Encrypted DBMS のための安全かつ高速な多属性索引の諸検討

金子 静花† 渡辺 知恵美† 柿澤 美穂† 天笠 俊之‡

†お茶の水女子大学〒112 -8610 東京都文区大塚 2-1-1
‡筑波大学〒305 -8573 茨城県つくば市天王台 1-1-1

E-mail: †{kaneko.shizuka, chiemi, g0920510}@is.ocha.ac.jp, ‡amagasa@cs.tsukuba.ac.jp

あらまし クラウドコンピューティング環境でユーザがデータ管理者から機密情報を守るための手段として，データを暗号化して保存しそれに対する問合せを可能にする暗号化データベースシステム(EDBMS)が多く研究されてきた．EDBMS においては、暗号化されたデータに付与される検索用索引を、どのように安全かつ検索性能を落とさず生成するかが問題となる．

本研究では，ブルームフィルタを用いて作成した索引にノイズを付与することによって攻撃者を攪乱させる多属性索引を提案する．まず我々は攻撃者が持つ統計情報とカテゴリ属性の特性を利用した攻撃モデルと，その攻撃に対するプライバシ指標としてビットパターン露呈率を定義した．そして，ビットパターン露呈率を下げるため 4 つのノイズ付加戦略を定義し，各属性の値の分布に基づき適切な戦略を適用することでビットパターンの候補を組み合わせ爆発的に増加させ安全性を担保できる．また，これらの戦略はノイズを加えることで実現しているため，性能の劣化なく安全な索引生成に成功していることを実験によって示した．

キーワード データベースアウトソーシング，クラウドコンピューティング，Database as a Service，ブルームフィルタ，プライバシ保護検索，Encrypted DBMS

1. はじめに

現在，クラウドコンピューティング環境においてデータベースの管理運用を請け負う Database as a Service (DBaaS) が注目を集めている． DBaaS には，現在 S3，EC2 ，SimpleDB1 ，Azure ，Google Apps Engine3 などがある．DBaaS を用いることで，マシン資源の総保有コスト (TCO) を削減することができ，データベースサーバを常時運用することが難しい個人や小規模な企業などによって利用されている．しかしながら，DBaaS 管理者は第三者であるため，管理者が情報を悪用してしまうという内部からの攻撃リスクも考えられる．さらに，様々なセキュリティインシデント [1][2] が報告されており，DBaaS は完全なデータ機密性を保証しているとはいえない．

このようなリスクに対応するため，暗号化したデータと索引を DBaaS に預け，暗号化したまま問合せを施す暗号化データベースシステム (EDBMS) [3]が多く研究されてきた．EDBMS では，元の値を暗号化したものに紐づく検索用の安全な検索用索引を用いて問合せを行っている．既存手法はタプルを列(属性)毎に暗号化した単属性索引を検索用索引として使用している．元の値とその単属性索引との間には 1 対 1 の対応関係がある．つまり，同一な元の値からは同一の単属性索引が生成されることとなるため，このように生成された検索用索引は元リレーションの統計情報に関する大きな手がかりを含んでしまっている．

我々は先行研究にて，ブルームフィルタを用いた DBaaS におけるスキーマ情報と複合的な検索条件を隠ぺいしたプライバシ保護検索手法[4]を提案した．この手法では，属性毎に検索用索引を生成するのではなく，タプル内の複数の属性を 1 つの多属性索引として生成することで安全性を担保しようと試みた．しかしながら素の多属性索引に対してもまた，多属性索引の値から元の値推測の可能性が否定できなかった．

そこで我々は，ブルームフィルタを更にもう一段階適用することで多属性索引の安全性を向上させた ShuffledBF [5] や，多属性索引と ShuffledBF とを組み合わせることで ShuffledBF の性能を向上させた Semi-ShuffledBF [6] を提案し，検索用索引や統計情報を手がかりとしたスキーマ情報への攻撃を防ぐことに成功した．しかしながら， ShuffledBF や Semi-ShuffledBF の性能は，実用化に至るものではなかった．

我々は多属性索引の原点に立ち返り，ノイズを加えることで性能を損なわずに安全性を保障できる EDBMS を考えた．安全性を保障するため，まず敵の攻撃モデルを定義した:「攻撃者は，元テーブルの属性の統計情報を保持しており，検索用索引の特徴と攻撃者の持つ統計情報とを照合して攻撃を行う」．また，基本的な考え方「各属性に対するビットパターンの組が k 個以上含まれる時，ビットパターンが露呈する確率は 1/k 以下になる」をもとに，ビットパターン露呈率を定義した．この露呈率を下げるため，k-1 の同頻度なノイズを偽属性として検索用索引に加えるという基本手法をベースとして，様々なノイズ混入手法を組み合わせて攻撃への攪乱を行い，安全かつ高速な検索用索引を提案した．

本稿の構成は以下の通りである．2 節で基本手法，3 節で先行研究：Semi-ShuffledBF について述べる．4 節で PerturbedBF，5 節で今後の課題について述べる．

