Crystal Dilithium Algorithm For Post Quantum Cryptography:Experimentation and Usecase for eSign

Conference Paper · February 2022

DOI: 10.1109/ICEEICT53079.2022.9768654

3 authors, including:

Srikanth Sailada
Centre for Development of Advanced Computing
5 PUBLICATIONS **4** CITATIONS

SEE PROFILE

Crystal Dilithium Algorithm For Post Quantum Cryptography:Experimentation and Usecase for eSign

Srikanth Sailada
C-DAC
Pune, India
sailadas@cdac.in

Neeti Vohra
C-DAC
Pune, India
neetiv@cdac.in

N. Subramanian
C-DAC
Pune, India
subbu@cdac.in

2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT) | 978-1-6654-3647-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICEEICT53079.2022.9768654

***Abstract*— Cryptography has been useful in many applications such as transactions, computer or mobile passwords, internet connectivity, e-commerce transactions etc. Most of the applications use RSA algorithm for digital signature (dongle based or eSign) or digital certificate (issued by a certified authority). RSA strength is based on the fact that it is difficult to factorize a large integer. With the improvements in the quantum computers, classical cryptography is under the threat of quantum attacks. Quantum computers can factorize the large modulus with the help of Shor's algorithm in much shorter time. This paper provides literature survey of Crystals Dilithium Algorithm which is one of the post quantum digital signature schemes that has reached the final round of NIST competition for post quantum cryptography and its use case in online digital signing facility, eSign.**

Keywords*— *RSA algorithm, Classical Cryptography, Post Quantum Cryptography, Lattice cryptography, Learning with Errors (LWE), Crystal Dilithium, eSign

I. INTRODUCTION

Cryptography is a technique used to keep the information safe and secret by transforming it into cipher text which cannot be understood by unintended recipients. Encryption and Decryption are the main functions in cryptography for turning plain text to cipher text and vice versa. The encryption and decryption process always involves both an algorithm and a key. A key can be private or public which is just a piece of information that specifies how the algorithm is applied to the plain text in order to encrypt or decrypt it.

Digital signature is widely used to verify the authenticity and integrity of digital content, providing a method parallel to manual signature on a hard copy document. Digital signature provides the details of the source, identity and status and also helps in detecting the tampering occurred in electronic documents, transactions or digital messages. It uses public key cryptographic techniques such as RSA or ECC algorithm and is a legally acceptable form of signing. There are two widely adopted methods of digital signatures - Cryptographic Token (dongle) based and Online mechanism, eSign. Cryptographic tokens are issued by Certifying Authorities (CA) upon successful verification of the applicant and have a validity of 1-2 years. eSign leverages eKYC (Electronic Know Your Customer) service of government authorized KYC databases for authentication of the signer and uses Public Key Infrastructure (PKI) mechanisms for issuing the signature.

"A **Digital certificate** is used to share public keys to the be used for authentication. Digital certificate includes public key certified by a Certifying Authority (CA), identifying information about the entity and digital signature on public key signed by issuer of the certificate". Digital certificates are important to make secure connection between users and service providers. Reference [1] provides a detailed explanation of digital signatures and certificates.

RSA algorithm is widely used asymmetric key cryptography for key exchanging, digital signature or digital certificate generation. RSA's hardness depends on the factorization of a number that is obtained by multiplication of two large primes. Now a days, Elliptic Curve Cryptography has also been widely used because of its complexity. ECC depends on the complexity of the elliptical curves. The strength of ECC with 256 key size is equivalent to 3072 bits key size of RSA. These two algorithms are known to be safe against attack by a classical computer, but are expected to be broken by quantum computers. Quantum computers can easily solve Shor's algorithm which can factorize the secret key of RSA and can even break ECC using brute force. To avoid quantum attacks, research efforts towards Quantum Key Distribution (QKD) and Post Quantum Cryptography (PQC) are being widely pursued by researchers across the world. The primary aim of PQC is to develop a cryptographic system that assures safety against both quantum and classical attacks.

