エラー正規化処理なしの完全準同型暗号の利用可能性 : 計算可能回数および要因の調査と学力試験への応用

池田, 響

(Citation)
課題研究優秀論文集,2023:318-368

(Issue Date)
2024-01

(Resource Type)
departmental bulletin paper

(Version)
Version of Record

(JaLCDOI)
https://doi.org/10.24546/0100485453

(URL)
https://hdl.handle.net/20.500.14094/0100485453

2023年度　卒業研究最終論文

エラー正規化処理なしの完全準同型暗号の利用可能性

-計算可能回数および要因の調査と学力試験への応用-

神戸大学附属中等教育学校10回生

6年1組03番

池田　響

（指導教員　米田　貴　/　中田　雅之）

2023 年度　卒業研究最終論文

エラー正規化処理なしの完全準同型暗号の利用可能性

-計算可能回数および要因の調査と学力試験への応用-

神戸大学附属中等教育学校 10 回生　6 年 1 組 03 番　池田　響

（指導教員　米田　貴　/　中田　雅之）

キーワード：完全準同型暗号、秘匿計算技術、Bootstrap、学力試験

要旨

完全準同型暗号とは復号せずに加算および乗算が計算可能な特殊な暗号である。クラウドコンピューティングにおいて、復号せずに計算できることは高度な安全性を保障することになる。しかし、計算速度の課題から社会での利用が進んでいない。完全準同型暗号の処理過程の中で、特に計算速度に影響するのは Bootstrap とよばれるエラー正規化処理である。多くの先行研究では、アルゴリズムの改良や数学的な操作の工夫から、Bootstrap にかかる時間を短縮しようとしていた。しかし、本研究では Bootstrap の必要性を疑い、Bootstrap なしの場合における暗号文の計算可能回数と、それを決定する要因について、加算と乗算それぞれに対し、実験・考察した。また、それらの結果をもとに、一例として学力試験への応用を考えた。実験の結果、法 q、格子 $m \times m$、エラーベクトルの要素 σ において、加算では $\frac{q}{4m\sigma}$ 回未満、乗算では $\log_{(n+1)l} \frac{q}{8m\sigma}$ 回未満の計算が可能という結果が得られた。要因の調査より、法は計算可能回数と計算の実行範囲を決定するために選択できる一方、格子はなるべく小さくとる必要があることがわかった。以上をもとに、適切な法と格子を選べば、学力試験における平均値と分散を計算するアルゴリズムの実装が行えることが示された。また、計算可能回数で満たすべき法の値に比べて、計算実行範囲を満たすために必要な法の値の方が大きいことがわかった。完全準同型暗号は Bootstrap 処理なしでも、法や格子の値の設定によって小規模なデータの分析においては十分に利用が可能であるといえる。

2023 GRADUATION THESIS

エラー正規化処理なしの完全準同型暗号の利用可能性

Possibility of Fully Homomorphic Encryption without Using Bootstrap Technique

Kobe University Secondary School
池田響
Hibiki Ikeda

Abstract

Secure computation technology using FHE (Fully Homomorphic Encryption) has not yet been widely used in society because of its calculation speed. Bootstrap, the calculation normalization process, is the main cause of the problem. Therefore, this research focuses on the case to avoid using bootstrap and allow an error to occur. I implemented GSW-style FHE in Python and investigated the number of computable times and related properties.

As a result, the modulus and lattice were found to affect the number of calculations. Therefore, assuming the case with the least number of calculations, the following equation was obtained. In the case of the $n \times m$ lattice, the modulo q and element σ of the private key, it can be calculated less than $\frac{q}{4m\sigma}$ times in addition process and $\log_{(n+1)l} \frac{q}{8m\sigma}$ times in multiplication process. From all results, by limiting the use of FHE to the small scale analysis, which is calculation of averages and standard deviations for small numbers of people, it hypothesized that high speed FHE without using Bootstrap technique is at a level where it can be implemented in society.

Keywords: GSW-style FHE, Secure Computation, Bootstrap, Post-Quantum Cryptography

目次

第 1 章　　序論

第 1 節　社会における暗号の利用

電子メールから宇宙開発にいたるまで、現代社会で使われる通信機器において暗号化は不可欠である。暗号化されていなければ、通信におけるデータを第三者が流出、改ざんすることが可能となってしまう。あらゆる家電製品から軍事機器までもが、インターネットを介して繋がる現代社会において、通信情報の流出・改ざんは経済・社会に大きな影響を与えるリスクがある。

現在は様々な種類の暗号が用途に合わせて開発されている。その暗号の種類は、主に公開鍵暗号方式と共通鍵暗号方式の 2 つに分けられる。(図 1) 公開鍵暗号方式には電子署名などで広く利用されている RSA 暗号や、IC カードやビットコインなどで利用されている楕円曲線暗号がある。一般に、公開鍵暗号方式では暗号化に使う公開鍵と復号で使う秘密鍵が 1：1 で対応している。また、公開鍵から秘密鍵を求めることが難しいということが知られている。これによって、送信者は受信者側の公開鍵（誰でも閲覧できる）を用いて平文の暗号化をおこない、秘密鍵（受信者のみが所持）を用いて受信者のみがその暗号文を復号できる。よって、受信者のみが平文を得ることが可能となる。共通鍵暗号方式では AES などが利用されている。共通鍵暗号方式では、暗号化と復号に使う鍵が同一のもので、受信・送信のやり取りを行う相手と 1：1 で鍵を交換することによって安全な通信経路を確保する。

このように、暗号にはさまざまな種類がある。そして、それらの種類は、誰と通信をするのか、どの程度の安全性を必要とする通信なのか、など目的・用途に合わせて選択される。さらに、その時に使われているコンピュータがもつ計算速度、暗号解読の効率的なアルゴリズムの研究成果など、時代によっても使われる暗号は変化する。ゆえに、作成された暗号が、革新的なアルゴリズムの発見や、計算速度の急な向上によって利用できなくなることもある。そのため、社会では日々、暗号解読や新たな暗号の作成が行われている。

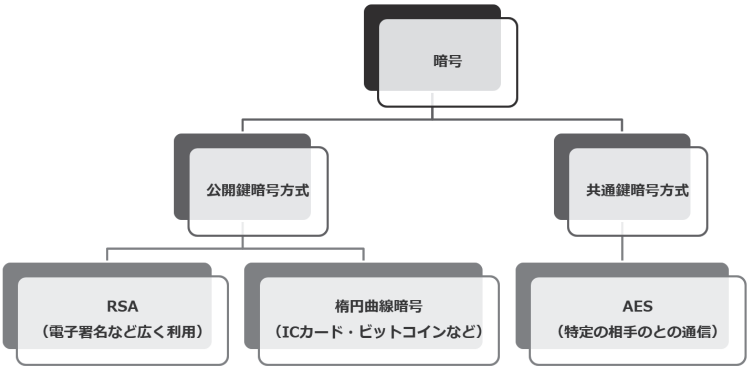


図 1　暗号の分類と利用目的

第 2 節　秘匿計算およびエラー正規化処理の先行研究

第 1 項　秘匿計算の研究動向

本章では、2021 年度の調査をもとに現在の秘匿計算技術に関係する傾向を分析する。2021 度の個人研究 [1] によると、完全準同型暗号を含む秘匿計算技術の研究では下の 3 つの図のような特徴があげられる (図 2〜図 4)。以

下の調査は、日本電信電話株式会社 (NTT)[2]、千葉大学と NTT の共同研究 [3]、ZAMA[4]、Crypto Experts[5]、富士通株式会社 [6]、三菱電機株式会社 [7]、IBM[8]、EAGLY 社 [9]、Microsoft[10]、Acompany[11] の 10 社を対象に実施し、独自の観点別に分類したものである。

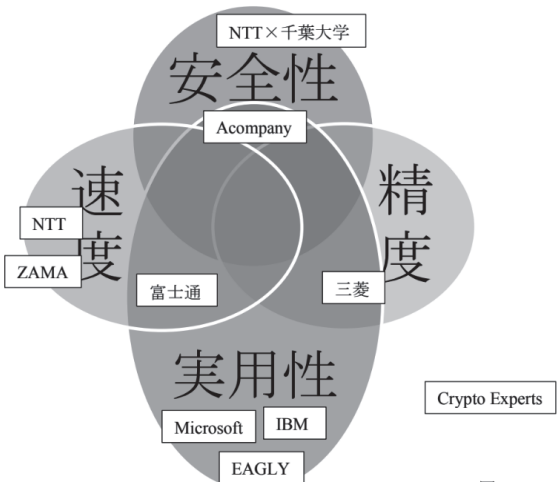


図 2　観点 1　それぞれの会社が解決しようとしている課題

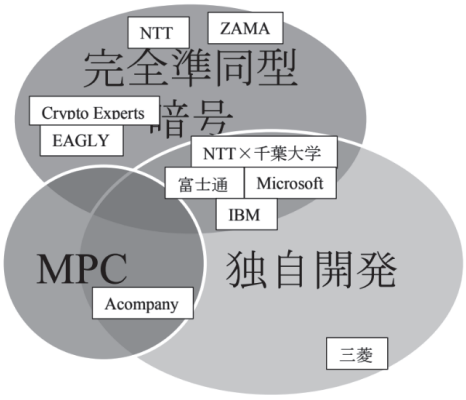


図 3　観点 2　秘匿計算に使っている暗号の種類

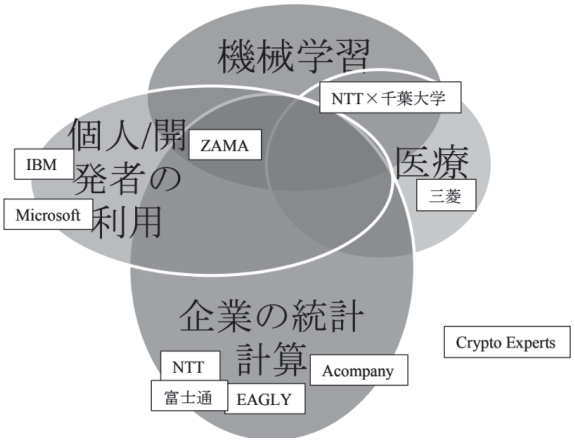


図 4　観点 3　研究目的

図 2 より、近年の秘匿計算技術は速度、実用性に重きを置いた研究がなされていることがわかる。また、観点 2 より 10 団体の中で秘匿計算技術として完全準同型暗号が主流となっているともいえる。NTT× 千葉大学や富士

通、Microsoft、IBM などは、それぞれ目的に応じて完全準同型暗号を独自に発展させて開発することで、暗号の実用性を高める工夫を行っていた。Acompany においても 、MPC+秘密分散法とすることで 、QuickMPC という新たな暗号アルゴリズムを構築していた。このことから、秘匿計算技術を実際に活用するためには、秘匿計算を利用する環境に応じた最適なアルゴリズムの構築が必要といえる。対して NTT、ZAMA、Crypto Experts、EAGLY 社は、新たな手法を使わずに完全準同型暗号をそのまま用いている。NTT については調査した情報が 2013 年ごろの完全準同型暗号の開発段階のものであったためと考えられるが、ZAMA は新たな暗号アルゴリズムを目的に応じて作成するのではなく、単純な計算処理を超高速化することによって、それらを組み合わせて複雑な処理を高速で行うことを目指しているため、完全準同型暗号をそのまま用いていると考える。Crypto Experts については、耐量子暗号の開発の中で秘匿計算を用いているため、研究主旨とことなる秘匿計算技術に工夫を施す必要がなかったのではないかと考える。EAGLY 社は、あるものを使って、その組み合わせと最適化によって、企業向けのソフトウェアを提供する会社であるためだといえる。

続いて、観点 3 について、Microsoft と Crypto Experts を除いたすべての団体が企業・医療施設向けの開発である。暗号の利用は個人で行うのではなくアプリケーションを作ったり、企業間での大規模な情報処理を行ったりするためであることから、必然ともいえる。特に、統計処理に重きが置かれていることも特徴的である。これは近年ビッグデータを活用したデータサイエンスが注目されていることが大きく影響している。より安全に企業間で大規模なデータの共有を行っていくために、秘匿計算システムが必要不可欠になってくるであろう、と多くの暗号開発団体が考えたためであるといえる。AI の開発も同様に、千葉大学と NTT の共同研究の事例より、ビッグデータを解析する上で必要になってくるためであるからといえる。従って、秘匿計算を行う目的はビッグデータの管理・分析のためであるといえる。Crypto Experts においては前段落でも示したように、耐量子暗号の開発の一部として用いるため、分類から外れる結果になっている。

最後に国内外で 10 団体を比較する。近年の国内での秘匿計算技術はビッグデータを活用するための実用に向かった研究がなされている。また、団体によって独自のアプローチでそれぞれの使用する暗号のデメリットをカバーするように細かく目的を設定し、それに応じた最適なアルゴリズムを構築している。たいして海外の研究団体については暗号そのものの処理性能向上をめざし、ツールキットの作成や エラー正規化処理などの 1 単位での速度向上が行われている。

第 2 項　エラー正規化処理の研究動向

前項より、近年の秘匿計算技術は完全準同型暗号を用いるものが主流であること、また、ビッグデータ等の大規模データを解析しようと実用に向けた取り組みがなされていること、処理性能の向上を目指した研究がなされていることがわかった。本研究では、特に処理性能を決定するエラー正規化処理（Bootstrap）を省いたケースに着目をした。前項からも分かるように、暗号文のノイズを削減する Bootstrap 処理は完全準同型暗号の処理の中で計算量が多いことが知られているからである。

Bootstrap は Gentry によって 2009 年に提案された [12]。Bootstrap を用いることによって、暗号文計算によるノイズを緩和し、暗号文のみでの加算・乗算が実質的に無制限でおこなえるようになった。しかし、一方で Bootstrap 処理では、その特性から処理に時間がかかるという課題があった。その後、改良がなされ MSB[13] をもとに現在主流である CKKS[14] が提案された。現在では CKKS における Bootstrap 処理時間短縮を目指した研究 [15] などがなされている。このようにして完全準同型暗号における Bootstrap 処理は、その処理の時間的課題解決にむけて研究がなされてきた。そして、それらどの研究も、Bootstrap 処理の実行時間短縮を目指した研究がなされている。完全準同型暗号はクラウドコンピューティングに関して高度な安全性を保証することから、これからの社会利用に期待の大きい暗号である。しかし、暗号の構造によって計算処理時間が長くかかるという課題のため一般的な社会への利用までは進んでいない。

以上の現状をふまえ、本研究では、本研究では、Bootstrap そのものの存在意義について疑問を持ち、Bootstrap を使用しない場合に着目して研究を進めた。計算制限下の本暗号についても、十分に利用する価値がある計算回

数を確保できるのではないか、Bootstrap を回避することによって時間的課題を回避できるのではないか、と考えたためである。

また研究で、準同形暗号を 1 から実装するために、改良の加えられていない初期の完全準同型暗号である GSW 式完全準同型暗号を用いて研究を行った。この暗号は、耐量子暗号の性質を備えた秘匿計算技術でもある。先行研究では Bootstrap や各処理に必要なコードの簡易化や新たな手法の開発により、処理実行時間を解決しようとしているものが多い。しかし、Bootstrap なしの完全準同型暗号の持つ可能性を考えることは、完全準同型暗号の社会への実装を考えるという点で社会的意義をもつ。また、この Bootstrap を利用しないケースで、GSW 式完全準同型暗号がどの程度の性能をもつのか、復号失敗割合をもとに評価を行うことは学問的意義も持つといえる。

第 3 節　本論文の構成・目的

第 1 項　構成および目的

本研究では主に 2 つのことを目標とする。1 つ目は、Bootstrap 処理なしの GSW 式完全準同型暗号における計算可能回数と、それにかかわる法や格子の関係を調べることである。そして 2 つ目は、法・格子の調節によって利用可能な完全準同型暗号の応用先を考えることである。1 つ目は 3 章実験で、2 つ目は 4 章で考察した。4 章の考察の際には、身近な例である学力試験を取り上げて将来的応用の可能性を説明した。また、3 章と 4 章を説明するための準備として 2 章を設けた。

第 2 項　なぜ学力試験を例としたのか

学力試験を例にした理由は 3 つある。1 つ目は学力試験がビッグデータ、AI のディープラーニングに比べて、比較的小規模で単純な分析のみでよいことである。そして 2 つ目は、昨年度の調査 [1] で、学校な等の教育機関への完全準同型暗号利用という視点が一切も見られなかったことに疑問を感じたことだ。3 つ目は教育分野への将来的な秘匿計算技術の応用は、考査期間における教員負担減少、ICT 教育の更なる普及につながるのではないかと考えたためである。

本研究では採点された試験結果から、平均値および標準偏差を求める手法のみについて考察したが、将来的には採点部分を含めて秘匿計算技術を扱いたい。(図 5) それによって、学力試験の個人のデータを秘匿したまま採点に関する処理が行えるようになる。つまり、最終的には安全に学力試験の採点を完全に一貫して外部委託することが可能になる。よって業務の多い教員の負担減少につながるのではないかと考えた。近年、教員数の減少、過労などによって教員の負担は増加している [16]。働き方改革が推進される一方、未だ多くの教育関係者が長時間の残業を強いられている。そのため、学力試験への応用を考えることは意義があると考えた。

