Part 2 – LWE-based cryptography

Douglas Stebila McMaster University

SAC Summer School • Université d'Ottawa • August 14, 2017
https://www.douglas.stebila.ca/research/presentations

Funding acknowledgements:
NSERC CRSNG

Post-quantum crypto

Classical crypto with no known exponential quantum speedup

Hash-based
- Merkle signatures
- Sphincs

Code-based
- McEliece
- Niederreiter

Multivariate
- multivariate quadratic

Lattice-based
- NTRU
- learning with errors
- ring-LWE

Isogenies
- supersingular elliptic curve isogenies

Quantum-safe crypto

Classical post-quantum crypto

Hash-based

- Merkle signatures
- Sphincs

Code-based

- McEliece
- Niederreiter

Multivariate

- multivariate quadratic

Lattice-based

- NTRU
- learning with errors
- ring-LWE

Isogenies

- supersingular elliptic curve isogenies

Quantum crypto

Quantum key distribution

Quantum random number generators

Quantum channels

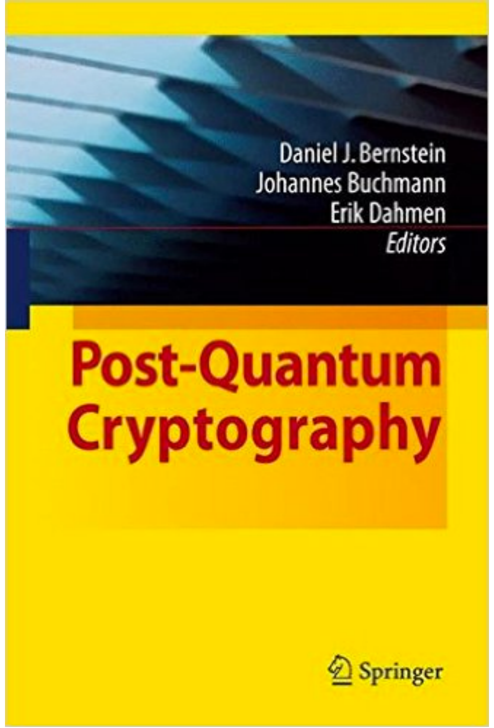
Quantum blind computation

Today's agenda

1. Quantum computing and its impact on cryptography (Mosca)
2. LWE-based cryptography (Stebila)
3. Isogeny-based cryptography (Jao)
4. Additional topics
   - Security models for post-quantum cryptography (Jao)
   - Applications (Stebila)

Topics excluded:

- Code-based cryptography
- Hash-based signatures
- Multivariate cryptography



Learning with errors problems

Solving systems of linear equations

$$\underbrace{\begin{bmatrix} 4 & 1 & 11 & 10 \\ 5 & 5 & 9 & 5 \\ 3 & 9 & 0 & 10 \\ 1 & 3 & 3 & 2 \\ 12 & 7 & 3 & 4 \\ 6 & 5 & 11 & 4 \\ 3 & 3 & 5 & 0 \end{bmatrix}}_{\mathbb{Z}_{13}^{7\times 4}} \times \underbrace{\begin{bmatrix} \\ \\ \\ \end{bmatrix}}_{\text{secret } \mathbb{Z}_{13}^{4\times 1}} = \underbrace{\begin{bmatrix} 4 \\ 8 \\ 1 \\ 10 \\ 4 \\ 12 \\ 9 \end{bmatrix}}_{\mathbb{Z}_{13}^{7\times 1}}$$

Linear system problem: given blue, find red

Solving systems of linear equations

$$\underbrace{\begin{bmatrix} 4 & 1 & 11 & 10 \\ 5 & 5 & 9 & 5 \\ 3 & 9 & 0 & 10 \\ 1 & 3 & 3 & 2 \\ 12 & 7 & 3 & 4 \\ 6 & 5 & 11 & 4 \\ 3 & 3 & 5 & 0 \end{bmatrix}}_{\mathbb{Z}_{13}^{7\times 4}} \times \underbrace{\begin{bmatrix} 6 \\ 9 \\ 11 \\ 11 \end{bmatrix}}_{\substack{\textbf{secret} \\ \mathbb{Z}_{13}^{4\times 1}}} = \underbrace{\begin{bmatrix} 4 \\ 8 \\ 1 \\ 10 \\ 4 \\ 12 \\ 9 \end{bmatrix}}_{\mathbb{Z}_{13}^{7\times 1}}$$

Easily solved using Gaussian elimination (Linear Algebra 101)

Linear system problem: given blue, find **red**

Learning with errors problem

$$\overset{\substack{\textbf{random}\\ \mathbb{Z}_{13}^{7\times 4}}}{\begin{bmatrix} 4 & 1 & 11 & 10 \\ 5 & 5 & 9 & 5 \\ 3 & 9 & 0 & 10 \\ 1 & 3 & 3 & 2 \\ 12 & 7 & 3 & 4 \\ 6 & 5 & 11 & 4 \\ 3 & 3 & 5 & 0 \end{bmatrix}} \times \overset{\substack{\textbf{secret}\\ \mathbb{Z}_{13}^{4\times 1}}}{\begin{bmatrix} 6 \\ 9 \\ 11 \\ 11 \end{bmatrix}} + \overset{\substack{\textbf{small noise}\\ \mathbb{Z}_{13}^{7\times 1}}}{\begin{bmatrix} 0 \\ -1 \\ 1 \\ 1 \\ 1 \\ 0 \\ -1 \end{bmatrix}} = \overset{\mathbb{Z}_{13}^{7\times 1}}{\begin{bmatrix} 4 \\ 7 \\ 2 \\ 11 \\ 5 \\ 12 \\ 8 \end{bmatrix}}$$

Learning with errors problem

$$
\underset{\mathbb{Z}_{13}^{7\times 4}}{\textbf{random}}\;
\begin{bmatrix} 4 & 1 & 11 & 10 \\ 5 & 5 & 9 & 5 \\ 3 & 9 & 0 & 10 \\ 1 & 3 & 3 & 2 \\ 12 & 7 & 3 & 4 \\ 6 & 5 & 11 & 4 \\ 3 & 3 & 5 & 0 \end{bmatrix}
\times
\underset{\mathbb{Z}_{13}^{4\times 1}}{\textbf{secret}}\;
\begin{bmatrix} \ \\ \ \\ \ \\ \ \end{bmatrix}
+
\underset{\mathbb{Z}_{13}^{7\times 1}}{\textbf{small noise}}\;
\begin{bmatrix} \ \\ \ \\ \ \\ \ \\ \ \\ \ \\ \ \end{bmatrix}
=
\underset{\mathbb{Z}_{13}^{7\times 1}}{}\;
\begin{bmatrix} 4 \\ 7 \\ 2 \\ 11 \\ 5 \\ 12 \\ 8 \end{bmatrix}
$$

Search LWE problem: given blue, find red

Search LWE problem

Let n, m, and q be positive integers. Let χ_s and χ_e be distributions over $\mathbb{Z}$. Let $\mathbf{s} \xleftarrow{\$} \chi_s^n$. Let $\mathbf{a}_i \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^n)$, $e_i \xleftarrow{\$} \chi_e$, and set $b_i \leftarrow \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \mod q$, for $i = 1, \ldots, m$.

The *search LWE problem* for $(n, m, q, \chi_s, \chi_e)$ is to find s given $(\mathbf{a}_i, b_i)_{i=1}^m$.