Towards development of standardised algorithms for Post Quantum Public Key Cryptography, National Institute of Standards and Technology (NIST) has initiated a standardization process to improve Digital Signature Standard (DSS). 69 submissions were made for combining both "Key exchange" and "Digital signature" algorithms. By the end of second round evaluation, nine Key exchange algorithms and six Digital signature algorithms have reached the final round of evaluation.

Post Quantum Digital signature algorithms that reached final round of NIST are given below along with the post quantum cryptography approach as given in [2].

1. Crystals Dilithium (Lattice-based cryptography)
2. FALCON (Lattice-based cryptography)
3. RAINBOW (Multivariate based cryptography)
4. GEMSS (Multivariate based cryptography)
5. Picnic (Symmetric Key quantum resistant)
6. SPHINCS+ (Hash based cryptography)

In this paper, Crystals Dilithium Scheme has been chosen for study and its implementation using OpenSSL fork based on liboqs has been experimented with. This digital signature

scheme is used to sign a text and also used to generate digital certificates. The outputs are verified with the details given in the submission. A brief study of key generation, signing and verification has been done and the techniques used in the process have been studied. As a usecase, its applicability in eSign mode of Digital Signing has been studied.

II. Related Work

RSA has been widely used algorithm in classical computers to provide security, integrity and authenticity. The strength of RSA algorithm is based on prime factorization of the modulus. Frequently used RSA algorithm modulus size is of 2048, 3072 and 4096 bits based on the application. Large size of modulus ensures the security from various attackers. There have been different attacks on RSA algorithm in the past. Few of them are explained in [3]. Author explains the math behind the strength of RSA and then provides the description of 20 types attacks that makes RSA vulnerable.

There has been lot of research going on in the development of Quantum computers. Quantum computers uses the properties of quantum physics to store data and perform computations. They use qubits instead of binary bits for computations. Quantum computers utilize the concept of superposition and entanglement to perform parallel computation. The effects of quantum computers in the field of cryptography is discussed in [4]. Author describes Quantum Key Distribution (QKD) which deals with secure communication where a secret key can be exchanged between two parties using light particles (photons). Quantum computers can factorize the large integer in polynomial time using Shor's algorithm. Thus quantum computers have become a threat for classical cryptography methods such as RSA, ECC etc.

Digital signature is used in various application domains such as citizen ID cards, inter/intra bank messaging systems, online trading etc. Digital signature offers privacy, authenticity, integrity and non-repudiation in digital transactions. Digital signature using symmetric and asymmetric encryption are given in [5].

Crystals Dilithium is one of the lattice based post quantum cryptography digital signature schemes that are submitted to NIST standardization competition. The final submission of crystal dilithium is given in [6] and [7]. Authors have given the information of the template on which algorithm is based on and also the final version of crystal dilithium algorithm. SHAKE algorithm is used for hashing and as a random number generator in Crystals Dilithium algorithm. Detailed explanation of SHA-3 family where SHAKE algorithm belongs to is presented in [8]. Detailed explanation will be given in further sections.

"The first Very High Speed Integrated Circuit Hardware Description Language (VHDL) implementation of the Crystals Dilithium signature scheme for Field Programmable Gate Arrays (FPGAs)" is presented in [9]. They have implemented most of the basic supporting algorithms used in Crystals Dilithium Algorithm from scratch using VHDL. This paper gives the comparisons of VHDL-based implementation of Crystals Dilithium with related High-Level Synthesis (HLS)-based implementations and also with software implementation written in C.

III. RSA Algorithm and eSign

A. RSA Algorithm

RSA is a public key cryptography algorithm. Its strength depends on the prime factorisation of a number which is also known as Modulus. The strength of the algorithm is proportional to the size of the modulus. Let modulus $N = pq$ where p and q are two large prime integers of same size (n/2) where n is size of N. Let e, d be two integers such that $ed = 1 mod \varphi(N)$ where $\varphi(N) = (p-1)(q-1)$. e is known as the encryption exponent and d the decryption exponent. The pair $< N, e >$ is called the public key. The pair $< N, d >$ is called the secret key or private key.