2. 基本手法

2.1 暗号化データベース管理システム～ Encrypted Database Management System (EDBMS) ～

EDBMS は，サーバプロバイダを通すデータをすべて暗号化することでサーバプロバイダからもデータの機密情報を守りつつ問合せを行うことのできるデータベース管理システムである．暗号化したデータとその索引をサーバに預けることで，データを暗号化したまま問合せを施すことができる．

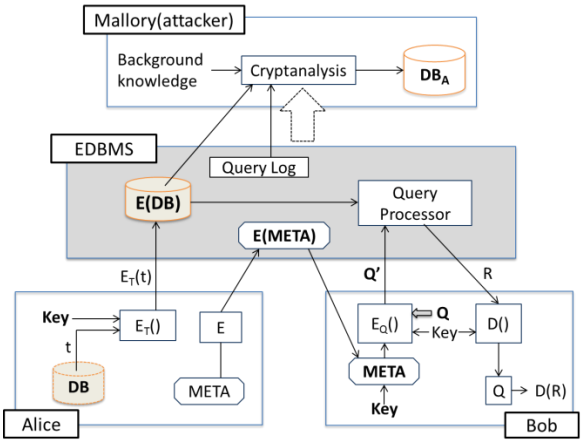

図 1：EDBMS の流れ

一般的な EDBMS のモデルを図 1 に示す．Alice はデータ所有者，Bob はデータベース利用者であり，Alice から閲覧権限を付与されている．Malone は DBaaS におけるデータベース管理者であり，攻撃者とする．

2.1.1データの挿入（保存）

Alice はデータベース DB 内のデータを，クライアント側にある EDBMS で暗号化しサーバに保存する．この時，暗号化したデータとその検索用索引を生成し，双方をサーバに送っている．一般にこの生成はタプル毎に行われる．暗号化されたデータに問合せを行うことはできないため，検索用索引を生成するが，これは検索用索引から元の値を推測することのできない値をとることとする．このようにして，暗号化されたデータとその検索用索引を持つデータベース E(DB)をサーバ側に保存する．また，データのスキーマ情報やどの属性にたいしてどの検索用索引を付与したかというメタデータも暗号化を施された形 E(META)に変換し，サーバに格納される．

2.1.2データの検索（問合せ）

Bob は事前に Alice から暗号化・復号化のための key を受け取り，この key を用いて問合せを行う．まずサーバからメタデータ E(META)を取得し，EDBMS がメタデータを復号化する．復号化されたメタデータ META を元に，EDBMS 内の問合せ変化ツールで Bob が発行した問合せ Q を検索用索引で検索できる形 Q’に変換してサーバへ発行する．サーバでは検索用索引と Q’を用いて問合せが行われ，暗号化されたデータが EDBMS に返される．検索用索引を用いての問合せは，安全ため，一般に false-positive を含んでいる．そのため，EDBMS はこれを復号化し，Q で再び問合せを行った結果を Bob へ返す．

2.1.3攻撃者の立場

Malone は暗号化されたデータベース E(DB)と暗号化されたクエリ Q’のみを閲覧することができ，データベース管理者ゆえに知り得た外部情報も保有していると考えられる．Malone はこれらの情報を組み合わせて元データを推測しようと試みる．

Malone が推測するデータベースを DBM とする．我々は，Malone が DB と一致する DBM を推測することのできないような検索用索引を生成するべきである．次項にて，検索用索引の関連研究を紹介する．

2.2 関連研究

Hacigumus らによる EDBMS フレームワークの提案を先駆けとし，現在も数多くの関連研究が行われている．Hacigumus らの検索用索引生成方法では，元の値が同じであると検索用索引 の値も一致するため，検索用索引の統計から元の値が推測される可能性があった．これに対して Hore らは統計による値の推測を困難にするバケット分割法を提案している[6]. Agrawal ら[1]は，数値属性の順序関係を保存できる検索用索引の生成方法(OPE) を提案している．変換した値の分散が元の値の分散と異なるように変換することによって元の値の推測を防いでおり，属性の比較演算や結合などが可能である．ただ，順序関係が保存されているため一部のデータが漏洩した場合に他の値がなし崩しに判明するという問題があり，Lee ら[7]， Hasan ら[8] などにより改良手法が対案されている．また Mykletun ら[9]，Ge ら[10] による準同型性を持つ暗号化手法を利用した集約演算を可能にする検索用索引生成手法や k-近傍検索に対応した検索用索引生成手法なども提案されている．2011 年には，これらの検索用索引を組み合わせて CryptDB という EDBMS のオープンソースパッケージが Popa[13] ら（MIT）により公開された．

2.3 検索用索引

検索用索引は，EDBMS の問合せで用いられる検索

用の安全な索引である．EDBMS の安全性を確保するためには攻撃者に手がかりを与えない検索用索引を生成することが必要である．本項では，2 種類の検索用索引について説明する．

