Report On CRYSTALS-Dilithium

Shivam Agarwal CSE2020123

14 May 2023

1 Introduction

This Report goes over the basic workings of the CRYSTALS-Dilithium signature scheme and gives a overview of its security. I will consider the uncompressed public key version of the scheme for this report. Both the methods offer similar security, however the compressed key version is more efficient in terms of size.

CRSTALS-Dilithium is Lattice based Digital Signature Scheme which is one of the major contenders for winning the NIST PQC project. Major advantages of this Scheme are :

- It is simple to implement,
- Size of public key + signature is smaller as compared to other schemes
- It is very modular; meaning it is easy to vary the security of the scheme, the methods for which we will go over later.

2 Basic Approach

2.1 Key Gen

The Key Gen algorithm generates a k x l matrix A, where q = $2^{23} - 2^{13} + 1$ and n = 256. Then the algorithm samples the random secret vectors s_1 and s_2. Finally, the second part of the public key is computed as t = $As_1 + s_2$. All algebraic operations in this scheme are assumed to be over the polynomial ring R_q. To reduce the size of the public key, instead of storing the entire matrix for the key, we can store a random seed ρ and use an XOF(extendable output function) such as SHAKE-128 to generate the matrix. So the final public key is (ρ, t).

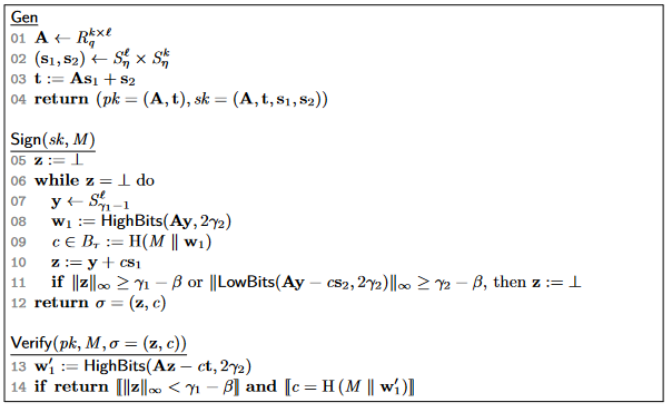

Figure 1: Template for the signature scheme without public key compression

2.2 Signing Procedure

Inside a loop, the signing algorithm first generates a masking vector of polynomials y with coefficients less than γ_1. The parameters γ_1 and γ_2 are set strategically – it is large enough that the eventual signature does not reveal the secret key (i.e. the signing algorithm is zero-knowledge), yet small enough so that the signature is not easily forged. There values are fixed depending on the size of the matrix A. The signer then computes Ay and sets w_1 to be the "high-order" bits of the coefficients in this vector. In particular, every coefficient w in Ay can be written in a canonical way as $w = w_1 \cdot 2\gamma_2 + w_0$ where $|w_0| \leq \gamma_2$; w_1 is then the vector comprising all the w_1's.

Then we compute c as the hash of the message and w_1. The potential signature is then computed as $z = y + cs_1$. Outputting z like this is insecure and can lead to attacks to figure out the secret vectors form z, so to avoid the dependency of z on the secret key, we use rejection sampling. The parameter β is set to be the maximum possible coefficient of cs_i. If any coefficient of z is larger than or equal to $\gamma_1 - \beta$, then we reject the computed z and restart the signing procedure by selecting a new y. Also, if any coefficient of the low-order bits of $Az - ct$ is greater than or equal to $\gamma_2 - \beta$, we restart. The first check is necessary for security, while the second is necessary for both security and correctness. The while loop in the signing procedure keeps being repeated until the preceding two conditions are satisfied. The parameters are set such that the number of repetitions is not too high. The Signature thus computed is denoted by σ, which is equal to (z, c).

2.3 Verification

The verifier has the following inputs : The public key (A, t) or (ρ, t), using which he can compute (A, t), the message M, and the Signature $\sigma = (z, c)$. The

verifier then first computes w_1' to be the high-order bits of $Az - ct$, and then accepts if all the coefficients of z are less than $\gamma_1 - \beta$ and if the c of the signature is equal to the hash of the message and w_1'.

Let us understand why verification works, in particular as to why HighBits($Az - ct, 2\gamma_2$) = HighBits($Ay, 2\gamma_2$). The first thing to notice is that $Az - ct = Ay - cs_2$. This is because $z = y + cs_1$ and $t = As_1 + s_2$.

