Vol.2019-DPS-178 No.32
Vol.2019-CSEC-84 No.32
2019/3/5

匿名加工を伴う2パーティ秘匿クロス集計の性能比較

片山 源太郎[1,a)]　吉浦 裕[1]

概要：複数組織が持つデータを集積して解析することで，単一組織のみでは実現しえないサービスの実現が期待できる．こうした組織間データ利用において，データのプライバシー保護は必要不可欠である．本稿では特に匿名加工を伴う 2 パーティ秘匿クロス集計を取り上げ，先行研究を紹介したうえで，完全準同型暗号を用いた方式を提案した．先行研究の方式と提案方式との理論比較および実装比較を行った．理論比較においては，計算量のオーダーに差異はなかったが，通信量のオーダーを削減した．実装比較では，総実行時間を LAN 環境において 86.1%，WAN 環境において 77.7%に削減することができた．

キーワード：完全準同型暗号，加法準同型暗号，秘匿積集合，クロス集計

Performance Comparison of Two-Party Secure Anonymized Cross Tabulation Protocol

Keywords: Fully homomorphic ecnryption, Additive homomorphic encryption, Private Set Intersection, Cross Tabulation

1. まえがき

複数組織が持つデータを集積して解析すること（以下組織間データ利用と呼ぶ）は，単一組織のみでは知り得ないデータを利用することができるようになるため，単一組織では実現困難なサービスや製品の実現が期待できる．例えば，性別や住所などの個人の属性情報を持つ組織と商品の販売情報を持つ小売店とが協力し，共通の ID を用いて名寄せしたうえで解析を行い，ある個人へ商品を推薦するサービス [1] がある．

組織同士が互いを信頼して自組織のデータを他方の組織に提供することができれば，組織間データ利用は容易に実現できる．しかし，現実では他方の組織による情報漏洩・不正利用の懸念がある場合やデータの内容（個人情報や企業機密）により第三者へ提供することが適当でない場合があり，組織間データ利用の障害となる．このため，互いを信頼できない組織同士では互いのデータを秘匿して解析することが求められる．さらに，解析の結果を一方の組織に開示すると，解析結果から他方の組織のデータの一部が推定されるおそれがある．これを防止するため，解析と解析結果への匿名加工とをデータ秘匿状態で行い，匿名加工後の解析結果のみを一方または両方の組織に開示する方式が必要となる．

このような匿名加工を伴う 2 パーティ秘匿解析の方式のなかでも，本稿ではクロス集計（2 章で詳述）と呼ばれる解析を行うことを考える．クロス集計はナイーブベイズ法や決定木分析の前段階で使用され，基本的かつ重要な解析手法である．匿名加工を伴う 2 パーティ秘匿クロス集計は千田ら [2] によって，提案されている（3.2.1 節で詳述）．

千田ら [2] の方式では，通信量のオーダーが 2 つの組織がそれぞれ持つデータのレコード数に依存する．例として，顧客情報管理組織が日本人の 1 割程度の属性情報を持ち，小売店は来店した顧客に関する販売情報のみを持つような場合を考える．このとき顧客情報管理組織が持つレコード数と小売店が持つレコード数には大きな差が生じるが，千田ら [2] の方式では，通信量が顧客情報管理組織が持つレコード数に依存するため，通信がクリティカルな設定で総実行時間が大きくなる．

そこで，通信がクリティカルな設定において高速に実行

[1] 電気通信大学, 〒 182-8585 東京都調布市調布ヶ丘 1-5-1 , The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo, 182-8585, Japan
[a)] g.katayama@uec.ac.jp

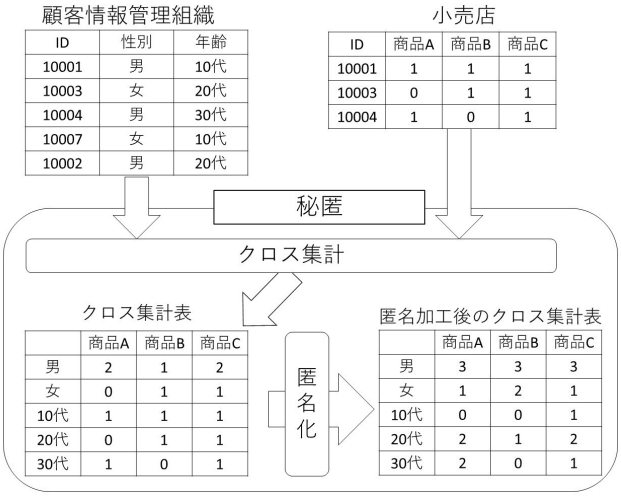

図 1 匿名加工を伴う 2 パーティの秘匿クロス集計

できる匿名加工を伴う 2 パーティ秘匿クロス集計の方式を検討する．秘匿計算の要素技術には Garbled Circuit[3] や秘密分散に基づく秘匿計算 [4]，完全準同型暗号 [5] などがあるが，なかでも完全準同型暗号は Garbled Circuit や秘密分散に基づく秘匿計算とは異なり，乗算やその他の操作の過程において通信が発生しないという特徴を持ち，通信量の削減が期待できる．しかし，完全準同型暗号は暗号文のサイズが大きいことや乗算やその他の操作の計算量が大きいといった問題を持つ．

本稿では完全準同型暗号を用いる方式を提案する．このとき，完全準同型暗号を用いることによる計算量の増加を抑制したうえで，通信量の削減をはかる．最後に 6 章で比較対象方式との理論比較を行い，7 章で実装比較を行う．

2. 匿名加工を伴う 2 パーティ秘匿クロス集計問題

クロス集計とは，データの複数の属性に着目し，着目した属性に関して，それらの属性値の組み合わせに該当する度数を集計することを言う．例えば図 1 のように性別，年齢と商品 A，商品 B，商品 C 購入に着目し，クロス集計すると図 1 に示すクロス集計表のようになる．ここで，図 1 のクロス集計には名寄せの処理も含む．