In particular, for algorithm $\mathcal{A}$, define the advantage

$$\mathsf{Adv}^{\mathsf{lwe}}_{n,m,q,\chi_s,\chi_e}(\mathcal{A}) = \Pr\Big[\mathbf{s} \xleftarrow{\$} \chi_s^n; \mathbf{a}_i \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^n); e_i \xleftarrow{\$} \chi_e; \\ b_i \leftarrow \langle \mathbf{a}_i, \mathbf{s}_i \rangle + e \bmod q : \mathcal{A}((\mathbf{a}_i, b_i)_{i=1}^m) = \mathbf{s})\Big] \ .$$

Decision learning with errors problem

$$\begin{array}{ccccccc} \textbf{random} & & \textbf{secret} & & \textbf{small noise} & & \textbf{looks random} \\ \mathbb{Z}_{13}^{7\times 4} & & \mathbb{Z}_{13}^{4\times 1} & & \mathbb{Z}_{13}^{7\times 1} & & \mathbb{Z}_{13}^{7\times 1} \\ \begin{bmatrix} 4 & 1 & 11 & 10 \\ 5 & 5 & 9 & 5 \\ 3 & 9 & 0 & 10 \\ 1 & 3 & 3 & 2 \\ 12 & 7 & 3 & 4 \\ 6 & 5 & 11 & 4 \\ 3 & 3 & 5 & 0 \end{bmatrix} & \times & \begin{bmatrix} \ \\ \ \\ \ \\ \ \end{bmatrix} & + & \begin{bmatrix} \ \\ \ \\ \ \\ \ \\ \ \\ \ \\ \ \end{bmatrix} & = & \begin{bmatrix} 4 \\ 7 \\ 2 \\ 11 \\ 5 \\ 12 \\ 8 \end{bmatrix} \end{array}$$

Decision LWE problem: given **blue**, distinguish **green** from random

Decision LWE problem

Let n and q be positive integers. Let χ_s and χ_e be distributions over $\mathbb{Z}$. Let $\mathbf{s} \xleftarrow{\$} \chi_s^n$. Define the following two oracles:

- $O_{\chi_e,\mathbf{s}}$: $\mathbf{a} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^n)$, $e \xleftarrow{\$} \chi_e$; return $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e \bmod q)$.
- U: $\mathbf{a} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^n)$, $u \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q)$; return $(\mathbf{a}, u)$.

The *decision LWE problem* for (n, q, χ_s, χ_e) is to distinguish $O_{\chi,\mathbf{s}}$ from U.

In particular, for algorithm $\mathcal{A}$, define the advantage

$$\mathsf{Adv}^{\mathsf{dlwe}}_{n,q,\chi_s,\chi_e}(\mathcal{A}) = \left| \Pr(\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n : \mathcal{A}^{O_{\chi_e,\mathbf{s}}}() = 1) - \Pr(\mathcal{A}^U() = 1) \right| .$$

Choice of error distribution

- Usually a discrete Gaussian distribution of width $s = \alpha q$ for error rate $\alpha < 1$
- Define the Gaussian function

$$\rho_s(\mathbf{x}) = \exp(-\pi \|\mathbf{x}\|^2 / s^2)$$

- The continuous Gaussian distribution has probability density function

$$f(\mathbf{x}) = \rho_s(\mathbf{x}) / \int_{\mathbb{R}^n} \rho_s(\mathbf{z}) d\mathbf{z} = \rho_s(\mathbf{x}) / s^n$$

Short secrets

- The secret distribution χ_s was originally taken to be the uniform distribution

- Short secrets: use $\chi_s = \chi_e$
- There's a tight reduction showing that LWE with short secrets is hard if LWE with uniform secrets is hard

Toy example versus real-world example

$\mathbb{Z}_{13}^{7\times 4}$

4	1	11	10
5	5	9	5
3	9	0	10
1	3	3	2
12	7	3	4
6	5	11	4
3	3	5	0

$\mathbb{Z}_{2^{15}}^{752\times 8}$

8

752

2738	3842	3345	2979	…
2896	595	3607		
377	1575			
2760				
…				

752 × 8 × 15 bits = **11 KiB**

Ring learning with errors problem

random
$\mathbb{Z}_{13}^{7\times 4}$

4	1	11	10
10	4	1	11
11	10	4	1
1	11	10	4
4	1	11	10
10	4	1	11
11	10	4	1

Each row is the cyclic shift of the row above

Ring learning with errors problem

random

$\mathbb{Z}_{13}^{7\times 4}$

4	1	11	10
3	4	1	11
2	3	4	1
12	2	3	4
9	12	2	3
10	9	12	2
11	10	9	12

Each row is the cyclic shift of the row above

…

with a special wrapping rule: *x* wraps to –*x* mod 13.

Ring learning with errors problem

random
$\mathbb{Z}_{13}^{7\times 4}$

4	**1**	**11**	**10**

Each row is the cyclic shift of the row above
…
with a special wrapping rule:
x wraps to –*x* mod 13.

So I only need to tell you the first row.

Ring learning with errors problem

$\mathbb{Z}_{13}[x]/\langle x^4 + 1 \rangle$

	$4 + 1x + 11x^2 + 10x^3$	**random**
$\times$	$6 + 9x + 11x^2 + 11x^3$	**secret**
$+$	$0 - 1x + 1x^2 + 1x^3$	**small noise**
$=$	$10 + 5x + 10x^2 + 7x^3$	

Ring learning with errors problem

$\mathbb{Z}_{13}[x]/\langle x^4 + 1 \rangle$

	$4 + 1x + 11x^2 + 10x^3$	**random**
×		**secret**
+		**small noise**
=	$10 + 5x + 10x^2 + 7x^3$	

Search ring-LWE problem: given blue, find red

Search ring-LWE problem

Let $R = \mathbb{Z}[X]/\langle X^n + 1\rangle$, where n is a power of 2.

Let q be an integer, and define $R_q = R/qR$, i.e., $R_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$.

Let χ_s and χ_e be distributions over R_q. Let $s \xleftarrow{\$} \chi_s$. Let $a \xleftarrow{\$} \mathcal{U}(R_q)$, $e \xleftarrow{\$} \chi_e$, and set $b \leftarrow as + e$.

The *search ring-LWE problem* for (n, q, χ_s, χ_e) is to find s given (a, b).

In particular, for algorithm $\mathcal{A}$ define the advantage

$$\mathsf{Adv}^{\mathsf{rlwe}}_{n,q,\chi_s,\chi_e}(\mathcal{A}) = \Pr\left[s \xleftarrow{\$} \chi_s; a \xleftarrow{\$} \mathcal{U}(R_q); e \xleftarrow{\$} \chi_e; b \leftarrow as + e : \mathcal{A}(a, b) = s\right] .$$

Decision ring-LWE problem

Let n and q be positive integers. Let χ_s and χ_e be distributions over R_q. Let $s \stackrel{\$}{\leftarrow} \chi_s$. Define the following two oracles:

- $O_{\chi_e,s}$: $a \stackrel{\$}{\leftarrow} \mathcal{U}(R_q)$, $e \stackrel{\$}{\leftarrow} \chi_e$; return $(a, as+e)$.
- U: $a, u \stackrel{\$}{\leftarrow} \mathcal{U}(R_q)$; return (a, u).

The *decision ring-LWE problem* for (n, q, χ_s, χ_e) is to distinguish $O_{\chi_e,s}$ from U.

