Constructing the Cryptographic Boundaries for Lattice-based Cryptography on Hardware Security Module

Junting Xiao[1,a)] Tadahiko Ito[1,b)]

Abstract: Post-Quantum Cryptography (PQC) is regarded as an effective way to resist attacks with quantum computers. Since National Institute of Standards and Technology (NIST) proposed its PQC standardization project in 2016, many candidates have been submitted and their quantum-resistant capability has been measuring by researchers. Besides this research, this paper evaluates the separation of hash and asymmetric operations in PQC operations. This paper is relatively focused on encryption of large data (e.g. document signing or code signing), instead of small data (e.g. key encryption, authentication). Regardless of the size of the data, it is desirable to be able to use Hardware Security Module (HSM) in the key management of PQC. In addition, it is desirable that the encryption processing API is the same regardless of the size of the data, the presence or absence of HSM, use case, key usage, etc. This document describes that, for a given device (such as HSM) and usecase, the same API may not be available. For a usecase with a small message size, people may input a plain message into the PQC operation, for another use case with a large message size, people may input a hash of plain message into the PQC operation. In theory, this issue can be easily solved by adding a hash function to each of the current PQC candidates and construct an API for that. In that case, the input to the PQC calculation cannot be a plain message, and will always be the hash of message. However, such an approach seems to be outside the scope of the current PQC evaluation, and there is a possibility that difficulties in cryptographic boundaries handling, theoretical proof, or patents may occur. If that approach meets such a difficulty, then there is a possibility that the API of the encryption processing cannot be unified with regard to the size of the data, the presence or absence of HSM, use case, or key usage. We believe this is an undesirable results for PQC user, so we call for the need for integration of API, regardless of any conditions.

Keywords: Lattice-based cryptography, hardware security module, cryptographic boundary

1. Introduction

A hardware security module (HSM) is a physical computing device that provides a trusted environment to perform cryptographic operations such as encryption, decryption, authentication, etc. HSMs typically satisfy the FIPS 140-2 [12] and/or common criteria standard to achieve high security. Relying on secure mechanisms to create an isolated environment from normal computing environments, HSMs ensure reliable generation, protection, and managment of keys and sensitive data. They are now widely used in some critical infrastructure, for instance, be used as a part of public key infrastructure (PKI) or internet bank infrastructure. In these cases, many HSMs are connected together to preserve high practicality and efficiency. On the other hand, Shor introduced an algorithm [20] to solve the integer factorization problem and discrete logarithms problem that can break RSA and elliptic curve cryptography (ECC) on quantum computers. In that case, HSMs which are still using traditional cryptography may not be able to protect and manage their keys and sensitive data securely under quantum attacks. To solve this problem foundamently, migration from traditional cryptography to quantum-resistant cryptography is necessary.

Post-Quantum Cryptography (PQC) is regarded as an effective way to resist attacks from quantum computers. Since National Institute of Standards and Technology (NIST) proposed its PQC standardization project in 2016, many candidates have been submitted and the quantum-resistant capabilities of each post-quantum cryptography scheme have been evaluated by researchers. There are several different ways to construct quantum-resistant cryptographic schemes, such as lattice-based cryptography [17], hash-based cryptography [15] [3], multivariate-based cryptography [22], code-based cryptography [14], etc. Comparing with traditional digital signature schemes such as RSA or ECDSA, post-quantum digital signature schemes generate much larger key pairs or signatures, and they cost more time for key generating, signing and verification. These features restricted their practicality, especially when applied to the constrained environments.

[1] Intelligent Systems Laboratory, SECOM CO., LTD.
[a)] shu-sho@secom.co.jp
[b)] tadahi-ito@secom.co.jp

Lattice-based cryptography is one of the competitive candidates. Its practicality on constrained environments had been investigated by some researchers. For many lattice-based cryptographic schemes, polynomial multiplication and discrete Gaussian sampling are two main challenges on devices with constrained memory and limited computing power. Albrecht et al. [1] implemented "Kyber", presented in [4] on some smart card platforms by using RSA/ECC co-processor and APIs. Yuan et al. [23] proposed a memory-constrained implementation of several lattice-based cryptographic schems on a standard Java Card platform by improving Montgomery modular multiplication (MMM) [16] and number theoretic transform (NTT) for polynomial multiplication and modifying several discrete Gaussian sampling algorithms. On the other hand, another factor that is likely to affect the practicality and the efficiency, is the cryptographic boundary as defined in FIPS 140-2 [12]. This factor have not been evaluated much. To perform trusted cryptographic operations in HSM, the way of applying each component of digital signature schemes should be clearly designed. Most wildly used cryptographic algorithms such as RSA and ECDSA allow the separation of the hash function and asymmetric operations as default. Sugiyama et al. [21] implemented and evaluated the performance of one of such separable algorithm, TESLA#, on Safenet ProtectServer Network HSM. It indicated that it is possible to apply PQC on HSM. In the other hand, the usage of the SHA3 hash function is not separated with asymmetric operation for some lattice-based cryptographic schemes. Because of that, those non-separable implementations of PQC may have limited performance according to varied sizes of input messages. We aim to evaluate the practicality of lattice-based cryptographic schemes which are using SHA3 hash functions on HSM. The details will be introduced in Section 2 and Section 3.

