耐量子計算機暗号のHSMへの実装方法とその Cryptographic boundary構成方法についての考察

肖 俊廷[1,a)]　伊藤　忠彦[1,b)]

概要：耐量子計算機暗号 (PQC) は、量子コンピュータによる攻撃に対する効果的な方法と考えられている。2016 年にアメリカ国立標準技術研究所 (NIST) が PQC 標準化プロジェクトを提案して以来、多くの候補が研究者によって提出され、その量子耐性能力が測定されてきた。一方で、PQC の実用性を評価するためには、暗号オペレーションを実行するための信頼できる環境を提供するように設計されたハードウェアセキュリティモジュール (HSM) などの制約された環境での検討を行うことも不可欠となる。そこで本稿では、鍵管理に HSM を用いた事例を想定し、NIST の PQC プロジェクトの候補の一種である格子暗号を適用する実用性について考察する。本書においては、2 つの Cryptographic Boundary を想定した上で、暗号処理における非対称演算操作ではなく、暗号処理におけるハッシュ操作の効率に焦点を当てて検討を行う。また、PQC における暗号操作のボトルネックは、非対称演算操作ではなくハッシュ操作であることを指摘する。特に、ドキュメント署名やコード署名のような、大きなファイルに対する署名が行われる暗号システムにおいては、このボトルネックが効率に大きく影響する。本書では NIST の PQC プロジェクトのラウンド 2 から、3 種類の格子ベースの署名を選び、評価した。また、これらの署名におけるボトルネックを分析し、Cryptographic Boundary の異なる構造下で HSM を利用する場合のパフォーマンスを比較する。その後、格子ベースの暗号方式を、HSM を伴う Cryptographic Boundary 内に構築するための適切な方法を提案する。我々の結果は、（PQC の標準化に際して行われるであろう）安全性証明と特許調査について、その検討を行う範囲を定義するのに役立つものと考えられる。

Efficient Implementation of Post-Quantum Cryptography on Hardware Security Module

Junting Xiao[1,a)]　Tadahiko Ito[1,b)]

Abstract: Post-Quantum Cryptography (PQC) is regarded as an fundamental solution to resist attacks with quantum computers. National Institute of Standards and Technology (NIST) proposed its PQC standardization project in 2016, and candidates had been submitted. Besides these researches, it is necessary to evaluate performance of PQC in “real use case” We assume use of Hardware security module (HSM), for key management, and measure performance of candidate lattice-based cryptographies. In this paper, we focus on the time consumption of hash operations (instead of asymmetric operations), with different constructions of cryptographic boundaries. Then we show that bottleneck of PQC operations can be hash operation (instead of asymmetric operation). Especially for the cases of signing large data, (e.g. document signing and code signing), this bottleneck will affect their efficiency a lot. We chose several lattice-based digital signature schemes and analyses the bottleneck of these schemes and compare their performances under different constructions of cryptographic boundary with HSM. After that, we discuss the appropriate way to construct cryptographic boundary.

1. イントロダクション

本書では、3 種類の格子ベース署名アルゴリズムを、1 種

[1] セコム株式会社 IS 研究所
Intelligent Systems Laboratory, SECOM CO., LTD
[a)] shu-sho@secom.co.jp
[b)] tadahi-ito@secom.co.jp

類の HSM に実装を行った[*1]。

暗号、復号化、認証などの暗号化操作を実行する信頼できる環境を提供する、物理コンピューティングデバイスをハードウェアセキュリティモジュール (HSM) と呼ぶ。多くの HSM は、高度なセキュリティを実現していることを示すために FIPS 140-2 [12] や Common Criteria 等の認証を受けている。HSM は、安全なメカニズムを利用して、通常のコンピューティング環境から分離された環境を作成することで、鍵や機微なデータの生成、保護、管理を実現する。現在 HSM は、公開鍵基盤 (PKI) やインターネットバンキングのインフラの一部などの重要なインフラで広く使用されている。このような場合、多くの HSM が、高い実用性と効率を維持するために接続される。