A message is an integer $M \in Z_N^*$. To encrypt a message, one computes $C = M^e mod N$. To decrypt the cipher text, the receiver computes $C^d mod N$ which will be message M. The strength of RSA-2048 is 112bits and RSA-3072 is 128bits.

B. eSign

eSign is an online Digital Signing method recognized by IT Act 2000 [10]. eSign leverages eKYC mechanism for authenticating the signer and uses PKI primitives for performing signing. eKYC can be performed either in online or offline mode. Online mode of authentication uses Aadhaar eKYC service and offline mode requires eSign CA to maintain a verified record of KYC in its database. There are provisions to integrate with various government recognized KYC databases for authentication purpose. The verified KYC data can be used at the time of signing for fetching the details of signer [11].

For signing purpose, a key pair (based on RSA or ECC algorithm) is generated on behalf of the signer, which is then used to sign the document hash and create the Certificate Signing Request (CSR) for the signer. The KYC data obtained from KYC service is used to form the subject in the CSR generated for signer. CSR is then signed by eSign CA to issue a Digital Signature Certificate (DSC). The response to the signer includes digitally signed document hash and certificate issued by eSign CA. The key pair is deleted once the signing has been completed. The entire process is performed in real-time.

Key difference in signing using eSign method with regard to traditional dongle method, is that a new key pair needs to be generated for each transaction. Typically, a Hardware Secure Module (HSM) is used for generating the key. HSMs are traditionally used for maintaining a key safely. However, eSign framework requires keys to be generated at high speed for serving real-time demand. Implementation of HSMs using Lattice Cryptography algorithms is presented in [12].

IV. Crystals Dilithium

Most of the schemes being evaluated in round 3 of NIST evaluation are based on Lattice Cryptography method. To establish understanding of the method, Crystals-Dilithium has been chosen as the scheme to be studied in detail.

The hardness of Crystals-Dilithium algorithm is based on module - learning with errors (LWE) and Short Integer Solution (SIS). "Fiat-Shamir with Aborts" protocol is used for non-interactive zero knowledge identification. Rejection sampling is used in this algorithm to make the scheme

compact and secure. Crystals-Dilithium has 6 variations - Dilithium2, Dilithium3, Dilithium5, Dilithum2_AES, Dilithium3_AES and Dilithium5_AES. Variations Dilithium2, Dilithium3, Dilithium5 are on the size of the polynomial matrix, secret key range and masking vector coefficient range. Dilithium_AES version uses AES-256 in counter mode instead of SHAKE-128 or 256 to expand the matrix and the masking vectors and to sample the secret polynomials.

Section IV-A provides simplified version of Crystals Dilithium algorithm [6] [7]. This version gives a basic idea behind the final submission. Section IV-B provides the final version which is submitted for the 3rd round of NIST submission. Section IV-C shows the variations present in Crystals Dilithium algorithm.

A. Template of Crystal Dilithium Algorithm

The key generation, signing and verification steps of simplified Crystals Dilithium algorithm are given in below sub sections.

1) Key Generation algorithm:

This algorithm generates a $k \times l$ matrix A, whose elements are polynomials in ring $R_q = Z_q[X]/(X^n + 1)$. Value of $q = 2^{23} - 2^{13} + 1$ and $n = 256$. Then algorithm samples s_1 and s_2 which are secret key vectors. The coefficients of these vectors are elements in R_q with a small coefficient size η.

Steps involved in key generation:

1. $A \leftarrow R_q^{k \times l}$
2. $(s_1, s_2) \leftarrow S_\eta^l \times S_\eta^k$
3. $t := As_1 + s_2$
4. return $(pk = (A, t), sk = (A, t, s_1, s_2))$

2) Signing algorithm:

In this algorithm first a masking vector is generated whose polynomials are coefficients less than γ_1. Next signer computes Ay and sets w_1 = Higher order bits of the coefficients in Ay. Each coefficient w in Ay can be written as $\omega = \omega_1 . 2\gamma_2 + \omega_0$ where $|\omega_0| \leq \gamma_2$; w_1 = Vector(ω_1's). The challenge vector c = Hash(input message + w_1). The vector "c" is a polynomial in R_q with exactly $\tau \pm 1$ and the rest are 0's. The signature is generated as $\boldsymbol{z = y + cs_1}$.

