完全準同型暗号を用いた委託頻出パターンマイニングの FP-growth の発展的アルゴリズムによる性能改善に関する検討

種村真由子† 小口 正人†

† お茶の水女子大学 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 近年，ビッグデータの利活用が多くの分野で行われている．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．しかし，外部委託するデータがプライバシに関わる場合や機密情報の場合は，特に管理に注意する必要がある．本研究では，完全準同型暗号 (Fully Homomorphic Encryption，FHE) で暗号化したデータを使用し，これを用いて，頻出パターンマイニングを行うクライアント・サーバシステムを作成する．また，頻出パターンマイニングのアルゴリズムとして，FP-growth を使用した場合のパラメータの検討と，FP-growth の派生アルゴリズムである FP-growth* を使用した実装の提案を行う．

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・分析などが，ビジネスを中心とする多くの分野で進んでいる．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が，外部委託するデータが個人情報等，プライバシに関わる場合は，特に管理に注意する必要がある．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，暗号文同士の加算と乗算が成立する公開鍵暗号で，委託先サーバに復号鍵を渡さず，すなわちデータの内容を見せることなく，統計処理を行うことが可能となる．一方で FHE の性質上, 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．本研究では，これを用いて，頻出パターンマイニングを行うシステムを作成している．頻出パターンマイニングはデータマイニングの一種で，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 については，3 章で記述する．FHE と頻出パターンマイニングを組み合わせた関連研究として，Apriori を使用した Liu ら (2015) の P3CC(Privacy Preserving Protocol for Counting Candidates) という手法がある．また，P3CC を 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や，サーバ側の分散処理化を行った例もある．先述の通り，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っている．しかし，以前実装したプログラムでは，クライアントがサーバに容易に委託可能なタスクが少ないことにより，クライアントで多くの処理を行っている．クライアント側の処理の負荷を抑えるためさらにサーバに処理を委託する方法を検討する．

2 完全準同型暗号

2.1 FHE の概要

完全準同型暗号 (FHE) は，加法準同型性と乗法準同型性の特徴をあわせ持った公開鍵暗号方式である．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同士を直接計算したものと同じ結果が得られる．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 [1]．また，2009 年に Gentry が Lattice ベースの実現手法を提案した [2]．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，実際にはこの処理も計算量が非常に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[3]．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．Leveled FHEは，事前に設定し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．暗号文同士の計算によるノイズを減らすために，暗号文のbootstrappingより処理が軽量なmodulus switchingを使用しており，あらかじめ暗号文同士の計算回数が把握できる場合に有効である．

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．

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，頻出パターンを抽出する処理を行う．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最小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の3.1章，3.2章に示す．

3.1 Apriori

Aprioriは，アイテム長1，すなわちアイテムを1つだけ含むパターンから順にサポート値を最小サポート値と比較し，頻出と判断されたパターンを列挙していくアルゴリズムである[4]．実装は比較的容易である．アイテムの種類が少ない場合でも，アイテム同士の組み合わせの種類の総数は膨大になり得るという問題がある．したがって，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う枝刈りを行い，計算量を削減している．この手法では，アイテム長が長いパターンが頻出であるとき，アイテム長の分だけ処理が長引くという特徴がある．4章で述べる先行研究で使用されている．

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アルゴリズムである[5]．まず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の木構造に格納する．その部分木を取り出しながら部分木の枝分かれがなくなるまで再帰的に走査することで，頻出パターンを求める．データベースの走査回数が最低2回で済むという特徴がある．Aprioriと大きく異なる点は，探索に木構造データを用いる，頻出パターンの候補を列挙しないという部分である．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．木の規模が大きくなるほど計算量が大きくなり，特にトランザクションデータにおいては，トランザクション数よりもアイテム数の増加が木の規模の拡大に強く影響する．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[6]．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[7]．

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 [?, 8]

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 [9]．

5 システム

5.1 システム概要

本研究で使用するシステムは，すべてクライアント・サーバ型の委託処理を行うものである．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を暗号化されたデータとともに共有する．サーバ側ではデータを復号する必要のない処理のみを行うこととする．図 1 に概要を示す．

