通信データの特徴を用いたRing-LWE問題に基づく鍵交換プロトコルの通信量削減方法の検討

古川 智也[1]　猪俣 敦夫[2,1]　新井 イスマイル[3]　藤川 和利[3]

概要：量子計算機の実現により RSA，Diffie-Hellman 鍵共有などの現在用いられている公開鍵暗号の脆弱化が問題となっている．このことから量子計算機に対抗できるポスト量子暗号が研究されており，その一つに Ring-LWE 問題に基づいた鍵交換プロトコルが提案されている．しかし，このプロトコルは鍵交換時の通信量が既存の鍵共有プロトコルの 2 倍以上になる．このため，IoT などの多くの機器を用いる環境下で用いる場合，通信量が爆発的に増加してしまう．そこで，本研究では鍵交換プロトコルにおける通信量の削減を目的とし，通信データの特徴を分析し，特徴に基づいた符号化法を提案し，圧縮率を評価した．その結果全体の 2%削減した．

キーワード：セキュリティプロトコル，鍵共有，Ring-LWE，ハフマン符号

An overhead reduction method of key exchage protocol from the Ring-LWE problem with feature of communication data

TOMOYA FURUKAWA[1]　ATSUO INOMATA[2,1]　ISMAIL ARAI[3]　KAZUTOSHI FUZIKAWA[3]

1. はじめに

近年，量子計算機の実現による Diffie-Hellman 鍵共有などの公開鍵暗号プロトコルの危殆化が問題となっている．この問題に対し，量子計算機にも安全であるポスト量子暗号の研究が盛んに行われており，既存の公開鍵暗号プロトコルを置き換えるポスト量子暗号を用いたプロトコルが提案されている．ポスト量子暗号を用いたプロトコルの一つとして，Erdem らはポスト量子暗号の一つである Ring-LWE を用いた鍵共有プロトコルを提案している [1]．しかし，このプロトコルは鍵共有を行う際の通信データサイズが既存の鍵共有プロトコルに比べ 2 倍以上であり，IoT などの多くの機器を用いる環境下では通信量が爆発的に増加してしまい，実用的でないと考えられる．

そこで本稿では通信データサイズの縮小を目的として，通信データの特徴を用いたデータの符号化方法を提案する．さらに，提案手法を用いた時の圧縮率を評価する．

本稿の構成を以下に示す．まず，2 節で本研究で用いる鍵共有プロトコルの説明を行う．3 節では送信データの特徴について説明し，その特徴を利用した符号化方法を提案する．4 節では提案手法を用いた際の圧縮率，符号，復号化の処理時間を評価し，最後に総括を行う．

2. 関連研究

この節では本研究の関連研究について説明する．まずはじめに提案手法を適応する鍵共有プロトコルの安全性の根拠となる Ring-LWE 問題について説明し，次に鍵共有プロトコルの流れについて説明し，最後に一般的な符号化方法について説明する．

1 奈良先端科学技術大学院大学　情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
2 東京電機大学
Tokyo Denki University
3 奈良先端科学技術大学院大学総合情報基盤センター
Information Intiative Center, Nara Institute of Science and Technology

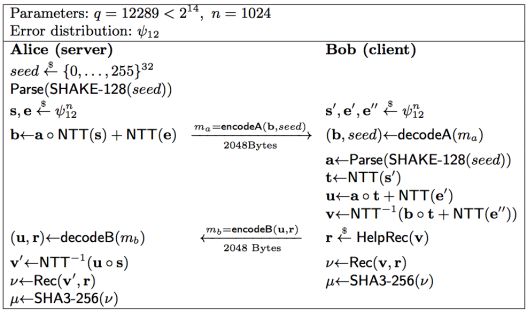

図 1 鍵共有プロトコルの流れ [1]