In particular, for algorithm $\mathcal{A}$, define the advantage

$$\mathsf{Adv}^{\mathsf{drlwe}}_{n,q,\chi_s,\chi_e}(\mathcal{A}) = \left| \Pr(s \stackrel{\$}{\leftarrow} R_q : \mathcal{A}^{O_{\chi_e,s}}() = 1) - \Pr(\mathcal{A}^{U}() = 1) \right| .$$

Problems

Computational LWE problem	Decision LWE problem
Computational ring-LWE problem	Decision ring-LWE problem

with or without short secrets

Search-decision equivalence

- **Easy fact**: If the search LWE problem is easy, then the decision LWE problem is easy.
- **Fact**: If the decision LWE problem is easy, then the search LWE problem is easy.
  - Requires nq calls to decision oracle
  - Intuition: test the each value for the first component of the secret, then move on to the next one, and so on.

NTRU problem

For an invertible $s \in R_q^*$ and a distribution χ on R, define $N_{s,\chi}$ to be the distribution that outputs $e/s \in R_q$ where $e \xleftarrow{\$} \chi$.

The **NTRU learning problem** is: given independent samples $a_i \in R_q$ where every sample is distributed according to either: (1) $N_{s,\chi}$ for some randomly chosen $s \in R_q$ (fixed for all samples), or (2) the uniform distribution, distinguish which is the case.

"Lattice-based"

Hardness of decision LWE – "lattice-based"

Lattices

Let $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_n\} \subseteq \mathbb{Z}_q^{n \times n}$ be a set of linearly independent basis vectors for $\mathbb{Z}_q^n$. Define the corresponding **lattice**

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} z_i \mathbf{b}_i : z_i \in \mathbb{Z} \right\} .$$

(In other words, a lattice is a set of *integer* linear combinations.)

Define the **minimum distance** of a lattice as

$$\lambda_1(\mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{v}\| .$$

Shortest vector problem

The **shortest vector problem** (SVP) is: given a basis $\mathbf{B}$ for some lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, find a shortest non-zero vector, i.e., find $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$.

The **decision approximate shortest vector problem** (GapSVP_γ) is: given a basis $\mathbf{B}$ for some lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ where either $\lambda_1(\mathcal{L}) \leq 1$ or $\lambda_1(\mathcal{L}) > \gamma$, determine which is the case.

Regev's iterative reduction

Theorem. **[Reg05]** For any modulus $q \leq 2^{\text{poly}(n)}$ and any discretized Gaussian error distribution χ of parameter $\alpha q \geq 2\sqrt{n}$ where $0 < \alpha < 1$, solving the decision LWE problem for $(n, q, \mathcal{U}, \chi)$ with at most $m = \text{poly}(n)$ samples is at least as hard as quantumly solving GapSVP_γ and SIVP_γ on arbitrary n-dimensional lattices for some $\gamma = \tilde{O}(n/\alpha)$.

The polynomial-time reduction is extremely non-tight: approximately $O(n^{13})$.

[Regev; STOC 2005]

Solving the (approximate) shortest vector problem

The complexity of GapSVP_γ depends heavily on how γ and n relate, and get harder for smaller γ.

Algorithm	Time	Approx. factor γ
LLL algorithm	$\mathrm{poly}(n)$	$2^{\Omega(n \log\log n / \log n)}$
various	$2^{\Omega(n \log n)}$	$\mathrm{poly}(n)$
various	$2^{\Omega(n)}$ time and space	$\mathrm{poly}(n)$
Sch87	$2^{\tilde{\Omega}(n/k)}$	2^k
	$\mathrm{NP} \cap \text{co-NP}$	$\geq \sqrt{n}$
	NP-hard	$n^{o(1)}$

In cryptography, we tend to use $\gamma \approx n$.

Picking parameters

- Estimate parameters based on runtime of lattice reduction algorithms.

- Based on reductions:
  - Calculate required runtime for GapSVP or SVP based on tightness gaps and constraints in each reduction
  - Pick parameters based on best known GapSVP or SVP solvers or known lower bounds
- Based on cryptanalysis:
  - Ignore tightness in reductions.
  - Pick parameters based on best known LWE solvers relying on lattice solvers.

Ring-LWE

$$\mathbb{Z}_{13}^{7\times 4}$$

4	**1**	**11**	**10**

Cyclic structure

$\Rightarrow$ Save communication,
more efficient computation

4 KiB representation

LWE

$$\mathbb{Z}_{2^{15}}^{752\times 8}$$

8 (columns), **752** (rows)

2738	**3842**	**3345**	**2979**	**…**
2896	**595**	**3607**		
377	**1575**			
2760				
…				

752 × 8 × 15 bits = **11 KiB**

Why consider (slower, bigger) LWE?

Generic vs. ideal lattices

- Ring-LWE matrices have additional structure
  - Relies on hardness of a problem in **ideal** lattices
- LWE matrices have no additional structure
  - Relies on hardness of a problem in **generic** lattices
- NTRU also relies on a problem in a type of ideal lattices

- Currently, best algorithms for ideal lattice problems are essentially the same as for generic lattices
  - Small constant factor improvement in some cases
  - Very recent quantum polynomial time algorithm for Ideal-SVP (http://eprint.iacr.org/2016/885) but not immediately applicable to ring-LWE

If we want to eliminate this additional structure, can we still get an efficient protocol?

Public key encryption from LWE

Regev's public key encryption scheme

Let n, m, q, χ be LWE parameters.

- KeyGen(): $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$. $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$. $\mathbf{e} \xleftarrow{\$} \chi(\mathbb{Z}_q^m)$. $\tilde{\mathbf{b}} \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e}$. Return $pk \leftarrow (\mathbf{A}, \mathbf{b})$, $sk \leftarrow \mathbf{s}$.
- Enc$(pk, x \in \{0,1\})$: $\mathbf{s}' \xleftarrow{\$} \{0,1\}^m$. $\mathbf{b}' \leftarrow \mathbf{s}'\mathbf{A}$. $v' \leftarrow \langle \mathbf{s}', \mathbf{b} \rangle$. $c \leftarrow x \cdot \mathrm{encode}(v')$. Return $(\mathbf{b}', c)$.
- Dec$(sk, (\mathbf{b}', c))$: $v \leftarrow \langle \mathbf{b}', \mathbf{s} \rangle$. Return $\mathrm{decode}(v)$.

[Regev; STOC 2005]

Encode/decode

$$\mathrm{encode}(x \in \{0,1\}) \leftarrow x \cdot \left\lfloor \frac{q}{2} \right\rfloor$$

$$\mathrm{decode}(\overline{x} \in \mathbb{Z}_q) \leftarrow \begin{cases} 0, & \text{if } \overline{x} \in [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor) \\ 1, & \text{otherwise} \end{cases}$$

[Regev; STOC 2005]

Lindner–Peikert public key encryption

Let n, q, χ be LWE parameters.

- KeyGen(): $\mathbf{s} \xleftarrow{\$} \chi(\mathbb{Z}^n)$. $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times n}$. $\mathbf{e} \xleftarrow{\$} \chi(\mathbb{Z}^n)$. $\tilde{\mathbf{b}} \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e}$. Return $pk \leftarrow (\mathbf{A}, \tilde{\mathbf{b}})$ and $sk \leftarrow \mathbf{s}$.
- Enc($pk, x \in \{0,1\}$): $\mathbf{s}' \xleftarrow{\$} \chi(\mathbb{Z}^n)$. $\mathbf{e}' \xleftarrow{\$} \chi(\mathbb{Z}^n)$. $\tilde{\mathbf{b}}' \leftarrow \mathbf{s}'\mathbf{A} + \mathbf{e}'$. $e'' \xleftarrow{\$} \chi(\mathbb{Z})$. $\tilde{v}' \leftarrow \langle \mathbf{s}', \tilde{\mathbf{b}} \rangle + e''$. $c \leftarrow \mathrm{encode}(x) + \tilde{v}'$. Return $ctxt \leftarrow (\tilde{\mathbf{b}}', c)$.
- Dec($sk, (\tilde{\mathbf{b}}', c)$): $v \leftarrow \langle \tilde{\mathbf{b}}', \mathbf{s} \rangle$. Return $\mathrm{decode}(c - v)$.

