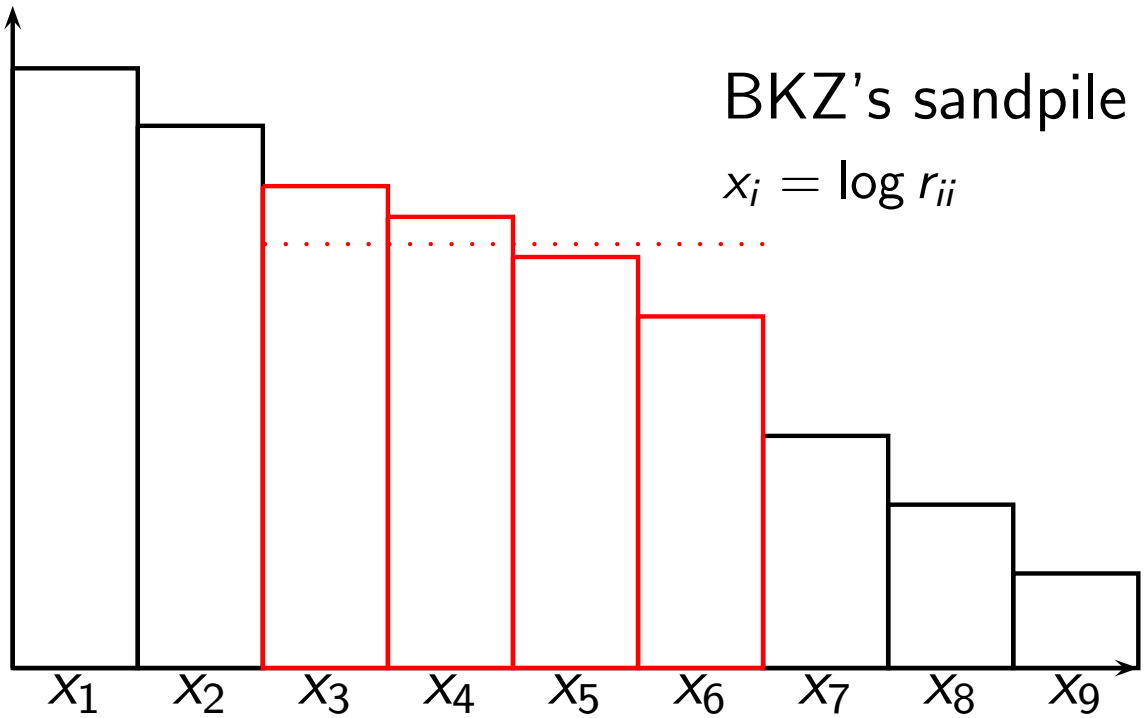


A dynamic analysis for BKZ

- BKZ_k proceeds by k-dimensional HKZ reductions, performed circularly on the diagonal of the R-factor.

$\Rightarrow$ Let's look at the evolution of the r_{ii}'s!

- [Madritsch-Vallée'10]: In LLL, the $\log r_{ii}$'s evolve like a sandpile.

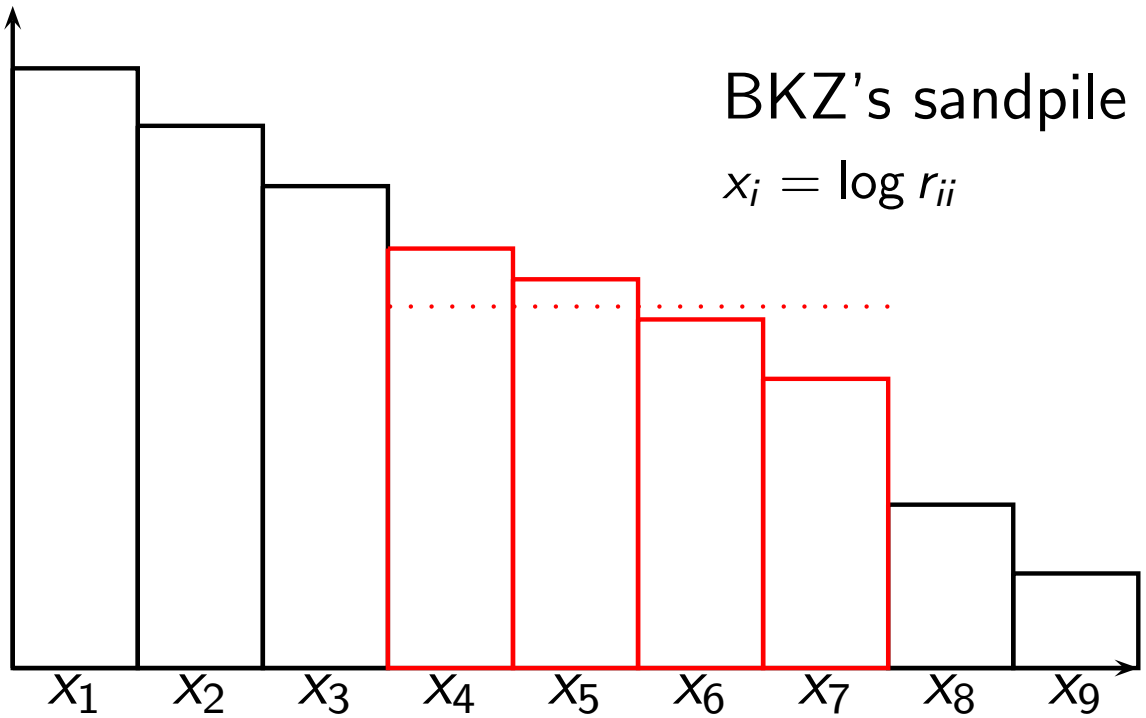


A dynamic analysis for BKZ

- BKZ_k proceeds by k-dimensional HKZ reductions, performed circularly on the diagonal of the R-factor.

$\Rightarrow$ Let's look at the evolution of the r_{ii}'s!

- [Madritsch-Vallée'10]: In LLL, the $\log r_{ii}$'s evolve like a sandpile.

