ストリーム処理による安全頻出パターンマイニングの高速化

今林 広樹† 石巻 優‡ 馬屋原 昂‡ 佐藤 宏樹† 山名 早人†

†早稲田大学大学院 基幹理工学研究科 〒169-0072 東京都新宿区大久保 3-4-1

E-mail: †{imabayashi, yuishi, uma, hsato, yamana}@yama.info.waseda.ac.jp

あらまし 第三者のサーバ上で医薬品や生体情報などの機密データに対して解析を行う際，センシティブな情報の秘匿性と解析の正確性の双方が求められる. この秘匿性と正確性を保持したまま任意の解析を可能にする手法に，完全準同型暗号（FHE: Fully Homomorphic Encryption）がある．本稿では，データ解析手法の対象として頻出パターンマイニングを取り上げ, 信頼できない第三者サーバ上で FHE によって暗号化されたデータセットから頻出パターンを抽出することを考える. 我々は以前, FHE を頻出パターンマイニングに適用した場合に問題となる膨大な時間・空間計算量を大幅に削減するプロトコルを提案した．しかしこのプロトコルでは，1)サーバが全頻出パターン候補のサポート値を計算し終えるまでの待機時間が発生し後続の処理が遅延すること，2)サーバが全サポート値（暗号文）をクライアントに一斉送信する際に長い通信時間が発生すること，3)クライアント側で全サポート値を一斉復号するのに長時間かかること，が依然として課題である．そこで本稿では，計算処理が終了したサポート値から順にファイル書込，通信，ファイル読込，復号の一連の処理を行う（ストリーム処理）ことにより，処理時間の長いサポート値計算の間に，一連の処理を実行できるプロトコルを提案する．これにより，プロトコル内でデータ通信や復号のみに要する時間を削減することができる.

キーワード 頻出パターンマイニング，秘匿計算，完全準同型暗号，ストリームデータ処理 ，クラウドコンピューティング

1. はじめに

近年のクラウドコンピューティング及びビッグデータ関連技術の発展に伴い，データ収集・管理・解析などの各種処理をクラウド上で行う機会が増えている. 組織は顧客・個人情報，特許や企業秘密などの機密情報を蓄積しており，クラウド上でこれらのデータから解析結果を得る場合，解析対象の入力データだけでなく，解析途中のデータや解析結果の出力データもセンシティブな情報となる．特に医薬品や生体情報等のデータは，インサイダーや外部攻撃者の盗聴対象になりやすいため，そのような機密データを扱う医療機関等には，データを秘匿したまま解析を行うクラウド環境が必要となってくる.

本稿では，クラウド（サーバ）へのデータ委託解析を前提とした秘匿技術として，プライバシー保護データマイニング（PPDM: Privacy Preserving Data Mining）について考える[8]. PPDM の研究は，1) 入力プライバシー保護, 2) 出力プライバシー保護, 3) 暗号化システムの 3 つのアプローチに分類できる．入力プライバシー保護は，クライアントがサーバに解析対象のデータを委託する前に，そのデータに抽象化，ランダム化，ノイズ付加，匿名化等の操作により，センシティブな情報をデータ中から取り除くことができる手法である[7][17][18][20]. 出力プライバシー保護は，サーバが解析結果としてのデータを出力する前に，データへのノイズ付加や摂動化を行うことにより，センシティブな情報の漏洩を防ぐ手法である[4][5]. これらの入力・出力プライバシー保護手法は，1) データの保護範囲が，それぞれ入力データか出力データのどちらか一方である点, 2) データの曖昧性を高めることでセンシティブな情報を保護するため，解析結果も曖昧になる点，の 2 点から，機密データの安全性，及び解析結果の正確性が求められる医療分野には適さない．一方で，暗号化システム[10][11][12][13][14][15][16]は，暗号化されたデータに対して計算を行うことができるため，データの抽象化，匿名化，ノイズ付加などの曖昧性を高める処理を行う必要がない．そのため，データの安全性と解析結果の正確性の双方を同時に満たすことができる．そこで本稿では，PPDM の中でも暗号化システムによるアプローチを選択する.

