分散環境におけるFP-growthによる相関関係抽出の完全準同型暗号を用いたデータベース秘匿化の検討

種村真由子† 小口 正人†

† お茶の水女子大学理学部情報科学科 〒112–8610 東京都文京区大塚 2-1-1
E-mail: †{g1520528,oguchi}@is.ocha.ac.jp

あらまし 大規模なデータを収集し活用することにより，IoT や医療など，様々な技術の進展が期待される．一方で，機密データをクラウド等の外部に委託して処理する場合には，セキュリティ管理が重要である．暗号文同士の加算，乗算が成立する完全準同型暗号（FHE）を用いることで，委託先サーバに平文データを送信することなく処理を行うことが可能である．FHE を使用してデータの処理を行った例として，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 らの P3CC があげられる．本研究では，同様に FHE を使用し，Apriori と同様の結果が得られる別のアルゴリズムである FP-growth を用い，頻出パターンマイニングの実装を行う．

キーワード 分散処理，FP-growth，完全準同型暗号

1 はじめに

大規模なデータを収集し分析することによるビジネスなどの分野で利活用が進んでいる．大きなデータを扱う計算を行うには，処理能力の高い計算機システムが必要となるが，自社でシステムを用意する事が困難な場合には，クラウド等外部サービスに委託することが現実的である．しかし，特に個人情報や医療データ等の機密データ処理の委託に関しては，送信する情報の漏えい対策が重要となる．外部委託して処理する場合において，完全準同型暗号（FHE）を用いることで，委託先サーバに平文データを渡さずに処理を行うことが可能になる．したがって，信頼できるサーバに対してでなくとも委託処理を行えるという期待がある．FHE の応用例として，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 らの P3CC や，その高速化，分散処理化を行った研究などがある． P3CC とその高速化についての先行研究は，4 章で説明する．

本研究では，暗号文同士の加算と乗算が成立する暗号でる FHE を使用し，同様にパターンマイニングを行う他アルゴリズムの FP-growth による頻出パターンマイニングの実装に取り組む．

2 完全準同型暗号

完全準同型暗号は，公開鍵暗号方式の機能を持ち，暗号文同士の加算，乗算が成立する暗号である．2009 年に Gentry が実現手法を提案した [1]．各暗号文は，暗号の解読不可能性を高めるため，ノイズパラメータという値が付加されている．この値は，暗号文同士で計算を行うたびに増加し，特に乗算を行う際に大きく増加する．ノイズパラメータの値は，閾値に対して十分に小さい必要があり，閾値を超えると復号が不可能となる．加算や乗算を行う前にブートストラップというノイズを減らす処理を行うことで，計算後のノイズを十分に小さくすることができる．復号鍵を使用する事なく，暗号化されたデータのままで送信先サーバが計算を行えるという特長により，活用が期待されている．しかし，一般に処理の計算量が大きいことから，現在実用化に向けての研究が盛んに行われている状況である．

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，大量のトランザクションデータの中から，頻出であるアイテムの組み合わせや条件を取り出す手法である．応用例として，購買データから頻繁に購入される商品の組み合わせを抽出するマーケットバスケット分析が挙げられる．以下に頻出パターンマイニングのアルゴリズムを 2 つ挙げる．

3.1 Apriori

Apriori は，アイテム長 1 のパターンから順に頻出アイテムの組を列挙していく，幅優先探索型のアルゴリズムである．後述する先行研究で使用されている．アイテムセットの出現頻度から計算するサポート値という値を用いて，一定以上のサポート値を持つパターンのみを抽出する．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．あるアイテム長 n のパターンのサポート値が，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．アイテムの種類が少なくてもアイテムの組み合わせの種類は膨大になり得るため，このような枝刈りを行うことにより，計算量を削減している．

3.2 FP-growth

FP-growth は，本研究で用いる頻出パターンマイニングのアルゴリズムである．先述の Apriori に対して，深さ優先探索のような形で頻出パターンの抽出を行う．トランザクションデー

タを FP-tree というデータ構造に格納し，それを走査することで頻出パターンを求める．また，Apriori と異なり頻出アイテムセットの候補を列挙しないという特徴がある．データの規模や特徴にも依存するが，頻出アイテムの列挙がボトルネックになる Apriori と，それに類似するアルゴリズムと比較して，探索を効率化できるという期待がある．

FP-tree は，以下のように定義されている．[2].