2.3.1単属性索引

単属性索引は EDBMS で用いられている一般的な検索用索引生成方法で，属性ごとに検索用索引が生成される検索用索引である．属性のデータ型や属性に対して行われる演算によって適用されるアルゴリズムが変化する．CryptDB をはじめとする，2.2 節で述べた研究のほとんどは単属性索引を用いて行われている．

単属性索引を検索用索引にもつ暗号化テーブルのモデルを図 2 に示す．Id, name, age, gender で構成された元テーブルに対し，各属性に対する単属性索引として S_name, S_age, S_gender が格納される．

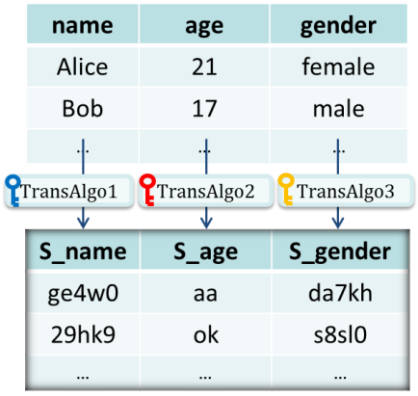

図 2：単属性索引の生成

このように，元の値とその単属性索引との間には 1 対 1 の対応関係がある．つまり，同一な元の値からは同一の単属性索引が生成されることとなるため，単属性索引の値の傾向を分析して元の値の推測を可能にし，元リレーションの統計情報に関する大きな手がかりを含んでしまっている．

2.3.2多属性索引

多属性索引は，複数の属性の索引情報をまとめて 1 つの検索用索引を生成する検索用索引である．複数の属性を 1 つの多属性索引としてまとめてしまうことで，検索用索引の値に対する統計情報の漏えいを防ぐ．また，暗号化クエリ Q’の検索条件文においても，どのような属性に対する問い合わせが行われているのかを隠蔽することができる．また，多属性索引では文書データからのキーワード索引技術を利用することができる．

Boneh らの提案した PAKX[13]は暗号化された文章そのものから，あるキーワードが含まれているかどうかを検索できる公開鍵暗号を用いた手法である．Song ら[14]も同様に，文章からのキーワード検索が可能な多属性索引を提案している．我々もブルームフィルタを用いて多属性索引を生成する方法を提案してきた．

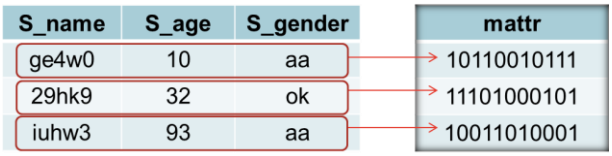

図 3：多属性索引概要

多属性索引を検索用索引にもつ暗号化テーブルのモデルを図 3 に示す．Id, name, age, gender で構成された元テーブルに対し，各属性に対する多属性索引とした mattr が各タプルに対して 1 つ格納される．今回は我々が提案してきたブルームフィルタを用いて生成した多属性索引を提示する．

このように，検索用索引を 1 つにまとめることで，多属性索引の値のどこが元の値のどの属性に対応するのかをわかりにくくすることができる．単属性索引に比べて元の値の推測の可能性は低いが，多属性索引の値のビットパターンを分析することで，元の値の推測が不可能というわけではない．

3. 先行研究：Semi-ShuffledBF

我々は，多属性索引の安全性を担保するため，様々な研究を行ってきた．

ShuffledBF はブルームフィルタを適用して生成された多属性索引をより安全にするため，更にもう一段階ブルームフィルタによるハッシュ関数を適用（シャッフル）して生成された安全な多属性索引である．

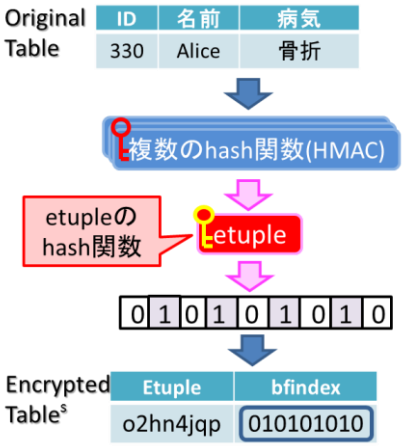

図 4：ShuffledBF の生成方法

ShuffledBF の生成方法を図 4 に示す．元テーブルの属性と語の集合”ID:330”,” 名前:Alice”，“病気:

fracture”に対し，各々複数のハッシュ関数を適用したのち，タプル全体を暗号化したものである etuple をキーにしたハッシュ関数を更に適用（シャッフル）した結果を多属性索引として生成する．
このように，タプル毎に生成される etuple をキーとして 2 段階目のハッシュ関数を適用しているので，安全性は完全に保たれている．