完全準同型暗号（FHE: Fully Homomorphic Encryption）は，暗号化システムの一つであり，暗号文に対して任意回数の加算・乗算を行える暗号である[9]. この FHE は，統計計算[16]，機械学習アルゴリズム[10][14]，頻出パターンマイニング[11][13][15]といった様々なデータ解析の研究に応用されている. この FHE を用いた計算の課題点として，膨大な時間・空間計算量を消費することが挙げられる．時間計算量に関しては，暗号文サイズに伴う一つあたりの演算処理の大きさ，及び暗号文数に伴う演算回数の増加により，データ解析に膨大な実行時間を要する. また空間計算量に関しては，

大きな暗号文サイズにより，膨大なメモリ領域やストレージ領域を消費する．本稿では，基本的なデータ解析手法として知られる頻出パターンマイニングを取り上げ，FHE の適用により発生する時間・空間計算量を削減することを考える．

FHE を頻出パターンマイニングに応用した研究に，Liu らのプロトコル（P3CC: Privacy Preserving Protocol for Counting Candidate）[15]，及びその時間・空間計算量を削減した我々のプロトコル[11][21]（以後，参照はプロトコル[21]のみとする[1]）がある．P3CC では，1) サーバでのサポート値（あるパターンがトランザクション中に出現する頻度）の計算時に，適用している FHE 方式に起因する膨大な計算時間とメモリ領域（のりきらない場合はストレージ領域），及び 2) サポート値計算の待機時間，後続の計算結果（暗号文）の同時送信や復号に要する時間が課題となっていた．我々のプロトコル[21]では，FHE を他方式に変更し，暗号文パッキング手法と暗号文キャッシング手法の 2 つを適用することにより，課題 1)のサーバ側のサポート値計算時間とメモリ使用量を大幅に削減した．しかし，課題 2) の待機時間と通信時間については，依然として課題である．

そこで本研究では，頻出パターンマイニングプロトコルにおける待機時間と通信時間を削減することにより，プロトコルの高速化を図る．具体的には，我々のプロトコル[21]では，全サポート値計算終了までの待機時間の発生（処理遅延），全計算結果（暗号文）の一斉送信による大きな通信時間の発生，クライアント側での全計算結果の一斉復号時間の発生が課題となっている．そこで，計算処理が終了したサポート値から順に，ファイル書込，通信，ファイル読込，復号の一連の処理を行う（ストリーム処理）ことを考える．これにより，プロトコル中での処理遅延や一斉処理を省けるだけでなく，一連の処理を処理時間の長いサポート値計算の間に実行（処理時間の隠蔽）でき，結果的に各種処理のみに要していた時間を削減できる．

本稿の構成は，次の通りである．まず第 2 節で関連研究を紹介し，第 3 節で既存プロトコルについて概要を述べる．第 4 節では，ストリーム処理適用によるプロトコルの高速化について詳細に説明し，第 5 節で提案プロトコルの実行時間評価を行う．最後に，第 6 節でまとめと今後の課題を述べる．

2. 関連研究

本節では，暗号化システムをデータマイニングに適用した関連研究を紹介する．またその中で，本研究に最も関連する，FHE による頻出パターンマイニングについて，Liu らの P3CC，及び P3CC の時間・空間計算量を削減した我々のプロトコルについて具体的に説明する．

暗号化システムを用いたデータマイニングの研究[11][12][13][15]は，マルチパーティ・コンピューテーション(MPC: Multi-Party Computation)と準同型暗号（HE: Homomorphic Encryption）の 2 つのアプローチに大別できる．各アプローチの違いは，MPC がプログラム回路への入力をパーティ間で秘匿することで秘匿計算を実現するのに対し，HE は暗号化されたデータに対し演算を行うことで秘匿計算を実現することである．なお, HE は一定回数までしか演算できないが, FHE は任意回数の演算を可能にする点でより適用範囲が広い．

MPC を適用した研究として, Kantarcioglu ら[12]は，分散データベースに対して頻出パターンマイニングを行う際に，マルチパーティ間に漏洩する情報を保護するアルゴリズムを提案した．HE を適用した研究では，Kaosar ら[13]は，2-パーティ間の相関ルールマイニングにおいて，FHE によって暗号化された数値同士を比較する手法を提案し，同時に，MPC の相関ルールマイニングへの適用は，ストレージコスト，通信コスト，計算コストが大きいことから適さないことを示した．

