DEIM2020 E1-1(day1 p30)

完全準同型暗号を用いたFP-growthによる頻出パターンマイニングにおける委託タスクに関する一検討

種村真由子† 小口 正人†

† お茶の水女子大学大学院 人間文化創成科学研究科 理学専攻 情報科学コース
〒 112–8610 東京都文京区大塚 2-1-1
E-mail: †{mtanemura,oguchi}@is.ocha.ac.jp

あらまし ビッグデータ等の大規模データの収集・分析による利活用がビジネスなどの分野で進んでいる．大規模データを扱う計算には処理能力の高い計算機システムが必要となるため，クラウドなど外部の計算機に処理を委託するなどの方法が現実的であるが，一方で，個人情報に関わるデータはプライバシ保護のため漏えいへの対策が重要である．本研究では，暗号化した状態で計算が可能な完全準同型暗号（FHE）を使用し，先行研究の完全準同型暗号を用いた頻出パターンマイニングの委託システム（P3CC）のアルゴリズムを FP-growth に変更した．このシステムについて，パラメータの変化による実行時間や通信量を測定し，その傾向を考察する．これらの検討結果も踏まえ，クライアント側の負荷をさらに減らす処理を検討する．

キーワード 完全準同型暗号，頻出パターンマイニング，FP-growth

1 はじめに

近年，ビッグデータの収集・分析により，ビジネスを中心とする多くの分野で利活用が進んでいる．大規模なデータを扱う統計処理を行う際，処理能力の高い計算機システムを用意する事が困難な場合には，クラウド等の外部の計算資源を利用する方法がある．一方，外部委託するデータが個人情報や医療情報等，プライバシに関わる場合は，十分に注意することが重要である．多くの場合で行われているプライバシ保護のためのデータ処理の一例としては，個人を特定するような要素をマスキングした上で計算を行うなどの方法がある．

本研究では，プライバシ保護のため，外部サーバに送信するデータを完全準同型暗号 (Fully Homomorphic Encryption，FHE) で暗号化する．FHE は，暗号文のままで加算と乗算を行うことが可能な公開鍵暗号で，委託先サーバに復号鍵を渡さず，統計処理を行うことが可能になる [1]．サーバ上で復号を行わないため，信用できないサーバ上であってもデータの中身を見ることは容易ではないと考えられる．一方で，その性質上 FHE の暗号文同士の比較演算は困難であり，さらに暗号文のデータ量と計算量が多いという特徴もある．FHE については，2 章で説明する．

データマイニングの一種で，頻出パターンマイニングというものがある．これは購買データなどの大規模なトランザクションデータを分析するために使われるものであり，どのアイテムの組み合わせが高頻度で出現するか，そこにどのような傾向があるかを調べるための手法である．頻出パターンマイニングと代表的な 2 つのアルゴリズムの Apriori と FP-growth について，3 章で記述する．

すでに公表されている FHE と頻出パターンマイニングを組み合わせた例として，Apriori アルゴリズムによるトランザクションデータの頻出パターンマイニングを行った Liu ら (2015) の P3CC(Privacy Preserving Protocolfor Counting Candidates) [2] という手法がある．また，P3CC を中国剰余定理に基づく SV(Smart-Vercauteren) パッキングや暗号文のキャッシュを利用し高速化した例 [3] [4] や，サーバ側の分散処理化 [5] を行った例もある．関連研究については，4 章で詳しく説明する．

本研究では，これらの先行研究において Apriori アルゴリズムで処理している部分を FP-growth に変更した頻出パターンマイニングのシステムの実装を行い，アルゴリズムによる比較や，処理の委託部分の増加について検討を行っている．FP-growth はこれらの課題を踏まえたうえで，クライアント側の負荷を抑え，サーバに処理を委託するプログラムを実装する．現行の実装については，5 章に示す．このプログラムの実行時間や通信量は暗号に設定するパラメータによって変化する，その変化量などを 6 章で説明する実験によって確認し，8 章にて，それを踏まえて負荷のかからない範囲での委託タスクを検討する．