（1） 「null」とラベリングされたルートノード，ルートの子として item prefix subtree（アイテムの接頭辞部分木）の組と，frequent-item header table を持つ．

（2） 各 item prefix subtree は item-name，count，node-link を保持する．count は，そのノードに至る部分パスによって表されるトランザクション数を示す．node-link は FP-tree 内に存在する同名アイテムをもつ次のノードへのリンクであり，存在しない場合は null となる．

（3） frequent-item header table の各エントリは item-name と head of node-link を保持する．head of node-link は，前項（2）で述べた node-link の先頭のノードを示すものである．

図 1 に，FP-tree の簡易な例を示す．ただし，図中の長方形は各ノードを示し，実線はノード間の接続，点線矢印は node-link の示す先のノードを指しているものとする．各ノードに書かれている A, B, C は item-name の例，数字は count である．

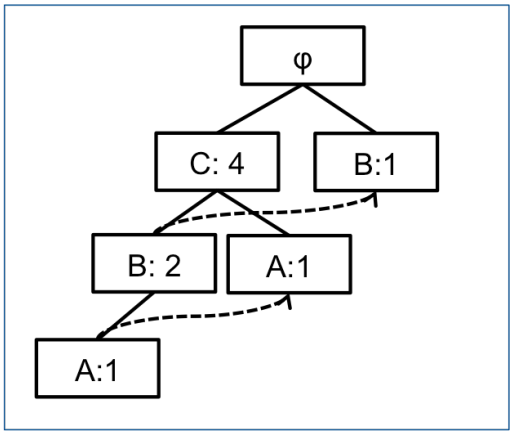

図 1　FP-tree の簡易な例

FP-tree を構築するには，トランザクションデータと，最小サポート値が入力として必要である．

構築手順の概要を以下に示す．

（1） データベースを 1 回走査し，頻出アイテムのセットと各アイテムのサポート値を計算する．アイテムのサポート値が最小サポート以上の場合に頻出であるとする．

（2） 頻出アイテムのセットをサポート値の降順でソートする．

（3） FP-tree のルートを null とする．

（4） 各トランザクション内から頻出アイテムを取り出し，それらを（2）の順に従ってソートする．

（5） ソート済みのトランザクションを作成済みの tree に挿入する．トランザクションの最初の要素と，tree のルートの子 N のアイテム名が一致した場合，N の count を 1 増やす．そうでない場合，新しい子ノードを作成し，count を 1 とする．この処理を，トランザクションの要素に繰り返し適用する．

図 1 に示した FP-tree は，例えば T_1={C}，T_2={A, C}，T_3={B, C}，T_4={A, B, C}，T_5={B} のようなトランザクションから構築される．ただし，アイテム A, B, C は全て頻出アイテムに含まれるとする．FP-tree を一度作成した後は，パターンの探索を FP-tree によって行うため，データベースを走査する必要がない．したがって，アイテム数に依らず，データベース全体を走査する回数は 2 回となる．

作成した FP-tree を，出現頻度の低いアイテムを含むアイテムの組から順に，木の枝分かれがなくなるまで条件付 FP-tree を構築しながら再帰的に走査（FP-growth）を行う．走査手順の概要は以下である．

（1） 頻出アイテムのリストの中で，一番下（最も非頻出なアイテム）から処理を行う．

（2） 各頻出アイテムに対する conditional pattern base を作成．これは，各頻出アイテムに至るまでのノードとその頻度のリストである．

（3） conditional pattern base から conditional FP-tree を作成．

（4） tree に枝分かれが無い場合

（a） tree のパスに含まれるノードの各組み合わせに対して，条件付けに使用したアイテムを合わせたパターンを作成する．パターンのサポート値はパスに含まれるノードの最小サポート値とする．

（5） tree に枝分かれがある場合

（a） 再帰的にアイテムで条件付けした conditional FP-tree を作成し，走査する．

4　先 行 研 究

FHE を使用し頻出パターンマイニングを行った先行研究の概要を紹介する．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 [3]．完全準同型暗号の暗号文同士は比較演算が困難なため，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，クライアントで処理を行う．また，送信するデータの全てではなく，トランザクションにどのアイテムが含まれているかを示す行列に対してのみ暗号化を行うことで，処理の高速化を行っている．アイテム数やトランザクション数などは暗号化されないが，クライアント側でダミーデータを含ませておくことで，委

