Implementation Challenges for Ideal Lattice-Based Cryptography on Reconfigurable Hardware

Cryptarchi 2014, Annecy

Thomas Pöppelmann

Ruhr University Bochum, Germany
Horst Görtz Institute for IT-Security
Advisor: Prof. Dr.-Ing. Tim Güneysu

1 July 2014

Outline

- Introduction and Motivation
- Ideal Lattices
- Ring-LWE Public Key Encryption
- Lattice-Based Signatures
- Conclusion

Motivation - Lattice-Based Cryptography

- Post-quantum and alternative cryptography
  - Quantum computers break ECC and RSA - we need alternatives
  - "Penetrating Hard Targets." - 79.7 million dollar NSA quantum computer research program
  - Classical cryptanalysis of ECC and RSA (e.g., Antoine Joux's work)
- Why focus on lattice-based cryptography?
  - More versatile than code-based, MQ, and hash-based schemes
  - Large amount of theoretical foundations and progress
  - Practical aspects only researched since approx. 3 years

Outline

- Introduction and Motivation
- **Ideal Lattices**
- Ring-LWE Public Key Encryption
- Lattice-Based Signatures
- Conclusion

Ideal Lattices

- Ideal lattices
  - Ideal lattices correspond to ideals in the ring $R = \mathbb{Z}_q[x]/\langle f(x) \rangle$ for some irreducible polynomial function f
  - Introduces algebraic structure into previously random lattices - no serious advantage for attackers so far
  - Common choice is $\mathbb{Z}_q[x]/\langle x^n + 1 \rangle$ for n being a power of two and q a prime such that $q = 1 \mod 2n$

- Basic operation is polynomial multiplication
  - Like point multiplication for ECC or exponentiation for RSA
  - Available algorithms:
    - Schoolbook multiplication: $\mathcal{O}(n^2)$
    - Karatsuba: $\mathcal{O}(n^{\log_2(3)})$
    - FFT/NTT: $\mathcal{O}(n \log n)$

Example

Fix $q = 5$ and $n = 4 \rightarrow v, k \in \mathbb{Z}_5[x]/\langle f = x^4 + 1 \rangle$

- $v = 4x^3 + 2x^2 + 0x^1 + 1 = (4, 2, 0, 1)$
- $k = 2x^3 + 1x^2 + 4x^1 + 0 = (2, 1, 4, 0)$

Addition is usual coordinate-wise addition:

- $s = v + k = (4 + 2 \bmod 5, 2 + 1, 4, 1) = (1, 3, 4, 1)$

Multiplication is usual polynomial multiplication followed by reduction modulo $x^n + 1$

$$z = s \cdot k = 1 \quad 3 \quad 4 \quad 1 \quad \cdot \quad 2 \quad 1 \quad 4 \quad 0 \tag{1}$$

$$4 \; 12 \; 16 \; 4 \tag{2}$$

$$1 \; 3 \;\; 4 \;\; 1 \tag{3}$$

$$2 \; 6 \; 8 \;\; 2 \tag{4}$$

$z = s \cdot k = (2, 7, 15, 18, 17, 4, 0) \bmod 5 \equiv (2, 2, 0, 3, 2, 4, 0) \bmod x^4 + 1 \equiv (3, 0, 2, 0) \bmod 5$

Challenge 1: Number Theoretic Transform

Theorem (Wrapped Convolution)

Let ω be a primitive n-th root of unity in $\mathbb{Z}_q$ and $\psi^2 = \omega$.

1. *Let d be the negative wrapped convolution of a and b. Let $\bar{a}, \bar{b}$ and $\bar{d}$ be defined as $(a_0, \psi a_1, \ldots, \psi^{n-1} a_{n-1})$, $(b_0, \psi b_1, \ldots, \psi^{n-1} b_{n-1})$, and $(d_0, \psi d_1, \ldots, \psi^{n-1} d_{n-1})$. Then $\bar{d} = NTT_w^{-1}(NTT_w(\bar{a}) \circ NTT_w(\bar{b}))$.*

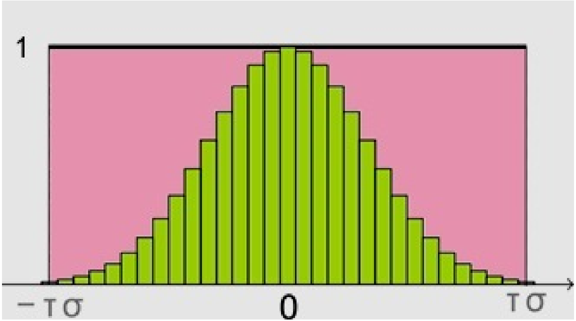
Advantages:

- Reduction by $x^n + 1$ for free and no zero padding
- Store constants (e.g., **a**) in NTT representation
- Only $\frac{1}{2} n \log n$ multiplications for one NTT

Disadvantage:

- Storage or computation of powers of $\omega, \psi, \omega^{-1}, \psi^{-1}$
- Parameter dependent

Challenge 2: Discrete Gaussian Sampling



- D_σ is defined by assigning weight proportional to $\rho_\sigma(x) = \exp(\frac{-x^2}{2\sigma^2})$ for all integers x
- Tailcut τ and precision λ define approximation to real Gaussian with std. deviation σ
- Rejection sampling
  - Sample $x \in [-\tau\sigma, \tau\sigma]$
  - Choose $r \in [0, 1]$
  - Accept if $r < \rho_\sigma(x)/\rho_\sigma(\mathbb{Z})$

Outline

Hardness Assumptions

Established lattice hardness assumption:

Definition (**Decisional Ring-LWE**)

Given $(\mathbf{a}_1, \mathbf{t}_1), \ldots, (\mathbf{a}_m, \mathbf{t}_m) \in \mathcal{R} \times \mathcal{R}$. Decide whether $\mathbf{t}_i = \mathbf{a}_i \mathbf{s} + \mathbf{e}_i$ where $\mathbf{s}, \mathbf{e}_1, \ldots, \mathbf{e}_m \leftarrow D_\sigma$ and $\mathbf{a}_i \xleftarrow{\$} \mathcal{R}$ or uniformly random from $\mathcal{R} \times \mathcal{R}$ (D_σ denotes a Gaussian distribution).

- In search version asks to find $\mathbf{s}$
- Decisional and search problem are equivalent
- Basic problem (besides SIS) used for encryption, signatures, homomorphic cryptography

Ring-LWE Encryption [LPR10,LP11]

$\textsc{Gen}(\mathbf{a})$:	Choose $\mathbf{r}_1, \mathbf{r}_2 \in D_\sigma$ and let $\mathbf{p} = \mathbf{r}_1 - \mathbf{a}\mathbf{r}_2 \in R$. The public key is $\mathbf{p}$ and the secret key is $\mathbf{r}_2$.
$\textsc{Enc}(\mathbf{a}, \mathbf{p}, m \in \{0,1\}^n)$:	Choose the noise terms $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \in D_\sigma$. Let $\bar{\mathbf{m}} = \textsc{encode}(m) \in R$, and compute the ciphertext $[\mathbf{c}_1 = \mathbf{a}\mathbf{e}_1 + \mathbf{e}_2, \mathbf{c}_2 = \mathbf{p}\mathbf{e}_1 + \mathbf{e}_3 + \bar{\mathbf{m}}] \in R^2$.
$\textsc{Dec}(\mathbf{c} = [\mathbf{c}_1, \mathbf{c}_2], \mathbf{r}_2)$:	Output $\textsc{decode}(\mathbf{c}_1\mathbf{r}_2 + \mathbf{c}_2) \in \{0,1\}^n$.

- Correctness: $\mathbf{c}_1 \cdot \mathbf{r}_2 + \mathbf{c}_2 = \mathbf{r}_2\mathbf{a}\mathbf{e}_1 + \mathbf{r}_2\mathbf{e}_2 + (\mathbf{r}_1 - \mathbf{a}\mathbf{r}_2)\mathbf{e}_1 + \mathbf{e}_3 + \bar{\mathbf{m}} = \bar{\mathbf{m}} + \overbrace{\mathbf{e}_1 \cdot \mathbf{r}_1 + \mathbf{e}_2 \cdot \mathbf{r}_2 + \mathbf{e}_3}^{\text{small Gaussian noise}}$
- $\textsc{encode}()$ assign bit $0 \to 0$ and bit $1 \to q/2$. Thus needs $|\mathbf{e}_1 \cdot \mathbf{r}_1 + \mathbf{e}_2 \cdot \mathbf{r}_2 + \mathbf{e}_3| < q/4$
- Security against chosen plaintext attacks (CPA) follows from Ring-LWE assumption