本研究に最も関連する研究としては，Liu ら[15]の P3CC，及び P3CC の時間・空間計算量を削減した我々のプロトコル[21]が挙げられる．P3CC[15]では, 暗号文上での数値計算を可能にするため，各トランザクションがアイテム集合をもつ従来のデータベース（図 1(a)）を，行方向に各アイテム ID，列方向に各トランザクション ID を並べたバイナリ行列（図 1(b)）に変換している．Liu らが適用した FHE は，整数で暗号文を表現する Dijk ら[19] の方式であり，その性質により，上記バイナリ行列の各要素を暗号化することになる（図 1(c)）．P3CC では，次の 2 点が問題となる．

1) サーバでサポート値計算を行う際に，暗号文上で要素同士の独立な演算が必要になり，膨大な時間・空間計算量がかかる．
2) サポート値計算結果（暗号文）が全計算対象について集まった後に一斉送信されるため，待機時間と通信時間が発生する．また，その後クライアント側での全サポート値の一斉復号時間も発生する．

上記の問題点 1)に対し, 我々のプロトコル[21]では，以下の a) 暗号文パッキング，及び b) 暗号文キャッシングを適用することで，時間・空間計算量を大幅に削減した．問題点 2)に関しては，未だ解決されていないため，本研究の対象とする．

1 研究[21]は研究[11]について実験結果を加え，再度評価したものである．そのため，以後本文中では，我々のプロトコルを述べる際の参照は研究[21]のみとする．

a) 暗号文パッキング
一つの暗号文で複数の平文を扱うことにより（図 1(d)），生成される暗号文の数を削減し，同時にそれに伴う暗号文同士の演算回数も削減した．

b) 暗号文キャッシング
サポート値計算の途中結果として生成される暗号文をキャッシング・再利用することにより，類似パターンにおけるサポート値計算を高速化した．

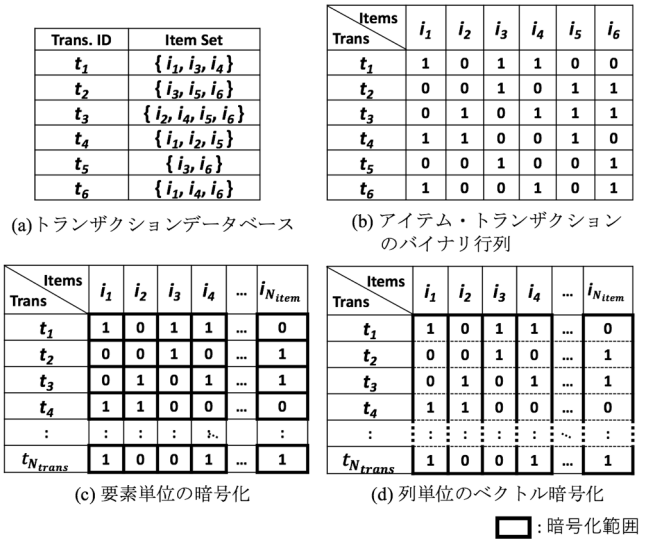


図 1　データベースのバイナリ化・暗号化[21]

3. FHE による頻出パターンマイニングプロトコル

本節では，既存研究で提案された FHE による頻出パターンマイニングプロトコルについて説明する．3.1 項では，Liu らの P3CC[15]で取り上げられている Apriori アルゴリズムについて，3.2 項では P3CC のプロトコル概要，及びその時間・空間計算量を削減した我々のプロトコル[21]について説明する．

3.1. Apriori アルゴリズム [21]

Agrawal と Srikant[3]は，図 1(a)のトランザクションデータベースから頻出パターンを抽出するアルゴリズムとして Apriori を提案した．以下に，頻出パターン（定義 1）及び Apriori アルゴリズムの手続きを示す．なお，説明の都合上，以後パターン長lの頻出パターン候補集合をC_l，パターン長lの頻出パターン集合をL_lとする．詳細な説明やアルゴリズムについては，我々の研究[21]を参照されたい．

