CRYSTALS

(Cryptographic Suite for Algebraic Lattices)

CCA KEM: Kyber
Digital Signature: Dilithium

Roberto Avanzi – ARM
Joppe Bos – NXP
Leo Ducas – CWI
Eike Kiltz – RUB
Tancrede Lepoint – SRI

Vadim Lyubashevsky – IBM Research
John Schanck – U. Waterloo
Peter Schwabe – U. Radboud
Gregor Seiler – IBM Research
Damien Stehle – ENS Lyon

www.pq-crystals.org
Lattice Cryptography
Easy Problem

Given \((A, z)\), find \(y\)

Easy! Just invert \(A\) and multiply by \(z\)
Hard Problem

Given \((A, z)\), find \((y, e)\)

Small coefficients to enforce uniqueness

Seems hard (would have many positive non-cryptographic applications if it were easy).
Given \((A, z)\), find \((y, e)\)

Seems hard.
Even when \(A\) is over \(\mathbb{Z}_p[X]/(f(X))\) for certain \(f(X)\).
Why is this “Lattice” Crypto?

All solutions \( (y_e) \) to \( Ay + e = z \mod p \) form a “shifted” lattice.

We want to find the point closest to the origin (BDD Problem).
Brief History

• Lattices studied algorithmically at least since 1982 (LLL)

• Algebraic lattices since at least 1996 (NTRU)

• Lattices over $\mathbb{Z}_p[X]/(X^n+1)$ since at least 2008 (SWIFFT)

• Last 10 years – one of the hottest area in cryptography. Lots of attention and some interesting algorithms discovered

  ➢ But ... 0 attacks against lattice crypto based on (Module-) SIS / LWE

  ➢ Parameters were increased (around 50%) due to conservative considerations of “sieving” attacks requiring exponential space
CRYSTALS Math
Operations

Only two main operations needed (and both are very fast):

1. Evaluations of SHAKE256 (can use another XOF too)
2. Operations in the polynomial ring $R = \mathbb{Z}_p[X]/(X^{256}+1)$
   
   - prime $p = 2^{13} - 2^9 + 1$ (for Kyber)
   - prime $p = 2^{23} - 2^{13} + 1$ (for Dilithium)

Very easy to adjust security because the same hardware/software can be reused
Ring Choice Rationale

• 256-dimensional rings are “just right”
  ➢ Large enough to efficiently encrypt 256-bit keys
  ➢ Allow for a large enough challenge space for signatures
  ➢ Allow for enough “granularity” to get the security we want

• $\mathbb{Z}_p[X]/(X^n+1)$, for a prime $p$, has been the most widely-used ring in the literature
  ➢ Very easy to use and the most efficient one
  ➢ Has a lot of properties that are useful in more advanced constructions
Basic Computational Domain:

Polynomial ring $\mathbb{Z}_p[x]/(x^{256}+1)$

Operations used in the schemes:

- Small coefficients
Modular Security

768-dim \times \quad + \quad = \quad
to increase the security margin

1024-dim \times \quad + \quad = \quad

Just do more of the same operation
CRYSTALS-Dilithium

Leo Ducas – CWI
Eike Kiltz – RUB
Tancrede Lepoint – SRI

Vadim Lyubashevsky – IBM Research
Peter Schwabe – U. Radboud
Gregor Seiler – IBM Research
Damien Stehle – ENS Lyon
Digital Signatures Overview

- [HHPSW ‘03] Use NTRU trapdoor for Signatures
- [GPV ‘08] Made it Secure via Gaussian Sampling
- [Lyu ‘09] “Fiat-Shamir with Aborts” Digital Signature
- [Lyu ‘12] Gaussian Rejection Sampling SIS + LWE Based
- [GLP ‘12] [BG ‘14] Signature Compression
- FALCON
  - Based on NTRU Uses Discrete Gaussian Sampling
- BLISS [DDLL ‘13] Bimodal Gaussian Sampling
- Dilithium
  - Public Key + Signature Compression
  - Based on (Module-) LWE / SIS Uses Uniform Sampling
  - Additionally useful for ZK-Proofs

Additionally useful for IBE

Signature Size
Signatures with Uniform Sampling

[Lyu ‘09] \rightarrow \quad \cdots \quad \rightarrow \quad [BG ‘14]