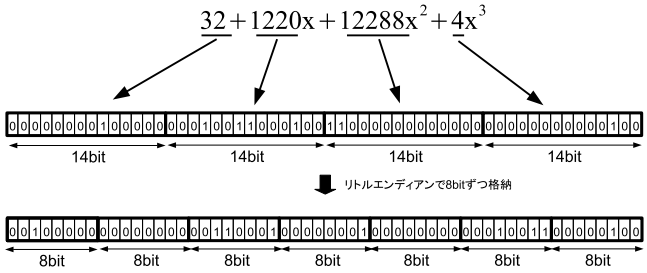

図 2 多項式 $32 + 1220x + 12288x^2 + 4x^3$ のエンコード例

2.1 Ring-LWE 問題

整数集合を $\mathbb{Z}$, q を整数, 多項式環を $R_q = \mathbb{Z}[x]/(x^n+1)$, R_q 上のエラー分布を $\mathcal{X}^n$ とする．また, $a, s \in R_q$, $e \in \mathcal{X}^n$, $b = as + e$ とする．このとき (a, b) から s を求める問題が量子計算機を用いても困難であると考えられており，この問題を Ring-LWE 問題と呼ぶ.

2.2 Ring-LWE 問題を用いた鍵共有プロトコル

Ring-LWE 問題を用いた鍵共有プロトコルについて説明する．このプロトコルは Perkit らが提案した認証なし格子暗号ベースの鍵共有プロトコル [3] を基に提案されている．このプロトコルでは法を $q = 12289$，多項式環の次数を $n = 1024$ に設定しており，これにより 128bit security level を保証していることが示されている．プロトコルの流れを図 1 に示す．サーバ側はまず公開パラメータ $\mathbf{a} \in R_q$ をランダムに生成する．$\mathbf{a}$ を生成するために用いる $seed$ は 32byte でランダム生成され，これを用いて FIPS-202[5] に公開されているハッシュ関数でハッシュ値を求める．さらにこのハッシュ値を分割して a を生成する．次に秘密鍵の $\mathbf{s}$，エラー多項式の $\mathbf{e}$ を生成し，$\mathbf{a}, \mathbf{s}, \mathbf{e}$ を用いて公開鍵 b. 多項式演算を高速化するために NTT(Number Theoretic Transforms)[4] を用いている.

次にの encodeA について説明する．encodeA() では多項式 $\mathbf{b}$ の 14bit の係数と seed を符号なし 8bit に変換し, 一つの配列に格納され送信する．このとき，リトルエンディアン方式で配列に格納される．多項式 $32+1220x+12288x^2+4x^3$ を 8bit の配列にエンコードした例を図 2 に示す．図 2 から多項式中の係数を二進数で表すと，

$$32_2 = 00000000100000, 1220_2 = 00010011000100$$
$$12288_2 = 11000000000000, 4_2 = 00000000000100$$

となる．これをリトルエンディアンで 8bit ずつ格納していくので 1 番目の配列には一つ目の係数 32 の 1bit 目から 8bit 目が格納される．次に 2 番目の配列には 1bit 目から係数 32 の 7bit 目から 14bit 目が格納され，残りの 2bit に二つ目のである係数 1220 の 1bit 目と 2bit 目が格納される．これを繰り返すことにより，図 1 下の char 型配列が得られる.

次にクライアント側でも同様に公開パラメータ a を求め，秘密鍵 s'，エラー多項式 e', e'' を求める．そして，$\mathbf{u}, \mathbf{v}$ を求め，共有パラメータを求めるための値 $\mathbf{r}$ を生成する.

次に $\mathbf{u}, \mathbf{r}$ を encodeB により，一つの配列とし，サーバ側に送る．そして，お互いに共有パラメータ ν を求める．最後に SHA3-256(ν) でハッシュ値 μ を求め，この μ を共通鍵として用いる.