以上の理由より、本研究では、将来的な応用の可能性、新たな完全準同型暗号利用先の開拓として学力試験を応用例として扱うこととした。

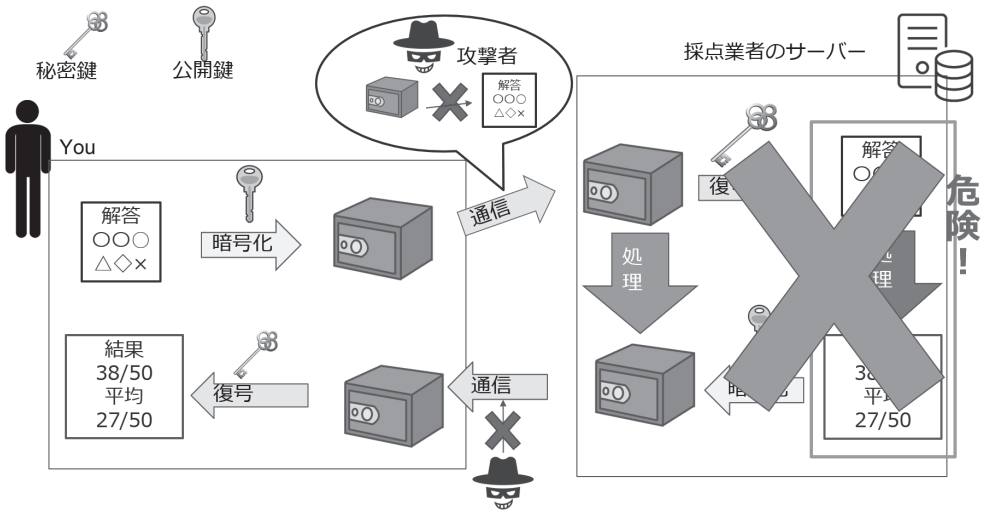


図 5　完全準同型を用いた学力試験採点処理の外部委託最終イメージ

第 2 章　　準備

第 1 節　本論文で使用する記号

本論文で使用する記号について定義する。

- $\mathbb{Z}$:　整数の集合
- $\mathbb{D}$:　離散正規分布
- $Dec(C_1)$:　暗号文 C_1 を復号する
- $Enc(m_1)$:　平文 m_1 を暗号化する
- $\phi(n)$:　$n \in \mathbb{N}$ に対して n と互いに素な 1 以上 n 以下の自然数の個数（オイラー関数）
- ${}^t s$:　s の転置行列
- $\mathbb{Z}_q$:　q が整数のとき、0 以上 q 未満の整数の集合
- $a \in \mathbb{Z}^{n \times m}$:　a は要素が全て整数の n 行 m 列の行列
- $\overrightarrow{0^m}$:　0 を要素とする m 次元のベクトル
- $\gcd(m,n)$:　整数 m と整数 n の最大公約数

第 2 節　一般的な暗号の課題と耐量子暗号の重要性

暗号には数学的な問題が背景に用いられている。ここでは現在広く使用されている RSA 暗号をもとに、暗号に数学的な問題が使われていることを説明する。そして、それをもとに本研究で用いた GSW 式完全準同型暗号が持つ耐量子暗号の性質に関する研究開発の重要性を説明する。

RSA 暗号とは Rivest, Shamir, Adleman によって考案された最初の公開鍵暗号である。大きな 2 つの素数を用意し、それらの積のみが与えられたときに 2 つの大きな素数への素因数分解が難しいことが用いられている。公開鍵には大きな 2 つの素数の積が用いられる。また、秘密鍵は大きな 2 つの素数が用いられる。公開鍵暗号には鍵生成、暗号化、復号の 3 つの段階がある。それぞれの段階について順に説明する。

第 1 項　鍵生成

まず、2 つの素数 p, q を生成し $n = pq$ を計算する。オイラーの定理より、$\phi(n) = (p-1)(q-1)$ と互いに素な整数 $e > 0$ を計算する。また、$ed \equiv 1 \pmod{\phi(n)}$ を満たす d 求める。このとき公開鍵を (e,n)、秘密鍵を d とする。

第 2 項　暗号化・復号

暗号化では平文 $0 \leqq m < n$ について $c \equiv m^e \pmod{n}$ を計算する。復号は c に対して $c^d \equiv m \pmod{n}$ を計算する。

証明　公開鍵 (e,n)、秘密鍵 d、平文 m について $m^{ed} \equiv m \pmod{n}$ を示す。
(1)$\gcd(m,n) = 1$ の場合

$$
\begin{aligned}
& ed \equiv 1 \pmod{\phi(n)} \\
\Leftrightarrow & \exists k \in \mathbb{Z}, ed = 1 + k\phi(n) \\
\Rightarrow & m^{ed} \equiv m \cdot m^{k\phi(n)}
\end{aligned}
$$

ここでオイラーの定理より $m^{\phi(n)} \equiv 1 \pmod{n}$ なので

$$m \cdot m^{k\phi(n)} \equiv m \cdot 1^k \equiv m \pmod{n}$$

したがって $m^{ed} \equiv m \pmod{n}$。

(2)$\gcd(m,n) = p$ の場合

$ed \equiv 1 \pmod{(p-1)(q-1)} \Rightarrow ed \equiv 1 \pmod{q-1}$ より

$$m^{ed} \equiv m \equiv 0 \pmod{p}, m^{ed} \equiv m \pmod{q}$$

中国式剰余定理の一意性より $m^{ed} \equiv m \pmod{n}$。
したがって $\gcd(m,n) = q$ の場合も同様に成り立つ。

以上 $\gcd(m,n) = 1$、$\gcd(m,n) = q$ の 2 つの場合において成立するので、$m^{ed} \equiv m \pmod{n}$ は示された。 □

第 3 項　計算量と耐量子暗号の重要性

「計算できる関数」という概念は、チャーチ＝チューリングの提唱によって数学的に定義されている。これは一般に、チューリングマシン（コンピュータ）で実行できるプログラム、その計算を実行できる有限のアルゴリズムが存在するような関数である。チューリングマシンとは計算を物理的現象、すなわち電気回路のオン・オフという 2 つの状態に置き換えて実行する機械である。したがって、計算は物理現象に置き換えが可能とも考えられる。その、逆として物理現象も何らかの計算を行っていると考えられる。1980 年代初頭に、数学者のマニンや物理学者のファインマンによって量子力学のチューリングマシンへの応用が考察され、1985 年には、イギリスの物理学者であるドイッチェによって量子チューリング機械という計算の数学的概念 [17] が提唱された。これは多数の状態が並存した量子状態をユニタリ変換することは、同時に複数の演算を並列で可能とすることを意味している。よって、量子チューリング機械は従来のチューリングマシーンが実行していた処理を多重に並列でおこなえるので、指数関数的に計算速度が向上することになる。

現在ほとんどのサービスで使われている主流の暗号は RSA 暗号、楕円曲線暗号、DSA 署名である。これらの暗号の安全性は、全数探索の計算量がはてしなく大きいこと・計算に時間がかかることに由来している。たとえば、RSA 暗号は素因数分解の困難性に依存している。つまり、量子チューリング機械によって計算速度が指数関数的に向上した場合、これらの素因数分解や離散対数問題に基づく暗号は簡単に全数探索や効率的なアルゴリズムによって低次の多項式時間で解読されてしまう。

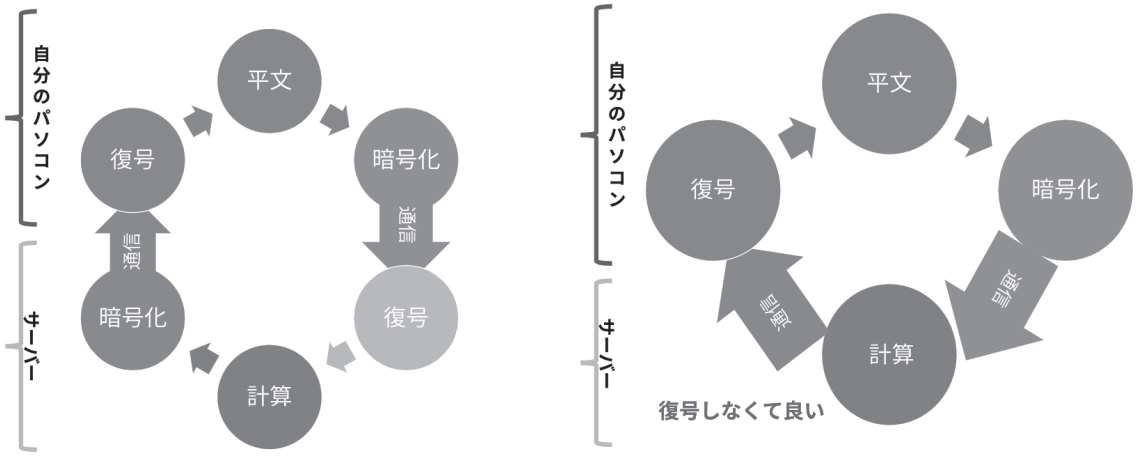
このことから、現在使われている公開鍵暗号の代替として耐量子暗号の開発が求められている。耐量子暗号は解読の困難性が計算量に由来するのではなく、数学的に解くことが難しいとされる NP 困難に由来する。ただし、耐量子暗号の一つで最短ベクトル問題（NP 問題）に由来する格子暗号には、パラメータの設定によって安全性が大きく変化するという課題がある。ケースによっては簡単に解読されてしまう場合があり、逆にワーストケース

でパラメータを設定すると暗号を構成することが極端に難しくなる。つまり、格子暗号は安全性と構成容易性のバランスがとりづらいというデメリットが存在する。

第 3 節　完全準同型暗号について

第 1 項　完全準同型暗号の重要性

電子決済、電子印鑑、オンライン会議、IoT 関連機器の拡充など新型コロナウイルスの拡大とともに増す電子サービスの重要性と利用に伴い、多種多様のデータを利用者から得られるようになった。これらの情報社会におけるビッグデータは、分析され、機械学習やデータマイニングなど様々なサービスに活用されている。しかし、それらのデータの所有や運用には初期コストや運用コスト、専門知識が必要となる。よって計算資源を自組織で所有・運用せずに外部委託するクラウドコンピューティングが利用されるようになった。クラウドコンピューティングのような計算の外部委託を行う際にデータは委託元（ユーザ）と委託先（サーバ）の間で安全にやり取りを行わなければならない。従来型の暗号ではユーザが暗号化したデータをサーバに送信し、その後サーバ内で一度データの復号を行う (図 6 左)。その後サーバで復号されたデータに計算処理を行い、処理結果を再び暗号化、ユーザへと通信する。この際にユーザとサーバ間の元のデータの安全性は暗号化により担保されるが、サーバ内でデータは一度復号された状態になる。そのため、クラッキングによる情報漏洩やサーバ側のデータの不正使用などサーバ内でのデータの安全性が担保されない。機密性の高い個人データについて、サーバ内で復号される手続きは、サーバ側の運用を信用する他なく、ユーザにとって懸念となってしまう。完全準同型暗号を用いた場合（図 6 右)、復号を行うことなくサーバ内で処理が可能になるため上記のような懸念をする必要がなくなり、高い安全性を担保しつつクラウドコンピューティングを行うことが可能となる。



通常暗号を用いた場合　　完全準同型暗号を用いた場合

図 6　クラウドコンピューティングについて

第 2 項　準同型性について

定義 1. 暗号における準同型性とは、秘密鍵や復号の必要なく暗号文のまま計算ができる性質のことである。

例 1. RSA 暗号の場合、平文 m_1、平文 m_2 に対してそれぞれの暗号文を $c_1 = Enc(m_1)$、$c_2 = Enc(m_2)$ とすると $Dec(c_1 * c_2) = m_1 * m_2$ が成立

証明　RSA 暗号における準同形性について証明する

公開鍵 (e,n), 秘密鍵 d とする。この時 2 つの平文 m_1, m_2 に対して c_1, c_2 は

$$c_1 \equiv m_1^e \pmod{n} \qquad c_2 \equiv m_2^e \pmod{n}$$

である。この時、$c_1 \cdot c_2$ を復号すると、

$$\begin{aligned}(c_1 \cdot c_2)^d &= (m_1^e \cdot m_2^e)^d \\ &= (m_1 \cdot m_2)^{ed} \\ &\equiv m_1 \dot{m}_2 \pmod{n}\end{aligned}$$

よって、$c_1 \cdot c_2$ を復号したものは $m_1 \cdot m_2$ であり、平文の積に一致している。

すなわち、RSA は乗算に関して、復号なく計算できることが示された。

□

上記の RSA 暗号のように、公開鍵暗号方式を用いた暗号には、乗算や加算どちらか一方の準同型性を持つものが知られている。

第 3 項　完全準同型暗号の成立

完全準同型暗号とは加算・乗算ともに回数に際限なく暗号文のまま計算が出来る暗号である。これによって加算・乗算によって表されている論理ゲートの演算も暗号文のまま実装が可能となる。完全準同型暗号の存在は公開鍵暗号方式が発見された 1978 年にリベスト、エイドルマン、デルトゾスによって示唆され [18]、その後スタンフォード大学のジェントリによってイデアル格子を用いた完全準同型暗号 [19] が提案された。格子に基づく暗号の実装により、限られた回数で加算・乗算ともに暗号文のまま計算可能になった。さらに Bootstrap 処理を用いることで、限られた回数でしか計算できなかった処理からノイズを取り除き、計算回数の制限を取り払うこのが可能になった。しかし、暗号文での演算処理の実行には多くの計算と時間が必要となる。

第 4 節　GSW 式完全準同型暗号

本研究では GSW 式完全準同型暗号を用いて実装・実験等をおこなった。GSW 式完全準同型暗号は Gentry, Sahai, Waters によって提案された格子に基づく暗号方式 [20] である。ワーストケースの LWE 問題が最短ベクトル問題（NP 困難）に基づくという LWE 仮定 [21] のもとに、安全性が保障されている。また、LWE 仮定に基づくこの暗号は耐量子暗号の性質も備えた暗号である。なお、本研究では計算可能な回数が制限されたうえでの応用可能性を仮定し、暗号プログラムを実装した。よって実装をおこなっていない Bootstrap 処理についての数学的説明は省く。

第 1 項　鍵生成

GSW 式完全準同型暗号は LWE 仮定 [21] に基づいて安全とされる。よって鍵は、LWE 分布よりサンプルする。LWE 分布からのサンプルとは、LWE 暗号によって定義される公開鍵、秘密鍵のサンプル手法のことである。以下、$n, m, q \in \mathbb{N}$ とする。

まず、LWE 分布を構成する A', s', e について定義する。A' は $n \times m$ の行列、s' はサイズ n のベクトルである。A', s の要素は、ともに 一様な分布 $\mathbb{Z}_q$ からランダムにサンプルされている。e は要素を $\mathbb{D}$ からランダムにサンプルした m 行のベクトルである。

A', s', e を用いて、

$$b = {}^t s' A' + {}^t e$$

とする。このとき、(A,s) を LWE 分布とよぶ。また、e をエラーベクトル、ノイズ、誤り、と呼び $\overrightarrow{0^m}$ に近似される。これらをもとに GSW 式完全準同型暗号の公開鍵 A と秘密鍵 s を以下のように定義する。

$$A = \begin{bmatrix} A' \\ {}^t b \end{bmatrix} \in \mathbb{Z}_q^{(n+1)\times m}$$

$$s = (-s', 1)$$

このとき ${}^t s \cdot A = {}^t e \approx \overrightarrow{0^m}$ が成立している。

第 2 項　暗号化

平文 μ（スカラー量）に対し、各成分が $\{0,1\}$ の二値をランダムにとる $m \times nl$ 行列 R（$l = \log_2 q$）を選び、ガジェットマトリックス [22] を G とする。このもとで暗号文 C は

$$C = \mu G - AR$$

と計算される。ガジェットマトリックスとは $u \in \mathbb{Z}_q^n$ が与えられたときに、$Gx = u$ となる小さな x を簡単に求められる特殊な行列である。具体的にはは次項で述べる。

第 3 項　ガジェットマトリックス

$g = (1,2,4,...,2^{l-1}) \in \mathbb{Z}_q^l$（ただし $l = \log_2 q$）となる縦ベクトル g を定義する。これをもとに、ガジェットマトリックス G を以下のように定義する。

$$G = I_n \otimes {}^t g = \begin{pmatrix} {}^t g & & 0 \\ & \ddots & \\ 0 & & {}^t g \end{pmatrix}$$

ここで LWE 問題では、G に以下の 2 つの性質が成り立つ。

定理 1. ${}^t b = {}^t s' G + {}^t e \quad \in \mathbb{Z}_q^{nl}(s', e$ は LWE 分布からサンプル）となる (G,b) が与えられたとき、s が簡単に求められる。

$s = (s_1,...,s_l)$,　$e = (e_1,...,e_l,...,e_{nl})$,$q = 2^l$ としたときに、$s_1 \in \mathbb{Z}_q$ に着目すると、

$$\begin{cases} b_1 & = s_1 \cdot 2^0 + e_1 \\ b_2 & = s_1 \cdot 2^1 + e_2 \\ & \vdots \\ b_{l-1} & = s_1 \cdot 2^{l-2} + e_{l-1} \\ b_l & = s_1 \cdot 2^{l-1} + e_l \end{cases}$$

となっている。ここで、$s_1 = s_{1,0} \cdot 2^0 + s_{1,1} \cdot 2^1 + \cdots + s_{1,l-1} \cdot 2^{l-1} \in \mathbb{Z}_q$ と 2 進展開することができるので、

$$
\begin{aligned}
b_l &= s_{1,0} \cdot 2^0 \cdot 2^{l-1} + s_{1,1} \cdot 2^1 \cdot 2^{l-1} + \cdots + s_{1,l-1} \cdot 2^{l-1} \cdot 2^{l-1} + e_l \\
&= s_{1,0} \cdot 2^{l-1} + s_{1,1} \cdot 2^l + \cdots + s_{1.l-1} \cdot 2^{2l-2} + e_l \\
&\equiv s_{1,0} \cdot 2^{l-1} + e_l \pmod{q}
\end{aligned}
$$

