# Paillier 暗号を用いたデータベース演算実装方式における性能解析に関する検討

内藤 華 (指導教員：小口 正人)

## 1 はじめに

第三者のサーバ上で機密情報を含むデータを扱う際，情報漏洩や改ざんを防ぐために暗号化が必要である．準同型暗号という方式を用いると，処理のために復号する必要がなく，データを暗号化したまま演算結果を取り出すことができる．加算または乗算のどちらかのみ計算可能な部分準同型暗号（以下 PHE）や演算回数に制限のない完全準同型暗号などが存在し，行える演算の種類や回数の制限が少ないほど，有用である反面，処理負荷が大きくなることが知られている．本研究で用いる Paillier 暗号 [1] は，Paillier によって提案された公開鍵暗号方式で，加算のみ演算可能な加法準同型性を持つ．したがって，他の方式と比べてデータサイズが小さく低いコストでの演算が可能である．さらに，Chowdhury らの Crypt$\epsilon$[2] では，Paillier 暗号を用いて暗号文同士の加算に加えて乗算も行う手法が提案されいる．本稿では，Crypt$\epsilon$ を基に 7 種類のデータベース演算プリミティブを実装し，演算の手法や処理速度について考察する．さらに，演算の手順の簡素化を図り，Crypt$\epsilon$ とは異なる手法での演算方法の提案を行う．

## 2 関連研究

Crypt$\epsilon$ は，暗号化されたデータに対して 2 種類のサーバを用いてプログラムを実行することで，差分プライバシーを満たした演算結果を出力するシステムである．データ解析者や処理を行う 2 つのサーバにはデータ所有者の個人情報を知られることなく実行することが可能である．

演算を実行する Analytics Server（以下 AS）と，鍵やプライバシーコストの管理を行う Cryptographic Service Provider（以下 CSP）の 2 つのサーバを中心に動作する．CSP はデータ所有者のプライバシーコストの見積もり（以下 $\epsilon^B$）を記録し，Paillier 暗号で秘密鍵と公開鍵の鍵ペアを生成する．各データ所有者は，CSP が生成した公開鍵で自分のデータを暗号化し AS に送信する．暗号化に Linearly Homomorphic Encryption（以下 LHE）を用いると暗号文同士の加算，暗号文と平文の乗算が可能であるが，LHE を発展させた Labeled Homomorphic Encryption（以下 labHE）[3] を用いることで暗号文同士の乗算も可能となる．AS はそれらのデータを集めてプログラムを実行，ノイズを付与した演算結果を CSP に送信する．CSP は，演算によって発生したプライバシーコストが $\epsilon^B$ を超えていないことを確認できると，保管しておいた秘密鍵で復号し結果を出力する．

## 3 実験

### 3.1 処理の手順

図 1 は，属性 Gender について GroupByCount を実行する際の暗号化から演算，復号までの流れを示している．赤い矢印は，暗号化，演算，復号の実行時間

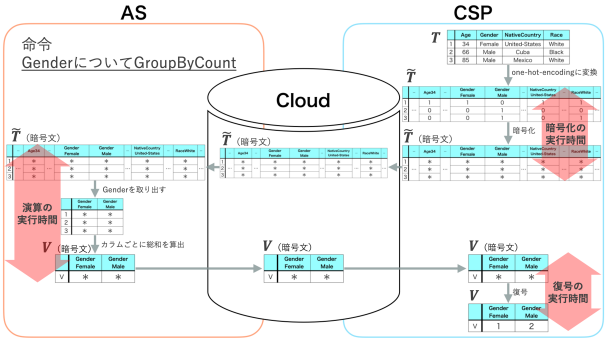



**図 1:** システム構成

**表 1:** 実験環境

| | AS | CSP |
|---|---|---|
| CPU | Intel Core i5 1.6 GHz | Intel Core i5 1.6 GHz |
| OS | macOS 13.1 | macOS 13.1 |
| メモリ | 16 GB | 8 GB |

として計測した箇所を表している．2 台の PC 上に作成した AS と CSP が通信を行い，演算処理前と処理後の暗号データは Google Cloud Storage を介して共有を行う．実験で使用したマシンの性能は表 1 の通りである．CSP はデータを one-hot-encoding に変換後 Paillier 暗号で暗号化し，クラウドに送信する．one-hot-encoding では，例えば $\langle 22, Female\rangle$ というレコードは，$\langle [\underbrace{0,\ldots,0}_{21},1,\underbrace{0,\ldots,0}_{78}],[1,0]\rangle$ のように表現する．

