秘匿した状態で処理可能なデータベースの設計

桐淵 直人 †　　五十嵐 大 †

†NTT セキュアプラットフォーム研究所
〒 180-8585 東京都武蔵野市緑町 3-9-11
{ kiribuchi.naoto, ikarashi.dai }@lab.ntt.co.jp

あらまし　本稿ではデータベースからの情報漏洩を防ぐため，格納するデータを秘匿し，サーバに元のデータが一切知られることなくデータベース処理ができるシステムを提案する．既存研究では，元のデータを類推される恐れのある暗号方式を用いていたが，本研究ではマルチパーティ計算を用いることで，従来よりも高い秘匿性を実現する．マルチパーティ計算は，複数の断片に分割することでデータを秘匿し，断片の状態で演算を行う技術である．提案した設計に基づき，基本的なデータベース演算を実行するシステムを実装し，大規模データの一括登録と一括取得における性能を測定する．ただし報告する性能は向上の途中であり，今後改善する予定である．

Design of a Database System Processable Under Keeping Data Confidentiality

Naoto Kiribuchi †　　Dai Ikarashi †

†NTT Secure Platform Laboratories,
3-9-11, Midori-cho Musashino-shi, Tokyo 180–8585 Japan
{ kiribuchi.naoto, ikarashi.dai }@lab.ntt.co.jp

Abstract　In this paper, to prevent information leakage from databases, we propose a system which can store and process data keeping its confidentiality. Although existing works which use encryption scheme with a risk of conjecture, our system realize higher confidentiality by multi-party computation. Multi-party computation is a technique to keep data secret by dividing data into pieces under secret sharing method and to process and work on the pieces without decrypting the original data. Based on the proposed system design, we implement a system which process basic database SQL operations, and the performance of system shows reasonable result for small table data set. Improving running time of batch copy to and from client for large scale data is our future work.

1　はじめに

IT システムが社会に浸透し，近年では大量の利用者情報がデータベースに蓄積されるようになった．こうした情報の中には個人の氏名やクレジットカード番号など，金銭目的の犯罪で狙われるものが含まれていることも珍しくなく，実際にデータベースに対する攻撃による情報漏洩が発生している．

こうした攻撃から情報を守るため，市場に出回っているデータベースは暗号化機能を備えており，特に近年では，上位のアプリケーションを変更せずにデータベースに格納されたデータを暗号化する透過的暗号化機能が登場している．

しかし，これらのデータベースは暗号化されたデータを暗号文のまま検索するといった処理ができず，対象データを全て復号するため，データベースの管理者権限を乗っ取る攻撃やメモリ上のデータを盗聴する攻撃に対しては，復号されたデータを盗みとられる危険性がある．

暗号化したデータを処理する際に復号しなければならないという問題は暗号学の主軸として研究されており，例えば 1977 年に発表された RSA 暗号 [12] は，暗号文の状態で乗算が可能であることが知られている．2009 年には暗号文の状態で加算と乗算の両方が可能である「完全準同型暗号」が提案され [5]，実用的な処理速度とは言い難いものの理論的には任意の論理回路を暗号化した状態で処理できることが証明された．

こうした秘匿した状態でデータを処理する技術は,「秘密計算」と呼ばれ，完全準同型暗号の他にも，検索可能暗号 [14]，順序保存暗号 [2]，マルチパーティ計算 [16] などが知られている．

ここ数年では秘密計算技術の高速化が進み，データベース処理を秘密計算技術によって実行することで，従来の暗号化では達成できなかった格納されたデータを復号せずに処理するデータベースの実装も報告されている [11, 15, 17]．

しかし，これらの実装はいずれも検索可能暗号や順序保存暗号といった特定の処理だけを秘密計算できる暗号方式を複数組み合わせたものであり，限られたデータベース処理しか秘匿した状態で処理できず，対応する処理を増やそうとして元のデータが類推される恐れのある暗号方式を用いている．