託先サーバに情報を推測されることを防ぐ方法を取っている．

4.2　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 [4]．

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 [5]．

4.3　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．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 アルゴリズムによる再計算は Apriori アルゴリズムによる再計算と比較して，約 3〜4 倍の計算時間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 [6]．

5　提案システム

本研究では，以下のようなシステムを提案する．

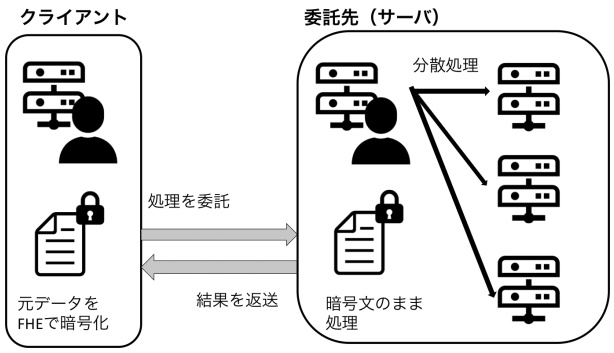

図 2　システム概観

図 1 のシステムにおいて，クライアント側で暗号化したトランザクションデータをサーバに送信し，FP-growth を使用した頻出パターンマイニングにおける一部の計算を委託する．サーバ側はマスタ・ワーカ型の分散・並列処理を行う．実装は C++ で行い，完全準同型暗号のライブラリは Helib [7]，分散・並列処理のライブラリは MPI を使用している．