図 1 のように，2 つの組織が互いのデータを組み合わせてクロス集計表を算出し，匿名加工するまでを秘匿計算し，匿名加工後のクロス集計表を一方または両方の組織に開示するという問題を匿名加工を伴う 2 パーティ秘匿クロス集計問題と呼ぶ．この問題の適切な実装や実行時間は 2 つの組織の持つデータのサイズや形式に依存する．

そこで，本稿ではクロス集計のユースケースとして，山口らのシステム [1] を取り上げる．このシステムでは 2 パーティ秘匿クロス集計を行った後，プライバシーを保護した推薦を行う．プレーヤーやデータテーブル，クロス集計表，匿名加工後のクロス集計表の例を図 1 に示す．図 1 に示すように，2 つの組織として顧客情報管理組織と小売店を考える．また，各組織のパラメータとその大きさを表 1 に示す．

表 1 各組織のパラメータとその大きさ

顧客情報管理組織 (A)	ユーザ数	$10000000(N_A)$
	属性値総計	$57(AT_A)$
小売店 (B)	ユーザ数	$10000(N_B)$
	属性値総計	$10000(AT_B)$

3. 先行研究

3.1 概要

3.1.1 加法準同型暗号および完全準同型暗号

暗号 E と平文 m_1, m_2，演算 $\circ$ に関して $E(m_1), E(m_2)$ から $E(m_1 \circ m_2)$ を計算できるとき，E を準同型暗号と呼ぶ．このとき，$\circ$ が加算であれば加法準同型暗号，乗算であれば乗法準同型暗号，加算・乗算の両方に関して成り立つとき完全準同型暗号と呼ぶ．完全準同型暗号は 2009 年に Gentry ら [5] によって実現された．しかし，文献 [6] にあるように当初は乗算制限をなくすための処理である bootstrapping の計算には 30 分かかることが報告されており，実用性には乏しかった．その後，効率化が進められ，現在ではデータ活用への利用例 [7][8] もある．また，計算コストの観点から現在研究されている多くの完全準同型暗号は LWE 仮定に基づく方式 [9] ではなく，RLWE 仮定に基づく方式 [10] である．

RLWE 仮定に基づく方式では，多項式を暗号化できる．そのため，1 つの暗号文に複数の整数の平文を格納することができる．これはパッキングと呼ばれ，特に CRT パッキング [11] は中国人剰余定理を用いてパッキングを行う．CRT パッキングでは格納した複数の平文に関して完全準同型演算ができる．例えば，整数平文 x_1, x_2, x_3 を 1 つの暗号文 $E(x_1, x_2, x_3)$ に暗号化する．同様に暗号化された $E(y_1, y_2, y_3)$ に関して，$E(x_1 + y_1, x_2 + y_2, x_3 + y_3)$ や $E(x_1 \cdot y_1, x_2 \cdot y_2, x_3 \cdot y_3)$ を計算することができる．また，CRT パッキングされた暗号文は平文の右シフトを実行できるという特徴を持つ．例えば，$E(x_1, x_2, x_3)$ を $E(x_3, x_1, x_2)$ のように右シフトすることができる．

また，その他の完全準同型暗号の周辺技術として，暗号文の法を変換する Modulus switching[12] もある．これは，演算によるノイズ増加の抑制や暗号文の送受信における通信量削減などのために用いられる．完全準同型暗号の中でも，RLWE 仮定に基づく方式の実装例として SEAL[14] があり，CRT パッキングやパッキングした平文の右シフトなどが実装されている．

加法準同型暗号の代表例として Paillier 暗号 [13] があり，暗号文同士を乗算することで平文同士の和の暗号文を計算できる．

3.1.2 秘匿積集合 (PSI) および秘匿等結合

PSI(Private Set Intersection) は複数の組織が互いのデータを秘匿しながらその積集合をいずれかの組織に開示する．多者が参加する PSI[15] や 2 者のみが参加する PSI[7] がある．

秘匿等結合は互いのデータを秘匿しながら共通の識別子で等結合する秘匿計算である．出力する等結合表を平文でいずれかの組織に公開する方式として，可換ハッシュ関数（冪剰余など）を用いる方式 [16] がある．また，出力する等結合表を暗号化したうえでいずれかの組織に開示する方式として，加法準同型暗号を用いる方式 [17] がある．

3.1.3 匿名加工

匿名性の指標として $k-$ 匿名性や $l-$ 多様性，差分プライバシー [18] などがある．匿名加工では，これらの指標を満たすように匿名加工を行う．ここでは数学的に安全性を定式化でき，背景知識に依存しない差分プライバシーを取り上げる．差分プライバシーは対象となる統計量にノイズを重畳する．そのノイズの大きさは統計量の形式だけに依存し，具体的な値には依存しない．統計量の形式を公開すれば，値を秘匿したまま，ノイズの大きさを決めることができる．

3.2 代表的な関連研究

3.2.1 冪剰余の可換性と加法準同型暗号を用いる秘匿クロス集計

千田らは，秘匿等結合してから秘匿クロス集計する方式 [17] を改良し，暗号化状態でクロス集計表を匿名加工し，匿名加工後のクロス集計表を開示する方式を提案した [2]. 文献 [17] では，秘匿等結合に Agrawal ら [16] の秘匿等結合を改良したものを用いている．Agrawal らの秘匿等結合を概説する．組織 A, B が可換ハッシュ関数 f_{ka}, f_{kb} と照合キー id_A, id_B をそれぞれ持つとする．このときに

$$id_A = id_B \Leftrightarrow f_{kb}(f_{ka}(id_A)) = f_{ka}(f_{kb}(id_B)) \qquad (1)$$