2.3 一般的な符号化法

符号化の手法としてハフマン符号とランレングス符号がある．ハフマン符号は符号化する値の出現頻度からハフマン木を構成することにより，符号化を行う．ハフマン符号は出現頻度が高い値には小さい符号長を与え，出現頻度の低い値には大きい符号長を与える特徴がある．このため，出現頻度に偏りがあるほど，効果が大きく，反対に出現頻度に偏りがなければ効果が小さい．通信する際にハフマン符号を用いる方法は二つあり，符号化するたびに出現頻度を求めてハフマン木を生成する動的な方法と，事前に出現頻度を統計的に求めてハフマン木を構築して符号化テーブルを事前に作っておき，お互いに共有しておく静的な方法がある．動的な方法の場合圧縮率は静的な方法よりも良いが，復号のために符号化テーブルを送る必要があるため通信量が大きい．これに比べ，静的な方法は圧縮率は動的なものに劣るが，通信量は小さい．伊藤らはセンサーネットワークでのハフマン符号の利用において共有の符号化テーブルを用いることで，動的な方法に比べ圧縮率は 5%程度低下するが，通信量を大幅に抑えることを示している [6].

ランレングス符号は「0000000」を「07」と符号化するように符号化する値とその値が連続している数を一つの符号とする符号化法である．このことから，連続している値が多ければ多いほど効果がある．反対に値が連続していなければ符号長が符号化する前よりも大きくなる可能性がある.

3. 提案手法

本提案では後述する通信データの特徴から値の出現頻度に偏りが出るため，ハフマン符号を利用する．しかし，鍵共有をするたびにハフマン木を生成して符号化を行う場

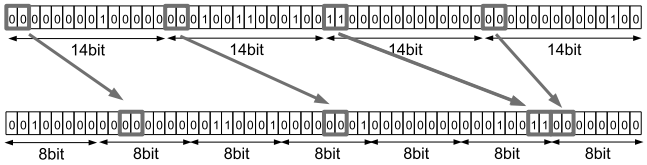

図 3 多項式 $32 + 1220x + 12288x^2 + 4x^3$ の制限ビットの移動

合，通信データに符号化テーブルを付加する必要があるため，データサイズがあまり小さくなる可能性がある．そこで事前に通信データに出現する値の頻度を調べておき，その頻度を元にした符号化テーブルを構築しておく．さらにサーバ，クライアント間で事前に符号化テーブルを共有しておくことにより通信データに符号化テーブルを付加することを防ぐ．

まず始めに通信データに見られる特徴について説明する．次に通信データの特徴を用いた変換テーブルの作成方法について説明する．

3.1 メッセージにおける特徴

2 節で説明したようにサーバーとクライアント間で通信するデータには多項式が含まれている．この多項式の係数の特徴について説明する．以降 a の二進数表記を a_2 として表す．この多項式の係数は法が $q = 12289$ であるため，14bit で構成されている．さらに，12288 の二進数表記が $12288_2 = 11000000000000$ であることから，係数の 13, 14bit が「11」となる値は 12288 と 12289 の二つしかないことがわかる．係数としてとれる値は 0 から 12289 の全部で 12290 であるため，係数の 13, 14bit 目が「11」となる確率は $\frac{2}{12290} \approx 0.02$ となり，13, 14bit 目が「11」となる確率が非常に小さいことがわかる．以降，この 13, 14bit 目のビットを制限ビットと呼ぶ．

次にサーバからクライアントへ送る通信データ m_a(図 1 中の m_a) の特徴について説明する．図 1 から通信データ m_a は多項式を 8bit 間隔の配列に格納して送られる．この時制限ビットは図 3 のように移動する．

図 3 から係数 32 の制限ビットは二番目の配列の 5, 6bit 目に，32 の制限ビットは 2 番目の配列の 5, 6bit 目に，1220 の制限ビットは 4 番目の配列の 5, 6bit 目に，12288 の制限ビットは 6 番目の配列の 1, 2bit 目に，制限ビットを持つことがわかる．さらに 8 番目の配列は 1 番目の配列と同じ構成になるため，通信データの配列の番号を 1 から始めるとすると，図 3 から次の特徴を持つことがわかる．