1.1 Usecases

HSM is used not only for authentication operations, but also used for protecting contract data, medical data, CAD data, timestamps, etc. For the latter case, data controller and key manager are likely to be different stakeholders, and different information management policy and operations are applied to their devices. If the devices of each stakeholder are unified to the same security level, a same information management policy has to be chosen, and each stakeholder needs to share operations of their devices. It can be operationally and legally challenging to deal with such a change.

1.2 Our Contributions

First, we consider the issue of dealing with cryptographic boundaries when applying lattice-based cryptography to HSM, and our work evaluates the practicality of lattice-based digital signature schemes by comparing the performance of hash functions operated inside versus not inside of the same cryptographic boundary for HSM.

Some people may say the issue we have raised can be solved by just using fixed-length digests of a message instead of the message itself. However, when using such means without unifying API, it is conceivable that the role of hashes in a cryptosystem with HSM and the role of hashes in a cryptosystem without HSM may change, and we may not able to have interoperability between them. Change in the role of hash can also affect theoretical proof. In addition, it is hard to say that there is no possibility that a new patent risk will occur due to changes on the use of hashes.

We also propose an appropriate way to construct cryptographic boundaries for lattice-based cryptographic schemes when applied to HSM. Moreover, some real world usecases were considered and the relative challenges were introduced.

The rest of this paper is organized as follows. We give a brief mathematical background of lattice and an introduction to HSM and cryptographic boundary in Section 2. We describe the details of three lattice-based digital signature schemes and analyze the challenge for applying them to HSM in Section 3. Evaluation of the results is given in Section 4. We then evaluate the cost in Section 5. Finally, we conclude the paper in Section 6. We believe that our results help to define cryptographic boundary for PQC, where theoretical proof and clearance of patents should be done.

2. Preliminaries

In this section, we give a brief mathematical description of lattice-based cryptography in Section 2.1, we introduce the general way of using digital signature schemes in HSM. In Section 2.2, we point out the challenge of our work. Then in Section 2.3, we introduce the concept of the cryptographic boundary which plays a very important role in this paper .

As defined below. bold italic letters denote polynomials (e.g. $\boldsymbol{f}$), bold upper-case letters denote matrices (e.g. $\mathbf{A}$), and bold lower-case denote vectors (e.g. $\mathbf{v}$). For a probability distribution S, $s \leftarrow S$ denotes that s is chosen according to S. Let n and q be positive integers, $\mathbb{R}$ represents the field of real numbers, $\mathbb{Z}_q$ is the set of integers $0, 1, ..., q-1$, and $\mathbb{R}_q = \mathbb{Z}_q[x]/(x^n+1)$ is the quotient polynomial ring.

2.1 lattice

A lattice is a subgroup of the Euclidean space. Let $\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\} \in \mathbb{R}^{m\times n}$ be a set of linearly independent vectors, the lattice generated by $\mathbf{A}$ is the set $L(\mathbf{A}) = \{\sum_{i=1}^{n} \mathbf{a}_i x_i | x_i \in \mathbb{Z}\}$.

$\mathbf{A}$ is referred to as a basis of the lattice $L(\mathbf{A})$, where m and n are the dimension and the rank of the lattice, respectively. In this paper, we will be concerned with full rank integer lattices, i.e. $n = m$ and $L(\mathbf{A}) \in \mathbb{Z}^m$.

Many provably secure lattice-based cryptographic schemes are based on the hardnesss of lattice problems in the worst-case. Besides the most classical problems which are shortest vector problem (SVP) and closest vector problem (CVP), other problems such as learning with error's problem (LWE) [18] or ring learning with error's problem (R-LWE) [13] are also used to construct provably secure cryptographic schemes.

2.2 HSM and Hash Functions

A hardware security module (HSM) is a physical computing device that provides a trusted environment to perform cryptographic operations such as encryption, decryption and authentication, etc. HSMs typically satisfy FIPS 140-2 [12] and the Common Criteria standard to achieve high security. Relying on secure mechanisms to create an isolated environment from normal computing environments, HSMs ensure reliable generation, protection, and managment of keys and sensitive data. They are now widely used in some critical infrastructure, for instance, as a part of public key infrastructure (PKI) or internet bank infrastructure. In these cases, many HSMs are connected together to preserve high practicality and efficiency.

Most of the current signing systems allow the separation of hash functions and asymmetric operations. Figure 1 shows that the message is hashed in a message management server and the fixed sizes hash values are sent into the HSM. The transmission of the fixed length digest between the message management server and HSM is quite efficient to implement. Then the signature generation is done inside of the HSM. Traditional cryptography such as RSA or ECDSA that use SHA2 families of hash functions can be combined with HSM in this way.

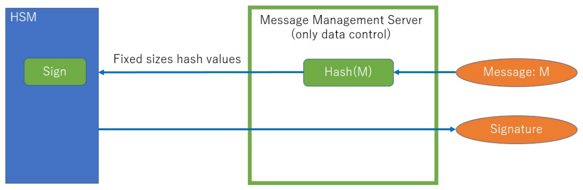

Fig. 1 In general cases, a message is hashed in the message management server and sent to the HSM. The signature generation is finished in the HSM.