ここで、$s_{1,0}$ は 1 または 0 であり、e_l は $e \approx \overrightarrow{0}$ より 2^l に対して小さな値であるため、$b_l \equiv s_{1,0} \cdot 2^{l-1} \pmod{q}$ が 0 に近いとき、すなわち、$b_l < q/4$ または $b_l \geqq 3q/4$ のとき、$s_{1,0} = 0$ である。逆に $q/4 \leqq b_1 < 3q/4$ のとき $s_{1,0} = 1$ である。（図 7）

$s_{1,1}$ 以降についても、以下と同様の手順を踏むことによって求められる。

$b_{l-1} = s_1 \cdot 2^{l-2} + e_{l-1} \pmod{q}$、すなわち $s_{1,0} \cdot 2^{l-2} + s_{1,1} \cdot 2^{l-1}$ について、$s_{1,0}$ がすでに分かっているので、それを引き算する。そして、$s_{1,0}$ 導出時と同様の評価方法を用い、s の最上位 2 ビット目を計算する。これを最下位ビットまで、求めたビットを減算し、評価することを繰り返す。

これらによって、s_1 の値を b_1 から b_l までアルゴリズム的に求めることができる。

s_2 以降も、s_1 導出と同様の手順を b_{l+1} から b_{2l} まで踏むことにより求めることができる。以上により、(G, b) が与えられたときに s がアルゴリズム的に、すなわち簡単に、求められる。

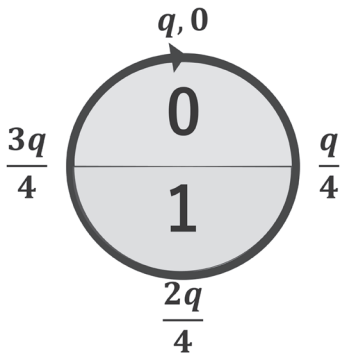


図 7　0,1 の評価と法 q の関係についてのイメージ

定理 2. $u \in \mathbb{Z}_q^n$ が与えられたとき $Gx = u$ となる小さな x を求められる。

縦ベクトル u, x を $u = (u_1, ..., u_n)$,　$x = (x_1, ..., x_l, ..., x_{nl})$ とする。ただし、x の要素は 0 または 1 である。

まず u_1 に着目すると

$$x_1 + 2x_2 + \cdots + 2^{l-1}x_l = u_1$$

となるため、u_1 の 2 進数表現から $(x_1, ..., x_l)$ が容易に求められる。同様に u_2 からは 2 進数表現によって $(x_{1+1}, ..., x_{2l})$ が容易に求まる。以降、同様にして、x は u から容易に求めることができる。このとき、G を解く関数を G^{-1} と表す。この関数について $G \cdot G^{-1}(u) = u$ が成立する。ただし、G^{-1} は G の逆関数でありで、G の逆行列ではない。

第 4 項　復号

暗号文 C に対して秘密鍵の ${}^t s$ をかけることで、

$$
\begin{aligned}
{}^t s \cdot C &= \mu {}^t s \cdot G + {}^t s \cdot AR \\
&= \mu {}^t s \cdot G + {}^t e \cdot R \\
&\approx \mu {}^t s \cdot G
\end{aligned}
$$

より、G に関する式が得られ、前項のガジェットマトリックスを解く方法を用いて $\mu {}^t s$ をもとめることができる。平文 μ は秘密鍵 ${}^t s$ のスカラー倍より、平文が得られる。

第 5 項　準同型性に関わる証明

GSW 式完全準同型暗号は加算・乗算について準同型性が成立する。すなわち、2 つの平文 m_1, m_2 に対してそれぞれの暗号文を C_1, C_2 とすると、

$$
m_1 + m_2 = Dec(C_1 + C_2) \qquad\qquad m_1 \cdot m_2 = Dec(C_1 \cdot G^{-1}(C_2))
$$

が成立する。このとき、$G^{-1}(C_2)$ はガジェットマトリックスの性質より容易に求めることができる。また、$C_1 \cdot C_2$ では $m_1 \cdot m_2$ を求めることはできない。準同型性の証明は以下の通りである。

証明　まず加算についての準同型性を示す。

$$
\begin{aligned}
C_1 + C_2 &= (m_1 G - AR_1) + (m_2 G - AR_2) \\
&= (m_1 + m_2)G - A(R_1 + R_2)
\end{aligned}
$$

ここで両辺に秘密鍵 ${}^t s$ をかけて復号すると、

$$
\begin{aligned}
{}^t s(C_1 + C_2) &= {}^t s(m_1 + m_2)G - {}^t sA(R_1 + R_2) \\
&\approx {}^t s(m_1 + m_2)G
\end{aligned}
$$

よって $C_1 + C_2$ の復号結果は $m_1 + m_2$ となり加算の準同型性が示された。

次に乗算についての準同型性を示す。

$$
\begin{aligned}
C_1 \cdot G^{-1}(C_2) &= (m_1 G - AR_1)G^{-1}(C_2) \\
&= m_1 G \cdot G^{-1}(C_2) - AR_1 G^{-1}(C_2) \\
&= m_1 C_2 - AR_1 G^{-1}(C_2) \\
&= m_1 m_2 G - A(R_1 G^{-1}(C_2) + m_1 R_2)
\end{aligned}
$$

ここで両辺に秘密鍵 ${}^t s$ をかけて復号すると、

$$\begin{aligned} {}^t s(C_1 \cdot G^{-1}(C_2)) &= {}^t s m_1 m_2 G - {}^t s A(R_1 G^{-1}(C_2) + m_1 R_2) \\ &\approx {}^t s(m_1 m_2) G \end{aligned}$$

よって $C_1 \cdot G^{-1}(C_2)$ の復号結果は $m_1 \cdot m_2$ となり乗算の準同型性が示された。

□

この際、加算・乗算について ${}^t s \cdot A = {}^t e \approx \overrightarrow{0^m}$ を用いている。加算の場合は $R_1 + R_2$、乗算の場合は $R_1 G^{-1}(C_2) + m_1 R_2$ を $\overrightarrow{0^m}$ と近似することによって復号が可能となる。これらの値は計算を重ねるごとに大きくなる。この増加する値を完全準同型暗号におけるノイズとよぶ。ノイズの値の増加によって、計算を重ねると $\overrightarrow{0^m}$ へと近似できなくなることがある。これをノイズによるエラーという。

第 3 章　　加算・乗算における実験および考察

第 1 節　加算実験概要

Bootstrap 処理なしの場合における完全準同型暗号利用に向けて、基本的な完全準同型暗号かつ、耐量子暗号の性質をもち、将来的にも使用可能な GSW 式完全準同型暗号の利用を考えた。利用するにあたり Bootstrap 処理なしで何回まで計算が可能であるのか、また計算可能な回数を決める要因は何であるのかについての調査をおこなった。

第 1 項　目的

Bootstrap なしでの完全準同型暗号の応用を考えるため、加算における計算可能回数とそれを決定する要因を調査すること。

第 2 項　加算研究手法

実験は加算処理に対して、大きく分けて 2 種類おこなった。１つ目の調査は、何回目まで復号失敗とならずに計算ができるのか、すなわちエラーベクトルによって削除ができる係数 R の限界を調査した。このとき、法の値は 1024、格子の値は 3×3 で設定した。実際に扱われる法及び格子の大きさはこの値よりはるかに大きい。しかし、GoogleColaboratory かつノートパソコン上での実装をおこなったため、大きな値を用いる実行は困難であった。そこで、法及び格子は実験が容易にでき、実験環境で評価が可能なサイズでおこなった。２つ目の調査は、暗号を定義する際に法・格子の広さを変化させることによって法の中での加算における１００回中に復号失敗がでた割合を調査した。それぞれの実験における具体的手順は実験の各節で説明する。

加算実験においては、復号失敗という語を用いる。復号失敗とは、平文 m_1, m_2 に対して、それぞれの暗号文を C_1, C_2 としたとき $Dec(C_1 + C_2) - (m_1 + m_2) \neq 0$ のことである。すなわち、完全準同型性を持つのにもかかわらず、暗号の特性によって暗号文同士の加算結果を復号したものが、平文同士の正しい加算の結果に一致しないことを復号失敗とよぶ。ただし、本研究では Bootstrap を用いないケースに限定して復号失敗を考えた。

第 3 項　Python を用いた GSW 式完全準同型暗号の実装

本実験では GoogleColaboratory 環境下で Python (version3.7) を用いた。実装においては Python の NumPy モジュールを使用した。以下の手順それぞれについてコードを定義した。実装したコードについては付録を参照されたい。また、暗号生成に関する数学的な操作は第 2 章第 4 節を参照されたい。

鍵生成

1、LWE 分布を作成するための分布を生成する

2、LWE 分布の生成

3、GSW 式完全準同型暗号の秘密鍵・公開鍵の生成

暗号化

1、ガジェットマトリックスの生成

2、暗号文の計算

復号

1、秘密鍵による暗号文解読

2、ガジェットマトリックスの逆関数演算

3、スカラー倍の演算

第 2 節　実験 1 : 加算回数を求める

計算回数が何回まで可能であるのか実験によって調べた。同じ数の加算、法固定における計算実行回数の最大値は平文の値に依存する。平文が小さいほど、法範囲で多くの加算ができ、大きいほど少ない回数の加算しかおこなえない。よって、平文の値を変化させたときに復号失敗の出る割合がどのように変化するのかを調査した。

第 1 項　方法

1、GSW 式完全準同型暗号の作成 (本章第 1 節第 3 項参照)
2、法を 1024、格子の大きさを 3 × 3、シグマ範囲は 1 と定義し鍵を生成した
3、2 について、平文を 1 として法を超えない計算範囲（1023 以下）で復号失敗が起こらずに繰り返し同じ値を加算できるかどうかを調査した
（繰り返し加算する暗号文は同じ暗号文を用いた）
4、2,3 を 100 回繰り返し、100 回中で復号失敗が発生した回数を調べて復号失敗割合とした
（毎回鍵生成をおこない違う鍵の値で繰り返した）
5、3 について平文の値を 1〜10 で変化させ、それぞれの平文の値に対して 2〜4 を 10 回ずつ繰り返した

(本章第 3 項参照)

第 2 項　結果

変化させたときの平文を横軸にとり、100 回中のエラーの割合を縦軸に取ったときの散布図を図 8 に示す。

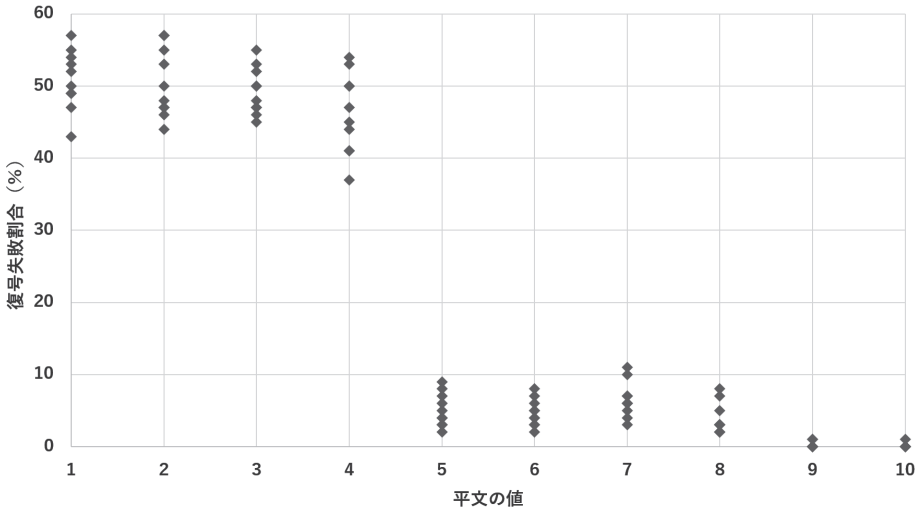


図 8　加算における平文と復号失敗割合の関係（同じ暗号文加算）

この図より、平文の値が 4 から 5 にした時に復号失敗の発生する割合が急激に減少していることがわかる。また、平文の値が小さい時の演算では約 50%程度の割合で復号失敗が発生してしまうことがわかる。

第 3 項　考察

平文の値が 4 の場合と 5 の場合において、法を超えない範囲での可能な最大の計算回数が異なる。4 の場合は 256 回、5 の場合は 200 回程度である。計算回数によって復号失敗がおこると考えた場合、200〜256 回目の間の計算回数目で多くの復号失敗が発生しているのではないかと考えられる。

第 3 節　追加実験：復号失敗発生時の分析

前節で、平文の値が 4 から 5 に変わるとき (200〜256 回目) に復号失敗の発生回数が大きく変化していることがわかった。このことから、平文が 4 より小さいときに、どの計算回数目で復号失敗が出ているのかを調査すれば、計算可能回数が具体的にわかるのではないかと考えた。そこで、平文の値を 1 にしたときの復号失敗について、計算回数が何回目であるのかについてを調査した。

第 1 項　結果

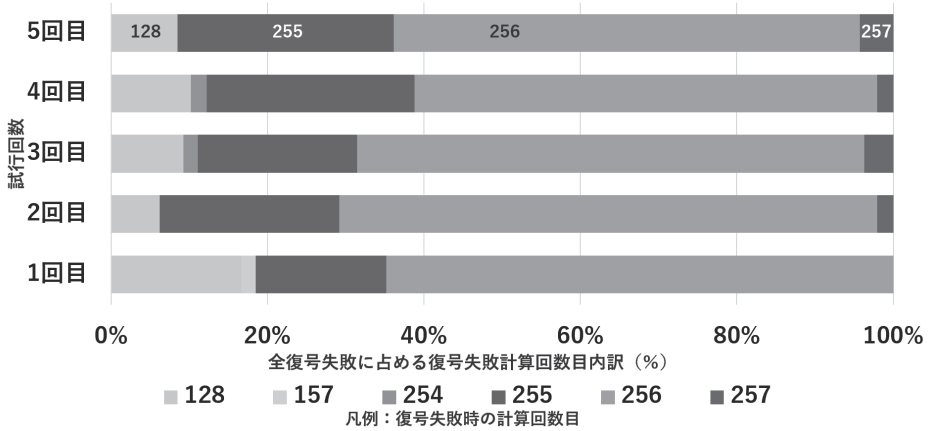


図 9　復号失敗が出たときの加算回数目の比較
（実験 1 における平文「1」加算の復号失敗時の加算回数目内訳）

図 9 は 5 回分の試行（実験 1 第 1 項の 1〜4 を 5 回繰り返した時）のそれぞれの場合で、100 回中でのエラーが発生した回数を 100%として、それぞれのエラーについて何回目の計算で復号失敗となったのかの割合をグラフにしたものである。特定の計算回数で復号失敗が多く発生し、254 回以上の計算過程での復号失敗が全失敗中のほぼ 9 割を占めていることが読み取れる。

第 2 項　考察

実験 1、及び追加実験より、3×3 の格子で法を 1024 としたときに、復号失敗は 254 回目以降で多く発生し、計算回数を 254 回未満に抑えることで復号失敗割合は格段に抑えることが出来ることが分かった。

今回の実験では、暗号文を取り換えずに同じ暗号文加算をおこなった。よって、254 回目の暗号文 C は平文 1 を μ、ガジェットマトリックスを G、ランダムな 0,1 で構成される行列を R としたとき、GSW 式完全準同型暗号の加算準同性証明の式より $C = 254\mu \times G - A \times 254R$ と表される。これを復号する際、秘密鍵 s を用いて $s \cdot AR$ が 0 に近似されることを利用する。254 回目以降の計算で復号失敗となった時この近似が上手くいかなかったことが考えられる。つまり、$s \cdot AR \times 254R$ が 0 に近似されなかったのではないかと考えられる。よってこの実験からは計算回数を 254 回未満に抑えることが、復号失敗を防ぐ観点から重要であるといえる。

また、今回の実験では同じ暗号文で加算をおこなったため、R の値がその値のまま加算された。この場合、はじめに構成する R の要素に 1 が一つでも含まれていた場合、254 回目の計算で 254 となる要素が R 内に現れる。よって、R の値が大きくなりやすい。しかし、実際に利用する場面では資料ごとに暗号化をおこなうため R の値は取り換えられる。この場合、一回の計算ごとにそれぞれの R の要素に加算される値は 0 または 1 であるので、0 と 1 が等確率で選択されたとき 254 回目付近の計算で R に含まれる要素の期待値は 128 となる。よって、同じ暗号文加算より多く計算できるのではないかと予想できる。

第 4 節　追加実験：暗号文を都度生成した場合（追加実験 1）

追加実験 1 の考察で、暗号文を都度生成して加算した場合に、復号失敗割合が減少するのではないかと考えた。そこで、加算する時に違う暗号文を生成した場合で実験 1 と同様の方法の実験をおこなったときに復号失敗の発生する割合がどのように変化するのかを調査した。この調査によって暗号文をデータによって生成しなおす実践により近いケースでの計算可能回数が実験値として調べられる。

第 1 項　方法

1、GSW 式完全準同型暗号の作成 (本章第 3 項参照)
2、法を 1024、格子の大きさを 3×3、シグマ範囲 1 と定義し鍵を生成した
3、2 ついて、平文を 1 として法を超えない計算範囲（1023 以下）で同じ鍵を用いて暗号文を生成し、加算することを繰り返した。
4、2,3 を 100 回繰り返し、100 回中で法を超えない範囲で復号失敗となった回数を調べ、復号失敗割合とした（毎回鍵生成をおこない違う鍵の値で繰り返した）
5、3 について平文の値を 1〜10 で変化させ、それぞれの平文の値に対して 2 と 4 を 10 回ずつ繰り返した