一方で、量子コンピュータを用いて素因数分解や離散対数問題を解決し、RSA 暗号や楕円曲線暗号 (ECC) を破る可能性のあるアルゴリズムが、Shor [20] により提案されました。このような攻撃により、HSM が従来の暗号方式しか利用できない局面においては、量子コンピュータを用いた攻撃に対して鍵と機微なデータを安全に保護および管理できない可能性がある。このような問題を根本的に解決するためには、HSM で利用する暗号方式を、耐量子計算機暗号 (PQC) 方式に移行する必要がある。耐量子計算機暗号 (PQC) は、量子コンピュータからの攻撃に対抗する効果的な方法と考えられている。2016 年にアメリカ国立標準技術研究所 (NIST) が PQC 標準化プロジェクトを提案して以来、多くの候補が研究者によって提出され、その量子耐性能力が測定されてきた。耐量子暗号を構築するには、格子ベースの暗号方式 [17]、ハッシュベースの暗号方式 [15] [3]、多変量ベースの暗号 [22]、コードベースの暗号 [14] など、いくつかの異なる方法がある。RSA や ECDSA などの従来の署名アルゴリズムと比較すると、耐量子暗号の署名方式は、大きな鍵ペア又は署名を生成し、キーの生成、署名又は検証にかかる時間が増える。これらの機能は、制約された環境に適用される場合に、その実用性を特に制限する。表 1 は、NIST の PQC プロジェクトのラウンド 2 における耐量子暗号の候補のうち、署名方式の候補を示している。

ラティスベースの暗号は、有力な候補の 1 つであり、制約された環境における実用性も、一部の研究者によって調査されてきた。多くの格子ベースの暗号方式を、メモリが制約される又は計算能力が限られているデバイスに実装する場合においては、乗算と離散ガウシアンサンプリングが 2 つの主要な課題となる。Albrecht らは [1] は、RSA/ECC コプロセッサと API を使用して、一部のスマートカードプラットフォームで”Kyber”を実装した。Yuan らは [24] モンゴメリ乗算 (MMM) [16] と多項式乗算のための数値定理変換 (NTT) を改善し、いくつかの離散ガウシアンサンプリングアルゴリズムを変更することにより、標準的な Java カードプラットフォーム上にいくつかの格子ベースの暗号方式を実装した。

一方で、実用性と効率に影響を与える可能性のあるもう 1 つの要因として、(FIPS 140-2 [12] で定義された) Cryptographic Boundary があげられる。Cryptographic Boundary の議論においては、非対称演算とハッシュ演算が分離可能であるかが重要となる。RSA や ECDSA などの現状広く使われている暗号アルゴリズムでは、ハッシュ操作と非対称操作を分離できる。例えば、杉山ら [21] は、分離可能な時の評価として非対称操作とハッシュ操作が分離可能な耐量子計算機暗号 (TESLA#) の性能を、Safenet ProtectServer Network HSM に実装した。

一方で、SHA3 ハッシュ関数を利用する一部の格子ベースの暗号方式では、非対称操作とハッシュ操作は分離されていない。そのため, 入力メッセージの長さは変化し、HSM においては効率が著しく低下する可能性がある。

本書では、SHA3 ハッシュ関数を使用する格子ベースの暗号方式を HSM で実装し、実用性を評価することを目指す。実装の詳細は 2 節と 3 節で節紹介する。

表 1 NIST PQC プロジェクトのラウンド 2 候補 (デジタル署名)

Type	Signature
Lattice-based	CRYSTALS-DILITHIUM [9] FALCON [11] qTESLA [2]
Hash-based	SPHINCS+ [5]
Multivariate-based	GeMSS [7] LUOV [6] MQDSS [19] Rainbow [8]
Zero-knowledge Proofs	Picnic [25]

1.1 成果

本書では、HSM に格子ベースの暗号を実装する上での Cryptographic Boundary の問題を考察し、HSM と同一の Cryptographic Boundary 内で実行されるハッシュ関数に必要なリソースを比較することによって、格子ベースの署名方式の実用性を評価した。また、HSM に PQC を実装する上で、実用性を上げる為の方策を提案した。

我々が挙げた問題は、固定長のメッセージの暗号化の代わりに、そのメッセージに対する固定長のダイジェストを置き換える事で解決できるという見方もある。しかしながら、API を統一する事なくそのような手段を用いた場合、HSM を利用する場合とソフトウェアでの実行において、暗号処理全体におけるハッシュの役割が変わる事が考えられ、他にも相互運用性が害される可能性も考えられる。ハッシュの役割が変わるという事は、安全性証明へ影響を

[*1] 本書は eprint archive で公開した論文の初版 [23] を日本語化及び改訂したものとなる。

及ぼす可能性もある。加えて、ハッシュの使い方の変化に起因する、新たなパテントリスクが発生する可能性も皆無とは言い難い。

本書は、以下のように構成される。2 節では、格子理論の数学的背景や、HSM と Cryptographic Boundary について概説する。 3 節では、3 つの格子ベースの署名方式の詳細を説明し、HSM に適用するための課題を分析する。 4 節では分析に基づく考察を行い、 5 節では費用についての考察を行う。最後に 6 節でまとめを述べる。

