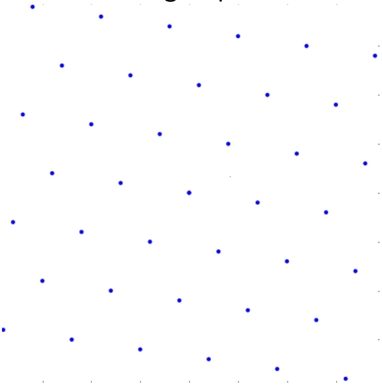
Faster Gaussian Lattice Sampling using Lazy Floating-Point Arithmetic

Léo Ducas, École Normale Supérieure
Phong Nguyen, INRIA & Tsinghua Univ.

Asiacrypt 2012

Lattices

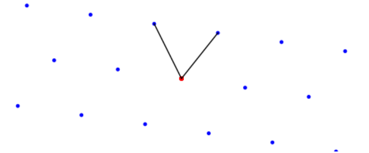
A lattice Λ is a discrete subgroup of $\mathbb{R}^n$.



Basis of Lattices

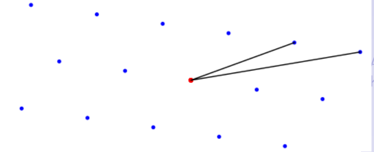
Lattices have two kinds of basis:

Good Basis (short)



Derive bad basis
Solve geometric problem
as **Approx-CVP**

Bad Basis (large)



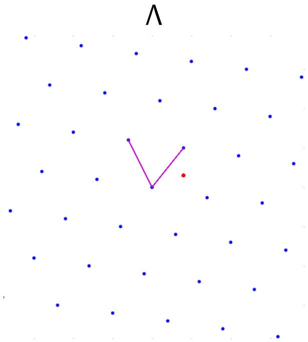
test membership $t \in \Lambda$
generate random element in Λ

Good setting for Public Key Cryptography !

Approximate the Closest Vector Problem

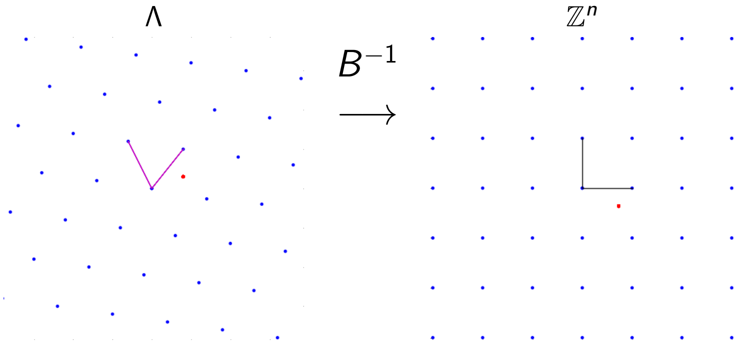
The **Approx-CVP** Problem:

Given $\mathbf{t} \in \mathbb{R}^n$, **find** $\mathbf{c} \in \Lambda$ close to $\mathbf{t}$



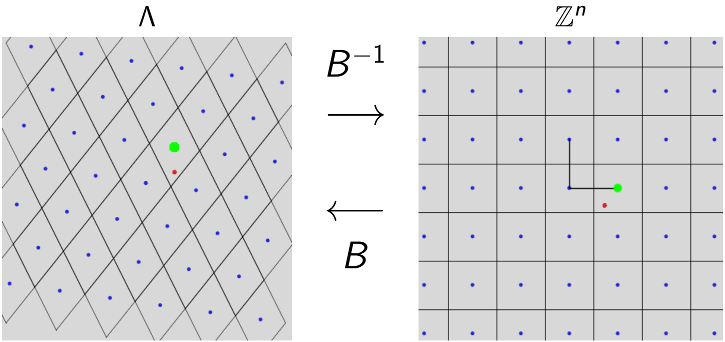
Approximate the Closest Vector Problem

Problem: **Given** $\mathbf{t} \in \mathbb{R}^n$, **find** $\mathbf{c} \in \Lambda$ close to $\mathbf{t}$



Approximate the Closest Vector Problem

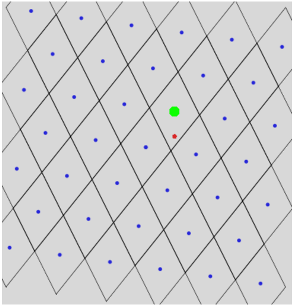
Solution: $\mathbf{s} = \lceil \mathbf{t} \cdot B^{-1} \rfloor \cdot B$ (Babaï's Round-Off [Bab86])



Approximate the Closest Vector Problem

Solution: $\mathbf{s} = \lceil \mathbf{t} \cdot B^{-1} \rfloor \cdot B$ (Babaï's Round-Off [Bab86])

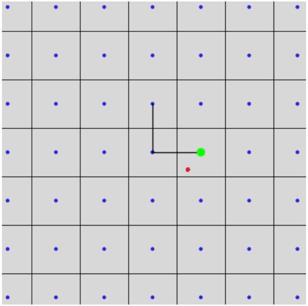
Λ



$\xrightarrow{B^{-1}}$

$\xleftarrow[B]{}$

$\mathbb{Z}^n$

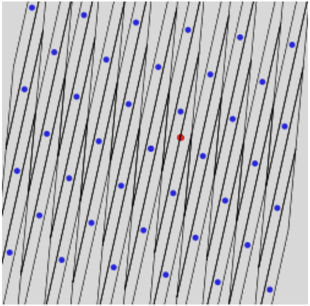


Approximate the Closest Vector Problem

Solution: $\mathbf{s} = \lceil \mathbf{t} \cdot B^{-1} \rfloor \cdot B$ (Babaï's Round-Off [Bab86])

Quality of the solution depends on the basis B.