Ring-LWE Encryption

n	q	Bit Sec.	Size [bits]		
			Secret Key	Public Key	Ciphertext
			n	$n\lceil\log_2(q)\rceil$	$2n\lceil\log_2(q)\rceil$
256	4093	≈ 100	1792	3072	6144
256	7681	≈ 100	1792	3328	6656
512	12289	≈ 256	4096	7168	14336

- Scheme is a good benchmark - proably not ready for practice, yet
- Parameters proposed by Göttert et al.[1] and Linder/Peikert[2]
- Relatively large ciphertext expansion of $2\lceil\log_2 q\rceil$

[1]Norman Göttert, Thomas Feller, Michael Schneider, Johannes Buchmann, Sorin A. Huss: On the Design of Hardware Building Blocks for Modern Lattice-Based Encryption Schemes. CHES 2012

[2]Richard Lindner, Chris Peikert: Better Key Sizes (and Attacks) for LWE-Based Encryption. CT-RSA 2011

Techniques for High-Performance: NTT

Domain Parameters

Temporary value: $r_1 = \texttt{sample}()$, Global constant: $\tilde{a} = \texttt{NTT}(a)$
Secret key: $\tilde{r}_2 = \texttt{NTT}(\texttt{sample}())$, Public key: $\tilde{p} = \texttt{NTT}(r_1 - \texttt{INTT}(\tilde{a} \circ \tilde{r}_2))$

Algorithm Enc$(\tilde{a}, \tilde{p}, m \in \{0,1\}^n)$

1: $e_1, e_2, e_3 = \texttt{sample}()$
2: $\tilde{e}_1 = \texttt{NTT}(e_1)$
3: $\tilde{h}_1 = \tilde{a} \circ \tilde{e}_1, \tilde{h}_2 = \tilde{p} \circ \tilde{e}_1$
4: $h_1 = \texttt{INTT}(\tilde{h}_1), h_2 = \texttt{INTT}(\tilde{h}_2)$
5: $c_1 = h_1 + e_2$
6: $c_2 = h_2 + e_3 + \texttt{encode}(m)$

Algorithm Dec$(c_1, c_2, \tilde{r}_2)$

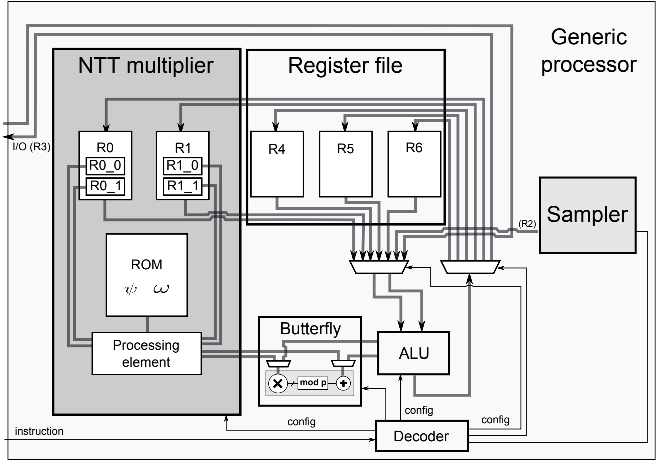
1: $\tilde{h}_1 = \texttt{NTT}(c_1)$
2: $\tilde{h}_2 = \tilde{c}_1 \circ \tilde{r}_2$
3: $m = \texttt{decode}(\texttt{INTT}(\tilde{h}_2) + c_2)$

- Encryption/Decryption: 3/2 NTT operations
- If $\mathbf{c}_1, \mathbf{c}_2$ are send in NTT format even more savings possible (3/1 NTT operations)

Techniques for High-Performance: Processor

- Polynomial arithmetic is a basic operation in ideal lattice-based cryptography
  - Building hardware is expensive (PhD student perspective: time consuming)
  - Parameters may change - the implementation should cover that
  - Provide a useful building block
- Available instructions (one register = one polynomial)
  - NTT(r_1): Execute the NTT on register r_1
  - INTT(r_1): Execute the inverse NTT on register r_1
  - PW_MUL(r_1, r_2): Perform point-wise multiplication ($r_1 \leftarrow r_1 \circ r_2$)
  - MOV(r_1, r_2): Move polynomial from one register to another ($r_1 \leftarrow r_2$)
  - ADD(r_1, r_2): Add two polynomials ($r_1 \leftarrow r_1 + r_2$)
  - SUB(r_1, r_2): Subtract two polynomials ($r_1 \leftarrow r_1 - r_2$)