上記のような類推の恐れなく格納された情報を秘匿した状態でデータベース処理を行う手法として，マルチパーティ計算を用いる方式が志村らによって提案され [21]，五十嵐，濱田らを筆頭にデータベースにおける各種演算アルゴリズムの改良や性能の検証がなされてきた [18, 19, 20, 22]．

本稿では，これらのマルチパーティ計算のアルゴリズムを実行することで，格納されたデータを秘匿した状態でデータベース処理ができるシステムの設計と実装による検証を行う．

1.1 関連研究

データベースの処理を秘密計算で実行する研究には，クライアントが要求したクエリのデータを秘匿する研究とサーバに格納されたデータを秘匿する研究がある．

前者の実装報告としては，Pappas らの Blind Seer がある [10]．Blind Seer ではデータベースの各レコードが葉ノードに対応するように複数の Bloom フィルタ [1] で木構造を構成し，Yao の Garbled Circuit[16] により秘匿した状態で木構造の探索を行うことでクライアントからのクエリを秘匿した状態で大規模データ検索を実現した．しかし，Blind Seer の前提はサーバがデータの所有者であり，木構造の生成時にサーバは元データを参照するため，先に述べたデータベースに格納されたデータを取得する攻撃には対応していない．

サーバに格納されたデータを秘匿する実装報告としては，Popa らによる CryptDB[11]，Tu らによる monomi[15]，森らによるデータベース [17] があり，本研究もこちらに該当する．

CryptDB では，複数の暗号方式を用いて暗号化し，実行する演算に対応した暗号文により処理を行う．実際に用いられている暗号は，加算，乗算に用いられる準同型暗号 [9]，テキスト検索に用いられる検索可能暗号 [14]，等号判定に用いられる確定的暗号（CMC モードの AES），結合に用いられる暗号（独自方式）ソートに用いられる順序保存暗号 [2]，秘匿性を確保するためだけに用いられる暗号（CBC モードの AES）の 6 種類である．しかし，これらの機能を組み合わせて使うことはできないため，例えば `a + b < c` のような処理には対応していない．

monomi も実行する演算によって複数の暗号文を使い分ける点は CryptDB と同様である．monomi では，暗号方式が対応していない演算をクライアントとサーバの間に設置したプロキシで復号し，平文で処理することによって，対応する演算の種類を広げているが，プロキシに対してはデータが秘匿されていない．また，サーバで実行する処理において，暗号化したままの検索をするために確定的暗号を用いており，平文の分布がわかると元のデータを復号できる恐

れがある．

森らによるデータベースでは，確定的暗号を一意性制約を設定した列にしか使用しないことで前述の問題を回避しているが，最大値や最小値を求めるために用いられる順序保存暗号では，暗号文の大小関係が平文と対応するため，元データを類推される恐れが残っている．また，サーバだけで処理できない演算をプロキシで復号して処理する点は monomi と同様である．

マルチパーティ計算によりデータベース処理を行う手法は，志村らによって提案された [21]．マルチパーティ計算は，理論的には任意の演算に対応できるため，データをプロキシで復号して処理する必要がなく，検索可能暗号や順序保存暗号とは異なり元のデータを類推される恐れのないアルゴリズムを構成することができる．志村らは，関係データベースの演算モデルである関係代数演算の具体的な秘密計算アルゴリズムを示した．関係代数は，和，差，交わり，直積，制限，射影，結合，商の 8 種類の表演算からなる演算体系であり，データベースの表演算は全てこれらの 8 種類に帰着される．

さらに濱田らは，志村らのアルゴリズムを改良し [19]，表演算の他にも効率的な集約関数も提案している [18]．

本稿では，こうしたマルチパーティ計算による秘密計算のアルゴリズムを取り入れることで，データを復号するプロキシが不要で，かつ検索可能暗号や順序保存暗号とは異なり，元のデータを類推される恐れのないデータベースを設計し，基本的なデータベース演算に対応したシステムの実装により検証を行う．

2 準備

2.1 記法・定義

- p:素数．
- $\mathbb{Z}_p$: 0 以上 p 未満の整数の集合．
- $[\![s]\!]_p$: 秘密情報 $s \in \mathbb{Z}_p$ のシェアの集合．