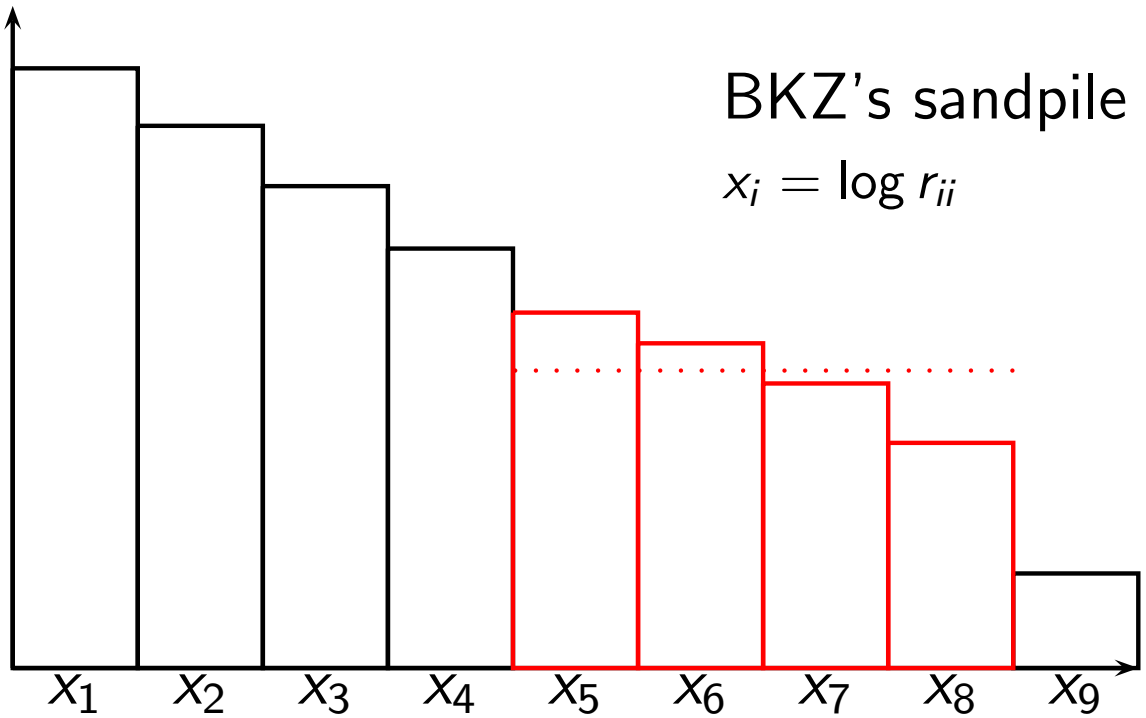


A dynamic analysis for BKZ

- BKZ_k proceeds by k-dimensional HKZ reductions, performed circularly on the diagonal of the R-factor.

$\Rightarrow$ Let's look at the evolution of the r_{ii}'s!

- [Madritsch-Vallée'10]: In LLL, the $\log r_{ii}$'s evolve like a sandpile.

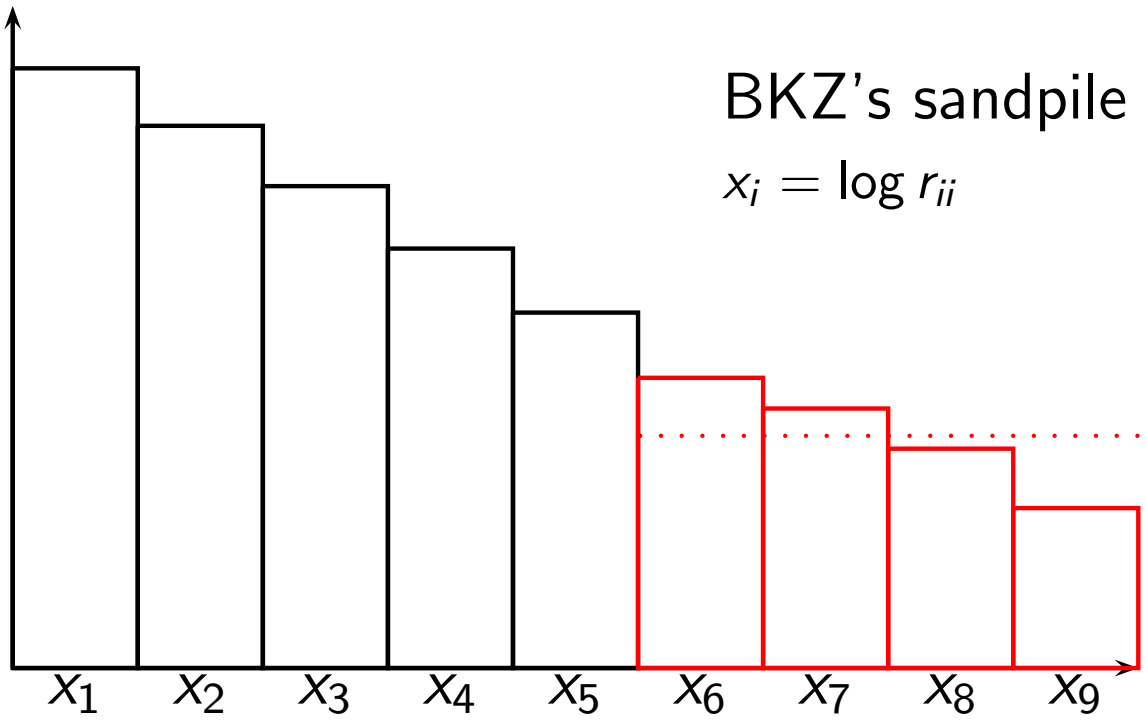


A dynamic analysis for BKZ

- BKZ_k proceeds by k-dimensional HKZ reductions, performed circularly on the diagonal of the R-factor.

$\Rightarrow$ Let's look at the evolution of the r_{ii}'s!

- [Madritsch-Vallée'10]: In LLL, the $\log r_{ii}$'s evolve like a sandpile.