Rejection sampling is used instead of Gaussian sampling to avoid the dependency of z on the secret key. The parameter β is choosen such that it is the maximum possible coefficient of cs_i. The signing algorithm iteration is rejected and restarts if (1) any coefficient of $z > \gamma_1 - \beta$ (2) any coefficient of the low-order bits (refer section: V for LowBits function) of $\mathbf{Az - ct} > \gamma_2 - \beta$. The first condition is requisite for security check, while the second one is for both security check and correctness.

Steps for signing a message (M) given secret key (sk):

1. $z := \perp$
2. while $z = \perp$ do
   a) $y \leftarrow S_{\gamma-1}^l$
   b) $w_1 := HighBits(Ay, 2\gamma_2)$
   c) $c \in B_\tau := H(M||w_1)$
   d) $z := y + cs_1$
   e) $if\ (||z||_\infty \geq \gamma_1 - \beta)$ or $(||LowBits(Ay - cs_2, 2\gamma_2)||_\infty \geq \gamma_2 - \beta)$, then $z := \perp$
3. return $\sigma = (z, c)$

3) Verification:

In this step, the verifier computes w_1' to be higher-order bits (refer section: V for HighBits function) of $\boldsymbol{Az - ct}$, and if the coefficients of $z < \gamma_1 - \beta$ and if vector c = Hash(message + w_1'), verifier accepts.

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

Steps for verification using pk, M, $\sigma = (z, c)$:

1. $w_1' := HighBits(Az - ct, 2\gamma_2)$
2. return $[\![||z||_\infty < \gamma_1 - \beta]\!]$ and $[\![c = H(M||w_1')]\!]$

B. Final Version of Crystal Dilithium

Crystals Dilithium algorithm has been modified for the final round of submission by making few changes that reduces the public key length and reduces number of mathematical operations. The changes made in key generation algorithm, signing algorithm and verification process are given below.

1) Key Generation algorithm:

A seed value is used to generate matrix A, vectors s_1 and s_2. The extracted values are stored in NTT representation, since the multiplication is done using NTT approach. To extract matrix and vectors, SHAKE-256 algorithm is used. Instead of t, t_1 (HighBits) of t is used as public key. This reduces the size of public key.

Steps for key generation:

1. $\zeta \leftarrow \{0,1\}^{256}$
2. $(\rho, \upsilon, K) \in \{0,1\}^{256 \times 3} := H(\zeta)$
3. $(s_1, s_2) \in S_\eta^l \times S_\eta^k := H(\zeta)$
4. $A \in R_q^{k \times l} :=$ ExpandA(ρ)
5. $t := As_1 + s_2$
6. $(t_1, t_0) :=$ Power2Roundq(t,d)
7. $tr \in \{0,1\}^{384} := CRH(\rho||t_1)$
8. return $(pk = (\rho, t_1),\ sk = (\rho, K, tr, s_1, s_2, t_0))$

In the above steps, H is instantiated as SHAKE-256 throughput, Power2Round$_q$ is a supporting algorithm that will be discussed later, CRH is Collision Resistant Hash function. K is a key which is used in signing procedure.

2) Signing Algorithm:

During signing, matrix A is expanded using seed and stored as $\hat{A}$ which is NTT representation of A. Instead of using message directly, hash of message is used for further process. Dilithium provides an option of generating either deterministic signatures (where the signature of same message is always the the same) or randomized signatures (where different signatures for every implementation even if the message is same). The only difference in implementation for the two options occurs in choosing the seed value ρ'. For deterministic signing, the seed value ρ' can be calculated from the message & a key and for randomised signing, the seed value ρ' is chosen completely at random. There is a chance of compromising the security of the message in the deterministic method. So for applications where security is at most importance randomized signing is implemented.

