CRYSiALs: (Cryptographic Suite for Algebraic Lattices) Dilithium

Leo Ducas  Eike Kiltz  Tancrede Lepoint  Vadim Lyubashevsky
Peter Schwabe  Gregor Seiler  Damien Stehle
Dilithium

Lattice-based digital signature

Based on Generalized (a.k.a Module)-LWE / SIS problems

For all security levels, only need two main operations:
1. SHAKE (or any other XOF)
2. Operations in the polynomial ring
   \[ R = \mathbb{Z}_p[X]/(X^{256}+1) \text{ for prime } p = 2^{23} - 2^{13} + 1 \]
Dilithium Operations

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

small coefficients
Modular Security

to increase the security margin, do more of the same operation
Dilithium Features

- Very simple to implement – all sampling is uniform

- It’s fast (for all operations) and has the 2nd-smallest pk+sig size (after FALCON)

- Uses NTT for multiplication – very fast and can be done in place to reduce stack size

- Lattices over $\mathbb{Z}_p[X]/(X^n+1)$ used in concrete schemes since SWIFFT [LMPR ’08]. Algebraic lattices since NTRU [HPS ’96].
  - The algorithmic framework for cryptanalysis is stable since [S ’87] and [AKS ’01]. These techniques are being “squeezed out” right now.
  - Some parameter increase due to conservative considerations of “sieving” attacks requiring exponential space
# Parameters and Runtime

<table>
<thead>
<tr>
<th>Quantum Security:</th>
<th>90</th>
<th>128</th>
<th>160</th>
</tr>
</thead>
<tbody>
<tr>
<td>pk size (bytes)</td>
<td>1184</td>
<td>1472</td>
<td>1760</td>
</tr>
<tr>
<td>sig size (bytes)</td>
<td>2044</td>
<td>2701</td>
<td>3366</td>
</tr>
<tr>
<td>key gen. cycles</td>
<td>110K</td>
<td>156K</td>
<td>221K</td>
</tr>
<tr>
<td>verify cycles</td>
<td>110K</td>
<td>155K</td>
<td>220K</td>
</tr>
<tr>
<td>sign cycles (median)</td>
<td>315K</td>
<td>440K</td>
<td>465K</td>
</tr>
<tr>
<td>sign cycles with 64B sk</td>
<td>345K</td>
<td>475K</td>
<td>496K</td>
</tr>
</tbody>
</table>

* on an Intel Core-i7 6600U (Skylake) CPU using SHAKE as the XOF

**Changes from round 1 submission:**

- No changes in the design or parameter settings
- Included randomized signing mode in addition to deterministic
- Optimizations of the code (and fixed 1 implementation bug in Dec. 2017)
Dilithium

=  
LWE / SIS - Fiat-Shamir [L ‘09] + [L ‘12]  
+  
Signature Size Reduction [BG ‘14]  
+  
Public Key Reduction [DKL+ ‘18]
Dilithium Algorithms

**KeyGen()**

\[ A \leftarrow R^{5 \times 4};
    s_1 \leftarrow [-5, 5]^4,
    s_2 \leftarrow [-5, 5]^5
\]

\[ As_1 + s_2 = t = \text{low}(t) + \text{high}(t)\]

**SK:** \((s_1, s_2)\), **PK:** \((A \leftarrow R^{5 \times 4}, \text{high}(t))\)

---

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

\[ y \leftarrow [-\gamma, \gamma]^4\]

\[ c := H(\text{high}(Ay), \mu)\]

\[ z := y + cs_1\]

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

Create a small carry bit hint vector \(h\)

Signature = \((z, c, h)\)

---

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

Use \(h\) and \(Az - c \cdot \text{high}(t)\) to reconstruct \(\text{high}(Az - ct)\)

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

- Makes the distribution of \(z\) independent of \(s_i\)
- Carry bits caused by ignoring \(c \cdot \text{low}(t)\)

\[ = \text{high}(Ay)\]
Security Proof Reduction in the QROM

Tight reduction from:
1. LWE
2. ST-SIS: given random $A, t$, find $\mu$, short $c \neq 0$, $z_i$ satisfying $H(Az_1 + z_2 - ct, \mu) = c$

