Improvement and Efficient Implementation of a Lattice-based Signature scheme

Rachid El Bansarkhani, Johannes Buchmann
Technische Universität Darmstadt

TU Darmstadt
August 2013

Outline

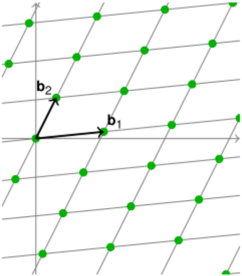
- Introduction to Lattice-based Crypto
- Lattice-based Hash Function
- Lattice-based Signature Scheme
- Contributions
- Experimental Resaults

A lattice is the set of all integer linear combinations of (linearly independent) basis vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \in \mathbb{R}^n$:

$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z} = \{\mathbf{Bx} \ : \mathbf{x} \in \mathbb{Z}^n\} \subset \mathbb{R}^n$$

A lattice has infinitely many bases:

$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{c}_i \cdot \mathbb{Z}$$



Definition (Lattices)

A discrete additive subgroup of $\mathbb{R}^n$

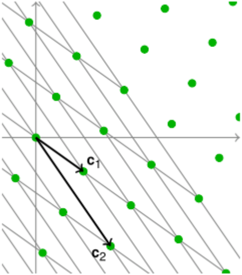
Introduction

A lattice is the set of all integer linear combinations of (linearly independent) basis vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \in \mathbb{R}^n$:

$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z} = \{\mathbf{B}\mathbf{x} \ : \mathbf{x} \in \mathbb{Z}^n\} \subset \mathbb{R}^n$$

A lattice has infinitely many bases:

$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{c}_i \cdot \mathbb{Z}$$



Definition (Lattices)

A discrete additive subgroup of $\mathbb{R}^n$

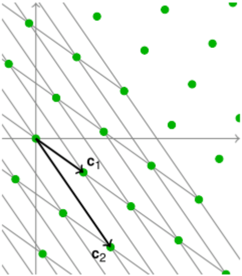
Introduction

A lattice is the set of all integer linear combinations of (linearly independent) basis vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \in \mathbb{R}^n$:

$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z} = \{\mathbf{Bx} \ : \mathbf{x} \in \mathbb{Z}^n\} \subset \mathbb{R}^n$$

A lattice has infinitely many bases:

$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{c}_i \cdot \mathbb{Z}$$

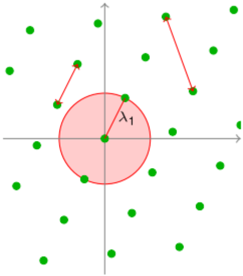


Definition (Lattices)

A discrete additive subgroup of $\mathbb{R}^n$

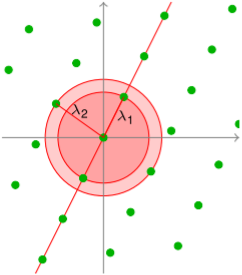
The shortest vector v in a lattice:
lattice point with minimum distance $\lambda_1 = \| v \|$ to the origin

- $\lambda_1(\mathcal{L}) = \min_{\mathbf{x} \neq \mathbf{0},\ \mathbf{x} \in \mathcal{L}} \| x \|$
- More generally, λ_k denotes the smallest radius of a ball containing k linearly independent vectors



The shortest vector v in a lattice:
lattice point with minimum distance $\lambda_1 = \| v \|$ to the origin

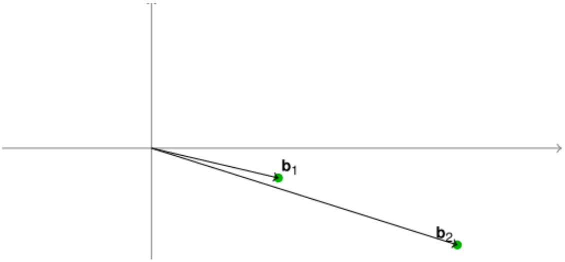
- $\lambda_1(\mathcal{L}) = \min_{\mathbf{x} \neq \mathbf{0},\ \mathbf{x} \in \mathcal{L}} \| x \|$
- More generally, λ_k denotes the smallest radius of a ball containing k linearly independent vectors