On the other hand, an alternative of SHA2, which is called the SHA3 [10] families of hash functions, began to be applied to many PQC cryptographic schemes to improve their security against quantum attacks. In many of these PQC algorithms, random values are generated and hashed with the message together. For some lattice-based cryptographic schemes, the random values are secret or associated with a private key. To prevent leakage of the random values, we may introduce HSMs. In that case, the whole message (instead of hash value) has to be tranmitted into the HSM, and therefore the call for hash functions is calculated inside of it. In this case, the random values and message locate in the same level of the cryptographic boundary whose introduction is given in Section 2.3.

Figure 2 shows the operation when the flexible size message (which could be very large) is sent to the HSM directly. This may require much more resources in the HSM. In this paper, we give experimental results of the differences of the resources cost of hash functions for three lattice-based cryptographic schemes and propose a way to avoid that increase.

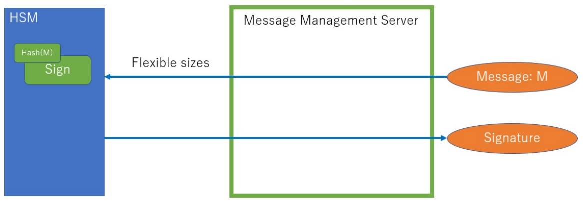

Fig. 2 For some cryptographic schemes in which the message is hashed with secure values, the digest generation and signature generation should be finished in the HSM.

2.3 Cryptographic boundary

Cryptographic boundary was defined in FIPS 140-2 [12]. For a cryptographic module used within a cyber system, the cryptographic boundary establishes the physical bounds that contain all the software, hardware, and firmware of this cryptographic module. It is essential to clearly define the range of the cryptographic boundary in order to guarantee the security of the cryptographic module.

In this section, we describe the method to construct a cryptographic boundary for efficient operations. For example, if we want to implement a system that has a similar architecture to Figure 1 or Figure 2 and execute in a single cryptographic boundary (as illustrated at Figure 3), access controls need to be prepared for protecting keys in the cryptographic boundary, which tend to be costly. On the other hand, if the cryptographic boundary contains more components of a cryptographic module, operations across the cryptographic boundary may increase, and therefore a more complex access control mechanisms would be needed, which would again increase implementation cost.

It is considered to be more efficient to build more than one cryptographic boundary for a given cryptographic module. To be more precise, a system can be implemented and located in several cryptographic boundaries like the way shown in Figure 4. By this implementation, processes related to key objects are stored in the inner cryptographic boundary, and data management of to-be-signed data would be done with access control of the outer cryptographic boundary. Benefits of such an implementation include the following.

- Access control of keys and their metadata should be extremely strict. This implementation can minimize the scope of such strict access control.
- Basically, data flows across the inner cryptographic boundary are fixed size data. Therefore, it is much easier to facilitate data into the cryptographic boundary.
- System migration is accomplished easier. The transition of the whole system can be divided into the migrations of inner boundary and outer boundaries. Although the API of the inner boundaries needs to have an interoperability, migration can be divided into non-dependent steps, and costs of migration can be reduced.

In Section 3 and Section 4, we will give the introduction of the three lattice-based digital signature schemes and analyse the way of constructing appropriate cryptographic boundaries when applying them to HSM. We discuss the details of migration costs in Section 5.

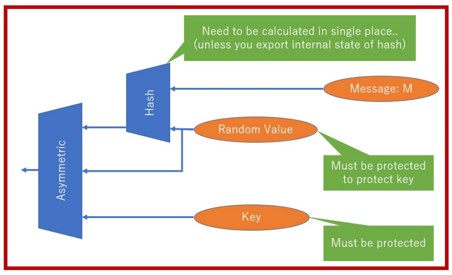
Need to be calculated in single place..
(unless you export internal state of hash)
Hash
Message: M
Asymmetric
Random Value
Must be protected to protect key
Key
Must be protected

Fig. 3 Message locates in same boundary with signature generation operation for cryptographic schemes in which the message digest is derived by hashing the conjunction of the message and some secret value.

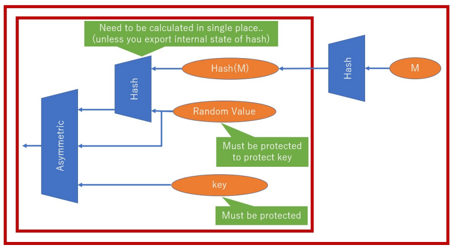
Need to be calculated in single place..
(unless you export internal state of hash)
Hash
Hash(M)
Hash
M
Asymmetric
Random Value
Must be protected to protect key
key
Must be protected

Fig. 4 Message locates in different boundary from signature generation operation for those cryptographic schemes that allow the separation of the hash function and asymmetric operation.

3. Lattice-based Digital Signature Schemes

In this section, we give introductions to three lattice-based digital signature schemes, FALCON, CRYSTALS-DILITHIUM and qTESLA. All of these were second round submissions of NIST's PQC project. FALCON and CRYSTALS-DILITHIUM were selected into the newly third round submissions in July, 2020. However, qTESLA could not be in the third round candidate list, because the authors had modified the signature generation of qTESLA in version 2.8, so that the message is hashed once before the SHA3 operation to derive the message digest. This change may make qTESLA more suitable to be used in HSM. We will first give the introduction to these digital signature schemes. Then we will analyze the practicality of their signature generation operations executing on HSMs.

3.1 FALCON

Fouque et al. [11] proposed the fast fourier lattice-based compact signatures (FALCON) which is based on NTRU lattices. Algorithm 1 shows the signature generation of FALCON.

