Vol.2015-HPC-150 No.23
2015/8/5

暗号化データベースシステムにおける総和計算処理の並列化

堀尾 健太郎[1,3,a)]　川島 英之[2,3]　建部 修見[2,3]

概要：暗号化データベースシステムはデータを暗号化することにより情報漏洩対策を行いながら，正規のユーザは従来のデータベースシステム同様に動作させることができるシステムである．様々な性質を持つ暗号化手法を用いることで，データベース上で平文が露わになることなく問合せ処理を行なうことができるが，暗号化によってデータサイズと演算コストが増大するため性能劣化が著しい．しかしながら既存研究ではこのような性能劣化を考慮している物は少ない．暗号化データベースはバックエンドに汎用の RDBMS を利用しているためクエリプロセス中に最適化の可能性がある．クエリプロセスの最適化として処理の並列化が挙げられる．効率的な暗号化データベースシステムである MONOMI で用いられている総和計算処理効率化手法を適用すると，並列データベースシステムで広く用いられるハッシュ分割には適さないデータレイアウトとなる．そこで MONOMI 方式の総和計算処理に適した並列化手法として，Round-Robin 分割を用いた方式を提案する．

1. はじめに

近年，クラウドコンピューティングサービスの普及により DBMS を第三者が管理するサーバ上に設置する場合が多くなった．DBaaS 等，ネットワーク経由でデータベースを提供するクラウドサービスも存在する．一方でベネッセ流出事件のように情報漏洩事件も起こっており [1][2][3][4]，DBMS におけるセキュリティの重要性は高まっている．一部の RDBMS は暗号化機能 [5] を持つためそれを用いて情報漏洩対策が行われることがあるが，既存の暗号化機能の多くはデータ処理が必要になった際に一時的に DBMS サーバ上で復号して処理を行うため，DBMS サーバが攻撃され管理者権限を奪取される場合や第三者であるサーバ管理者が情報の不正取得を企てるような場合に情報漏洩を防ぐことができない．このような問題を解決するセキュア DBMS として暗号化データベースシステムが注目されており，研究・開発が進められている．

CryptDB[6][7] に代表される暗号化データベースシステムは，問合せを仲介し暗号化・復号処理及び暗号化鍵の管理を行なうプロキシサーバとデータの管理を行なう DBMS サーバによって構成される．プロキシサーバはユーザから問合せを受け取ると，問合せ中の変数等を暗号化して DBMS サーバに送る．DBMS サーバは変数等が暗号化された問合せを処理し，暗号化された結果をプロキシサーバに返す．プロキシサーバは結果を復号し，平文の状態でユーザに結果を返す．DBMS サーバは暗号化されたデータを取り扱うのみであるため，第三者である DBMS サーバの管理者が情報の不正取得企てる場合や，DBMS サーバが攻撃されすべてのファイルを閲覧する権限を奪われるような場合にも，暗号が解かれない限りは情報漏洩を防ぐことができる．一方で暗号化データベースシステムは暗号化による性能劣化について考慮されているものは少なく，性能の点で実用には適さない可能性が高い．

我々はこれまで，暗号化データベースシステムの総和計算処理について注目し，暗号化データベースシステムの先駆け的研究である CryptDB，及びその高性能化を図った MONOMI[8] で用いられている手法について，スクラッチ実装した DBMS 上でミクロ的評価を行った．[9] では MONOMI で用いられている総和計算効率化手法の有効性を示し，一方でファイル I/O 及び暗号文上での計算が処理のボトルネックとなりうることを示した．又，既存研究ではバックエンドに PostgreSQL など汎用 RDBMS を用いているが，そのクエリプロセス中に効率化手法の適用による性能向上を阻害する要因が存在する可能性が示唆された．

総和計算処理に MONOMI 方式を用いる場合，計算処理及びファイル読み込みの並列化が可能であり，高性能化が期待できる．しかしながら既存研究では並列化を用いた高

[1] 筑波大学大学院 システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
[2] 筑波大学 システム情報系
Faculty of Engineering, Information and Systems, University of Tsukuba
[3] 国立研究開発法人 科学技術振興機構 CREST
JST CREST
[a)] horio@hpcs.cs.tsukuba.ac.jp