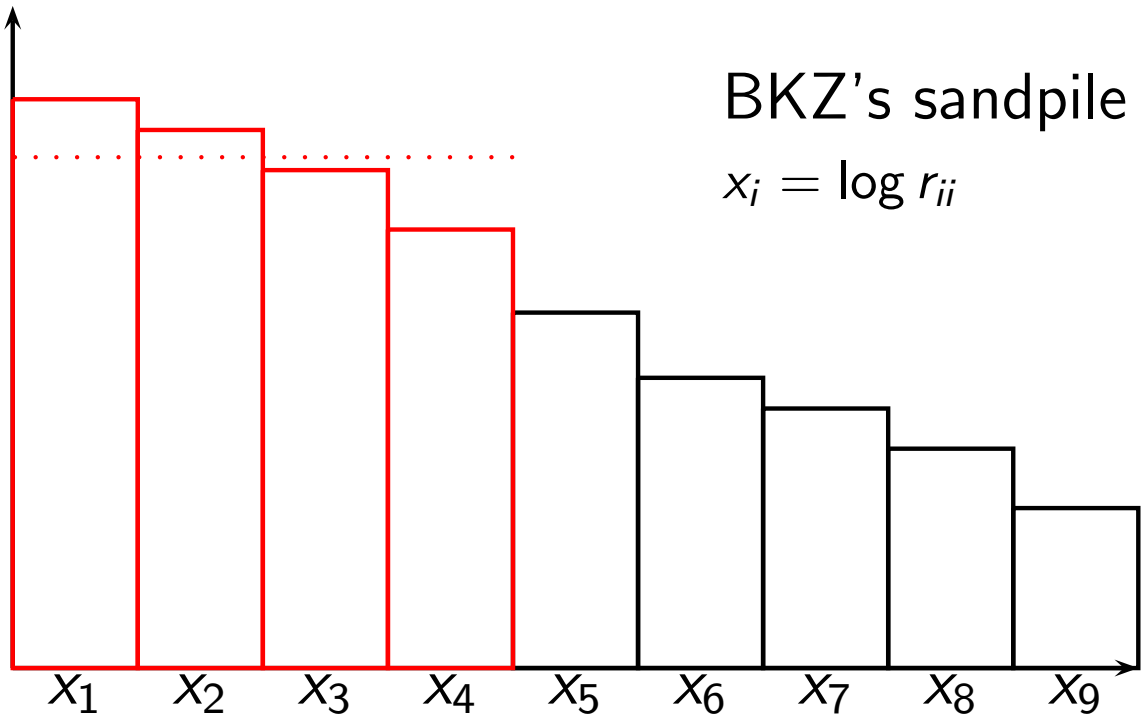


A dynamic analysis for BKZ

- BKZ_k proceeds by k-dimensional HKZ reductions, performed circularly on the diagonal of the R-factor.

$\Rightarrow$ Let's look at the evolution of the r_{ii}'s!

- [Madritsch-Vallée'10]: In LLL, the $\log r_{ii}$'s evolve like a sandpile.



Analyzing BKZ's sandpile

A BKZ tour on the sandpile: $X' \lesssim AX + \Gamma$.

- A: successive averages.
- Γ: combinations of Hermite constants.
- $\lesssim$: can be made rigorous via amortizing.

Behavior of BKZ $\leftrightarrow$ discrete-time affine dynamical system.

After $\mathcal{O}\left(\frac{n^3}{k^2}\log(\frac{n\beta}{\varepsilon})\right)$ calls to HKZ_k, BKZ_k returns C s.t.:

$$\mathrm{HF}(C) \leq (1+\varepsilon)\cdot k^{\frac{n-1}{2(k-1)}+\frac{3}{2}}.$$

Analyzing BKZ's sandpile

A BKZ tour on the sandpile: $X' \lesssim AX + \Gamma$.

- A: successive averages.
- Γ: combinations of Hermite constants.
- $\lesssim$: can be made rigorous via amortizing.

Behavior of BKZ $\leftrightarrow$ discrete-time affine dynamical system.

After $\mathcal{O}\left(\frac{n^3}{k^2}\log(\frac{n\beta}{\varepsilon})\right)$ calls to HKZ_k, BKZ_k returns C s.t.:

$$\mathrm{HF}(C) \leq (1+\varepsilon)\cdot k^{\frac{n-1}{2(k-1)}+\frac{3}{2}}.$$

Analyzing BKZ's sandpile

A BKZ tour on the sandpile: $X' \lesssim AX + \Gamma$.

- A: successive averages.
- Γ: combinations of Hermite constants.
- $\lesssim$: can be made rigorous via amortizing.

Behavior of BKZ $\leftrightarrow$ discrete-time affine dynamical system.

After $\mathcal{O}\left(\frac{n^3}{k^2}\log(\frac{n\beta}{\varepsilon})\right)$ calls to HKZ_k, BKZ_k returns C s.t.:

$$\mathrm{HF}(C) \leq (1+\varepsilon)\cdot k^{\frac{n-1}{2(k-1)}+\frac{3}{2}}.$$

Road-map

1. Euclidean lattices: definitions and algorithmic problems.
2. Reducing lattice bases efficiently.
3. Paying more to get nicer bases.
4. **Fast lattice-based cryptography.**
5. Future directions.

Lattice-based cryptography

- Cryptography: science of securing digital information.
- Design methodology: exploit the appearant hardness of an algorithmic problem to create a computational gap between valid and malicious parties.
- Finding very short bases seems exponentially hard.

Two opposite strategies in lattice-based crypto:

- NTRU: Superfast schemes, but heuristic security.
- [Ajtai'96, Regev'05, ...]: Somewhat inefficient schemes, but provably as secure as worst-case lattice problems.

Lattice-based cryptography

- Cryptography: science of securing digital information.
- Design methodology: exploit the appearant hardness of an algorithmic problem to create a computational gap between valid and malicious parties.
- Finding very short bases seems exponentially hard.

Two opposite strategies in lattice-based crypto:

- NTRU: Superfast schemes, but heuristic security.
- [Ajtai'96,Regev'05,...]: Somewhat inefficient schemes, but provably as secure as worst-case lattice problems.

Lattice-based cryptography

- Cryptography: science of securing digital information.
- Design methodology: exploit the appearant hardness of an algorithmic problem to create a computational gap between valid and malicious parties.
- Finding very short bases seems exponentially hard.

Two opposite strategies in lattice-based crypto:

- NTRU: Superfast schemes, but heuristic security.
- [Ajtai'96,Regev'05,...]: Somewhat inefficient schemes, but provably as secure as worst-case lattice problems.

Lattice-based cryptography

- Cryptography: science of securing digital information.
- Design methodology: exploit the appearant hardness of an algorithmic problem to create a computational gap between valid and malicious parties.
- Finding very short bases seems exponentially hard.