① 入力としてトランザクションデータベース（図 1(a)）を受け取った後，アイテム毎にそれが含まれているトランザクション数をカウントする（サポート値計算）．ミニマムサポート値以上のアイテムを頻出アイテムとして抽出し，パターン長 1 の頻出パターン集合L_1を得る．

定義 1. 頻出パターン [2][21]

m個の重複しないアイテム集合を$I = \{i_1, i_2, \dots, i_m\}$，トランザクション集合を$T$とする．各トランザクション$t \in T$は，$I$の中の元からなるアイテム集合をもつ．つまり，$t$は，$i_k$をもつなら$t[k] = 1$，そうでなければ$t[k] = 0$を満たすような，$I$の部分集合となる．ここで，パターン$p$を$I$の部分集合とすると，$p$のもつ全てのアイテム$i_k$について，$t[k] = 1$が成立するときのみ，トランザクション$t$は$p$を満足するという．$p$のサポート値は，$p$を満足する，$T$の中のトランザクション数と等しい．また，そのサポート値が，与えられたミニマムサポート値以上であるならば，そのパターンは頻出である，とい

② パターン長$l+1$の頻出パターン候補集合C_{l+1}をL_lから生成する．以後，ステップ②-④を繰り返すが，1 回目のイタレーション（①の直後）では$l = 1$である（例：$l = 1$のとき，$L_1 = \{\{i_1\}, \{i_2\}, \{i_3\}\}$から$C_2 = \{\{i_1, i_2\}, \{i_2, i_3\}, \{i_1, i_3\}\}$を生成）．

③ C_{l+1}の各頻出パターン候補について，それが出現するトランザクション数をカウントし，サポート値を得る（例：$l = 1$のとき，C_2の各サポート値を 8, 9, 7 とする）．

④ C_{l+1}の各頻出パターン候補のうち，サポート値がミニマムサポート値以上となるものを選択し，パターン長$l+1$の頻出パターン集合L_{l+1}とする（例：$l = 1$のとき，ミニマムサポート値を 8 とすると，$L_2 = \{\{i_1, i_2\}, \{i_2, i_3\}\}$となる）．

⑤ パターン長lをインクリメントして，②-④を繰り返す．②でC_{l+1}が生成されない場合イタレーションを終了し，各イタレーションで生成された頻出パターン集合の総集合が最終的な出力となる．

3.2. P3CC と計算量削減プロトコル [21]

Liu ら[15]は，FHE の適用先として Apriori アルゴリズム[2][3]を対象とした，頻出パターンマイニングプロトコル（P3CC: Privacy Preserving Protocol for Counting Candidate）を提案した．Apriori には，1) 頻出パターン候補集合の生成，2)サポート値計算，3)サポート値とミニマムサポート値の比較後，頻出パターン集合の生成，の 3 つのステップがある（3.1 項参照）．P3CC では，サーバ側で 2)のサポート値計算部分のみを暗号文に対して行い，クライアント側で 3)の数値比較（各サポート値を復号後，平文上での比較）と頻出パターン集合の生成，1)の頻出パターン候補集合の生成を行う．クライント側で処理 1)や処理 3)を行うのは，P3CC で用

いられる FHE 方式が暗号文上での数値比較をサポートしていない[2]ためである．P3CC において暗号文で表現されるのは，サーバ側で保持しているアイテム・トランザクションバイナリ行列の各要素（図 1(b)参照），及び計算後のサポート値のみである．また，バイナリ行列中の各アイテムは，アイテム ID（平文）によって表現され，アイテム名はサーバ側から秘匿される．

我々のプロトコル[21]でも，P3CC と同様，上述の条件やプロトコルに従っている．異なる点は以下に説明する a) 暗号文パッキング適用による暗号文の表現変化，及び b)暗号文キャッシング適用によるキャッシングステップの追加である．これらの手法により，P3CC の課題の一つであったサポート値計算における膨大な時間・空間計算量を削減した．

a) 暗号文パッキングの適用
P3CC ではバイナリ行列の各要素を暗号化するのに対し，我々のプロトコルでは複数の要素を一つのベクトルにつめこみ，それを暗号化する（図 1(d)参照）．一つの暗号文が内部に複数の要素情報を格納できることにより，生成される暗号文数が削減され，サポート値計算時に必要な暗号文同士の演算回数も削減される．暗号文同士の演算は，ベクトル要素同士の演算として内部で並列化される．