我々の結果は、(PQC の標準化に際して行われるであろう) 安全性証明と特許調査について、その検討を行う範囲を定義するのに役立つものと考えられる。

2. 準備

本節では、2.1 節で格子ベース暗号の数学について概説し、2.2 節で HSM における署名の一般的な実装方法と本書が扱う課題を紹介し、その後 2.3 節では本書において重要な概念である Cryptographic boundary について紹介する。

本書を通して、太字の斜体の文字は関数を示し (e.g. $\mathbf{f}$ or $\mathbf{F}$)、太字の小文字はベクトルを示し (e.g. $\mathbf{v}$)、太字の大文字は行列を示す (e.g. $\mathbf{A}$)。集合 S の要素 s が一様ランダムに選ばれることを $s \leftarrow S$ と表す。S が確率分布である場合、$s \leftarrow S$ は S に従って s が選択されることを示す。実数は $\mathbb{R}$ で表す。n と q が正の整数とし、$\mathbb{Z}_q$ は $0, 1, ..., q-1$ の整数の集合を表し、$\mathbb{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$ となる。

2.1 格子 (Lattice)

格子はユークリッド空間上の subgroup であり、$\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\} \in \mathbb{R}^{m \times n}$ は線型独立のベクトルの集合であり、$\mathbf{A}$ により生成される格子は $L(\mathbf{A}) = \{\sum_{i=1}^{n} \mathbf{a}_i x_i | x_i \in \mathbb{Z}\}$ の集合となる。本書で $\mathbf{A}$ は格子 $L(\mathbf{A})$ の基底となり、m と n は格子の次元と成分となる。格子ベースの暗号では、整数格子のみを考慮することが一般的であり、本書でも $L(\mathbf{A}) \in \mathbb{Z}^m$ とする。また本書では、$n = m$ の格子について述べる。

多くの格子ベースの暗号方式は、最悪の場合の格子問題の困難性に基づいています。古典的な最短ベクトル問題 (SVP) や最近ベクトル問題 (CVP) 問題に加えて、LWE 問題 [18] またはエラーの問題を伴うリング LWE(R-LWE) 問題 [13] を使用して、最悪の場合もある程度安全な暗号アルゴリズムを構築する事もできます。

2.2 HSM とハッシュ関数

暗号、復号化、認証などの暗号化操作を実行する信頼できる環境を提供する、物理コンピューティングデバイスをハードウェアセキュリティモジュール (HSM) と呼ぶ。多くの HSM は、高度なセキュリティを実現していることを示すために FIPS 140-2 [12] や Common Criteria 等の認証を受けている。

HSM は、安全なメカニズムを利用して、通常のコンピューティング環境から分離された環境を作成することで、鍵や機微なデータの生成、保護、管理を実現する。現在 HSM は、公開鍵基盤 (PKI) やインターネットバンキングのインフラの一部などの重要なインフラで広く使用されている。なお、多くの HSM が、高い実用性と効率を維持するために接続される。

現状多くの署名システムにおいては、ハッシュ関数と非対称演算の分離が可能となる。図 1 は、典型的な利用形態であり、メッセージのハッシュ値がメッセージ管理サーバーで計算され、固定サイズのハッシュ値が HSM に送信されることを示しています。メッセージ管理サーバーと HSM 間において、固定長のダイジェストを送信する事はセキュリティシステムを作る上で効率的であり、署名生成は HSM の内部で行われるため安全性も担保できる。RSA や ECDSA などの SHA2 を使用する従来の暗号方式は、このように HSM と組み合わせることができる。

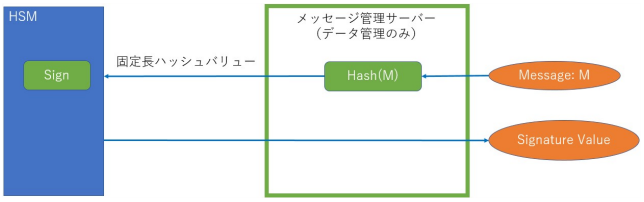

図 1 一般に、平文のハッシュをメッセージ管理サーバーで計算し、その後、HSM へ送る。署名生成は HSM 内で行う

一方で、量子攻撃に対するセキュリティを向上させるために SHA2 の代替である SHA3 [10] ファミリのハッシュ関数が、多くの PQC アルゴリズムに適用され始めている。これらの PQC アルゴリズムの多くにおいて、生成されたランダム値がメッセージと共にハッシュされる。また、一部の格子ベース暗号方式では、ランダム値は秘密とされるか、または秘密鍵に関連付けられる。それらのランダム値を生成または処理する為に十分安全な環境として、HSM を想定するのは自然な流れとなる。そして、HSM を想定した場合、ランダム値の漏洩を防ぐために (ハッシュ値のみではなく) メッセージ全体が HSM に送付され、ハッシュ関数は HSM 内で計算される。この場合、ランダム値とメッセージは、同じ Cryptographic Boundary 内に位置する事となる (詳細は 2.3 節を参照)。