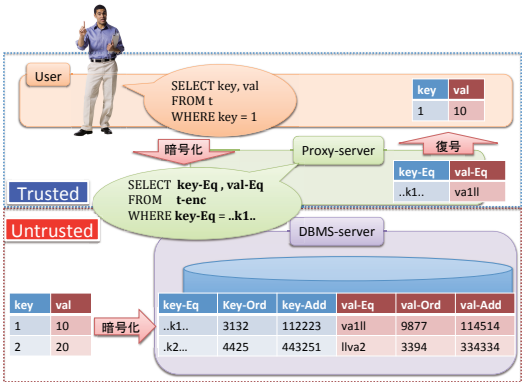

図 1　CryptDB の概要

性能化は行われていない．本稿ではグルーピング演算を含む総和計算処理の並列化手法を提案し，スクラッチ開発したマイクロ DBMS 上に実装する．

2. 先行研究

2.1 暗号化データベースシステム

暗号化データベースシステムについて CryptDB, MONOMI で用いられている仕組みを述べる．暗号化データベースシステムは, ユーザから発行された問合せと DBMS サーバから返ってきた結果の仲介及び暗号鍵の管理を行なうプロキシサーバ，暗号化されたデータの管理を行なう DBMS が稼働する DBMS サーバの 2 つのサーバにより構成される．ユーザから問合せが発行されると，プロキシサーバは問合せに含まれる変数の値，テーブル名を適切に暗号化して DBMS サーバに送る．DBMS サーバ上では平文が一切露わになることなく問合せが処理され，プロキシサーバに対して暗号化された結果を返す．プロキシサーバは結果を復号し，ユーザに返す (図 1)．

DBMS には暗号 Onion と呼ばれる多重に暗号化されたデータが保管される．暗号 Onion には 4 つの種別がある．等号チェックのみ可能な暗号文（Eq-Onion），大小関係チェックのみ可能な暗号文（Ord-Onion），加算演算のみ可能な暗号文（Add-Onion），文章から単語を検索することが可能な暗号文（Search-Onion）が用意されており，それぞれ異なる暗号化手法により暗号化される．元データの種類に応じて必要な Onion が生成されるため，1 つの値に対応する暗号文が複数用意されることになる．例えば数値型のデータを暗号化データベースシステムに保管する際には，そのデータに対応する Eq-Onion，Ord-Onion，Add-Onion が生成されデータベースに保管される．問合せ処理の際には問合せに応じて適切な Onion が参照される．

DBMS サーバ上には暗号化されたデータのみが置かれ，データ処理が必要な際にも平文まで復号されることはない．そのため，DBMS サーバが攻撃され管理者権限を奪取される場合や第三者であるサーバ管理者が情報の不正取得を企てるような場合にも情報漏洩を防ぐことが可能となる．

$$Enc(A) \times Enc(B) = Enc(A + B)$$

図 2　Paillier 暗号の持つ性質

2.2 暗号化データベースシステムにおける総和計算

暗号化データベースシステムにおける総和計算処理と，MONOMI で用いられている効率化手法について述べる．暗号化データベースシステムでは総和計算に Paillier 暗号 [10] を用いている．Paillier 暗号は加法準同型性という性質を持つ暗号化手法である．加法準同型性とは暗号文同士の乗算をすることで，平文を露わにすることなく平文同士の和を暗号化した暗号文を得ることができる性質である（図 2），これは暗号化データベースシステム上での加算処理に用いることが可能である．