2.2 (k, n) 閾値秘密分散法

秘密情報 $s \in \mathbb{Z}_p$ から n 個のシェアとよばれる断片情報を生成する手法である．元の情報 s は k 個以上のシェアを集めることによってのみ復元可能であり，高々 $k-1$ 個のシェアからは何もわからない．代表的な手法としては，Shamir の秘密分散法 [13] や複製秘密分散法 [3, 8] が知られている．

2.2.1 秘密分散に基づくマルチパーティ計算

n 人の参加者に (k, n) 閾値秘密分散法で生成したシェアを配布し，元の情報を復元することなく演算結果のシェアを得る手法である．Shamir の秘密分散法においては，$[\![a]\!]_p$ と $[\![b]\!]_p$ の各シェアの和が秘密情報 a, b そのものの和のシェア $[\![a+b]\!]_p$ になることが知られており，乗算プロトコルでは $[\![a]\!]_p, [\![b]\!]_p$ から積 $[\![ab]\!]_p$ を得ることができる [4]．秘密分散に基づくマルチパーティ計算では，和と積の両方を秘匿した状態で求めることができるため，AND 回路と NOT 回路を構成でき，理論的には論理回路で表現できる任意の演算を秘匿した状態で実行可能である．

3 システムの設計

3.1 システム構成

図 1 にシステム構成の概要を示す．提案するシステムは，データベースに対して要求を行うクライアント，(k, n) 閾値秘密分散法のシェアの形で秘匿されたデータを格納し，データベース演算を行う n 台の秘密計算サーバ，シェアの不整合を防ぐための管理サーバから構成される．クライアントは複数台存在してもよく，管理サーバは 1 台もしくは冗長性のための 2 台構成とする．

管理サーバは，2 相コミット [7] における調停者の役割を果たす．2 相コミットは，複数台のサーバに対してデータの一貫性を保つためのプロトコルであり，本システムでは，n 台の秘密計算サーバに対するデータの登録，更新，削除において用いる．

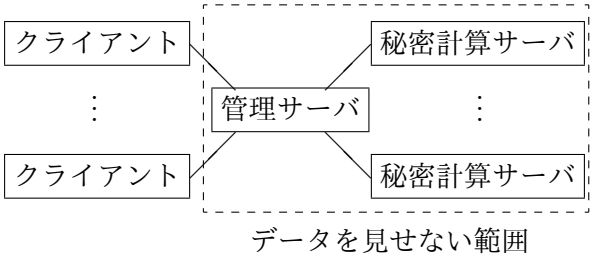

図 1: システム構成の概要

3.1.1 CAP 定理との関係

CAP 定理とは，Gilbert らによって証明された分散システムにおいて次の 3 つを同時に保証できないという定理である [6].

- 一貫性：全ノードにおいて同時に同じデータが見えること.
- 可用性：ノード障害時に生存ノードが応答を返すこと.
- 分断耐性：障害により通信可能なノード群が分断されても動作が継続すること.

提案するシステムは 2 相コミットにより一貫性を確保する．また，格納するデータを (k, n) 閾値秘密分散法で秘匿する際に $n > 2k + 1$ とすることで可用性を持たせることができる．分断耐性については，演算の度にマルチパーティ計算のアルゴリズムによる秘密計算サーバ同士での通信が必要となるため，保証していない．

よって提案システムは，分断耐性は無いが，データの一貫性を持ち，秘密分散のパラメータによって可用性も持たせることができる.

3.2 セキュリティ・モデル

提案するシステムにおいて秘匿する対象は，クライアントが秘密計算サーバに格納するデータの中身である.

図 1 において点線で囲んだのがデータを見せない範囲である．クライアントはデータの所有者であるため，データを取得し中身を知ることができる．複数のクライアントが存在する場合も，サーバに格納したデータの中身を互いに知ることができる.

秘密計算サーバおよび管理サーバは，データの中身を知ることができない．ただし，格納されたデータの件数，各テーブルの列数といったデータの中身に関わらない情報は秘匿する対象とはしない．また，最大値を取得しようとしているといったクライアントからのクエリの種別についても秘匿の対象としない.