2 完全準同型暗号

2.1 完全準同型暗号の概要

完全準同型暗号は，加法準同型性と乗法準同型性の特徴をあわせ持った，すなわち，暗号化した状態での暗号文同士の加算，乗算が成立する暗号方式である．すなわち，暗号文同士の計算をしたものを復号すると，暗号化される前の平文を直接計算したものと同じ結果が得られる．FHE は公開鍵暗号方式の機能を持つ．これにより，FHE を用いて暗号化されたデータは，委託先サーバに平文データを見られることなく計算処理を委託できるという期待がある．

完全準同型暗号は，概念そのものは Rivest ら (1978) によって提案されている [6]．また，2009 年に Gentry が Lattice ベースの実現手法を提案した [1]．各暗号文には，暗号の解読不可能性を高めるため，ランダムなノイズが付加されている．問題点としては，一般に暗号文と鍵のデータサイズが大きいことにより処理の計算量が膨大になることと，ノイズの値が暗号文同士の計算を行うたびに増加し，閾値を超えると復号が不可能となること，比較演算が困難であることが挙げられる．ノイズの値は，特に乗算を行った際に大きく増加する．ノイズが増加した際は bootstrapping という処理を行うことで，暗号文のノイズを初期値に近い量に減少させ，暗号化した状態での計算が理論上何度でも可能となるが．しかし，実際にはこの処理も計算量が大きいという問題がある．

2.2 Leveled FHE

bootstrapping を使用しない完全準同型暗号の実装として，Brakerski らによって提唱された Leveled FHE がある [7]．Leveled FHE は，決まった深さの論理回路の結果を評価することができる，完全準同型暗号の実装の一つである．
事前に与えたレベルに対して，計算の論理回路の深さが小さければ，bootstrapping を行う必要がないため，あらかじめ計算回数が把握できる場合に有効である．本研究で使用するプログラムでは，この Leveled FHE を使用している．

3 頻出パターンマイニング

頻出パターンマイニングは，データマイニングの一種であり，大量のデータの中から，相関ルールを抽出することを目的とした手法である．例として，トランザクションデータを対象とした．バスケットデータ分析がある．
本研究で扱う頻出パターンマイニングでは，指定したアイテムのうち，各トランザクションがどのアイテムを含んでいるかを，バイナリ行列で表したデータを対象に行う．頻出であることの判定は，各パターンのサポート値があらかじめ指定したミニマムサポート値以上であるかを比較することにより行う．サポート値は，全体のトランザクション数に対する，あるアイテムセットが含まれるトランザクション数の割合で表される．代表的なアルゴリズムとして，先行研究で使用されている Apriori と，本研究で使用している FP-growth がある．各アルゴリズムについての概要を以下に述べる．

3.1 Apriori

Apriori は，アイテム長 1 のアイテムセットから順にサポート値をミニマムサポート値と比較し，頻出アイテムセットを列挙していく，幅優先探索型のアルゴリズムである [8]．4 章で述べる先行研究で使用されている．アイテムの種類が少ない場合でも組み合わせの総数は膨大になり得るため，あるアイテム長 n のアイテムセットのサポート値が事前に設定した最小サポート値未満の場合は，そのアイテムセットを含むアイテム長 n+1 のパターンも頻出でないと判断し，その後の探索を行わないようにするといった枝刈りを行うことにより計算量を削減している．実装は比較的容易である．

3.2 FP-growth

FP-growth は，Apriori に対して，FP-growth は，まず頻出パターンを全て含む FP-tree という木構造データを作成し，それを走査することにより，頻出アイテムセットを求める深さ優先探索型のアルゴリズムである [9]．データベースを走査し，トランザクションデータを FP-tree という prefix tree の木構造に格納，その部分木を取り出しながら再帰的に走査することで結果を求める．
頻出アイテムセットの探索に木構造のデータを用いる，頻出アイテムセットの候補を列挙しないという点で，Apriori と大きく異なる．また，データの特性にも依存するが，頻出アイテムの列挙がボトルネックになる Apriori と比較して，探索を効率化できるという期待がある．アルゴリズムの実装は Apriori よりも複雑である．効率化が期待できる点がある一方で，FP-tree の構築と走査には多数かつ頻繁な比較演算が必要であるため，FHE を使用して行うことのできる処理が制限される．

4 先 行 研 究

FHE を使用し秘匿データマイニングを行った先行研究について，概要を紹介する．なお，本項で挙げる先行研究で採用されている頻出パターンマイニングのアルゴリズムは，すべて Apriori である．