第 2 項　結果

変化させたときの平文を横軸にとり、100 回中復号失敗割合を縦軸に取ったときの散布図を図 10 に示す。

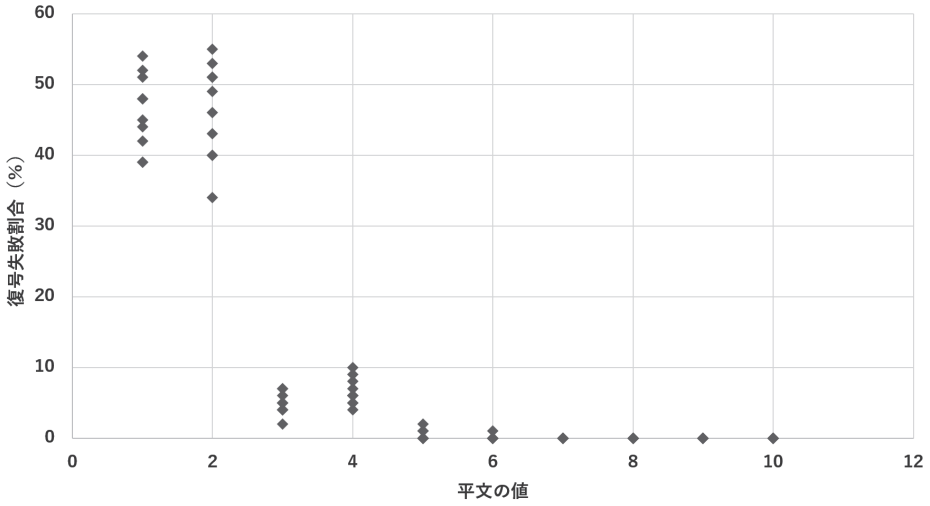


図 10　加算における平文と復号失敗割合の関係（違う暗号文加算）

この図より平文の値が 2 から 3 に変化するときに大きく復号失敗割合が減少していることがわかる。また、復号失敗割合は平文が 1,2 の時は 50%程度、3,4 で 10%程度、5〜10 はほぼ 0%である。

第 3 項　考察

結果より、10%程度の復号失敗を無視した場合に、計算可能な回数は平文の値が 2 の時と 3 の時の計算回数の間に存在すると考えられ、340〜512 回であると考えられる。

同じ暗号文を加算させた場合と都度、暗号文を取り換えて暗号文を加算させた場合を比較した散布図は図 11 のようになる。

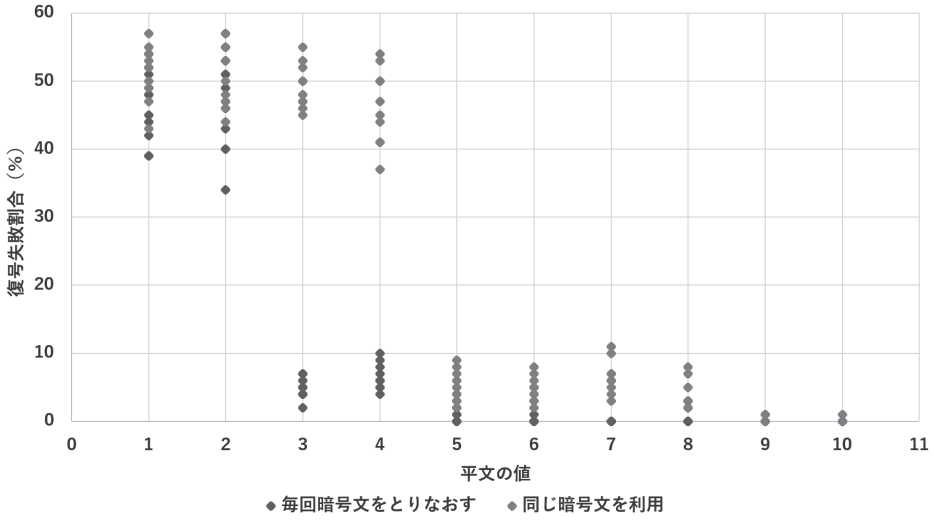


図 11　同じ暗号文を用いた場合と都度生成した場合の比較

図 11 より、都度暗号文を取り直して加算を行った場合のほうがより、復号失敗となる可能性を低減しているといえる。また、R のそれぞれの要素において一度の計算で加算される値の期待値が 1/2 であることから、512 回以上の計算は、R の近似が可能と予想される $256R$ を超えてしまったため、復号失敗になったと考えられる。このことからも、$S \cdot A$ によって 0 に近似できる R の値は $256R$ 程度であることがわかる。

同じ暗号文加算については、全復号失敗したケースに占める復号失敗時の計算回数目について調べ、具体的なエラー発生回数を求めた。しかし、本実験では R の値が実験ごとにことなる。なぜなら、$s \cdot A$ によって R が 0 に近似できなくなる計算回数目が大きく異なるためである。よって、具体的な計算限界回数を調べる実験はおこなわなかった。

第 5 節　計算可能な回数の理論値の計算

第 1 項　理論値の計算

実験１、及び追加実験１、２を通して求めた計算可能な回数の実験値は 256 回程度であった。本節では暗号の生成、暗号化、復号のプロセスを通じて、計算可能回数の導出に試みた。また、その導出値をもって、実験値と同様の法 1024、格子 3×3、シグマ範囲 1 の場合において計算可能な回数は何回であるのかを理論値として導出した。

法 q、格子 $n \times m$、シグマ範囲 σ と定義した場合、実験で用いたアルゴリズムにおいて、公開鍵は $A \in \mathbb{Z}_q^{(n+1)\times m}$、秘密鍵は ${}^t s \in \mathbb{Z}_q^{n+1}$ である。また、${}^t s \cdot A$ は平均が 0、標準偏差が σ 範囲の正規分布からサンプルした小さなエラーベクトル ${}^t e$ に等しい。

このとき、平文 μ の暗号文 $C = \mu G - AR$ を x 回加算した後に、復号することを考える。R は 0 と 1 で構成されるランダムな行列で、G はガジェットマトリックスである。

一般に R は、加算する度に変化する値であるが、変化しないものとして考える。すなわち、もっとも復号失敗しやすい場合（以下ワーストケース）を考えて乱数を固定する。この時、x 回目における暗号文 C' は

$$C' = x\mu G - xAR$$

である。これに ${}^t s$ をかけ、復号した場合、右辺は ${}^t sx\mu G - {}^t sAxR = {}^t sx\mu G - {}^t exR$ である。この時に、${}^t exR$ が 0 に近似されない場合に復号ができない。つまり、xR が大きくなると復号できない。

いま議論したいのは x の回数が何回より小さければエラーがでないか、であるから R の値が最大となるケースにおいて、復号可能となるような x を求めればよい。そこで、ワーストケースとして R は要素がすべて１の

$m \times (n+1)l$ の行列であると仮定する。また、エラーベクトルについても、正規分布から取りうる最も大きい値がすべての要素に含まれたケースと仮定して、各要素を以後 σ と統一して表すこととする。

このとき、x 回目の計算における ${}^texR$ の各要素は $xm\sigma$ である。このとき、x 回加算した暗号文 C' に ${}^ts$ をかけたときの i 番目の各要素を C'_i と表すこととすると、

$$C'_{l(n+1)} = x\mu(2^{l-1} \cdot s_{n+1}) + mx\sigma$$

である。ここで、ガジェットマトリックスの復号より、

$$mx\sigma < \frac{q}{4} \pmod q$$

であれば正しく評価が行え、復号が可能である。これは他の C_i についても同様の不等式が成立する。

すなわち、ワーストケースにおける計算可能な回数 x は

$$x < \frac{q}{4m\sigma}$$

である。法が 1024、格子が 3×3、シグマ範囲 1 で定義した場合において、エラーベクトルの要素が必ず 0,1,-1 のいずれかの値をとるとき、計算可能な回数 x は

$$x < \frac{q}{4m\sigma} = \frac{1024}{4 \cdot 3 \cdot 1} = \frac{1024}{12} \approx 85$$

よって 85 回以下であれば必ず復号失敗とならずに計算できるということがわかる。

第 2 項　理論値と実験値の考察

実際に 85 回目で復号失敗の発生確率は、R のみに着目し、R 内の要素の 0、1 が 1/2 の確率で選択されるとき $(1/2)^{m\times(n+1)l\times x} = (1/2)^{3\times4\times10\times85} = 2^{-10200}$ で限りなく 0 に近い。

このとき、R の各要素の期待値は 1/2 と考えられる。同じ列に同じ割合で 0 と 1 が必ず含まれているとした場合、${}^teR = (m\sigma/2, ..., m\sigma/2)^{(n+1)l}$ より、計算可能な回数の期待値は $x < q/2m\sigma$ ともとめられる。よって、実験での定義の場合、おおよそ 170 程度となる。また、実験ではエラーベクトルの値がすべてが 1 ではなく 0 や -1 も含まれているので、さらに小さい。よって、実験の場合において R を固定すると、254〜256 回目に復号失敗が集中したのではないかと考えられる。128 回目で発生した復号失敗についても計算可能な理論値が 85 回以下であることを念頭に考えると、起こりうるといえる。

復号失敗が発生した実験値が 2 の累乗と関連が見られるのは、理論値分析より、判定閾値であるためと考察される。実験値同様の、法が 1024 のとき、それを 4 で割った 256 が判定閾値である。同じ暗号文の加算において、ノイズの値は計算回数に比例して増加する。よって、ノイズの値の増加係数が 1 のとき、すなわち計算回数が 1 増えるごとに各要素のノイズが 1 増加するようなとき、256 回目付近で復号失敗が発生する。ノイズの増加係数が 2 のとき、すなわち計算回数が増えるごとに各要素のノイズが 2 増加するようなとき、128 回目付近で復号失敗が発生する。このように、判定閾値である 256 が 2 の累乗値であるため、復号失敗の値が 2 の累乗と関連がみられると考えられる。

平文が 1 の時に復号失敗なく法の範囲の最後まで計算できた場合、1024 回分加算しているので理論値の 10 倍の計算がおこなえている。これは、あきらかにおかしく、エラーベクトルの値がすべて 0 であったためではないかと考えられる。この場合 LWE 仮定より安全性が低く、簡単に平文の値が計算されてしまうことが懸念される。よって、実験 1 での 50%程度の復号失敗は無視することができない。この場合、復号失敗確率を 0%にするうえでは理論値の 85 を採択すべきであり、10%程度の復号失敗を許容する場合は実験値の 254 回未満を採択すべきと考える。ただし、実用上における 10%の復号失敗は許容し難い。よって、85 回付近までを計算可能回数とするのが妥当であるといえる。

第 3 項　実験２にむけた仮説

実験２では、法を大きくした場合と格子を広くした場合に復号失敗割合がどのように変化するのかを実験により観察する。ここで、今回求めた理論値より $x < q/4m\sigma$ であるから、法の値（q）を大きくすると分子が大きくなり、格子の大きさ（m）を大きくすると分母がおおきくなる。よって、法を大きくして格子を広くすればするほど計算可能な回数は増えると考えられる。すなわち、計算可能な回数は法の大きさに比例し、格子の大きさに反比例すると考えられる。

第 6 節　実験２：法と格子を変化させたときの復号失敗割合

実験 1 では、計算可能な回数を R に着目して考えた。しかし、実験 2 では計算可能な回数を決める要因は何であるのかを調べるために、暗号を定義する値である法・格子の広さを変化させ、復号失敗割合がどのように変化するのか、調査をおこなった。

第 1 項　方法（法）

1、GSW 式完全準同型暗号の格子の広さを 3×3、平文の値を 1 に設定した
2、法の大きさを 2^3 で定義した
3、暗号の鍵・暗号文を生成した
4、平文の値が、2 で設定した法の値を超えない範囲で、繰り返し加算を行った
5、3,4 を 100 回行い、100 回中で法の超えない範囲にもかかわらず復号失敗となった回数を調べ、復号失敗割合とした
6、2〜5 を 10 回繰り返した 7、2 について法の値 $2^4, 2^5, ..., 2^{10}$ と変化させ他の法の値についても同様に 2〜6 を行った

第 2 項　結果（法）

変化させた法の値を 2 で対数をとったものを横軸に、復号失敗割合を縦軸に取った場合の散布図を図 12 に示す。

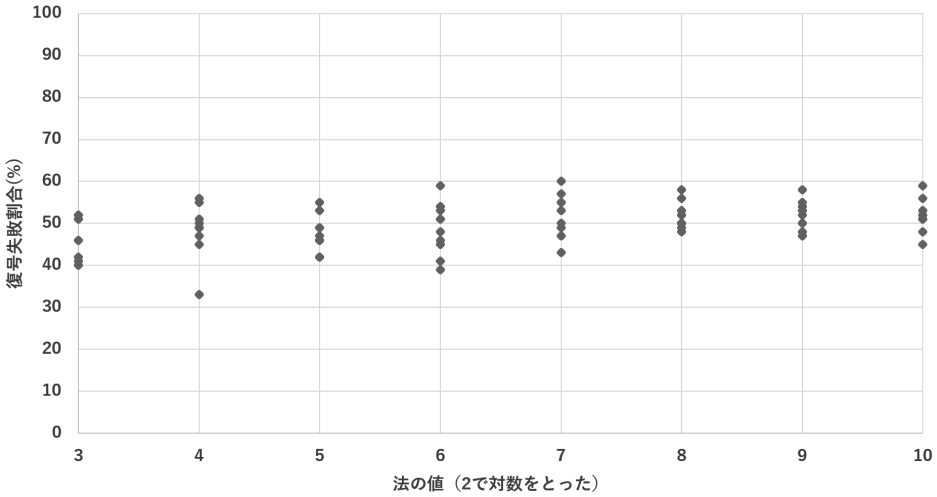


図 12　加算における法と復号失敗割合の関係（同じ暗号文加算）

図 12 より、法の値を大きくしても復号失敗割合はあまり変化せず、40〜60 回程度である。また、法の値を 2 で対数をとった値と復号失敗割合の相関係数は 0.38、P 値は自由度 78（標本数 80）の t 分布による両側検定（相関係数が 0 であることを仮定した）を行った結果 0.0004 であった。線形回帰を行った結果 $y = 0.84x + 44.19$ であった。

第 3 項　考察（法）

法を大きくすると計算可能な範囲が大きくなるため、計算回数が大きくなる。しかし、法を変化させても復号失敗の発生する割合に大きな変化は見られなかった。

0 ～ 100 まで縦軸が取れるのに対し、回帰直線の傾きは 0.8 で、横軸は 2 を底にもち、法を進数にもつ対数関数であるから、法を大きくしても復号失敗割合に変化がないといえる。相関係数については、P 値の値が 0.0004 ＜ 0.05 であることから相関係数が 0 という帰無仮説は棄却され、法を 2 で対数をとった値と復号失敗割合に弱い正の相関があることが認められる。しかし、相関係数及び P 値は、縦軸と横軸のプロット幅が異なり、単位の異なる値であるため、分析の値として十分でない。そこでデータを標準化して分析をおこなうこととした。

データに対して平均０、標準偏差１になるよう標準化すると次の図 13 がえられた。

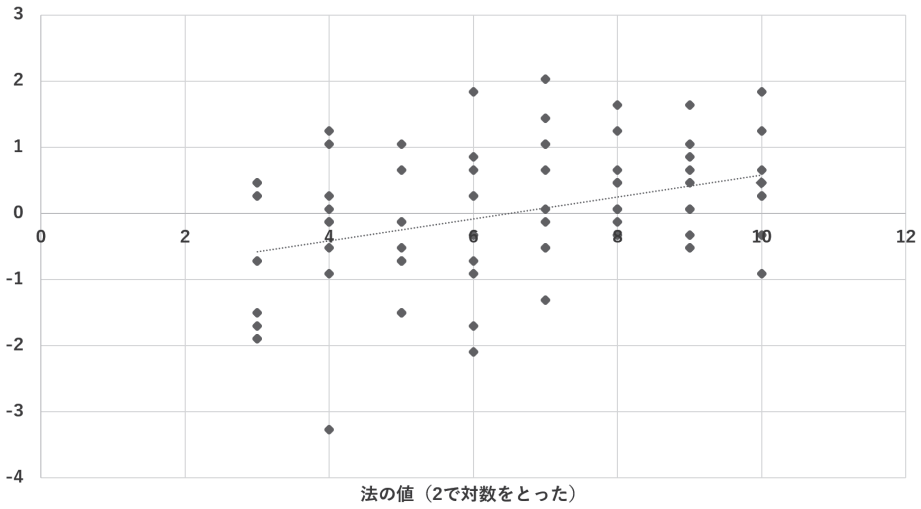


図 13　加算における法と復号失敗割合の関係（標準化）

ここでの回帰直線は、決定係数が 0.15 と小さく出たことから、あまり意味をなしていないことがわかる。また、データも正規分布には従わないことが読み取れる。縦軸を 0〜100 で取った場合に比べ、標準化した散布図からはデータの相関関係が弱いことも読み取れる。法の値が同じであっても復号失敗回数には大きくばらつきがあり、法を大きくしたからといって、データの復号失敗割合が大きくなるとはいえないことが標準化後もいえる。

論値を計算した際に、法を大きくすると計算可能な回数が大きくなると予想したが、本実験では復号失敗割合のみの評価で、何回目の計算回数目で復号失敗となったかについては復号失敗の分析として困難であると考えたため、判定していなかった。そのため、理論値との関係性については検証することはできなかった。

第 4 項　方法（格子）

1、GSW 式完全準同型暗号の法の値を 1024、平文の値を 1 に設定した

2、格子の広さを 3×3 で定義した

3、暗号の鍵・暗号文を生成した

4、平文の値が、法の値を超えない範囲で、繰り返し加算を行った

5、3,4 を 100 回行い、100 回中で法の超えない範囲にもかかわらず復号失敗となった回数を調べ、復号失敗割合とした

6、2〜5 を 10 回繰り返した 7、2 について格子の広さを $5 \times 5, 7 \times 7, \ldots, 17 \times 17$ と変化させ他の方の値についても同様に 2〜6 を行った

