POST-QUANTUM CRYPTOGRAPHY EXPLAINED

for security professionals

DILITHIUM SIGNATURES (ML-DSA)

ALFRED MENEZES

INTRODUCTION

- Dilithium (ML-DSA) is a quantum-safe signature scheme standardized by NIST in August 2024.
- The NIST standard is FIPS 204.
- Dilithium is beginning to see deployment in the ongoing effort to secure communications against potential threats from quantum computers.
- Dilithium is considered a lattice-based cryptosystem because its security is based on the hardness of both LWE and SIS, which in turn are based on the hardness of the lattice problems SVP and approx-SVP.

FRAMEWORK

- Dilithium was designed using the "Fiat-Shamir" framework.
- Alice has a private signing key sk, and a public verification key vk.
- To sign a message M, Alice does the following:
    1. Select a secret y, and compute a commitment w to y.
    2. Generate a challenge $c = H(M, w)$.
    3. Compute the response $z = F(sk, c, y)$.
    4. Alice's signature on M is $\sigma = (c, z)$.
- To verify Alice's signature $\sigma = (c, z)$ on M, Bob does the following:
    1. Use Alice's verification key vk and signature σ to compute the commitment w'.
    2. Accept if and only if $c = H(M, w')$.

DILITHIUM SIMPLIFIED

- The Dilithium signature scheme is rather complicated, in part because of the introduction of errors during the key generation and signing procedure.
- Instead of presenting the full scheme, I'll present a highly-simplified version of Dilithium.
- *This version is both incomplete and insecure, but illustrates the main structure of Dilithium.*

KEY GENERATION

- Key generation. Alice does the following:

    1. Select a random $m \times n$ matrix A with entries in $\mathbb{Z}_q$.
    2. Select random length-n vectors s and e with entries in $[-B, B]$.
    3. Compute $b = As + e \pmod{q}$.
    4. Alice's public verification key is (A, b); her private signing key is (s, e).

- Note: Computing the private key from the public key is an instance of ss-LWE.

SIGNATURE GENERATION (first attempt)

- Signature generation. To sign a message M, Alice does:

    1. Select a length-n secret vector y whose entries are randomly selected from the interval $[-\gamma, \gamma]$.
    (Here, γ is significantly smaller than $q/2$.)
    2. Compute the commitment $w = Ay \pmod{q}$.
    3. Compute the challenge $c = H(M, w)$; c is relatively small.
    4. Compute the response $z = y + cs$.
    5. Alice's signature on M is $\sigma = (c, z)$.

- Note: Computing the secret y from (A, w) is an instance of SIS.

SIGNATURE VERIFICATION (first attempt)

- Bob, the verifier, has Alice's public key (A, b) and the signed message M, $\sigma = (c, z)$.
- Multiplying both sides of $z = y + cs$ by A and simplifying gives $Az - cb = w - ce$.
- Thus, Bob can compute $w - ce$, but not w itself.
- Note that ce has small components.
- To remedy this, the commitment is defined to be the "HighBits" of w. Alice ensures the "LowBits" of $w - ce$ are small so that w and $w - ce$ have the same HighBits.

SIGNATURE GENERATION (revised)

- Signature generation. To sign a message M, Alice does:

1. Select a length-n secret vector y whose entries are randomly selected from the interval $[-\gamma, \gamma]$.
2. Compute $w = Ay \pmod{q}$ and the commitment $w_1 = \text{HighBits(w)}$.
3. Compute the challenge $c = H(M, w_1)$.
4. If LowBits$(w - ce)$ are not small, then go to step 1.
5. Compute the response $z = y + cs$.
6. Alice's signature on M is $\sigma = (c, z)$.

SIGNATURE VERIFICATION (revised)

- Signature verification. To verify Alice's signature $\sigma = (c, z)$ on M, Bob does the following:

    1. Compute $w_1' = \text{HighBits}(Az - cb)$.
    2. Accept if and only if $c = H(M, w_1')$.

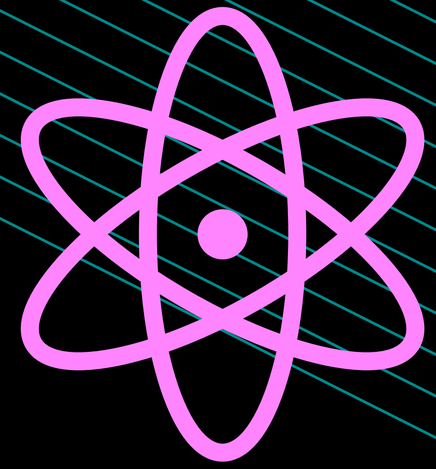
- Correctness. A properly signed message will be accepted because

$$\begin{aligned} w_1' &= \text{HighBits}(Az - cb) \\ &= \text{HighBits}(w - ce) \\ &= \text{HighBits}(w) \\ &= w_1 \,. \end{aligned}$$

FULL DILITHIUM

- As previously mentioned, the version of Dilithium presented is incomplete and insecure.
- Important fixes include:
  1. Replace elements in $\mathbb{Z}_q$ with polynomials of degree at most 255 with coefficients in $\mathbb{Z}_q$:
     $a_0 + a_1 x + a_2 x^2 + \cdots + a_{255} x^{255}$.
  2. Rejection sampling to circumvent leakage of information about the signing key.
  3. Public key compression.

CLOSING



- For a full description of Dilithium, see:
  - My online course “Kyber and Dilithium”: cryptography101.ca/kyber-dilithium
  - My lecture notes “A Gentle Introduction to Lattice-Based Cryptography”: cryptography101.ca/latticecrypto