[Lindner, Peikert; CT-RSA 2011]

Correctness

Sender and receiver approximately compute the same shared secret $\mathbf{s}'\mathbf{A}\mathbf{s}$

$$\tilde{v}' = \langle \mathbf{s}', \tilde{\mathbf{b}} \rangle + e'' = \mathbf{s}'(\mathbf{A}\mathbf{s} + \mathbf{e}) + e'' = \mathbf{s}'\mathbf{A}\mathbf{s} + \langle \mathbf{s}', \mathbf{e} \rangle + e'' \approx \mathbf{s}'\mathbf{A}\mathbf{s}$$

$$v = \langle \tilde{\mathbf{b}}', \mathbf{s} \rangle = (\mathbf{s}'\mathbf{A} + \mathbf{e}')\mathbf{s} = \mathbf{s}'\mathbf{A}\mathbf{s} + \langle \mathbf{e}', \mathbf{s} \rangle \approx \mathbf{s}'\mathbf{A}\mathbf{s}$$

Difference between Regev and Lindner–Peikert

Regev:

- Bob's public key is $\mathbf{s}'\mathbf{A}$ where $\mathbf{s}' \xleftarrow{\$} \{0,1\}^m$
- Encryption mask is $\langle \mathbf{s}', \mathbf{b} \rangle$

Lindner–Peikert:

- Bob's public key is $\mathbf{s}'\mathbf{A} + \mathbf{e}'$ where $\mathbf{s}' \xleftarrow{\$} \chi_e$
- Encryption mask is $\langle \mathbf{s}', \mathbf{b} \rangle + e''$

In Regev, Bob's public key is a subset sum instance. In Lindner–Peikert, Bob's public key and encryption mask is just another LWE instance.

IND-CPA security of Lindner–Peikert

Indistinguishable against chosen plaintext attacks

Theorem. If the decision LWE problem is hard, then Lindner–Peikert is IND-CPA-secure. Let n, q, χ be LWE parameters. Let $\mathcal{A}$ be an algorithm. Then there exist algorithms $\mathcal{B}_1, \mathcal{B}_2$ such that

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathbf{LP}[n,q,\chi]}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{dlwe}}_{n,q,\chi}(\mathcal{A} \circ \mathcal{B}_1) + \mathsf{Adv}^{\mathsf{dlwe}}_{n,q,\chi}(\mathcal{A} \circ \mathcal{B}_2)$$

[Lindner, Peikert; CT-RSA 2011]

IND-CPA security of Lindner–Peikert

Game 0: → Decision-LWE → Game 1: → Rewrite → Game 2:

Game 0:

1: $\mathbf{A} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^{n\times n})$
2: $\mathbf{s}, \mathbf{e} \xleftarrow{\$} \chi(\mathbb{Z}_q^n)$
3: $\tilde{\mathbf{b}} \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e}$
4: $\mathbf{s}', \mathbf{e}' \xleftarrow{\$} \chi(\mathbb{Z}_q^n)$
5: $\tilde{\mathbf{b}}' \leftarrow \mathbf{s}'\mathbf{A} + \mathbf{e}'$
6: $e'' \xleftarrow{\$} \chi(\mathbb{Z}_q)$
7: $\tilde{v}' \leftarrow \mathbf{s}'\tilde{\mathbf{b}} + e''$
8: $c_0 \leftarrow \mathrm{encode}(0) + \tilde{v}'$
9: $c_1 \leftarrow \mathrm{encode}(1) + \tilde{v}'$
10: $b^* \xleftarrow{\$} \mathcal{U}(\{0,1\})$
11: **return** $(\mathbf{A}, \tilde{\mathbf{b}}, \tilde{\mathbf{b}}', c_{b^*})$

Game 1:

1: $\mathbf{A} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^{n\times n})$
2: $\boxed{\tilde{\mathbf{b}} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^n)}$
3: $\mathbf{s}', \mathbf{e}' \xleftarrow{\$} \chi(\mathbb{Z}_q^n)$
4: $\tilde{\mathbf{b}}' \leftarrow \mathbf{s}'\mathbf{A} + \mathbf{e}'$
5: $e'' \xleftarrow{\$} \chi(\mathbb{Z}_q)$
6: $\tilde{v}' \leftarrow \mathbf{s}'\tilde{\mathbf{b}} + e''$
7: $c_0 \leftarrow \mathrm{encode}(0) + \tilde{v}'$
8: $c_1 \leftarrow \mathrm{encode}(1) + \tilde{v}'$
9: $b^* \xleftarrow{\$} \mathcal{U}(\{0,1\})$
10: **return** $(\mathbf{A}, \tilde{\mathbf{b}}, \tilde{\mathbf{b}}', c_{b^*})$

Game 2:

1: $\mathbf{A} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^{n\times n})$
2: $\tilde{\mathbf{b}} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^n)$
3: $\mathbf{s}' \xleftarrow{\$} \chi(\mathbb{Z}_q^n)$
4: $\boxed{[\mathbf{e}' \| e''] \xleftarrow{\$} \chi(\mathbb{Z}_q^{n+1})}$
5: $\boxed{[\tilde{\mathbf{b}}' \| \tilde{v}'] \leftarrow \mathbf{s}'[\mathbf{A} \| \tilde{\mathbf{b}}] + [\mathbf{e}' \| e'']}$
6: $c_0 \leftarrow \mathrm{encode}(0) + \tilde{v}'$
7: $c_1 \leftarrow \mathrm{encode}(1) + \tilde{v}'$
8: $b^* \xleftarrow{\$} \mathcal{U}(\{0,1\})$
9: **return** $(\mathbf{A}, \tilde{\mathbf{b}}, \tilde{\mathbf{b}}', c_{b^*})$

[Lindner, Peikert; CT-RSA 2011]

IND-CPA security of Lindner–Peikert

Game 2: → Decision-LWE →

1: $\mathbf{A} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^{n \times n})$
2: $\tilde{\mathbf{b}} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^{n})$
3: $\mathbf{s}' \xleftarrow{\$} \chi(\mathbb{Z}_q^{n})$
4: $\boxed{[\mathbf{e}' \| e''] \xleftarrow{\$} \chi(\mathbb{Z}_q^{n+1})}$
5: $\boxed{[\tilde{\mathbf{b}}' \| \tilde{v}'] \leftarrow \mathbf{s}'[\mathbf{A} \| \tilde{\mathbf{b}}] + [\mathbf{e}' \| e'']}$
6: $c_0 \leftarrow \mathrm{encode}(0) + \tilde{v}'$
7: $c_1 \leftarrow \mathrm{encode}(1) + \tilde{v}'$
8: $b^* \xleftarrow{\$} \mathcal{U}(\{0,1\})$
9: **return** $(\mathbf{A}, \tilde{\mathbf{b}}, \tilde{\mathbf{b}}', c_{b^*})$