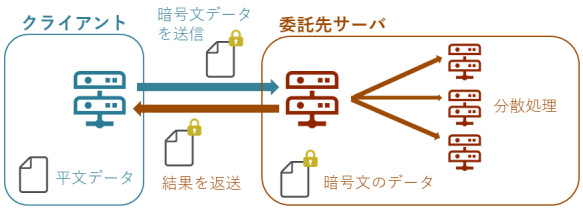

図 1　提案システム概要

5.2 処理の流れ

Leveled FHE はその性質上，可能な計算の種類に制限があることから，頻出パターンマイニングの処理全てをサーバ上で行うことはできない．とくに，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．したがって，本システムでは頻出パターンマイニング処理のサーバへの委託は，部分的に行っている．具体的には，FP-tree を構築する手前の，データ中の各アイテムの頻出・非頻出を調べる段階の処理までをサーバで行う．これは，FP-tree を構築する際に比較演算を多用するため，サーバ上で FP-tree をそのままの形で構築するのが困難なためである．

処理手順は以下の通りである．

プログラムの手順

（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） トランザクションデータ送信前準備
公開鍵と必要なパラメータ等を事前に送信する．

（3） クライアントでのトランザクションデータ送信
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）と，最小サポート値に対応するアイテムの出現回数の閾値を FHE で暗号化し，サーバに送信する．続いてアイテム ID のリスト（暗号化されていないデータ）をサーバに送信する．

（4） 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．各アイテムの出現回数を累計する．さらに，別のデータとして各アイテムの出現回数と閾値の差を取る．その後，クライアントに 2 つの処理の結果を返送する．

（5） 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．出現回数と閾値の差のデータが 0 以上であるアイテムを頻出と判断する．頻出であったアイテムを取り出し，FP-tree の構築を行う．FP-tree の構築には出現回数のデータを用いる

（6） クライアント側での FP-tree 走査
構築した FP-tree の走査を行う．（走査のため再帰的に FP-tree を構築する処理を含む）

本プログラムでは，下線で示した部分の処理をクライアント，サーバで追加している．これにより，クライアント側で行っていたサポート値とサポート値の閾値を比較するために差分を取る処理をサーバに担わせている．

5.3 実装環境

使用し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

また，本プログラムでは，オープンソースの完全準同型暗号ライブラリである HElib を使用している [10]．

5.4 セキュリティパラメータの検討

一般に，暗号文にはその強さを示すセキュリティレベル (bit) がある．これは暗号文の鍵長を示し，値が大きいほど解読されにくく強いといわれている．完全準同型暗号にもセキュリティレベルが存在する．HElib においてこれは複数のパラメータに

よって定まっている．セキュリティパラメータ k（セキュリティレベルとは別物である）や暗号文のレベルによってセキュリティレベルが変化する．k 以外ののパラメータについては以下の表 2，表 3 の通りである．本実験の入力データは人工的に作成した．データ生成は，IBM Quest Synthetic Data Generator を用いて行った．

表 2　入力データのパラメータ

平均トランザクション長	5
最大パターンの大きさ	5
アイテムの種類	30
トランザクション数	9900

表 3　プログラムにおいて設定したパラメータ

暗号文のレベル	3
最小サポート値	0.1(10%)

以下の表 4 は本章で示したプログラムについて，セキュリティパラメータ k を変化させた際のセキュリティレベルと実行時間を測定したものである．k は 100 から 350 まで，50 おきに設定した．セキュリティパラメータの測定は，HElib に実装されている関数を用いた．

表 4　セキュリティパラメータ k と暗号文のセキュリティレベル (bit)