図 2 では、(非常に大きいサイズである可能性もある) 可変長メッセージ全体が、HSM へ直接送付される操作を表している。このような操作を行う為には、相当大きなリソースを持つ HSM が必要となる。本書では、どの程度のリソースが新たに必要となるかを分析し、リソース増加を避ける手段について検討する。

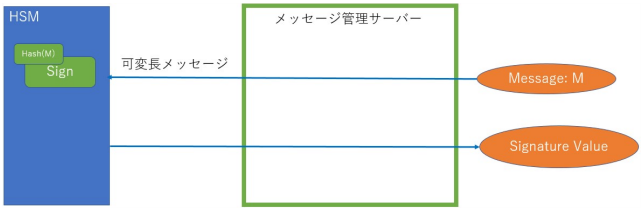

図 2　平文と秘密情報を合わせてハッシュの入力としなければいけない暗号アルゴリズムも存在する。その場合、署名操作のみでなく、秘密情報の生成も HSM で行うことが望ましい

2.3　Cryptographic Boundary

暗号化境界は FIPS 140-2 [12] で定義されている。サイバーシステム内で使用される暗号化モジュールの場合、Cryptographic Boundary は、当該暗号モジュールを構成するすべてのソフトウェア、ハードウェア、およびファームウェアを含む物理境界により確立される。暗号モジュールのセキュリティを保証するためには、Cryptographic Boundary の範囲を明確に定義することが不可欠となる。

本節では、効率的な運用を行えるような Cryptographic Boundary を構築する方法についても説明する。たとえば、図 1 または図 2 と同じようなアーキテクチャを持つシステムを、(図 3 に示すような) 単一の Cryptographic Boundary で実装・実行する場合、そこで実装されるアクセス制御は暗号化境界内の鍵を保護する機能を持つ必要があり、(人的リソースを含む) 多くのリソースを消費しうる。

一方、Cryptographic Boundary に暗号モジュールにコンポーネントが多く含まれている場合は、Cryptographic Boundary を越えた操作が増加する事も考えられ、より複雑なアクセス制御メカニズムが必要になりうえ、その点でも実装コスト等が上昇しうる。また、暗号化モジュールの一部が変更された場合、潜在的な脅威を防ぐために、どのコンポーネントを結論付けるかについての Cryptographic Boundary の再設計が必要となる。

このような状況においては、暗号モジュールに複数の Cryptographic Boundary を構築する方法が効率的であると考えられる。より正確に言うと、システム内に、図 4 に示すような Cryptographic Boundary を配置することが考えられるこの構築により、鍵に関連するプロセスは Cryptographic Boundary の内部で行われ、署名データのデータ管理は 2 つの Cryptographic Boundary の間で行われる。

このような実装の利点には、次のようなものがある。

- 鍵やそのメタデータに関するアクセス制御は、特に厳密にする必要があるが、そのような厳密なアクセス制御を行う対象を最小化できる。
- 内側の Cryptographic Boundary を跨ぐデータはほぼ固定長でありバッファ管理等が容易になる。
- システム移行において（相互運用性が確保されている事が前提ではあるが)、内側のバンダリ内と、2 つのバンダリの間の移行を別個に行う事が可能となる。