が成り立つことを利用し，照合キーそのものを明かさずに照合する．また，式 (2) のように $f_{ka}(id_A)$，$f_{kb}(id_B)$ に id_A, id_B のタプルデータ $tup(id_A), tup(id_B)$ を紐づけることで，組織 A, B の持つデータ同士の等結合が実現できる．

$$tup(id_A) \sim f_{kb}(f_{ka}(id_A)) = f_{ka}(f_{kb}(id_B)) \sim tup(id_B) \quad (2)$$

可換ハッシュ関数としては冪剰余 $f_{ka}(x) = x^{ka} \mod p$ が代表的である．文献 [17] では，紐づけるタプルデータを準同型暗号化し，タプルデータが暗号化された等結合表を秘匿計算する．そして，文献 [2] では暗号化された等結合表に準同型性を用いて匿名加工を加え，いずれかの組織に開示する．

3.2.2 完全準同型暗号を用いた秘匿積集合

Chen ら [7] は完全準同型暗号を用いた 2 パーティ PSI における効率化手法を提案している．この効率化手法は，秘匿クロス集計への応用が期待できる（5 章で詳述）．そこで，文献 [7] について紹介する．まず，基本となるアルゴリズムをアルゴリズム 1 に示す．この基本となるアルゴリズムに Cuckoo hashing[19], Permutation-based hashing[20], CRT パッキング [11]，Modulus switching[12] などを適用することで，計算量や通信量を削減する．

アルゴリズム 1 文献 [7] の基本となるアルゴリズム

Input: A の ID 集合 ID_A，B の ID 集合 ID_B
Output: ID_A と ID_B の積集合
1: B は ID_B の各要素を完全準同型暗号 FHE を用いて暗号化する．これを $c_1, \ldots, c_{N_B}$ とし，A に送信する
2: A は $d_i \leftarrow r_i \prod_{id_A \in ID_A}(c_i - x)$ を準同型性演算する
3: /* r_i は乱数である*/
4: A は $d_1, \ldots, d_{N_B}$ を B に送信する
5: B は $X \cap Y \leftarrow \{y_i : Dec_{FHE}(d_i) = 0\}$ を得る
6: /*Dec_{FHE} は FHE の復号関数とする*/

アルゴリズム 2 insert

Input: 大きさ m のハッシュテーブル T，格納したい値 x，ハッシュ関数の数 h，整数 $i \in \{0, h\}$
1: $x_R \leftarrow x$ の下位 $\log(m)$ ビット，$x_L \leftarrow x$ の下位 $\log(m)$ ビット以外
2: $Loc := h_i(x_L) \oplus x_R$
3: **if** $T[Loc]$ が空 **then**
4: $\quad T[Loc] \leftarrow (x_L, i)$
5: **else**
6: $\quad (x'_L, j) \leftarrow T[Loc]$
7: $\quad T[Loc] \leftarrow (x_L, i)$
8: $\quad$ insert(T, x', h, j')
9: /* x'_L から x' は導出可能とし，$j' \in \{0, h\} \backslash j$ とする */
10: **end if**

Cuckoo hashing，Permutation-based hashing を用いて，2 パーティの各組織が持つ ID 集合をハッシュテーブルに格納する．Cuckoo hashing は複数のハッシュ関数を用いてハッシュテーブルを作成する方式である．Permutaion-based hashing は格納したい値をビット分割し，ハッシュ値と格納したい値の下位ビットとの排他的論理和を格納位置として，残りの格納したい値の上位ビットを格納する方式である．

この 2 種類のハッシュ化を組み合わせ，ある値 x をハッシュテーブルに格納する手順をアルゴリズム 2 に示す．また，各組織の ID 集合全体をハッシュテーブルへ格納する手順をアルゴリズム 3 に示す．Cuckoo hashing により ID 同士の比較回数を削減でき，Permutaion-based hashing により，ハッシュテーブルに格納する要素のビット数を小さくすることができる．格納するデータにより，アルゴリズム 3 は失敗する場合があるが，文献 [7] にあるようにハッシュテーブルの大きさや格納するデータ数を調整することで失敗確率を十分小さくすることができる．詳細については文献 [7] を参照されたい．

Vol.2019-DPS-178 No.32
Vol.2019-CSEC-84 No.32
2019/3/5

アルゴリズム 3 InsertRoutine

Input: A の ID 集合 ID_A，ID_A を格納するハッシュテーブル T_A，ID_B, T_B ついても同様．T_A は大きさ $m \times slot$ の 2 次元配列，T_B は大きさ m の 1 次元配列

Output: ID_A が格納されたハッシュテーブル T_A，ID_B が格納されたハッシュテーブル T_B

1: 組織 A：
2: **for all** $x \in ID_A$ **do**
3: $x_R \leftarrow x$ の下位 $\log(m)$ ビット，$x_L \leftarrow x$ の下位 $\log(m)$ ビット以外
4: 乱数 $i \in \{0, \ldots, h\}$ を用意する
5: $Loc := h_i(x_L) \oplus x_R$
6: $T_A[Loc][j]$ が空であるような j を選ぶ
7: $T_A[Loc][j] \leftarrow (x_L, i)$
8: **end for**
9: 組織 B：
10: **for all** $x \in ID_B$ **do**
11: 乱数 $i \in \{0, \ldots, h\}$ を用意する
12: insert(T_A, x, h, i) を実行する
13: **end for**