4.1 P3CC

P3CC は，Liu ら（2015）が提案した，完全準同型暗号を用いた安全な頻出パターンマイニング委託システムである [2]．完全準同型暗号の暗号文同士は比較演算が困難であるため，比較演算が必要な部分に関してはクライアントにデータを返送し処理を行う．また，暗号文のデータサイズ削減のため，アイテム数，トランザクション数は暗号化されず，各トランザクションに含まれるアイテムを示すバイナリ行列にのみ暗号化を適用している．さらに，クライアント側でダミーデータを加えることで，委託先サーバからの平文データの推測を防いでいる．

4.2 P3CC の暗号文パッキングと暗号文キャッシングによる高速化

高橋ら（2016）は，P3CC を SV パッキングを用いて高速化する手法を提案した．複数の整数をベクトルとして一括に暗号化できるパッキングという方式を用いて，暗号文の個数，暗号文同士の乗算を削減を行った．その結果，パッキングを用いない場合と比較して 10 倍以上の高速化を実現した．この手法は，Apriori に限らず，秘匿検索や他のデータマイニングアルゴリズムにも応用することができるとしている [3].
今林ら（2017）は，完全準同型暗号による頻出パターンマイニングの時間・空間計算量を削減する，暗号文パッキングの適用手法と暗号文キャッシング手法を提案した．提案手法が P3CC による Apriori の実行時間とメモリ使用量を大きく削減することができることを示し，データセットが大きい場合や，ダミー

セットを加えた場合により効果が大きかったとしている．特にトランザクション数 10,000 のとき，P3CC と比較して，430 倍の高速化と 94.7%のメモリ使用量削減を実現している [4].

4.3 P3CC の分散環境への実装とデータベース更新時の処理の高速化

山本ら（2018）は，データベース更新時の Apriori アルゴリズムの高速化を行う FUP（Fast UPdate）アルゴリズムを用いた秘匿データマイニングシステムの実装を行った．また，マスタ・ワーカ型分散処理を適用し，システムの高速化を行った．分散処理方法には，アイテムセットごとの分割を適用した．その結果，データベース更新時において FUP アルゴリズムを導入した際の再計算の計算時間は，Apriori アルゴリズムによる再計算と比較して約 3〜4 倍の短縮が可能になった．また分散処理化によって，マスタ側の計算時間が分散台数に応じて減少している [5].

5 システム概要

クライアント・サーバ型のデータの委託処理システムを実装した．クライアント側のマシンには秘密鍵，公開鍵の両方があり，サーバ側には公開鍵のみが置かれている．サーバ側ではデータを復号する必要のない処理を行う．

5.1 処理の流れ

FHE はその性質上，可能な計算の種類と回数に制限があることから，本システムでは頻出パターンマイニングのサーバへの委託は部分的に行っている．また，比較演算を必要とする処理はクライアントに戻して行う必要がある．現行のシステムでの手順は以下の通りである．

（1） サーバの立ち上げと接続準備
サーバ側で通信の受付を行い，クライアントとの接続を確立する．

（2） クライアントでのデータの送信準備
クライアント側で保持しているトランザクションデータ（各要素が 0 または 1 のバイナリ行列）を FHE で暗号化する．暗号化したデータを，アイテム ID とミニマムサポート値とあわせてサーバに送信する．

（3） サーバでの委託処理
クライアントから暗号化されたデータを受信する．各アイテムの出現頻度を計算することにより，各アイテムのサポート値を計算する．その後，クライアントに結果を返送する．

（4） クライアントでの FP-tree 構築
サーバから受信したファイルの復号を行う．アイテムごとのサポート値とミニマムサポート値を比較し，サポート値の条件を満たしていたアイテムを抽出し，FP-tree の構築を行う．

（5） クライアント側での FP-tree 走査
構築した FP-tree の走査を行い，結果を出力する．

5.2 実　　装

C++により実装．サーバ側はマスター／ワーカ型の分散・並列処理を行うことができる．FHE を扱うためのライブラリは HElib [10](2016/8 時点の実装)，分散・並列処理のライブラリには Open MPI [11] を使用した．

