3C4-4

MapReduce を用いたセキュアな頻出パターン抽出手法の提案
An Approach to Secure Frequent Pattern Mining Using MapReduce

小林由真 Yuma Kobayashi　王家宏 Jiahong Wang　児玉英一郎 Eiichiro Kodama　高田豊雄 Toyoo Takata

岩手県立大学 ソフトウェア情報学部
Faculty of Software and Information Science，Iwate Prefectural University, JAPAN

Today it is often required to discover association rules from databases that are located at multiple geographically-separated areas and belong to different companies or organizations. Accompanying with the distributed association rule mining has the risk of privacy leakage and the high cost in computation and communication. This research addresses the subject of mining association rules from multiple autonomous databases efficiently and securely. A distributed privacy-preserving association rule mining approach is proposed, which is characterized by its high collusion-resistance ability and low computation and communication cost.

keywords: Frequent Pattern Mining, Privacy Preservation, MapReduce

1. はじめに

近年，企業で取り扱われるデータが電子化され，データベースへ蓄えられることが一般的になった．蓄えられたデータを活用するために，頻出パターンマイニングというデータマイニング技術がよく利用されている．これは，「牛乳を買う人はたまごも買う」のような相関ルールと呼ばれる項目間の相関関係を表す頻出パターンを抽出する技術である．

企業は一般的に複数地域にまたがってビジネスを展開する．例えば，インターネットショッピングモール(以下電子モール)は数多くの通販サイトを出店し，コンビニエンスストアチェーン(以下コンビニチェーン)は各地域に数多くの店舗を持っている．

そこで，各通販サイトや店舗が共通する有益な情報を抽出するために，全ての通販サイトや店舗に置かれたデータベースから分散データマイニングを行う必要がある．しかし，分散データマイニングには，自身のデータが他者に知られてしまうプライバシー漏洩の防止と大規模データを扱うことによる計算時間の低減という2つの課題がある．例えば，電子モールの場合は，対象となるデータは利用者の購入履歴となり，他方の通販サイトへ情報を漏洩してしまうことは深刻な問題である．また，各通販サイトはオンラインシステムであり，購入履歴が時間とともに蓄積されており，データベースは大規模化なものであり，データマイニングのコストは高くなる．よって，計算の効率化が求められている．

以上の問題を解決するために，2つの技術の利用が考えられる．1つは，プライバシーを保護した分散データマイニング技術であり，CRDM[浦邊 07]アルゴリズムが挙げられる．しかし，CRDM は大規模なシステムにおいては通信コストが高く，計算コストが高い問題点がある．もう1つは，分散処理で用いるプログラミングモデルである MapReduce が挙げられる．MapReduce は，1つの処理を複数のノードで並列的に行うことができる．大規模データを扱うために設計されており，ビッグデータを対象にデータマイニングを行うためによく利用される．分散データマイニングにおいては，複数のデータベースを対象として処理を行うために，入力データが大規模になりやすい．よって，MapReduce での実装は効果的であると考えられる．しかし，MapReduce を直接適用した場合には，プライバシー漏洩の可能性が残ってしまう．

連絡先：小林由真，岩手県立大学ソフトウェア情報学部
g031k056@s.iwate-pu.ac.jp

本研究では，CRDM に MapReduce を適用することで，分散データマイニングにおける高いプライバシー保護性能の維持と計算効率の向上を目指す．

2. 関連研究

最も広く利用されている頻出パターンマイニングアルゴリズムとして，Apriori アルゴリズムが知られている．Apriori アルゴリズムを MapReduce 上で実現した研究として，Ming-Yen Lin らの研究[Lin 12]がある．この研究においては，プライバシー保護については考慮されていない．セキュリティを考慮した研究として Indrajit Roy らの研究[Roy 10]が知られている．この研究では，MapReduce において扱うデータの値をぼかすことによって，プライバシー保護の向上を図っている．しかし，ぼかす値が大きくなるほどセキュリティが向上されるが，正確な値が出力されなくなるという特性をもっている．Apriori アルゴリズムを MapReduce で実装したほとんどの関連研究では，基本的に計算効率の向上に成功しているが，プライバシー保護についての課題が残っている．