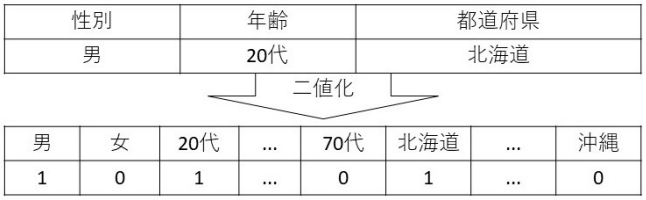

図 2 二値化の例

4. 比較対象の方式

4.1 準備

変数や記法を以下のように定義する．

- 2 つの組織 A, B
- A の ID 集合 ID_A，B の ID 集合 ID_B
- ID_A の要素数 N_A，ID_B の要素数 N_B
- $id_A \in ID_A$ について組織 A が持つタプルデータ $tup_A(id_A)$，組織 B についても同様．タプルデータは二値化されているものとする．例を図 2 に示す．図 2 のように都道府県が北海道であれば，二値化したときには北海道のセルに 1 が立つ．
- 組織 A の持つタプルデータの要素数 AT_A，組織 B の持つタプルデータの要素数 AT_B
- 加法準同型暗号の暗号化関数 AHE
- 加法準同型暗号の平文の法 p
- 完全準同型暗号の暗号化関数 FHE
- 完全準同型暗号の整数平文の法 p'
- FHE の 1 つの暗号文に格納できる整数平文の数 $slot$
- ID_A, ID_B を格納するハッシュテーブル T_A, T_B
- ハッシュテーブルの大きさ m
- 冪剰余の法 p''

4.2 冪剰余の可換性と加法準同型暗号を用いる方式 (EXP-HE)

文献 [2] の詳細な手順をアルゴリズム 4 に記述する．ここで，具体化にあたり以下を行う．(1) 高速化のためタプルデータをまとめて，1 つの加法準同型暗号の暗号文とする（5.2 節にて詳述）．(2) 匿名化は差分プライバシー基準を満たすようにラプラスノイズを重畳する．

また，f_{key} は鍵 key と事前に共有された法 p'' を用いて，$f_{key}(x) = x^{key} \mod p''$ と定義する．アルゴリズム 4 の 11 行目では，$(f_{kb}(f_{ka}(h(id_A)) = f_{ka}(f_{kb}(h(id_B))$ により i と $AHE(tup_A(id_A))$ を照合し，さらに i は id_B と紐づいているため，$tup_B(id_B)$ と $AHE(tup_A(id_A))$ を照合できる．

アルゴリズム 4 は，$ID_A \cap ID_B$ が B に漏洩する問題がある．文献 [17] では，B のタプルデータを暗号化することで，この問題を解決する手法が示されており，それにより $ID_A \cap ID_B$ の漏洩を防ぐことができる．しかし，実行速度はアルゴリズム 4 に劣るため，本稿ではアルゴリズム 4 との性能比較を行う．

その他の安全性については，組織 A に対する ID_B の秘匿は冪剰余の一方向性に帰着する．タプルデータの秘匿は加法準同型暗号の安全性と差分プライバシー基準に帰着する．

5. 提案方式

5.1 提案方式の基本形

提案方式の基本となる組織間クロス集計の手順をアルゴリズム 5 に示す．アルゴリズム 5 は，ID 集合を入力としてクロス集計表を出力するアルゴリズムである．アルゴリズム 5 の 3 行目では，i と id_B の対応関係を記憶しておくことで，16 行目で i から id_B を求められる．7 行目では，完全準同型性を利用して等号判定を行う．等号判定としては，フェルマーの小定理を応用したものがある．これは平文の法が p であるとき，等号判定したい値同士の差を $\phi(p)$ 乗して，1 から引くアルゴリズムである．等号判定の結果は完全準同型暗号の暗号文である．本稿でも今後完全準同型暗号の等号判定にはこの方法を用いる．また，7 行目は完全準同型暗号の準同型演算の乗算，17 行目は加法準同型暗号による準同型演算の加算を行う．

基本形のアルゴリズム 5 では組織 A の ID は秘匿されているが，組織 A のタプルデータは秘匿されていない．そこで，5.2 節で述べる加法準同型暗号へのパッキングを行い，タプルデータを暗号化する．

5.2 破綻確率を考慮した加法準同型暗号へのパッキング

アルゴリズム 5 の 7 行目において，タプルデータ秘匿のために AHE による暗号化を行う．各属性を 1 つずつ暗号化した場合，$N_B AT_A N_A$ 回の暗号化処理が必要となるため，複数の属性をまとめて 1 つの加法準同型暗号の暗号文

アルゴリズム 4 EXP-HE

Input: ID_A, ID_B
Output: 匿名加工されたクロス集計表
1: 組織 A：
2: 鍵 ka と AHE の鍵 (pk, sk) を生成し，pk を B へ送信する
3: 全ての $id_A \in ID_A$ に関して $(f_{ka}(id_A), AHE(tup(id_A))$ を B に送信する
4: 組織 B：
5: 鍵 kb を生成し，全ての $id_B \in ID_B$ に関して $(f_{kb}(h(id_B)), i)$ を A に送信する
6: ここで，i は重複しない適当な乱数
7: $(f_{kb}(f_{ka}(h(id_A)), AHE(tup(id_A)))$ を計算する
8: 組織 A：
9: $(f_{ka}(f_{kb}(h(id_B)), i)$ を B に送信する
10: 組織 B：
11: $(f_{kb}(f_{ka}(h(id_A)) = f_{ka}(f_{kb}(h(id_B))$ となる $tup_B(id_B)$ と $AHE(tup_A(id_A))$ を見つける
12: AT_B の 1 次元配列 CT を用意し，$AHE(0)$ で初期化する
13: **for all** $x \in \{0, \ldots, AT_B\}$ **do in parallel**
14: **if** $tup_B(id_B)[x] = 1$ **then**
15: $CT[x] \leftarrow CT[x] + AHE(tup_A(id_A))$
16: 準同型性を利用し，平文同士の和の暗号文を求める
17: **end if**
18: **end for**
19: CT の全ての要素に準同型性を利用してそれぞれ異なる $\log(p'')$ ビットの乱数を加算して，A に送信する
20: 組織 A：
21: CT の全ての要素を復号し，ラプラスノイズを重畳して B へ送信する．
22: 組織 B：
23: 受信したクロス集計表から 21 行目で加算した乱数を減算し，匿名加工されたクロス集計表を得る