Two opposite strategies in lattice-based crypto:

- NTRU: Superfast schemes, but heuristic security.
- [Ajtai'96,Regev'05,...]: Somewhat inefficient schemes, but provably as secure as worst-case lattice problems.

Our contributions to lattice-based cryptography

Context: The rigorous approach becomes more efficient.

- Use of polynomial rings and ideal lattices.
- [Lyubashevsky-Micciancio'06-'08,Peikert-Rosen'06]: hash functions and digital signatures with quasi-optimal complexities.

$\Rightarrow$ Use polynomial rings and ideal lattices for encryption.

Fast public key encryption, semantically secure under (quantum) worst-case hardness assumptions for ideal lattices.

- S., Steinfeld, Tanaka and Xagawa. *Efficient Public-Key Encryption Based on Ideal Lattices*, ASIACRYPT'09.
- S. and Steinfeld. *Making NTRU as secure as worst-case problems over ideal lattices*, EUROCRYPT'11.

Our contributions to lattice-based cryptography

Context: The rigorous approach becomes more efficient.

- Use of polynomial rings and ideal lattices.
- [Lyubashevsky-Micciancio'06-'08,Peikert-Rosen'06]: hash functions and digital signatures with quasi-optimal complexities.

$\Rightarrow$ Use polynomial rings and ideal lattices for encryption.

Fast public key encryption, semantically secure under (quantum) worst-case hardness assumptions for ideal lattices.

- S., Steinfeld, Tanaka and Xagawa. *Efficient Public-Key Encryption Based on Ideal Lattices*, ASIACRYPT'09.
- S. and Steinfeld. *Making NTRU as secure as worst-case problems over ideal lattices*, EUROCRYPT'11.

Our contributions to lattice-based cryptography

Context: The rigorous approach becomes more efficient.

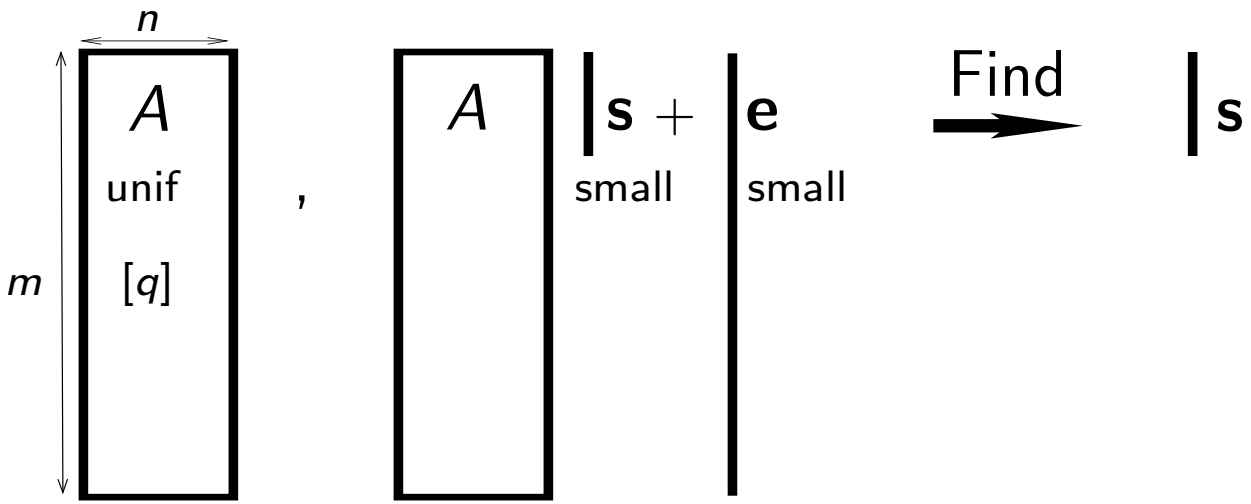
- Use of polynomial rings and ideal lattices.
- [Lyubashevsky-Micciancio'06-'08,Peikert-Rosen'06]: hash functions and digital signatures with quasi-optimal complexities.

$\Rightarrow$ Use polynomial rings and ideal lattices for encryption.

Fast public key encryption, semantically secure under (quantum) worst-case hardness assumptions for ideal lattices.

- S., Steinfeld, Tanaka and Xagawa. *Efficient Public-Key Encryption Based on Ideal Lattices*, ASIACRYPT'09.
- S. and Steinfeld. *Making NTRU as secure as worst-case problems over ideal lattices*, EUROCRYPT'11.

The Learning With Errors Problem

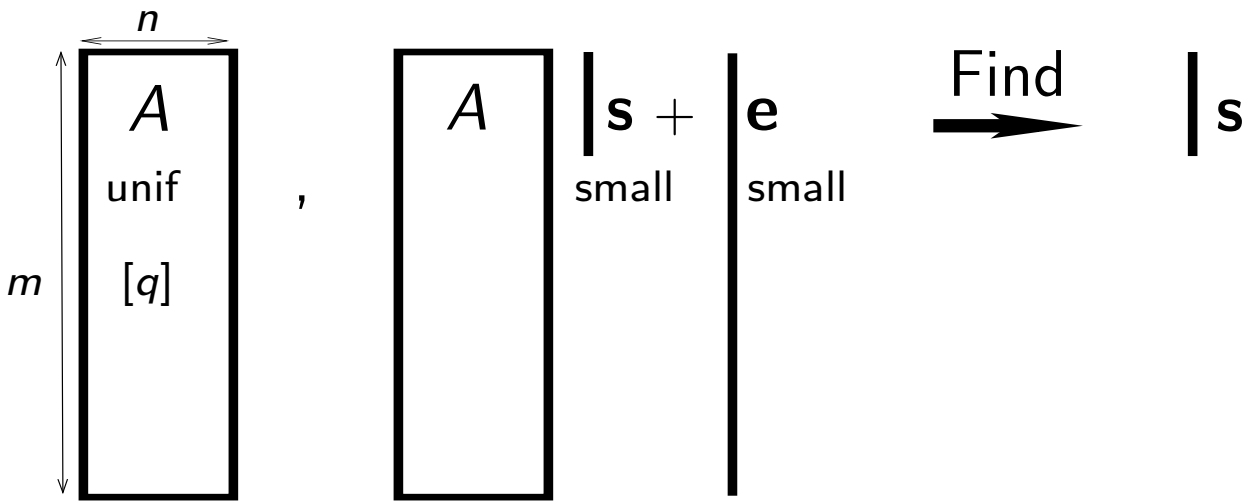


- $A \hookleftarrow U(\mathbb{Z}_q^{m \times n})$ public.
- $\mathbf{s} \in \mathbb{Z}_q^n$ small, to be found .
- $\mathbf{e} \in \mathbb{Z}_q^m$: small Gaussian noise, unknown.