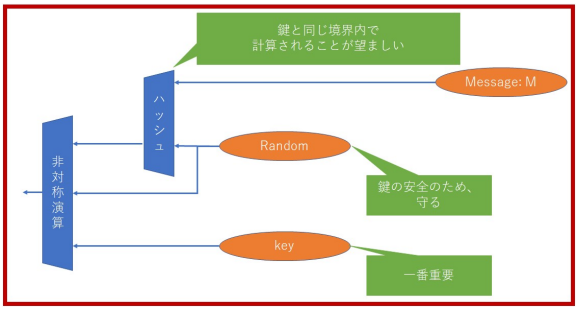

図 3　秘密情報の生成、秘密情報と平文のハッシュ計算、及び署名生成が単一の boundary 内で行われる構成

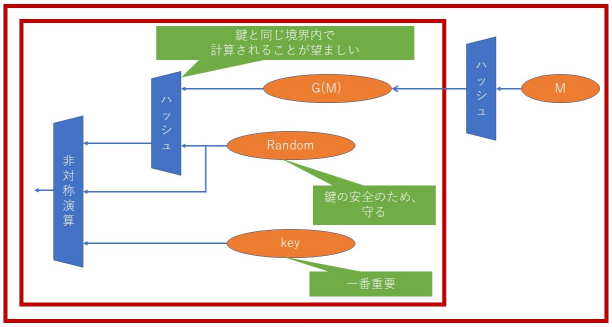

図 4　秘密情報の生成、及び署名生成が、平文に対するハッシュと異なる boundary 内で行われる構成

3 節及び 4 節では、3 つの格子ベースの署名の評価を行い、HSM に実装した場合に適切となる Cryptographic Boundary の構築方法を分析する。又、以上を踏まえた移行コストについての議論を 5 節で述べる。

3.　格子ベースの署名アルゴリズム

本節では、NIST のラウンド 2 における格子ベースの署名アルゴリズム候補である FALCON、qTESLA、CRYSTALS-DILITHIUM を、HSM に実装する場合の実用性を検討します。

3.1　FALCON

Fouque らは、NTRU 格子に基づく高速フーリエ変換を利用した、格子ベースのコンパクトシグネチャ (FALCON) を提案した [11] 。アルゴリズム 1 は、FALCON の署名生成を示す。

アルゴリズム 1 のステップ 1 において、ソル $\mathbf{r} \in \{0,1\}^{320}$ は一様ランダムに導き出され、メッセージ m でハッシュされる。$\mathbf{r}$ を一様ランダムな生成可能かつ安全な実行環境を HSM 外で提供できる場合、ステップ 1 とステップ 2 は HSM の外で行うことができる。その場合、固定長のダイジェスト c が、残りの暗号処理を行う為に HSM に入力される。このような状況下では、cryptographic boundary は図 5 の右側に示すように構築できる。それ以外の場合は、ステップ 1 とステップ 2 を HSM 内で実行し、Cryptographic Boundary 構造が図 5 の左側に表示されるように構成さ

Algorithm 1: Signature Generation of FALCON

Input : Private key sk; Message m; A bound β.
Output: The signature $sig = (\mathbf{r}, \mathbf{s})$.

$\mathbf{r} \in \{0,1\}^{320}$ uniformly;
$c = \mathbf{HashToPoint}(\mathbf{r}||\mathbf{m})$;
$\mathbf{t} = (\mathbf{FFT}(c), \mathbf{FFT}(0)) \cdot \widehat{\mathbf{B}}^{-1}$;
do
  $\mathbf{z} = \mathbf{ffSampling}_n(\mathbf{t}, \mathbf{T})$;
  $\mathbf{s} = (\mathbf{t} - \mathbf{z})\widehat{\mathbf{B}}$
while $||s|| > \beta$;
$(s_1, s_2) = \mathbf{invFFT}(\mathbf{s})$;
$\mathbf{s} = \mathbf{Compress}(s_2)$;
return $sig = (\mathbf{r}, \mathbf{s})$

れる。

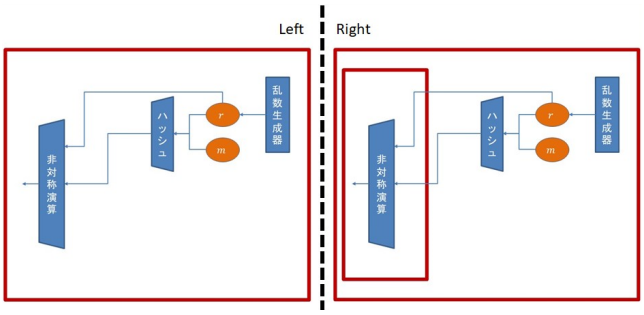

図 5 FALCON の cryptographic boundary の構成。(左:ハッシュと非対称演算が同一の boundary 内で行う。右:ハッシュと非対称演算が異なるの boundary 内で行う)