アルゴリズム 5 提案方式の基本となる組織間クロス集計

Input: ID_A, ID_B
Output: クロス集計表 CT
1: 組織 B：
2: FHE の鍵を生成し，公開鍵を A に送信する
3: 全ての $id_B \in ID_B$ に関して $c_i \leftarrow FHE(id_B)$ を A に送信する，i と id_B の対応関係を記憶しておく
4: 組織 A：
5: **for all** $i \in \{1, \ldots, N_B\}$ **do**
6: **for all** $j \in \{1, \ldots, AT_A\}$ **do**
7: $d_{i,j} \leftarrow \sum_{id_A \in ID_A}(id_A = c_i) \times tup_A(id_A)[j]$
8: **end for**
9: **end for**
10: 全ての $d_{i,j}$ を B に送信する
11: 組織 B：
12: $AT_B \times AT_A$ の 2 次元配列 CT を用意し，0 で初期化する
13: **for all** $i \in \{1, \ldots, N_B\}$ **do**
14: **for all** $j \in \{1, \ldots, AT_A\}$ **do**
15: **for all** $k \in \{1, \ldots, AT_B\}$ **do**
16: **if** $tup_B(id_B)[k] = 1$ **then**
17: $CT_{k,j} \leftarrow CT_{k,j} + Dec_{FHE}(d_{i,j})$
18: **end if**
19: **end for**
20: **end for**
21: **end for**

に格納し，アルゴリズム 5 の 6,8 行目の for 文を削減する．

加法準同型暗号の 1 つの暗号文に複数の平文を格納するとき，クロス集計演算やラプラスノイズを重畳する処理および加法準同型暗号の復号後のアンパックが適切に行えるようにパッキングを行わなければならない．具体的な手順を説明する．n 個の l ビットの平文 $t_1, \ldots, t_n$ を式 (3) のように 1 つの平文にする．$||$ はビットによる連接を意味する．

$$t = t_1 || \cdots || t_n \tag{3}$$

このとき $l \times n$ が加法準同型暗号の 1 つの暗号文のビットサイズ $\log(p)$ を超えている場合，$\lceil (l \times n)/\log(p) \rceil$ 個の暗号文とする．また，計算やラプラスノイズの重畳の結果，値が 2^l を超えたとき，平文 t_i での繰り上がりが t_{i+1} に伝搬するため，集計結果が異常な値となる．繰り下がりに関しても t_i, t_{i-1} について同様である．

クロス集計処理が成功確率 Pr となるビット数 l を決める方法について説明する．集計結果の最大値を max とし，$\mathcal{L}(0, \sigma)$ に従う x をラプラスノイズとして重畳することを考える．このとき，ある値 s に関してノイズ $\mathcal{L}(0, \sigma) \sim x > \pm s\sigma$ となる確率は $\exp(-s)$ となる．また，n 個のラプラスノイズ全てが $\pm s\sigma$ を超えない確率は $(1 - \exp(-s))^n$ となる．よって $(1 - \exp(-s))^n > Pr$ が成り立つ s を選び，$\log(s\sigma + max) < l$ となるような l を選べばよい．そのうえで，ラプラスノイズ重畳前に MSB に 1 を立てておくことで繰り下がりが下位の平文に伝搬しない．

5.3 そのほかの工夫

完全準同型暗号を用いた PSI[7] でも用いられていた Cuckoo hashing，Permutaion-based hashing，Modulus switching を提案方式にも応用する．

さらに，アルゴリズム 5 の 7 行目の改良にあたり，5.2 節で説明した方法でタプルデータをパッキングし，AHE で暗号化する．しかし，AHE の暗号文の大きさは FHE の整数平文の法の大きさを上回る．そこで AHE の暗号文 w を $\lfloor \log(p') \rfloor$ ビットに分割し FHE の暗号文に格納する．w の i 番目の分割を w_i とすると，図 3 のように分割された暗号文をずらしながら配置し，等号結果を右シフトしながら等号結果との積をとり，格納結果へ加算していく方法がある．このとき右シフトを $\lceil \log(p)/\lfloor \log(p') \rfloor \rceil$ 回実行する必要がある．

一般に右シフトは定数乗算に比べ，低速である．そこで，図 4 のように格納結果の FHE 暗号文が増えることを許容することで右シフトの回数を削減できる．増加する FHE の暗号文の数を r 個とすると，右シフトの回数は $1/r$ 倍になり，通信量は r 倍になる．通信量が増加することが懸念されるが，Modulus switching による暗号文のサイズ削減により緩和できる．