ShuffledBF の問合せ方法を図 5 に示す．索引生成時と同様に，検索したい元データの属性と語の集合”name:Alice”に対し，1 段階目の複数のハッシュ関数を適用した結果 QMAI’を検索条件としてサーバへ送信する．サーバ側では，受け取った QMAI’にタプル全体を暗号化したものである etuple をキーにしたハッシュ関数をタプル毎に適用（シャッフル）して QMAI”とし，多属性索引と一致するかどうかを調べる．

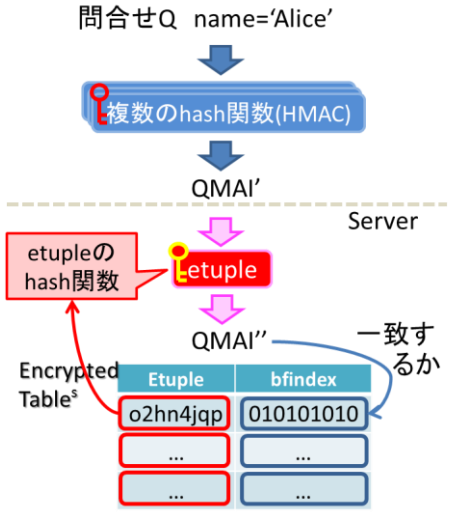

図 5：ShuffledBF の問合せ方法

このように，サーバ側でタプル毎にハッシュ関数を適用しているため，ShuffledBF の性能は低くなってしまっている．

そこで，ShuffledBF と通常の多属性索引とを組み合わせることで性能を改善した Semi-ShuffledBF を提案した．Semi-ShuffledBF の問合せでは，まず通常の多属性索引で検索を行った後一致したタプルに対してのみ etuple での 2 段階目のハッシュ関数適用(シャッフル)を行い，一致するかどうかを調べる．このようにして ShuffledBF の処理性能を改善することに成功したが，Semi-ShuffledBF の性能もまた実用化に至るものではなかった．（図 15 参考）

4. PertuebedBF

多属性索引をシャッフルせずノイズを加えることで，性能を損なわず安全性を保障しようと考えた．敵とその攻撃モデルを定め，攻撃モデルに対して索引のビットパターン露呈率を定義し，この露呈率を低くするようにノイズを加える攪乱戦略を複数提案した．

4.1 Motivation Example

ここでは実際に以下の例を用いて検索用索引に対してどのように攻撃が行われるかを示す．

name	gender	blood_type	country
Alice	female	Null	Japan
Bob	male	O	Japan
Carol	female	AB	China
Dave	male	A	Japan
Eve	female	A	China
Franc	male	O	Korea
Giorgia	male	B	Japan
Hillary	female	B	Japan
Isaac	male	A	Japan

図 6：テーブル例（サンプルテーブル）

多属性索引の値は，複数の属性の検索用索引情報を保有している．よって，多属性索引集合に対して”あるビットパターン集合(BS)”を抽出することで検索用索引の統計情報を得られると考える．図 6 の例では，多属性索引の最初の 4 ビットが前 2 ビットと後 2 ビットに分かれている（ビットパターン集合となっている）ことに注目し，ある属性が 2 値のみを持つと推測できる．

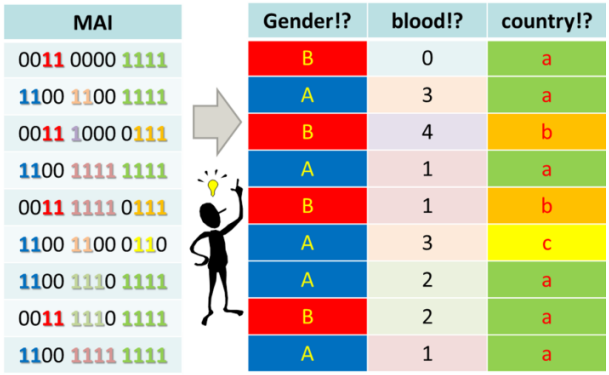

図 7：多属性索引に含まれるビットパターンと推測される属性値[1]

このビットパターン集合より得られた統計情報が敵の背景情報より知り得た統計情報と一致する，あるいは極めて近い値の場合，候補とみなす．先ほどと同様に，敵が「男女比が 4:5 である」もしくは「2 つに分かれる属性は男女だけ」というサンプルテーブルに関する

[1] 図 6 では簡単のため単純なビット配列を用いているが，実際のビット配列はより長く複雑なものとなっている．

情報を外部から取得していた場合，gender 属性が判明してしまう．

リレーション R における多属性索引 M(Γ)中のカテゴリ属性のビットパターン集合(BS)は以下の 2 つの性質を持つ．

(a) 排他性
各索引値 m(t, Γ)は，ビットパターン集合 BS(a)においてたった 1 つの特徴を持つ

(b) 相補性
全索引値 m(t, Γ)は，ビットパターン集合 BS(a)において少なくとも 1 つの特徴を持つ

攻撃者が入手した統計情報が正しければ，検索用索引から必ず 1 つ以上の特徴集合候補が抽出されているので，そのうちの 1 つは元の属性のリレーションから得られた特徴集合である．攻撃者がこれをただ 1 つに定めることができてしまった場合，攻撃が成功し，その属性の属性値が推測できてしまうこととなる．