Algorithm 1 shows the signature generation of FALCON. Function $HashToPoint()$ is based on a SHAKE-256 hash function and is referred to as algorithm 7 in [11]. $ffSampling()$ represents the fast fourier sampling algorithm, referred to as algorithm 19 in [11]. The compression function $Compress()$ is referred to as algorithm 21 in [11]. The function $FFT()$ is the fast fourier transform representation and $invFFT()$ is its inverse.

In step 1 of Algorithm 1, a salt value $\mathbf{r} \in \{0,1\}^{320}$ is generated uniformly at random. In step 2, the message digest is derived by hashing the conjuction of $\mathbf{r}$ and $\mathbf{m}$. Two kinds of cryptographic boundaries can be designed for FALCON. The asymmetric operations are processed in HSM. In order to protect the random generation of $\mathbf{r}$, three components, "Generator", "Hash" and "Asymmetric" can be included into a single cryptographic boundary to share the same level of protection as shown in the left side of Figure 5. But if a secure execution environment for random generation of $\mathbf{r}$ can be prepared carefully, step 1 and step 2 of algorithm 1 can be handled outside of the HSM as shown in the right side of Figure 5.

```
Algorithm 1: Signature Generation of FALCON
    Input  : Private key sk; Message m; A bound $\beta$.
    Output: The signature $sig = (\mathbf{r}, \mathbf{s})$.
 1  $\mathbf{r} \in \{0,1\}^{320}$ uniformly
 2  $c = HashToPoint(\mathbf{r}||\mathbf{m})$
 3  $\mathbf{t} = (FFT(c), FFT(0)) \cdot \widehat{\mathbf{B}}^{-1}$
 4  do
 5      $\mathbf{z} = ffSampling_n(\mathbf{t}, \mathbf{T})$
 6      $\mathbf{s} = (\mathbf{t} - \mathbf{z})\widehat{\mathbf{B}}$
 7  while $||s|| > \beta$;
 8  $(s_1, s_2) = invFFT(\mathbf{s})$
 9  $\mathbf{s} = Compress(s_2)$
10  return $sig = (\mathbf{r}, \mathbf{s})$
```

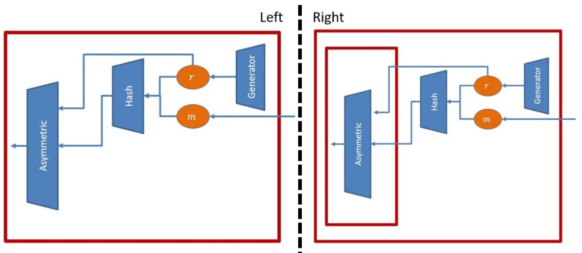

Fig. 5 The structure of cryptographic boundaries for FALCON (Left: the hash operation locates in same cryptographic boundary as asymmetric operations in HSM. Right: the hash operation locates in different cryptographic boundary from asymmetric operations in HSM.)

3.2 CRYSTALS-DILITHIUM

Ducas et al. [9] proposed the CRYSTALS-DILITHIUM digital signature scheme, which is based on the hardness of the Shortest Vector Problem (SVP) in lattice.

Algorithm 2 shows the signature generation of DILITHIUM. Let k, l, $\gamma_1 \in \mathbb{Z}$, q be the moduli, $ExpandA : \{0,1\}^{256} \rightarrow R_q^{k \times l}$, the hash function $CRH() : \{0,1\}^* \rightarrow \{0,1\}^{384}$, $ExpandMask()$ maps a seed ρ' and a nonce k to $\boldsymbol{y} = S_{\gamma_1 - 1}^{l}$. The function $NTT()$ is to compute the number theoretic transform representation, and $NTT^{-1}()$ is to compute its inverse. Funtions $HighBits()$, $Decompose_q()$ and $MakeHint_q()$

refer to figure 3 in [9].

In step 2 of Algorithm 2, $\boldsymbol{\mu}$ is derived by hashing the conjunction of **tr** and message **m**. Since **tr** is part of the private key which should be stored in HSM, and the hash function $CRH()$ have to be calculated inside of HSM. The message **m** needs to be sent into the HSM directly, and the digest is derived from hashing the conjuction of **tr** and **m**. Under these circumstances, the structure of the cryptographic boundary is shown in Figure 6.