Techniques for High-Performance: Processor



An Evolution of Implementations

Scheme	Device	Resources	Speed
Ring-LWE (n=256) [Göttert et al., CHES'2012]	V6 V6 V6	[Gen] 146k LUT/82k FF [Enc] 298k LUT/143k FF [Dec] 124k LUT/65k FF	- 8.05 μs 8.10 μs
Our Work (n=256) [Pöppelmann et al., SAC'2013]	V6	[Gen/Enc/Dec] 4k LUT/3k FF/ 12 BRAM(18K)/1 DSP48	27.61 μs 26.19 μs 16.80 μs
Ring-LWE (**n=512**) [Roy et al., Eprint2013/866.]	V6	[Enc/Dec] 1.8k LUT/1.1k FF/ 3 BRAM(18K)/1 DSP48	53.1 μs 21.3 μs
Ring-LWE [Enc/Dec] (n=256) [Pöppelmann et al., ISCAS'2014]	S6	[Enc] 0.4k LUT/0.3k FF/ 1 BRAM(18K)/1 DSP48 [Dec] 0.1k LUT/0.1k FF/ 0.5 BRAM(18K)/1 DSP48	1070 μs 370 μs

- Huge improvements since first implementation in 2012
- Roy et al. provide smaller implementation for higher security level
- Lightweight is also possible

Outline

- Introduction and Motivation
- Ideal Lattices
- Ring-LWE Public Key Encryption
- **Lattice-Based Signatures**
- Conclusion

Lattice-Based Signature Schemes

- Most promising lattice-based signature schemes
  - GLP[3]: $\approx$ 80 bit security, 9000 bit signature, 11800 bit public key, fast
  - BLISS[4]: 128 bit security, 5600 bit signature, 7000 bit public key, very fast

- Comparison
  - Schemes are quite similar and based on similar ideas
  - Advantage of BLISS possible due to usage of discrete Gaussian noise (instead of uniform).
  - Both rely on (more or less) non-standard assumptions

[3] Practical lattice-based cryptography: A signature scheme for embedded systems, Tim Güneysu, Vadim Lyubashevsky, Thomas Pöppelmann, CHES 2012

[4] Leo Ducas, Alain Durmus, Tancrede Lepoint, Vadim Lyubashevsky: Lattice Signatures and Bimodal Gaussians. CRYPTO 2013

BLISS: Algorithm

Algorithm KeyGen()

Choose $\mathbf{f}, \mathbf{g}$ with $d_1 = \lceil \delta_1 n \rceil$ entries in $\{\pm 1\}$
$\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2)^t \leftarrow (\mathbf{f}, 2\mathbf{g} + 1)^t$
$\mathbf{a}_q = (2\mathbf{g} + 1)/\mathbf{f} \bmod q$ (**restart** if $\mathbf{f}$ is not invertible)
Return$(pk = \mathbf{A}, sk = \mathbf{S})$ where $\mathbf{A} = (\mathbf{a}_1 = 2\mathbf{a}_q, q - 2) \bmod 2q$

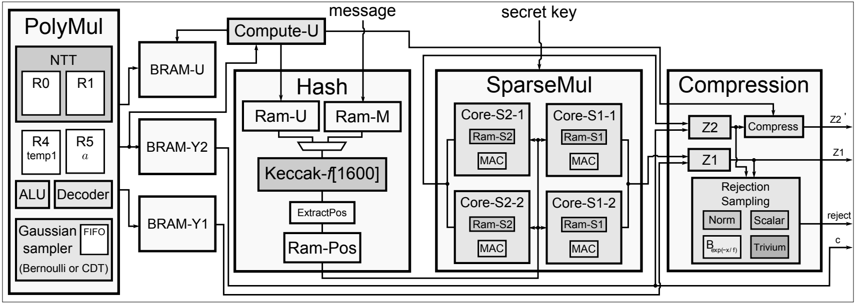
Alg. Sign$(\mu, pk{=}\mathbf{A}, sk{=}\mathbf{S})$

$\mathbf{y}_1, \mathbf{y}_2 \leftarrow D_{\mathbb{Z}^n, \sigma}$
$\mathbf{u} = \zeta \cdot \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$
$\mathbf{c} \leftarrow H(\lfloor \mathbf{u} \rceil_d \bmod p, \mu)$
Choose a random bit b
$\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
$\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$
Continue with probability
$1 \Big/ \left(M \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2} \right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc} \rangle}{\sigma^2} \right) \right)$
otherwise **restart**
Return $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$