b) 暗号文キャッシングの適用
サポート値計算において，同様の内部演算を行う場面が多い．例えば，頻出パターン候補$\{i_1, i_2, i_3\}$のサポート値計算の内部演算には，頻出パターン候補$\{i_1, i_2\}$のサポート値計算と同じ演算が含まれている．ここで，i_1, i_2間の演算結果（以後，中間値と表記）をキャッシュしておくことで，i_1, i_2, i_3間の演算に再利用することができ，これによりサポート値計算が高速化される．

以下に我々のプロトコルの手続きを述べる．暗号文パッキングにより①の暗号化ステップと③や⑤のサポート値計算ステップの処理が変化し，暗号文キャッシングにより，①内部でのキャッシングと⑥のキャッシングステップが追加される．それ以外の手続きは P3CC と変わらない．なお，詳細な説明やアルゴリズムについては，我々の研究[21]を参照されたい．

① クライアントは，公開鍵と秘密鍵ペアを生成し，公開鍵でバイナリ行列の複数要素を一つのベクトルに詰め込み暗号化する（図 1(d)）．暗号化が終了後，公開鍵と暗号化バイナリ行列をサーバに送る．
② クライアントは，パターン長 1 の頻出パターン候補集合C_1を平文のままサーバに送る．ここでC_1は，バイナリ行列中の全アイテム ID を要素にもつアイテム集合 I と一致する．
③ サーバは，各パターン$c \in C_1$ (各アイテム $c \in I$) に対応するバイナリ行列の列を読み込み，暗号文上で各サポート値を計算する．計算時に読み込んだ暗号文は，以後の再利用のためメモリにキャシングする．使用全アイテムについて計算終了後，クライアントに全サポート値（暗号文）を一斉に送る．クライアントは，全サポート値について復号後に，各サポート値とミニマムサポート値を比較することにより，パターン長lが 1 の頻出パターン集合L_1を得る．（3.1 項 Apriori の手続き①に対応）
④ クライアントは，L_lから頻出パターン候補集合C_{l+1}を生成し，それを平文のままサーバに送る．以後，ステップ④-⑦を繰り返すが，1 回目のイタレーション（③の直後）では$l = 1$である．つまり，$l = 1$のときL_1からC_2を生成する．（3.1 項 Apriori の手続き②に対応）
⑤ サーバは，各頻出パターン候補$c \in C_{l+1}$ についてキャッシュから再利用可能なパターン長lの頻出パターン候補（平文）をキーに，対応する中間値（暗号文）を読み込み，サポート値を計算する．（3.1 項 Apriori の手続き③に対応）．
⑥ 各サポート値計算時の中間値（暗号文）を，そのパターン長$l+1$の頻出パターン候補とセットにしてキャッシングしておく．全ての頻出パターン候補について計算終了後，全サポート値（暗号文）を一斉にクライアントに送る．
⑦ クライアントは，復号後の各サポート値とミニマムサポート値を比較し，頻出パターン集合L_{l+1}を生成する．（3.1 項 Apriori の手続き④に対応）
⑧ パターン長lをインクリメントして，④-⑦を繰り返す．④でC_{l+1}が生成されない場合イタレーションを終了し，各イタレーションでクライアント側に生成された頻出パターン集合の総集合を最終的な出力とする．（3.1 項 Apriori の手続き⑤に対応）

4. ストリーム処理によるプロトコル高速化

本節では，我々のプロトコル[21]が対処していない課題である「サーバ側での全サポート値計算結果の待機時間」「それらの一斉送信による通信時間」「クライ

2 有田と中里[1]による任意の数値に対する暗号文上での比較手法の適用，もしくは数値のバイナリ表現により数値比較は可能であるが，条件分岐先の全パターンについて演算を実行する必要があり，計算量が爆発的に増大するため，本研究の高速化の目的には適さない．