(Computational)-LWE [Regev'05]

[Technical conditions on the parameters]
LWE is no easier than finding short bases for arbitrary lattices.

The Learning With Errors Problem

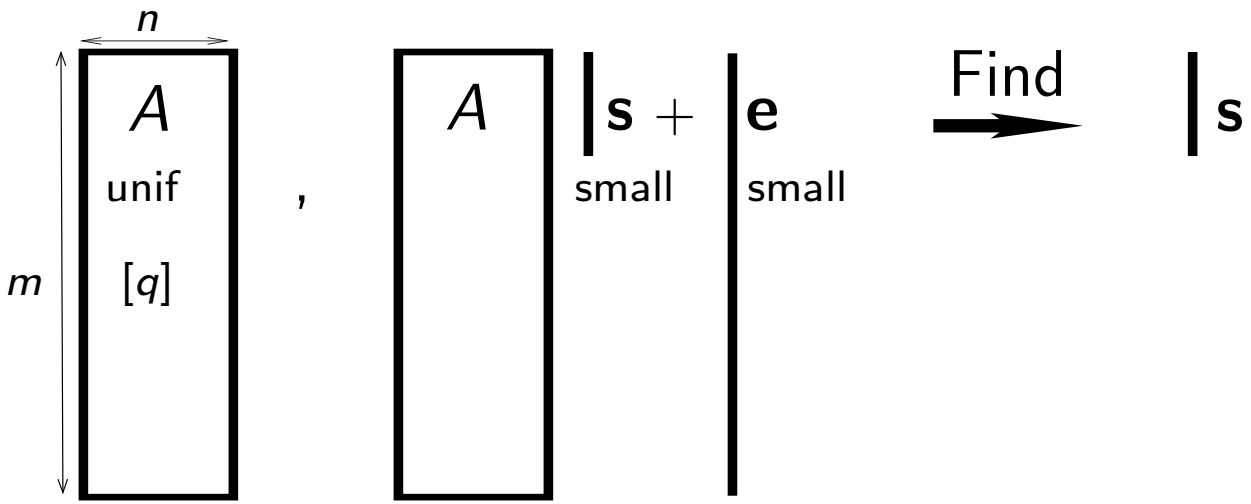


- $A \hookleftarrow U(\mathbb{Z}_q^{m \times n})$ public.
- $\mathbf{s} \in \mathbb{Z}_q^n$ small, to be found .
- $\mathbf{e} \in \mathbb{Z}_q^m$: small Gaussian noise, unknown.

(Computational)-LWE [Regev'05]

[Technical conditions on the parameters]
LWE is no easier than finding short bases for arbitrary lattices.

The Learning With Errors Problem

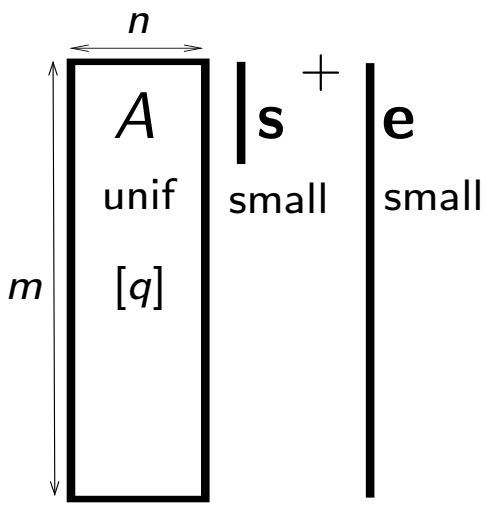


- $A \hookleftarrow U(\mathbb{Z}_q^{m \times n})$ public.
- $\mathbf{s} \in \mathbb{Z}_q^n$ small, to be found .
- $\mathbf{e} \in \mathbb{Z}_q^m$: small Gaussian noise, unknown.

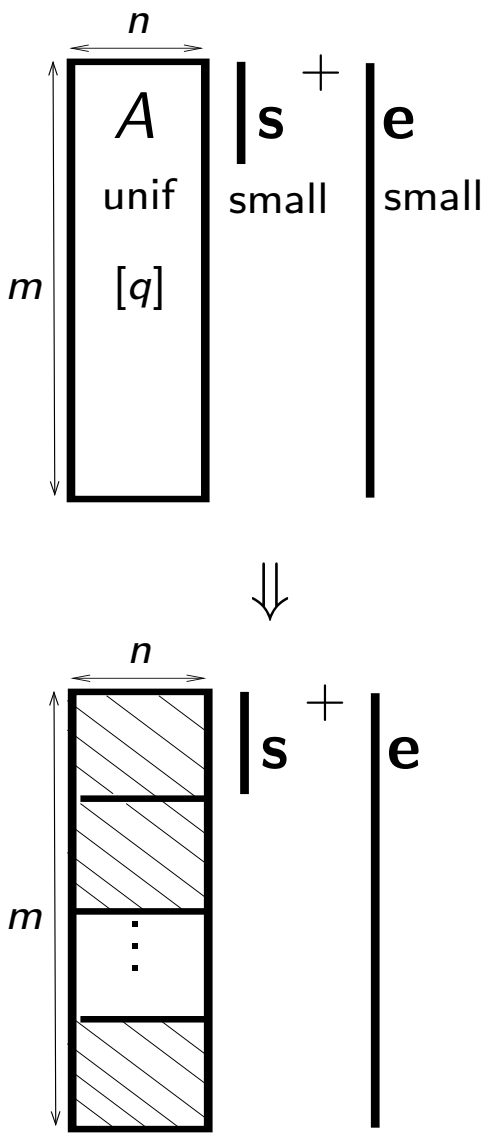
(Computational)-LWE [Regev'05]

[Technical conditions on the parameters]
LWE is no easier than finding short bases for arbitrary lattices.