Algorithm 2: Signature Generation for Dilithium

```
Input   : $sk = (\boldsymbol{\rho}, \mathbf{K}, \mathbf{tr}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$; Message $\mathbf{m}$.
Output: The signature $\boldsymbol{\sigma} = (\mathbf{z}, \mathbf{h}, c)$.
1  $\mathbf{A} \in R_q^{k\times l} = ExpandA(\boldsymbol{\rho})$
2  $\boldsymbol{\mu} \in \{0,1\}^{384} = CRH(\mathbf{tr}||\mathbf{m})$
3  $k = 0,\ (\boldsymbol{z}, \boldsymbol{h}) = \perp$
4  $\boldsymbol{\rho}' \in \{0,1\}^{384} = CRH(\mathbf{K}||\boldsymbol{\mu})$ (or $\rho' \leftarrow \{0,1\}^{384}$ for
     randomized signing)
5  while $(\boldsymbol{z}, \boldsymbol{h}) = \perp$ do
6      $\boldsymbol{y} \in S^l_{\gamma_1 - 1} = ExpandMask(\boldsymbol{\rho}', k)$
7      $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{y}$
8      $\boldsymbol{w}_1 = HighBits_q(\boldsymbol{w}, 2\gamma_2)$
9      $\mathbf{c} \in B_{60} = H(\boldsymbol{\mu}, \boldsymbol{w}_1)$
10     $\boldsymbol{z} = \boldsymbol{y} + \mathbf{c}\boldsymbol{s}_1$
11     $(\boldsymbol{r}_1, \boldsymbol{r}_0) = Decompose_q\ (\boldsymbol{w}$ - $\mathbf{c}\boldsymbol{s}_2,\ 2\gamma_2)$
12     if $||\boldsymbol{z}||_\infty \geq \gamma_1 - \beta$ or $||\boldsymbol{r}_0||_\infty \geq \gamma_2 - \beta$ or $\boldsymbol{r}_1 \neq \boldsymbol{w}_1$
         then
13         $(\boldsymbol{z}, \boldsymbol{h}) = \perp$
14     end
15     else
16         $\boldsymbol{h} = MakeHint_q(-\mathbf{c}\boldsymbol{t}_0, \boldsymbol{w} - \mathbf{c}\boldsymbol{s}_2 + \mathbf{c}\boldsymbol{t}_0, 2\gamma_2)$
17         if $||\mathbf{c}\boldsymbol{t}_0||_\infty \geq \gamma_2$ or the # of 1's in $\boldsymbol{h}$ is greater
             than $\omega$ then
18             $(\boldsymbol{z}, \boldsymbol{h}) = \perp$
19         end
20     end
21     $k = k + 1$
22 end
23 return $\boldsymbol{\sigma} = (\boldsymbol{z}, \boldsymbol{h}, \mathbf{c})$
```

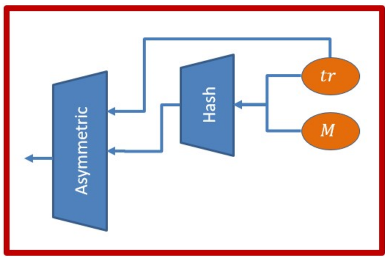

Fig. 6 The structure of cryptographic boundaries for CRYSTALS-DILITHIUM: the hash operation locates in the same cryptographic boundary as asymmetric operations in the HSM)

3.3 qTESLA

Akleylek et al. [2] proposed the qTESLA digital signature scheme, which is based on the hardness of the decisional ring learning with errors (R-LWE). Algorithm 3 shows the signature generation for qTESLA.

Algorithm 3 shows the signature generation of qTESLA, $PRF_2() : \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^{320} \rightarrow \{0,1\}^k$ is performed as a pseudorandom function. $ySampler() : \{0,1\}^k \times \mathbb{Z} \rightarrow R_{[B]}$ is referred to as algorithm 12 in [2]. $GenA() : \{0,1\}^k \rightarrow R_q^k$ is referred to as algorithm 10 in [2]. In step 3 of Algorithm 3, the function $PRF_2()$ hashes message m with secret data $\mathbf{seed}_y$ and a random value $\mathbf{r}$, the usage of $G() : \{0,1\}^* \rightarrow \{0,1\}^{320}$ was first introduced in the Ver. 2.8 (11/08/2019) of qTESLA in [2], which made it possible to transmit the fixed length message digest $G(\mathbf{m})$, instead of $\mathbf{m}$, into the HSM.

This modification made it possible to redesign the cryptographic boundaries used for qTESLA in HSM. The difference is shown in Figure 7. Since the secure value $\mathbf{seed}_y$ needs to be stored in the HSM, without the optimization in Ver. 2.8, the original message m is transferred into HSM directly and the structure of the cryptographic boundary is shown in the left part of Figure 7. After Ver. 2.8, the cryptographic boundary can be constructed in the way shown in the right part of Figure 7. The comparison for the practicalities of these two structures of qTESLA will be introduced in the next section.

Algorithm 3: Signature Generation for qTESLA

```
Input   : $sk = (\boldsymbol{s}, \boldsymbol{e}_1, ..., \boldsymbol{e}_k, \mathbf{seed}_a, \mathbf{seed}_y, \mathbf{g})$; Message
          $\mathbf{m}$.
Output: The signature $sig = (\boldsymbol{z}, \boldsymbol{c}')$.
1  counter = 1
2  $\mathbf{r} \in \{0,1\}^k$
3  $\mathbf{rand} = PRF_2(\mathbf{seed}_y,\ \mathbf{r},\ G(\mathbf{m}))$
4  $\boldsymbol{y} = ySampler(\mathbf{rand}$, counter$)$
5  $\boldsymbol{a}_1, ..., \boldsymbol{a}_k \leftarrow GenA(\mathbf{seed}_a)$
6  for $i = 1, ..., k$ do
7      $\boldsymbol{v}_i = \boldsymbol{a}_i \boldsymbol{y} \bmod^{\pm} q$
8  end
9  $\boldsymbol{c}' = H(\boldsymbol{v}_1, ..., \boldsymbol{v}_k, G(\mathbf{m}), \mathbf{g})$
10 $\boldsymbol{c} = \{pos\_list, sign\_list\} \leftarrow Enc(\boldsymbol{c}')$
11 $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{s}\boldsymbol{c}$
12 if $\boldsymbol{z} \notin R_{[B-S]}$ then
13     counter = counter + 1
14     Restart as step 4
15 end
16 for $i = 1, ..., k$ do
17     $\boldsymbol{w}_i = \boldsymbol{v}_i - \boldsymbol{e}_i \boldsymbol{c} \bmod^{\pm} q$
18     if $||[\boldsymbol{w}_i]_L||_\infty \geq 2^{d-1} - E \vee ||\boldsymbol{w}_i||_\infty \geq \lfloor q/2 \rfloor - E$ then
19         counter = counter + 1
20         Restart as step 4
21     end
22 end
23 return $(\boldsymbol{z}, \boldsymbol{c}')$
```