Computational Problems

Definition (Shortest Vector Problem)

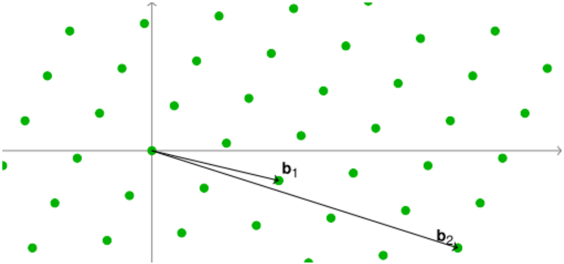
Given a basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$, find the shortest nonzero vector $\mathbf{v}$ in the lattice $\mathcal{L}(\mathbf{B})$, i.e. $\| \mathbf{v} \| = \lambda_1$



Computational Problems

Definition (Shortest Vector Problem)

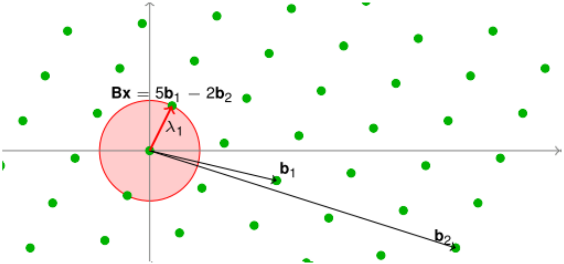
Given a basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$, find the shortest nonzero vector $\mathbf{v}$ in the lattice $\mathcal{L}(\mathbf{B})$, i.e. $\| \mathbf{v} \| = \lambda_1$



Computational Problems

Definition (Shortest Vector Problem)

Given a basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$, find the shortest nonzero vector $\mathbf{v}$ in the lattice $\mathcal{L}(\mathbf{B})$, i.e. $\| \mathbf{v} \| = \lambda_1$



Hash function

Lattice-based hash function [Ajtai96]:

$$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \mod q$$

Input parameters:

- $q \in \mathbb{Z}$ (e.g. 2^{19})
- Choose $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ uniformly at random, n (e.g. n=256) is main security parameter
- $m > n \cdot \log_2 q$
- $\mathbf{x}$ is from a bounded domain, e.g. $\mathbf{x} \in \{0, 1\}^n$

Hash function

Lattice-based hash function [Ajtai96]:

$$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \mod q$$

Input parameters:

- $q \in \mathbb{Z}$ (e.g. 2^{19})
- Choose $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ uniformly at random, n (e.g. n=256) is main security parameter
- $m > n \cdot \log_2 q$
- $\mathbf{x}$ is from a bounded domain, e.g. $\mathbf{x} \in \{0,1\}^n$

Hash function

Lattice-based hash function [Ajtai96]:

$$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \mod q$$

Input parameters:

- $q \in \mathbb{Z}$ (e.g. 2^{19})
- Choose $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ uniformly at random, n (e.g. n=256) is main security parameter
- $m > n \cdot \log_2 q$
- $\mathbf{x}$ is from a bounded domain, e.g. $\mathbf{x} \in \{0, 1\}^n$

Hash function

Lattice-based hash function [Ajtai96]:

$$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \mod q$$

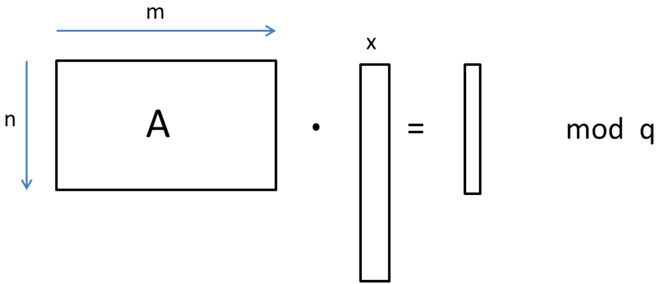
Input parameters:

- $q \in \mathbb{Z}$ (e.g. 2^{19})
- Choose $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ uniformly at random, n (e.g. n=256) is main security parameter
- $m > n \cdot \log_2 q$
- $\mathbf{x}$ is from a bounded domain, e.g. $\mathbf{x} \in \{0,1\}^n$

Hash Function

$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \mod q$:

- is a compression function
- maps m bits to $n \log_2 q$ bits
- inversion and finding collisions as hard as worst-case lattice problems



Hash Function

Hardness of finding collisions

Finding collisions in the average case, where $\mathbf{A}$ is chosen at random, is hard, provided approximating SIVP is hard in the worst-case

From Hash Functions to a Signature Scheme

Signature scheme by Gentry, Peikert and Vaikunthanatan [GPV08] using Preimage Sampleable Trapdoor Functions (PSTF):

Hash-and-Sign for lattices

- Keygen: random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and trapdoor $\mathbf{R}$, RO $\mathbf{H}(\cdot)$, PSTF: $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \mod q$
- Signing of message $\mathbf{m}$: signature $\sigma = f_{\mathbf{A}}^{-1}(\mathbf{H}(\mathbf{m}))$ using trapdoor $\mathbf{R}$.
- Verification: $\| \sigma \| \leq \mathit{bound}$ and $f_{\mathbf{A}}(\sigma) = \mathbf{H}(\mathbf{m})$

- Similar to RSA Hash-and-Sign, but Verification process differs
- Forging signatures as hard as inverting lattice-based hash functions
- Secure in the RO

From Hash Functions to a Signature Scheme

Signature scheme by Gentry, Peikert and Vaikunthanatan [GPV08] using Preimage Sampleable Trapdoor Functions (PSTF):

Hash-and-Sign for lattices

- Keygen: random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and trapdoor $\mathbf{R}$, RO $\mathbf{H}(\cdot)$, PSTF: $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \mod q$
- Signing of message $\mathbf{m}$: signature $\sigma = f_{\mathbf{A}}^{-1}(\mathbf{H}(\mathbf{m}))$ using trapdoor $\mathbf{R}$.
- Verification: $\| \sigma \| \leq \textit{bound}$ and $f_{\mathbf{A}}(\sigma) = \mathbf{H}(\mathbf{m})$

- Similar to RSA Hash-and-Sign, but Verification process differs
- Forging signatures as hard as inverting lattice-based hash functions
- Secure in the RO

From Hash Functions to a Signature Scheme

Signature scheme by Gentry, Peikert and Vaikunthanatan [GPV08] using Preimage Sampleable Trapdoor Functions (PSTF):

Hash-and-Sign for lattices

- Keygen: random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and trapdoor $\mathbf{R}$, RO $\mathbf{H}(\cdot)$, PSTF: $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \mod q$
- Signing of message $\mathbf{m}$: signature $\sigma = f_{\mathbf{A}}^{-1}(\mathbf{H}(\mathbf{m}))$ using trapdoor $\mathbf{R}$.
- Verification: $\| \sigma \| \leq$ *bound* and $f_{\mathbf{A}}(\sigma) = \mathbf{H}(\mathbf{m})$

- Similar to RSA Hash-and-Sign, but Verification process differs
- Forging signatures as hard as inverting lattice-based hash functions
- Secure in the RO

From Hash Functions to a Signature Scheme

Signature scheme by Gentry, Peikert and Vaikunthanatan [GPV08] using Preimage Sampleable Trapdoor Functions (PSTF):