CryptDB では 1 つの平文に対し 1 つの暗号文を生成している（CryptDB 方式）．Paillier 暗号では一般に 1024 bit の暗号鍵が用いられ，暗号文は 2048 bit となる．数値型（int 型，32 bit）のデータを暗号化する事を考えると，Add-Onion だけでデータサイズが 64 倍となる．また，加法準同型性によって実現される加算は実際には 2048 bit 整数同士の乗算であり，平文の状態での加算と比べると計算コストが大きい．MONOMI では複数の平文を結合して平文ブロックを作り，それを暗号化するという方式 [11] を用いている．本稿ではこの方式を MONOMI 方式と呼ぶ．平文ブロックに含まれる個々の平文をスライスと呼ぶ，平文ブロックを暗号化した暗号文同士を乗算して得られる暗号文を復号すると，各スライス同士が加算された平文ブロックが得られる．復号した後に平文ブロックの各スライスを足し合わせることで総和が求められる．1 つの平文ブロックに含めるスライスの数を S とする．MONOMI 方式での単純な総和計算処理は，暗号文データサイズは CryptDB 方式の $1/S$，暗号文の状態での乗算の回数が CryptDB 方式の $1/S - 1$ となる．$S = 4$ とした時にに 1 から 16 までの総和を求める場合の例を図 3 に示す．CryptDB 方式の場合，16 個の暗号文が格納され乗算は 15 回必要となるが，MONOMI 方式の場合 4 個の暗号文が格納され，乗算の回数は 3 回となる．

選択演算を含む総和計算を考えると一部の値を取り除いた加算が必要になるが，MONOMI 方式では 1 つの暗号文に複数の値が含まれておりその内の一部の値を取り除いて一度に加算することはできない．そこで各スライス毎に正しい部分和を求め，プロキシサーバ上で復号した後に正しい部分和のみを統合する，という方法で総和を得る．選択演算の際にはスライスごとに i 番目の暗号文に含まれる値が選択に含まれるかどうかを表すビット列を生成する．各スライス毎に，対応するビットが 1 の暗号文のみで総乗計算を行い，平文ブロックの該当するスライスに正しい部分

```
SELECT sum(val) FROM table WHERE key!=5 AND key !=10 AND key!=15;
```

図 5　選択演算を含む総和計算問合せ

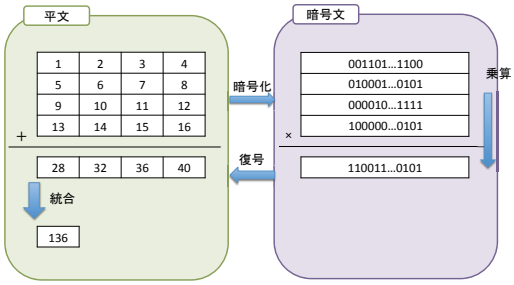

図 3　MONOMI 方式での総和計算の概要

$$
\begin{aligned}
&(\quad Enc\_Add(1\circ 2\circ 3\circ 4)\quad)^1\\
\times&(\quad Enc\_Add(5\circ 6\circ 7\circ 8)\quad)^0\\
\times&(\quad Enc\_Add(9\circ 10\circ 11\circ 12)\quad)^1\\
\times&(\quad Enc\_Add(13\circ 14\circ 15\circ 16)\quad)^1\\
=\ & Enc\_Add(23\circ 26\circ 29\circ 32)
\end{aligned}
$$

図 4　選択演算を含む総和計算

和を含む暗号文を生成する．プロキシサーバは正しい部分和を含んでいる S 個の暗号文を受け取り，復号してすべての正しい部分和を足し合わせる．$S=4$ として，1 から 16 までの数を含むテーブルから 5 の倍数を取り除いた和を求めることを考える．問合せは図 5 を想定する．先頭のスライスの正しい和を求める際の式を図 4 に示す．ここで $Enc\_Add()$ は Paillier 暗号化を行なう関数を表し，$\circ$ は平文の値の結合を表す．はじめに Eq-Onion で選択演算を行なう．Eq-Onion には 1 から 16 までの値がリストで保持されている．先頭のスライスに格納される値に注目すると，2 番目の暗号文に含まれる 5 を総和から取り除く必要がある．そこで先頭のスライスに対応するビット列には $\{1,0,1,1\}$ と保存する．Add-Onion ではビット列に従い 2 番目の暗号文のみ 0 乗して総乗計算をすることで，先頭のスライスのみ正しい部分和を含む暗号文が得られる．続く 3 つのスライスについても同様に計算し，すべての暗号文が得られたら復号して正しい部分和のみを足し合わせることで結果が得られる．