Game 3: → Rewrite →

1: $\mathbf{A} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^{n \times n})$
2: $\tilde{\mathbf{b}} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^{n})$
3: $\boxed{[\tilde{\mathbf{b}}' \| \tilde{v}'] \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^{n+1})}$
4: $c_0 \leftarrow \mathrm{encode}(0) + \tilde{v}'$
5: $c_1 \leftarrow \mathrm{encode}(1) + \tilde{v}'$
6: $b^* \xleftarrow{\$} \mathcal{U}(\{0,1\})$
7: **return** $(\mathbf{A}, \tilde{\mathbf{b}}, \tilde{\mathbf{b}}', c_{b^*})$

Game 4:

1: $\mathbf{A} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^{n \times n})$
2: $\tilde{\mathbf{b}} \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^{n})$
3: $[\tilde{\mathbf{b}}' \| \tilde{v}'] \xleftarrow{\$} \mathcal{U}(\mathbb{Z}_q^{n+1})$
4: $b^* \xleftarrow{\$} \mathcal{U}(\{0,1\})$
5: **return** $(\mathbf{A}, \tilde{\mathbf{b}}, \tilde{\mathbf{b}}', \tilde{v}')$

Independent of hidden bit

[Lindner, Peikert; CT-RSA 2011]

Public key validation

- **No public key validation possible** in IND-CPA KEMs/PKEs from LWE/ring-LWE

- **Key reuse in LWE/ring-LWE** leads to real attacks following from search-decision equivalence
    - Comment in [Peikert, PQCrypto 2014]
    - Attack described in [Fluhrer, Eprint 2016]

- Need to ensure usage is okay with just IND-CPA
- Or construct IND-CCA KEM/PKE using Fujisaki–Okamoto transform or quantum-resistant variant [Targhi–Unruh, TCC 2016] [Hofheinz et al., Eprint 2017]

Direct key agreement

LWE and ring-LWE public key encryption and key exchange

Regev
STOC 2005
- Public key encryption from LWE

Lyubashevsky, Peikert, Regev
Eurocrypt 2010
- Public key encryption from ring-LWE

Lindner, Peikert
ePrint 2010, CT-RSA 2011
- Public key encryption from LWE and ring-LWE
- Approximate key exchange from LWE

Ding, Xie, Lin
ePrint 2012
- Key exchange from LWE and ring-LWE with single-bit reconciliation

Peikert
PQCrypto 2014
- Key encapsulation mechanism based on ring-LWE and variant single-bit reconciliation

Bos, Costello, Naehrig, Stebila
IEEE S&P 2015
- Implementation of Peikert's ring-LWE key exchange, testing in TLS 1.2

Basic LWE key agreement (unauthenticated)

Based on Lindner–Peikert LWE public key encryption scheme

public: “big” A in $\mathbf{Z}_q^{n \times m}$

Alice

secret:
random “small” s, e in $\mathbf{Z}_q^m$

Bob

secret:
random “small” s', e' in $\mathbf{Z}_q^n$

$b = As + e$ →

← $b' = s'A + e'$

shared secret:
$b's = s'As + e's \approx s'As$

shared secret:
$s'b \approx s'As$

These are only approximately equal ⇒ need rounding

Rounding

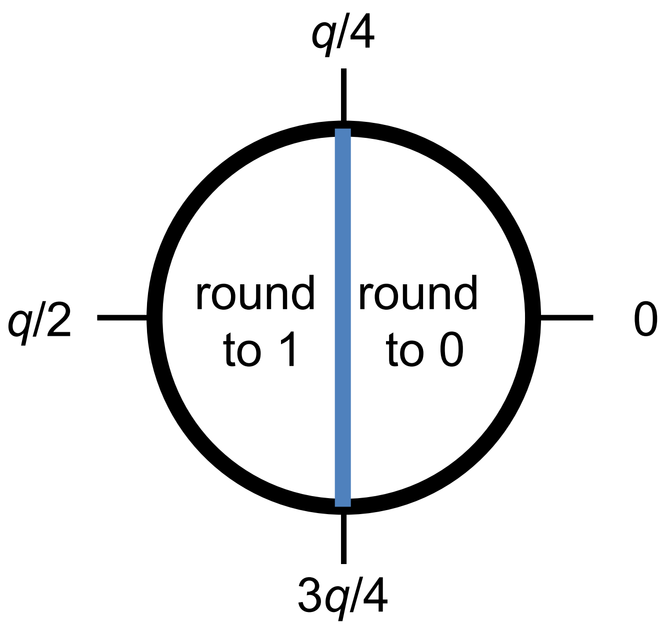
- Each coefficient of the polynomial is an integer modulo q
- Treat each coefficient independently

- Techniques by Ding [Din12] and Peikert [Pei14]

[Ding; eprint 2012] [Peikert; PQCrypto 2014]

Basic rounding

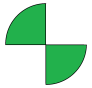
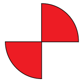
- Round either to 0 or $q/2$
- Treat $q/2$ as 1

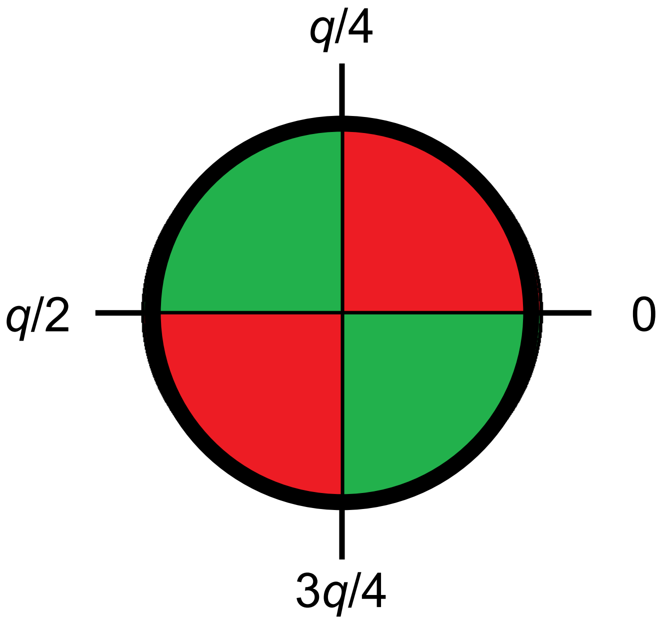


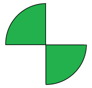
This works most of the time: prob. failure 2^{-10}.

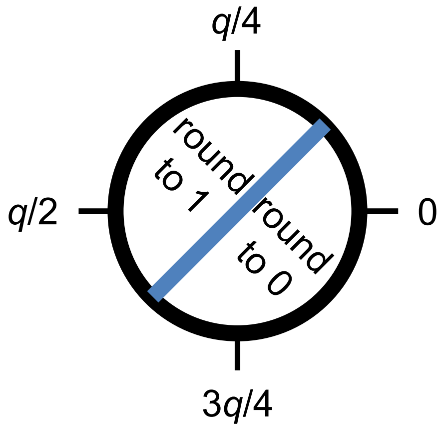
Not good enough: we need exact key agreement.

