Dilithium algorithm report

LECONTE Corentin, LEDIREACH Gurvan, GRASSER Pierre

2022

1 Dilithium

Dilithium is a digital signature scheme that is strongly secure under chosen message attacks based on the hardness of lattice problems over module lattices. The security notion means that an adversary having access to a signing oracle cannot produce a signature of a message whose signature he hasn't yet seen, nor produce a different signature of a message that he already saw signed

2 Design

The design of Dilithium is based on the "Fiat-Shamir with Aborts" technique proposed by Lyubashevsky which use rejetion samplong to make lattice-based Fiat-Shamir schemes compact and secure.

3 Key

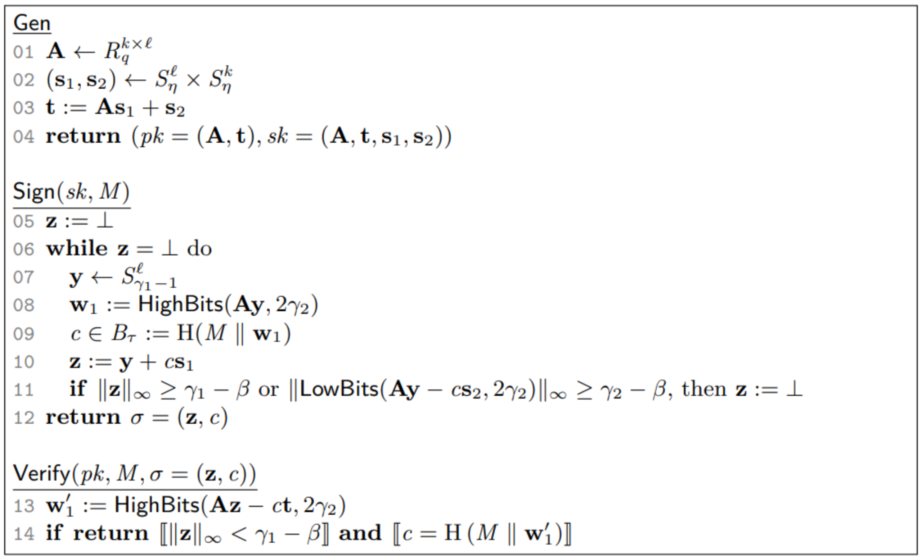
The team who created Dilithium say that Dilithium has the smallest public key + signature size of any lattice-based signature scheme that only uses uniform sampling, using a new technique made by Bai and Galbraith `https://eprint.iacr.org/2013/838`

In the second round of NIST project, a variant of the algorithm was proposed : "Dilithium-AES", who uses an AES-256 in "counter mode" instead of "SHAKE" to expend the matrix and the masking vectors, and to sample the secret polynomials.

4 Deterministic and Randomized Signatures

Dilithium gives an option of producing either deterministic (i.e the signature of a particular message is always the same) or randomized signatures. The only implementational difference vetween the two options is where the seed ρ' is either derived from the message and a key (in deterministic signing) or is chosen completely at random (for randomized signing).

5 Dilithium behaviour



```
Gen
$\mathbf{A} \leftarrow R_q^{k\times\ell}$
$(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^\ell \times S_\eta^k$
$\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
return $(pk = (\mathbf{A}, \mathbf{t}), sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1, \mathbf{s}_2))$

Sign$(sk, M)$
$\mathbf{z} := \perp$
while $\mathbf{z} = \perp$ do
  $\mathbf{y} \leftarrow S_{\gamma_1-1}^\ell$
  $\mathbf{w}_1 := \mathsf{HighBits}(\mathbf{A}\mathbf{y}, 2\gamma_2)$
  $c \in B_\tau := \mathrm{H}(M \parallel \mathbf{w}_1)$
  $\mathbf{z} := \mathbf{y} + c\mathbf{s}_1$
  if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\mathsf{LowBits}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2, 2\gamma_2)\|_\infty \geq \gamma_2 - \beta$, then $\mathbf{z} := \perp$
return $\sigma = (\mathbf{z}, c)$

Verify$(pk, M, \sigma = (\mathbf{z}, c))$
$\mathbf{w}_1' := \mathsf{HighBits}(\mathbf{A}\mathbf{z} - c\mathbf{t}, 2\gamma_2)$
if return $[\![\|\mathbf{z}\|_\infty < \gamma_1 - \beta]\!]$ and $[\![c = \mathrm{H}(M \parallel \mathbf{w}_1')]\!]$
```

5.1 Key generation

Variables used :

- A : Matrix of size k * l
- s1, s2 = secret keys
- t = second part of public key with $A * s_1 + s_2$
- Return : public key and secret key (pk = (A,t), sk = (a,t, s1, s2))

The key generation algorithm generate a matrix A of size k * l, where each of his entry is a polynomial on the ring $Rq = Zq[X]/(X^n + 1)$. After which the algorithm take a random sample of secret key vectors s1 and s2. Each coefficient of those vectors is a element of Rq, with minimum size n. Then the second part of the key is computed with $t = A * s_1 + s_2$

5.2 Signature procedure

The signature procedure generate a mask from a vector of polynomials y where the polynomials have a smaller coefficient than γ_1. The $\gamma_1$parameter is choosen strategically. It need to be big enough to prevent that the signature doesn't reveal the secret key.

It shouldn't be small enough to make sure that the signature isn't easily forgeable.

Then Ay is compute and we sets w_1 to be the "high-order' bits of the coefficient in this vector.

Each coefficient w of Ay should be writable in a canonical solution as $w = w_1 \times 2\gamma_2 + w_0$ where $|w_0| \leq \gamma_2$

Then the challenge c is created as the hash of the message and w_1

5.3 Verify procedure

The verifier first compute $w'1$ to be the high-order bits of $Az - ct$ and then accept if all coefficients of z are less than $\gamma_1 - \beta$ and if c is the hash of the message and $w'1$.

To check if verification works, we need to show that :

$$(1)\ HighBits(Ay, 2\gamma_2) = HighBits(Ay - cs_2, 2\gamma_2)$$

The reason for this is that valid signature will have :

$$||LowBits(Ay - cs_2, 2\gamma_2)||_\infty < \gamma_2 - \beta$$

And since we know that the coefficient of cs_2 are smaller than β we know that adding cs_2 is not enough to cause any carries by iuncreasing any low-order coefficient to have magnitude at least γ_2. Thus Eq. (1) is true and the signature verifies correctly.