アント側での全サポート値の復号時間」を削減(隠蔽)する手法とプロトコルを提案する．なお，3.2 項のプロトコルの手続きに従った上で，本処理を適用する．

図 2 に，3.2 項のプロトコル手続きにおいて，サーバが頻出パターン候補集合を受信してから，全サポート値（暗号文）を送信し，クライアントがそれらを復号するまでのプロセスを具体化する．また，以下 3 つの課題をどのように解決するかを示す．

- 全サポート値の計算終了までの待機時間
- 全計算結果（暗号文）の一斉送信による大きな通信時間
- クライアント側での全計算結果の一斉復号時間

図 2 に示すように，①サーバがクライアントから頻出パターン候補集合を受け取った後，②サポート値計算が完了した頻出パターン候補から順に，③対応するサポート値のファイル書込，④通信，⑤ファイル読込，⑥復号の一連の処理を行う（ストリーム処理）．

これにより，他の頻出パターン候補について②の処理を行っている間に，②の処理を既に終えた頻出パターン候補は続きの③～⑥の処理を進めることができるようになる．つまり，②と③の処理間の待機時間，④通信時間，⑥一斉復号時間を，②サポート値計算を行う処理時間に隠蔽することが可能になり，結果的にプロトコルの実行時間を削減することができる．図 3 にストリーム処理適用前と適用後の違いを示す．なお，図 3 中の各種処理ステップ番号についても，図 2 中のそれと対応している．

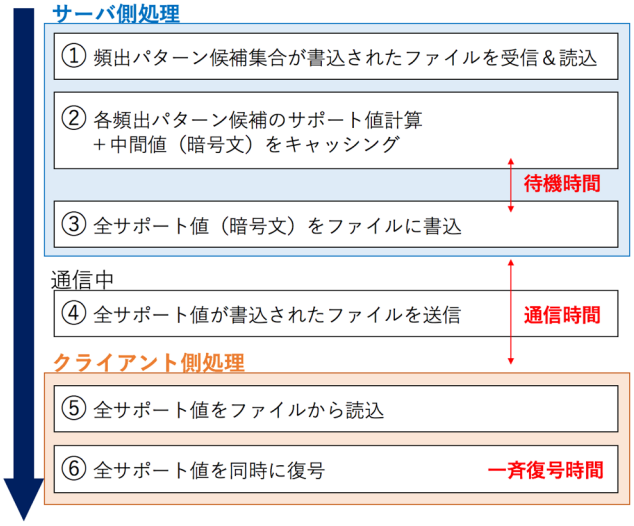
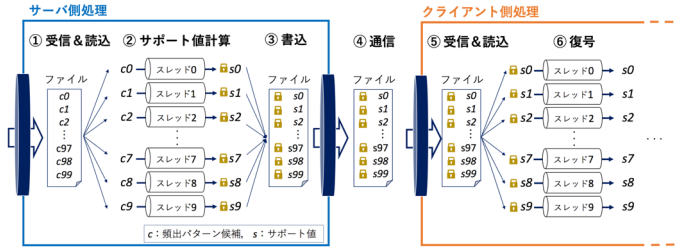
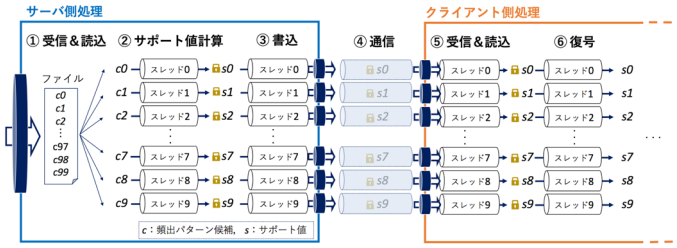


図 2　処理の流れと処理の遅延



(a) ストリーム処理適用前



(b) ストリーム処理適用後

図 3　ストリーム処理適用によるプロトコル比較

5. 実験評価

本節では，我々のプロトコル[21]（3.2 項参照）に対してストリーム処理を適用し，プロトコルの実行時間から有用性を評価する．5.1 項で実験環境を説明し，5.2 項で実験結果を述べる．

5.1. 実験環境

本実験評価に用いるデータセットは，Frequent Itemset Mining Dataset Repository (FIMI)[3] に公開されている chess データセット（3,196 トランザクション，75 アイテム）である．ミニマムサポート値は 90%に設定する．