グルーピング演算は選択演算とほぼ同一の方法で実現できる．すなわち，グルーピングキーが取りうる値毎に，その値を持つ暗号文に対応するビットが 1 となるビット列を生成する．選択演算またはグルーピング演算を含む総和計算の場合，暗号文の状態での乗算回数は CryptDB 方式と MONOMI 方式で同一であるが，暗号文データサイズについては MONOMI 方式は CryptDB 方式の $1/S$ となる．

データレイアウトに関して，CryptDB 方式では平文と暗号文は 1 対 1 で対応しているため，行方向のデータをまとめて保持する NSM(N-ary Storage model) の DBMS（MySQL，PostgreSQL 等）に問題なくデータを保持させることができる．一方で MONOMI 方式では複数の平文が 1 つの暗号文に対応するため，特定の列のみ行数が合わなくなることから NSM でのデータの保持は困難である．MONOMI では，MONOMI 方式で暗号化した列のデータのみ別ファイルに保持する DSM(Decomposed Storage Model)[12] をとっている（図 8）．

これまでの研究 [9] で，MONOMI の形式は選択演算などを含まない単純な総和計算問合せの時に最も効率的に動作することが示された．一方で暗号文での計算処理には未だ大きなコストが掛かっていることが判明した．暗号化データベースシステムでの総和計算処理は，加算が乗算に置き換わっている点を除けば通常のデータベースシステムと同様の操作を行っている．そのため，処理の並列化など既存のデータベースシステムの効率化手法が暗号化データベースシステムにも有効である可能性がある．一方で MONOMI 方式では DSM データレイアウトを用いているため従来のデータ分割方式を用いた並列化手法は適さない可能性がある．そこで本稿では暗号化データベースシステムにおける総和計算処理について，並列化手法を提案する．

3. 提案

この節では図 6 に示す問合せ実行することを想定し，グルーピング演算を含む複数の総和計算を行なう問合せ処理を並列化する手法を示す．

3.1 Round-Robin 分割を用いた並列グルーピング演算

データベースシステムにおける処理の並列化ではデータを何らかの手法で分割し，分割されたデータに対して並列に処理を行い，結果をまとめる，ということが行われる．グルーピング演算においてはデータを分割し，分割したそれぞれで独立にグルーピングを行うことで分割数だけ並列にグルーピングが可能である．データの偏りの小さいデータ分割は並列処理の鍵となる要素の 1 つである．多くの商用並列データベースシステムでハッシュ分割が採用されている．ハッシュ分割を用いたグルーピング処理の並列化を考えると，グルーピングキーのハッシュ値で複数のバケツに分ける．それぞれのバケツにおいて独立にグルーピングを行うことで，並列にグルーピング処理が行える．並列度

```
SELECT key, sum(val1), sum(val2), sum(val3), sum(val4), sum(val5) FROM table GROUP BY key;
```

図 6　想定する問合せ

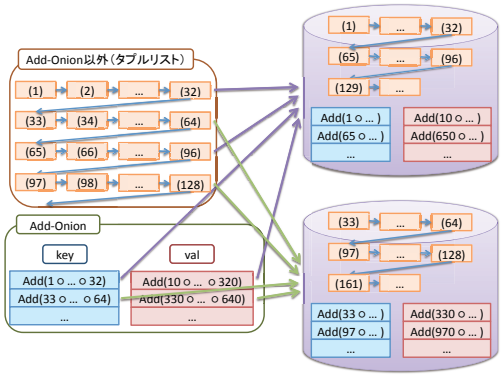

図 7　Round-Robin 分割

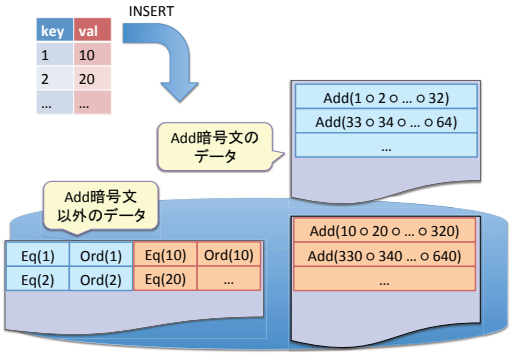

図 8　Decomposed Storage Model

はバケツの数となる．ハッシュ分割ではそれぞれのバケツでユニークなグルーピングユニットが生成されるため結果をまとめる際に追加の計算などは必要ない．異なるデータ分割法を用いた例として Round-Robin 方式が挙げられる．こちらはグルーピングキーの値に関わらず，タプルを平等に複数のバケツに分ける．それぞれのバケツにおいて独立にグルーピングを行い，すべてのバケツでグルーピングが終わったら同一のグルーピングキーを持つグループを結合する．並列度はバケツの数となる．