6 実　　験

6.1 実 験 概 要

FP-growth 同一ネットワーク内の 2 台の計算機を用いて，各計算機上でクライアントプログラムとサーバプログラムを動作させた．本研究のプログラムは，2 章内で示した，暗号文の持つレベルというパラメータにより実行時間が大きく左右されることが分かっている．今後実装を改善するにあたって，FHE による処理が増加した場合，暗号文に事前に設定するレベルの値も増加させる必要がある．加算の処理が増える場合は大きく必要レベルは増加しないが，乗算を行う場合は特に必要なレベルが増加する．そこで，現行のプログラムを用いて，レベルによる実行時間，ノイズ増加の傾向を把握ことを目的として，実験を行う．

6.2 実 験 環 境

実験で用いた計算機の性能を表 1 に示す．

表 1　実験で用いた計算機の性能

OS	CentOS 6.9
CPU	Intel ® Xeon ® プロセッサ E5-2643 v3 3.6GHz 6 コア　12 スレッド
メモリ	512GB

クライアント，サーバとして表に示す同型のマシンを 1 台ずつ使用した．本プログラムはサーバ側がマスタ／ワーカ型の分散処理が可能な実装であるが，本実験では分散処理は行わない．クライアント，サーバとして利用するマシンは同一ネットワーク内に存在する．使用した入力データは，IBM Quest Synthetic DataGenerator で生成した人工データである．アイテム数は 30，トランザクション数は 9900 に設定して生成を行う．使用した HElib のバージョンは，2016 年 8 月 12 日にコミットされたものである．

6.3 実 験 方 法

5 章で示したプログラムにおいて，暗号文のレベルを 3 から 17 まで変化させ，その実行時間，暗号文のノイズ量，通信データサイズを測定する．レベル 3 から開始する理由は，このプログラムを実行するために必要最小限のレベルが 3 であるためである．（レベルを 2 以下に設定すると，暗号文の復号ができないことがわかっている．）実行時間の測定には，サーバ側のプログラムが立ち上がった後からクライアントの接続待ちをしている時間は含まない．また，各レベルについて 5 回測定し，その平均をとったものを結果として示す．全ての測定で，ミニマムサポート値は 0.05 に指定している．ノイズの値は，サーバで

の計算を終え，暗号文のパッキングを行った後のものを測定している．

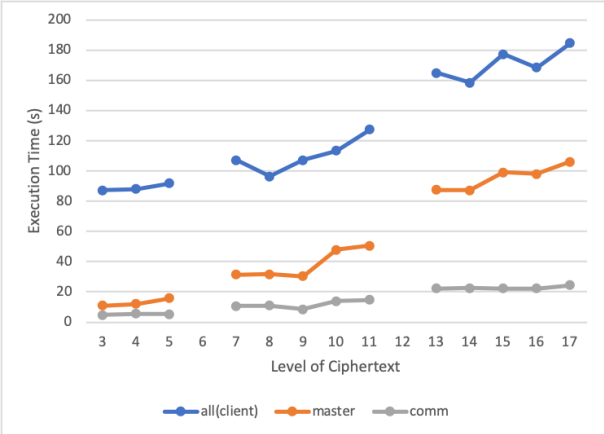

図 1　プログラムの実行時間とサーバ側での実行時間

6.4　実 験 結 果

実行時間の計測結果を以下の図 1 に示す．all が全体の実行時間，master がサーバにおける実行時間，comm が通信にかかった時間である．master と comm は all に含まれる．レベルの変化に伴い，暗号文で計算を行うサーバ側での処理時間が増加しており，また，全体の実行時間に対する割合はほぼ単調増加していることがわかる．レベル 3 では全体の実行時間の 12.5%ほどだが，レベル 17 では 57.5%に及ぶ．また，例外としてレベルの値を 6，12 に設定した際には暗号文が復号不可となり実行時間のデータの取得ができなかった．該当箇所はグラフ中で空欄としてある．