以上の改良を加えた提案方式の具体的な手順をアルゴリ

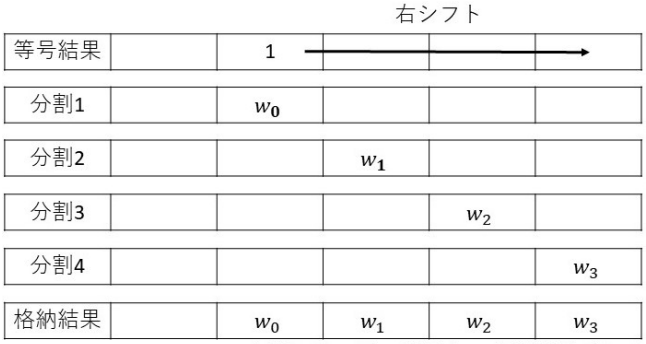

図 3 AHE の FHE への格納

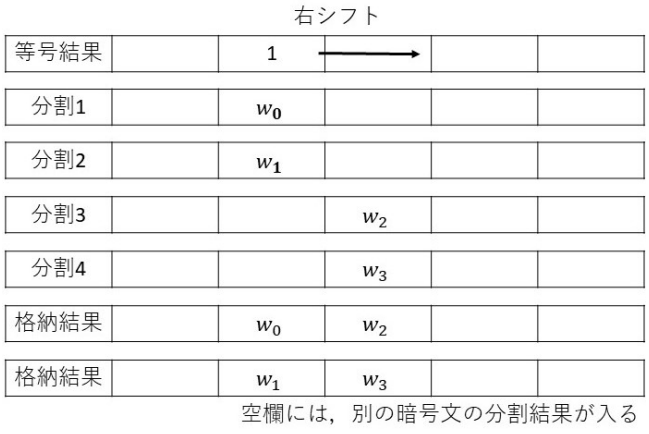

図 4 右シフトの削減

ズム 6 に示す．アルゴリズム 6 の 7 行目において，通信量を最小化するため，1 つの完全準同型暗号の暗号文に $slot$ 個の T_B の要素を入れて暗号化する．そのため，暗号文の数は $\lceil m/slot \rceil$ となる．9〜12 行目において，1 つの暗号文に詰め込まれた T_B の要素を 1 つずつ取り出す．具体的には B から受信した暗号文 c_i に関して $c_i \cdot [0,\ldots,1,\ldots,0]$（取り出したい要素 w の部分にだけ 1 を立てたベクトル）を準同型性を利用して計算すると $FHE([0,\ldots,w,\ldots,0])$ となる．これを Halevi と Shoup の totalSums アルゴリズム [21] を用いて，$FHE([w,\ldots,w,\ldots,w])$ にする．17 行目は，上記で述べた通りに AHE の暗号文を分割したものを完全準同型暗号の暗号文に格納する．

29 行目に関して，$\lceil \log(p'') \rceil$ ビットの乱数を加算しているが，ここで $\lceil \log(p'') \rceil$ ビットの乱数ではなく，組織 B が生成したラプラスノイズを重畳し，組織 A に送信し，組織 A が送られた暗号文を復号するという手順を行うと組織 A に匿名加工後のクロス集計表を開示することができる．このとき，30 行目以降は実行しない．

ID の秘匿は完全準同型暗号の安全性に帰着し，タプルデータの秘匿は Paillier 暗号の安全性と差分プライバシー基準に帰着する．

6. コストの比較

本章では，4 章の比較対象方式と 5 章の提案方式の 2 つ

アルゴリズム 6 FHE-PSI

Input: ID_A, ID_B
Output: 匿名加工後のクロス集計表

1: 組織 B：
2: 　FHE の鍵を生成し，公開鍵を A に送信する
3: 組織 A：
4: 　AHE の鍵を生成し，公開鍵を B に送信する
5: アルゴリズム 3 の InsertRoutine を用いて，ID_A，ID_B をそれぞれハッシュテーブル T_A,T_B に格納する
6: 組織 B：
7: 　T_B の要素を FHE で暗号化し，A へ送信する
8: 組織 A：
9: 　**for all** $i \in \{1,\ldots,m\}$ **do in parallel**
10: 　　$d_i \leftarrow B$ から受信したハッシュテーブルのうち i 番目の要素を取り出す
11: 　　$d_i \leftarrow totalSums(d_i)$
12: 　**end for**
13: 　**for all** $i \in \{1,\ldots,m\}$ **do in parallel**
14: 　　$d_i \leftarrow (d_i = T_B[i])$
15: 　　d_i に $AHE(tup_A(T_A[i]))$ を格納する
16: 　　d_i を B に送信する
17: 　**end for**
18: 組織 B：
19: 　d_i を復号し，$AHE(tup_A(T_A[i]))$ を得る
20: 　大きさ AT_B の 1 次元配列 CT を用意し，$AHE(0)$ で初期化する
21: 　**for all** $x \in \{0,\ldots,AT_B\}$ **do in parallel**
22: 　　**if** $tup_B(id_B)[x] = 1$ **then**
23: 　　$CT[x] \leftarrow CT[x] + AHE(tup_A(id_A))$
24: /* 準同型性を利用し，平文同士の和の暗号文を求める*/
25: 　　**end if**
26: 　**end for**
27: 　CT の全ての要素に準同型性を利用してそれぞれ異なる $\lceil \log(p'') \rceil$ ビットの乱数を加算して，A に送信する
28: 組織 A：
29: 　CT の全ての要素を復号し，ラプラスノイズを重畳して B へ送信する
30: 組織 B：
31: 　受信したクロス集計表から行 27 で加算した乱数を減算し，匿名加工後のクロス集計表を得る

のコストを理論比較する．計算量については剰余や FHE，AHE の暗号化，FHE の準同型演算を計数する．これらは平文の四則演算や AHE の準同型演算に比べて処理が大きいためである．通信量については冪剰余の結果，FHE，AHE の暗号文の送信を計数する．

6.1 EXP-HE

計算量についてはアルゴリズム 4 より，3 行目において $\lceil \log(p'') \rceil$ ビットの冪剰余 N_A 回，加法準同型暗号の暗号化 N_A 回，5 行目において冪剰余 N_B 回，7 行目において冪剰余 N_A 回，21 行目において加法準同型暗号の復号 AT_AAT_B 回である．ゆえに，計算量のオーダーは $\mathcal{O}(N_A + N_B + AT_AAT_B)$ である．

通信量については，アルゴリズム 4 より 3 行目において $\lceil \log(p'') \rceil$ ビットの数 N_A 個，加法準同型暗号の暗号文 N_A

個，5 行目において $\lceil \log(p'') \rceil$ ビットの数 N_B 個，9 行目において $\lceil \log(p'') \rceil$ ビットの数 N_B 個，19 行目において加法準同型暗号の暗号文 AT_AAT_B 個，21 行目において加法準同型暗号の暗号文 AT_AAT_B 個である．ゆえに，通信量のオーダーは $\mathcal{O}(N_A + N_B + AT_AAT_B)$ である．