Λ



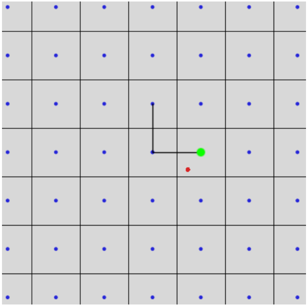
B^{-1}

$\longrightarrow$

$\longleftarrow$

B

$\mathbb{Z}^n$



GGH and NTRUSign Signature Schemes

The Goldreich-Goldwasser-Halevi [GGH97] signature scheme:

- **Secret Key:** a short basis B of Λ

NTRUSign [HGP+03] is an optimized instantiation of GGH, using compact lattices.

GGH and NTRUSIGN Signature Schemes

The Goldreich-Goldwasser-Halevi [GGH97] signature scheme:

- **Secret Key:** a short basis B of Λ
- **Public Key:** a large basis of Λ

NTRUSIGN [HGP+03] is an optimized instantiation of GGH, using compact lattices.

GGH and NTRUSIGN Signature Schemes

The Goldreich-Goldwasser-Halevi [GGH97] signature scheme:

- **Secret Key:** a short basis B of Λ
- **Public Key:** a large basis of Λ
- **Signature:** $\mathbf{t} = H(m) \in \mathbb{R}^n$ the hash of a message
  $\mathbf{s} = \lceil \mathbf{t} \cdot B^{-1} \rfloor \cdot B$ the signature of m

NTRUSIGN [HGP+03] is an optimized instantiation of GGH, using compact lattices.

GGH and NTRUSign Signature Schemes

The Goldreich-Goldwasser-Halevi [GGH97] signature scheme:

- **Secret Key:** a short basis B of Λ
- **Public Key:** a large basis of Λ
- **Signature:** $\mathbf{t} = H(m) \in \mathbb{R}^n$ the hash of a message
  $\mathbf{s} = \lceil \mathbf{t} \cdot B^{-1} \rfloor \cdot B$ the signature of m
- **Verification:** Check that $\mathbf{s} \in \Lambda$ and $\mathbf{s} - H(m)$ is small

NTRUSign [HGP+03] is an optimized instantiation of GGH, using compact lattices.

Gaussian Sampling: Why ?

The previous algorithm to find pre-image leaks information about the good basis B:

- Raw version broken in [NR09]

Gaussian Sampling: Why ?

The previous algorithm to find pre-image leaks information about the good basis B:

- Raw version broken in [NR09]
- Heuristic countermeasures later broken [DN12]

Gaussian Sampling: Why ?

The previous algorithm to find pre-image leaks information about the good basis B:

- Raw version broken in [NR09]
- Heuristic countermeasures later broken [DN12]
- Gaussian Sampling [Kle00] proposed by Gentry *et al.* [GPV08] as a provably secure countermeasure

How to Provably prevents information leakage ?

Let H be a hash function modelized as a **Random Oracle**.
The proof rely on statistical indistinguishability between:

Real-World	**Simulation**
Get $\mathbf{t} = H(m) \in \mathbb{R}^n$	Choose $\mathbf{s} \in \Lambda$ uniformly
Find $\mathbf{s} \in \Lambda$ close to $\mathbf{t}$	Choose $\mathbf{t} = \mathbf{s} + \mathbf{r}$ for short $\mathbf{r} \in \mathbb{R}^n$
using the good basis B	Program the **R.O.** : $H(m) \leftarrow \mathbf{t}$
Output $(\mathbf{t}, \mathbf{s})$	Output $(\mathbf{t}, \mathbf{s})$

Provably prevent information leakage

In the Simulation we set $\mathbf{t} = \mathbf{s} + \mathbf{r}$ for a certain distribution $\mathbf{r} \leftarrow \mathcal{D}$.
In the Real-World we set $\mathbf{t} = H(m) \in \mathbb{R}^n$ that is uniform.

$\Rightarrow$ Two constraints:

Provably prevent information leakage

In the Simulation we set $\mathbf{t} = \mathbf{s} + \mathbf{r}$ for a certain distribution $\mathbf{r} \leftarrow \mathcal{D}$.
In the Real-World we set $\mathbf{t} = H(m) \in \mathbb{R}^n$ that is uniform.

$\Rightarrow$ Two constraints:

- **Smoothness:** $\mathbf{s} + \mathbf{r}$ for $\mathbf{r} \leftarrow \mathcal{D}$ must be (almost) uniform

Provably prevent information leakage

In the Simulation we set $\mathbf{t} = \mathbf{s} + \mathbf{r}$ for a certain distribution $\mathbf{r} \leftarrow \mathcal{D}$.
In the Real-World we set $\mathbf{t} = H(m) \in \mathbb{R}^n$ that is uniform.

$\Rightarrow$ Two constraints:

- **Smoothness:** $\mathbf{s} + \mathbf{r}$ for $\mathbf{r} \leftarrow \mathcal{D}$ must be (almost) uniform
- **Pre-image Sampling Correctness:** In the Real-World, knowing a short basis B, and given $\mathbf{t}$, the signer should sample $\mathbf{s} \in \Lambda$ such that follows the **conditional distribution** $\{\mathbf{s} \leftarrow \mathcal{D} + \mathbf{t} | \mathbf{s} \in \Lambda\}$

trapdoor OWF with pre-image sampling

Formalized by Gentry *et al.* [GPV08].