分散データマイニングの関連研究として，CRDM[浦邊 07]がある．この研究では，分散するデータベースから共通する相関ルールを抽出している．また，プライバシー保護のために，自身のサイトのサポート値を分割し，他サイトとデータを共有する手法をとっている．これにより，暗号化などの負荷の重い処理を行わずに高い結託耐性を実現している．しかし，データ共有のための通信コストが高く，またサポート値の集計を単一サイトで行っているために処理効率が悪いという課題が残っている．

3. プライバシーを保護したデータマイニング

分散データマイニングには，プライバシー保護と大規模データを短時間で処理することが要求される．そのため，2節で述べた2つの手法を合わせて用いることで，これらの要求を満たす．本節では，結託耐性の高い CRDM をベースに，集計処理を MapReduce で実装する，プライバシー保護と処理効率の向上を図った頻出パターンマイニング手法を提案する．

3.1 システムモデル

コンビニチェーンなどいくつかの店舗(サイトと呼ぶ)を持つ企業がこのシステムを利用して，利用者の購入履歴からよく一緒に購入される商品に関する相関ルールを抽出することを考える．

各サイトが同じ種類の商品を扱い，利用者の購入履歴が格納されているトランザクションデータベースを保持しており，トランザクションデータベースの商品 ID は共通しているものとする．システムは定期的に全てのサイトのトランザクションデータベースから共通する相関ルールを抽出することを目的として利用される．この時，プライバシー保護のため，あるサイトの商品売り上げ個数と頻出するアイテムセットを他のサイトに知られないようにする．

システムに参加するサイトをノードとして扱う．ノードの集合を S，ノードの総数を M，各ノードを $S_i(0 \leqq i < M)$と記す．また，サイトの1つを管理ノードとして扱い，他を参加ノードとする．管理ノードを S_0 で記す．例えば，コンビニチェーンの場合は，本部が管理ノード，支店が参加ノードとなる．

ノード S_i が持つデータベースを DB_i と表す．また，DB_i に格納される各トランザクションを$(T_{i,1}, T_{i,2},\cdots)$とし，$DB_i = \{T_{i,1}, T_{i,2},\cdots, T_{i,Ni}\}$, $DB = \cup DB_i$ とする．ノード S_i が持つアイテムセットを X_i，X_i に対するサポート値を V_i と表す．X_i に含まれるアイテムの数を X_i の長さと呼び，現在求めている頻出アイテムセットの長さを len と表す．アイテムセットが頻出とされる基準を表す最低サポート値を min_sup と表す．min_sup 以上のサポート値を持つアイテムセットを頻出とする．

トランザクションデータベースとは別に，各ノードが2つのデータマイニング作業用データベースを保持する．1つは，自身のノードの各アイテムセットとそのサポート値を格納するためのデータベースであり，$MyDB_i$ と表す．もう1つは秘匿された値を格納するためのデータベースであり，$SecureDB_i$ と表す．

CRDM と同様に，本研究も Semi-Honest モデルに従って，システムを設計する．これは，ユーザがシステムを改ざんすることなく，システムプロトコルに則って利用することを示すモデルである．ただし，ユーザは入力と出力のデータを得ることができる．実際には，システムを改ざんすることにより，不正な結果を出力するといったことが可能になってしまう．しかし，各ノードが協力してより有益な情報を出力することがこのシステムを利用する際の目的であるため，ユーザがシステムへのハッキング行為を行わないことを前提とする．