In the ROM, ST-SIS = SIS: (with the usual Schnorr-type security loss)

given random $A, t$, find short $c \neq 0$, $z_i$ satisfying $Az_1 + z_2 - ct = 0$
Dilithium Security

1. In the QROM, tightly based on LWE and STSIS [Unr ’17, KLS ’18]
   • For a ring R with a bigger p, ST-SIS is vacuously hard, so the scheme is based on just LWE in the QROM. Dilithium-Q [KLS ’18]

2. In the ROM, based on LWE and SIS [L ’09, L ’12]

3. In the QROM, based on the special-sound and collapsing properties of the underlying interactive protocol [DFMS ‘19].
   • Special soundness based on SIS [L ‘12, DKL+ ‘18]
   • It is conjectured in [DFMS ‘19] that the Dilithium protocol is collapsing

4. In the QROM, the collapsing property is (non-tightly) based on LWE. [LZ ‘19]
**Comparison to qTESLA**

same “style” as Dilithium (i.e. uses [L ‘09]+[L ’12]+[BG ‘14] as a starting point) but ... qTESLA had an incorrect security argument that bypassed the requirement for SIS to be hard

<table>
<thead>
<tr>
<th></th>
<th>qTESLA Round2 128-bit</th>
<th>qTESLA Round2 128-bit</th>
<th>qTESLA Round2 160-bit</th>
<th>qTESLA Round1 128-bit</th>
<th>Dilithium 128-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>pk size (bytes)</td>
<td>800</td>
<td>2336</td>
<td>38432</td>
<td>2976</td>
<td>1472</td>
</tr>
<tr>
<td>sig size (bytes)</td>
<td>2432</td>
<td>2144</td>
<td>5664</td>
<td>2720</td>
<td>2701</td>
</tr>
</tbody>
</table>

- completely broken [LS ‘19] (attack is faster than real signing)
- relies on a version of SIS with much less security than Dilithium
- security claims like Dilithium-Q [KLS ‘18] which is based on only LWE in the QROM
- proof of a stronger claim was wrong, but may have the same security as Dilithium instantiation of [BG ‘14] – no public key reduction
- parameters for 160-bit Dilithium-Q: pk: 9632 sig: 7098

Can be made somewhat fast using ideas from e.g. [B ‘19]. Guess: ≈ 10X slower than Dilithium
Dilithium and FALCON

If the goals are:
• Compactness
• Very easy implementation on all devices

Use Fiat-Shamir signatures with uniform sampling: **Dilithium**

If the goal is:
• Maximum Compactness

Use hash-and-sign signatures over NTRU lattices with Gaussian sampling: **FALCON**

<table>
<thead>
<tr>
<th></th>
<th>Dilithium (90-bit)</th>
<th>FALCON (100-bit)</th>
<th>Dilithium (128-bit)</th>
<th>Dilithium (160-bit)</th>
<th>FALCON (256-bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pk size (bytes)</td>
<td>1184</td>
<td>897</td>
<td>1472</td>
<td>1760</td>
<td>1793</td>
</tr>
<tr>
<td>sig size (bytes)</td>
<td>2044</td>
<td>652</td>
<td>2701</td>
<td>3366</td>
<td>1261</td>
</tr>
</tbody>
</table>
Dilithium and FALCON

**Dilithium**
- + Fast Verification
- + Fast Signing
- + Simple to implement everywhere – particularly important for low-power devices where generic signatures (e.g. SPHINCS) are too slow [KRSS ‘19]
- + Compact

**FALCON**
- + Fast Verification
- + Fast Signing (if Floating Point Unit is Present)
- + Very compact
- - Very delicate signing procedure – messing up the floating point precision can lead to leaking the secret key
- - Emulating the FPU using integer arithmetic can lead to significant slow-downs
- ? How easy is it to mask?

Both schemes serve a purpose

Techniques lead to practical ZK-based privacy primitives

Techniques lead to a practical IBE
CRYS\(\text{TALS:}\) (Cryptographic Suite for Algebraic Lattices) Dilithium

Thank You

www.pq-cryptals.org/dilithium
Bibliography

- [BG ‘14] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. CT-RSA ‘14
- [LMPR ‘08] Vadim Lyubashevsky, Daniele Micciancio, Alon Rosen, Chris Peikert. SWIFFT: A Modest proposal for FFT hashing. FSE ‘08