プログラム全体の処理の流れの概要は以下の図 3 に示す．

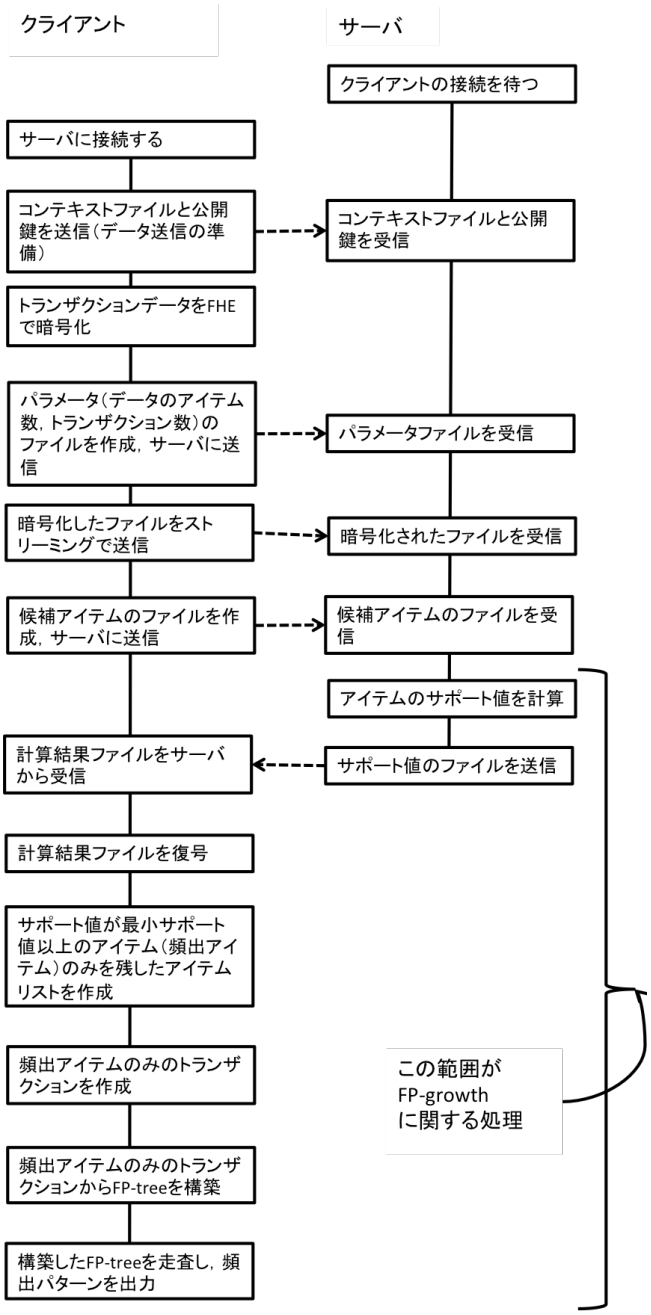

図 3　システムの処理の流れ

左の列に書かれた処理がクライアントで行うもの，右側のものがサーバで行うものである．上から下へ処理が進んでいくことを示している．左と右の処理の列の間をつなぐ点線矢印は，データの送信の方向を示したものである．クライアントの処理が終了していなくても，サーバはサポート値の計算結果をクライアントに送信した後はプログラムを終了する．

前半部分は，計算と FHE の処理に必要なデータをサーバに送信する処理が中心を中心に行い，後半部分は，FP-growth に関する処理を行う．本プログラムでは，FP-growth の処理のうち，FP-tree を構築する際に必要な，アイテムのサポート値の計算をサーバに委託する．FP-tree 構築から走査はクライアントプ

ログラムにて行っている．FP-tree 構築や走査を行うには多数回の比較演算を行うことが必要であるため，本研究ではサーバに委託せずに処理を行った．前半は

6 実　　験

6.1 実験概要

同一ネットワーク内の 2 台のマシンを用い，クライアントプログラムとサーバプログラムを各マシン上で動作させた際の実行時間と，使用リソースを測定した．

本実験のデータは，人工的に作成したものを用いた．データセットの生成は，IBM Quest Synthetic Data Generator で行った．データ作成においては，平均アイテム長，最大パターンの大きさ，アイテムの種類数，トランザクション数を指定した．各パラメータの値は表 1 のように指定する．

表 1　データ作成時のパラメータ

平均アイテム長	5
最大パターンの大きさ	5
アイテムの種類数	10, 20, 30
トランザクション数	3300, 6600, 9900

実験で用いた計算機の性能を表 2 に示す．

表 2　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
コア数	6
スレッド数	12
メモリ	512GB

クライアント，サーバ共に 2 に示す同型のマシンを使用した．また，本実験においては，サーバで行われるマスタ・ワーカ型の並列処理において，ワーカ数はすべて 1 としている．マスタとワーカは同一マシン上で動作する．

6.2 実験結果

サポート値計算，FP-tree 構築，FP-tree の走査，全体の実行時間をそれぞれクライアントプログラムで測定した．実験結果は図 4 に示す．図中の「その他」の実行時間には，元データを暗号化する時間，受信データの復号にかかる時間，サーバとの通信時間，入出力時間等が含まれている．各アイテム数ごとに示されている 3 つの棒グラフは，左から順にトランザクション 3300，6600，9900 の場合の実行時間を示している．

サポート値計算時間はアイテム数に比例している．FP-tree の構築時間は，全体の処理時間に対しての割合が非常に小さいが，アイテム数によって増加していることが確認できる．「その他」の処理はアイテム数やトランザクション数ではほぼ変化していない．

一方，FP-tree の走査時間は，アイテム数とトランザクション数の増加に伴い増加している．トランザクションの増加の影響よりも，アイテム数の増加の影響をより大きく受けることがわ

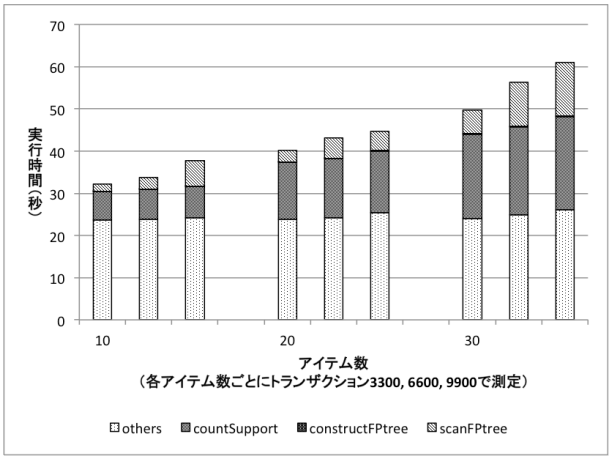

図 4　クライアントプログラムの実行時間

かる．アイテム数の増加により，tree が大規模になり再帰的に条件付き FP-tree を生成する部分においての処理時間が増加するためであると考えられる．また，本実験では最小サポート値を 0.1 としているが，サポート値を小さくするなど，より大きなパターンが出力される条件下において同様の実験を行う場合も，FP-tree が大規模になり全体の処理時間がさらに長くなると予想される．