4. Evaluation

In this section, we evaluate the practicality of FALCON, DILITHIUM, and qTESLA when applied to HSM. Hashing operations for three lattice-based digital signature schemes are implemented for evaluating the performance of different cryptographic boundary structures for these digital signature schemes.

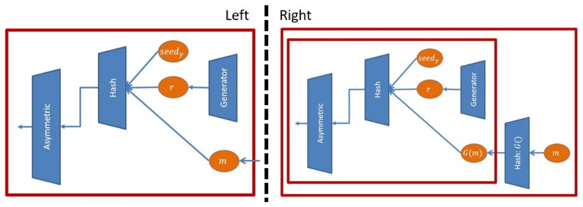

Fig. 7 The structure of cryptographic boundaries for qTESLA (Left: the hash operation locates in the same cryptographic boundary as asymmetric operations in the HSM for old version. right: the hash operation locates in different cryptographic boundary from asymmetric operations in the HSM from Ver. 2.8)

4.1 HSM Specification

Throughout the paper, two types of HSMs are chosen for our experiments. One is the Protect Server External 2 (PSE-2), and the other one is the LUNA NETWORK HSM A750 (LunaSA-7), both of which are owned by Thales S.A.. Table 1 gives some descriptions of PSE-2 and LunaSA-7. In order to make use of the SHA3 mechanism in our experiments, suitable version of HSM's software and firmware were selected and installed[*1].

4.2 Experimental Results

We call it boundary type "A" when the message (no matter how long the length is) is sent to the HSM directly, and there is only one cryptographic boundary constructed for signature generation. This structure is similar to Figure 3. The boundary type "B" indicates that only the fixed length message digest is transfored into HSM, and there is another cryptographic boundary for protecting the generation of message digest from message. This structure is similar to Figure 4. Table 2 shows the time costs for hashing operations executed inside of HSM for FALCON, CRYSTALS-DILITHIUM, and qTESLA with different constructions of cryptographic boundaries. The time is measured as milliseconds (ms). The size of the message is measured as kilobyes (K) or megabytes (M). For FALCON, when message is hashed in another cryptographic boundary from the HSM, the time cost of hash operations inside the HSM is 0. However, for boundary type A, when the message size becomes larger and larger, the time cost rises linearly. If the message size is 10M, the time cost of hash operation is about 11667.19ms($\approx$11.7s) in PSE-2, and about 4911.52ms($\approx$4.9s) in LunaSA-7. For CRYSTALS-DILITHIUM, because the message digest is derived by hashing the conjunction of message and components of secret key together, all the operations of signature generation locate in the same boundary and therefore the time cost rises with the extension of the message size. For instance, if the message size is 10M, the time cost of hash operation is about 12727.83ms($\approx$12.7s) in PSE-2, and about 6294.30ms($\approx$6.3s) in LunaSA-7. For qTESLA, when boundary type A is applied, the time cost grows with the extension of message size. For instance, if the message size is 10M, the time cost of hash operation is about 11810.52ms($\approx$11.8s) in PSE-2, and about 4922.03ms($\approx$4.9s) in LunaSA-7. After the optimization from Ver. 2.8 of qTESLA, no matter how long the size of the original message, a fixed size message digest is generated and sent into the HSM. Therefore, the time cost is almost the same for each case, as shown in the second row of the experiment results for qTESLA. Comparing the performance of type A and type B applied to qTESLA, when the message size is 10M, the speed is about three-hundred times faster in PSE-2, and more than one-thousand times faster in LunaSA-7.

5. Migration costs for each cryptographic boundary

RSA (with sha2) and ECDSA (with sha2) signing systems using HSMs typically utilize type B cryptographic boundaries. This section describes migration costs of those signing systems toward lattice-based signatures with cryptographic boundaries of type A or type B.

5.1 Migration costs for boundary type A

To migrate to type A, many more components of lattice-based signatures are contained in a single cryptographic boundary, so as a general rule, it is expected that the access control mechanism has to be designed to be more complicated, which is likely to require much more processing resources for access control. These changes should be done after a thorough threat analysis, redefining of threat models, and redefining of human operations.

In addition, as described in Section 4, a lot of hash calculations are accomplished inside of the type A cryptographic boundary for key management, so much more protected computing resources inside of the boundary may be required to sign large files.

5.2 Migration costs for boundary type B

To migrate to type B, it is possible to utilize the same kind of cryptographic boundary as traditional systems like RSA or ECDSA. In this case, although it is necessary to support the lattice-based cryptographic algorithms with corresponding object ID, the change in architecture of cryptographic boundary between lattice-based and traditional implementations is likely to be limited.

As shown in Figure 8, if it were possible to place both lattice-based and traditional cryptographic modules into the same boundary, or, if it were possible to switch the lattice-based-signatures-related inner boundary and traditional-signatures-related inner boundary from inside of their common outer boundary, changes of human operation would also be limited. This approach also require interoperability and standardization of APIs, but the benefits of success would be great.