3.2 qTESLA

Akleylek らは、R-LWE に基づく qTESLA 署名アルゴリズムを提案した [2]。アルゴリズム 2 は、qTESLA の署名生成を示す。

アルゴリズム 2 のステップ 3 では、関数 $\mathbf{PRF}_2()$ はメッセージ m と秘密のデータ $\mathbf{seed}y$ とランダム値 r を入力としてハッシュ値を計算する。このハッシュ計算の入力として、Ver. 2.8 (11/08/2019) [2] 以降では、メッセージ m の代わりに、関数 $\mathbf{G}() : \{0,1\}^* \rightarrow \{0,1\}^{320}$ で計算されるメッセージ m のハッシュ値(固定長)が入力となる。Ver. 2.8 の上記変更を行わない場合、元のメッセージ m は HSM に入力する必要がある為、Cryptographic Boundary の構造は図 6 の左部分となる。Ver. 2.8 移行の場合は、Cryptographic Boundary は、6 図の右側に示すように構築する事ができる。

3.3 CRYSTALS-DILITHIUM

Ducas らは、格子における最短ベクトル問題の硬度に基づく CRYSTALS-DILITHIUM 署名アルゴリズム提案した [9]。

アルゴリズム 3 は、DILITHIUM 署名生成を示す。ここ

Algorithm 2: Signature Generation for qTESLA

Input : $sk = (s, e_1, ..., e_k, \mathbf{seed}_a, \mathbf{seed}_y, g)$; Message m.
Output: The signature $sig = (z, c')$.

counter = 1 ;
$r \in \{0,1\}^k$;
rand = $\mathbf{PRF}_2(\mathbf{seed}_y, r, \mathbf{G}(m))$;
$y = \mathbf{ySampler}$(rand, counter) ;
$a_1, ..., a_k \leftarrow \mathbf{GenA}(\mathbf{seed}_a)$;
for $i = 1, ..., k$ **do**
  $v_i = a_i y \bmod^{\pm} q$
end
$c' = \mathbf{H}(v_1, ..., v_k, \mathbf{G}(m), g)$;
$c = \{pos_list, sign_list\} \leftarrow \mathbf{Enc}(c')$;
$z = y + sc$;
if $z \notin R_{[B-S]}$ **then**
  counter = counter + 1 ;
  Restart as step 4
end
for $i = 1, ..., k$ **do**
  $w_i = v_i - e_i c \bmod^{\pm} q$;
  if $||[w_i]_L||_\infty \geq 2^{d-1} - E \vee ||w_i||_\infty \geq \lfloor q/2 \rfloor - E$ **then**
    counter = counter + 1 ;
    Restart as step 4
  end
end
return (z, c')

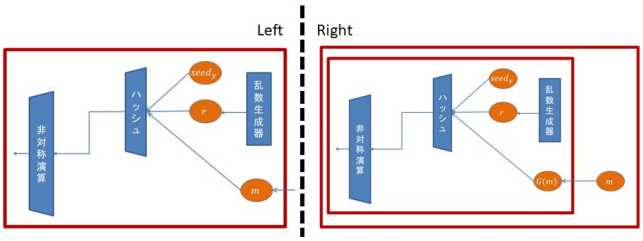

図 6 qTESLA の cryptographic boundary の構成。(左 (Ver. 2.8 より前):ハッシュと非対称演算が同一の boundary 内で行う。右 (Ver. 2.8 以降):ハッシュと非対称演算が異なるの boundary 内で行う)

で、μ は、tr とメッセージ M の結合をハッシュすることによって導き出される。また、ρ は、K と μ の結合をハッシュすることによって導出される。tr と K は秘密キーの一部であるため、ハッシュ関数 $\mathbf{CRH}()$ は HSM の内部で計算する必要があり、メッセージ M を HSM に送信する必要がある。このような状況下の Cryptographic Boundary の構造が、図 7 となる。

4. 必要とするリソースの検討

本節では、FALCON、qTESLA 及び DILITHIUM を HSM に実装する上で必要となるリソースを検討する。

本節では 3 つの格子ベースの署名アルゴリズムにおけるボトルネックとなるであろうハッシュ演算を実装し、これらの署名アルゴリズムを所定の Cryptographic Boundary

Algorithm 3: Signature Generation for Dilithium

Input : $sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$; Message M.
Output: The signature $\sigma = (\mathbf{z}, \mathbf{h}, c)$.