MONOMI 方式を適用した暗号化データベースシステムにおける総和計算処理を考えると，ADD-Onion において異なるグルーピングキーを持つ値がまとめて 1 つの暗号文に含まれる場合がある．そのためグルーピングキーのハッシュ値で分割すると，複数のバケツに同一の Add-Onion 暗号文をコピーする必要が生じる場合がある．そのため，MONOMI 方式を適用した暗号化データベースシステムにおいてハッシュ分割によるグルーピング処理並列化は適さないと考えられる．

Round-Robin 方式を用いることを考える場合，1 つの平文ブロックに含めるスライスの数を S とすると，Add-Onion 以外のデータは S 個のタプル毎に平等に分割し，それに対応する Add-Onion のデータを割り当てる．$S = 32$ として，2 つのバケツにデータを分割する例を図 **7** に示す．この場合，Add-Onion 以外のデータが含まれるタプルは 32 個毎にバケツに振り分ける．そのタプルに対応する Add－Onion のデータも同じバケツに振り分ける．

Round-Robin 分割を用いると，Add-Onion のコピーを行なうことなくデータを平等に分割することができる．そのため，MONOMI 方式を適用した暗号化データベースシステムには Round-Robin 分割が適していると考えられる．

3.2　Add-Onion ファイル読み込み並列化

MONOMI 方式を適用したデータベースでは，図 8 に示すように Add-Onion のみ列ごとに異なるファイルに分けて保持される．図 6 に示すような問合せの処理では table の Add-Onion 以外が格納されたファイル，val1～val5 の Add-Onion が格納されたファイル，計 6 つのファイルの読み込みが必要になる．val1～val5 の Add-Onion が格納されたファイルは問合せ処理時に読み込みが必要となるが，これは容易に並列化が可能である．すなわち，複数列の加算が必要な問合せの際に，Add-Onion が参照される列数分ファイルの読み込みを並列に行なうことが可能である．

SSD などのフラッシュデバイスを用いる場合，高速化が期待できる．

3.3　総和計算処理並列化

sum(val1)～sum(val5) についてはそれぞれ独立な計算であるため，並列化が可能である．すなわち，複数の総和計算を含む問合せでは個々の総和計算を並列に処理することができる．グルーピング演算を含む総和計算の場合は各グルーピングユニットごとの総和は独立であるためグルーピングユニットの数だけ並列に処理することが可能である．3.1 節で示したデータ分割を用いる場合，バケツ内で各グルーピングユニットの部分和を求め，すべてのバケツで結果が出揃ったら同一グルーピングユニットの部分和を統合する．その様子を図 **9** に示す，グルーピングユニットを統合した後に総和計算を行なうのではなく，部分和を統合することにより総和計算の細粒度での並列化が可能となる．

また，[11] で示されているように，選択演算かグルーピング演算を含む総和計算の際各スライス毎に総乗計算が行われるが，各スライスの計算も独立であるためスライスの数だけ並列に処理可能である．