3.2 基本アルゴリズム

基本アルゴリズムは，以下のステップを繰返し全ノードから頻出アイテムセットの抽出を行う．(1)各ノード S_i が長さ len のアイテムセットに対するローカルサポート値を求める．(2)求めたローカルサポート値からグローバルサポート値を求める．ローカルサポート値の集計は，図1と図2で示すローカルサポート値を秘匿しつつグローバルサポート値を全ノードで共有する手順を用いて，プライバシー保護処理を施す．また，後述する MapReduce による集計の機能を用いて，参加ノードが並列で効率よく集計処理を行う．集計結果は管理ノード S_0 が管理する．(3)サポート値の集計後，min_sup 以上のサポート値を持つアイテムセットを頻出するアイテムセットとして扱い，S_0 が全ノードへブロードキャストする．また，S_0 は頻出するアイテムセットから，長さ$(len + 1)$の頻出アイテムセットの候補を生成し，全ノードへブロードキャストする．

図1と図2は，ローカルサポート値を秘匿しつつグローバルサポート値を全ノードと共有する機能を示す．各ノードの共通する頻出アイテムセットを求めるには，各ノードのサポート値の合計が必要となる．そのために，各ノードのサポート値を管理ノード S_0 へ集め，S_0 で集計するという手法が用いられる．しかし，サポート値をそのまま S_0 へ送信してしまうと，簡単に自身のノードの持っているサポート値が漏洩してしまう．それを防ぐために，サポート値をランダムな値に分割し，その一部を他のノードと共有

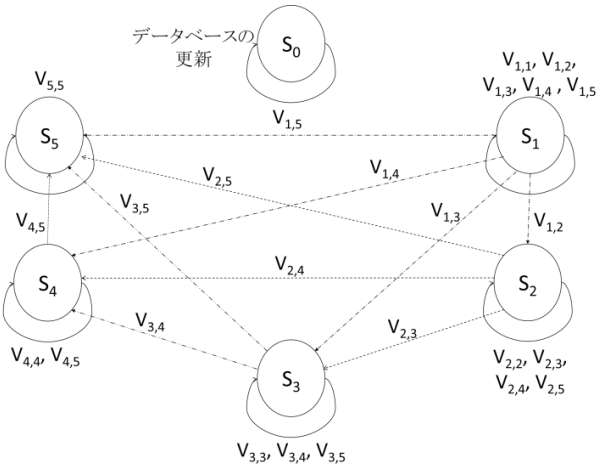

図 1　各ノードのサポート値の共有

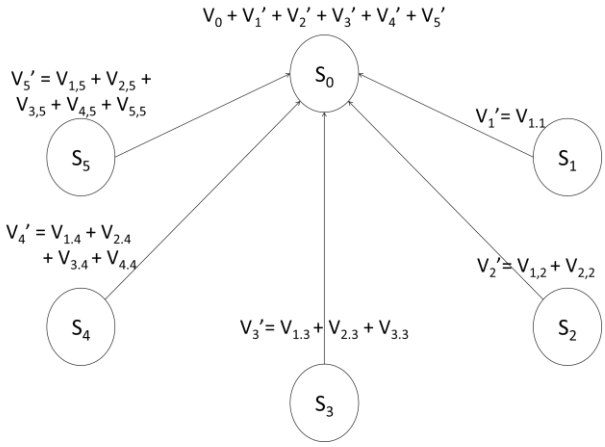

図 2　秘匿されたサポート値の集計

することで，送信するデータを秘匿する手法を用いる．

ノード S_i が持つローカルサポート値 V_i を分割した値を $V_{i,j}$ $(i \leqq j < M, V_i = \sum_{j=i}^{M-1} V_{i,j})$，秘匿されたサポート値を V_i' $(V_i' = \sum_{j=1}^{i} V_{j,i})$とする．図1は，参加ノード S_i が分割したサポート値 V_i を各ノードへ共有する流れを示している．各ノードは，V_i を$(M - i)$個のランダムな値で分割し，その一部 $V_{i,j}$ をノード S_j へ送信する．これにより，各ノードは自身の分割した値，或いは自身と他ノードの一部で構成された値を持つ(図1)．