Already used before [GPV08]: Rabin Signature Scheme

Let $N = pq$ be an RSA modulus.

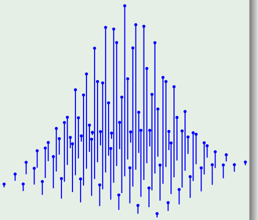
- The function $x \in Z_N \mapsto x^2 \in \mathbb{Z}_N$ is a one-way function,
- The factorization (p, q) can be used as a trapdoor: recover $\sqrt{\cdot}$ by CRT over $\mathbb{Z}_p$ and $\mathbb{Z}_q$
- Yet, each square have 4 pre-image.
- One should choose it uniformly at random to achieve **smoothness**

trapdoor OWF with pre-image sampling

Formalized by Gentry *et al.* [GPV08].

For Lattice-based OWF: Gaussian Sampling

- Best smoothness/width ratio
- **Explicit** and simple formulae for the Conditional Distribution
- **Known algorithm** to sample the conditional distribution using a short basis from Klein [Kle00]



Somehow, GPV is similar to Rabin Signature, with a non-trivial pre-image sampling algorithm.

The issue: efficiency

Lattice Based Cryptography is usually praised for:

- Resistance to sub-exponential and quantum attacks
- Efficiently Parrallelizable Efficiently Parrallelizable
- Operation in a small modulus $\mathbb{Z}_q$

The issue: efficiency

Lattice Based Cryptography is usually praised for:

- Resistance to sub-exponential and quantum attacks
- ~~Efficiently Parrallelizable~~
- Operations in $\cancel{\text{a small modulus } \mathbb{Z}_q}$ $\mathbb{Q}$ **with large operands**

Some algorithms in fact require real numbers ($\mathbb{Q}$ or $\mathbb{R}$), including Klein's Algorithm!
Parallelizability repaired by Peikert [Pei10].

What about **Floating Point Arithmetic** (FPA) to formalize, and maybe accelerate operations in $\mathbb{Q}$?

Our Results

In this work, we analyze and optimize the use of **FPA** in Klein's Alg. as well as the offline part of Peikert's Alg.

- First rigorous analysis of FPA for **provable security**

Our Results

In this work, we analyze and optimize the use of **FPA** in Klein's Alg. as well as the offline part of Peikert's Alg.

- First rigorous analysis of FPA for **provable security**
- **Concrete** requirement for the FPA variant of those Alg.

Our Results

In this work, we analyze and optimize the use of **FPA** in Klein's Alg. as well as the offline part of Peikert's Alg.

- First rigorous analysis of FPA for **provable security**
- **Concrete** requirement for the FPA variant of those Alg.
- Laziness/backtracking technique to improve the running time from $\tilde{\mathcal{O}}(n^3)$ to $\tilde{\mathcal{O}}(n^2)$ or even $\tilde{\mathcal{O}}(n)$ in some cases

Our Results

In this work, we analyze and optimize the use of **FPA** in Klein's Alg. as well as the offline part of Peikert's Alg.

- First rigorous analysis of FPA for **provable security**
- **Concrete** requirement for the FPA variant of those Alg.
- Laziness/backtracking technique to improve the running time from $\tilde{\mathcal{O}}(n^3)$ to $\tilde{\mathcal{O}}(n^2)$ or even $\tilde{\mathcal{O}}(n)$ in some cases
- Allow implementation using mostly **double-float**, thus benifiting from **hardware** acceleration

Floating Point Arithmetic Definition

Definition (Floating Point of Mantissa m)

A floating-point number $\bar{f} \in \mathbb{FP}_m$ is a triplet $\bar{f} = (s, e, v)$ where $s \in \{0, 1\}$, $e \in \mathbb{Z}$ and $v \in \{0 \dots 2^m - 1\}$. It represents the real number $\mathbb{R}(\bar{f}) = (-1)^s \cdot 2^{e-m} \cdot v \in \mathbb{R}$.

FPA operations verify relative **error bounds**:

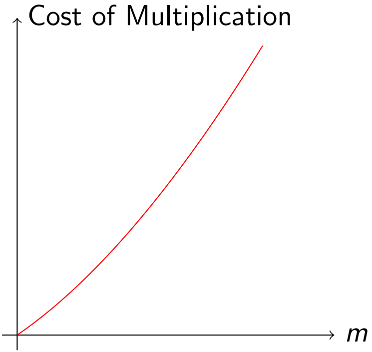
Property (FPA axioms)

Let $\epsilon = 2^{1-m}$. All arithmetic operations $\bar{\circ} \in \{\bar{+}, \bar{-}, \bar{\cdot}, \bar{/}\}$ verify for any $\bar{f}_1, \bar{f}_2 \in \mathbb{FP}_m$:

$$|\mathbb{R}(\bar{f}_1 \bar{\circ} \bar{f}_2) - (\mathbb{R}(\bar{f}_1) \circ \mathbb{R}(\bar{f}_2))| \leq |\mathbb{R}(\bar{f}_1) \circ \mathbb{R}(\bar{f}_2)|\epsilon$$

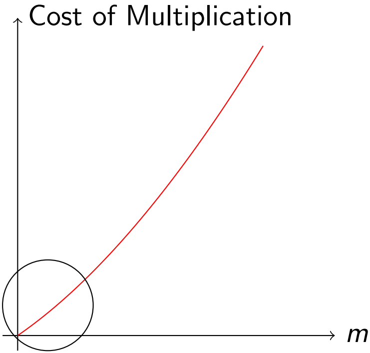
FPA Efficiency, Theory

In **theory**, the cost of multiplication is $O(m \log m \log\log m)$, using **Schönhage-Strassen** Algorithm (aka. FFT).