Public key
\[ A := \text{XOF(seed)} \quad \text{uniform \ mod \ } p \]

Secret key
\[ s_1, s_2 \text{ with coefficients in \{-5,...,5\}} \]

Public key
\[ t := As_1 + s_2 \]
Signatures with Uniform Sampling

\[ \text{Sign}(\mu) \]

- \[ y \leftarrow \text{Coefficients in } [-\gamma, \gamma] \]
- \[ c := H(\text{high}(Ay), \mu) \]
- \[ z := y + cs_1 \]
- If \(|z| > \gamma - \beta\) or \(|\text{low}(Ay - cs_2)| > \gamma - \beta\)
  - restart

Signature = \((z, c)\)

\[ \text{Verify}(z, c, \mu) \]

- Check that \(|z| \leq \gamma - \beta\) and \(c = H(\text{high}(Az - ct), \mu)\)

\[ \text{As}_1 + s_2 = t \]
Signatures with Uniform Sampling

[Lyu ‘09] → ... → [BG ‘14]

\[ A \mathbf{s}_1 + \mathbf{s}_2 = \mathbf{t} \]

**Sign(μ)**

\[ \mathbf{y} \leftarrow \text{Coefficients in } [-\gamma, \gamma] \]
\[ c := H(\text{high}(\mathbf{A}) - c \mathbf{s}_2), \mu) \]
\[ \mathbf{z} := \mathbf{y} + c \mathbf{s}_1 \]

If \(|\mathbf{z}| > \gamma - \beta\) or \(|\text{low}(\mathbf{A} - c \mathbf{s}_2)| > \gamma - \beta\)

restart

Signature = (\(\mathbf{z}, c\))

**Verify(\(\mathbf{z}, c, \mu\))**

Check that \(|\mathbf{z}| \leq \gamma - \beta\)

and

\[ c = H(\text{high}(\mathbf{A} - c \mathbf{t}), \mu) \]

Because \(|\text{low}(\mathbf{A} - c \mathbf{s}_2)| \leq \gamma - \beta, \quad \text{high}(\mathbf{A} - c \mathbf{s}_2) = \text{high}(\mathbf{A})\]
Chopping off Low-Order PK bits

\[ As_1 + s_2 = t_0 + bt_1 \]

**Sign(\(\mu\))**

\[ y \leftarrow \text{Coefficients in } [-\gamma, \gamma] \]
\[ c := H(\text{high}(Ay), \mu) \]
\[ z := y + cs_1 \]

If \(|z| > \gamma - \beta\) or \(|\text{low}(Ay - cs_2)| > \gamma - \beta\)

restart

Signature = (\(z, c\))

**Verify(z, c, \(\mu\))**

Check that \(|z| \leq \gamma - \beta\)

and

\[ c = H(\text{high}(Az - cbt_1), \mu) \]

\[ Ay - cs_2 + ct_0 \]

Want \(\text{high}(Ay - cs_2) = \text{high}(Ay - cs_2 + ct_0)\)
The Carry Hint Vector

Want $\text{high}(A_y - cs_2) = \text{high}(A_y - cs_2 + ct_0)$

The signer knows $A_y - cs_2 + ct_0$ and $ct_0$

The verifier knows $A_y - cs_2 + ct_0$

Each 23-bit coefficient

- $A_y - cs_2 + ct_0$
- $-ct_0$
- $A_y - cs_2$

High Bits Low Bits High Bits

Carry bit
Dilithium
(high-level overview)

\[ As_1 + s_2 = t_0 + bt_1 \]

Sign(μ)

\[ y \leftarrow \text{Coefficients in } [-\gamma, \gamma] \]
\[ c := H(\text{high}(Ay), \mu) \]
\[ z := y + cs_1 \]
If \( |z| > \gamma - \beta \) or \( |\text{low}(Ay - cs_2)| > \gamma - \beta \)
   restart

Create carry bit hint vector \( h \)

Signature = (z, h, c)

Hint \( h \)
• adds 100 – 200 bytes to the signature
• Saves ≈ 2KB in the public key

Verify(z, c, μ)

Check that \( |z| \leq \gamma - \beta \)
and
\[ c = H(\text{high}(h \text{ “+” Az - cbt}_1), \mu) \]
\[ \text{high}(Ay - cs_2) \]
## Parameters for CRYSTALS-Dilithium