図2は，秘匿されたサポート値を生成し，管理ノード S_0 へ送信する流れを示している．図1に示した全てのノードが分割したサポート値を送信後，送信された値を集計して V_i'を生成し，S_0 へ送信する．これにより，各ノードのサポート値を秘匿しながら S_0 へ全ノードのサポート値を集計することができる．しかし，全ノードのサポート値の集計処理は S_0 のみで行わなければならず，処理の負荷が集中し計算効率が落ちてしまう．よって，この集計処理を MapReduce で実装することで，参加ノードを用いた並列分散を行い，処理効率の向上を図る．

3.3 MapReduce による集計

本研究では，MapReduce のフレームワークとして Hadoop を用いる．Hadoop は ASF が開発した大規模データを効率的に処理するためのソフトウェア基盤で，プログラミング部である MapReduce と独自の分散ファイルシステムである HDFS (Hadoop Distributed File System)で構成される．MapReduce の処理は，入力データを key と value に分ける Map 処理と，集計処理を行う Reduce 処理に分けられる．ここでは，key を頻

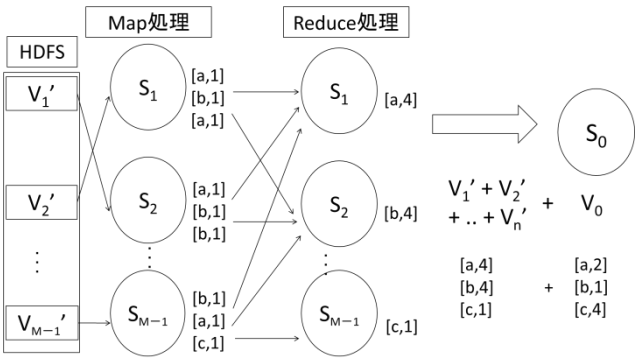

図 3 MapReduce を用いたサポート値の集計

出アイテムセット候補, value を秘匿されたサポート値とする. 集計されたサポート値を管理ノード S_0 へ出力する. MapReduce で実装されたサポート値の集計について図3に示す.

始めに各ノードが後述する HDFS から秘匿されたサポート値を入力として, Map 処理を行う. その後, Map 処理されたデータはそれぞれのノードで Reduce 処理として集計される. 最後に, 管理ノード S_0 へ参加ノードの集計されたサポート値を出力する. 図3より, 各ノードが並列で処理されていることと秘匿されたサポート値が S_0 へ出力されていることがわかる.

V_i'は HDFS に保存し, 入力として呼び出される. HDFS で保存された V_i'は複数のノードで分散して管理され, 処理実行時には Hadoop によって選択されたノードにて Map 処理が行われる. V_i'は Map 処理を通して key と value のペアに分けられ, Hadoop によって選択されたノードで Reduce 処理としてサポート値を集計して管理ノード S_0 へ出力する. この処理は並列で行われるため, 処理が集中していた CRDM の集計処理を分散して行うことで効率を上げることができる. 処理後, S_0 はこの出力と自身のサポート値 V_0 を集計することにより, 全ノードのあるアイテムセットのサポート値を集計することができる. 全ての長さ len のアイテムセットのサポート値の集計が終了すると, 集計された値と最低サポート値 min_sup と比較して, min_sup 以上のサポート値を持つアイテムセットを頻出アイテムセットとして抽出し, 各参加ノードへブロードキャストする.

長さ len の頻出アイテムセットの抽出を行った後は, Apriori アルゴリズムを用いて長さ(len + 1)の頻出アイテムセット候補を生成する. 生成後, 各参加ノードへ頻出アイテムセット候補をブロードキャストし, サポート値の集計へ処理は戻る.

3.4 通信コストの低減について

上述した基本アルゴリズムの欠点として, サポート値を共有する際に, ノード数が多いほど通信回数が増加し, 処理速度の低下に繋がる. この原因は 3.2 節に述べたサポート値の秘匿処理にある. 秘匿処理ではアイテムセットごとに(M − i)個のノードと通信しなければならず, アイテムセットとノード数の増加と共に通信コストも増加してしまう. そこで, システムを長期利用することを前提とした設計に変更することで, 処理の効率化を図る. 具体的には, 一度抽出したアイテムセットは $SecureDB_i$ に格納したままにして, 次回以降データベースを更新するシステムにすることによって, 通信コストを下げる. 修正したアルゴリズムを以下に示す. このアルゴリズムは長さ len の頻出アイテムセットを抽出する. このアルゴリズムを頻出アイテムセットが抽出されなくなるまでループさせることにより, 全ての頻出アイテムセットを抽出することができる.

