分散データベースにおける匿名化可能判定プロトコルの効率化

櫻田 潤一† 上土井 陽子‡ 若林 真一‡

†, ‡ 広島市立大学大学院情報科学研究科
731-3194 広島市安佐南区大塚東 3-4-1
†sakurada@lcs.info.hiroshima-cu.ac.jp, ‡{yoko,wakaba}@hiroshima-cu.ac.jp

あらまし 近年，様々な機関で蓄積された情報を共有，統合しデータマイニングに利用することの有効性が検討され始めた．データマイニングは大規模なデータベースを解析することで有益な情報を抽出する技術であるが，データ保有者のプライバシ保護を行うことができないのであればデータの共有は実現されない．本稿ではデータセットが列方向に分割され，２つのサイト上に別々に保存されているとする．そして共通のキーを用いた結合によって再構築できるものとしたとき，再構築されたデータセットが k-匿名性を満たすか判定する要素数判定問題を安全かつ効率よく解く手法を提案し，性能に関して実験的に従来手法と比較する．

An Efficient Decision Protocol for Anonymization of Distributed Databases

Jun'ichi SAKURADA† Yoko KAMIDOI‡ Shin'ichi WAKABAYASHI‡

†,‡Graduate School of Information Sciences, Hiroshima City University
3-4-1 Otuka-higashi, Asaminami-ku, Hiroshima, 731-3194, Japan
†sakurada@lcs.info.hiroshima-cu.ac.jp , ‡{yoko,wakaba}@hiroshima-cu.ac.jp

Abstract In this paper, we focus on the distributed k-anonymization problem and its solutions to integrate databases distributed and managed by multiple agencies with protecting the privacy. In addition, we turn our attention to a previous distributed framework of k-anonymization based on a protocol that solves a decision problem for cardinarity of an intersection between two sets. In this paper, we propose a protocol that computes the set intersection cardinarity efficiently by reducing the number of encryption used to protect the privacy.

1 はじめに

分散 k-匿名化問題に対する従来研究 [1] では分散フレームワークが提案されている．このフレームワークでは，各サイトで局所的に k-匿名部分データを作成し，２つのサイトの局所的な k-匿名化データに関して結合後にも k-匿名化データとなるかを判定する．この匿名化可能判定では，２つの集合間の積集合の要素数に関する判定問題を機密性を保ちながら解くプロトコル $SSI_t(SecureSetIntersection)$[1] を用いる．$SSI_t$ プロトコルでは，一方のサイトに保存されているデータの要素数と同じ回数，値を暗号化して他方に送り，データを受け取った側で必要なものだけを使用して２つの集合の積集合の要素数に関する判定問題を機密性を保ちながら判定している．しかし，それでは使われないデータも暗号化しているので暗号化にかかる計算コストが無駄になっている．またデータの要素数が増えることで計算コストも増大してしまう．

本稿では 2 つの集合の積集合の要素数判定に用いる暗号化回数を抑え，機密性を保ちながら結合後も k-匿名性を保持するデータセットになるかを高速に判定するプロトコルを提案する．また，1 回の要素数判定にかかる計算時間を従来プロトコルと比較し考察する．

2 分散 k-匿名化

2.1 k-匿名化

本稿では，データセットはレコードの集合とし，各レコードは複数の属性の値の組合せで表現されているとする．k-匿名化とは元のデータセットを k-匿名性を満たすデータセットに変換することでプライバシ保護を行うものである．ここで k-匿名性を満たすとは，データセット内の各レコードと同じ値の組合せを持つレコードが少なくとも $(k-1)$ 個存在することとする．データを匿名化するため，特定の値を一般化する．一般化の規則は入力として各属性に対して値一般化階層の形式で与えられている．例えば図 1 のように値一般化階層が設定されている場合，福岡，大分出身を九州出身と一般化することができる．