6.2 FHE-PSI

計算量については，アルゴリズム 6 より，7 行目において完全準同型暗号の暗号化 $\lceil m/slot \rceil$ 回，10 行目において準同型演算の定数乗算を m 回，totalSums では 1 回につき準同型演算の右シフトを $\log(slot)$ 回行うので 11 行目において準同型演算の右シフト $m\log(slot)$ 回，完全準同型暗号の等号判定は $\log(p')$ 回程度の乗算を行うので，14 行目において準同型演算の乗算 $m\log(p')$ 回，15 行目において準同型演算の定数乗算を $m\lceil lAT_A/\log(p) \rceil$ 回，準同型演算の右シフトを $m\lceil lAT_A/\log(p)/r \rceil$ 回，Paillier 暗号の暗号化 N_A 回，19 行目において完全準同型暗号の復号 m 回，29 行目において Paillier 暗号の復号 AT_AAT_B 回である．ゆえに，m を N_B の定数倍とみなせば，計算量のオーダーは $\mathcal{O}(N_A + N_B + AT_AAT_B)$ である．

通信量については，アルゴリズム 6 より，7 行目において完全準同型暗号の暗号文 $\lceil m/s \rceil$ 個，16 行目において完全準同型暗号の暗号文 m 個，27 行目において加法準同型暗号の暗号文 AT_AAT_B 個，29 行目において加法準同型暗号の暗号文 AT_AAT_B 個である．ゆえに，m を N_B の定数倍とみなせば，通信量のオーダーは $\mathcal{O}(N_B + AT_AAT_B)$ である．

6.3 理論比較のまとめ

オーダーによる比較をまとめて表 2 に示す．比較対象方式と提案方式の計算量に差異はないが，通信量は異なる．提案方式の通信量はオーダーから N_A を削減した．これにより，組織 B のレコード数にのみ依存する．

5.3 節で述べたように組織 A, B の役割は，匿名加工後のクロス集計表の開示先には依存しない．そのため，例えば顧客情報管理組織と小売店が提案方式を実行する場合に計算量や通信量が小さくなるようにどちらが組織 A, B の役割をそれぞれ担うのかを設定すればよい．組織 B の役割をレコード数が小さい組織が担うことで，通信量はレコード数が小さい組織のレコード数にのみ依存する．よって，最も支配的な N_A をオーダーから削減した．

表 2 比較対象方式および提案方式の計算量，通信量のオーダー

	計算量	通信量
比較対象方式	$\mathcal{O}(N_A + N_B + AT_AAT_B)$	$\mathcal{O}(N_A + N_B + AT_AAT_B)$
提案方式	$\mathcal{O}(N_A + N_B + AT_AAT_B)$	$\mathcal{O}(N_B + AT_AAT_B)$

7. 実装比較

7.1 実装評価

比較対象方式と提案方式を実装し，比較を行った．C++ を用いて実装し，Boost C++ libraries や Intel TBB を用いてソケット通信や並列計算を実装した．比較対象方式のプログラムは 1000 行程度，提案方式のプログラムは 2000 行程度のプログラムである．加法準同型暗号には Paillier 暗号のライブラリ [22]，完全準同型暗号には文献 [7] で用いられている SEAL_3.0[14] を用いた．

パラメータに関しては，2 章で説明した通り $N_A = 10^7, N_B = 10^4, AT_A = 57, AT_B = 10^4$ とし，それ以外のパラメータについても以下のように設定した．$r = 4, m = 16384, slot = 2048, \log(p) = 1024, p' = 65537, \log(p'') = 1024, l = 32$．また，$ID_A, ID_B$ の要素は $\log(N_A)$ ビット程度であるとする．

l に関しては，0.1-差分プライバシーを満たすラプラスノイズを重畳することを考え，設定した（$\epsilon = 0.1$）．また，想定するクロス集計表の形式では性別，年齢，都道府県に変化があったとき，それぞれ最大 2 万セルが最大 1 だけ変化する．よって $\Delta f = 6 \times 10^4$ を設定した．このとき提案方式の破綻確率を $1/10^6$ とし，5.2 節で述べたように l を算出すると，$\pm 27\sigma$ までのラプラスノイズが加わることを想定すればよいことがわかる．$\pm 27\sigma = \pm 1.62 \times 10^7 \approx \pm 2^{24}$ である．このため l は 25bit 程度に設定すればよいが，2 のべき乗のビット数の方が扱いやすいため，今回は 32bit に設定する．

2 台の計算機を用い，組織 A 側は CPU が Ryzen Threadripper 2950X，メモリ 128GB のサーバであり，組織 B 側は CPU が Core i7-8550U，メモリ 16GB のノート PC である．実験は LAN 環境と WAN を想定した環境でそれぞれ行う．LAN 環境は帯域が 790Mbps(ネットワーク性能測定ツール iperf による計測) である．WAN 環境は文献 [23] を参考に，tc コマンドを用いて帯域を 40Mbps，通信遅延を 40ms に設定した．

比較対象方式と提案方式の実行結果を表 3 に示す．実行結果はそれぞれの方式と環境の計算時間と通信時間，総実行時間である．LAN 環境における実験では提案方式は比較対象方式の 86.1%の総実行時間となった．WAN 環境では提案方式は比較対象方式の 77.7%の総実行時間であった．

7.2 実装評価の考察

理論比較の結果通り，実装比較でも通信がクリティカルとなる WAN 環境で提案方式は優位性が顕著であった．理論比較におけるオーダーでは差異がなかった計算量に関しては，実装比較において，提案方式が比較対象方式に比べて小さかった．一般に完全準同型暗号の乗算などの処理は