続いて，図 5 と図 6 にはクライアントのマシン，図 7 と図 8 にはサーバのマシンのリソース使用量を示した．グラフに示しているのは，アイテム数 30，トランザクション数 9900 のデータを入力して実行した際のものである．マシンのリソースの取得には dstat を用い，CPU 使用率，メモリ使用量，ネットワークで送信・受信したデータ量を測定した．

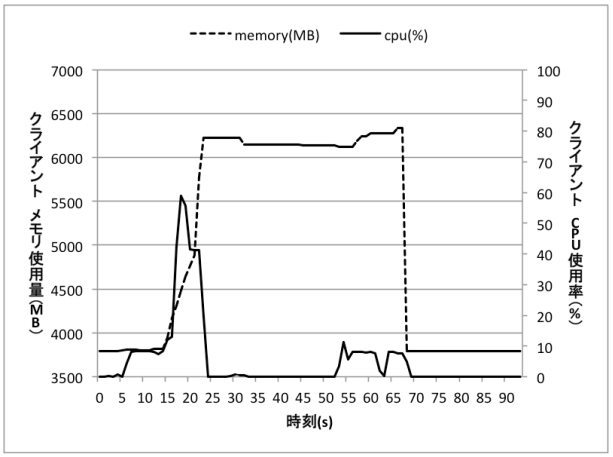

図 5　クライアントの CPU 使用率とメモリ使用量

クライアントの時刻 18～23 秒，サーバの時刻 36～37 秒頃，CPU 使用率の最初のピークがあり，メモリ使用量も大きく上昇している．これはデータ送受信の 2 回目のピークの時刻とも近いことがわかる．クライアントのマシンでは CPU 使用率のピークからデータ送信のピークが約 12 秒遅れている．この時の CPU 負荷はデータの送信前，受信後の FHE に関する処理によって起こっているものと考えられる．クライアントの時刻 54 秒，サーバの時刻 56 秒頃，クライアント時刻 53 秒ごろ，3