以降, アイテムセット X が既に MyDB と SecureDB に格納されている前提で, 本アルゴリズムについて説明する. Step1 では,

Algorithm: 各ノードの値を秘匿しながら, 集計を行う
Input:
(1) X: 頻出アイテムセット候補, Z: 頻出アイテムセット候補の集合(X ∈ Z), len: 抽出する頻出アイテムセットの長さ(len > 1)
(2) $MyDB_i$: S_i が持つ X とそのサポート値を格納するデータベース
(3) $SecureDB_i$: S_i が持つ関連ノードからの X の断片を格納するデータベース
(4) min_sup: 最小サポート値

Output : min_sup を上回る長さ len のアイテムセット

Step1: 各ノード S_i は, DB_i から X の出現数をサポート値として計算し, X とそのサポート値を $MyDB_i$ に追加する. X が既に存在している場合は, 更新前後の差分を求めておき, 値を更新する.

Step2: $MyDB_i$ に格納されている X のサポート値を分割し, 他の参加ノードへ送信する. このとき, X が既に $SecureDB_i$ に格納されており, かつ X のサポート値を更新する場合は, ランダムな数のランダムなノードへ, Step1 で求めていた差分を送信する.

Step3: S_j は, Step2 で受信した X の断片を $SecureDB_j$ へ更新する.

Step4: 各参加ノード S_k は, $SecureDB_k$ から Hadoop を用いた X のサポート値の集計処理を行い, 管理ノード S_0 へ送信する.

Step5: S_0 は, 自身の持つ X のサポート値と Step4 で得られた X のサポート値の合計を集計し, アイテムセットが頻出かどうかを判断する. 頻出である場合, 頻出アイテムセットとして各参加ノードへブロードキャスト送信する.

Step6: 各ノードは, Step5 で頻出とされなかったアイテムセットが $SecureDB_k$ に格納されていれば, そのアイテムセットを削除する.

Step7: Z の全ての要素について以上の処理を行った後, 管理ノードは頻出アイテムセットから長さ len+1 のアイテムセット候補を生成し, 各参加ノードへブロードキャストで通知する.

X のサポート値を計算する. サポート値を $MyDB_i$ へ更新すると共に, 更新前後の差分を求める. Step2 では, サポート値の差分を (ランダムなノード数 + 1)だけランダムな値で分割しランダムで決められたノードへ送信する. Step3 から Step5 までは 3.2 節で述べたとおり, SecureDB の更新を行った後, 参加ノードで集計処理を行い管理ノード S_0 へ送信する. S_0 は自身の X のサポート値と参加ノードからのサポート値を集計し, X が頻出かどうか判断する. Step6 では, X が頻出でなかった場合, 全ノードの SecureDB から X を削除する. このシステムを利用していく上で, 処理を実行すればするほどデータベースが大きくなっていく. このデータベースの中に, 次回使われないデータも含まれており, 定期的にこの不要なデータを削除する必要がある. そのため, 頻出と見なされなかったアイテムセットを Step6 で $SecureDB_i$ から削除することで, データベースの利用の効率化を図る.

通信コスト削減のための改善方法として, SecureDB の更新機能を導入する. 各 SecureDB は MyDB を分割したサポート値が格納されている. MyDB が更新されたとき, 更新された差分を SecureDB へ加算することにより更新を行う. この更新はランダムなノードに対してのみ行い, 全体的な通信コストを削減する. 通信コストの削減量について, 改善前では, 各ノード S_i が(M − i − 1)個のノードへ一対一の通信を行っていた. 改善後は, (i + 1)から(M − 1)までの範囲のランダムな数のノードへ送信となっている.