Steps to sign a message using sk is given below:

1. $A \in R_q^{k\times l} := ExpandA(\rho)$
2. $\mu \in \{0,1\}^{384} \coloneqq CRH(tr||M)$
3. $k \coloneqq 0, (z,h) \coloneqq \perp$
4. $\rho' \in \{0,1\}^{384} := CRH(K||M)(or\ \rho' \leftarrow \{0,1\}^{384}\ for\ randomized\ signing)$
5. while (z,h) = ⊥ do
   a) $y \in S_{\gamma_1}^{l} \coloneqq ExpandMask(\rho', k)$
   b) $w := Ay$
   c) $w_1 \coloneqq HighBits_q(w, 2\gamma_2)$
   d) $\tilde{c} \in \{0,1\}^{256} \coloneqq H(\mu||w_1)$
   e) $c \in B_\tau \coloneqq SampleInBall(\tilde{c})$
   f) $z \coloneqq y + cs_1$
   g) $r_0 \coloneqq LowBits_q(w - cs_2, 2\gamma_2)$
   h) if $||z||_\infty \geq \gamma_1 - \beta\ or\ ||r_0||_\infty \geq \gamma_2 - \beta, then(z,h) := \perp$
   i) else:
      - $h := MakeHint_q(-ct_0, w - cs_2 + ct_0, 2\gamma_2)$
      - if $||ct_0||_\infty \geq \gamma_2$ or the number of 1's in h is greater than ω, the (z, h) := ⊥
   j) $k \coloneqq k + l$
6. return $\sigma = (z, h, \tilde{c})$

The signing and verification procedures begin with extracting the matrix $\hat{A}$ (NTT domain representation of A) from seed ρ so that the size of public key gets small. A 32-byte seed ζ is stored by signer that is used to create random ρ, K and s_1, s_2 in key generation algorithm, if the signer wants to reserve a secret key as small as possible. The challenge vector c is also stored in NTT representation ($\hat{c}$).

3) Verification:

If $||ct_0||_\infty < \gamma_2$, then:

$$UseHint_q(h, w - cs_2 + ct_0, 2\gamma_2) = HighBits_q(w - cs_2, 2\gamma_2) \quad (2)$$

Since $w = Ay$ and $t = As_1 + s_2$, that implies

$$w - cs_2 = Ay - cs_2 = A(z - cs_1) - cs_2 = Az - ct \quad (3)$$

and

$$w - cs_2 + ct_0 = Az - ct_1.2^d \quad (4)$$

Therefore verifier computes

$$UseHint_q(h, Az - ct_1.2^d, 2\gamma_2) = HighBits_q(w - cs_2, 2\gamma_2) \quad (5)$$

Furthermore, because β is such that $||cs_2||_\infty \leq \beta$ and the signer checks that $LowBits_q(w - cs_2, 2\gamma_2) < \gamma_2 - \beta$, that implies to

$$HighBits_q(w - cs_2, 2\gamma_2) = HighBits_q(w - cs_2 + cs_2, 2\gamma_2) = HighBits_q(w, 2\gamma_2) = w_1 \quad (6)$$

Therefore, the w_1' calculated in the verification process matches with the w_1 calculated in the signing procedure. Thus the verification procedure accepts.

Steps to verify the signature:

1. $A \in R_q^{k\times l} := ExpandA(\rho)$
2. $\mu \in \{0,1\}^{384} :=$ CRH($tr||M$)
3. $c :=$ SampleInBall($\tilde{c}$)
4. $w_1' := UseHint_q(h, Az - ct_1.2^d, 2\gamma_2)$
5. return $[\![||z||_\infty < \gamma_1 - \beta]\!]$ and $[\![\tilde{c} = H(\mu||w_1,)]\!]$ and [[number of 1's in h is $\leq \omega$]]

C. Variations in Crystals Dilithium Algorithm

As stated before Crystal Dilithium has 6 variations, they are Dilithium(2,3,5) and Dilithium(2,3,5)_AES. In these 6 variations, three of them uses SHAKE algorithm (i.e. Dilithium(2,3,5)) and other three uses AES (i.e. Dilithium(2,3,5)_AES) in counter mode. In these variations, 2, 3 and 5 indicate the NIST security Level. As the Dilithium security level increases the signature size and public key size also increases as shown in Table: I. According to analysis, Dilithium3 achieves more than 128 bits of security against classical and quantum attacks as given in [13].