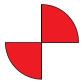
Rounding (Peikert)

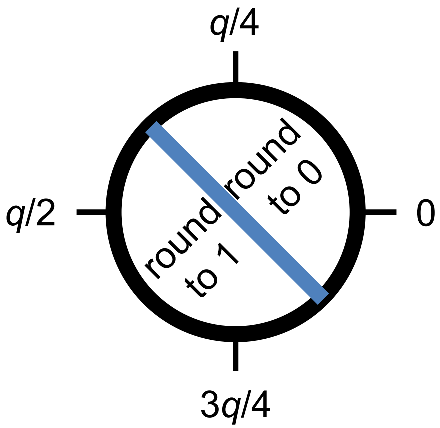
Bob says which of two regions the value is in:  or 



If 



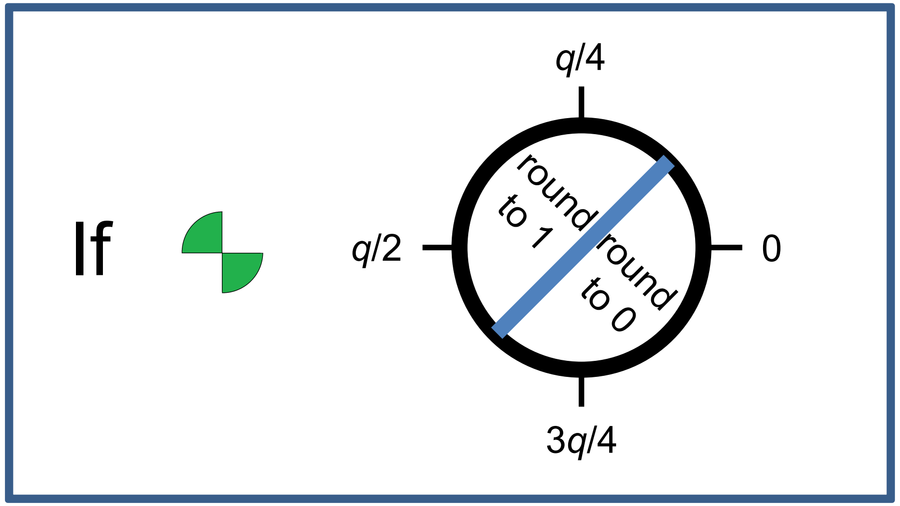
If 

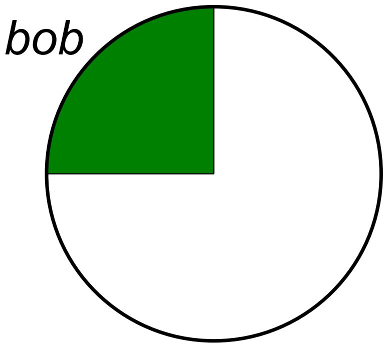


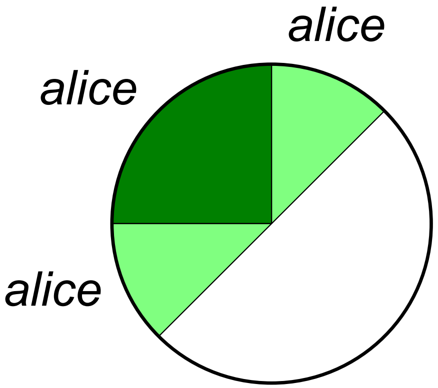
[Peikert; PQCrypto 2014]

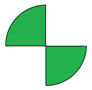
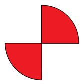
Rounding (Peikert)

- If | *alice* – *bob* | ≤ *q*/8, then this always works.







- Security not affected: revealing  or  leaks no information

[Peikert; PQCrypto 2014]

Exact LWE key agreement (unauthenticated)

public: "big" A in $\mathbf{Z}_q^{n \times m}$

Alice

secret:
random "small" s, e in $\mathbf{Z}_q^m$

Bob

secret:
random "small" s', e' in $\mathbf{Z}_q^n$

$b = As + e$ $\longrightarrow$

$\longleftarrow$ $b' = s'A + e'$,  or 

Alice shared secret:
round($b's$)

Bob shared secret:
round($s'b$)

Exact ring-LWE key agreement (unauthenticated)

public: “big” a in $R_q = \mathbf{Z}_q[x]/(x^n+1)$

Alice		**Bob**
secret: random “small” s, e in R_q		secret: random “small” s', e' in R_q
	$b = a \bullet s + e$ $\longrightarrow$	
	$\longleftarrow$ $b' = a \bullet s' + e'$, or	
shared secret: round($s \bullet b'$)		shared secret: round($b \bullet s'$)

Exact LWE key agreement – "Frodo"

Alice		Bob
$\text{seed}_{\mathbf{A}} \xleftarrow{\$} U(\{0,1\}^s)$		
$\mathbf{A} \leftarrow \text{Gen}(\text{seed}_{\mathbf{A}})$		
$\mathbf{S}, \mathbf{E} \xleftarrow{\$} \chi(\mathbb{Z}_q^{n \times \bar{n}})$		
$\boxed{\mathbf{B} \leftarrow \mathbf{A}\mathbf{S} + \mathbf{E}}$	$\xrightarrow[\in \{0,1\}^s \times \mathbb{Z}_q^{n \times \bar{n}}]{\text{seed}_{\mathbf{A}}, \mathbf{B}}$	$\mathbf{A} \leftarrow \text{Gen}(\text{seed}_{\mathbf{A}})$
		$\mathbf{S}', \mathbf{E}' \xleftarrow{\$} \chi(\mathbb{Z}_q^{\bar{m} \times n})$
		$\boxed{\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'}$
		$\mathbf{E}'' \xleftarrow{\$} \chi(\mathbb{Z}_q^{\bar{m} \times \bar{n}})$
		$\boxed{\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''}$
	$\xleftarrow[\in \mathbb{Z}_q^{\bar{m} \times n} \times \mathbb{Z}_2^{\bar{m} \times \bar{n}}]{\mathbf{B}', \mathbf{C}}$	$\mathbf{C} \leftarrow \langle \mathbf{V} \rangle_{2^B}$
$K \leftarrow \text{rec}(\boxed{\mathbf{B}'\mathbf{S}}, \mathbf{C})$		$K \leftarrow \lfloor \mathbf{V} \rceil_{2^B}$

A generated pseudorandomly

Uses two matrix forms of LWE:
- Public key is *n* x <u>*n*</u> matrix
- Shared secret is <u>*m*</u> x <u>*n*</u> matrix

Secure if decision learning with errors problem is hard (and Gen is a random oracle).

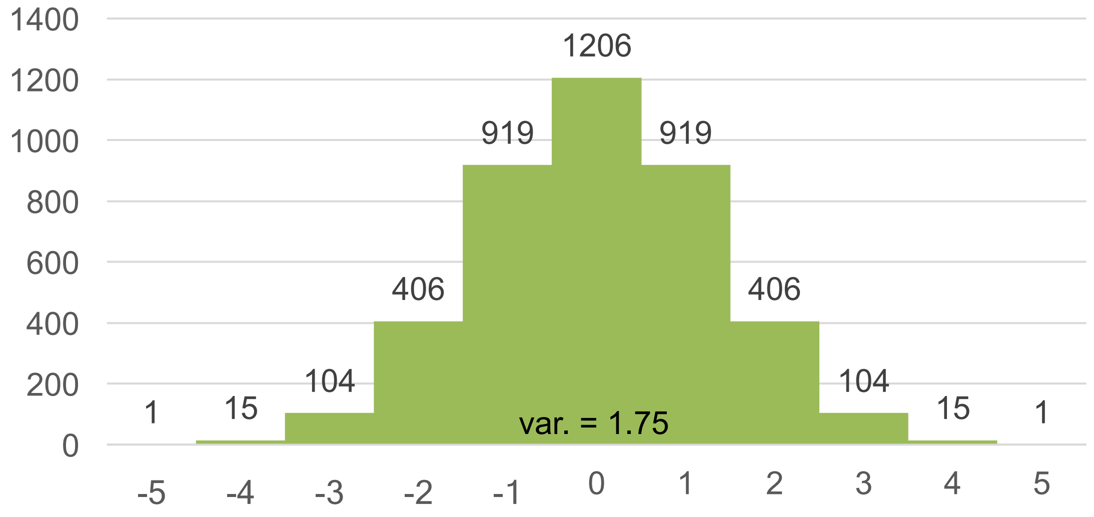
[Bos et al.; ACM CCS 2016]

Rounding

- We extract 4 bits from each of the 64 matrix entries in the shared secret.
  - More granular form of Peikert's rounding.