AS は，取り出した暗号データに対して演算を実行した結果を暗号文のままクラウド上に保存，CSP が読み込んで復号する．Adult データセット [4] から属性 $\langle Age, Gender, NativeCountry, Race\rangle$ を抽出し，レコード数 10000 件のテーブルを作成して使用した．また，$\tilde{\boldsymbol{T}}$ に加えて，各レコードが演算に必要であるか否かを 0 と 1 で表すベクトル $\boldsymbol{B}$ を作成し，すべての要素を 1 で初期化する．これは，複数の演算プリミティブを連続して利用する際に有用である．例えば Filter の条件を満たさないレコードを 0 にすると，それ以降の演算には用いられないようにすることができる．データの暗号化は，Paillier 暗号を用いて labHE で行なった．なお，本稿では差分プライバシーのためのノイズ付与とプライバシーコストの計算は行わない．

### 3.2 暗号化手法

labHE による暗号化手法を示す．なお，LHE による暗号化を $Enc_{pk}(m)$，復号を $Dec_{sk}(c)$ と表す．

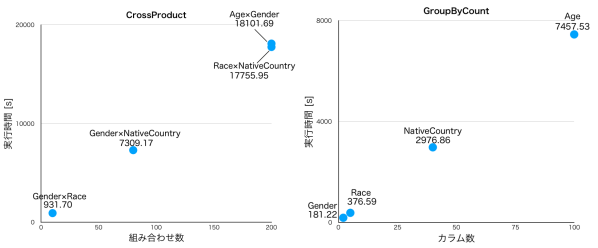
**Setup**$(1^\lambda)$：セキュリティパラメータ $\lambda$ に基づいてマスタ公開鍵（以下 $mpk$）とマスタ秘密鍵（以下 $msk$）の鍵ペアを作成する．

**KeyGen**$(mpk)$：ランダムにユーザごとのシード $\sigma_i$ を生成し，$mpk$ で暗号化することでユーザ秘密鍵（以下 $usk_i$）を作成する．

**labEnc**($mpk$, $usk_i$, $\tau, m$)：ユーザごとのラベル $\tau$ と $usk_i$ を基に，擬似ランダム関数によって $b$ を求める．$a = m - b$，$\beta = Enc_{mpk}(b)$ とすると，$C = (a, \beta)$ が $m$ の暗号文となる．
**labDec**($msk$, $c$)：$C = (a, \beta)$ とすると，$m = a + Dec_{msk}(\beta)$ として復号できる．
**Add**($C_1$, $C_2$)：平文と暗号文を加算する場合と，暗号文同士を加算する場合の 2 通りが考えられる．$C_1$ が平文，$C_2$ が暗号文の場合，$C_1 = a_1$，$C_2 = (a_2, \beta)$ とする．このとき，$C' = (a_1 + a_2, \beta)$．$C_1$，$C_2$ ともに暗号文の場合，$C_1 = (a_1, \beta_1)$，$C_2 = (a_2, \beta_2)$ とする．このとき，$C' = (a_1 + a_2, \beta_1 + \beta_2)$.
**cMult**($c$, $C$)：$C = (a, \beta)$ とすると，$C' = (a \cdot c, \beta \cdot c)$.
**Mult**($C_1$, $C_2$)：$C_1 = (a_1, \beta_1)$，$C_2 = (a_2, \beta_2)$ とすると，$C' = Enc_{mpk}(a_1, a_2) + cMult(\beta_1, a_2) + cMult(\beta_2, a_1)$.
**genLabMult**($C_1$, $C_2$)：$genLabMult$ を用いると，複数回乗算を行うことが可能になる．なお，AS と CSP の間でデータのやりとりが必要となる．$C_1 = (a_1, \beta_1)$，$C_2 = (a_2, \beta_2)$ とする．
AS：乱数 $r$ を生成し，$e' = Mult(c_1, c_2) + Enc_{mpk}(r)$ と $\beta_1$，$\beta_2$ を CSP へ送信する．
CSP：$e'' = Dec_{msk}(e') + Dec_{msk}(\beta_1) + Dec_{msk}(\beta_1)$ を計算する．ランダムなシード $\sigma'$ とユーザごとのラベル $\tau'$ を基に，擬似ランダム関数によって $b'$ を求める．$\bar{a} = e'' - b'$，$\beta' = Enc_{mpk}(b')$ として，$\bar{e} = (\bar{a}, \beta')$ を AS に送信する．
AS：$a' = \bar{a} - r$ とすると，$e = (a', \beta')$ が $C_1$ と $C_2$ の積となる．