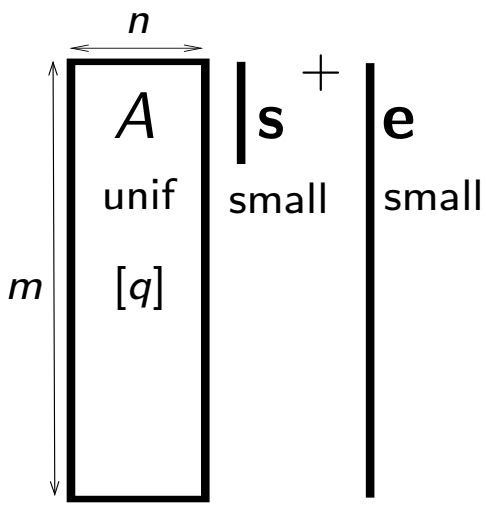
A structured LWE problem

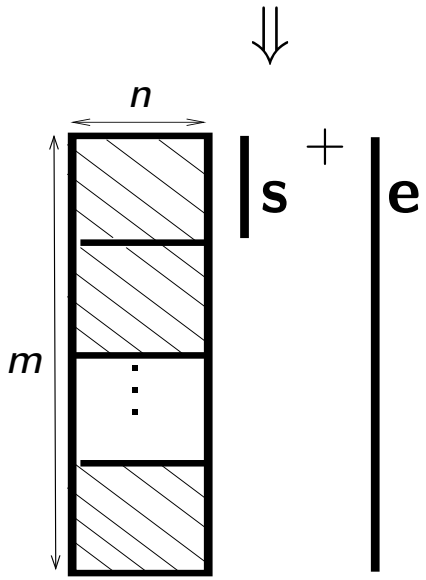


A structured LWE problem



A structured LWE problem





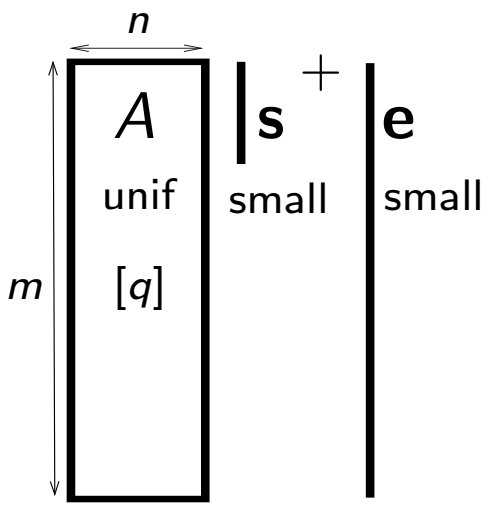
- Structured matrices $\Rightarrow$ faster operations.
- Structured matrices $\leftrightarrow$ polynomials.
- Here: $R_q = \dfrac{\mathbb{Z}_q[x]}{x^n + 1}$.

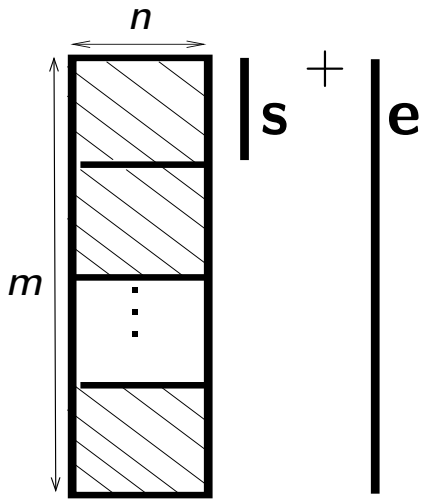
$\Rightarrow$ One-way function, at least as hard to invert as worst-case problems for ideal lattices.

- Ideal lattices $\leftrightarrow$ ideals of $R = \dfrac{\mathbb{Z}[x]}{x^n + 1}$.

$\Rightarrow$ With a trapdoor & generic hard-core bits: PK-encryption with quasi-optimal efficiency.

A structured LWE problem





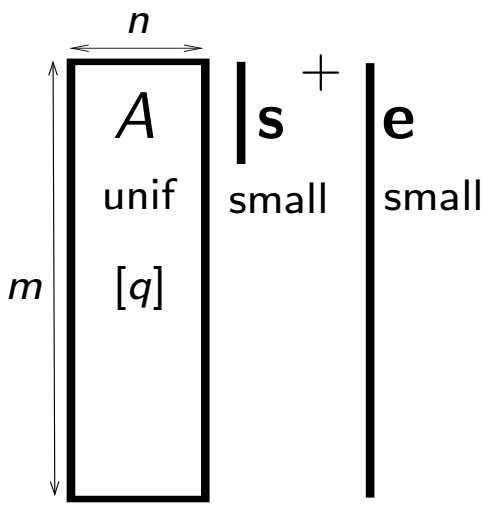
- Structured matrices $\Rightarrow$ faster operations.
- Structured matrices $\leftrightarrow$ polynomials.
- Here: $R_q = \dfrac{\mathbb{Z}_q[x]}{x^n + 1}$.

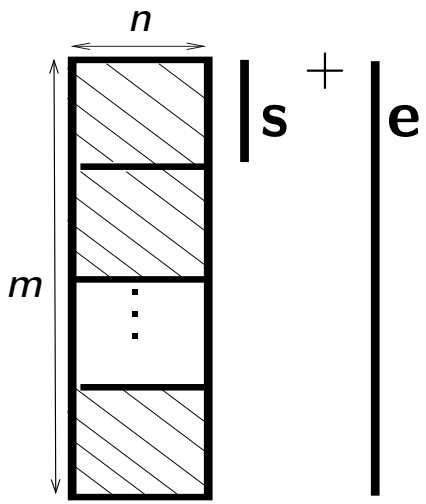
$\Rightarrow$ One-way function, at least as hard to invert as worst-case problems for ideal lattices.

- Ideal lattices $\leftrightarrow$ ideals of $R = \dfrac{\mathbb{Z}[x]}{x^n + 1}$.

$\Rightarrow$ With a trapdoor & generic hard-core bits: PK-encryption with quasi-optimal efficiency.

A structured LWE problem





- Structured matrices $\Rightarrow$ faster operations.
- Structured matrices $\leftrightarrow$ polynomials.
- Here: $R_q = \dfrac{\mathbb{Z}_q[x]}{x^n + 1}$.