Parameter sizes, rounding, and error distribution all found via search scripts.

Error distribution



- Close to discrete Gaussian in terms of Rényi divergence (1.000301)
- Only requires 12 bits of randomness to sample

Parameters

All known variants of the sieving algorithm require a list of vectors to be created of this size

“Recommended”

- 144-bit classical security,
  130-bit quantum security,
  103-bit plausible lower bound

- $n = 752$, $m = 8$, $q = 2^{15}$
- χ = approximation to rounded Gaussian with 11 elements

- Failure: $2^{-38.9}$

- Total communication: 22.6 KiB

“Paranoid”

- 177-bit classical security,
  161-bit quantum security,
  128-bit plausible lower bound

- $n = 864$, $m = 8$, $q = 2^{15}$
- χ = approximation to rounded Gaussian with 13 elements

- Failure: $2^{-33.8}$

- Total communication: 25.9 KiB

Exact ring-LWE key agreement – "BCNS15"

BCNS15		
Public parameters: $n, q, \chi,\ a \leftarrow_{\$} \mathcal{U}(R_q)$		
Alice		**Bob**
$s, e \leftarrow_{\$} \chi(R_q)$		
$\boxed{\tilde{b} \leftarrow as + e} \in R_q$	$\xrightarrow{\tilde{b}}$	$s', e' \leftarrow_{\$} \chi(R_q)$
		$\boxed{\tilde{b}' \leftarrow as' + e'} \in R_q$
		$e'' \leftarrow_{\$} \chi(R_q)$
		$\boxed{\tilde{v} \leftarrow bs' + e''} \in R_q$
		$\overline{v} \leftarrow_{\$} \mathrm{dbl}(\tilde{v}) \in R_{2q}$
	$\xleftarrow{\tilde{b}', c}$	$c \leftarrow \langle \overline{v}/2 \rangle_2 \in \{0,1\}^n$
$k_A \leftarrow \mathrm{rec}_2(\boxed{2\tilde{b}'s}, c) \in \{0,1\}^n$		$k_B \leftarrow \lfloor \overline{v}/2 \rceil_2 \in \{0,1\}^n$

[Bos, Costello, Naehrig, Stebila; IEEE S&P 2015]

Parameters

160-bit classical security,
80-bit quantum security

- $n = 1024$
- $q = 2^{32}-1$
- χ = discrete Gaussian with parameter sigma = 8/sqrt(2π)

- Failure: 2^{-12800}

- Total communication: 8.1 KiB

Implementation aspect 1: Polynomial arithmetic

- Polynomial multiplication in $R_q = \mathbf{Z}_q[x]/(x^{1024}+1)$ done with Nussbaumer's FFT:

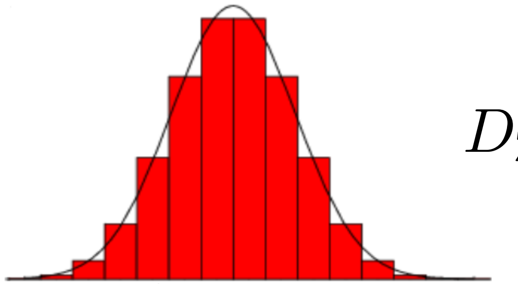
If $2^m = rk$, then

$$\frac{R[X]}{\langle X^{2^m}+1\rangle} \cong \frac{\left(\frac{R[Z]}{\langle Z^r+1\rangle}\right)[X]}{\langle X^k - Z\rangle}$$

- Rather than working modulo degree-1024 polynomial with coefficients in $\mathbf{Z}_q$, work modulo:
  - degree-256 polynomial whose coefficients are themselves polynomials modulo a degree-4 polynomial,
  - or degree-32 polynomials whose coefficients are polynomials modulo degree-8 polynomials whose coefficients are polynomials
  - or …

Implementation aspect 2:

Sampling discrete Gaussians



$$D_{\mathbb{Z},\sigma}(x) = \frac{1}{S} e^{-\frac{x^2}{2\sigma^2}} \qquad \text{for } x \in \mathbb{Z}, \sigma \approx 3.2, S = 8$$

- Security proofs require "small" elements sampled within statistical distance 2^{-128} of the true discrete Gaussian
- We use inversion sampling: precompute table of cumulative probabilities
  - For us: 52 elements, size = 10000 bits
- Sampling each coefficient requires six 192-bit integer comparisons and there are 1024 coefficients
  - 51 • 1024 for constant time

Sampling is expensive

Operation	**Cycles** constant-time	non-constant-time
sample $\xleftarrow{\$} \chi$	1 042 700	668 000
FFT multiplication	342 800	—
FFT addition	1 660	—
$\mathrm{dbl}(\cdot)$ and crossrounding $\langle\cdot\rangle_{2q,2}$	23 500	21 300
rounding $\lfloor\cdot\rceil_{2q,2}$	5 500	3,700
reconciliation $\mathrm{rec}(\cdot,\cdot)$	14 400	6 800

[Bos, Costello, Naehrig, Stebila; IEEE S&P 2015]

"NewHope"

Alkim, Ducas, Pöppelman, Schwabe.
USENIX Security 2016

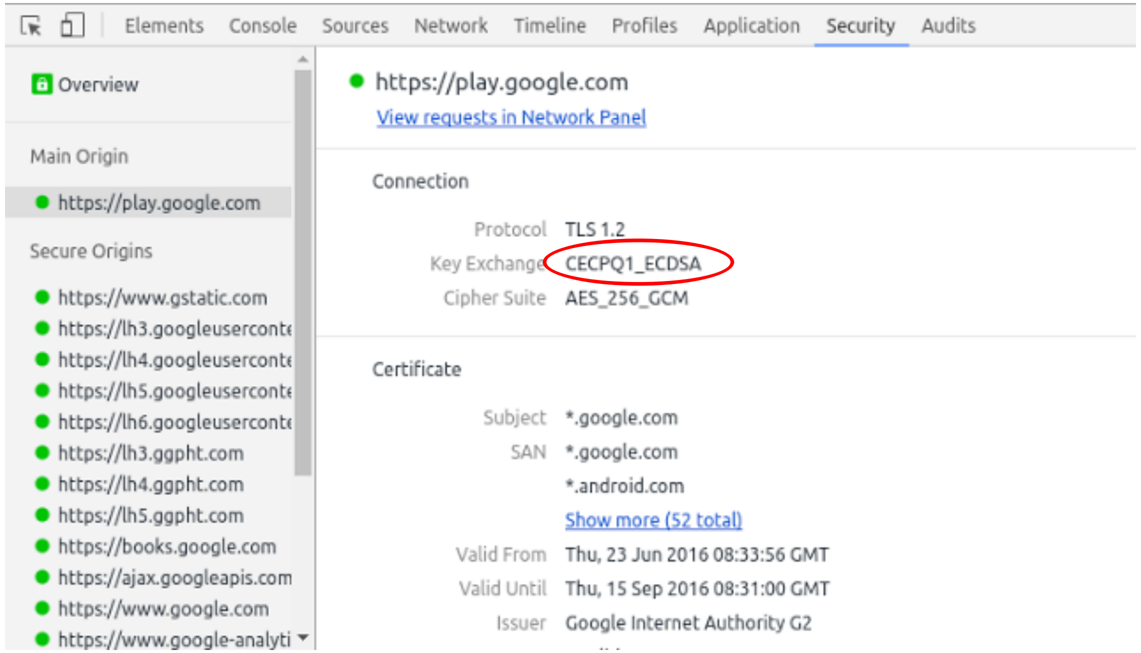
- New parameters
- Different error distribution
- Improved performance
- Pseudorandomly generated parameters