V. Supporting Algorithms and Basic Operations

A. Module - Learning With Errors(LWE)

In lattice based cryptography, Learning with Errors adds complexity to the algorithm. Let A be known matrix and s be solution for the equation $\boldsymbol{As} = \boldsymbol{t}$. It is easy to compute s given (A, t). To improve the complexity in finding solution s, some noise or error is added to the equation such that $\boldsymbol{As} + \boldsymbol{e} = \boldsymbol{t}$, where "$e$" is a small error. A detailed explanation on LWE is given in [14].

If A is $m \times n$ matrix, then s and e are $n \times 1$ vectors and t is $m \times 1$ vector. In ring LWE, each element of the matrix A is restricted to less than a given value q, i.e. each element in $A \in Z_q$. When it comes to Module LWE, each element in matrix A is a polynomial such that the coefficients of the polynomials are restricted to less than a value q and the maximum degree of polynomial is less than 'n'. Each element in $A \in R_q = Z_q/(x^n + 1)$. Mostly q is a large prime number.

B. Shake

The SHA-3 family of algorithms has four variations in cryptographic hash functions, that are SHA3-224, SHA3-256, SHA3-384, and SHA3-512, and two extendable-output functions (XOFs), called SHAKE-128 and SHAKE-256 as given in [8]. For hash functions, suffixes '224', '256' and '384' define the length of output (i.e. digest or hash). Whereas for XOF functions suffixes '128' and '256' defines the cryptographic strength of the method. The hash length of the SHAKE algorithm is flexible as per the users need. In Crystals-Dilithium, SHAKE-128 or SHAKE-256 are used for expansion of seed ρ into polynomial matrix A, seed ρ' into masking vector y and SHAKE-256 is used as hashing function.

C. Fiat-Shamir

Fig.1 represents a Fiat Shamir (FS) interactive model. It is a technique used to make zero knowledge interactive proofs for generating digital signature based on it. Fiat Shamir heuristic model is a template which allows for designing non-interactive argument scheme which replaces interactive access by using cryptographic hash function. The

detailed explanation and also failures of the Fiat Shamir Heuristic model are explained in [15].

D. NTT Representation

First, polynomial $(X^{256}+1)$ splits into $(X - r^i modulo(q))$ with i = 1,3,5,...511 and $r = 1753$. By Chinese remainder theorem (CRT), a cyclotomic ring R_q is thus isomorphic to the product of the rings $Z_q[X]/(X-r^i) \cong Z_q$ as given in [6] and [16]. In the product of rings, multiplying elements is easy since the multiplication is point-wise. Sample mathematical examples are also provided in [16] and [17].

$$a \mapsto (a(r), a(r^3), \ldots, a(r^{511})) : R_q \to \Pi Z_q[X]/(X-r^i) \quad (7)$$

NTT domain representation $\hat{a} = NTT(a) \in Z_q^{256}$ of a polynomial $a \in R_q$ is as shown in (8).

$$\hat{a} = NTT(a) = (a(r_0), a(-r_0), \ldots, a(r_{127}, a(-r_{127})) \quad (8)$$

where $r_i = r^{brv(128+i)}$ where brv(n) the bit-reversal of the 8 bit number n.

E. Expanding the Matrix A

The function ExpandA uses SHAKE-128 which generates a matrix $A \in R_q^{k*l}$ in NTT domain representation from a uniform seed $\rho \in \{0,1\}^{256}$. To reduce the number of multiplication we need NTT domain representation of the matrix A that can be used for reduced multiplication.