Hash-and-Sign for lattices

- Keygen: random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and trapdoor $\mathbf{R}$, RO $\mathbf{H}(\cdot)$, PSTF: $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \mod q$
- Signing of message $\mathbf{m}$: signature $\sigma = f_{\mathbf{A}}^{-1}(\mathbf{H}(\mathbf{m}))$ using trapdoor $\mathbf{R}$.
- Verification: $\| \sigma \| \leq \mathit{bound}$ and $f_{\mathbf{A}}(\sigma) = \mathbf{H}(\mathbf{m})$

- Similar to RSA Hash-and-Sign, but Verification process differs
- Forging signatures as hard as inverting lattice-based hash functions
- Secure in the RO

From Hash Functions to a Signature Scheme

Signature scheme by Gentry, Peikert and Vaikunthanatan [GPV08] using Preimage Sampleable Trapdoor Functions (PSTF):

Hash-and-Sign for lattices

- Keygen: random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and trapdoor $\mathbf{R}$, RO $\mathbf{H}(\cdot)$, PSTF: $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \mod q$
- Signing of message $\mathbf{m}$: signature $\sigma = f_{\mathbf{A}}^{-1}(\mathbf{H}(\mathbf{m}))$ using trapdoor $\mathbf{R}$.
- Verification: $\| \sigma \| \leq$ *bound* and $f_{\mathbf{A}}(\sigma) = \mathbf{H}(\mathbf{m})$

- Similar to RSA Hash-and-Sign, but Verification process differs
- Forging signatures as hard as inverting lattice-based hash functions
- Secure in the RO

From Hash Functions to a Signature Scheme

Signature scheme by Gentry, Peikert and Vaikunthanatan [GPV08] using Preimage Sampleable Trapdoor Functions (PSTF):

Hash-and-Sign for lattices

- Keygen: random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and trapdoor $\mathbf{R}$, RO $\mathbf{H}(\cdot)$, PSTF: $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \mod q$
- Signing of message $\mathbf{m}$: signature $\sigma = f_{\mathbf{A}}^{-1}(\mathbf{H}(\mathbf{m}))$ using trapdoor $\mathbf{R}$.
- Verification: $\| \sigma \| \leq$ *bound* and $f_{\mathbf{A}}(\sigma) = \mathbf{H}(\mathbf{m})$

- Similar to RSA Hash-and-Sign, but Verification process differs
- Forging signatures as hard as inverting lattice-based hash functions
- Secure in the RO

From Hash Hunctions to a Signature Scheme

Main challenge:

- How to generate random Matrix $\mathbf{A}$, enabling the signer to sign messages?
- **Solution**: Use the trapdoor $\mathbf{R}$ to generate a random matrix $\mathbf{A}$.

Construction of A according to Micciancio an Peikert [MP12]:

$$\mathbf{A} = \left[\ \bar{\mathbf{A}} \quad \middle| \quad \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}\ \right]$$

Parameters:

- $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times n}$ is uniformly dist.
- $\mathbf{R} \in \mathbb{Z}^{n \times nk}$ is the secret/trapdoor (small entries)

$\Rightarrow$ $\mathbf{A}$ is pseudorandom (comp. instantiation)

From Hash Functions to a Signature Scheme

Implementation issues:

- $q = 2^k$ more suitable for practice
- entries of $\mathbf{R}$ are sampled from a discrete Gaussian
- $$\mathbf{G} = \begin{bmatrix} 1 & 2 & \dots & 2^{k-1} & & & & 0 \\ & & & & \ddots & & & \\ & & 0 & & & 1 & 2 & \dots & 2^{k-1} \end{bmatrix}$$

From Hash Functions to a Signature Scheme

Implementation issues:

- $q = 2^k$ more suitable for practice
- entries of $\mathbf{R}$ are sampled from a discrete Gaussian
- $$\mathbf{G} = \begin{bmatrix} 1 & 2 & \dots & 2^{k-1} & & & & 0 \\ & & & & \ddots & & & \\ & & 0 & & & 1 & 2 & \dots & 2^{k-1} \end{bmatrix}$$

From Hash Functions to a Signature Scheme

Implementation issues:

- $q = 2^k$ more suitable for practice
- entries of $\mathbf{R}$ are sampled from a discrete Gaussian
- $$\mathbf{G} = \begin{bmatrix} 1 & 2 & \dots & 2^{k-1} & & & & & 0 & & \\ & & & & & \ddots & & & & & \\ & & 0 & & & & & 1 & 2 & \dots & 2^{k-1} \end{bmatrix}$$

From Hash Functions to a Signature Scheme

How to compute signature $f^{-1}(\mathbf{u}),\ \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \in \mathbb{Z}_q^n$:

- Sample $\mathbf{x} \in \mathbb{Z}^{nk}$ according to the discrete Gaussian distribution s.th. $\mathbf{G} \cdot \mathbf{x} = \mathbf{u} \mod q$
- Then signature $\sigma = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{x}$ is a preimage of $\mathbf{u}$
- Proof:

$$\mathbf{A} \cdot \sigma = \left[\ \bar{\mathbf{A}} \quad \middle| \quad \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}\ \right] \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{x} =$$

$$\bar{\mathbf{A}}\mathbf{R} \cdot \mathbf{x} + (\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}) \cdot \mathbf{x} = \mathbf{G} \cdot \mathbf{x} = \mathbf{u}$$

From Hash Functions to a Signature Scheme

How to compute signature $f^{-1}(\mathbf{u}),\ \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \in \mathbb{Z}_q^n$:

- Sample $\mathbf{x} \in \mathbb{Z}^{nk}$ according to the discrete Gaussian distribution s.th. $\mathbf{G} \cdot \mathbf{x} = \mathbf{u} \mod q$
- Then signature $\sigma = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{x}$ is a preimage of $\mathbf{u}$
- Proof:

$$\mathbf{A} \cdot \sigma = [\ \bar{\mathbf{A}} \ \mid \ \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}\ ] \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{x} =$$

$$\bar{\mathbf{A}}\mathbf{R} \cdot \mathbf{x} + (\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}) \cdot \mathbf{x} = \mathbf{G} \cdot \mathbf{x} = \mathbf{u}$$

From Hash Functions to a Signature Scheme

How to compute signature $f^{-1}(\mathbf{u}),\ \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \in \mathbb{Z}_q^n$:

- Sample $\mathbf{x} \in \mathbb{Z}^{nk}$ according to the discrete Gaussian distribution s.th. $\mathbf{G} \cdot \mathbf{x} = \mathbf{u} \mod q$
- Then signature $\sigma = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{x}$ is a preimage of $\mathbf{u}$
- Proof:

$$\mathbf{A} \cdot \sigma = \left[\ \bar{\mathbf{A}} \ \mid\ \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}\ \right] \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{x} =$$

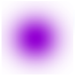
$$\bar{\mathbf{A}}\mathbf{R} \cdot \mathbf{x} + (\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}) \cdot \mathbf{x} = \mathbf{G} \cdot \mathbf{x} = \mathbf{u}$$

From Hash Hunctions to a Signature Scheme

Problem:

- Distribution of σ is skewed
- Leaks information about the trapdoor

- Need for spherically distributed signatures



Signature Scheme

Solution: Add perturbations $\mathbf{p}$ to correct distribution of signature

- Sample perturbations $\mathbf{p}$ with covariance matrix

$$\mathbf{C} = s^2\mathbf{I} - r^2 \begin{bmatrix} \mathbf{R}\mathbf{R}^\top & \mathbf{R} \\ \mathbf{R}^\top & \mathbf{I} \end{bmatrix} \text{ and perturbation matrix } \sqrt{\mathbf{C}}$$

- Compute perturbed syndrome $\mathbf{v} = H(m) - \mathbf{A}\mathbf{p} = u - \mathbf{A}\mathbf{p}$
- Sample $\mathbf{x}$ such that $\mathbf{G}\mathbf{x} = \mathbf{v}$
- Signatures: $\sigma = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{x} + \mathbf{p}$
- Distribution of signatures independent from secret key