4.2 プライバシの指標とビットパターン露呈率

本節では，まず攻撃者とその攻撃者が入手可能なデータについて整理し，プライバシの指標について述べ，索引のビットパターン露呈率を定義する．

4.2.1攻撃者が知りうる情報

データベースに保存された内容にアクセス可能なデータベース管理者などであり，サーバに格納されている暗号化されたデータベース E(DB)とサーバへの問合せ Q'を閲覧することができる．本論文では，攻撃者が問合せ Q'を利用することは考えない．

攻撃者は EDBMS から得られるデータの他に，データベースに関する背景知識として，データベースのスキーマ情報と，その一部の統計情報を得ていることを想定する．例えばデータベースのアクセスもとが Web サイトであればサイトを閲覧したり，そのサイトのアカウントを取得してサイト内を観覧することもありうる．サイトの全てのデータを閲覧することができなくても，住居地や性別など，属性のドメインを特定すれば，行政が一般に公開している統計などから大まかな統計情報を入手できる．よって，以下を攻撃者が知りうる情報とする．

- E(DB)　（検索用索引を含む）
- 属性の統計情報
- リレーションの統計情報

4.2.2攻撃モデル

4.1 項にもとづき，多属性索引に対する攻撃モデルのアルゴリズムを以下のように定義する．

```
Algorithm 1: StatisticalAttack_MultiAttributeIndex (attribute)
1.  Input : M(Γ), H^A(attribute)={h^A(v_1),...,h^A(v_n)}
2.    Output : F
3.       Sort(H^A(attribute), descending)
4.       F = FindAllFeatures(M(Γ))
5.       for each h^A(v_i) in H^A(attribute) do
6.          F(vi) = SelectFeatures_withSimilarFrequency(F)
7.       end do

8.       for each f in F(v_0) do
9.          FS = {f}
10.         put FS in F
11.      end do

12.      for i = 1 to n do
13.         for each FS in F do
14.            for each f in F(vi) do
15.               if Satisfy_Exclusiveness(FS,f) then
16.                  put f in FS
17.               else
18.                  remove FS in F
19.               end
20.            end do
21.         end do
22.         if | F | > 100,000  then exit end
23.      end do

24.      for each FS in F do
25.         if Satisfy_Complementary(FS) == false then
26.            remove FS in F
27.         end
28.      end do
```

まず 2 行目で，すべてのビットパターンを探し出して整理する．5,6 行目で属性値と同じ頻度を持つビットパターンを見つけ出し，値のビットパターンの候補とする．それを用いて 12～23 行目でビットパターンの組を増やしながら排他性を満たすかどうかチェックする．最後に 24～28 行目で相補性を満たすかどうかチェックし，満たされる場合最終的に候補とみなされる．

4.2.3ビットパターン暴露率

前項で定義した多属性索引への攻撃モデルに対するビットパターン暴露率を定義する．

- **基本的な考え方**

『索引に各属性に対するビットパターンの組が k 個以上含まれる時，攻撃者は統計情報を用いてこれらの特徴集合候補をさらに絞り込むことはできないため，ビットパターンが露呈する確率は 1/k 以下となる．

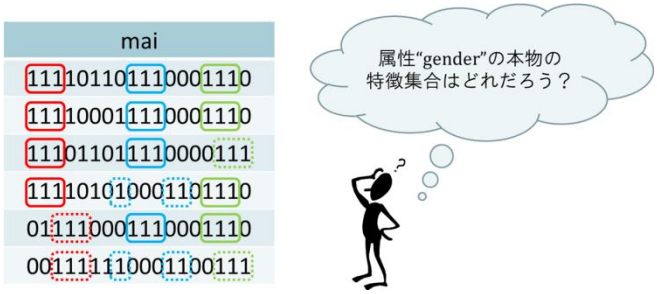

図 8 : gender={male,female}, k=3 とした多属性索引の例

図 8 の例では，敵がこのテーブルに gender という 2 値のみを含む属性が含まれていることを知っているとする．敵はこの情報を用いて多属性索引からビットパターン集合 を探しだし，どのタプルが male か female かを特定したい．しかしながら，この多属性索引は k=3 に設定されているため，この条件に該当するビットパターン集合は 3 候補現れ，敵はこの 3 候補のうちどのビットパターン集合が本物か区別することはできない．

攻撃者がカテゴリ属性の特性と頻度分布を利用したときのビットパターン露呈率を以下に定義した．多属性索引の暴露率の説明のため，単属性索引の暴露率についても言及する．

・単属性索引に対するビットパターン露呈率

(a) 値の暴露率

$$\mathrm{rev}_{\mathrm{value}}(\mathrm{attribute}) = \sum_{v \in attribute} \frac{1}{|Cand_v|}$$