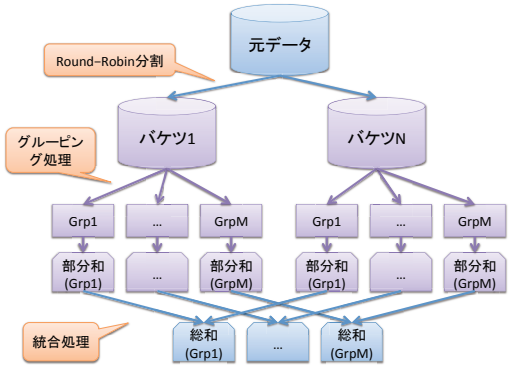

図 9　グルーピング演算を含む総和計算処理の並列化

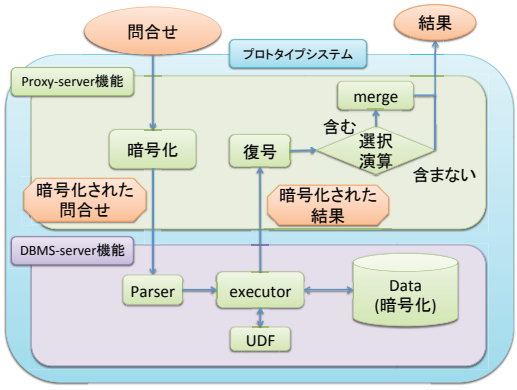

図 10　プロトタイプシステムのモジュール構成

4. プロトタイプの実装

3 節で示した並列化手法について，我々が開発しているマイクロ DBMS 上で実装を行った．C++言語を用いて実装し，Paillier 暗号化ライブラリについては [13] を利用した．Paillier 暗号文の処理については GMP ライブラリ [14] を用い，処理の並列化には OpenMP を用いた．yacc/lex で生成されたパーサのソースコードを除き，システムのソースコードは 2741 行であった．コンパイラは gcc 4.7.4 を用いた．

今回は単純化のためにプロキシサーバと DBMS サーバを単一プログラムとして実装した．構成の概要を図 **10** に示す．図 7 に示すように Add-Onion 以外のデータはタプルリストの形で保持し，Add-Onion のデータは列ごとに配列の形で保持している．バケツは Add-Onion 以外のタプルリストと Add-Onion の配列を保持する構造体として定義した．グルーピングユニットはグルーピングキーとタプルリスト，タプルに対応する Add-Onion 暗号文を表すビット行列を持つ構造体として定義した．

グルーピング演算を含む総和計算処理においては，図 9 に示すように処理を行なう．はじめに Round-Robin 分割を行い，用意したバケツに平等に Add-Onion 以外のタプルリストと Add-Onion の配列を分配する．グルーピング処理は各バケツにおいて独立に行い，グルーピングキー毎にグルーピングユニットが生成される．グルーピング処理終了後，各グルーピングユニットはグルーピング処理によって得られたビット行列を用いて独立に総和計算を行い，中間結果を生成する．各グルーピングユニットにおける総和計算終了後，各バケツが持つグルーピングユニットに対してグルーピングキーを用いて照合を行い，同一グルーピングキーを持つグルーピングユニットの中間結果を統合する．統合処理終了後，得られた暗号文を結果として返す．

5. 関連研究

本研究は効率的かつセキュアなデータ処理基盤の構築を目的としたものである．これに関するものとして，ユーザ（アプリケーション）と DBMS の間にプロキシを設置し，ユーザがプロキシを DBMS とみなして従来通りに問合せを発行しても動作させる「透過的な暗号化」を用いた物がある．先駆け的研究として CryptDB[6][7] があり，それを効率化したものとして MONOMI[8] がある．これらはバックエンドとする DBMS に汎用の RDBMS を用いている点で異なる．製品として NEC の Transparent Data Encryption for PostgreSQL[15] がある．これはバックエンドとする DBMS に PostgreSQL を用いている点で本研究と異なる．また，SecureDBaaS[16] はユーザと DBMS の間にプロキシを設置しない形の暗号化データベースシステムであるが，暗号化データ上でデータベース構造の変更を可能にする等利便性を追求したものになっている．本研究は計算を効率的に行なうことを目的としたものであり，これと異なる．

6. おわりに

本稿では，MONOMI 方式を適用した暗号化データベースシステムにおける，グルーピング演算を含む総和計算問合せ処理の並列化手法を示した．グルーピング演算については，Add-Onion の 1 つの暗号文が含むスライス数に応じた単位で Add-Onion 以外のタプルリストを Round-Robin 分割することで過不足なく平等なデータ量で分割でき，処理の並列化が可能である．総和計算に必要なファイルの読み込みについては，問合せ処理に複数の Add-Onion の参照が必要な時，それぞれ異なるファイルに保存されているため容易に並列読み込みが可能であり，フラッシュデバイスを用いる場合に高速化が期待できる．総和計算処理については問合せ中に複数の総和計算が含まれている時，個々の総和計算は独立であるため，並列に処理できる．また個々の総和計算の中にも並列処理が可能な箇所がある．