[*1] For PSE-2, HSM appliance version and client software version are 5.6. For LunaSa-7, HSM appliance version is 7.4, client software version is 10.2

Table 1 Some descriptions of PSE-2 and LunaSA-7

Features	PSE-2	LunaSA-7
Operating Systems	• Windows, Linux, AIX, HP_UX, Solaris	• Windows, Linux, Solaris, AIX • Virtual: VMware, Hyper-V, Xen, KVM
APIs	• PKCS#11, CAPI/CNG, JCA/JCE, JCProv, OpenSSL	• PKCS#11, CAPI/CNG, JCA/JCE, JCProv, OpenSSL • REST API for administration
Security Certifications	• FIPS 140-2 Level 3	• FIPS 140-2 Level 3 • eIDAS CC EAL4+ (AVA_VAN.5 and ALC_FLR.2) against the Protection Profle 419221-5 *
Cryptography	• Asymmetric • Symmetric • Hash/Message Digest/HMAC • Message Authentication Codes	• Asymmetric • Symmetric • Hash/Message Digest/HMAC • Full Suite B support • Key Derivation • Key Wrapping • Random Number Generation • Digital Wallet Encryption • 5G Cryptographic Mechanisms for Subscriber Authentication

Table 2 The time cost for hashing operations executed inside of the HSM (PSE-2) for three lattice-based digital signature schemes with different structures of cryptographic boundary

Scheme	Boundary Type	Time (millisecond)				
		1k	10k	100k	1M	10M
FALCON	A	33.36	38.08	142.26	1240.59	11667.19
	B	0				
DILITHIUM	A	34.67	45.79	156.19	1351.4	12727.83
qTESLA	A (before Ver. 2.8)	34.78	44.78	138.05	1196.26	11810.52
	B (from Ver. 2.8)	30.84	38.25	38.63	38.63	31.42

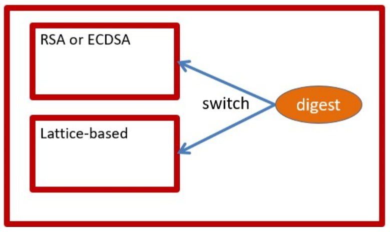

Fig. 8 A mechanism that allows both lattice-based-signatures-related inner boundary and traditional-signatures-related inner boundary to be located inside the same outer boundary

5.3 Migration costs from type B with multi-stakeholders

In the real world, it is quite common that the organization that manages data and the organization that provides the signing service (with HSM) are different. In general, the signing service providers use their HSM resources to sign data for different customers (or data controller). That separation is efficient, because the signing provider and data controller would have very different data management and data control policies, but each stakeholder can only focus on their resource for each policy.

If each stakeholder can make their devices the same security level, each operation can be regarded as located in a single cryptographic boundary similar to type A, when communication channels with strict secure principles are prepared to transfer data. However, two main challenges are likely to destroy this solution. The first one is that it is resource-consuming for each data controller to run their devices in a same security level as signing provider, and also it can be impossible for signing provider to make their system compliant with every data controller's policy. Even if the first challenge was solved, the second challenge of the time consumption of transferring data with secure communication channels is also unavoidable. As shown in Table 4, medical data or CAD data are quite big, so the type A cryptographic boundaries would be very inefficient.

Therefore, it is more realistic to construct more than one cryptographic boundary for multi-stakeholders. Under this circumstance, each stakeholder has its own security solutions for its devices, and a strict access mechanism is designed for the HSM service provider.

As shown in Table 4, for authentication purposes, the data size is small, and service provider may not need to store that data. For contract PDF files, the typical data size is several hundreds KB. That data needs to be preserved for the contract period, which can be several decades. For medical data, some data can be on the order of GBs. For instance, raw data for Multi-slice CT can be 2GB for a single inspection (2MB/slice, 1000 slices). That data may need to be stored for a patient's lifetime. For some PDF file with CAD data, the file size can be several GBs. Notice that HSM has the limitation on RAM resource, therefore, it is hard to process the GB files in it. The structure of cryptographic boundaries should be type B for this case.

We believe that the implementation and migration of PQC are critical for signing purposes. Because it is difficult to predict when PQC era will come, and as the lifetime of data becomes longer, it would be more desirable to prepare the implementation and migration of PQC further in advance. Furthermore, as data size become larger, the efficiency of cryptographic operations on these data would also be important.

Table 3 The time cost for hashing operations executed inside of the HSM (LunaSA-7) for three lattice-based digital signature schemes with different structures of cryptographic boundary

Scheme	Boundary Type	Time (millisecond)				
		1k	10k	100k	1M	10M
FALCON	A	4.29	9.06	52.40	501.48	4911.52
	B	0				
DILITHIUM	A	3.13	8.57	65.94	630.73	6294.30
qTESLA	A (before Ver. 2.8)	3.99	8.78	53.53	507.02	4922.03
	B (from Ver. 2.8)	2.27	3.49	3.15	3.42	2.99

Table 4 Some usecases that can be applied to HSM

Purpose	Usecase	Lifetime of Data	Data Sizes
Authentication	Authentication	days	several KiloBytes
Sign	Contract data	can be decades	hundreds KiloBytes
	Medical data		can be several GigaBytes
	CAD data		
Time stamp	Time stamp	around one decade	several KiloBytes

6. Conclusion

In this paper, we discussed the practicality of lattice-based cryptography, which is one of the candidates of NIST's PQC project, when applied to Hardware Security Module (HSM). We described the features of three lattice-based digital signature schemes selected from NIST's PQC project, and pointed out that the way of using the hash function restricts their practicality by comparing the performances of hash functions processed inside and outside of HSM with different designs of cryptographic boundaries. We also propose appropriate ways to construct cryptographic boundaries to improve the practicality of PQC when applied to HSM. Moreover, we analysed the migration challenges for some real world usecases which involve many stakeholder, and we introduced our considerations. We believe that our result helps to define the cryptographic boundary for PQC, where theoretical proof and clearance of patents should be done.