本実装には，BGV 方式[6]による公開 FHE ライブラリの HElib[4]，多倍長整数演算ライブラリの GMP[5]，整数多項式を扱うライブラリである NTL[6]を用いた．FHE を構築するため，HElib のパラメータ $\{p, r, k, l, c, w\}$ を事前設定する．本実験では，$\{p, r, k, l, c, w\} = \{2, 14, 80, 20, 3, 64\}$ と設定した．ここで p^r は平文空間，k はセキュリティパラメータ，l は FHE の回路の深さ（レベル），c はキースイッチ行列の行数，w は秘密鍵に用いるハミング重みである．平文空間 2^{14} は D の最大設定値まで扱うための値，k, c, w は HElib のデフォルト値である．また，本実験では，P3CC[15]と同様に bootstrapping[7]を用いないため，暗号文上で適当数の演算を可能とするレベル l を設定している．

実験環境は，同一ネットワーク内のクライアントとサーバの 2 つのマシンから構成され，10Gb Ethernet で

3 http://fimi.ua.ac.be/data/

4 http://shaih.github.io/HElib/index.html

5 https://gmplib.org/

6 http://www.shoup.net/ntl/

7 FHE 暗号文には，暗号文強度のためノイズと呼ばれる項が含まれている．このノイズは，回路レベル l まで演算の度に増加し，ある閾値を超えると復号エラーとなる．bootstrapping は，l までの計算結果の正確性を保持したまま，暗号文のノイズを小さくする手法である．

接続されている．クライアントの CPU は Intel Xeon CPU E5-2643 v3（3.4GHz），メモリは 512GB であり，サーバの CPU は Intel Xeon CPU E7-8880 v3（2.3GHz），メモリは 1TB である．OS は，CentOS6.6 を両マシン上に動かしている．

5.2. 実験結果

本実験では，以下の実行時間 A，B の 2 つを測定する．ストリーム処理適用による各実行時間の変化をみる．

A. 全体の実行時間
クライアントが暗号化データをサーバに送った直後から，Apriori が終了するまで（3.2 項手続き②から⑧まで）

B. ストリーム処理適用箇所の実行時間
サーバが頻出パターン候補集合を受信してから，全サポート値（暗号文）を送信し，クライアントがそれらを復号するまで（3.2 項手続き⑤から⑦の途中まで）

表 1 にストリーム処理適用による実行時間 A，B の変化を示す．実行時間 A について，ストリーム処理により，23.5%の時間が削減（1399 秒→1070 秒）された．また，実行時間 B については，26.2%の時間が削減（1252 秒→924 秒）された．また，「ストリーム処理適用箇所の実行時間（B）が全体の実行時間（A）に占める割合」が削減されている（89.5%→86.4%）ことから，従来各種処理にかかっていた時間が，処理時間の長いサポート値計算の時間に隠蔽されていることが確認できる．

表 1　ストリーム処理適用による実行時間の変化

	適用前	適用後	削減時間・割合
実行時間 A	1399 秒	1070 秒	329 秒（23.5%）
実行時間 B	1252 秒	924 秒	328 秒（26.2%）
B/A	89.5%	86.4 %	

6. おわりに

本稿では，FHE による安全頻出パターンマイニングの課題であった，サーバ側での全サポート値計算が終了するまでの待機時間，全ての計算結果の同時送信にかかる通信時間，クライアント側で計算結果の一斉復号にかかる時間について，大きく時間を削減するためのストリーム処理プロトコルを提案した．実験評価では，ストリーム処理を適用した箇所において，26.2%の実行時間削減を確認した．

今後の課題として，ストリーム処理のさらなる高速化を図るため，ネットワーク帯域の考慮，ディスク I/O の最適化，関数実行スケジューリングの工夫を行う必要がある．

謝辞

本研究は，科学技術振興機構（JST）CREST の支援を受けたものである．

参考文献

[1] Arita, S. and Nakasato, S.: Fully homomorphic encryption for point numbers, Cryptology ePrint Archive: Report 2016/402, Cryptology ePrint Archive (online), available from <https://eprint.iacr.org/2016/402> (accessed 2017-01-07).

[2] Agrawal, R., Imielin´ski, T. and Swami, A.: Mining association rules between sets of items in large databases, *Proc. the 1993 ACM SIGMOD*, pp. 207–216 (1993).