So $Az - ct = A(y + cs_1) - c(As_1 + s_2)$

$= Ay + Acs_1 - Acs_1 - cs_2$

$= Ay - cs_2$

So all we really need to show is that HighBits($Ay, 2\gamma_2$) = HighBits($Ay - cs_2, 2\gamma_2$). The reason for this is that for a valid signature, β will be greater than the coefficients of cs_2 and $\gamma_2 > \beta$, so adding cs_2 is not enough to cause any carries as w' is computed as $w_1' \cdot 2\gamma_2 + w_0'$. Thus value of the term added has to be greater than γ_2 to cause a change in w_1'. So $w_1 = w_1'$, thus $c = H(M||w_1')$, hence the signature verifies correctly.

3 Modifications and Implementation Considerations

One major problem with the above approach is that the public key consists of a matrix A of size $k \cdot l$, which has a rather large representation. To remedy this, we use a XOF like SHAKE-128 to generate A from a seed ρ. The public key is therefore (ρ, t) and its size is dominated by t. The Scheme can also produce both deterministic and random signatures, based on whether the seed ρ', which is used to generate the secret vectors s_1 and s_2 is derived from the message and the key, leading to a deterministic signature scheme or chosen completely at random, leading to a random signature scheme.

The main algebraic operation performed in this scheme is a multiplication of a matrix A, whose elements are polynomials in $Z_q[X]/(X^{256}+1)$, by a vector y of such polynomials. So the ring has been chosen in such a way that the multiplication operation has a very efficient implementation via the Number Theoretic Transform (NTT), which is just a version of FFT that works over the finite field Z_q rather than over the complex numbers. To do this the number q is chosen in such a way that it is a prime and the group Z_q^* has an element of order $2n = 512$, or equivalently $q \equiv 1 \ (mod\ 512)$.

4 Number of Repetitions

The probability that $\|z\|_\infty < \gamma_1 - \beta$ can be computed by considering each coefficient separately. For each coefficient σ of cs_1, the corresponding coefficient of z will be between $-\gamma_1 + \beta + 1$ and $\gamma_1 - \beta - 1$ (inclusively) whenever the corresponding coefficient of y_i is between $-\gamma_1 + \beta + 1 - \sigma$ and $\gamma_1 - \beta - 1 - \sigma$. The size of this range is $2(\gamma_1 - \beta) - 1$, and the coefficients of y have $2\gamma_1 - 1$

possibilities. Thus, the probability that every coefficient of y is in the good range is :

$$\begin{aligned}&(2(\gamma_1-\beta)-1)/(2\gamma_1-1))^{256\cdot l}\\&\approx e^{-256\cdot\beta l/\gamma_1}\end{aligned}$$

If we (heuristically) assume that the low order bits are uniformly distributed modulo $2\gamma_2$, then the probability that all the coefficients of the LowBits case are in good range is :

$$\begin{aligned}&(2(\gamma_2-\beta)-1)/(2\gamma_2))^{256\cdot k}\\&\approx e^{-256\cdot\beta k/\gamma_2}\end{aligned}$$

So the Probability that all the coefficients are in the good range is given by :

$$\approx e^{-256\cdot\beta(l/\gamma_1+k/\gamma_2)}$$

Since the expected number of iterations or repetitions is independent of the secret keys s_1 and s_2, no information can be gained about them by an attack that counts the iterations.

5 Changing the Security Level of the Scheme

NIST Security Level	2	3	5
Parameters			
q [modulus]	8380417	8380417	8380417
d [dropped bits from $\mathbf{t}$]	13	13	13
τ [# of ±1's in c]	39	49	60
challenge entropy [$\log\binom{256}{\tau}+\tau$]	192	225	257
γ_1 [$\mathbf{y}$ coefficient range]	2^{17}	2^{19}	2^{19}
γ_2 [low-order rounding range]	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
(k,ℓ) [dimensions of $\mathbf{A}$]	$(4,4)$	$(6,5)$	$(8,7)$
η [secret key range]	2	4	2
β [$\tau\cdot\eta$]	78	196	120
ω [max. # of 1's in the hint $\mathbf{h}$]	80	55	75
Repetitions (from Eq. (5))	4.25	5.1	3.85

Figure 2: Parameters of Dilithium

The most straightforward way of raising/lowering the security of Dilithium is by changing the values of (k,ℓ) and then adapting the value of η (and then β, γ_1, γ_2 and so on) accordingly as shown in the Figure 2. Increasing (k,ℓ) by 1 each results in the public key increasing by ≈ 300 bytes and the signature by ≈ 700 bytes; and increases security by ≈ 30 bits.