Alg. Verify$(\mu, pk = \mathbf{A}, (\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c}))$

if $\|(\mathbf{z}_1 | 2^d \cdot \mathbf{z}_2^\dagger)\|_2 > B_2$ **then** Reject
if $\|(\mathbf{z}_1 | 2^d \cdot \mathbf{z}_2^\dagger)\|_\infty > B_\infty$ **then** Reject
Accept iff $\mathbf{c} = H\big(\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} \rceil_d + \mathbf{z}_2^\dagger \bmod p, \mu\big)$

BLISS Implementation



- Implementation of BLISS using ideal lattice processor and Keccak hash[5]
- Lattice processor to compute $\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2$
- Sparse multiplier for $\mathbf{z}_{1,2} \leftarrow \mathbf{y}_{1,2} + (-1)^b \mathbf{s}_{1,2}\mathbf{c}$

[5] Thomas Pöppelmann and Léo Ducas and Tim Güneysu: Enhanced Lattice-Based Signatures on Reconfigurable Hardware, Eprint2014/254

BLISS: Gaussian Sampling

- Standard deviation $\sigma \approx 215$ much larger than for encryption ($\sigma \approx 4.5$)
- High speed Gaussian sampling is required (one signature at least $2n = 1024$ samples)
- High performance option is the cumulative distribution table (CTD)
  - Precompute table $T[k] = \sum_{i \leq k} \rho(i)/\rho(\mathbb{Z})$
  - Sample a uniform real $r \in [0, 1)$
  - Use binary search to find i s.t. $T[i] < r \leq T[i+1]$
  - Return $\pm i$ (reject 50% of all $i = 0$)
- However: Naive CDT sampler requires precomputed table of 50 KBytes (23 18K Block RAMs)

BLISS: Gaussian Sampling

- Efficient storage using floating point representation
- Fast search using short cut intervals
- Reduction of table size using a Kullback-Leibler divergence argument
- Usage of convolution theorem
  - Given Gaussian x_1, x_2 with variances σ_1^2, σ_2^2, then their combination $x_1 + kx_2$ is Gaussian with variance $\sigma_1^2 + k^2\sigma_2^2$ (under certain smoothing condition)
  - We set $k = 11$, $\sigma' = \sigma/\sqrt{1+k^2} \approx 19.53$, and sample $x = x_1 + kx_2'$ for $x_1, x_2' \leftarrow D_{\sigma'}$
  - No only table for σ' required
- Table now 1.8 KBytes with almost no performance impact

Results and Comparison

Operation	Resources	Ops/s
BLISS-SIGN [Eprint'14]	7491LUT/7033FF/7.5DSP/6BRAM	7,958
BLISS-VER [Eprint'14]	5275LUT/4488FF/4.5DSP/3BRAM	14,438
GLP-SIGN [unpublished]	5614LUT/6188FF/4DSP/18.5BRAM	1,715
GLP-VER [unpublished]	3966LUT/4318FF/4DSP/14.5BRAM	7,438
SIGN-Only [GLP'12]	7465LUT/8993FF/28DSP/29.5BRAM	931
VER-Only [GLP'12]	6225LUT/6663FF/8DSP/15BRAM	998
RSA-Signature (1024)	3937LS/17DSPs	548
ECDSA (NIST-P224)	1580LS/26DSPs	2,739
ECDSA (ECC $GF(2^m)$)	8300LUTs/7BRAMs	24,390

Outline

- Introduction and Motivation
- Ideal Lattices
- Ring-LWE Public Key Encryption
- Lattice-Based Signatures
- **Conclusion**

Conclusion

- Main challenges:
  - Fast, efficient, and small polynomial arithmetic (especially NTT)
  - Fast, efficient, and small Gaussian sampling
  - Sparse multiplication, rejection sampling, large keys

- Future challenges and opportunities:
  - First: Create trust in parameters through cryptanalysis
  - How efficient are "advanced" lattice construction (IBE, SHE/FHE, multilinear maps) or trapdoor-based signatures?
  - Protection against side-channel attacks
  - Can we move away from Gaussians - at what cost?

Thank you for your attention!
Any questions?

Papers, VHDL, and C code:
`sha.rub.de/research/projects/lattice/`

Contact:
thomas.poeppelmann@rub.de