[3] Agrawal, R. and Srikant, R.: Fast algorithms for mining association rules, *Proc. The International Conference on Very Large Data Bases (VLDB 1994)*, pp. 487–499 (1994).

[4] Atzori, M., Bonchi, F., Giannotti, F., et al.: Anonymity preserving pattern discovery, *The International Journal on Very Large Data Baes*, vol. 17, pp. 703–727 (2008).

[5] Bhaskar, R., Laxman, S., Smith, A., et al.: Discovering frequent patterns in sensitive data, *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2010)*, pp. 503–512 (2010).

[6] Brakerski, Z., Gentry, C. and Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping, *Proc. The 3rd Innovations in Theoretical Computer Science Conference (ITCS 2012)*, pp. 309–325 (2012).

[7] Evfimievski, A., Gehrke, J. and Srikant, R.: Limiting privacy breaches in privacy preserving data mining, *Proc. ACM SIGMOD-SIGACT- SIGART Symposium on Principles of Database Systems (PODS 2003)*, pp. 211–222 (2003).

[8] Gellman, R.: Privacy in the clouds: risks to privacy and confidentiality from cloud computing, *Proc. World Privacy Forum* (2009).

[9] Gentry, C.: Fully homomorphic encryption using ideal lattices, *Proc. Annual ACM Symposium on Theory of Computing (STOC 2009)*, pp. 169–178 (2009).

[10] Graepel, T., Lauter, K. and Naehrig, M.: Ml confidential: Machine learning on encrypted data, *Information Security and Cryptology–ICISC 2012, LNCS*, vol. 7839, pp. 1–21 (2012).

[11] Imabayashi, H., Ishimaki, Y., Umayabara, A., et al.: Secure frequent pattern mining by fully homomorphic encryption with ciphertext packing, *International workshop on Data Privacy Management (DPM 2016)*, LNCS, vol. 9963, pp. 181-195 (2016).

[12] Kantarcioglu, M. and Chris, C.: Privacy-preserving distributed mining of association rules on horizontally partitioned data, *IEEE Transactions on Knowledge and Data Engineering (TKDE 2004)*, vol. 16, pp. 1026-1037 (2004).

[13] Kaosar, M.G., Paulet, R. and Yi, X.: Fully homomorphic encryption based two-party association rule mining, *Data & Knowledge Engineering,* vol. 76, pp. 1–15 (2012).

[14] *Khedr, A., Gulak, G.* and Vaikuntanathan, V.: Shield:

Scalable homomorphic implementation of encrypted data-classifiers, *IEEE Transactions on Computers*, vol. 65, No.9, pp. 2848-2858 (2015).

[15] Liu, J., Li, J., Xu, S., et al.: Secure outsourced frequent pattern mining by fully homomorphic encryption, *Big Data Analytics and Knowledge Discovery, LNCS,* vol. 9264, pp. 70–81 (2015).

[16] Naehrig, M., Lauter, K. and Vaikuntanathan, V.: Can homomorphic encryption be practical?, *Proc. 3rd ACM workshop on Cloud computing security workshop*, pp. 113–124 (2011).

[17] Qiu, L., Li, Y. and Wu, X.: Protecting business intelligence and customer privacy while outsourcing data mining tasks, *Knowledge and Information Systems,* vol. 17, no.1, pp. 99–120 (2008).

[18] Tai, C.H., Yu, P.S. and Chen, M.S.: k-support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining, P*roc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2010)*, pp. 473–482 (2010).

[19] Van Dijk, M., Gentry, C., Halevi, S., et al.: Fully homomorphic encryption over the integers, *Advances in cryptology–EUROCRYPT 2010*, LNCS, vol. 6110, pp. 24–43 (2010).

[20] Wang, Y. and Wu, X.: Approximate inverse frequent itemset mining: Privacy, complexity, and approximation, *Proc. IEEE International Conference on Data Mining (ICDM 2005)*, pp. 482–489 (2005).

[21] 今林広樹, 石巻優, 馬屋原昂ほか: 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), TOD73 号(2017)[8].

8 巻数，号数，ページ数は 2017 年 2 月 4 日現在未定.