References

[1] Martin R. Albrecht, Christian Hanser, Andrea Hoeller, Thomas=Pöppelmann, Fernando Virdia, and Andreas Wallner. "Implementing RLWE-based schemes using an RSA co-processor." In Cryptology ePrint Archive, 2018/425, 2018.

[2] Nina Bindel, Sedat Akeylek, Erdem Alkim, Paulo SLM Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Julaine Kramer, Patrick Longa, Harun Polat, Jefferson E. Richardini, and Gustavo Zanon. qtesla. submission to the nist's post-quantum cryptography standardization process.(2018), 2018.

[3] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. "XMSS - a practical forward secure signature scheme based on minimal security assumptions." In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pp. 117-129. Springer Berlin / Heidelberg, 2011.

[4] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. "CRYSTALS - Kyber: a CCA-secure module-lattice-based KEM." In IACR Cryptology ePrint Archive, Report 2017/634, 2017.

[5] Daniel J. Bernstein, Andreas Hülsing, Stefen Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. "The SPHINCS+ Signature Framework." submission to the nist's post-quantum cryptography standardization process, 2019. `https://sphincs.org/data/sphincs+-paper.pdf`.

[6] Ward Beullens, Bart Preneel, Alan Szepieniec, and Frederik Vercauteren. LUOV, Csrc.nist.gov, Jun 2019, [online] Available: `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions`.

[7] A. Casanova, J.-C. Faugere, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem. "GeMSS: A great multivariate short signature." Jun 2019, [online] Available: `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions`.

[8] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, and Bo-Yin Yang. Rainbow specifications. NIST PQC Round 2 Submission (2019).

[9] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS-Dilithium: A lattice-based digital signature scheme." *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238-268, 2018. `https://tches.iacr.org/index.php/TCHES/article/view/839`.

[10] Morris J. Dworkin. "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions." `https://www.nist.gov/publications/sha-3-standard-permutation-based-hash-and-extendable-output-functions?pub_id=919061`, 2015.

[11] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. "Falcon: Fast-fourier lattice-based compact signatures over ntru." submission to the nist's post-quantum cryptography standardization process.(2018), 2018.

[12] FIPS 140-2. "Security Requirements for Cryptographic Modules." `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf`.

[13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors Over Rings." In IACR Cryptology ePrint Archive, Report 2012/230, 2012.

[14] Robert J. McEliece. "A public key cryptosystem based on algebraic coding theory." *DSN progress report*, 42-44:114-116, 1978.

[15] Ralph Merkle. "A certified digital signature." In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO?89 Proceedings, volume 435 of Lecture Notes in Computer Science*, pp. 218-238. Springer Berlin / Heidelberg, 1990.

[16] Peter L. Montgomery. "Modular multiplication without trial division." In *Mathematics of Computation*, Vol. 44, No. 170, pp. 519-521, 1985.

[17] Daniele Micciancio and Oded Regev. "Lattice-based cryptography." In *Post-Quantum Cryptography*, pp. 147-191. Springer, 2008

[18] Oded Regev, "On lattices, learning with errors, random linear codes, and cryptography." In Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (Baltimore, MD, USA: ACM, 2005), 84-93, `http://portal.acm.org/citation.cfm?id=1060590.1060603`.

[19] Simona Samardjiska, Ming-Shing Chen, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. MQDSS specifications. NIST PQC Round 2 Submission (2019).

[20] Peter Williston Shor. "Algorithms for quantum computation: discrete logarithms and factoring." In *Proceedings of the 35th Annual Symposium on Fundamentals of Computer Science*

(FOCS), pp. 124-134, 1994.

[21] Shotaro Sugiyama, Tadahiko Ito, and Kohei Isobe. "Implementation and Evaluation of Post Quantum Cryptography on Hardware Security Module." In *2019 Symposium on Cryptography and Information Security*, Shiga, Japan. Jan. 22-25, 2019.

[22] Matsumoto Tsutomu and Imai Hideki. "Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and MessageEncryption." *Lecture Notes in Computer Science*. Berlin / Heidelberg. Springer, 1988.

[23] Ye Yuan, Kazuhide Fukushima, Junting Xiao, Shinsaku Kiyomoto, and Tsuyoshi Takagi. "Memory-Constrained Implementation of Lattice-based Encryption Scheme on the Standard Java Card Platform." In IACR Cryptology ePrint Archive, Report 2018/1238, 2018.

[24] Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao Wang, Vladmir Kolesnikov, and Daniel Kales. Picnic. submission to the nist's post-quantum cryptography standardization process, 2018. `https://microsoft.github.io/Picnic/`.