$\mathbf{A} \in R_q^{k \times l} = \mathbf{ExpandA}(\rho)$ ▷$\mathbf{A}$ is generated and stored in NTT Representation as $\widehat{\mathbf{A}}$;
$\mu \in \{0,1\}^{384} = \mathbf{CRH}(tr||M)$;
$k = 0$, $(\mathbf{z}, \mathbf{h}) = \perp$;
$\rho' \in \{0,1\}^{384} = \mathbf{CRH}(K||\mu)$ (or $\rho' \leftarrow \{0,1\}^{384}$ for randomized signing);
while $(z, h) = \perp$ **do**
  $\mathbf{y} \in S^l_{\gamma_1 - 1} = \mathbf{ExpandMask}(\rho', k)$;
  $\mathbf{w} = \mathbf{Ay}$ $\mathbf{w} = \mathbf{NTT}^{-1}(\widehat{\mathbf{A}} \times \mathbf{NTT}(\mathbf{y}))$;
  $\mathbf{w}_1 = \mathbf{HighBits}_q(\mathbf{w}, 2\gamma_2)$;
  $c \in B_{60} = \mathbf{H}(\mu, \mathbf{w}_1)$ ▷ Store c in NTT representation as $\widehat{c} = \mathbf{NTT}(c)$;
  $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$ ▷ Compute $c\mathbf{s}_1$ as $\mathbf{NTT}^{-1}(\widehat{c} \cdot \widehat{\mathbf{s}}_1)$;
  $(\mathbf{r}_1, \mathbf{r}_0) = \mathbf{Decompose}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$ ▷ Compute $c\mathbf{s}_2$ as $\mathbf{NTT}^{-1}(\widehat{c} \cdot \widehat{\mathbf{s}}_2)$;
  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**
    $(\mathbf{z}, \mathbf{h}) = \perp$
  end
  else
    $\mathbf{h} = \mathbf{MakeHint}_q(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$ ▷ Compute $c\mathbf{t}_0$ as $\mathbf{NTT}^{-1}(\widehat{c} \cdot \widehat{\mathbf{t}}_0)$;
    if $||c\boldsymbol{t}_0||_\infty \geq \gamma_2$ *or the# of 1's in* $\boldsymbol{h}$ *is greater than* ω **then**
      $(\mathbf{z}, \mathbf{h}) = \perp$
    end
  end
  $k = k + 1$;
end
return $\sigma = (\mathbf{z}, \mathbf{h}, c)$

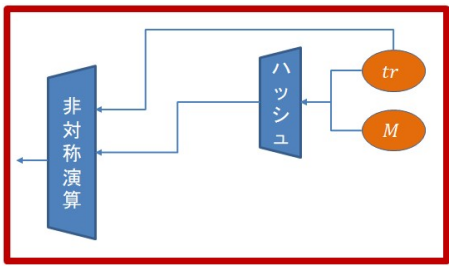

図 7 CRYSTALS-DILITHIUM の cryptographic boundary の構成。ハッシュと非対称演算が同一の boundary 内で行う

構造で実装した場合のパフォーマンスを評価する。

4.1 HSM の仕様

この論文では、セーフネット社製 HSM である ProtectServer External 2 (PSE-2) を利用して実験を行った。また、ファームウェアは、ProtectServer Client software version5.6 を利用した。 PSE-2 は FIPs 140-2 レベル 3 を満たし、また Function Module ユニットにより、任意の機能を実装し、実行することが可能となる。

な お 、PSE-2 は 、PKCS#11、openSSL、JCProv、JCA/JCE、および MS CAPI/CNG で定義されているさまざまなインターフェースがサポートしている。また、C、C#、および Java を用いた開発をサポートする為の Protect Server toolkit (PTK) が用意されている。

4.2 実験結果

本書ではメッセージが HSM に直接送信されるような構成であり、署名生成システムに構築された Cryptographic Boundary が 1 つだけの場合 (図 3)、この構造をタイプ”A”と呼ぶ。

一方でタイプ”B”では、固定長メッセージダイジェストのみが HSM に転送されることを表し、メッセージからハッシュ値を計算する為には別の Cryptographic Boundary が存在することとなる (図 4)。表 2 は、FALCON、qTESLA および CRYSTALS-DILITHIUM について、異なる Cryptographic Boundary に設置した HSM で実行した時のハッシュ計算の計測結果となる。実行時間はミリ秒単位で記載した。また、メッセージのサイズは、キロバイ (K) またはメガバイト (M) で記載した。

FALCON の場合、ハッシュ値が HSM の Cryptographic Boundary の外部で計算可能であり、その場合 HSM 内のハッシュ操作の時間コストは 0 になる（タイプ B)。一方、表のように、境界タイプ A の場合、メッセージ・サイズが大きくなるにつれ、時間時間は線形に増加する。メッセージサイズが 10M の場合、単一メッセージに対するハッシュ計算の実行時間は約 11667.19ms(約 11.7s) となる。

qTESLA でも、境界タイプ A の場合はメッセージサイズの増加に伴い実行時間は線形に増加する。Ver. 2.8 以降の qTESLA の場合は、固定サイズのハッシュ値が HSM に送られる形式となる為、qTESLA の実験結果の 2 行目に示すように、ファイルサイズによらず実行時間はほぼ同じとなる。