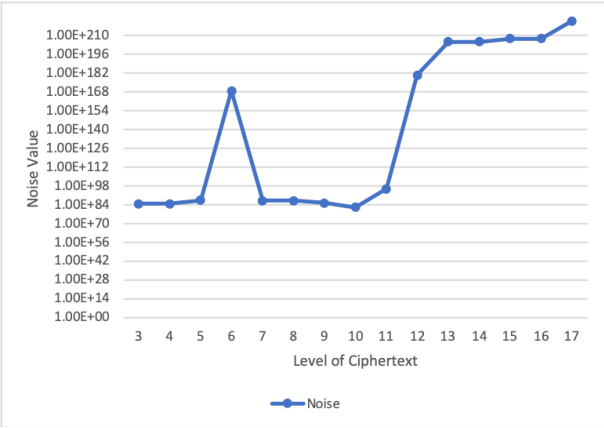

図 2　プログラム実行時の暗号文レベルに伴うノイズの変化

レベルの値を変化させた際のノイズの値は図 2 に示す通りである．グラフの縦軸は対数グラフとなっている．実行時間の測定が成功しなかったレベル 6 と 12 の周辺の値について注目すると，レベル 6 から 7 に増加する際にノイズの桁が 10^{81} 増加し，レベル 11 から 12 にかけてはノイズの桁数が 10^{85} 増加している．レベル 12 を超えた後は，レベル 6 前後でノイズの値が上下している様子と異なり，レベル 13 以降もノイズの桁数は減少しない．

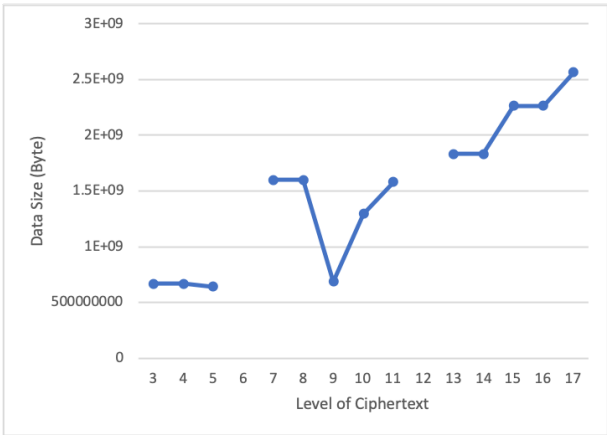

図 3　プログラム実行時の暗号文レベルに伴う通信データ量の変化

図 3 には，レベルによる通信データ量の変化を示す．図 1 に示したものと同じく，レベル 6 とレベル 12 では実行が完了しなかったため，該当箇所は空欄とした．全体の大きな傾向としては，レベルの増加に伴って，通信データ量も増加している．しかし，レベル 7 のノイズの値がレベル 5 と比較して大きく増加していることと，レベル 9 のノイズの値が大きく減少している点が特徴的である．

7　考　　察

実行時間を測定できなかった特定のレベルについて考える．図 1 のグラフと図 2 のグラフを合わせて考えると，復号不可となったレベル 6 と 12 でノイズが急激に増加していることから，この際に復号可能なノイズの値を大きく超え，暗号文が復号できなかったものと見られる．この特異なレベルにおける急激なノイズの値の増加が本プログラムと FHE ライブラリを組み合わせた際に起きる特有の事象か，FHE ライブラリそのものの暗号化処理が原因であるかについては今後さらに調査していく必要がある．今後，プログラムの実装を変更するにあたり，実行時間と通信データ量の負荷を抑えるためには，レベルを低く抑えることのできる手法を選択するのが適している．また，ノイズの値の予測がつかない場合には，現在使用している Leveled FHE ではなく，必要に応じてノイズ削減を行うことができる Bootstrap を用いる方法が適している可能性がある．

8　新規手法の検討

現行システムの改善のため，クライアントにサポート値の計算結果を返送する前に，サーバにおいて各アイテムのサポート値とミニマムサポート値の差をとるという処理を新しく追加することを検討する．5.1 章で示した通り，現在の実装では，クライアント側のマシンで FP-growth の頻出アイテムのサポート値計算の処理を行っている．したがって，サーバ側に委託する処理が増加することになる．
検討する処理全体の流れは以下のようになる．

（1）　サーバの立ち上げと接続準備

サーバ側で通信の受付を行い，クライアントとの接続を確立する．

（2） クライアントでのデータの送信準備

クライアント側で保持しているトランザクションデータ（各要素が 0 または 1 のバイナリ行列）を FHE で暗号化する．暗号化したデータを，アイテム ID とミニマムサポート値とあわせてサーバに送信する．