ここで，$|\mathrm{Cand}_v|$は属性値 v における索引の候補数を表す．

(b) レコードの暴露率

$$rev_{record}(attribute) = \sum_{v \in attribute} \frac{1}{|Cand_v|} \times \frac{|v|}{|R|}$$

ここで，$|R|$はリレーション R におけるレコードの数を表し，$|v|$は属性値 v であるレコードの数を表す．

・多属性索引に対するビットパターン露呈率

$$rev_{value}(attribute) = \max_{1 \le i \le n} \left(\frac{1}{|BS(\Delta_i)|} \times \frac{i}{n}\right)$$

$$rev_{record}(attribute) = \max_{1 \le i \le n} \left(\frac{1}{|BS(\Delta_i)|} \times \frac{\sum_{1 \le i \le n} |v_i|}{|M(\Gamma)|}\right),$$

ここで， $\Delta_i = \{v_1, \ldots, v_i\} \subset \Gamma$, and $|v_i| > |v_j|$ if $i < j$.

多属性索引においては，Algorithm1 からもわかるように，値の候補を一つずつ順番に調査しているため，候補順位の高いものは暴露率が高くなるということを考慮しなくてはならない．（4.4.3 項参照）

4.3 ノイズ付加攪乱戦略

前項で定義した暴露率を下げるため，4 つのノイズ付加戦略を定義し，リレーションの各属性の値の分布に基づき適切な戦略を適用する手法を提案した．

- 戦略 1：頻度の類似した属性集合で多属性索引を生成

頻度が類似する属性値を多数多属性索引に含むことで，攻撃者の背景知識（頻度情報）と一致するビットパターン候補を増やすことができる．

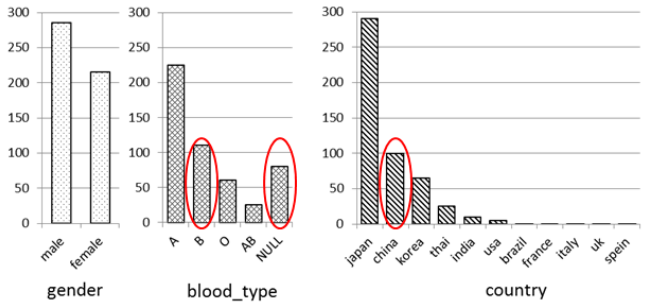

図 9：頻度の類似した属性集合例

- 戦略 2：偽属性の混入

各属性の頻度情報と頻度情報が全く等しい k-1 の偽属性を生成し，多属性索引に含める．この場合，偽属性が k-1 個含まれているので，必ず k 個のビットパターンの組が索引に含まれることとなる．

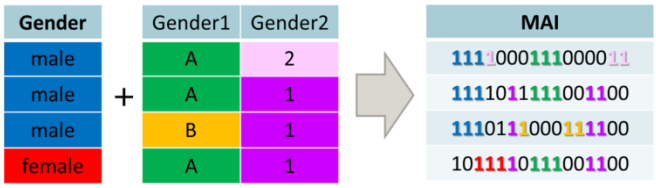

図 10：偽属性の混入例（k=3，2 つの偽属性を混入）

- 戦略 3：誤検出率の調整

多属性索引の誤検出を増加させることで，実際にはそのタプルに存在しない値の特徴が発生するため，ビットパターン候補を増加させることができる．しかしながら誤検出率の増加は問合せコストの増加をも招くため，注意が必要である．

- 戦略 4：属性値の分割

頻度の高い属性値は特定しやすく，さらに類似した属性集合を見つけることが難しい．そこで，そのような属性を 2 つ以上の属性集合に分割し，属性集合の頻度を下げることとする．ただし，この場合複数の属性集合に対して問合せを行わなければならなくなるため，検索コストへの影響も考慮しなければならない．