3.5 利用例

コンビニチェーンの例を基にシステムの流れについて説明する. 各店舗は, 1ヶ月に1回システムを利用し, 共通する頻出アイテムセットを抽出する. また, ノード S_i の前回利用した途中結果は $MyDB_i$ と Secure DB_i に格納したままにする.

始めに各店舗は自身の店舗のみでローカルデータマイニングを行い, よく一緒に購入されている商品のアイテムセットとそのサポート値を $MyDB_i$ へ格納する. その後, 店舗は $MyDB_i$ の

中に保存されているアイテムセットとそのサポート値を秘匿しながらサポート値を全ノードと共有する機能を用いて共有を行い，$SecureDB_j$ へ格納する．この時，前回抽出された頻出アイテムセットである場合，全ノードではなくランダムな数のノードにのみ更新内容を送る．次に $SecureDB_j$から MapReduce による集計機能を用いて，店舗のサポート値を集計し，本部へその結果を送信する．本部は，店舗のサポート値の合計と本部のサポート値を集計し，頻出アイテムセットを抽出する．よく一緒に買われた商品は店舗へブロードキャストして知らせ，店舗は頻出アイテムセットを把握する．また，頻出で無いと判断されたアイテムセットが $SecureDB_j$ に格納されていた場合，それを削除する．その後，Apriori アルゴリズムを用いて，アイテムセットの長さが1つ長い頻出アイテムセット候補の生成を行う．これを繰り返し行うことで，よく一緒に買われる共通の商品の集合を抽出する．

4. 考察

本節では，提案手法の実装上の注意点について検討を行い，その結託耐性について分析を行う．

4.1 実装について

図3を見るだけでは，HDFS は各ノードで参照できる状態にあり，秘匿できる状態には見えない．また，Hadoop の仕様として，耐故障性を高めるために処理中のデータはデータブロックごとにレプリカが生成されてしまう．これにより，他ノードへ V_i'を余計に渡してしまい，秘匿性が下がってしまう可能性がある．この問題に対しては，Hadoop の Transparent Encryption 機能を用いて，中のデータを暗号化することで秘匿の状態を保つことで対処することができる．この暗号化は Key-Management ノードと呼ばれるノードが管理し，処理と並列に暗号化と復号化を行うことができる．Key-Management ノードは，結託性に大きく影響しない管理ノードが兼用して機能することにより，暗号処理を行うことができるものと考えられる．また，HDFS は論理的な参照を行うために，Semi-Honest モデルを採用する前提条件で，HDFS のデータがどのノードからのレプリカであるか他のノードからは判断がつかない．よって，秘匿性は確保できるものと考えられる．

処理効率については，Hadoop を用いた並列処理により，集計処理の効率化は改善することができると考えられる．しかし，Hadoop 利用時には各ノードと多数の通信を行わなければならず，通信によるオーバーヘッドの増加が懸念される．これは，集計処理の効率化と Hadoop の通信コストにどちらに重点をおくかで判断するべきものと考えられる．

4.2 結託耐性について

結託耐性を，あるノードのプライバシーに関わるデータを得るために結託する必要のあるノードの数で測る．これは，いくつまでのノードの結託に耐えられるかを示している．ここでは各ノードのサポート値 V_i をプライバシーに関わるデータとし，本提案手法の結託耐性について分析を行う．

3.1 節に述べた通り，提案手法は Semi-Honest モデルに則り設計されている．そのため，MapReduce 利用時のレプリカ配置の際に，自ノードはどのノードへレプリカが配置されたか，また，他ノードはどのノードからレプリカが配置されたかわからない．さらに，レプリカまたは処理中のデータは Hadoop の機能により暗号化されており，Key-Management ノードが管理しているため，結託耐性は CRDM と同様であると考えられる．しかし，ある前提条件を満たすことにより，結託耐性が CRDM より低くなる．