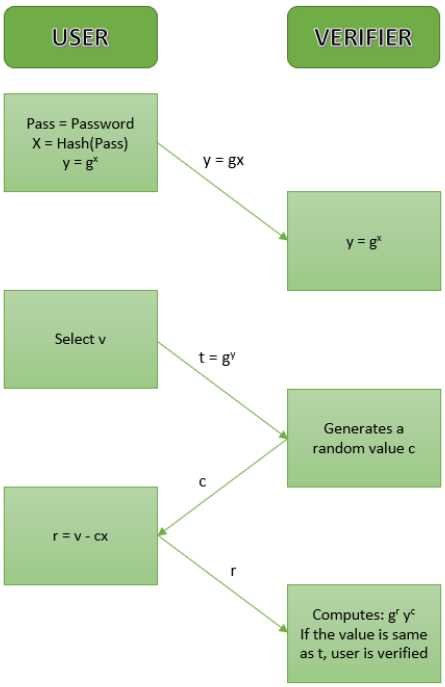

FIGURE 1. FIAT SHAMIR MODEL

F. Sampling the vector y

The function ExpandMask generates $y \in S_{\gamma 1-1}^l$ from a seed ρ' and a nonce k. SHAKE-128 is used for this function.

G. Collision Resistant Hash (CRH)

The function CRH used in this signature scheme is a collision-resistant hash function mapping to $\{0,1\}^{384}$. CRH uses SHAKE-256 as hashing function.

H. High/Low Order Bits and Hints

There are two different ways in which elements in Z_q are broken into their "high-order" bits and "low-order" bits. The algorithm, $Power2Round_q$, break the element "r" in a bitwise way. $r = r_1.2^d + r_0$ where $r0 = r.mod^{\pm}2^d$ and $r_1 = (r - r_0)/2^d$.

In second algorithm, an α is selected such that it divides $q-1$ and then let's say $r = r_1.\alpha + r_0$. Since q is odd, α will be even. The multiples $r_1 \cdot \alpha$'s are now $\{0, \alpha, 2\alpha, \ldots, q-1\}$. Note that the difference between $q-1$ and 0 is 1 since the numbers are within q modulus, and so $q-1$ is removed from the set of possible $r_1.\alpha's$, and the corresponding $r's$ are rounded to 0. Because $q-1$ and 0 differ by 1, the magnitude of the remainder $r_0$increases by 1. This procedure is called $Decompose_q$. Using this procedure as a sub-function, the $MakeHint_q$ and $UseHint_q$ algorithms produce a hint and make use of a hint, respectively are defined. $HighBits_q$ and $LowBits_q$ are used to extract r_1 and r_0, respectively, from the output of $Decompose_q$.

VI. EXPERIMENTATION AND RESULTS

OQS_OpenSSL_1_1_1 is a fork of OpenSSL that uses liboqs for PQC key exchange algorithms and digital signature algorithms for experimentation purpose. OpenSSL fork has been used to generate key pair and digital signature for RSA-2048, RSA-4096, Dilithium2, Dilithium3, Dilithium5, Dilithium2_AES, Dilithium3_AES and Dilithium5_AES algorithms. Steps for downloading the OpenSSL fork is given in [18] and steps to implement digital certificates and digital signature are provided in [18] and [19].

Table I shows the comparison of public key size and signature size of different algorithms in bytes of length.

TABLE I COMPARISON OF ALGORITHMS

Algorithm	Public Key size	Signature size
RSA-2048	256	256
RSA-4096	512	512
Dilithium2	1312	2420
Dilithium3	1952	3293
Dilithium5	2592	4595
Dilithium2_aes	1312	2420
Dilithium3_aes	1952	3293
Dilithium5_aes	2592	4595

VII. ESIGN AS A USECASE

eSign uses RSA-2048 or ECC-P256 based key pair for signing purpose. For the same or higher security strength, possible choices for digital signing can be Dilithium3, Dilithium5, Dilithium3-AES and Dilithium5-AES schemes.

In comparison to RSA 2048, using the above schemes will increase the Public Key size by a factor of 7.6 for Dilithium3 & Dilithium3-AES variants and a factor of 10.1 for Dilithium5 & Dilithium5-AES variants. Similarly, the

signature size will increase by a factor of 12.9 for Dilithium3 & Dilithium3-AES variants and a factor of 17.9 for Dilithium5 & Dilithium5-AES variants.