今後は本手法の定量的な評価を行う予定である．また SIMD 演算を用いた並列化や GPU を用いた暗号計算の高速化などを検討している．

謝辞 本研究の一部は，JST CREST「ポストペタスケールデータインテンシブサイエンスのためのシステムソフトウェア」，JST CREST「EBD：次世代の年ヨッタバイト処理に向けたエクストリームビッグデータの基盤技術」，JST CREST「広域撮像探査観測のビッグデータ分析による統計計算宇宙物理学」，JST 育成研究「AS262Z02895H」による．

参考文献

[1] SecurityNEXT: ホビーショップの通販サイトからクレカ情報が流出 - セキュリティコードも，，入手先 ⟨http://www.security-next.com/051541⟩（参照 2015-04-20）.
[2] ホビーショップタム・タム：不正アクセスによるお客様情報一部流出に関するお知らせとお詫び，，入手先 ⟨http://www.hs-tamtam.co.jp/apology/info/info$_2$0140825.*pdf*⟩（参照 2015-04-20）.
[3] SecurityNEXT: マーケティング情報サイト運営会社のDB に不正アクセス - 会員情報漏洩の可能性，，入手先 ⟨http://www.security-next.com/049308⟩（参照 2015-04-20）.
[4] ターゲットメディア株式会社：不正アクセスによる閲覧・流出可能性の範囲に関する調査報告，，入手先 ⟨http://www.marke-media.net/dl/tm20140604.pdf⟩（参照 2015-04-20）.
[5] Oracle Corporation: *MySQL 5.7 Reference Manual*, https://dev.mysql.com/doc/refman/5.7/en/encryption-functions.html.
[6] Popa, R. A., Redfield, C. M. S., Zeldovich, N. and Balakrishnan, H.: CryptDB: Protecting Confidentiality with Encrypted Query Processing., *SOSP*, pp. 85–100 (2011).
[7] Popa, R. A., Redfield, C. M. S., Zeldovich, N. and Balakrishnan, H.: CryptDB: processing queries on an encrypted database, *Commun. ACM*, Vol. 55, No. 9, pp. 103–111 (2012).
[8] Tu, S., Kaashoek, M. F., Madden, S. and Zeldovich, N.: Processing Analytical Queries over Encrypted Data., *PVLDB 6(5)*, pp. 289–300 (2013).
[9] 堀尾健太郎，川島英之，建部修見：暗号化データベースシステムにおける効率的な集約処理の評価，情報処理学会研究報告システムソフトウェアとオペレーティング・システム（OS），Vol. 2015-OS-133, No. 19, pp. 1–10 (2015).
[10] Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes., *EUROCRYPT*, pp. 223–238 (1999).
[11] Ge, T. and Zdonik, S. B.: Answering Aggregation Queries in a Secure System Model., *VLDB*, pp. 519–530 (2007).
[12] Copeland, G. P. and Khoshafian, S.: A Decomposition Storage Model, *ACM SIGMOD*, pp. 268–279 (1985).
[13] Bethencourt, J.: Paillier Library, Advanced Crypto Software Collection (online), available from ⟨http://acsc.cs.utexas.edu/libpaillier/⟩ (accessed 2015-04-20).
[14] GNU project: The GNU MP Bignum Library, GNU-project (online), available from ⟨https://gmplib.org/⟩ (accessed 2015-04-20).
[15] NEC: Transparent Data Encryption for PostgreSQL, NEC Corporation (online), available from ⟨http://jpn.nec.com/tdeforpg/⟩ (accessed 2015-07-05).
[16] Ferretti, L., Colajanni, M. and Marchetti, M.: Distributed, Concurrent, and Independent Access to Encrypted Cloud Databases, *IEEE Trans. Parallel Distrib. Syst.*, Vol. 25, No. 2, pp. 437–446 (2014).