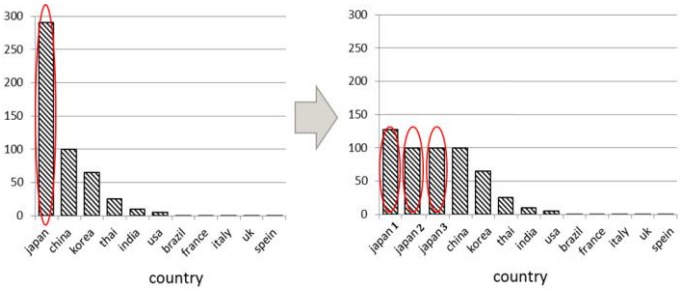

図 11：属性値の分割例

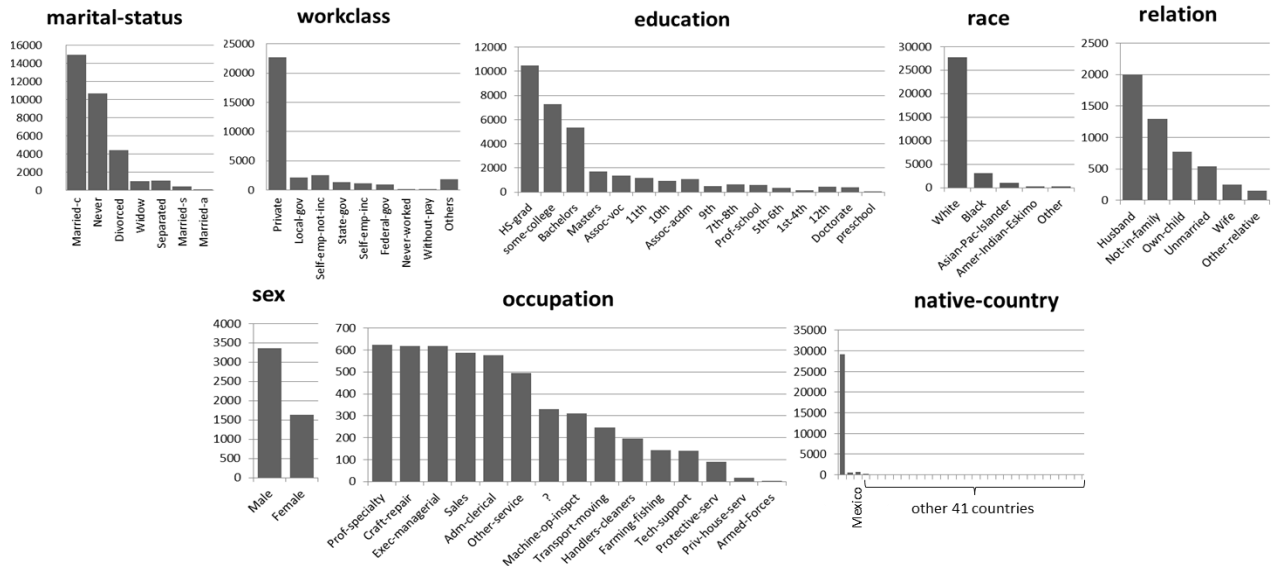

図 12：Adult Dataset におけるカテゴリ属性の頻度分布

4.4 実験

本稿では，前項で述べた各戦略を組み合わせ，実際のデータを用いて攪乱を行った結果を提示する

4.4.1データ定義

UCI[15]より，Adult Dataset を用いる．8 のカテゴリ属性の 32000 レコードで構成されている．属性値とその頻度は図 12 を参照されたし．

4.4.2設定

この Adult Dataset を用いて多属性索引を生成しビットパターン露呈率を測定する．

k = 3 とし，戦略を以下のように適用し各手法とする．

- 手法

手法 1 (M1)：戦略 2 のみを採用

手法 2 (M2)：戦略 1，3 を採用

手法 3 (M3)：戦略 1，2，3 を採用

手法 4 (M4)：戦略 1，2，4 を採用

手法 1～4 の誤検出率とブルームフィルタの長さは以下とする．

	M1	M2	M3	M4
誤検出率	0.001	0.001	0.01	0.001
ブルームフィルタ長	232	668	304	668

4.4.3ビットパターン暴露率算出結果

4.2.4 項で定義したビットパターン暴露率の式に基づき，各暴露率を算出した．SINGLE は単属性索引を表す．

戦略 2 を適用した手法 2～4 は暴露率 1/3 をおさえる結果となった．2 値しか含んでいない属性 Sex においては，戦略 4 の属性の分割を行わないと，候補が k より増えなかった．また，属性によっては 100,000 以上もの候補が現れる結果となった．

(a) 値の暴露率

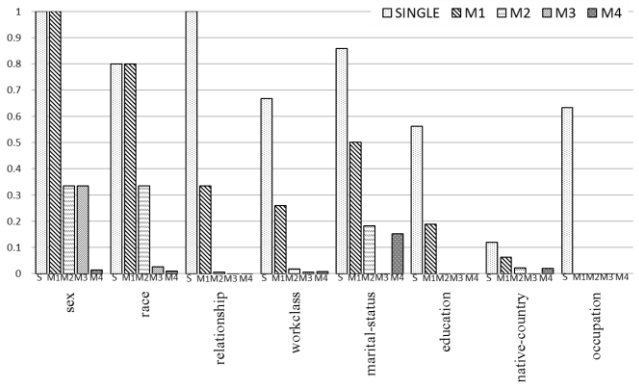

図 13：値の暴露率

(b) レコードの暴露率

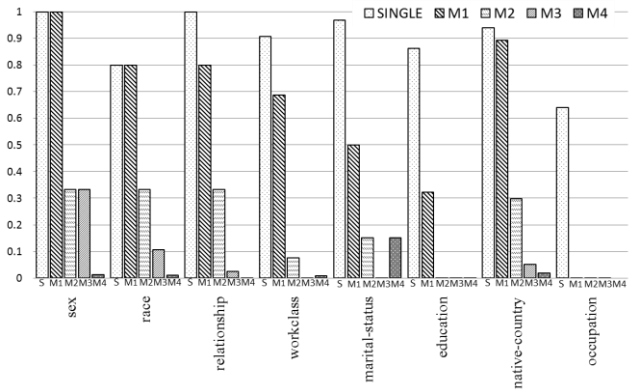

図 14：レコードの暴露率

また,値の暴露率・レコードの暴露率を比較すると,暴露率が極端に異なる属性が存在することがわかる.Relationship の手法 1 や native-country の手法 1・2 などがその代表的な結果である.このように,暴露率を指標として用いる場合,値だけ・レコードだけではなく双方を共に考慮する必要があることがわかった.