FPA Efficiency, Theory

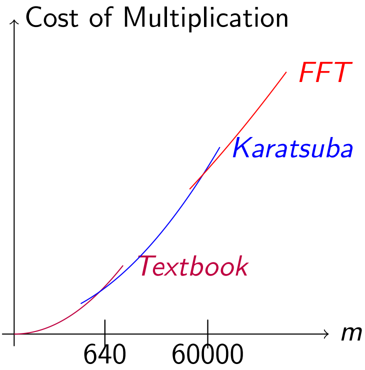
In **theory**, the cost of multiplication is $O(m \log m \log\log m)$, using **Schönhage-Strassen** Algorithm (aka. FFT).



FPA Efficiency, Practice

Yet, considering the constants and overhead , one rather use:

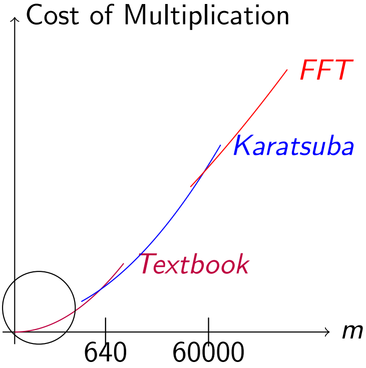
- Textbook mult. when $m \leq 640$: $\tilde{\mathcal{O}}(m^2))$
- Karatsuba mult when $m \leq 60000$: $\tilde{\mathcal{O}}(m^{1.585})$
- FFT otherwise



FPA Efficiency, Practice

Yet, considering the constants and overhead , one rather use:

- Textbook mult. when $m \leq 640$: $\tilde{\mathcal{O}}(m^2))$
- Karatsuba mult when $m \leq 60000$: $\tilde{\mathcal{O}}(m^{1.585})$
- FFT otherwise

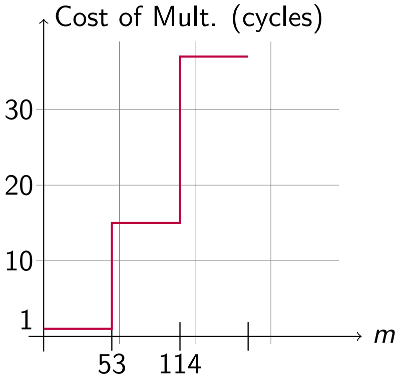


FPA Efficiency, Practice for small m

Below machine precision, operations are implemented in hardware: **they can be done in 1 cycle !**
Beyond, there is an important overhead because of software implementation.

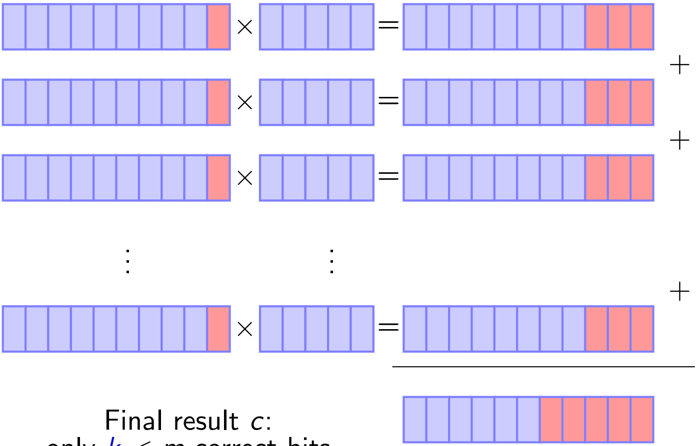
On x86-64 proc.

The speed ratio between *double* and *quad-float* **is about 1 to 15!**



Error Propagation during Klein's Alg.

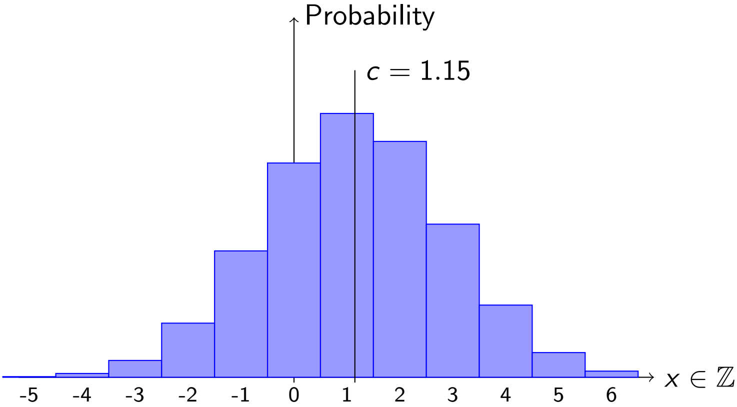
Val in $\mathbb{FP}_m$ Val in $\mathbb{Z}$