3.3 システムの処理の流れ

本節では，データベースにおけるデータの登録，参照，更新，削除について，それぞれプロトコル 1,2,3,4 に提案するシステムの処理の流れを示す．なお，クライアントはテーブルを定義する際に，テーブルのどの列を秘匿するかを設定できるものとする.

プロトコル 1 登録処理の流れ

1: クライアントはデータを登録するテーブルを管理サーバに送信する.
2: 管理サーバは少なくとも 1 台の秘密計算サーバから対象となるテーブルの情報（どの列を秘匿するか等）を取得して，クライアントに返却する.
3: クライアントは秘匿して登録するデータからシェアを生成する.
4: クライアントは事前に各秘密計算サーバと共有した鍵を用いてシェアを暗号化する.
5: クライアントは暗号化済みデータを含む登録データを管理サーバに送信する.
6: 管理サーバは 2 相コミットにより n 台の秘密計算サーバにデータ登録を試み，登録が完了したかの情報を得る.
7: 管理サーバは登録が完了したかの情報をクライアントに返却する.

プロトコル 2 参照処理の流れ

1: クライアントは参照するデータの対象を管理サーバに送信する.
2: 管理サーバは少なくとも 1 台の秘密計算サーバから対象となるテーブルの情報を取得して，クライアントに返却する.

3: クライアントはテーブルの情報から参照処理に秘密計算処理が含まれるかどうか，含まれない場合，シェアが含まれるかを判定する．
4: クライアントは判定結果を元に必要に応じて参照条件に使われるパラメータを秘密分散し暗号化した上で，参照要求を管理サーバに送信する．
5: 管理サーバは，x 台の秘密計算サーバに参照要求を送信する．ただし，x は秘密計算処理が含まれる場合は $2k-1$ 以上，含まれない場合で参照データにシェアがある場合は k 以上，平文のみの場合は 1 以上の整数である．
6: 秘密計算サーバは，参照対象のデータを取得し，秘密計算処理が含まれる場合は対応する秘密計算アルゴリズムによる結果を得て，管理サーバに送信する．この際，シェアには事前にクライアントと共有した鍵で暗号化を施す．
7: 管理サーバは秘密計算サーバからの結果をクライアントに送信する．
8: クライアントは必要に応じて結果を復号，シェアから結果を復元して取得する．

プロトコル 3 更新処理の流れ

1: クライアントは更新するデータの対象を管理サーバに送信する．
2: 管理サーバは少なくとも 1 台の秘密計算サーバから対象となるテーブルの情報を取得して，クライアントに返却する．
3: クライアントは更新処理に秘密計算処理が含まれるかを判定し，秘匿する列に格納するデータについては秘密分散を得て事前に秘密計算サーバと共有した鍵を用いてシェアを暗号化する．
4: クライアントは更新処理に秘密計算が含まれるかの情報と更新するデータを管理サーバに送信する．
5: 管理サーバはクライアントから受信した更新要求情報を 2 相コミットにより n 台の秘密計算サーバに対して送信する．
6: 秘密計算サーバは更新箇所を特定する．更新条件に秘密計算処理が含まれる場合は，対応する秘密計算アルゴリズムを実行して更新箇所を得る．
7: 秘密計算サーバは更新を試み，更新結果を管理サーバに送信する．
8: 管理サーバは 2 相コミットにより更新の完了を試み，その結果をクライアントに返却する．

プロトコル 4 削除処理の流れ

1: クライアントは削除するデータの対象を管理サーバに送信する．
2: 管理サーバは少なくとも 1 台の秘密計算サーバから対象となるテーブルの情報を取得して，クライアントに返却する．
3: クライアントは削除処理に秘密計算処理が含まれるかを判定し，判定結果と削除条件するを管理サーバに送信する．
4: 管理サーバはクライアントから受信した削除要求情報を 2 相コミットにより n 台の秘密計算サーバに対して送信する．
5: 秘密計算サーバは削除箇所を特定する．削除条件に秘密計算処理が含まれる場合は，対応する秘密計算アルゴリズムを実行して削除箇所を得る．
6: 秘密計算サーバは削除を試み，削除結果を管理サーバに送信する．
7: 管理サーバは 2 相コミットにより削除の完了を試み，その結果をクライアントに返却する．