第 5 項　結果（格子）

変化させた格子の広さを横軸に、復号失敗割合を縦軸にとった場合の散布図を図 14 に示す。

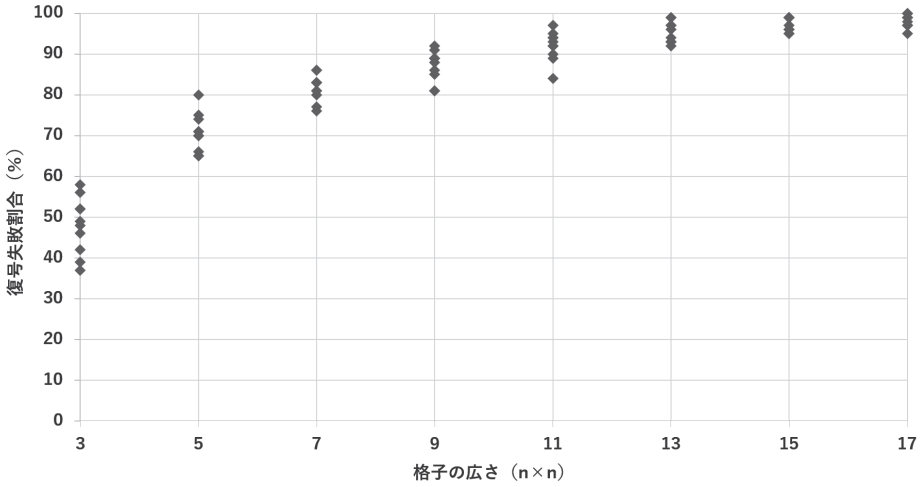


図 14　加算における格子の広さと復号失敗割合の関係（同じ暗号文加算）

格子の大きさを大きくすると復号失敗割合は増加し、100%に近付いていることが読み取れる。指数近似をおこなうと $y = 100 - 92.68e^{-0.29x}$ であった。その時の決定係数は 0.87 であった。

第 6 項　考察（格子）

指数近似、およびグラフから、100%に近づくように復号失敗割合が増えていることが読み取れる。格子と復号失敗割合には関係性があり、格子を広くすると復号失敗割合は大きくなることがわかった。実験 1 及び追加実験 1,2 より、復号失敗には R の要素が 256 程度を超えるか否かが影響している。よって、格子を大きくすることによって含まれる 1 の割合に変化がなくとも要素数が増加する。それにより、より早く R の値が膨らみやすくなり $s \cdot A$ による 0 への近似ができなく、復号失敗が発生したと考えられる。

第 7 節　追加実験：暗号文を都度生成した場合（追加実験 2）

実験 2 より、法の値を変化させても復号失敗割合に変化がない、また格子を変化させると復号失敗割合は 1 に近づくように増加することがわかった。しかし、これらの結果は同じ暗号文加算によるものである。そこで、より実際に使用する場合に近い、暗号文を加算する度に取り直した際に同様の実験を行い、実験 2 と同様の結果が得られるかを調査した。

第 1 項　方法（法）

1、GSW 式完全準同型暗号の格子の広さを 3×3、平文の値を 1 に設定した

2、法の大きさを 2^3 で定義した

3、暗号の鍵を生成した

4、平文の値が、2 で設定した法の値を超えない範囲で、暗号文を生成し加算することを繰り返しおこなった（加算する度に暗号文は生成しなおした、ただし鍵の値は同じものを用いた）

5、3,4 を 100 回行い、100 回中で法の超えない範囲にもかかわらずで復号失敗となった回数を調べ、復号失敗割合とした

6、2〜5 を 10 回繰り返した

7、2 について法の値を $2^4, 2^5, ..., 2^{10}$ と変化させ他の方の値についても同様に 2〜6 を行った

第 2 項　結果（法）

変化させた法の値を 2 で対数をとったものを横軸に、エラーの発生割合を縦軸に取った場合の散布図を図 15 に示す。

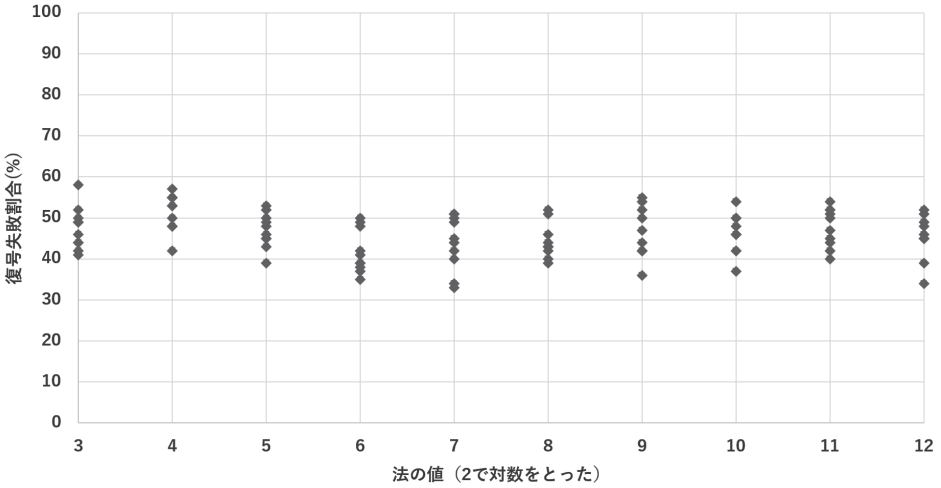


図 15　加算における法と復号失敗割合の関係（異なる暗号文加算）

2 で対数をとった法の値と、復号失敗割合について、相関係数は −0.11 であった。自由度 78 の t 分布によって相関係数が 0 になることを帰無仮説として両側検定を行った結果、P 値は 0.25 と 0.05 より大きく、帰無仮説は棄却されなかった。

第 3 項　考察（法）

法の値を 2 で対数をとった値と、復号失敗割合について、実験 2 と本実験結果は異なっている。相関係数が低く、P 値も大きく出ていることから相関はない。グラフの形状からも、法を大きくしても復号失敗割合は 40〜60 あたりに分布しており、同じ法の値でも復号失敗割合に大きくばらつきがあることがわかる。これらのことから、法と、復号失敗割合に関係がないといえる。法を大きくしたからといって、復号失敗割合が 60 を極端に超えたり、40 を極端に下回ったりすることはないと考えられる。

第 4 項　方法（格子）

1、GSW 式完全準同型暗号の法の値を 1024、平文の値を 1 に設定した

2、格子の広さを 3×3 で定義した

3、暗号の鍵を生成した

4、平文の値が、法の値を超えない範囲で、暗号文を生成し加算することを繰り返しおこなった（加算する度に暗号文は生成しなおし、鍵の値は同じものを用いた）

5、3,4 を 100 回行い、100 回中で法の超えない範囲にもかかわらず復号失敗となった回数を調べ、復号失敗割合とした

6、2〜5 を 10 回繰り返した

7、2 について格子の広さを $5 \times 5, 7 \times 7, \ldots, 17 \times 17$ と変化させ他の方の値についても同様に 2〜6 を行った

第 5 項　結果（格子）

変化させた格子の広さを横軸に、復号失敗割合を縦軸にとった場合の散布図を図 16 に示す。

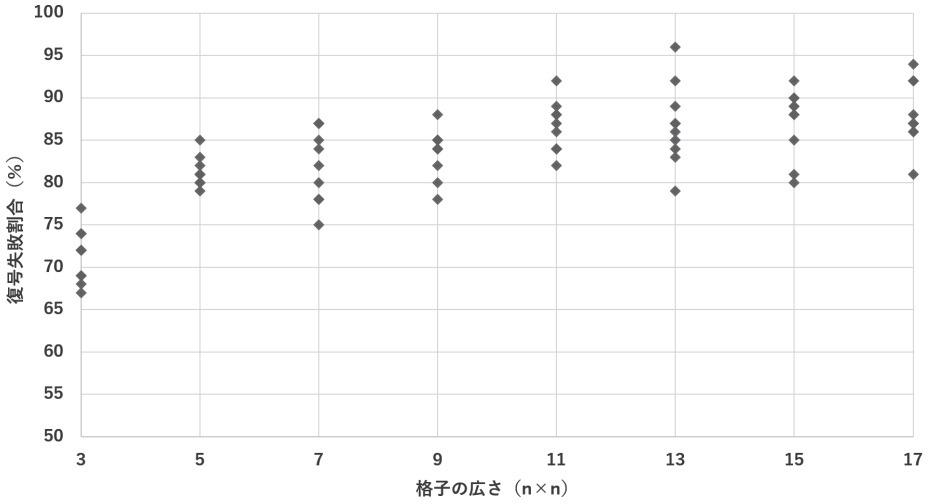


図 16　加算における格子と復号失敗割合の関係（異なる暗号文加算）

指数近似をおこなうと、$y = 100 - 27.73e^{-0.06x}$ であった。そのときの決定係数は 0.48 であった。

第 6 項　考察（格子）

決定係数は同じ暗号文加算の場合に比べて低く、指数近似による近似曲線の精度が低かった。実験 1 の結果より暗号文を都度取り換えた方が復号失敗がおこりにくくなるということから、格子を増やしたのにも関わらず実験 2 に比べて全体に復号失敗割合が大きくなりにくくなっていることを説明できる。実験 2 に比べて緩やかで、指数近似の精度も低く、100%に漸近しているとは言えないが、格子を大きくすることによって復号失敗割合が増加するということは言える。

第 8 節　加算のまとめ・考察

実験 1、及び追加実験 1,2 より、法 1024、格子 3×3 において、Bootstrap なしに GSW 式完全準同型暗号を用いる際は加算の計算回数を 254 回未満に抑えるのが妥当といえる。しかし、10%の復号失敗を許容してしまう。したがって、理論値として求めた、85 回を下回るように計算回数を制限することで、復号失敗となる確率を 0%にでき、より適切な計算回数設定といえる。また、理論値計算より計算回数は法の値に比例し、格子の大きさに反比例することがわかった。

実験 2、及び追加実験 3 より、法を 2 で対数をとったの値と復号失敗割合には、法の値が復号失敗割合に大きな変化を及ぼす関係性はみられなかった。したがって、法を大きくしても復号失敗割合に影響がないことがわかった。また格子を変化させた場合は、格子を大きくすると復号失敗割合は大きくなるという関係性がみられた。その関係性は、おおよそ指数関数によって近似され、100%へと近づくことが予想される。同じ暗号文を加算した場合と、その都度暗号文を取り直して加算実験を行った場合では、その都度暗号文を取り直した場合の方が復号失敗割合が抑えられることも分かった。

これらの計算可能回数及び要因の調査結果から総合して、実際応用を考える際には理論値である

$$x < \frac{q}{4m\sigma}$$

を満たすように格子 $n \times m$、法 q を定め、計算可能回数 x を確定させることが可能といえる。

第 9 節　乗算実験概要

加算と同様に GSW 式完全準同型暗号を用いて、Bootstrap 処理なしで何回まで計算が可能であるのか、また計算可能な回数を決める要因は何であるのかについての調査をおこなった。

第 1 項　目的

Bootstrap なしでの完全準同型暗号の応用を考えるため、乗算における計算可能回数とそれを決定する要因を調査すること。

第 2 項　研究手法

実験は乗算処理に対して、加算と同様の 2 種の調査をおこなった。ここでの乗算処理とは、同じ値を繰り返し乗算することをさしている。1 つ目の調査は、何回目まで復号失敗とならずに計算ができるのか、である。法および格子の設定は、加算実験との比較もかねて同様の値で設定した。加算と異なり、準同型性証明の計算式より、エラーベクトルは係数 R のみに依存していないことがわかっている。2 つ目の調査は、暗号を定義する際の法・格子の広さを変化させることによって法の中での乗算における復号失敗割合を調査した。それぞれの実験における具体的手順は実験の各節で説明する。

乗算実験では、加算と同様に復号失敗という語を用いる。乗算実験における復号失敗とは、平文 m_1, m_2 に対して、それぞれの暗号文を c_1, c_2 としたとき、$Dec(c_1 \cdot c_2) - (m_1 \cdot m_2) \neq 0$ のことである。すなわち、完全準同型性を持つのにも関わらず、暗号の特性によって暗号文同士の乗算結果を復号したものが、平文同士の正しい乗算の結果に一致しないことを復号失敗とよぶ。ただし、法で扱える値を超えてしまったことによる復号失敗は除く。また、Bootstrap を用いないケースに関して復号失敗を考えた。

第 3 項　Python による乗算計算の実装

本実験では GoogleColaboratory 環境下で Python(version3.10) を用いた。実装においては加算同様 Python の Numpy モジュールを使用した。乗算では加算と鍵生成、暗号化、復号の手順は同様であるが、計算処理の過程で異なる。そこで、以下に乗算 1 回分の計算手順を記す。ガジェットマトリックスの逆関数を求めるコードについては付録を参照されたい。

計算処理過程

1、暗号文 C_1, C_2 を生成する
2、C_1, C_2 どちらか一方を選択し、ガジェットマトリックスの逆関数によって $G^{-1}(C_2)$ を求める
3、$C_1 \cdot G^{-1}(C_2)$ を計算する

第 10 節　実験 1：乗算可能回数を求める

計算回数が何回まで可能であるのかを実験によって調査した。まず、加算と同様の鍵設定を行い、平文の値を変化させたときに複合失敗の発生割合がどのように変化するのかを調査した。

第 1 項　方法

1、GSW 式完全準同型暗号の作成
2、法を 1024、格子の大きさを 3×3、シグマ範囲は 1 と定義し鍵を生成した
3、2 について、平文を 1 として法を超えない範囲（1023 以下）でエラーが出ずに繰り返し同じ値を乗算できるか

どうかを調査した
4、2 について同様の鍵の値を用いて平文の値を 1〜10 で変化させ、調査した
5、鍵を生成し直して 2〜4 を 100 回繰り返した
6、2〜5 を 10 回繰り返し、散布図を作成した

第 2 項　結果

変化させたときの平文を横軸にとり、100 回中の複合失敗発生割合を縦軸に取ったときの散布図を図 17 に示す。

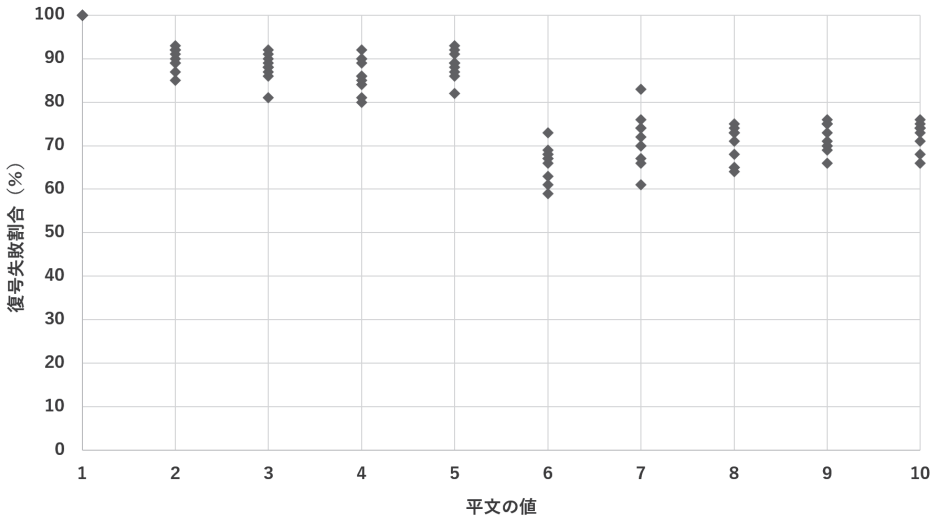


図 17　乗算における平文変化と復号失敗割合の関係

図 17 より、平文が 5 から 6 に変化する際に復号失敗割合がわずかに減少したものの、目立った変化が見られない。

第 3 項　考察

加算では、復号失敗割合が大きく変化した際の境界となる平文から、計算可能回数を検討し、調査をしたが今回の実験結果からは評価することは難しい。ただし、平文の値が 10 のとき、法 1024 の範囲で 10 は 3 回乗算できる。この、3 回乗算した際に発生する復号失敗割合が 50%を超えていることから、法 1024、格子 3×3 における計算可能回数は 3 回未満ではないかと考えられる。

第 11 節　追加調査１：直接的に計算可能回数を求める

加算と同様の方法では、法の限界を超えるよりも先に復号失敗が発生し、計算可能回数を正しくもとめることができなかった。そこで、復号失敗した各データについて何回目の計算で復号失敗したのかを調査をおこなった。特に、平文の値を 1 にすることで、1 の累乗が法の値を超えることない。そのため、暗号特性による乗算の復号失敗発生までの回数を正しく評価できると考えた。

第 1 項　方法

1、GSW 式完全準同型暗号の作成（本章第 3 項参照）
2、法を 1024、格子を 3×3、シグマ範囲を 1 と定義し鍵を生成した
3、平文を 1 として暗号化をおこない復号失敗が発生するまで繰り返し同じ暗号文を乗算した