ノード S_x をターゲットとし，S_x が保持するサポート値 V_x が漏洩するために必要なノードの結託数を調べる．結託耐性に影響を及ぼす条件として3つ挙げられる．(1)集計処理開始時に生成された V_i'が，Hadoop 実行時に他の参加ノードへレプリカとして配置されること．(2) この配置されたレプリカ V_i'の中身を見ることができること．(3) S_x が参加ノードであることである．このすべての前提条件を満たしたとき，本提案手法の結託耐性が CRDM の$(M-2)$から$(M-3)$に下がる．以下に，前提条件を満たした場合の結託耐性の証明を示す．

V_x の漏洩のためには，$V_{x,x}, V_{x,x+1}, \cdots, V_{x,M-1}$ が必要となる．$V_{x,x+1}, V_{x+2}, \cdots, V_{x,M-1}$ は，各ノード $S_{x+1}, S_{x+2}, \cdots, S_{M-1}$ が結託し，S_x から送られた値から得ることができる．$V_{x,x}$ については，レプリカとして配置される V_x'から得ることができる．V_x'は，$V_{1,x}, V_{2,x}, \cdots, V_{x,x}$ の和である．すなわち，V_x'から $V_{1,x}, V_{2,x}, \cdots, V_{x-1,x}$ の和を減算することで $V_{x,x}$ を求めることができる．$V_{1,x}, V_{2,x}, \cdots, V_{x-1,x}$ は，各ノード $S_1, S_2, \cdots, S_{x-1}$ と結託し，S_x へ送信した値から得ることができる．よって，結託耐性は，$(M-3)$となり，管理ノードとターゲットとなるノードを除く参加ノードと結託が行われない限り，V_x の漏洩が起こらない．

結託耐性低下の前提条件について述べる．(1)のレプリカ配置の前提条件については，Hadoop を用いる以上，耐故障性を確保するためにレプリカが自動的に生成されるため，避けることはできない．また，レプリカの参照については，暗号化機能を持たない Hadoop2.6.0 以前のバージョンを利用した場合も満たしてしまう．この条件を満たす場合，上記の証明により，結託耐性が$(M-3)$となる．すなわち，ノード数が 4 以上でなければ，このシステムの結託耐性が無効になってしまう．尚，集計処理に参加しない管理ノードはレプリカが存在せず，他ノードへの値の受け渡しも行っていないため，CRDM と同じ$(M-2)$の結託耐性を持つ．

対策として，Hadoop2.6.0 以降のバージョンを使用することが挙げられる．このバージョン以降は暗号化機能を用いてレプリカの秘匿が行うことができるので，2つ目のレプリカ参照の条件を満たすことができなくなり，結託耐性は CRDM と同等になる．

5. まとめ

本論文は，電子モールやコンビニチェーンのような複数地域にまたがってビジネスを展開している，大規模なデータベースを有する企業における相関ルールマイニングのプライバシー保護と処理速度向上を目的とした．先行研究で提案した分散データマイニングアルゴリズム CRDM に MapReduce プログラミングモデルを適用することで，高い結託耐性を維持したまま処理効率の向上した分散データマイニング手法を提案した．提案手法は，サポート値の秘匿処理に参加するサイトがランダムで決められ，CRDM と比べて通信コストが低くなることが考えられる．システムの実装し，性能評価を行う予定である．

参考文献

[浦邊 07] S. Urabe, J. Wang, E. Kodama, T. Takata: A High Collusion-Resistant Approach to Distributed Privacy-preserving Data Mining, IPSJ Transactions on Databases Vol. 48, No. SIG 11, pp.104--117, 2007.

[Lin 12] M. Lin, P. Lee, S. Hsueh: Apriori-based Frequent Itemset Mining Algorithms on MapReduce, In Proc. the 6th Inter-national Conference on Ubiquitous Information Management and Communication (ICUIMC`12), 2012.

[Roy 10] Indrajit Roy, Srinath T.V. Setty, Ann Kilze, Vitaly Shmatikov, Emmett Witchel: Airavat: Security and Privacy for Map-Reduce , In Proc. the 7th USENIX conference on Networked systems design and implementation, pp.297--312, 2010.