Contributions

Implementation and Improvements:

- **Construction of the ring variant for more efficiency and practicality**
- Space improvement of perturbation matrix used to sample preimages
- Runtime improvement of Keygen and Signing due to improved perturbation matrix (sparse) and ring variant
- Implementation of the signature scheme (ring and matrix variant)

Contributions

Implementation and Improvements:

- **Construction of the ring variant for more efficiency and practicality**
- Space improvement of perturbation matrix used to sample preimages
- Runtime improvement of Keygen and Signing due to improved perturbation matrix (sparse) and ring variant
- Implementation of the signature scheme (ring and matrix variant)

Contributions

Implementation and Improvements:

- **Construction of the ring variant for more efficiency and practicality**
- Space improvement of perturbation matrix used to sample preimages
- Runtime improvement of Keygen and Signing due to improved perturbation matrix (sparse) and ring variant
- Implementation of the signature scheme (ring and matrix variant)

Contributions

Implementation and Improvements:

- **Construction of the ring variant for more efficiency and practicality**
- Space improvement of perturbation matrix used to sample preimages
- Runtime improvement of Keygen and Signing due to improved perturbation matrix (sparse) and ring variant
- Implementation of the signature scheme (ring and matrix variant)

Contributions

Ring variant:

- Consider the Ring $R_q = \mathbb{Z}_q[X]/x^n + 1$ for $n = 2^d$ and $q = 2^k$
- Choose a polynomial **a** uniformly at random from R_q
- Draw k Ring-LWE-samples $\mathbf{ar}_i + \mathbf{e}_i$
- Furthermore, consider the primitive vector of polynomials $\mathbf{g}^\top = [\mathbf{1}, \ldots, \mathbf{2^{k-1}}]$
- The public key is

$$\mathbf{A} = [\mathbf{1}, \mathbf{a}, \mathbf{g_1} - (\mathbf{ar}_1 + \mathbf{e}_1), \ldots, \mathbf{g_k} - (\mathbf{ar}_k + \mathbf{e}_k)]$$

Contributions

$$\mathbf{A} = [\mathbf{1}, \mathbf{a}, \mathbf{g_1} - (\mathbf{a}\mathbf{r}_1 + \mathbf{e}_1), \ldots, \mathbf{g_k} - (\mathbf{a}\mathbf{r}_k + \mathbf{e}_k)]$$

- A primitive matrix of polynomials **G** is explicitly not required
- $[\mathbf{a}, \mathbf{a}\mathbf{r}_1 + \mathbf{e}_1, \ldots, \mathbf{a}\mathbf{r}_k + \mathbf{e}_k]$ is pseudorandom
- Sampling preimages slightly differs from the matrix variant

Contributions

How to sample $\mathbf{x} \in R_q^{k-1}$ such that $\mathbf{g}^\top \mathbf{x} = \sum_{i=0}^{k-1} \mathbf{2}^i \mathbf{x}_i = \mathbf{u} \in R_q$

- Consider matrix expansion of $\mathbf{g}^\top$:

$$\tilde{\mathbf{G}} = [\mathbf{I}_n | 2\mathbf{I}_n | \ldots | 2^{k-1}\mathbf{I}_n]$$

- There exists permutation matrix $\mathbf{P}$ s.th.

$$\tilde{\mathbf{G}} = \mathbf{G} \cdot \mathbf{P} = \begin{bmatrix} 1 & 2 & \ldots & 2^{k-1} & & & & 0 & & \\ & & & & \ddots & & & & & \\ & & 0 & & & & 1 & 2 & \ldots & 2^{k-1} \end{bmatrix} \cdot \mathbf{P}$$

- $\mathbf{G}$ from matrix variant

Contributions

How to sample $\mathbf{x} \in R_q^{k-1}$ such that $\mathbf{g}^\top \mathbf{x} = \mathbf{u} \in R_q$