セキュリティパラメータ k	100	150	200
セキュリティレベル (bit)	61.41	85.1639	110.359

セキュリティパラメータ k	250	300	350
セキュリティレベル (bit)	198.253	198.253	198.253

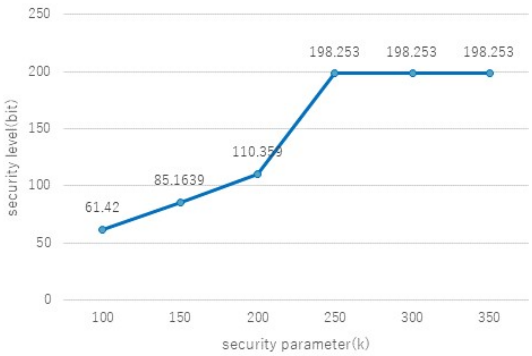

図 2　セキュリティパラメータと暗号文のセキュリティレベル（グラフ）

k を 100 から増加させるごとにセキュリティレベルも増加する傾向にあることが分かる．また，現在のパラメータにおいては，少なくとも k=250 から k=350 ではセキュリティレベルが 198.253 のまま変化しなかった．参考として，Homomorphic Encryption Standardization[12] で例として示されているセキュリティレベルは 128，192，256 である．暗号文のレベルを増加させるとセキュリティレベルは減少するため，198.253 以上に設定する場合はさらにパラメータを調整する必要があると考えられる．

6　FP-growth*を使用した実装の提案

6.1　概　　要

前章までで示した FP-growth のシステムについて，FP-tree を走査する過程において特に多くの計算コストがかかっていることが分かっている．これは条件付き FP-tree を再帰的に生成することの負荷が大きいためであると考えられる．また，入力データサイズのアイテムやトランザクションが大きくなるほど FP-tree の規模も大きくなるが，特にアイテムが増えた際には FP-tree の規模への影響が大きい．そこで，FP-growth を元にしたアルゴリズムである，FP-growth*を取り入れたシステムを提案する．

6.2　FP-growth*

FPgrowth*は，FP-growth において計算コストの高い，FP-tree の走査の過程を効率化するために，Array という新しい構造を組み込んだアルゴリズムである [11]．このアルゴリズム内で出てくる Array というデータ構造は，一般的なプログラムにおけるデータ構造の Array とは異なるものである．各 FP-tree 対応して Array は一つ作成される．アイテムで条件付けされた FP-tree を新たに構築する際の処理を効率化する．処理の流れは以下の通りである．

（1）　条件付けをするアイテムの列と交差する行のアイテムを取り出す
（2）　取り出したアイテムを頻度順に整列
（3）　Header Table（各 FP-tree に紐づいている，各アイテムごとに tree 内のノードをリンクとしてたどれるようにした Table）から条件付けするアイテムをノードからたどる
（4）　整列したアイテムを元に行と列を設定し，新しい Array を作成する

この手法は，疎なデータに対して効果があるといわれている．その反面，密なデータに対しては逆に計算コストがかかる場合がある．

6.3　実装の検討

現在実装されている FP-tree に，Array のデータ構造を紐づけることを検討している．処理の全体の流れは，5.2 節で示したものから変更しない．FP-growth の tree を操作する処理において，再帰的に FP-tree を作成する過程が変更される．

7 まとめ・今後の課題

頻出パターンマイニングのプログラムのさらなる実装の改善を行い，セキュリティパラメータについての測定を行った．また，このシステムにおいて処理の負荷が大きい FP-tree の再帰的な走査部分において，FP-growth*を新たに組み込んだシステムの提案を行った．今後の課題として，適切なパラメータの設定，入力データを変化させた測定を行うと同時に，FP-growth*を使用したプログラムの実装を進めていく．

文　献

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

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

[3] 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

[4] R. Agrawal, T. Imieli nski, and A. Swami:“Mining association rules between sets of items in large databases,”in Acm sigmod record, vol. 22, no. 2. ACM, 1993, pp. 207–216.

[5] J. Han, J. Pei, and Y. Yin:“Mining Frequent Patterns Without Candidate Generation,”in Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD ’00. New York, NY, USA: ACM, 2000, pp. 1–12. [Online]. `http://doi.acm.org/10.1145/342009.335372` accessed on 2020-05-15

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Shai Halevi and Victor Shoup: HElib. `https://github.com/homenc/HElib` accessed on 2020-05-15

[11] GRAHNE, Gösta; ZHU, Jianfei.: Efficiently using prefix-trees in mining frequent itemsets. In: FIMI. 2003. p. 65.

[12] Martin Albrecht and Melissa Chase and Hao Chen and Jintai Ding and Shafi Goldwasser and Sergey Gorbunov and Shai Halevi and Jeffrey Hoffstein and Kim Laine and Kristin Lauter and Satya Lokam and Daniele Micciancio and Dustin Moody and Travis Morrison and Amit Sahai and Vinod Vaikuntanathan: Homomorphic Encryption Security Standard, HomomorphicEncryption.org, 2018.