4 実装と性能評価

3 章で述べた設計に基づくシステムの実現性を確認するため，プロトタイプを実装した．

クライアントは，Windows 上で動作する C++ のプログラムで，CLI から SQL による操作が可能である．管理サーバと秘密計算サーバは CentOS 上で動作する Java のプログラムで，秘密計算サーバのデータを格納する部分については PostgreSQL を用いた．

今回のプロトタイプでクライアントが入力可能な SQL コマンドは，`CREATE TABLE`, `INSERT INTO`, `SELECT` の他，大規模データの登録のため

の COPY FROM，テーブルに含まれるデータを一括で取得する COPY TO コマンドも実装した．

クライアントで実行するデータの秘匿には Shamir の (k, n) 閾値秘密分散法 [13] を採用し，パラメータとして $k = 2, n = 3, p = 2^{61} - 1$ を用いた．

実装したコマンドのうち，データ量が処理時間に影響し，特に大量のデータを扱うことが想定される COPY FROM および COPY TO コマンドの性能を測定した．

4.1 測定環境

性能の測定に用いた構成は，クライアント 1 台，管理サーバ 1 台，秘密計算サーバ 3 台が 1Gbps の LAN で接続された環境で，測定に用いたマシンの仕様は表 1, 2 の通りである．3 台の秘密計算サーバと 1 台の環境サーバの仕様は同じものを利用した．

表 1: 測定環境（サーバ）

	秘密計算/管理サーバ
OS	CentOS 6.4
CPU	Intel Xeon E3-1220v2
コア数	4
プロセッサ数	1
メモリ	16GB
ディスク	100GB (SSD)

表 2: 測定環境（クライアント）

	クライアント
OS	Windows 7
CPU	Intel Core i5-2430M
コア数	4
プロセッサ数	1
メモリ	4GB
ディスク	320GB (HDD)

表 3: 測定結果

レコード数	COPY FROM（一括登録）	COPY TO（一括取得）
10^3 行	1.6 秒	2.9 秒
10^4 行	10.3 秒	22.3 秒
10^5 行	97.0 秒	220.2 秒
10^6 行	970.9 秒	2,348.8 秒
10^7 行	9,223.3 秒	23,899.0 秒

4.2 測定結果

測定結果を表 3 に示す．測定では，1 行が 100 列からなる 1,000 行から 1,000 万行のデータを用いた．データは測定用に生成されたもので，100 列のうち 60 列を秘匿するテーブルへの一括登録と一括取得を実行した．

測定結果から，一括登録と一括取得の両方において概ねレコード数に比例した処理時間であった．100 列 1,000 行のデータであれば登録，取得共に 3 秒以内に実行可能であり，対話的に実行できると言える．

また，100 列 1,000 万件のデータについては，登録が約 2.5 時間，取得が約 6.6 時間であったため，例えば夜間にバッチ処理を行うなどすれば，実行可能な時間内にある．

5 おわりに

本稿では情報漏洩を防ぐため，マルチパーティ計算を用いて，格納されたデータを秘匿した状態でデータベース処理ができるシステムの設計を実施した．さらに，設計に基づいた基本的なデータベース演算の実装と，大規模データの一括登録と一括取得の性能を測定し，現実的な時間内に処理可能であることを確認した．

今後は WHERE 句を含む SQL 文といったより高度な演算について実装と評価や更なる性能向上を行う予定である．

参考文献

[1] Bloom, B. H.: Space/time trade-offs in hash coding with allowable errors, *Communications of the ACM*, Vol. 13, No. 7, pp. 422–426 (1970).

[2] Boldyreva, A., Chenette, N., Lee, Y., AdamO'neill : Order-preserving symmetric encryption, *Advances in Cryptology-EUROCRYPT 2009*, Springer, pp. 224–241 (2009).