(1) 2+7n 番目の配列に含まれる値は 5，6bit 目が「11」となる確率が小さい．
(2) 4+7n 番目の配列に含まれる値は 3，4bit 目が「11」となる確率が小さい．
(3) 6+7n 番目の配列に含まれる値は 1，2bit 目が「11」となる確率が小さい．

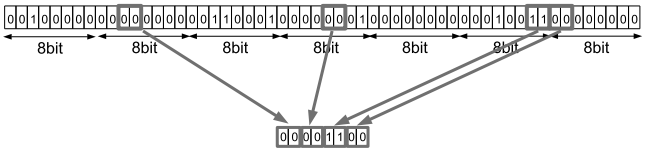

図 4 符号化用の値の構築例

表 1 マシンのスペック

OS	OS X Yosimite 10.10.5
メモリ	8GB
CPU	Core i7 2.3GHz

(4) 7+7n 番目の配列に含まれる値は 7，8bit 目が「11」となる確率が小さい．

この特徴を用いて符号化を行う．

3.2 制限ビットを用いたハフマン符号化法

出現する符号が各配列により頻度が異なるためハフマン符号による圧縮が可能であると考えられる．しかし，各特徴に合わせて変換テーブルを構築すると変換テーブルが多くなってしまい，冗長であると考えられる．そこで，各配列の制限ビットを 8bit ずつまとめて圧縮する．図 3 から符号化用の値を作る例を図 4 に示す．図 4 より，4 つの配列から制限ビットを抜き出し，8bit の値を作っていることがわかる．この値は制限ビットから作っているため，先に上げた 1, 2bit 目または 3, 4bit 目または 5, 6bit 目または 7, 8bit 目に「11」をとる確率が小さい．このことから出現する値により偏りが現れるため，圧縮しやすいと考えられる．4 つの係数から一つの符号を作れるため，符号化用の値は 256 個作ることができる．この 256 個の値の出現頻度からハフマン木を構築し，符号化用テーブルを構築する．

4. 評価

この節では提案手法の評価を行う．まず通信データのサンプリングを行い，3.2 節で示した通信データの特徴が表れていることを示す．最後に提案手法と制限ビットをまとめずにハフマン符号化をした場合との比較を行う．なお評価を行ったマシンのスペックを表 1 に示す．

4.1 通信データの特徴の検証

通信データの各配列での値の出現頻度を示すことにより，各配列番号により特徴があることを示す．通信データを 10000 回サンプリングし，各配列番号での出現頻度を求めた．その結果を図 5 から図 8 に示す．

図 5 より 2+7n 番目の配列は 48 等の 5, 6bit 目が「11」の値が，図 6 より，4+7n 番目の配列は 12 等の 3, 4bit 目が「11」の値が，図 7 より 6+7n 番目の配列は 3 等の 1, 2bit 目が「11」の値が，図 8 より，7+7n 番目の配列は 192 等の 5, 6bit めが「11」の値がほとんど出現しないことがわか