CRYSTALS-DILITHIUM の場合、メッセージのハッシュ値は、メッセージと秘密鍵の結合のハッシュ演算により導出され、署名生成のすべての操作が同じ境界内で実行される為、メッセージサイズの拡張に伴って時間コストが線形に上昇する。

5. 各 Cryptographic Boundary を採用した場合の移行コスト

現状における、HSM を利用した RSA(with sha2) や ECDSA(with sha2) の署名システムは、Type A のバンダリを採用している。それらの署名システムを、Type A 又は Type B の構成の格子ベース署名に移行するためのコストについて述べる。

5.1 Type A への移行コスト

Type B に移行するためには、Cryptographic Boundary

表 2　三つの格子ベース署名を、異なる cryptographic boundary 内の HSM で実行したときの、HSM 内のハッシュ計算の実行時間

Scheme	Boundary Type	Time (millisecond)				
		1k	10k	100k	1M	10M
FALCON	A	33.36	38.08	142.26	1240.59	11667.19
	B	0				
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
DILITHIUM	A	34.67	45.79	156.19	1351.40	12727.83

を再定義する必要がある。

Type B では Cryptographic Boundary により多くの構成要素が含まれる事になり、バンダリで行うアクセスコントロールのためのメカニズムが複雑になる事が予想される。加えて Cryptographic Boundary を跨ぐオペレーションも増加するため、それらのオペレーションに耐えうるように可用性を向上させる必要も生じる。これらの変更には、脅威分析や脅威モデルの再定義、人的運用の再定義及び大幅増加等が発生しうる。

加えて、4 節で述べたように、暗号鍵の管理を行うバンダリ内で多くのハッシュ計算を行う必要も生じるため、大きなファイルサイズのデータに対する署名を行うシステムの場合は、暗号鍵管理デバイスの大幅な高性能化が必要となる。

5.2　Type B への移行コスト

Type A に移行する場合においては、RSA や ECDSA で用いたものと同一の Cryptographic Boundary を利用する事が想定される。この場合においては、新たな暗号アルゴリズムのオブジェクト ID 等をサポートする必要があるものの、2 つのバンダリ間の構成の変更は非常に限定的となる。

移行においても、図 8 のように内側のバンダリ内で呼び出す鍵管理モジュールをスイッチする事が可能であれば、多くのコストを削減する事が可能であり、また人的運用の変更も限定的となる事が想定される。

もっとも、そのような移行を行うためには、新旧の鍵管理モジュールに対する、データ処理部の相互運用性が必要となり、API の統一等が必要となる。

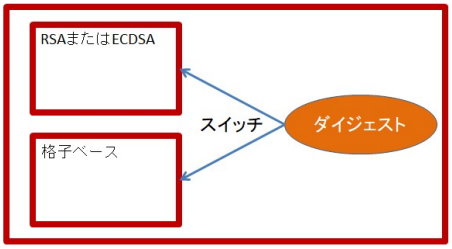

図 8　旧来の暗号の boundary と格子暗号の boundary の呼び出しを、外側の boundary 内で切り換えることができる仕組み

6.　結論

本稿では、NIST のプロジェクトの候補の 1 つである格子ベースの暗号を、ハードウェアセキュリティモジュール (HSM) に実装する場合の実用性について検討した。NIST のプロジェクトのラウンド 2 から選択された 3 つの格子ベースの署名アルゴリズムの特徴を説明し、HSM の内外で処理される各実装形態におけるハッシュ関数の性能を比較することによって、幾つかの実装における実用性の限界を指摘した。また、HSM を利用する場合に、PQC の実用性を向上させるアプローチを提案した。我々の結果は、(PQC の標準化に際して行われるであろう) 安全性証明と特許調査について、その検討を行う範囲を定義するのに役立つものと考えられる。

本書では、格子ベースの署名アルゴリズムのうち、大規模なファイルを扱う上でボトルネックとなるハッシュ部分のみを比較した。ボトルネックを解消した上で、署名生成のために必要な総処理時間を比較し、新たなボトルネックの発見及び解決が今後の課題となる。

参考文献

[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 - CRYPTO89 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] Junting Xiao and Tadahiko Ito. "Constructing the Cryptographic Boundaries for Lattice-based Cryptography on Hardware Security Module." In IACR Cryptology ePrint Archive, Report 2020/990, 2020.

[24] 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.

[25] 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/.