Final result c:
only $k < m$ correct bits

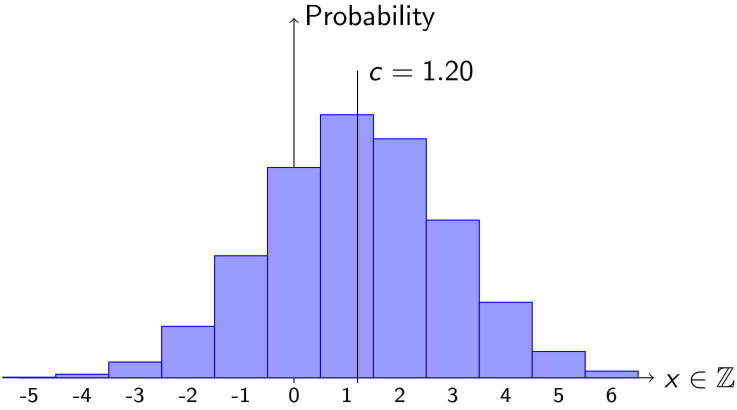
1-dimensional Discrete Gaussian

The previous result c is then used as the center of a discrete Gaussian:



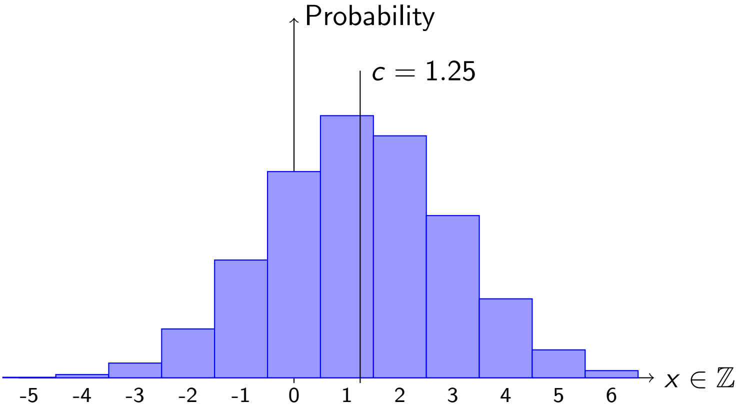
1-dimensional Discrete Gaussian

The previous result c is then used as the center of a discrete Gaussian:



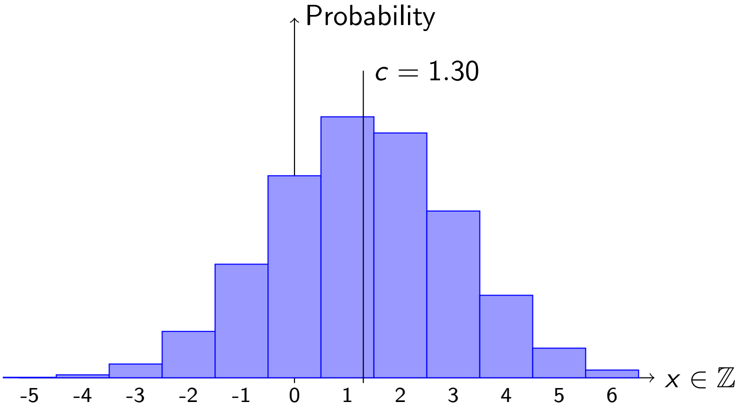
1-dimensional Discrete Gaussian

The previous result c is then used as the center of a discrete Gaussian:



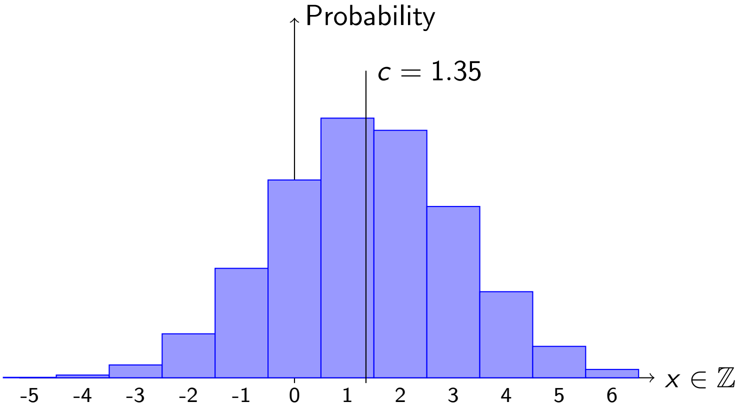
1-dimensional Discrete Gaussian

The previous result c is then used as the center of a discrete Gaussian:



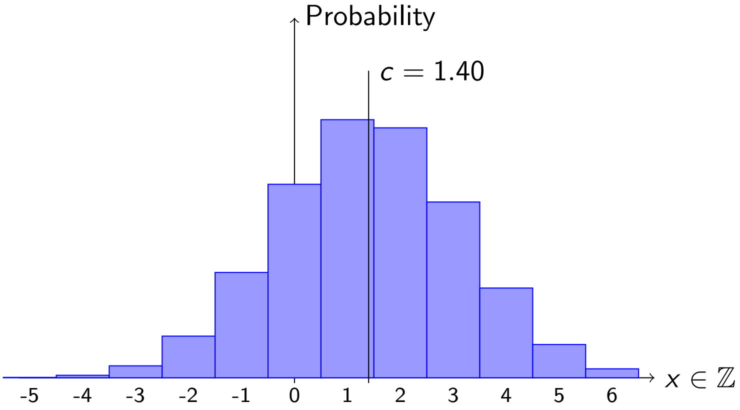
1-dimensional Discrete Gaussian

The previous result c is then used as the center of a discrete Gaussian:



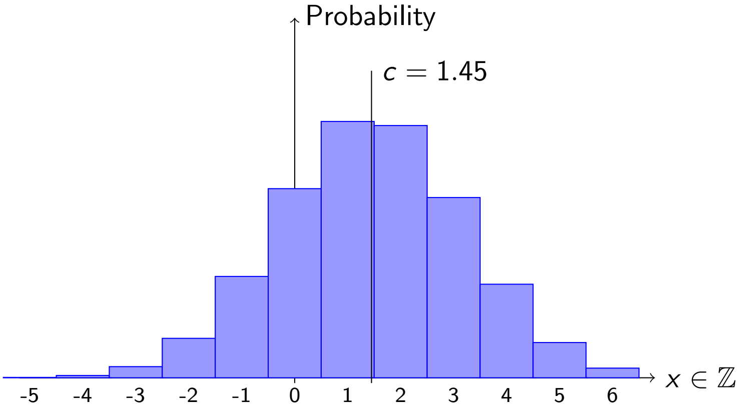
1-dimensional Discrete Gaussian

The previous result c is then used as the center of a discrete Gaussian:



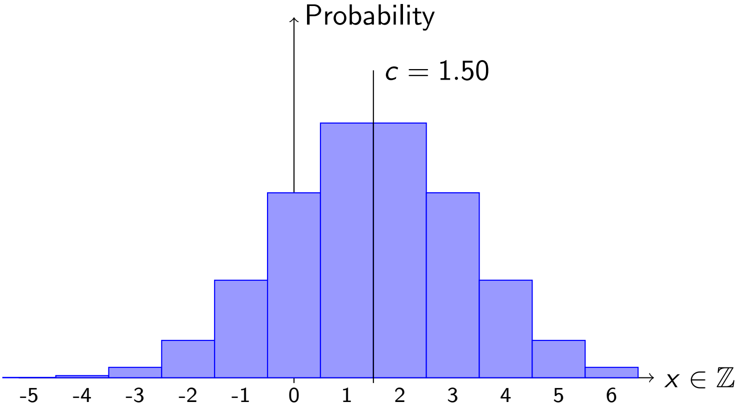
1-dimensional Discrete Gaussian

The previous result c is then used as the center of a discrete Gaussian:



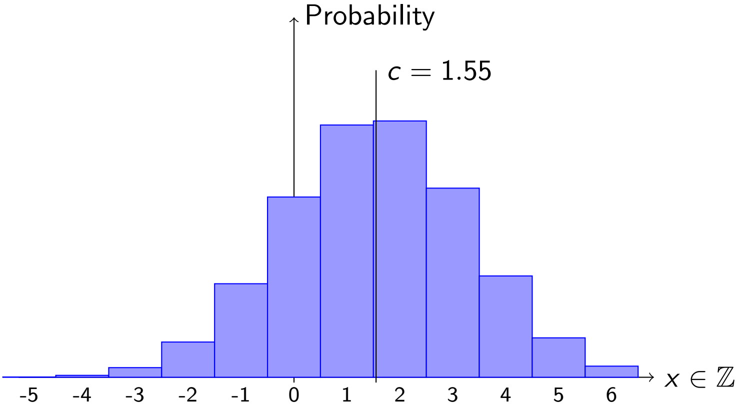
1-dimensional Discrete Gaussian

The previous result c is then used as the center of a discrete Gaussian:



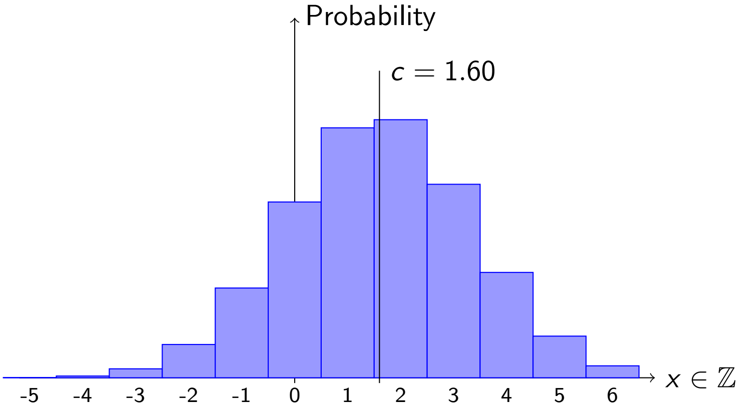
1-dimensional Discrete Gaussian

The previous result c is then used as the center of a discrete Gaussian:



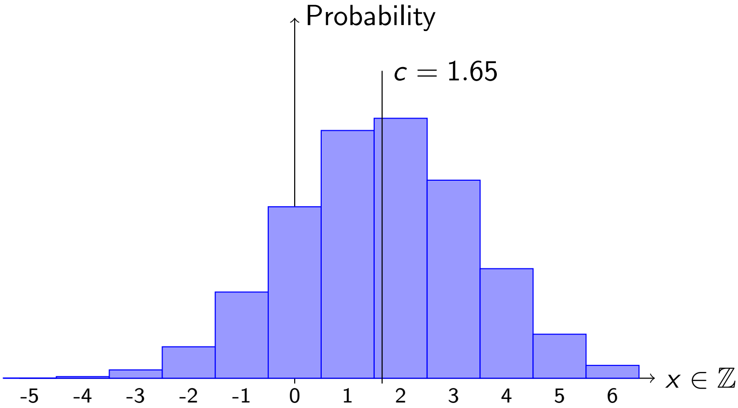
1-dimensional Discrete Gaussian

The previous result c is then used as the center of a discrete Gaussian:



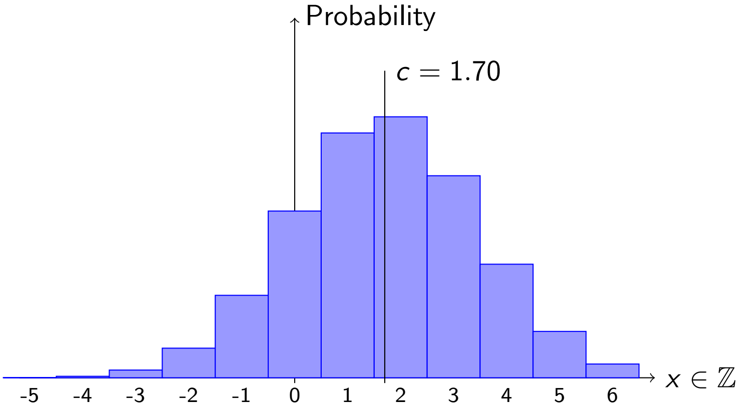
1-dimensional Discrete Gaussian

The previous result c is then used as the center of a discrete Gaussian:



1-dimensional Discrete Gaussian

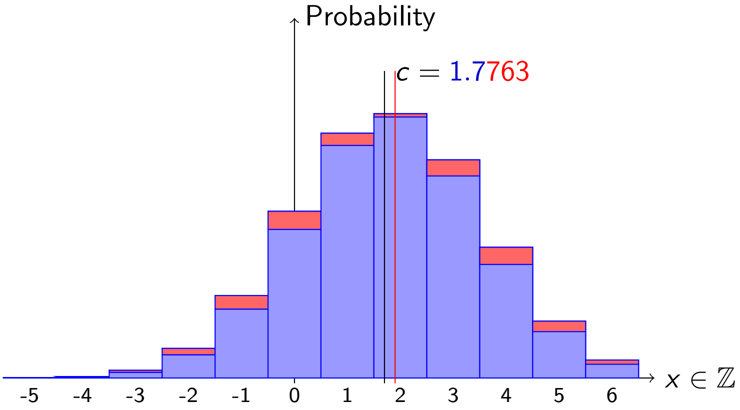
The previous result c is then used as the center of a discrete Gaussian:



1-dimensional Discrete Gaussian

Uncertainty propagation

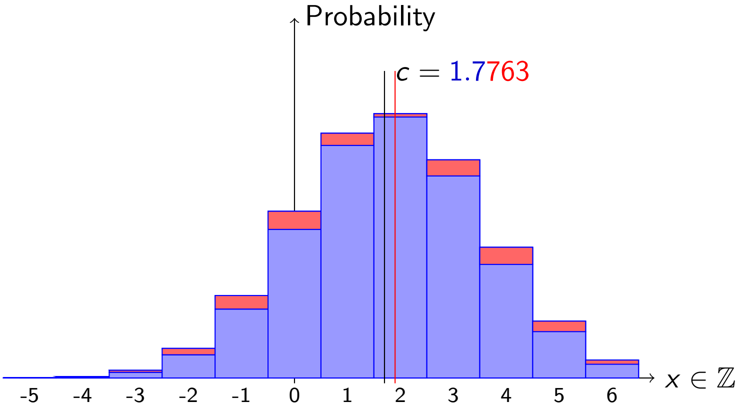
The output distribution can only be correct up to the correctness of the input center c.



1-dimensional Discrete Gaussian

Correction Requirement

We need the **statistical distance** between the **desired distribution** and the output distribution to be **negligible**.



Precision Requirement

Therefore, to prove security of λ bits, we need to compute c such that its λ first bits are surely correct: $m \geq \lambda = 80$.

Theorem (Sufficient Correctness Condition)

For any λ, the **statistical distance** *between $\mathcal{D}_{\Lambda(B),\sigma,c}$ and the output of* $\mathbf{Klein}_{\mathbb{FP}_m}(B,\sigma,c)$ *is less than $2^{-\lambda}$ if:*

$$m \geq \lambda + \mathrm{polylog}(\lambda)$$

Concrete Case

For security of $\lambda = 80$, with NTRUSign-type lattice, we require $m \approx 120$.

Efficiency of the previous Algorithm

The previous result let us run Klein's Alg. at precision $m = \lambda + \text{polylog}(\lambda)$.

- Asymptotic running time is still $\tilde{\mathcal{O}}(\lambda^3)$: only better than $\textbf{Klein}_{\mathbb{Q}}$ by a constant

Efficiency of the previous Algorithm

The previous result let us run Klein's Alg. at precision $m = \lambda + \text{polylog}(\lambda)$.

- Asymptotic running time is still $\tilde{\mathcal{O}}(\lambda^3)$: only better than $\mathbf{Klein}_{\mathbb{Q}}$ by a constant
- **double-float** are not suitable, and **quad-float** are barely enough

Efficiency of the previous Algorithm

The previous result let us run Klein's Alg. at precision $m = \lambda + \text{polylog}(\lambda)$.

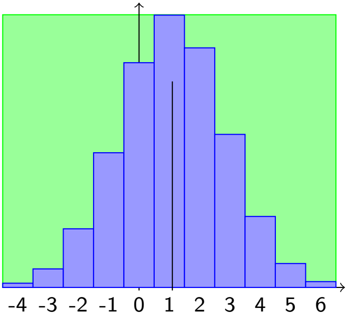
- Asymptotic running time is still $\tilde{\mathcal{O}}(\lambda^3)$: only better than $\mathbf{Klein}_{\mathbb{Q}}$ by a constant
- **double-float** are not suitable, and **quad-float** are barely enough
- We really do need that much precision for information theoretic reasons, but do we need it **every single time** ?

Rejection Sampling

The 1-dimensional discrete Gaussian is drawn using **Rejection Sampling**.