4，復号失敗が発生した乗算の回数目を記録した
5、2〜4 の操作を 1000 回繰り返した

第 2 項　結果

1000 回分のデータをまとめたグラフを図 18 に示す。横軸は復号失敗が発生した際の計算回数目、縦軸はその計算回数目で復号失敗となったデータの標本数を表す。

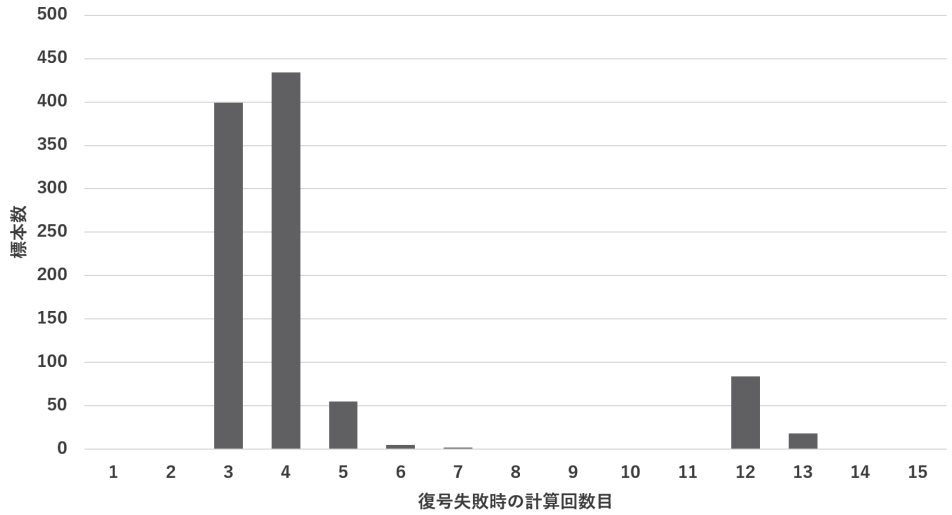


図 18　乗算における復号失敗割合の計算回数目（全標本数：1000）

図 18 の結果から、3 回未満の計算において復号失敗は発生していないことが読み取れる。また、3,4 回目での復号失敗が全標本中の 8 割以上をしめている一方で、3〜6 回目以外にも 12、13 回目でも復号失敗が発生している。

第 3 項　考察

12 回目、13 回目まで乗算が可能であったのはエラーベクトル e が全て 0 に近い小さな値であったのではないかと考えられる。加算では、計算回数を重ねるごとに R の値のみがノイズとして加算された。しかし、乗算 1 回分において、暗号文 C_1, C_2 に対し、$R_1 G^{-1}(C_2) - \mu_1 R_2$ という値がノイズになる。これを R' とおく。さらに、暗号文 C_3 を用いて乗算を行うことを考えると、ノイズは $R' G^{-1}(C_3) - \mu_1 \mu_2 R_3$ となる。これより、乗算ではノイズが積で増加するため、加算に比べノイズが大きくなりやすいことがわかる。その結果、加算の 254 回未満に比べて 3 回未満と、計算可能回数が大きく下回ったと考察される。このとき、ある値に 254 未満の値を乗算をする場合は加算でも代替することが可能である。つまり、3 回以上乗算が必要な処理は、法 1024、格子 3×3 の完全準同型暗号の場合、加算を用いた方が復号失敗なく計算できることがいえる。ただし、加算と乗算を組み合わせた場合における計算可能回数は、本実験の結果からは不明である。

第 12 節　計算可能な回数の理論値の計算

第 1 項　理論値の計算

実験 1、及び追加調査 1 を通して求めた計算可能な回数の実験値は 3 回未満であった。本節では、ワーストケースを仮定し、その際における計算可能回数を暗号文計算の式をもとに求める。

法 q、格子 $n \times m$、シグマ範囲 σ と定義した場合、実験で用いたアルゴリズムにおいて、公開鍵は $A \in \mathbb{Z}_q^{(n+1)\times m}$、秘密鍵は ${}^t s \in \mathbb{Z}_q^{n+1}$ である。また、${}^t s \cdot A$ は平均が 0、標準偏差が σ 範囲の正規分布からサンプルした小さなエ

ラーベクトル ${}^t e$ に等しい。

このとき、平文 μ の暗号文 $C = \mu G - AR$ を k 回乗算した後に、復号することを考える。R は 0 と 1 で構成されるランダムな行列で、G はガジェットマトリックスである。

k 回目の乗算処理における暗号文のノイズ部分を求めたい。一回の乗算でのノイズは $C_1 G^{-1}(C_2) = \mu_1\mu_2 G - A(R_1 G^{-1}(C_2) + \mu_1 R_2)$ より、$R_1 G^{-1}(C_2) + \mu_1 R_2$ である。これを $R^{(2)}$ とおくと、乗算を重ねた際のノイズは以下のように変化する

$$\begin{cases} R^{(2)} & = R_1 G^{-1}(C_2) + \mu_1 R_2 \\ R^{(3)} & = R^{(2)} G^{-1}(C_3) + \mu_1\mu_2 R_3 \\ R^{(4)} & = R^{(3)} G^{-1}(C_4) + \mu_1\mu_2\mu_3 R_4 \\ & \vdots \\ R^{(k+1)} & = R^{(k)} G^{-1}(C_{k+1}) + \mu_1 \cdot \mu_2 \cdots \mu_k \cdot R_{k+1} \end{cases}$$

ここで、計算回数各 k 回目でノイズが最大になることを考える。

k 回目の乗算処理におけるノイズは、$R^{(k)} G^{-1}(C_{k+1}) + \mu_1 \cdot \mu_2 \cdots \mu_n \cdot R_{k+1}$ で表される。このとき、$\mu_1 \cdot \mu_2 \cdots \mu_k \cdot R_{k+1}$ が最大になるよう仮定するとノイズは最大になる。各 $R^{(k)}$ についても同様の仮定をする。

このとき、各 $G^{-1}(C_k)$ は、$l = \log_2 q$ とすると、$(n+1)l \times (n+1)l$ の大きさの格子である。G の定義、および $G \cdot G^{-1}(C) = C$ より $G^{-1}(C_k)$ の各要素は 0、または 1 である。ここで各暗号文 $C_k^{(n+1)\times(n+1)l}$ の各要素値は法範囲での任意の値であるから、$G^{-1}(C_k)$ の要素が最大となるときを仮定できる。よって、全て 1 と仮定する。$R_k^{m\times(n+1)l}$ についても同様に全て 1 と仮定する。

以上の仮定によって得られる k 回目のノイズを $R_{max}^{(k+1)}$ とする。このとき、$R_{max}^{(k+1)}$ のサイズは $m \times (n+1)l$ で、各要素は

$$\{(n+1)l\}^k + \{(n+1)l\}^{k-1} + \{(n+1)l\}^{k-2} + \cdots + \{(n+1)l\} + 1$$

となる。このとき、

$$\{(n+1)l\}^k + \{(n+1)l\}^{k-1} + \{(n+1)l\}^{k-2} + \cdots + \{(n+1)l\} + 1 < 2\{(n+1)l\}^k$$

であるから、ワーストケース時のノイズ $R_{max}^{(k+1)}$ は $2\{(n+1)l\}$ とする。

k 回目で復号可能とは、k 回目の暗号文に秘密鍵をかけた値が、ガジェットマトリックスの評価によって正しく計算されるときである。よって、$R_{max}^{(k+1)}$ に ${}^t e$ をかけた値の各要素 K が $-q/4 < K < q/4$ のときになる。エラーベクトル ${}^t e$ の各要素を統一して σ と表すこととすると、$K = m2\{(n+1)l\}^k\sigma$ となる。よって、$|2m\{(n+1)l\}^k| < q/4$ をといて、ワーストケースにおける計算可能回数 k は

$$k < \log_{(n+1)l} \frac{q}{8\sigma m}$$

と表せる。

実験 1 での値、法 1024、格子 3×3 を上の式に代入すると計算可能回数 k は、

$$k < \log_{(n+1)l} \frac{q}{8\sigma m} = \log_{(3+1)\cdot 10} \frac{1024}{8 \cdot 1 \cdot 3} \approx 1.02$$

よって実験 1 状況下における乗算可能回数の理論値は、1 回は保証できることがわかった。

第 2 項　理論値と実験値の考察

理論値を下回った実験値が存在しないことから両方の検証が正しいと判断される。加算においては実験値と理論値の間に 100 程度の差があった。しかし、乗算では差がないことも特徴といえる。これは、加算と異なりそもそもの計算可能回数がどちらも小さいこと、ノイズが単純な R の比例関数ではないこと、が原因と考えられる。実験値、理論値 2 つの結果から、乗算の計算可能回数は極端に少ないことがわかる。乗算を実行するためには法 1024、格子が 3×3 では不十分である。また、乗算ではなく加算で乗算処理を実行する方が復号失敗を減らして計算をおこなえる可能性がある。

実際に、格子が 3×3、$\sigma_{max} = 1$ で固定し、法のみを変化させて理論値の式に代入すると図 19 のようになる。法増加に伴って、計算可能回数は加算に比べ増加しずらいことが読み取れる。これは、法を大きくする際に理論値結果より $(n+1)l$ で対数をとるためであると考えられる。逆に格子は、大きくした際に底を大きくし、真数を小さくすることから、計算可能回数への影響が大きい。

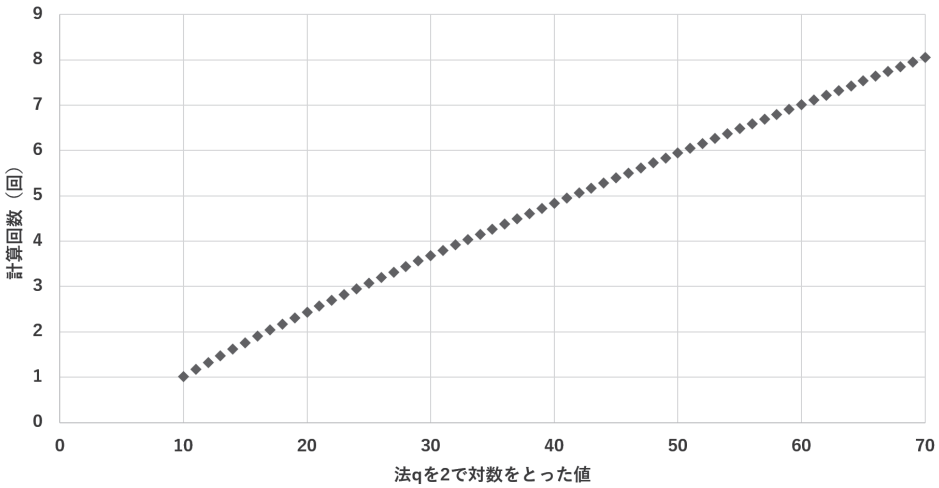


図 19　理論値における法変化と計算可能回数の関係

第 3 項　実験 2 にむけた仮説

法 q、格子 $m \times n$ において計算可能回数 k は $k < \log_{(n+1)l} \frac{q}{4\sigma m}$ であることから、格子を小さく、法を大きくすることが必要だと考えられる。しかし、法を大きくする際に $(n+1)l$ で対数をとることから、法変化は計算可能回数の変化に影響が少ないのではないかと考えられる。

第 13 節　実験 2：法と格子を変化させたときの復号失敗割合

実験 1 では、計算可能な回数を実験、及びワーストケース仮定下での理論値計算によって求めた。実験 2 では、暗号の鍵を生成する要素である法と格子を変化させ、加算同様に復号失敗割合にどのような変化があらわれるのかを観察した。ただし平文の値は加算実験時と異なり、10 とした。平文の値は 1 で設定すると、復号失敗割合が必ず 100%になるため 1 以外の値を設定する必要があった。そのため、法の値から実行される累乗計算が何回であるかを把握しやすい 10 を平文に採択した。

第 1 項　方法（法）

1、GSW 式完全準同型暗号の格子の大きさを 3×3、平文の値を 10 に設定した
2、法の大きさを 2^3 で定義した

3、暗号の鍵・暗号文を生成した
4、平文の値が、2 で設定した法の値を超えない範囲で、繰り返し加算を行った
5、2〜4 を 100 回行い、100 回中で法を超えない範囲にもかかわらず復号失敗が発生した標本数を数えた。その標本数を、その回数における復号失敗割合とした
6、2〜5 を 10 回繰り返した
7、2 について法の値を $2^4, 2^5, ..., 2^{22}$ と変化させ他の法の値についても同様に 2〜6 を行った

第 2 項　結果（法）

変化させた法の値を 2 で対数をとったものを横軸に、復号失敗発生割合を縦軸に取った場合の散布図を図 20 に示す。

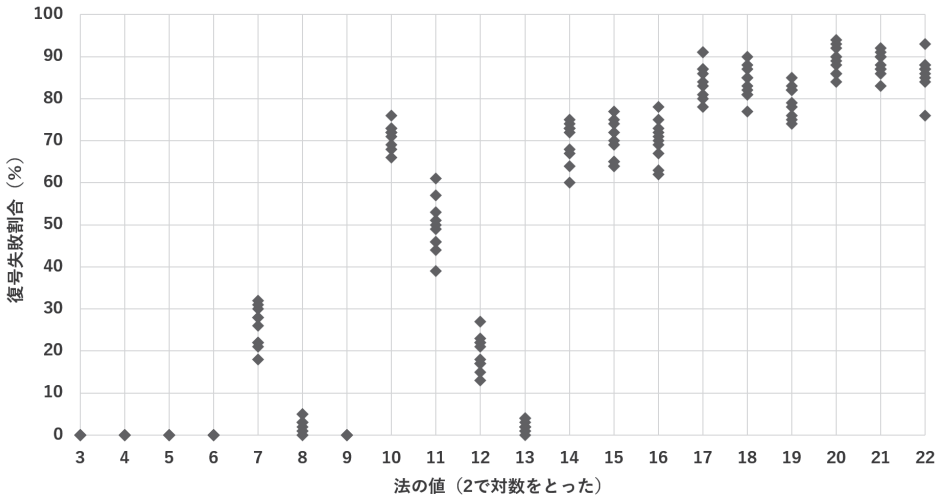


図 20　乗算における法と復号失敗割合の関係

図 20 より、法が $2^9, 2^{13}$ において復号失敗割合が少なくなっていることがわかる。また、2^{13} までは法を 2 で対数をとった値と復号失敗割合に何らかの関係性がみられるが、14 以降では、その規則性がみられないことが読み取れる。

第 3 項　考察（法）

$2^6 = 64$、$2^7 = 128$、$2^9 = 512$、$2^{10} = 1024$ より、6 まで復号失敗割合が 0%となるのは、そもそも 10 を乗算できないためである。7〜9 にかけては 10 を 2 回乗算できる。この場合、法の値が大きい 9 が復号失敗割合が最も少なくなった。これは理論値より、計算可能回数が増加したためである。10〜13 にかけては 10 を 3 回乗算している。よって、9 から 10 に変化した際に復号失敗割合が増加したと考えられる。その際も 7〜9 と同様に、法の値を大きくするにつれて計算可能回数が増加したためであるといえる。しかし、14 以降では規則性がみられない。これは、計算可能回数が $(n+1)l$ で対数をとった値であるためと考えられる。法増加による必要な計算実行回数の増加に比べて、計算可能回数が増えづらいためである。そこで、追加実験にて、計算回数を固定した場合における、法変化（2^{14} 以上) が復号失敗割合に与える影響を調査した。復号失敗割合が 7〜9、10〜13 と同じ挙動を見せた場合、法を大きくすることによって復号失敗割合は法を 2 で対数をとった値に比例するように、規則的に減少することがいえる。

第 4 項　方法（格子）

1、GSW 式完全準同型暗号の平文を 10、法を 2^{13} に設定した
2、格子の広さを 3×3 で定義した

3、暗号の鍵・暗号文を生成した
4、平文の値が法の値を超えない範囲で、繰り返し加算を行った
5、2〜4 を 100 回行い、100 回中で法を超えない範囲にもかかわらず復号失敗が発生した標本数を数えた。その標本数を、その計算回数における復号失敗割合とした
6、2〜5 を 10 回繰り返した
7、2 について格子の値を $5\times5, 7\times7, \ldots, 31\times31$ と変化させ他の格子の広さについても同様に 2〜6 を行った

第 5 項　結果（格子）

変化させた格子の広さを横軸に、復号失敗割合を縦軸に取った場合の散布図を図 21 に示す。

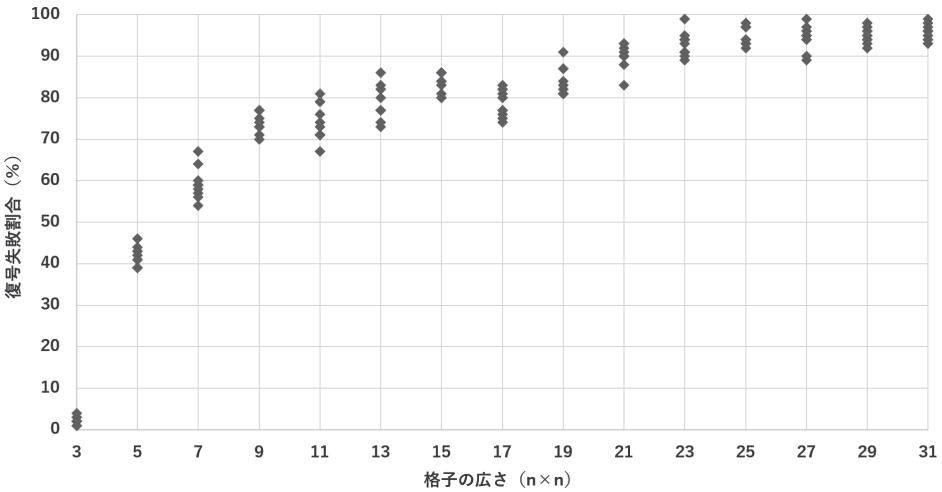


図 21　乗算における格子と復号失敗割合の関係

格子を増加させると図 21 のように復号失敗割合が増加していくことが読み取れる。このとき、指数近似をおこなうと $y = 100 - 96.15e^{-0.11x}$ であった。また、このときの決定係数 R^2 は 0.848 であった。