- We have

$$\tilde{\mathbf{G}} \cdot \begin{bmatrix} \mathbf{x}_1 \\ \ldots \\ \mathbf{x}_{k-1} \end{bmatrix} = \mathbf{u}$$

- Thus, sample $\mathbf{x}$ s.th. $\mathbf{G} \cdot \mathbf{x} = \mathbf{u}$
- $\tilde{\mathbf{x}} = \mathbf{P}^\top \cdot \mathbf{x}$ is a preimage for $\tilde{\mathbf{G}}$ since

$$\tilde{\mathbf{G}}\tilde{\mathbf{x}} = \mathbf{G} \cdot \mathbf{P}\mathbf{P}^\top \cdot \mathbf{x} = \mathbf{G}\mathbf{x} = \mathbf{u}$$

- If $\mathbf{x}$ spherically distributed, then so $\tilde{\mathbf{x}}$.

Contributions

How to sign a message $\mathbf{m}$:

- Sample perturbation polynomials $\mathbf{p} = [\mathbf{p}_1, \ldots, \mathbf{p}_{k+2}]$
- Compute perturbed syndrome $\mathbf{v} = \mathbf{H}(\mathbf{m}) - \mathbf{A} \cdot \mathbf{p}$
- Sample $\mathbf{x} \in R^k$ s.th. $\mathbf{g}^\top \mathbf{x} = \mathbf{v}$
- Signature is

$$\sigma = \mathbf{p} + [\mathbf{ex}, \mathbf{rx}, \mathbf{r_1 x_1}, \ldots, \mathbf{r_k x_k}]$$

- Signature is spherically distributed

Running times for ring (polynomials) and matrix version

Running times [ms]										
		Keygen			Signing			Verification		
n	k	Ring	Mat	M/R	Ring	Mat	M/R	Ring	Mat	M/R
128	24	277	984	3.6	5	9	1.8	0.6	1.4	2.3
128	27	317	1,108	3.5	6	11	1.8	0.7	1.7	2.4
256	24	1,070	5,148	4.3	12	30	2.5	1.5	5	3.3
256	27	1,144	5,728	4.1	14	36	2.5	1.7	6	3.5
512	24	4,562	28,449	5.0	27	103	3.8	3	18	6
512	27	5,354	30,458	5.1	31	125	4.0	4	21	5.3
512	29	5,732	34,607	5.4	35	136	3.8	5	22	4.4
1024	27	28,074	172,570	6.0	74	478	6.4	10	97	9.7
1024	29	30,881	198,620	6.3	81	518	6.4	11	102	9.3
Improvement factor		30-190 ↑	10 -40 ↑	-	2-6 ↑	1.4 - 2 ↑	-	-	-	-

Experimental results

Sizes for ring (polynomials) and matrix version

Sizes [kB]											
		Public Key			Secret Key			Pert. Matrix	Signature		
n	k	Ring	Mat	M/R	Ring	Mat	M/R	R and M	Ring	Mat	M/R
128	24	9.4	1200	128	4.4	528	163	257	5.8	5.3	0.9
128	27	11.8	1512	128	5.0	594	163	257	6.5	5.9	0.9
256	24	18.8	4800	256	9.8	2304	236	1026	12.5	11.4	0.9
256	27	23.6	6048	256	11.0	2592	236	1026	14.1	12.8	0.9
512	24	37.5	19,200	512	21.3	9984	469	4100	26.8	24.5	0.9
512	27	47.3	24,192	512	23.9	11232	470	4100	30.1	27.4	0.9
512	29	54.4	27,840	512	25.7	12064	470	4100	32.2	29.4	0.9
1024	27	94.5	96,768	1024	51.7	48384	936	16392	63.8	58.5	0.9
1024	29	108.8	111,360	1024	55.5	51968	936	16392	68.4	62.7	0.9
Improvement factor		-	-	-	-			170 - 260	-	-	-

Thanks for your attention!