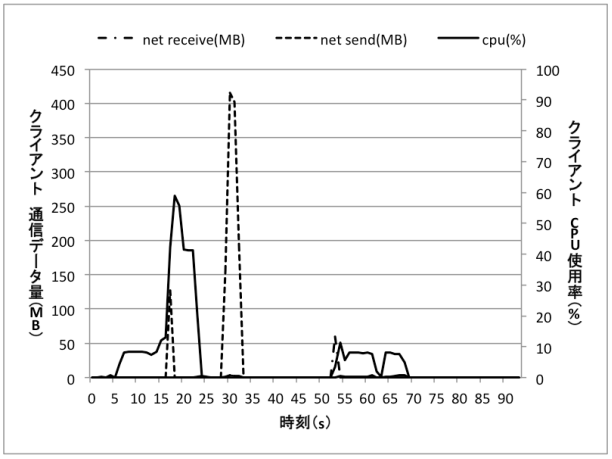

図 6　クライアントの CPU 使用率と送受信データ量

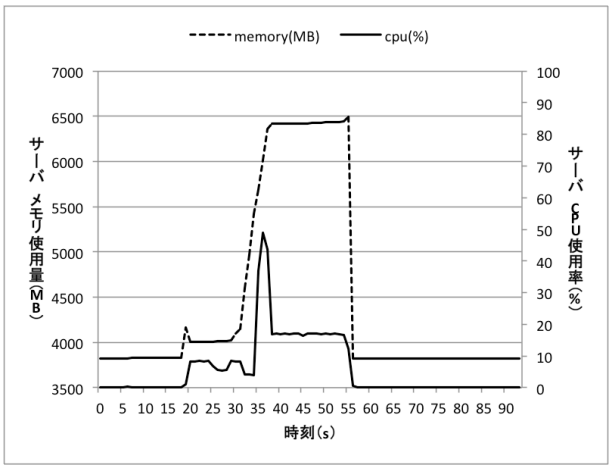

図 7　サーバの CPU 使用率とメモリ使用量

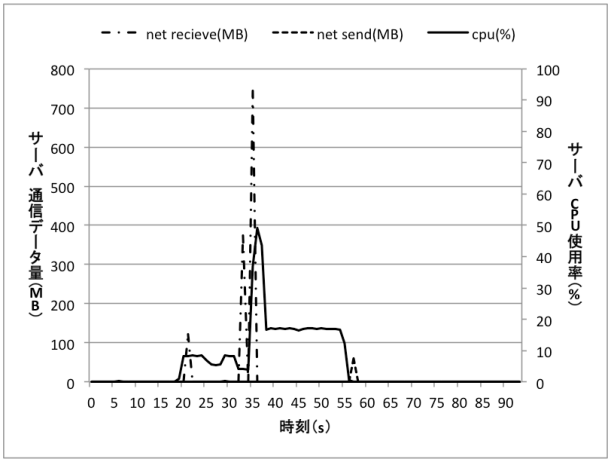

図 8　サーバの CPU 使用率と送受信データ量

回目のデータの送受信が行われており，2 回目の送受信と比較してデータ量がはるかに小さくなっている．2 回目の送受信では，全てのトランザクションデータを暗号化したファイルが送受信されているのに対して，3 回目では計算後のサポート値を暗号化したファイルのみが送受信されているため，処理によりデータが圧縮されているためであると考えられる．クライアントの時刻 55〜65 秒で，メモリ使用量と CPU 使用率が再び上昇している．メモリ使用率の上昇が階段状になっているが，これは FP-growth の tree 走査部分の処理において，再帰的に tree を構築する処理を行う部分でメモリを消費していることを示していると考えられる．

7　まとめと今後の予定

完全準同型暗号を使用した FP-growth アルゴリズムによる頻出パターンマイニングを一部サーバに委託するシステムを実装し，実行時間と使用リソースの測定を行った．クライアントでは暗号化と FP-tree の構築，走査を行い，FP-tree 構築時に必要となる各アイテムのサポート値の計算をサーバに委託した．測定の結果，FP-growth のにおいて処理時間がアイテム数とトランザクション数の増加に伴い長くなり，特にアイテム数への依存が大きいことがわかった．また，使用した 2 台のマシンのリソース使用量を測定したところ，FHE で暗号化したデータを送受信する際，マシンに特に負荷がかかることがわかった．
今後は，より大きなサイズの入力データに耐えうる，効率的な処理方法の検討をしていく．tree 走査部分の処理を部分的にサーバへ委託し，クライアント側のマシンで行う処理の削減に取り組む予定である．現状の実装では，サーバのプログラムはサポート値の計算を終えた後終了してしまうため，その後のクライアントで処理を行っている間はサーバは何も処理をしていない．クライアントの処理をより多くサーバに委託することにより，計算資源の有効活用を行っていきたい．サーバへの委託処理部分を増やすにあたっては，FHE 自体の処理時間が長く，マシンに負荷がかかることも考慮し，通信回数と送受信するデータ量を抑える工夫が必要であると考えている．
また，今回の実験は同一ネットワーク内のマシン 2 台のみで行ったが，想定しているシステムの実環境に近い環境での実行を行うため，外部のネットワークにあるマシンからサーバに処理を委託する実験を行っていきたい．

謝　　辞

本研究は一部，JST CREST JPMJCR1503 の支援を受けたものです．
また，研究を進めるにあたり，早稲田大学の山名早人先生をはじめ，多くの先生方からのご指導とご協力をいただいております．深く感謝申し上げます．

文　　献

[1] C. Gentry, ”A Fully Homomorphic Encryption Scheme,” Doctoral dissertation, 2009
[2] J. Han, J. Pei, and Y. Yin, ”Mining Frequent Patterns without Candidate Generation,” 2000
[3] 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. 7081. Springer, 2015
[4] 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

[5] 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
[6] 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
[7] V. Shoup , S. Halevi, ”HElib,” http://shaih.github.io/HElib/index.html, Accessed: 2018-10