表 3 実装実験結果（単位：秒）
全ての条件で 3 回実行し，その平均を取った

	LAN 環境			WAN 環境		
	計算時間	通信時間	総実行時間	計算時間	通信時間	総実行時間
比較対象方式	5989	75	6064	5909	1338	7247
提案方式	5170	53	5223	5160	473	5633
提案/比較対象	86.3%	70.3%	86.1%	87.3%	35.3%	77.7%

加法準同型暗号の暗号化処理や冪剰余の処理に比べ，計算量が大きい．しかし，提案方式における計算量削減により，顕在化しなかったとみられる．現在も完全準同型暗号の高速化 [24] は進められており，今後の進展により，さらに高速化されると考える．

また，計算量のほとんどは組織 A が担うものであり，組織 A がより高性能な計算機を用いることで計算量の増加による実行時間の短縮が期待できる．特に，並列化可能な処理が大半なため，複数台の計算機を用いた分散処理が効果を発揮することが期待できる．

8. 結論

組織間データ利用は，単一組織では実現しえないサービスの実現が期待できる．その中でも組織同士が互いのデータをクロス集計することを考え，匿名加工を伴う 2 パーティ秘匿クロス集計について検討した．先行研究として千田ら [2] の方式を紹介したうえで，その課題点を指摘し，新しい方式を提案した．

本稿では，完全準同型暗号を用いる方式を提案した．完全準同型暗号を用いた PSI[7] での高速化技術を取り入れたうえで，ハッシュテーブル送信時の通信量削減の工夫やラプラスノイズの重畳による破綻確率を考慮した加法準同型暗号へのパッキング，加法準同型暗号の完全準同型暗号への格納などの工夫を行い，通信量や計算量の削減をはかった．これらの工夫により，オーダーから最も支配的な N_A を削減した．また，実装による比較を行った結果，LAN 環境および WAN 環境において，比較対象方式より総実行時間が小さくなった．

今後の課題としては，本稿では山口ら [1] のユースケースを取り上げたが，パラメータを変化させた場合，具体的にどのようなケースでは提案方式が優れているのかを明らかにしたい．

参考文献

[1] 山口高康，寺田雅之：セキュアスムージング手法による組織間プライバシ保護リコメンドシステム，情報処理学会論文誌 Vol.56 No.9 pp.1754-1769 (2015)
[2] 千田浩司，寺田雅之，山口高康ほか：統計的開示制御を考慮したセキュアマッチングプロトコル，Vol.2011-CSEC-52 No.12 (2011)
[3] Yao, A. : Protocols for secure computations, SFCS 1982, pp 160-164, (1982)
[4] Chida, K., Genkin, D., Hamada, K., et al. : Fast Large-Scale Honest-Majority MPC for Malicious Adversaries, CRYPTO 2018, pp.34-64 (2018)
[5] Gentry, C. : Fully Homomorphic Encryption using Ideal Lattices, STOC 2009, pp.169-178 (2009)
[6] 佐久間淳，陸　文傑：準同型暗号を用いた秘密計算技術と実用化に向けた活動，情報処理学会論文誌特集 Vol.59, No.10, pp.898-903 (2018)
[7] Chen, H., Laine, K. and Rindal, P. : Fast Private Set Intersection from Homomorphic Encryption, CCS 2017, (2017)
[8] Lu, W., Kawasaki, S. and Sakuma, J. : Using Fully Homomorphic Encryption for Statistical Analysis of Categorical, Ordinal and Numerical Data, NDSS 2017 (2017)
[9] Regev, O. : On lattices, learning with errors, random linear codes, and cryptography, STOC 2005, pp.84-93 (2005)
[10] Lyubashevsky, V., Peikert, C, and Regev, O. : On Ideal Lattices and Learning with Errors over Rings, EUROCRYPT 2010, pp.1-23 (2010)
[11] Smart, N. and Vercauteren, F. : Fully homomorphic SIMD operations, Designs, codes and cryptography, Vol.71 No.1 pp.5781
[12] Brakerski, Z., Gentry, C. and Vaikuntanathan, V. : (Leveled) fully homomorphic encryption without bootstrapping, ITCS 2012, pp.309-325 (2012)
[13] Paillier, P. : Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, EUROCRYPTO 1999, LNCS 1592, pp.223-238 (1999)
[14] Microsoft Research : Simple Encrypted Arithmetic Library (release 3.0.0), https://github.com/Microsoft/SEAL
[15] Kolesnikov, V., Matania, N., Pinkas, B., et al. : Practical Multi-party Private Set Intersection from Symmetric-Key Techniques, CCS 2017, pp.1257-1272 (2017)
[16] Agrawal, R., Evfimievsski, A. and Srikant, R. : Information sharing across private databases, ACM SIGMOD 2003, pp. 86-97 (2003)
[17] 千田 浩司，寺田　雅之，山口　高康ほか：匿名等結合プロトコルとその応用，SCIS 2011 (2011)
[18] Dwork, C. : Differential privacy, ICALP, Vol. 4052, pp.1-12 (2006)
[19] Pagh, R. and Rodler, F. : Cuckoo hashing, ESA 2001, pp.121-133 (2001)
[20] Arbitman, Y., Naor, M. and Segev, G. : Backyard cuckoo hashing: Constant worst-case operations with a succinct representation, FOCS 2010, pp.787-796 (2010)
[21] Halevi, S. and Shoup, V.: Algorithms in helib, ICC, LNCS, vol. 8616, pp. 554571, (2014)
[22] mshcruz : PaillierLibrarySamples https://github.com/mshcruz/PaillierLibrarySamples
[23] Mohassel, P. and Rindal, P. : ABY3:A Mixed Protocol Framework for Machine Learning, CCS 2018 (2018)
[24] Halevi, S. and Shoup, V. : Faster Homomorphic Linear Transformations in HElib, CRYPTO 2018, pp93-120 (2018)