- Further performance improvements by others [GS16,LN16,AOPPS17,…]

Google Security Blog

Experimenting with Post-Quantum Cryptography

July 7, 2016



https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html

Implementations

Our implementations

- Ring-LWE BCNS15
- LWE Frodo

Pure C implementations

Constant time

Compare with others

- RSA 3072-bit (OpenSSL 1.0.1f)
- ECDH `nistp256` (OpenSSL)

Use assembly code

- Ring-LWE NewHope
- NTRU `EES743EP1`
- SIDH (Isogenies) (MSR)

Pure C implementations

Post-quantum key exchange performance

	Speed		Communication	
RSA 3072-bit	Fast	4 ms	Small	0.3 KiB
ECDH `nistp256`	Very fast	0.7 ms	Very small	0.03 KiB
Code-based	Very fast	0.5 ms	Very large	360 KiB
NTRU	Very fast	0.3–1.2 ms	Medium	1 KiB
Ring-LWE	Very fast	0.2–1.5 ms	Medium	2–4 KiB
LWE	Fast	1.4 ms	Large	11 KiB
SIDH	Med.–slow	15–400 ms	Small	0.5 KiB

See [Bos, Costello, Ducas, Mironov, Naehrig, Nikolaenko, Raghunathan, Stebila, ACM CCS 2016] for details/methodology

Other applications of LWE

Fully homomorphic encryption from LWE

- KeyGen(): $\mathbf{s} \xleftarrow{\$} \chi(\mathbb{Z}_q^n)$
- Enc($sk, \mu \in \mathbb{Z}_2$): Pick $\mathbf{c} \in \mathbb{Z}_q^n$ such that $\langle \mathbf{s}, \mathbf{c} \rangle = e \bmod q$ where $e \in \mathbb{Z}$ satisfies $e \equiv \mu \bmod 2$.
- Dec($sk, \mathbf{c}$): Compute $\langle \mathbf{s}, \mathbf{c} \rangle \in \mathbb{Z}_q$, represent this as $e \in \mathbb{Z} \cap [-\frac{q}{2}, \frac{q}{2})$. Return $\mu' \leftarrow e \bmod 2$.

[Brakerski, Vaikuntanathan; FOCS 2011]

Fully homomorphic encryption from LWE

$\mathbf{c}_1 + \mathbf{c}_2$ encrypts $\mu_1 + \mu_2$:

$$\langle \mathbf{s}, \mathbf{c}_1 + \mathbf{c}_2 \rangle = \langle \mathbf{s}, \mathbf{c}_1 \rangle + \langle \mathbf{s}, \mathbf{c}_2 \rangle = e_1 + e_2 \bmod q$$

Decryption will work as long as the error $e_1 + e_2$ remains below $q/2$.

[Brakerski, Vaikuntanathan; FOCS 2011]

Fully homomorphic encryption from LWE

Let $\mathbf{c}_1 \otimes \mathbf{c}_2 = (c_{1,i} \cdot c_{2,j})_{i,j} \in \mathbb{Z}_q^{n^2}$ be the tensor product (or Kronecker product).

$\mathbf{c}_1 \otimes \mathbf{c}_2$ is the encryption of $\mu_1 \mu_2$ under secret key $\mathbf{s} \otimes \mathbf{s}$:

$$\langle \mathbf{s} \otimes \mathbf{s}, \mathbf{c}_1 \otimes \mathbf{c}_2 \rangle = \langle \mathbf{s}, \mathbf{c}_1 \rangle \cdot \langle \mathbf{s}, \mathbf{c}_2 \rangle = e_1 \cdot e_2 \bmod q$$

Decryption will work as long as the error $e_1 \cdot e_2$ remains below $q/2$.

[Brakerski, Vaikuntanathan; FOCS 2011]

Fully homomorphic encryption from LWE

- Error conditions mean that the number of additions and multiplications is limited.
- Multiplication increases the dimension (exponentially), so the number of multiplications is again limited.

- There are techniques to resolve both of these issues.
  - **Key switching** allows converting the dimension of a ciphertext.
  - **Modulus switching** and **bootstrapping** are used to deal with the error rate.

Digital signatures [Lyubashevsky 2011]

- KeyGen(): $\mathbf{S} \stackrel{\$}{\leftarrow} \{-d, \ldots, 0, \ldots, d\}^{m \times k}$, $A \stackrel{\$}{\leftarrow} \mathbb{Z}_q^{n \times m}$, $\mathbf{T} \leftarrow \mathbf{AS}$.
  Secret key: $\mathbf{S}$; public key: $(\mathbf{A}, \mathbf{T})$.

- Sign($\mathbf{S}, \mu$): $\mathbf{y} \stackrel{\$}{\leftarrow} \chi^m$; $\mathbf{c} \leftarrow H(\mathbf{Ay}, \mu)$; $\mathbf{z} \leftarrow \mathbf{Sc} + \mathbf{y}$.
  With prob. $p(\mathbf{z})$ output $(\mathbf{z}, \mathbf{c})$, else restart Sign. "Rejection sampling"

- Vfy($(\mathbf{A}, \mathbf{T}), \mu, (\mathbf{z}, \mathbf{c})$): Accept iff $\|\mathbf{z}\| \leq \eta\sigma\sqrt{m}$ and $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc}, \mu)$

[Lyubashevsky; Eurocrypt 2012]

Post-quantum signature sizes

	Public key		Signature	
RSA 3072-bit	Small	0.3 KiB	Small	0.3 KiB
ECDSA `nistp256`	Very small	0.03 KiB	Very small	0.03 KiB
Hash-based (stateful)	Small	0.9 KiB	Medium	3.6 KiB
Hash-based (stateless)	Small	1 KiB	Large	32 KiB
Lattice-based (ignoring tightness)	Medium	1.5–8 KiB	Medium	3–9 KiB
Lattice-based (respecting tightness)	Very large	1330 KiB	Small	1.2 KiB
SIDH	Small	0.3–0.75 KiB	Very large	120–138 KiB

See [Bindel, Herath, McKague, Stebila PQCrypto 2017] for details

Summary

Summary

- LWE and ring-LWE problems
  - Search, decision, short secrets
- Reduction from GapSVP to LWE
- Public key encryption from LWE
  - Regev
  - Lindner–Peikert
- Key exchange from LWE / ring-LWE
- Other applications of LWE

More reading

- Post-Quantum Cryptography
  by Bernstein, Buchmann, Dahmen

- A Decade of Lattice Cryptography
  by Chris Peikert
  https://web.eecs.umich.edu/~cpeikert/pubs/lattice-survey.pdf