A different way to increase the security would be to decrease the value of γ_1 and/or γ_2. This would make forging signatures (whose hardness is based on the underlying SIS problem) more difficult. Rather than increasing the size of

the public key/signature, the negative effect of lowering the γ_i is that signing would require more repetitions. This can be seen from how γ_1 and γ_2 affect the probability of no repetition. One could similarly increase the value of η in order to make the LWE problem harder at the expense of more repetitions. Because the increase in running time is rather dramatic (e.g. halving both γ_i would end up squaring the number of required repetitions, it is recommend to increase (k, ℓ) when needing to "substantially" increase security. Changing the γ_i should be reserved only for slight "tweaks" in the security levels.

6 Signature Scheme Security

Tthe security of the signature scheme can be proved, in the Random Oracle model (ROM), based on the hardness of two problems. The first is the standard LWE (over polynomial rings) problem which asks to distinguish $(A, t := As_1 + s_2)$ from (A, u), where u is uniformly random. The other problem is, what was called the SelfTargetMSIS problem.

In the Random Oracle Model(ROM) one can get a non-tight reduction using the forking lemma from the standard MSIS problem of finding a z' with small coefficients satisfying $Az' = 0$ to SelfTargetMSIS. In the Quantum Random Oracle model (QROM), where an adversary can query the hash function H in superposition, the security of Dilithium is based on the hardness of the MLWE and SelfTargetMSIS in QROM, even with a tight reduction when the scheme is deterministic, the forking lemma cannot be directly used to give a quantum reduction from MSIS to SelfTargetMSIS. Nevertheless, there are still good reasons to believe that the SelfTargetMSIS problem and therefore Dilithium remain secure in the QROM.

The concrete security of Dilithium is as follows:

If H is a quantum random oracle (i.e., a quantum-accessible perfect hash function), the advantage of an Adversary A breaking the SUF-CMA(Strong Unforgetability under Chosen Message Attacks) security of the Signature scheme is as follows:

$$\mathrm{Adv}^{SUF-CMA}_{Dilithium}(A) \leq$$
$$Adv^{MLWE}_{k.l,D}(B) + Adv^{SelfTargetMSIS}_{H,k.l+1,\zeta}(C) + Adv^{MSIS}_{k.l,\zeta'}(D) + 2^{-254}$$

for a distribution D over S_η.

Intuitively, the MLWE assumption is needed to protect against key-recovery, the SelfTargetMSIS is the assumption upon which new message forgery is based, and the MSIS assumption is needed for strong unforgeability. We will now sketch some parts of the security proof that are relevant to the concrete parameter setting.

7 Concrete Security

7.1 Lattice Reduction and Core-SVP Hardness

The best known algorithm for finding very short non-zero vectors in Euclidean lattices is the Block–Korkine–Zolotarev algorithm (BKZ). BKZ with a block-size b makes calls to an algorithm that solves the Shortest lattice Vector Problem (SVP) in dimension b. The security of our scheme relies on the necessity to run BKZ with a large block-size b and the fact that the cost of solving SVP is exponential in b. The best known classical SVP solver runs in time $\approx 2^{c_C \cdot b}$ with $c_C = \log_2 \sqrt{3/2} \approx 0.292$. The best known quantum SVP solver runs in time $\approx 2^{c_Q \cdot b}$ with $c_Q = \log_2 \sqrt{13/9} \approx 0.265$. One may hope to improve these run- times, but going below $\approx 2^{c_P \cdot b}$ with $c_P = \log_2 \sqrt{4/3} \approx 0.2075$ would require a theoretical breakthrough.

7.2 Solving MLWE

Any $MLWE_{l,k,D}$ instance for some distribution D can be viewed as an LWE instance of dimensions $256.l$ and $256.k$.

Given an LWE instance, there are two lattice-based attacks. The primal attack and the dual attack. Here, the primal attack consists in finding a short non-zero vector in the lattice $\Lambda = x \in \mathrm{Z}^d : Mx = 0 \ mod \ q$ where M is an md matrix where $d = 256 \cdot l + m + 1$ and $m \leq 256 \cdot k$. The value of m is varied as it is not optimal to use all the given equations in Lattice Attacks.

The dual attack consists in finding a short non-zero vector in the lattice $\Lambda' = (x, y) \in Z^m Z^d : M^T x + y = 0 \ mod \ q$, where M is an md matrix where $d = 256 \cdot l$ and $m \leq 256 \cdot k$.