[3] Cramer, R., Damgård, I. and Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation, *Theory of Cryptography*, Springer, pp. 342–362 (2005).

[4] Gennaro, R., Rabin, M. O. and Rabin, T.: Simplified VSS and fast-track multiparty computations with applications to threshold cryptography, *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, ACM, pp. 101–111 (1998).

[5] Gentry, C.: Fully homomorphic encryption using ideal lattices, *STOC*, Vol. 9, pp. 169–178 (2009).

[6] Gilbert, S. and Lynch, N.: Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services, *SIGACT News*, Vol. 33, No. 2, pp. 51–59 (2002).

[7] Gray, J.: Notes on Data Base Operating Systems, *Operating Systems, An Advanced Course*, London, UK, Springer-Verlag, pp. 393–481 (1978).

[8] Ito, M., Saito, A. and Nishizeki, T.: Secret sharing scheme realizing general access structure, *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, Vol. 72, No. 9, pp. 56–64 (1989).

[9] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes, Advances in cryptology—EUROCRYPT'99, Springer, pp. 223–238 (1999).

[10] Pappas, V., Krell, F., Vo, B., Kolesnikov, V., Malkin, T., Choi, S. G., George, W., Keromytis, A. and Bellovin, S.: Blind Seer: A Scalable Private DBMS, *Security and Privacy (SP), 2014 IEEE Symposium on*, IEEE, pp. 359–374 (2014).

[11] Popa, R. A., Redfield, C., Zeldovich, N. and Balakrishnan, H.: CryptDB: protecting confidentiality with encrypted query processing, *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, ACM, pp. 85–100 (2011).

[12] Rivest, R. L., Shamir, A. and Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, Vol. 21, No. 2, pp. 120–126 (1978).

[13] Shamir, A.: How to share a secret, *Communications of the ACM*, Vol. 22, No. 11, pp. 612–613 (1979).

[14] Song, D. X., Wagner, D. and Perrig, A.: Practical techniques for searches on encrypted data, *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, IEEE, pp. 44–55 (2000).

[15] Tu, S., Kaashoek, M. F., Madden, S. and Zeldovich, N.: Processing analytical queries over encrypted data, *Proceedings of the VLDB Endowment*, Vol. 6, No. 5, VLDB Endowment, pp. 289–300 (2013).

[16] Yao, A.: How to generate and exchange secrets, *Foundations of Computer Science, 1986., 27th Annual Symposium on*, IEEE, pp. 162–167 (1986).

[17] 森　健吾，寺西　勇，荒木俊則，古川　潤：復号せずに処理する暗号化関係データベース，暗号と情報セキュリティシンポジウム (SCIS) 2014，電子情報通信学会 (2014).

[18] 濱田浩気，五十嵐大，千田浩司：秘匿計算上の集約関数中央値計算アルゴリズム，コンピュータセキュリティシンポジウム 2012 論文集，Vol. 2012, No. 3, pp. 509–516 (2012).

[19] 濱田浩気，五十嵐大，千田浩司：秘密計算上の関係代数演算アルゴリズムの改良，電子情報通信学会技術研究報告. LOIS, ライフインテリジェンスとオフィス情報システム， Vol. 112, No. 466, pp. 77–82 (2013).

[20] 濱田浩気，桐淵直人，五十嵐大：キーに重複がある場合の秘密計算向け結合アルゴリズム，暗号と情報セキュリティシンポジウム (SCIS) 2015，電子情報通信学会 (2015).

[21] 志村正法，宮崎邦彦，西出隆志，吉浦　裕：秘密分散データベースの構造演算を可能にするマルチパーティプロトコルを用いた関係代数演算，情報処理学会論文誌，Vol. 51, No. 9, pp. 1563–1578 (2010).

[22] 五十嵐大，千田浩司，濱田浩気，高橋克巳：軽量検証可能 3 パーティ秘匿関数計算の効率化及びこれを用いたセキュアなデータベース処理，暗号と情報セキュリティシンポジウム (SCIS) 2011，電子情報通信学会 (2011).