第 6 項　考察（格子）

格子を増加させると、加算と同様、100%に漸近するように復号失敗割合が増加している。しかし、加算に比べて漸近の動向は穏やかである。これは計算可能回数が対数をとっているため、法変化の場合と同様に変化しづらくなっているのではないかと考えられる。3×3 から 5×5 へと少し格子の大きさを変えると、復号失敗割合が 40%程度、急激に上昇した。したがって、格子が復号失敗割合が抑えられた状況に設定されているとき、格子の広さを増加させるのは正確性の面から問題が大きいといえる。

第 14 節　追加調査 2：計算回数を固定した場合の法変化と復号失敗割合

実験 2 の法を変化させた場合において、2^{13} 以降での規則性がみられなかったが、2^{13} 以前の規則性から、何らかの規則性があることが考えられる。そこで、本実験では計算実行回数を固定した状況下で法を変化させたときの復号失敗割合の変化を観察する。

第 1 項　方法

1、GSW 式完全準同型暗号の格子の大きさを 3×3、平文の値を 10 に設定した
2、法の大きさを 2^{14} で定義した

3、暗号の鍵・暗号文を生成した

4、4 回乗算をおこなう

5、2〜4 を 100 回行い、100 回中で復号失敗が発生した標本数を数えた。その標本数を、その回数における復号失敗割合とした

6、2〜5 を 10 回繰り返した

7、2 について法の値を $2^{14}, 2^{15}, \ldots$ と変化させ他の法の値についても同様に 2〜6 をおこなった 8、4 の操作について計算回数を 5 回に増やした場合に対しても、1〜7 を同様におこなった

第 2 項　結果

4 回乗算した場合の、法変化と復号失敗割合の関係は図 22 のようになった。縦軸は復号失敗割合、横軸は法を 2 で対数をとった値をあらわす。

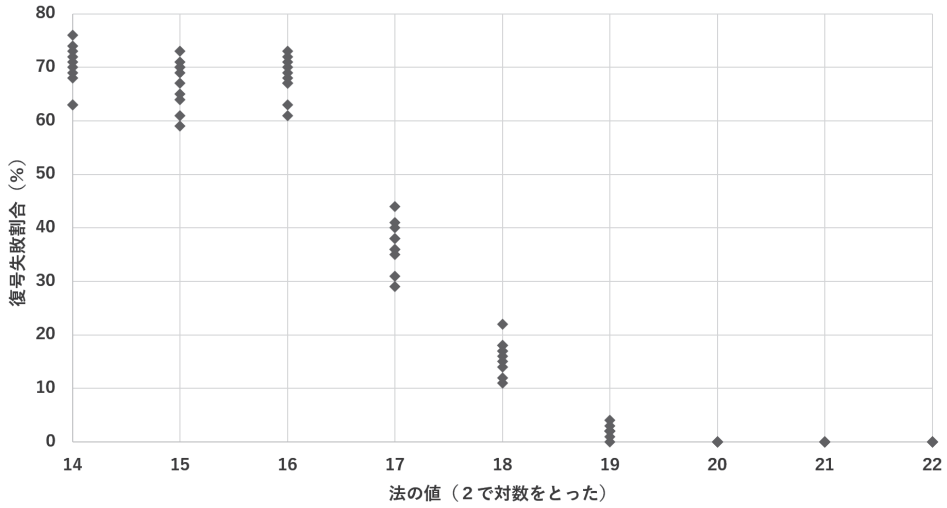


図 22　10 を 4 回乗算した時の復号失敗割合の関係

5 回乗算した場合の、法変化と復号失敗割合の関係は図 23 のようになった。縦軸は復号失敗割合、横軸は法を 2 で対数をとった値をあらわす。

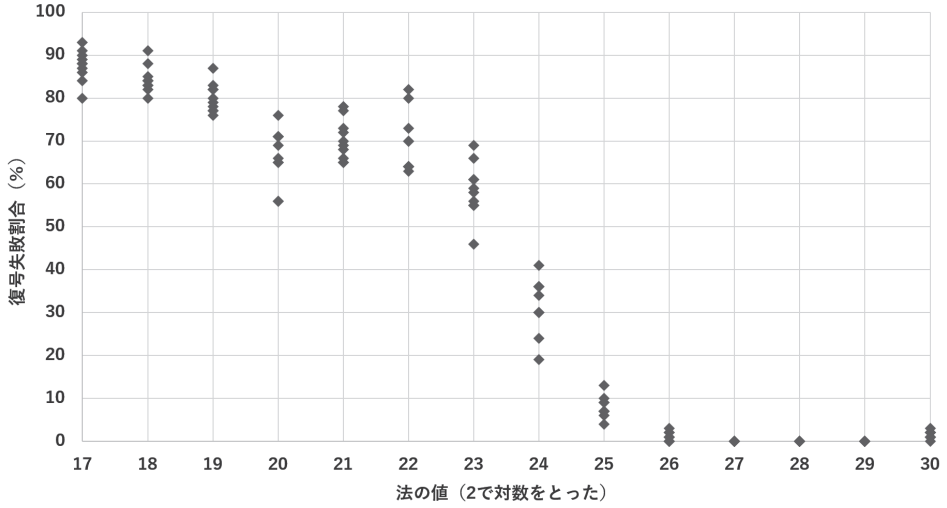


図 23　10 を 5 回乗算したときの復号失敗割合の関係

第 3 項　考察

グラフより、法を大きくすると復号失敗割合が減少することがわかった。また、計算回数を 1 回増やすために、法の値は 3 回から 4 回に増やす時よりも 4 回から 5 回に増やす場合の方が大きく増やす必要があることがわかる。すなわち、計算回数が増えるにつれて必要な法の値の増加の仕方が大きくなることがわかる。2 で対数をとった値に対して上記のことが言えるので、実際の法の値では計算回数を増加させるために法を増加させるのは非効率であるといえる。逆に、格子 3×3、平文 10 程度での計算を行う際は、実験 2 の結果より法範囲を最大限に活用するという観点において、法 2^{13} が最も効率的だといえる。

第 15 節　乗算のまとめ・考察

乗算の実験 1 および実験 2、理論値の計算から、乗算では計算可能回数がそもそも少なく、1 回分の計算を増加させるためには法や格子を大きく変化させる必要があるといえる。法 1024、格子 3×3 における計算可能回数は実験値では 2 回、理論値では 1 回である。また、法を大きくさせると計算可能回数を増やすことができるが、より多くの計算可能回数を得るためには、より大きく法を増加させる必要があるため非効率的である。格子は加算同様に、なるべく小さくとることが正確性の面で重要であるといえる。

以上のことから、複数回の乗算を完全準同型暗号で実装するには法を大きくとり、加算で代替する方がよいといえる。しかし、数回程度の乗算であれば法と格子の調整により実行することが可能である。

第 4 章　　エラー正規化処理なしの完全準同型暗号の活用提案

前章の実験より、完全準同型暗号は法および格子の調節によって計算可能な加算および乗算の回数を調節できる。よって、場合によってはエラー正規化処理である Bootstrap なしの状況下でも利用できる可能性があるといえる。そこで、本章では、完全準同型暗号は小規模な分析や解析に利用できることを提案する。その身近な例として小規模な学力試験への完全準同型暗号利用をあげ、具体的に説明する。

第 1 節　想定する状況設定

第 1 項　前章のまとめ

本章を考察する上では前章で分かった GSW 式完全準同型暗号の特徴が欠かせない。そこで、本項をもって前章の内容を整理する。

前章より分かったことは以下の 3 点である。

- 鍵生成の値（法と格子）の設定によって、復号失敗のない計算可能回数 x を導出可能である
- 加算計算回数は法増加に伴って増加しやすい（単純な比例関係である）
- 乗算回数は法増加に伴って増えにくい

第 2 項　具体的な状況設定

学校内等で行うテストや、学力試験等の結果を共有するケースを想定する。点数の採点部分については、完全準同型暗号を用いた照合一致処理等によって行えればよいが、今回のケースでは小規模な利用を想定しているため、あらかじめ答案が採点されたケースを考える。このとき、各人のテストの点数が採点され、なおかつ各人のデータは各個人のみ平文をみて確認できるものとする。また、テストの満点は 100 点、テストの受験人数は 300 人であったと仮定する。この場合に、各人が他の人に自身の点数を秘匿した状況下でテストの平均、及び分散を求めることを考える。

第 2 節　平均値についての考察

第 1 項　定義する値について

前章より、GSW 式完全準同型暗号の計算可能な回数の限界は、実験値 254 回未満、理論値 85 回未満である。また、法の値を変化させても復号失敗割合に大きな変化はないが、格子を大きくすると復号失敗割合が大きくなっていくことが分かった。このことから、復号失敗なく計算するために、平均値を計算するプログラム内では計算回数を $x < q/4m\sigma$ 回未満に抑え、法の値は、できる限り大きな値が計算できるように大きく、格子の値は大きくしすぎないことが必要といえる。ここで 100 点満点の試験について、$m = 3$、$\sigma = 1$ で計算回数 x 回の加算で取りうる最大値は $100 \times x = 100x$ である。よって、復号失敗が出ない範囲で最大限に計算するためには法の値は $q/12 > 100x$ を満たす q が妥当といえる。しかし、法の値を大きくすると暗号文のビットが長くなり複雑になり、処理にかかる計算時間が長くなることが [23] よりわかっている。そこで、法の値は応用例 2 において復号をせずに、計算が可能な $2^{13} = 8192$ を用い、格子については実験 1 と同様の 3×3 で定義し実装するのが良いと考える。

上記のように定義した場合に、複数の平均値の計算実装について考察した。

第 2 項　平均値算出への応用例１

まず、実装方法の１つ目は単純にテストデータを加算し平均を求める方法である。まず、法の値を超えない範囲で加算をできる人数に区分する。区分別に加算を行い、それぞれ法の値を超える直前に復号をおこなう。そして、各区分に含まれる人数で割り、各区分における平均を求める方法である。各区分の平均値は、複数名のデータによって算出された値であるから、個々人のデータは直接的には含まれていない。ここでの安全性は、個々のデータを守るという点で各区分に含まれる人数に由来していると考えられる。最終、復号された状態の各区分の平均値の平均を求める。これによって、全データの平均値を計算することが可能となる。

100 点満点のテスト、法 8192 で、この手法を用いた場合を考える。最悪のケースである区分に含まれる全員が 100 点を取ったときを考慮すると、80 人分ずつ合計の値を求めて、復号しリストに格納することになる。このときの 300 人規模のテストデータを計算する実装手順は以下のように考えられる。区分の分け方は、300 人がより均等に区分できるほど各区分を復号する際の加算データに含まれる人数が多く、安全性が高い、といえる。従って、80 人以下で 300 人を均等な人数で区分でき、かつ最大となる各区分の人数は $300 = 2^2 \cdot 5^2 \cdot 3 = 4 \cdot 75$ より、75 人である。

１、300 人の試験結果を（各データは GSW 式完全準同型暗号によって暗号化されている）75 人ずつランダムに区分する
２、各区分について暗号文を加算し復号する
３、復号した値を各区分の人数で割る
４、３で求めた各区分の平均値を加算し、区分数で割る

この方法を実際に行った場合は、サーバへの送信者側にしか復号ができないため、割り算の処理を送信者側でおこない加算しなければならない。

第 3 項　平均値算出への応用例２

次に、応用例 2 では仮平均を利用する。各値について、100 点満点のテストであるから、50 からの差分を計算し、それらを合計することを考えた。この方法においては、妥当な仮平均の平均値（法が 1024 のときであれば真の平均値と仮平均の差が $8192 \div 300 \fallingdotseq 27.3$ 以内）を選ぶことで、平文の合計値が法の値を超えずに加算できる。このとき、法の値を大きくするほど、妥当な仮平均のとれる値の範囲はおおきくなる。ただし、前章の実験より、復号失敗なく計算できる計算可能な回数が $x < q/4m\ \sigma = 8192/4 \cdot 3 \cdot 1 = 8192/12 \approx 682$ のため、最初の仮平均からの各値の引き算を含めて加算できる人数は、$682 \div 2 = 341$ より 341 人で、その時点で一度復号をした方がよいといえる。今回考えるケースでは、300 人規模であるから適切な仮平均の値を選択することで復号せずに平均値を求めることができる。

１、仮平均の平均値を決定する（平均を検討する）
２、１で決定した仮平均の平均値を暗号化
３、各値（各データは GSW 式完全準同型暗号によって暗号化されている）から２の値を引く
４、３で求めたすべての値を加算する
５、４の結果を復号し、人数で割り、その値を仮平均の値に加算し真の平均値が求まる

ただし、この方法を実際に行った場合は、サーバへの送信者側にしか復号ができないため、割り算の処理を送信者側でおこなってもらう必要がある。

第 4 項　応用例１と２の比較

検討した応用例１、２の特徴について、各々のメリット・デメリットを表１にまとめた。

表１　応用例１と２の比較（平均算出法）

	応用例１	応用例２
メリット	・計算回数を少なくできる	・法を極端に大きくしなくてよい
デメリット	・法の値が大きくなってしまう	・人数が増えた場合に適当な仮平均の 平均値を選ぶことが困難になる ・計算回数が多い（データ数の 2 倍計算が必要）

応用例１では 300 人分の試験結果をそのまま計算するため、加算可能な回数が、テストデータの加算値が法を超えないことに由来している。そのため、k 人が受験したテストで、全員の加算できる計算回数まで計算できる法を選んだ場合、$100 \cdot k < q$ となる法 q を選ぶことになる。これより法の値が小さい場合、計算可能な回数の限界まで加算することなく法の制限によって計算ができない。

対して、応用例２の場合、適当な仮平均の平均値（q/k の範囲内の値）を選択することで、より応用例１に比べてより多くの人数で、また、法を超えずに小さな値での加算が可能となる。ただし、この値の選択は、同じ法の値に対して、人数が増えれば増えるほど選択できる範囲が狭くなるため、選ぶことが困難になる。仮に適当な値を選択できた場合、暗号文を用いた部分で必要な計算回数は設定した値からの差を、テストデータ各値で求めて、それらの加算をおこなうため、人数の 2 倍になる。加算値が法を超えないことから、そのときの計算回数はエラーが出ずに計算ができる回数に由来する。よって、k 人が受験した時、復号失敗なく計算可能な回数の理論値計算より $2k < x < q/12$、すなわち、$k < q/24$ となる q を選べばよい。つまり、応用例 2 は応用例１に比べて、法を極端に大きくせずとも復号失敗せずに計算ができる限界回数まで計算できるといえる。

しかし、応用例１は計算回数として、各区分内で暗号文を加算するとき（データの個数分の暗号文加算）、また各区分の加算値の復号結果から平均値を求める計算（区分数分の割り算）、そして各区分の平均値から全体の平均値を求める計算（区分数分の加算と割り算 1 回）、と応用例２に比べて少ない回数で計算が可能であるというメリットがある。

第 3 節　分散についての考察

第 1 項　定義する値について

分散を求めるにあたって、前節でもとめた平均が必要である。よって、平均が前節の方法によって求められている場合を想定する。分散には 2 通りの計算方法がある。一つ目は、平均との差の 2 乗合計を求め、平均をとる手法である。2 つ目は、各値を 2 乗した値の平均をとり、平均の 2 乗をひく手法である。

1 つ目の方法では、各値に対して平均との差を求めて、2 乗の計算を行うので、ノイズ R が、最大 $2R$ となったものを 2 乗することとなる。ここで、乗算の理論値は $k < \log_{(n+1)l} \frac{q}{8\sigma m}$ で、理論値計算の際に仮定した R_1 の値が 2 倍になっていると考えればよいので、このときの、一回乗算した後の各要素のノイズは $\{2 \cdot 2(n+1)l\}^1$ である。（q の法、$n \times m$ の格子、$l = \log_2 q$）さらに、この値を 300 人分足し合わせることから、ノイズの値は $1200(n+1)l$ となる。よって、$1200m(n+1)l\sigma < \frac{q}{4}$ となるとき、復号可能である。格子のサイズを 3×3 で最も小さくとり、$\sigma = 1$ とすると、$57600 < \frac{q}{\log_2 q}$ を満たす q が必要である。このとき、q を求めると、およそ $1160000 > 1048576 = 2^{21}$ であるから、$2^{22} = 2097152$ の q が必要である。また、2 乗した値を足し合わせるので最

大で $50^2 \times 300 = 750000$ の法、つまり $2^{19} = 524288 < 750000 < 2^{20}$ より 2^{20} の法が必要である。以上 2 つの条件から、法は 2^{22} 必要であるといえる。

2 つ目の方法では、2 乗した値を足し合わせることから、$100^2 \times 300 = 3000000$ が最大値で、2^{22} の法が最大で必要である。このとき格子は復号失敗が少なくなるように、1 つ目の方法と同様 3×3 で小さくとるとよい。よって、こちらも必要な法の大きさは 2^{22} といえる。

方法 1 を用いた応用方法を応用例 1、方法 2 を用いた応用方法を応用例 2 として、それぞれ具体的な方法を以下のように考えた。

第 2 項　分散算出への応用例 1

分散算出への応用例の 1 つ目は、差の 2 乗を合計し、平均をとる手法である。ここでの平均をとる直前の、2 乗の差の合計データは、各人のデータの値が秘匿されているため公開されてもよいデータと考えられる。そこで、平均をとる直前の 2 乗の差の合計までを目標に応用を考える。

まず、法の範囲が大きすぎるため加算同様に人数を区分して合計値を求める方法を扱う。各区分に含まれる人をランダムに設定することで各データの秘匿性は担保できると考えた。区分する人数は加算同様の理由から 75 人で区分する。このとき、必要な法のサイズは、ノイズによる復号失敗を防止するために 2^{22}、計算上の最大値より法をおおきくとるために 2^{20} 必要である。よって 2^{22} で考える。格子は、乗算等の処理も含み、ノイズが大きくなりやすいので 3×3 で考える。