$\Rightarrow$ One-way function, at least as hard to invert as worst-case problems for ideal lattices.

- Ideal lattices $\leftrightarrow$ ideals of $R = \dfrac{\mathbb{Z}[x]}{x^n + 1}$.

$\Rightarrow$ With a trapdoor & generic hard-core bits: PK-encryption with quasi-optimal efficiency.

Decisional version of Ring-LWE

- a uniform in $R_q = \frac{\mathbb{Z}_q[x]}{x^n+1}$, known.
- $s, e \in R_q$, small and secret.

Comp-RLWE: $(a, a \cdot s + e) \rightarrow s$

$\Rightarrow$ Dec-RLWE: $(a, a \cdot s + e) \approx^c U(R_q \times R_q)$?

(Decisional)-RLWE [Lyubashevsky-Peikert-Regev'10]

[Technical conditions on parameters]
If finding short bases for arbitrary ideal lattices is hard, then $(a, a \cdot s + e)$ is computationally indistinguishable from uniform.

Decisional version of Ring-LWE

- a uniform in $R_q = \frac{\mathbb{Z}_q[x]}{x^n+1}$, known.
- $s, e \in R_q$, small and secret.

$$\begin{array}{lccc} \text{Comp-RLWE:} & (a, a \cdot s + e) & \rightarrow & s \\ \Rightarrow \text{Dec-RLWE:} & (a, a \cdot s + e) & \approx^c & U(R_q \times R_q)\ ? \end{array}$$

(Decisional)-RLWE [Lyubashevsky-Peikert-Regev'10]

[Technical conditions on parameters]
If finding short bases for arbitrary ideal lattices is hard, then $(a, a \cdot s + e)$ is computationally indistinguishable from uniform.

Decisional version of Ring-LWE

- a uniform in $R_q = \frac{\mathbb{Z}_q[x]}{x^n+1}$, known.
- $s, e \in R_q$, small and secret.

$$\begin{array}{lccc} \text{Comp-RLWE:} & (a, a \cdot s + e) & \rightarrow & s \\ \Rightarrow \text{Dec-RLWE:} & (a, a \cdot s + e) & \approx^c & U(R_q \times R_q) \ ? \end{array}$$

(Decisional)-RLWE [Lyubashevsky-Peikert-Regev'10]

[Technical conditions on parameters]
If finding short bases for arbitrary ideal lattices is hard, then $(a, a \cdot s + e)$ is computationally indistinguishable from uniform.

Using RLWE for NTRU

(Standard) `NTRUEncrypt`:

- Secret key: f, g small and $f = 1$ [2].
- Public key: $h = g/f \in R_q$, heuristically looks uniform.

- Enc: $M \mapsto C = 2hs + M$ [q], with s small & random.
- Dec: $fC = 2gs + fM$ is small $\Rightarrow$ Take it mod 2.

- Use RLWE to make C indistinguishable from uniform!
- Difficulty: RLWE hardness result requires h uniform.
- Obtained by sampling f, g from discrete Gaussians.

Using RLWE for NTRU

(Standard) `NTRUEncrypt`:

- Secret key: f, g small and $f = 1$ [2].
- Public key: $h = g/f \in R_q$, heuristically looks uniform.

- Enc: $M \mapsto C = 2hs + M$ [q], with s small & random.
- Dec: $fC = 2gs + fM$ is small $\Rightarrow$ Take it mod 2.

- Use RLWE to make C indistinguishable from uniform!
- Difficulty: RLWE hardness result requires h uniform.
- Obtained by sampling f, g from discrete Gaussians.

Using RLWE for NTRU

(Standard) NTRUEncrypt:

- Secret key: f, g small and $f = 1$ [2].
- Public key: $h = g/f \in R_q$, heuristically looks uniform.

- Enc: $M \mapsto C = 2hs + M$ [q], with s small & random.
- Dec: $fC = 2gs + fM$ is small $\Rightarrow$ Take it mod 2.

- Use RLWE to make C indistinguishable from uniform!
- Difficulty: RLWE hardness result requires h uniform.
- Obtained by sampling f, g from discrete Gaussians.

Using RLWE for NTRU

(Modified) NTRUEncrypt:

- Secret key: f, g small and $f = 1$ [2].
- Public key: $h = g/f \in R_q$, heuristically looks uniform.

- Enc: $M \mapsto C = 2(hs + e) + M$ [q], with s, e small & random.
- Dec: $fC = 2(gs + fe) + fM$ is small $\Rightarrow$ Take it mod 2.

- Use RLWE to make C indistinguishable from uniform!
- Difficulty: RLWE hardness result requires h uniform.
- Obtained by sampling f, g from discrete Gaussians.

Using RLWE for NTRU

(Modified) NTRUEncrypt:

- Secret key: f, g small and $f = 1$ [2].
- Public key: $h = g/f \in R_q$, heuristically looks uniform.

- Enc: $M \mapsto C = 2(hs + e) + M$ [q], with s, e small & random.
- Dec: $fC = 2(gs + fe) + fM$ is small $\Rightarrow$ Take it mod 2.

- Use RLWE to make C indistinguishable from uniform!
- Difficulty: RLWE hardness result requires h uniform.
- Obtained by sampling f, g from discrete Gaussians.

Using RLWE for NTRU

(Modified) NTRUEncrypt:

- Secret key: f, g small and $f = 1$ [2].
- Public key: $h = g/f \in R_q$, heuristically looks uniform.

- Enc: $M \mapsto C = 2(hs + e) + M$ [q], with s, e small & random.
- Dec: $fC = 2(gs + fe) + fM$ is small $\Rightarrow$ Take it mod 2.

- Use RLWE to make C indistinguishable from uniform!
- Difficulty: RLWE hardness result requires h uniform.
- Obtained by sampling f, g from discrete Gaussians.

Using RLWE for NTRU

(Modified) NTRUEncrypt:

- Secret key: f, g small Gaussian and $f = 1$ [2].
- Public key: $h = g/f \in R_q$, provably is uniform.

- Enc: $M \mapsto C = 2(hs + e) + M$ [q], with s, e small & random.
- Dec: $fC = 2(gs + fe) + fM$ is small $\Rightarrow$ Take it mod 2.