In traditional dongle based method of signing, the key pair resides in the dongle. The dongle can be used till the validity of the certificate issued by the CA. In eSign based digital signing, a key pair needs to be generated for each transaction. This requires high speed key generation facility in software or hardware used for the same. Typically, Hardware Secure Module (HSM) are used for key generation & signing purpose and traditionally HSMs are meant to secure the keys. This implies that key generation speeds may not be high.

Adoption of PQC digital signing schemes shall require implementation of the same in HSM framework as well. For serving real-time demand in frameworks like eSign, optimizing key generation speed shall be a key requirement.

CONCLUSION

In this paper, the algorithms which are used to build Crystals Dilithium algorithm and the signing procedure used in Crystals Dilithium algorithm are provided. It is found that Dilithium3 variation possess more than 128 bit strength against classical and quantum attacks which is on par with that of RSA-3072. From experimentation using OpenSSL and OpenSSL fork it is found that signature length using RSA-2048 and RSA-4096 are 256 and 512 bytes respectively whereas the signature length for Dilithium2, Dilithium3, Dilithium5 are 2420, 3293, 4595 respectively. As a use-case, the variant of Dilithium algorithm that can provide the same security level as being provided by currently used Digital Signing algorithm has been explored. Also the requirement of generating PQC based keys for digital signing has been highlighted in eSign framework.

REFERENCES

[1] "Digital Signatures and Certificates," 2020. [Online]. Available: https://www.geeksforgeeks.org/digital-signatures-certificates/.

[2] "NIST pqc Round 3 submissions," 2020. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions.

[3] D. Boneh, "Twenty Years of Attacks on the RSA Cryptosystem," *The New England journal of medicine,* 1999.

[4] M. A. Wright, "The Impact of Quantum Computing on Cryptography," *Computer Fraud and Security,* 2017.

[5] R. Kaur and A. K. , "Digital signature," *Proceedings: Turing 100 - International Conference on Computing Sciences, ICCS,* 2012.

[6] Shi Bai and Leo Ducas, "CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation," *National Institute of Standardisation and Technology,* 2020.

[7] Leo Ducas and Tancrede Lepoint, "Crystals - Dilithium: Digital Signatures from Module Lattices," 2021.

[8] Information Technology Laboratory NIST, "SHA-3 STANDARD: PERMUTATION-BASED HASH AND EXTENDABLE OUTPUT FUNCTIONS," *NIST Federal Information Processing Standard,* 2015.

[9] Sara Ricci and Lukas Malina, "Implementing Crystals - Dilithium Signature Scheme on FPGAs," 2021.

[10] "eSign Gazette Announcement," 2015. [Online]. Available: https://cca.gov.in/eSign_gazette_notification.html.

[11] "eSign," [Online]. Available: https://cca.gov.in/eSign.html.

[12] Junting Xiao and Tadahiko Ito, "Performance Comparisons and Migration Analyses of Lattice-based Cryptosystems on Hardware Security Module," 2020.

[13] "CRYSTALS Cryptographic Suite for Algebraic Lattices," 2020. [Online]. Available: https://pq-crystals.org/dilithium/.

[14] Vadim Lyubashevsky, Chris Peikert and Oded Regev, "On Ideal Lattices and Learning with Errors over Rings," *Advances in Cryptology – EUROCRYPT,* 2010.

[15] Nir Bitansky and Dana Dachman-Soled, "Why "Fiat-Shamir for Proofs" Lacks a Proof," vol. 7785, no. 1017660, 2013.

[16] "The Kyber/Dilithium NTT," 2020. [Online]. Available: https://dsprenkels.com/ntt.html.

[17] "Number theoretic transform," 2017. [Online]. Available: https://www.nayuki.io/page/number-theoretic-transform-integer-dft.

[18] "open-quantum-safe," 2020. [Online]. Available: https://github.com/open-quantum-safe/openssl.

[19] "How-to-use-OpenSSL," June 2019. [Online]. Available: https://opensource.com/article/19/6/cryptography-basics-openssl-part-2.