1，300 人の試験結果を（各データは GSW 式完全準同型暗号によって暗号化されている）75 人ずつランダムに区分する
2，各区分について平均値の暗号化データと各暗号文の差をとり、2 乗を計算し、加算する
3，復号した値を各区分の人数で割る
4，3で求めた各区分の平均値を加算し、区分数で割る

ただし、この方法を実際に行った場合は、サーバへの送信者側にしか復号ができないため、割り算の処理を送信者側でおこなってもらう必要がある。

第 3 項　分散算出への応用例 2

分散算出への応用例の 2 つ目は各値を 2 乗した値の平均と、平均の 2 乗を用いて計算する手法である。ここで、平均値は各人のデータが直接的に含まれないため秘匿する必要がない。さらに、各値の 2 乗の合計値も同様の理由で秘匿する必要がない。よって、2 乗した値を加算することを目標に考える。

まず、法の範囲が大きすぎるので応用例 1 と同様に 75 人に区分する方法を用いる。すると考えられる最大値は $100^2 \times 75 = 750000$ より、法は 2^{20} 必要である。ここで、加算のみの計算が 75 回できれば十分であるので、加算計算可能回数の理論値より $75 < \frac{2^{20}}{4m}$ となる。つまり、格子のサイズ $n \times m$ は最大で $m < 3495.2$ となる。よって、格子のサイズはある程度大きくても問題ないと考えられる。

1，各人が自身のテスト結果の 2 乗を計算する
2，300 人の試験結果を（各データは GSW 式完全準同型暗号によって暗号化されている）75 人ずつランダムに区分する
3，各区分についてデータを加算する
4，復号した値を各区分の人数で割る
5，3で求めた各区分の平均値を加算し、区分数で割る

ただし、この方法を実際に行った場合は、サーバへの送信者側にしか復号ができないため、割り算の処理を送信

者側でおこなってもらう必要がある。

第 4 項　応用例１と２の比較

検討した応用例１、２の特徴について、各々のメリット・デメリットを表 2 にまとめた。

表 2　応用例１と２の比較（分散算出法）

	応用例１	応用例２
メリット	・計算可能回数と扱う法範囲が近い	・安全性が向上する
デメリット	・ノイズの値が膨らみやすい	・計算可能回数で要する法に比べ計算自体に必要な法範囲が大きい

応用例１、応用例２に共通して、法が平均値の計算に比べて相当大きく必要である。また、分割することで法を減らせると考えたが、乗算が関係して、数 bit 程度の変化しか得られなかった。応用例１では乗算の処理を含んでいるため、応用例２に比べてノイズが大きくなりやすい傾向にある。その一方で、応用例２より小さな値でよいため、計算可能限界の直前まで法範囲を利用することや、bit に無駄なくアルゴリズムを実装するといった面において優れているといえる。

対して、応用例２では加算のみの計算であるから、ノイズによる復号失敗の制限より、実行するために必要な法がはるかに大きい。よって、法範囲にあまりが生じる。しかし、格子またはエラーベクトルを、応用例１に比べて大きくとることができるというメリットがある。エラーベクトルを大きくとれるということは、格子暗号における安全性を向上させることにつながる。

以上より、法に関していえば応用例１が、安全性に関しては応用例２の方がよいと考えられる。

第 4 節　将来的応用の可能性

300 人規模の学力試験における計算を考察することで、法、格子を設定することによって完全準同型暗号は Bootstrap なしでも利用可能であるとわかった。また、加算および乗算のみの応用方法では、復号失敗を防止することより、法範囲での計算のための法設定の方が、必要な値が大きいことがわかった。よって法が極端に大きくならないデータの解析、すなわち加算結果や乗算結果が小さくてもよい小規模なものに対して有用であるといえる。小規模な分析として学力試験を例に考えたが、学力試験以外にもアンケート調査等にも応用が可能だ。アンケート調査で、5 択程度であれば、より多くの人数、より大規模な分析が可能だと考えられる。小さな値で、多くの回数計算をおこなうので復号失敗による法制限の課題が出てくることも考えられる。

今回の 75 人区分法（少人数で区分して求める手法）は大規模にも応用できると考える。途中で復号処理が必要になるが、部分的に完全準同形暗号を用いることで高度な安全性を付与したまま平均値、および分散を算出できると考察する。例えば、数万人規模でのアンケートや学力試験を行う際を考える。まず、今回の手法と同様、ランダムに集団を区分する。そして、それらの小規模な集団の中で計算可能回数と法範囲によって定まる計算可能な中で平均値と分散を算出する。求めた平均値および分散は、各人個々のデータとは異なり、全体を代表する値であるから公開されても問題はない。そのため、各集団ごとにえられた平均値と分散は復号して、とりまとめ、全体の平均値および分散を求めることができる。このとき、完全準同形暗号による計算のため、各人のデータは秘匿されている。さらに、復号される小規模区分での平均および分散は、ランダムに定められた区分であるので、集団内のどの人のデータが、どの小集団に含まれているか分からず、高度な秘匿性を担保できる。このようにして、小規模分析であっても、大規模な集団に対して実装可能なアルゴリズムを考えることができる。このアルゴリズムにおいては、区分する人の数、扱うデータの最大値、処理に伴う計算回数によって、学力試験での分析のように最適な法と格子を定めることが重要となると考えられる。

完全準同型暗号を用いることによって法・格子の大きさを変えることで十分に計算回数を得ることができる。乗算では Bootstrap なしの場合での復号失敗は発生しやすい一方で、加算によって代替することでその発生割合を減少させることができる。今回は、データを分割し、複数の計算方法を用いることで応用の工夫を考えたが、より効率的な計算方法やアルゴリズムを探すことによって必要な法の数を減らせる可能性がある。以上のことから、完全準同型暗号は 300 人、100 点満点の学力試験等の小規模なデータの解析において十分に利活用が可能である。また、これらの応用によって、小規模分析を拡張した大規模分析に対しても各個人のプライバシーの高いデータを互いに知ることなく共有し、全体の概要結果（平均値および分散）を確認することができる。この暗号を用いることが、アンケート等の個人のデータが他人に見られないという安全性の保障につながるといえる。

第 5 章　研究のまとめと今後の展望

第 1 節　結論

Bootstrap 処理なしの完全準同型暗号において、法 q、格子 $n \times m$、エラーベクトルの各要素値 σ であるとき、加算では $\frac{q}{4m\sigma}$ 回未満、乗算では $\log_{(n+1)l} \frac{q}{8\sigma m}$ 回未満の計算可能回数が保障されることがわかった。これらの計算回数はワーストケース仮定下で求めた。対して、すべてをランダムにした実験値においては、これらの値よりも倍程度の計算が可能であった。

加算における法は、計算可能回数を増加させる役割を持つ一方、法による実行範囲の限界まで加算を行った際の復号失敗割合そのものには影響しない。対して乗算における法は、計算可能回数を増加させ、法による実行範囲の限界まで乗算をおこなった際の復号失敗割合も減少させる。しかし、ある一定回数以上の計算回数を増やすためには、法を極端に大きくとる必要があり、計算回数を増やすという側面で非効率的である。

加算における格子は、計算可能回数を減少させ、法による実行範囲の限界まで加算を行った際に復号失敗割合を増加させる働きがある。乗算における格子も同様に、計算可能回数を減少させ、復号失敗割合を増加させる働きがある。また、双方、格子の値を増加させると復号失敗割合が 100%に漸近するように近づいていく。

以上より、法は計算実行範囲と計算可能回数を決定するために設定し、その条件下で格子を小さくすることが Bootstrap 処理なしの完全準同形暗号の応用において重要である。そこで、300 人規模の学力試験を例に、平均値と分散に関する応用手法を考察した。結果、加算のみ使用した場合の実装では、法による実行範囲の限界設定をおこなう際に、復号失敗しないために必要な法の値よりも大きな値を必要とする。乗算を使用すると、ノイズの増加が激しいことが分かった。しかし、法と格子の設定によって平均値と分散は十分に計算可能であった。よって、Bootstrap なしの完全準同型暗号は 300 人規模の学力試験の分析に十分利用可能である。

Bootsatrap なしの完全準同型暗号は法と格子の設定により、小規模なデータを解析できる計算能力を持っている。特に、アンケート等の加算計算の多い分析での利用では多数のデータを秘匿したまま回収、分析ができる。これらを応用して、ランダム区分による少人数の小規模分析を積み重ね、大規模な分析にも将来的応用の可能性があるといえる。

第 2 節　今後の展望

今回の研究では、正確性に着目して研究をおこなった。しかし、実際の利用では正確性のほかに、効率性、安全性についても調査が必要だ。検証を通して、法や格子の変化が安全性・効率性にどの程度影響するのかを考えなくてはならない。それらの点についても実験・計算を通して評価が行えればよいと考える。

他にも、本研究ではワーストケース仮定を用いることで理論値を算出した。ワーストケース仮定では最も復号失敗が起こりやすい場合を仮定した。しかし、それでは乗算の計算回数等で十分な計算実行回数が得られない。加算でも、実験値に比べて半分以下の計算回数しかおこなえない。そこで、微小な復号失敗の可能性が許容される場合についても検証をおこないたい。法や格子によって、ある計算回数での復号失敗確率が求められないかを考えていきたい。

さらに、将来的応用を考える際、本論文では、小規模分析において最も効率的な人数と点数の関係を算出するまでに至ることができなかった。今後は、今回独自算出した理論値、扱うデータの大きさをもとにして、もっとも最適な法、格子、区分人数の関係性について言及していきたい。

最後に、本論文ではアルゴリズムの作成のしやすさから、参考論文 [20] に記載されていた法の値を 2^l とした場合で作成した。しかし、安全性の観点から、実際応用する際はメルセンヌ数（$2^l - x$）などの 2^l に近い素数を法と

して用いた場合について議論しなくてはならない。法を素数に近い値にすることで、除算をした際に復号時の評価で復号失敗の閾値になりづらく、除算での計算可能回数が増えるのではないかと考えられる。除算を効率的に実装できた場合、平方根をとったり、小さな値で計算したり、と応用の幅が広がるのではないかと考えられる。

謝辞

本研究を進めるのにあたり、大阪大学大学院工学研究科講師の王イントウ先生、並びに同学科教授の宮地充子先生には終始熱心なご指導をいただきました。心から感謝いたします。また、宮地研究室の皆様、大阪大学SEEDS 事務局様、神戸大学附属中等教育学校の中田雅之先生には本研究遂行にあたり多大なご助言、ご協力いただきました。本当にありがとうございました。

付録　GSW 式完全準同型暗号実装に用いたコード

分布作成のためのコード

一様分布

プログラム 1　Uniform-distribution

```
def uniform_distribution(q,n):
    return np.random.randint(0,q,size=n)
```

正規分布

プログラム 2　Normal-distribution

```
def normal_distribution(n):
    a=np.random.normal(0,sigma,size=n)
    return np.round(a).astype(int)
```

鍵生成に用いたコード

プログラム 3　GSW-style FHE Key Generation

```
def GSWgenerate(n,m,q,sig):
    sigma= sig/math.sqrt(2*math.pi)
    LWE_A=np.matrix(uniform_distribution(q,(n,m)))
    LWE_s=np.matrix(uniform_distribution(q,(1,n)))
    LWE_e=np.matrix(normal_distribution(m))
    LWE_b=(LWE_s*LWE_A+LWE_e)%q
    A=np.concatenate([LWE_A,LWE_b])
    s=np.c_[-LWE_s,1]
    return A,s
```

暗号化に用いたコード

プログラム 4　GSW-style FHE Encryption

```
def GSWenc(n,m,q,A,main_message):
    l=len(bin(q))-3
    g=np.zeros((n+1)*l)
    G=np.zeros(((n+1),(n+1)*l),int)
    for i in range(l):
        g[i]=2**(i)
    for k in range(n+1):
        G[k]=np.roll(g,l*k)
    R=np.matrix(uniform_distribution(2,(m,(n+1)*l)))
    Secret_message=(main_message*G-A*R)%q
    return Secret_message%q
```

復号に用いたコード

プログラム 5　GSW-style FHE Decryption

```
def GSWdec(n,q,s,Secret_message):
    l=len(bin(q))-3
    Dec_01=(s*Secret_message)%q
#ガジェットマトリックスの解読
    Dec_02=np.zeros((n+1))
    for h in range(n+1):
        ans=0
        for p in range(l):
            k=((h+1)*l-p)-1
            t=Dec_01[0,k]
            ss_0=t-ans*(2**(l-(p+1)))
            ss=ss_0%q
            if (q/4)<ss and ss<(3*q/4):
                sss=1
            else:
                sss=0
            ans+=(sss*(2**p))
#スカラー倍の解読
        Dec_02[h]=ans
    return Dec_02[n]
```

ガジェットマトリックスの逆関数

プログラム 6　GSW-style FHE Inverse Function

```
def G_inv(C,q,n):
    l=len(bin(q))-3
    n=n+1
    for i in range(n):
        G_s =np.zeros((l,n*l))
        for m in range(n*l):
            G_ss=[]
            nishin=bin(C[i,m])[2:]
            nishin_len=len(nishin)
            nishin_list=[int(x) for x in list(str(nishin))]
            zero_ad=l-nishin_len
            for p in range(zero_ad):
                G_ss.append(0)
            G_ss=G_ss+nishin_list
            G_ss.reverse()
            G_sst=np.array(G_ss).T
            G_s[:,m]=G_sst
        if i==0:
            G=G_s
        else:
            G=np.vstack((G,G_s))
```

```
22      return G
```

参考文献

[1] 池田響. 秘匿計算の将来を考える　-プライバシーを強化した試験結果の統計処理を考える-. 卒業論文一次論文, 神戸大学附属中等教育学校, 2021.

[2] NTT セキュアプラットフォーム研究所. 整数上完全準同型暗号の研究. NTT 技術ジャーナル, March, 2014.

[3] NTT コミュニケーションズ株式会社国立大学法人千葉大学医学部附属病院. 千葉大学と NTT Com,「秘密計算ディープラーニング」などの技術を活用した臨床データ分析の共同研究を開始. March, 2014.

[4] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. Cryptology ePrint Archive, Paper 2021/091, 2021. https://eprint.iacr.org/2021/091.

[5] CRYPTOEXPERTS. Post-quantum cryptography. https://www.cryptoexperts.com/. 2021 年 12 月 14 日閲覧.

[6] 株式会社富士通研究所. 世界初！暗号化したまま統計計算や生体認証などを可能にする準同型暗号の高速化技術を開発, August 28, 2013.

[7] 三菱電機情報技術総合研究所. 研究開発・技術一覧「関数型暗号」. https://www.mitsubishielectric.co.jp/corporate/randd/list/info_tel/a15/index.html. 2021 年 12 月 23 日閲覧.

[8] IBM Research Blog. "ibm releases fully homomorphic encryption toolkit for macos and ios; linux and android coming soon". June 4, 2020.

[9] 三原健太郎. 準同型暗号の近年の研究はどこに向かっているのか（格子暗号理論、実装研究紹介編）. https://www.eaglys.co.jp/developers-blog/trends/mihara_03/, August 30, 2021. 2021 年 11 月 28 日閲覧.

[10] Rathee M. Chandran N. Gupta D. Rastogi A. Sharma R. Kumar, N. Cryptflow: Secure tensorflow inference. In *2020 IEEE Symposium on Security and Privacy(SP)*, pp. 336–353. IEEE, 2020.

[11] Acompany. 入門　準同型暗号を用いた秘密計算を分かりやすく解説. https://acompany.tech/blog/secure-computing-he/, Feb 10, 2021. 2021 年 12 月 24 日閲覧.

[12] 佐藤宏樹, 山名早人. 完全準同型暗号における bootstrap problem 及び relinearize problem の厳密解法の高速化. 第 11 回データ工学と情報マネジメントに関するフォーラム (DEIM2019), 2019.

[13] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of computing*, pp. 169–178, 2009.

[14] Nakasato S. Arita, S. Fully homomorphic encryption for point numbers. In *Information Security and Cryptology - 12th International Conference, Inscrypt 2016, Beijing, China, November 4-6, 2016, Revised Selected Papers*, pp. 253–270. Springer Berlin Heidelberg, 2016.

[15] Kim A. Kim M. Song Y.S. Cheon, J.H. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, pp. 409–437. Springer Berlin Heidelberg, 2017.

[16] Park M. Kim J. Kim T. Min C. Kim, S. Evalround algorithm in CKKS bootstrapping. In *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part II*, pp. 161–187. Springer Berlin Heidelberg, 2022.

[17] Sahai A. Waters B. Gentry, C. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pp. 75–92. Springer Berlin Heidelberg, 2013.

[18] 佐野秀樹, 蒲原千尋. 教員ストレスに影響する要因の検討: 学校教員の労働環. 東京学芸大学紀要. 総合教育科学系 64(1), pp. 189–193, 2010.

[19] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, 42(1)*, pp. 230–265, 1937.

[20] Deutsch.D.E. Quantum thory, the church-turing principle and the universal quantum computer. *proceedingd of the Royal Socity of London A: Mathematical and Physical Sciences, 400(1818)*, pp. 97–117, 1985.

[21] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM, 56(6)*, pp. 1–40, 2009.

[22] 林卓也. 準同型暗号を用いた秘密計算とその応用. システム/制御/情報 63(2), pp. 64–70, 2019.

[23] Adleman L. Dertouzos M.L. Rivest, R.L. On data banks and privacy homomorphisms. *Foundations of secure computation, 4(11)*, pp. 169–180, 1978.

[24] 松本茉倫, 小口正人ほか. 完全準同型暗号と共通鍵暗号を組み合わせた IoT デバイスにおけるセンサデータ暗号化の高速化. マルチメディア, 分散協調とモバイルシンポジウム, pp. 712–719, 2020.

[25] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 700–718. Springer Berlin Heidelberg, 2012.