(> 128-bit quantum security)

<table>
<thead>
<tr>
<th></th>
<th>5 x 4 matrices</th>
<th>6 x 5 matrices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Key</td>
<td>≈ 1.5 KB</td>
<td>≈ 1.8 KB</td>
</tr>
<tr>
<td>Signature</td>
<td>≈ 2.7 KB</td>
<td>≈ 3.4 KB</td>
</tr>
</tbody>
</table>

Public key generation / verification: > 10,000 per second
Signing: > 3,000 per second
Security Reductions
**Signature Scheme Security**
*(Proof framework for Fiat-Shamir Schemes in the ROM)*

**Math Problem**
1. $A_3$ gets math problem
2. $A_3$ solves math problem

**Hybrid 1**
1. $A_2$ gets the public key and access to hash $H$
2. $A_2$ forges a signature

**Real Signature Scheme**
1. $A_1$ gets the public key and access to hash $H$
2. $A_1$ asks signature queries
3. $A_1$ forges a signature

- Reduction in the QROM [Unr ‘17]
- Tight reduction in the QROM if the signing is deterministic [KLS ‘18]
Dilithium Security

Non-tight in the ROM

Input: random $A, t$
Output: short $s_1, s_2$ and $c$ such that $As_1 + s_2 - tc = 0$

(Module)-SIS + (Module)-LWE

Non-tight reduction in the ROM using rewinding

Tight in the QROM

Input: random $A, t$, and an XOF $H$
Output: short $s_1, s_2, c$ and $\mu$ such that $H(As_1 + s_2 - tc, \mu) = c$

Hybrid 1 (Self-Target SIS)
The Same as Schnorr Signatures

**Non-tight in the ROM**
- **Input:** random $A$, $t$
- **Output:** short $s_1$, $s_2$ and $c$ such that $As_1 + s_2 - tc = 0$

**(Module)-SIS + (Module)-LWE**

**Tight in the QROM**
- **Input:** random $A$, $t$, and an XOF $H$
- **Output:** short $s_1$, $s_2$, $c$ and $\mu$ such that $H(As_1 + s_2 - tc, \mu) = c$

**Hybrid 1 (Self-Target SIS)**

**Input:** random $g, h$
- **Output:** $x$, $c$ such that $g^x h^c = 1$

**Discrete Log**

**“Self-Target Discrete Log”**
Is the “Self-Target” Assumption Worrisome in the QROM?

• We believe not

• No example where using “rewinding” in the proof left the scheme vulnerable to a quantum attacker

• Analogous to computationally-binding classical commitments not having a proof in the QROM (and there is no NIST competition for post-quantum commitments)
Base on (Module-)LWE in the QROM?

**Dilithium**

<table>
<thead>
<tr>
<th></th>
<th>recommended</th>
<th>high</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Key</td>
<td>≈ 1.5 KB</td>
<td>≈ 1.8 KB</td>
</tr>
<tr>
<td>Signature</td>
<td>≈ 2.7 KB</td>
<td>≈ 3.4 KB</td>
</tr>
</tbody>
</table>

“Katz-Wang” Tight Dilithium [AFLT ‘12, ABB+ ‘15, Unr ‘17, KLS ‘18]

<table>
<thead>
<tr>
<th></th>
<th>recommended</th>
<th>high</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Key</td>
<td>5X larger</td>
<td>≈ 7.7 KB</td>
</tr>
<tr>
<td>Signature</td>
<td>2X larger</td>
<td>≈ 5.7 KB</td>
</tr>
</tbody>
</table>

Also significantly (> 10X) slower
Basis for Our Parameter Settings

LWE parameters (i.e. secret key recovery) used the recently en vogue sieving analysis

SIS parameters (i.e. message forgery) – the same analysis + improved (potential) algorithm for \( l_\infty \)-SIS
Possible Trade-Offs (open to community suggestions)

• Smaller secret key coefficients e.g. \{-5,...,5\} \rightarrow \{-1,0,1\}
  • Signatures will be smaller
  • Makes combinatorial hybrid attacks more likely and gets further away from WC-AC parameters

• Module-LWE \rightarrow Module-LWR
  • (Maybe) a slight reduction in the key size
  • Probably nothing goes wrong with security if sk coefficients are large enough

• Increase the rejection probability
  • Slower, but smaller signatures