（3） サーバでの委託処理

クライアントから暗号化されたデータを受信する．各アイテムの出現頻度を計算することにより，各アイテムのサポート値を計算する．ここで，同時に各アイテムのサポート値とミニマムサポート値との差をとった新たな配列 Sup-Diff を作成する．クライアントに結果を返送する．

（4） クライアントでの FP-tree 構築

サーバから受信したファイルの復号を行う．Sup-Diff の各値について，0 以上か 0 未満かの判断を行い，0 以上であればミニマムサポート値以上のサポート値を持つと判断する．サポート値が基準以上のアイテムを抽出し，FP-tree の構築を行う．

（5） クライアント側での FP-tree 走査

構築した FP-tree の走査を行い，結果を出力する．

変更前のプログラムと比較して，サーバ側で暗号文に対して行う処理が増えるが，サーバ側で追加される演算は加減算のみであるため，ノイズの増え方は少なく，暗号文の必要レベルは大きく増加しない見込みである．これにより，実行時間を抑えつつ，新たに処理を委託することができると考えられる．一方で，結果を別に保存することで，新たに扱う暗号文が増えることから，通信データ量は増加することが予想される．

9 まとめと今後の課題

完全準同型暗号を用いた FP-growth において，現行のプログラムで暗号文に設定するレベルに対する実行時間，ノイズ，通信データ量の変化を測定した．全体の傾向としてはレベルが大きくなるほど実行時間，ノイズ，通信データ量の値共には増加するが，特定のレベルを設定した際に特異な挙動が現れた．この事象について，今後も原因を調査していく．また，今回得た結果を参考に，暗号文のレベルを抑えつつ，サーバへの委託処理を増加させる方法を検討した．新しく検討した実装によるシステムを使用し，実行時間，使用リソースの測定も行っていきたいと考えている．

10 謝　　辞

本研究は一部，JST CREST JPMJCR1503 の支援を受けたものです．

文　　献

[1] C. Gentry, ”A Fully Homomorphic Encryption Scheme, Doctoral dissertation,” 2009.

[2] J. Liu, J. Li, S. Xu, and B. CM Fung, ”Secure outsourced frequent pattern mining by fully homomorphic encryption”, In International Conference on Big Data Analytics and Knowledge Discovery, pp. 70 - 81. Springer, 2015.

[3] 高橋卓巳, 石巻優, 山名早人, ”SV パッキングによる完全準同型暗号を用いた安全な委託 Apriori 高速化,” DEIM Forum 2016 F8-6, 2016.

[4] 今林広樹, 石巻優, 馬屋原昂, 佐藤宏樹, 山名早人, ”完全準同型暗号による安全頻出パターンマイニング計算量効率化,” 情報処理学会論文誌データベース (TOD), Vol. 10, No. 1, 2017.

[5] 山本百合, 小口正人, ”完全準同型暗号を用いた秘匿データマイニング計算のデータベース更新時の分散処理による高速化,” DICOMO2018, 2018.

[6] R. L. Rivest et al.,“On data banks and privacy homomorphisms,”Foundations of secure computation, vol. 4, no. 11, 1978, pp. 169 - 180.

[7] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan: (Leveled) Fully Homomorphic Encryption without Bootstrapping. ACM Trans. Comput. Theory 6, 3, Article 13 (July 2014), 36 pages, 2014

[8] R. Agrawal, T. Imieli ´ nski, and A. Swami, “Mining association rulesbetween sets of items in large databases,” in Acm sigmod record,vol. 22, no. 2. ACM, 1993, pp. 207 - 216.

[9] J. Han, J. Pei, and Y. Yin, “Mining Frequent Patterns WithoutCandidate Generation,” in Proceedings of the 2000 ACM SIGMODInternational Conference on Management of Data, ser. SIGMOD ’00.New York, NY, USA: ACM, 2000, pp. 1 - 12. [Online]. Available:http://doi.acm.org/10.1145/342009.335372

[10] HElib. https://github.com/homenc/HElib (visited on 18/09/2019).

[11] Open MPI. https://www.open-mpi.org/ (visited on 18/09/2019).