Rejection Sampling:

- Draw uniform $(x, y) \in$ ■
- If $(x, y) \in$ ■ return x
- Else, restart

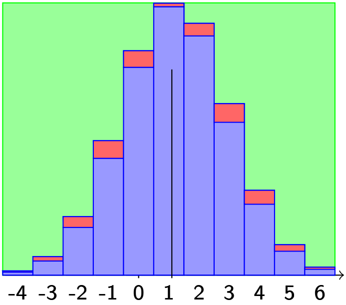


Dealing with Uncertainty

First define a Rej. Sampling Algorithm **with Trigger**: given an error-bound $\delta_{\mathbf{c}}$ on c, bound the uncertainty area ■.

Rejection Sampling:

- Draw uniform $(x, y) \in$ ■
- If $(x, y) \in$ ■ Trigger
- If $(x, y) \in$ ■ return x
- Else, restart



Our Optimized Algorithm: Lazyness/Backtracking

Use two FP types: **high prec.** m and **low prec** $m' < m$.

High prec. $\Rightarrow$ negligible ■ area (negligible error)

Low prec. $\Rightarrow$ small ■ area (rare backtracking)

- Start Rej.-Sampling with Trigger, using low precision c

Our Optimized Algorithm: Lazyness/Backtracking

Use two FP types: **high prec.** m and **low prec** $m' < m$.

High prec. $\Rightarrow$ negligible $\blacksquare$ area (negligible error)

Low prec. $\Rightarrow$ small $\blacksquare$ area (rare backtracking)

- Start Rej.-Sampling with Trigger, using low precision c
- With small probability, will **trigger backtracking**

Our Optimized Algorithm: Lazyness/Backtracking

Use two FP types: **high prec.** m and **low prec** $m' < m$.

High prec. $\Rightarrow$ negligible ■ area (negligible error)

Low prec. $\Rightarrow$ small ■ area (rare backtracking)

- Start Rej.-Sampling with Trigger, using low precision c
- With small probability, will **trigger backtracking**

  Recompute the same c at high precision

Our Optimized Algorithm: Lazyness/Backtracking

Use two FP types: **high prec.** m and **low prec** $m' < m$.

High prec. $\Rightarrow$ negligible ■ area (negligible error)

Low prec. $\Rightarrow$ small ■ area (rare backtracking)

- Start Rej.-Sampling with Trigger, using low precision c
- With small probability, will **trigger backtracking**

  Recompute the same c at high precision

  Return to Rej.-Sampling, with negligible ■ area.

Efficiency of this Our Optimized Algorithm

- Choosing m' carefully, we can show that this new algorithm runs in $\tilde{\mathcal{O}}(\lambda^2)$.

Efficiency of this Our Optimized Algorithm

- Choosing m' carefully, we can show that this new algorithm runs in $\tilde{\mathcal{O}}(\lambda^2)$.
- Same technique (+ other tricks) applies to Peikert's Offline Algorithm, for which we can reach **quasi-linear complexity**.

Efficiency of this Our Optimized Algorithm

- Choosing m' carefully, we can show that this new algorithm runs in $\tilde{\mathcal{O}}(\lambda^2)$.
- Same technique ($+$ other tricks) applies to Peikert's Offline Algorithm, for which we can reach **quasi-linear complexity**.
- Choosing $m' = 53$ (*double-precision*) for known crypto-grade lattice is enough: most operations are done in **1 cycle !**

Summary of this work

Provide another step toward **practicality** of Lattice-Based Cryptography.

Summary of this work

Provide another step toward **practicality** of Lattice-Based Cryptography.

- First (?) application of numerical analysis to provable security

Summary of this work

Provide another step toward **practicality** of Lattice-Based Cryptography.

- First (?) application of numerical analysis to provable security
- Give **concrete** conditions rather than asymptotic: implementation-ready

Summary of this work

Provide another step toward **practicality** of Lattice-Based Cryptography.

- First (?) application of numerical analysis to provable security
- Give **concrete** conditions rather than asymptotic: implementation-ready
- Integrate and analyze Lazyness technique: efficiency improved **in practice by a factor about 15**

Thank you !

Questions ?

L. Babai, On Lovász lattice reduction and the nearest lattice point problem, Combinatorica **6** (1986), 1–13.

L. Ducas and P. Q. Nguyen, Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures, Advances in Cryptology – Proceedings of ASIACRYPT '12, LNCS, Springer, 2012.

O. Goldreich, S. Goldwasser, and S. Halevi, Public-key cryptosystems from lattice reduction problems, Proc. of Crypto '97, LNCS, vol. 1294, IACR, Springer-Verlag, 1997, Full version vailable at ECCC as TR96-056., pp. 112–131.

Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan, Trapdoors for hard lattices and new cryptographic constructions, Proc. STOC '08, ACM, 2008, pp. 197–206.

J. Hoffstein, N. A. Howgrave Graham, J. Pipher, J. H. Silverman, and W. Whyte, NTRUSIGN: Digital signatures using the NTRU lattice, Proc. of CT-RSA, LNCS, vol. 2612, Springer-Verlag, 2003.

P. Klein, Finding the closest lattice vector when it's unusually close, Proc. of SODA '00, ACM–SIAM, 2000.

P. Q. Nguyen and O. Regev, Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures, J. Cryptology **22** (2009), no. 2, 139–160, Preliminary version in EUROCRYPT 2006.

Chris Peikert, An efficient and parallel gaussian sampler for lattices, Proc. CRYPTO '10, Lecture Notes in Computer Science, vol. 6223, Springer, 2010, pp. 80–97.