4.4.4性能測定結果

図 15 に性能測定結果を示す. NOENC は暗号化なし,SINGLE は単属性索引,PFB は PerturbedBF,SBF は ShuffledBF,SONG は他の多属性索引をそれぞれ表す.

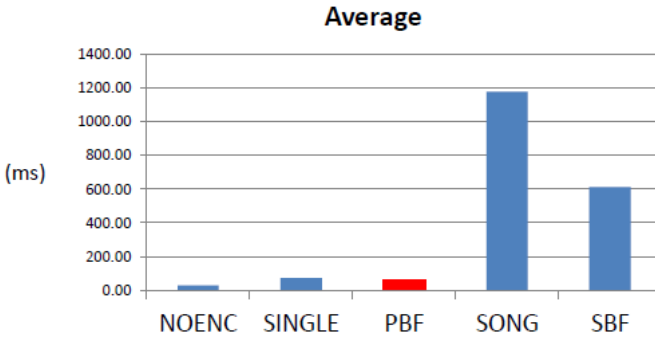

図 15：性能測定結果

性能に関しても問題なく実用化できるという結果を得ることができた.

5. 今後の課題

本稿では,Encrypted DBMS における安全かつ高速な多属性索：PerturbedBF の提案を行った.PerturbedBF は安全かつ高速な多属性索引を実現したが,選択演算にしか対応していない.今後は他演算に対応するため,CryptDB などの他システムとの統合を実現し,実際に EDBMS として機能するシステムを提案することが今後の課題である.

参 考 文 献

[1] ニュース Dropbox Password Failure Puts Users ’ Files at Risk for Four Hours, June2011

[2] ニュース Many companies have lost their data because of large-scale failure at FirstServer, June2012

[3] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra.: “Executing SQL over Encrypted Data in the Database-Service-Provider Model,” In Proceeding of the 2002 ACM SIGMOD International Conference on Management of Data, , pp. 216-227, June 2002.

[4] 渡辺知恵美, 新井裕子: “DaaS におけるスキーマ情報と複合的検索条件を隠蔽したプライバシ保護検索法,” 情報処理学会研究報告, 2008-DBS-146, Vol.2008, pp.163-168

[5] Watanabe C. and Arai Y.: “ Privacy-Preserving Queries for a DAS model using Two-Phase Encrypted Bloomfilter,” In Proceeding of International Conference on Database Systems for Advanced Applications, pp. 491-495, 2009.

[6] Kaneko S., Amagasa T. and Watanabe C.: “ Semi-ShuffledBF: Performance Improvement of a Privacy-Preserving Query Method for a DaaS Model Using aBloom filter,” In PDPTA 2011.

[7] Hore B, Mehrotra S. and Tsudik G.: “ A privacypreserving index for range queries,” In Proceedings of the 30th International Conference on Very Large Data Bases, 2004.

[8] S. Lee, T. Paek, D. Lee, T. Nam and S. Kim: “Chaotic Order Preserving Encryption for Efficien and Secure Queries on Databases,” In IEICE Transactions on Information and Systems E92.D(11), 2207-2217(2009)

[9] Kadhem H. Amagasa T. and Hiroyuki K.: “ A Secure and Efficient Order PreservingEncryption Scheme forRelational Databases,” In Int'l Conf. on Knowledge Management and Information Sharing（KMIS2010）, Valencia,(2010)

[10] E. Mykletun, G. Tsudik: “ Aggregation queries in the database-as-a-service model,” In IFIP WG 11.3 on Data and Application Security (2006)

[11] T. Ge, S. B. Zdonik: “ Answering Aggregation Queries in a Secure System Model,” In Proceedings of VLDB 2007, pp.519-530(2007)

[12] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan: “CryptDB: Protecting Confidentiality with Encrypted Query Processing,” In In Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP 2011)

[13] D . Boneh, G.D. Crescenzo, R. Ostrovsky and G. Persiano: “Public Key Encryption with Keyword Search,” In Proceedings of EUROCRYPT ’04, vol.3027 LNCS, pp.506 522 (2004)

[14] D.X. Song, D.Wagner, and A. Perrig: “Practial techniques for searches on encrypted data,” In In Proceedings of the 21th IEEE Symposium on Security and Privacy (2000)

[15] C. Blake and C. Merz. UCI repository of machine learning databases (1998)