- Use RLWE to make C indistinguishable from uniform!
- Difficulty: RLWE hardness result requires h uniform.
- Obtained by sampling f, g from discrete Gaussians.

Using RLWE for NTRU

(Modified) NTRUEncrypt:

- Secret key: f, g small Gaussian and $f = 1$ [2].
- Public key: $h = g/f \in R_q$, provably is uniform.

- Enc: $M \mapsto C = 2(hs + e) + M$ [q], with s, e small & random.
- Dec: $fC = 2(gs + fe) + fM$ is small $\Rightarrow$ Take it mod 2.

- Use RLWE to make C indistinguishable from uniform!
- Difficulty: RLWE hardness result requires h uniform.
- Obtained by sampling f, g from discrete Gaussians.

The modified NTRU is secure, and asymptotically efficient.

Road-map

1. Euclidean lattices: definitions and algorithmic problems.
2. Reducing lattice bases efficiently.
3. Paying more to get nicer bases.
4. Fast lattice-based cryptography.
5. **Future directions.**

Faster LLL-type reductions

Target: LLL as fast as matrix multiplication.

Considering the linear algebra contribution to the cost:

- We decreased the cost wrt $\beta = \log \max \|\mathbf{b}_i\|$.
- There exist strategies to decrease the cost wrt n:
  [Schönhage'84,Storjohann'96,Koy-Schnorr'01].
- Are these improvements compatible?

Breaking the linear precision barrier:

- Current numeric approach: $\Omega(n)$ bits of precision.
- What can we do with less?

Faster LLL-type reductions

Target: LLL as fast as matrix multiplication.

Considering the linear algebra contribution to the cost:

- We decreased the cost wrt $\beta = \log \max \|\mathbf{b}_i\|$.
- There exist strategies to decrease the cost wrt n: [Schönhage'84,Storjohann'96,Koy-Schnorr'01].
- Are these improvements compatible?

Breaking the linear precision barrier:

- Current numeric approach: $\Omega(n)$ bits of precision.
- What can we do with less?

Faster LLL-type reductions

Target: LLL as fast as matrix multiplication.

Considering the linear algebra contribution to the cost:

- We decreased the cost wrt $\beta = \log \max \|\mathbf{b}_i\|$.
- There exist strategies to decrease the cost wrt n: [Schönhage'84,Storjohann'96,Koy-Schnorr'01].
- Are these improvements compatible?

Breaking the linear precision barrier:

- Current numeric approach: $\Omega(n)$ bits of precision.
- What can we do with less?

Faster LLL-type reductions

Target: LLL as fast as matrix multiplication.

Considering the linear algebra contribution to the cost:

- We decreased the cost wrt $\beta = \log \max \|\mathbf{b}_i\|$.
- There exist strategies to decrease the cost wrt n: [Schönhage'84,Storjohann'96,Koy-Schnorr'01].
- Are these improvements compatible?

Breaking the linear precision barrier:

- Current numeric approach: $\Omega(n)$ bits of precision.
- What can we do with less?

Faster strong reductions

Sub-exponential HKZ reduction:

- Three main types of SVP solvers: [Kannan'83,Fincke-Pohst'83], [Ajtai-Kumar-Sivakumar'01] and [Micciancio-Voulgaris'10].
- All of (at least) exponential complexities.

▶ Can we do better? With polynomial approximation factors? With heuristics? With quantum computing?

Beating Schnorr's hierarchy:

- BKZ achieves $\gamma \approx k^{n/(2k)}$ in time $\approx 2^{\mathcal{O}(k)} \cdot \mathrm{Poly}(n)$.

▶ A different hierarchy, relating an SVP solver rather than strengthening LLL?

Faster strong reductions

Sub-exponential HKZ reduction:

- Three main types of SVP solvers: [Kannan'83,Fincke-Pohst'83], [Ajtai-Kumar-Sivakumar'01] and [Micciancio-Voulgaris'10].
- All of (at least) exponential complexities.
- Can we do better? With polynomial approximation factors? With heuristics? With quantum computing?

Beating Schnorr's hierarchy:

- BKZ achieves $\gamma \approx k^{n/(2k)}$ in time $\approx 2^{\mathcal{O}(k)} \cdot \mathrm{Poly}(n)$.
- A different hierarchy, relaxing an SVP solver rather than strengthening LLL?

Faster strong reductions

Sub-exponential HKZ reduction:

- Three main types of SVP solvers: [Kannan'83,Fincke-Pohst'83], [Ajtai-Kumar-Sivakumar'01] and [Micciancio-Voulgaris'10].
- All of (at least) exponential complexities.
- Can we do better? With polynomial approximation factors? With heuristics? With quantum computing?

Beating Schnorr's hierarchy:

- BKZ achieves $\gamma \approx k^{n/(2k)}$ in time $\approx 2^{\mathcal{O}(k)} \cdot \mathrm{Poly}(n)$.
- A different hierarchy, relaxing an SVP solver rather than strengthening LLL?

Faster strong reductions

Sub-exponential HKZ reduction:

- Three main types of SVP solvers: [Kannan'83,Fincke-Pohst'83], [Ajtai-Kumar-Sivakumar'01] and [Micciancio-Voulgaris'10].
- All of (at least) exponential complexities.
- Can we do better? With polynomial approximation factors? With heuristics? With quantum computing?

Beating Schnorr's hierarchy:

- BKZ achieves $\gamma \approx k^{n/(2k)}$ in time $\approx 2^{\mathcal{O}(k)} \cdot \mathrm{Poly}(n)$.
- A different hierarchy, relaxing an SVP solver rather than strengthening LLL?

The rise of lattice-based cryptography?

Towards practical lattice-based cryptography:

- Making crucial primitives extremely fast.
- Realizing more functionalities.

Firmer security grounding:

- Mount large-scale cryptanalyses to get meaningful security parameters.
- Are lattice problems hard even for ideal lattices?
- Are lattice problems quantumly hard?

The rise of lattice-based cryptography?

Towards practical lattice-based cryptography:

- Making crucial primitives extremely fast.
- Realizing more functionalities.

Firmer security grounding:

- Mount large-scale cryptanalyses to get meaningful security parameters.
- Are lattice problems hard even for ideal lattices?
- Are lattice problems quantumly hard?

Thank You!