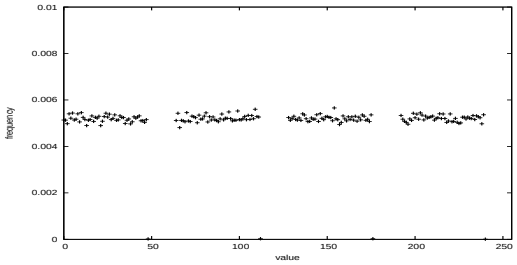

図 5 2+7n 番目の配列

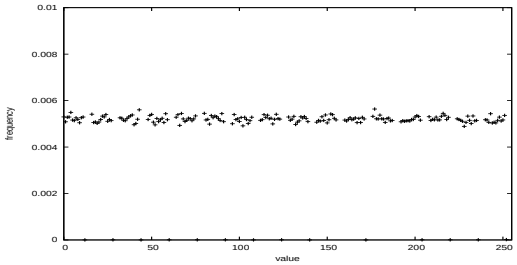

図 6 4+7n 番目の配列

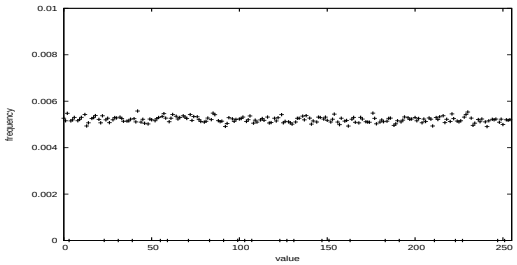

図 7 6+7n 番目の配列

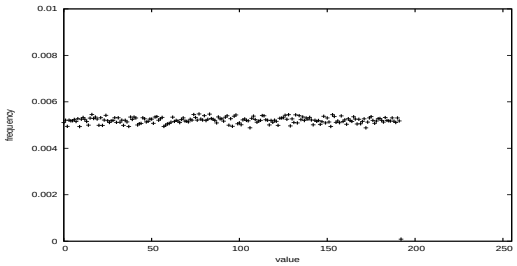

図 8 7+7n 番目の配列

る．以上のことから各配列によって値の出現頻度が異なることがわかる．

4.2 提案手法による圧縮率の評価

提案手法を用いた際の圧縮率を評価する．比較対象として制限ビットをまとめずにハフマン符号化したものを用意する．各手法について図 1 の通信データ m_a をランダムに 1000 回サンプリングし，通信データ中の多項式を含む部分を符号化し圧縮率の平均を求めた．結果を表 2 に示す．なお，符号化前の m_a は 2048byte である．

表 2 より，提案手法は全体の 2%を削減したのに対し，ハフマン符号のみでは 1bit 程度しか削減することができなかった．この理由として各配列で出現率が偏っていても特徴を持たない配列は一様にすべての値が出現してしまうため，配列全体としての出現頻度の偏りが小さかったことが考えられる．

表 2 圧縮率の評価

手法	符号化後 (byte)	削減データ (byte)
提案	1997.86	50.14
ハフマンのみ	2047.89	0.125

5. まとめ

本稿では Ring-LWE を用いた鍵共有プロトコルの通信データの特徴を利用する符号化方法を提案し，全体の 2%程度削減することができた．今後の課題として符号，復号化にかかる処理時間を計測し，通信データの削減と処理時間にどの程度のトレードオフがあるのかを考える必要がある．本提案では符号化テーブルを用いて符号化を行う前に事前に制限ビットをまとめる処理，制限ビットが抜けた部分へのビットのシフト処理等があるため，処理時間に見合った圧縮率にならない可能性がある．また，エンコードした後のデータを用いたがエンコード前のデータにも注目して符号化を考えることができる．例えば，エンコード前であるなら必ず 13, 14bit 目の値が制限ビットになるため符号化のアルゴリズムが複雑にならない．最後に出現頻度の小さい「11」を「10」や「01」とスワップすることにより，「00」や「11」などの連続した値が出やすくできると考えられる．こうすることで，ランレングス符号の適用も考えることができる．

参考文献

[1] Erdem Akin, Leo Ducas, Thomas Poppelmann, and Peter Scwabe. Post-quantum key exchange - a new hope. IACR Cryptoloy ePrint Archive, 2015. http://eprint.iacr.org/2015/1092.pdf

[2] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Proceedings of EUROCRYPT 2010, Lecture Notes in Computer Science, vol. 6110, pp. 123, 2010.

[3] Chris Peikert. Latice cryptography for the Internet. In Michele Mosca, editor, Post-Quantum Cryptography, volume8772 of LNCS, pp. 197-219. Springer, 2014.

[4] Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. Compact Ring-LWE Cryptoprocessor. in Cryptographic Hardware and Embedded Systems CHES2014, vol.8731 of LNCS, pp.371-391, Springer, 2015.

[5] National Institute of Standards and Technology. FIPS PUB 202-SHA-3 standrd: Permutation-basd hash and extendable-output function, 2015. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

[6] 観測データの相関を利用した符号化テーブルの削減方法. 研究報告モバイルコンピューティングとユビキタス通信, vol. 24, pp.1 - 4. 2012.