### 3.3 基本性能評価結果

本節では，CrossProduct，GroupByCount，CountDistinct の 3 種類の演算の実行時間について考察する．
**CrossProduct**：属性の組み合わせは全部で 6 通りあり，2 つの属性のカラム数の積 $s$ が小さい方から 4 通りの組み合わせについて実行時間を計測した．2 つの属性の直積を求めるためにレコード 1 件につき $genLabMult$ を $s$ 回行う必要があり，ほかの演算と比べて処理時間が非常に長くなった．2 つの属性に含まれる値の組み合わせ総数と実行時間は，図 2(a) の通りほぼ比例している．このデータセットで最も時間のかかる組み合わせは NativeCountry と Age で，値の組み合わせ総数が $40 \times 100 = 4000$ であることから 100 時間以上かかると予想できる．
**GroupByCount**：$\tilde{\boldsymbol{T}}$ のカラムごとにすべての要素を足し合わせることでそれぞれの値の個数を求め，ベクトル $\boldsymbol{V}$ を出力する．それぞれの属性を one-hot-encoding で表現したときのカラム数と GroupByCount の処理時間の関係は図 2(b) の通りである．カラム数に伴って処理時間も増加していることが読み取れる．
**CountDistinct**：属性 $A$ について重複しない値の個数を求める．Crypt$\epsilon$ と異なる手法で演算を行うことにより簡略化を図った．Crypt$\epsilon$ では，ほかの演算手法とは異なり CSP が生成した Garbled circuit[5] を用いて演算を行うが，提案手法では，Garbled circuit は用いず，AS と CSP がデータのやり取りを 1 回のみ行う．図 3 に属性 $Age$ について CountDistinct を行う方法を示す．他の演算と同様，one-hot-encoding で表現したときのカラム数と実行時間はほぼ比例した．GroupByCount の結果 $\boldsymbol{V}$ を利用して算出するため，GroupByCount よりも 1 割程度処理時間が増大した．




**(a)** CrossProduct　　**(b)** GroupByCount

**図 2:** カラム数と実行時間の関係

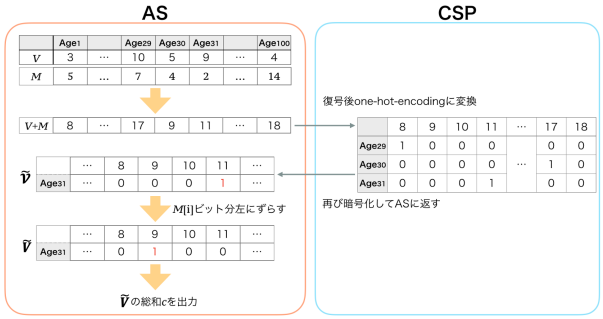



**図 3:** CountDistinct 演算方法

## 4 まとめと今後の課題

加法準同型性を持つ Paillier 暗号を用いてデータベース演算プリミティブを実装，性能評価を行った．Paillier 暗号は PHE の一種であるが暗号化方式として labHE を用いることにより，加算のみならず乗算の演算も可能となった．CrossProduct のように複数の属性の直積を用いる演算や，GroupByCount のように乗算を多用するような演算では，処理時間が増大した．一方で，Count ではベクトル $\boldsymbol{B}$ を利用することで非常に短い時間での処理が可能であった．また，データの暗号化にも比較的長い時間を要しており，改善の余地があると感じた．今後の課題として，直積演算やデータの暗号化の効率的な手法を検討していきたい．

## 参考文献


[1] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques, EUROCRYPT’99, pp. 223–238, Berlin, Heidelberg, 1999. Springer-Verlag.
[2] A. Roy Chowdhury, C. Wang, X. He, A. Machanavajjhala, and S. Jha, “Crypt$\epsilon$: Crypto-Assisted Differential Privacy on Untrusted Servers,” Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 603–619, 2020
[3] M. Barbosa, D. Catalano, and D. Fiore. Labeled homomorphic encryption - scalable and privacy-preserving processing of outsourced data. In ESORICS, 2017.
[4] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science.
[5] Y. Lindell and B. Pinkas, “A Proof of Security of Yao’s Protocol for Two-Party Computation,” Journal of Cryptology, Vol.22, No.2, pp. 161–188, April 2009.