2.2 分散 k-匿名化フレームワーク

文献 [1] で提案されている分散フレームワークを示す．表 1 のように別々のサイトでデータセットが保存されていて，この 2 つのデータセットを 3-匿名性を保つよう結合する場合を考察する．まず属性 {勤務地} を持つデータセットは図 1 の値一般化階層より 1 段階一般化する．また属性 {年齢，給与} を持つデータセットも図 3, 図 4 から値を一般化する．しかし，給与を 1 段階あげただけでは結合後に 3-匿名性を保てない．よって属性 {給与} に関してさらに値を一般化する．それにより，結合後も 3-匿名性を持つデータセット表 2 を作成することができる．また，分散管理されているデータの局所的 k-匿名化は文献 [3] に記されている Datafly で行う．局所的な匿名化後の同じ値のレコードの集合において積集合の要素数を調べることで結合後に匿名性を満たすかどうか判定する．例えば，勤務地が中国地方となる ID の集合を $D_1 = 1, 3, 4, 5, 8, 10$ とし，年齢，給与が 25-29,16k-25k となる ID の集合を $D_2 = 1, 4, 8$ としたとき，$|D_1 \cap D_2| = 3 \geqq k$ を判定する．

分散 k-匿名化フレームワークの流れを図 1 に示す．ここで要素数判定が SSI_t プロトコルで行われており，2 つのデータセットを結合可能と判定するまでに最低でもデータセットの要素数 m と同じ回数の要素数判定を行う必要がある．

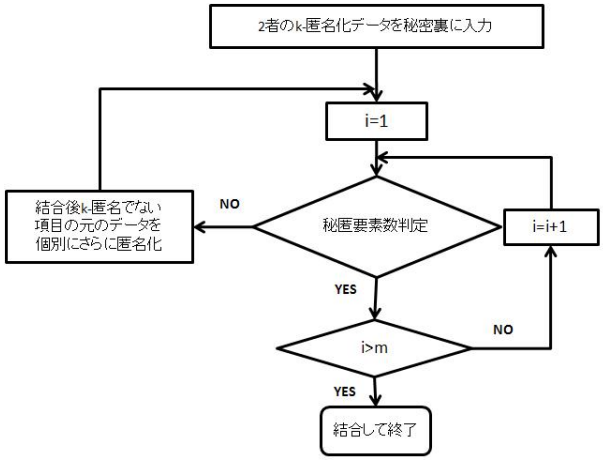

図 1: 分散 k-匿名化フレームワーク

表 1: 元のデータセット

ID	勤務地
1	広島
2	東京
3	島根
4	岡山
5	山口
6	青森
7	秋田
8	広島
9	埼玉
10	鳥取
11	千葉
12	岩手

ID	年齢	給与
1	26	210,000
2	33	360,000
3	46	310,000
4	28	250,000
5	49	330,000
6	41	290,000
7	43	300,000
8	26	220,000
9	34	370,000
10	48	300,000
11	34	370,000
12	41	320,000

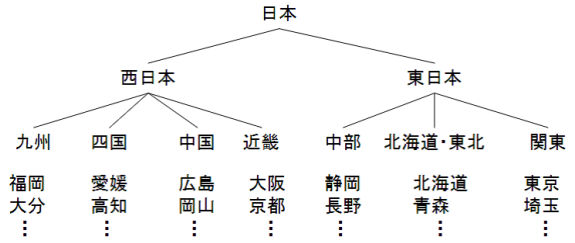

図 2: 勤務地の値一般化階層

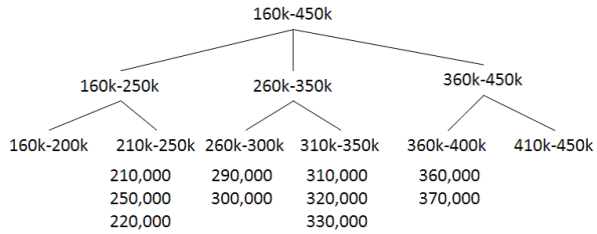

図 3: 給与の値一般化階層

3 $SSI_t(SecureSetIntersection)$プロトコル

SSI_t プロトコルは任意の準同型確率的公開鍵暗号システムを用いたプロトコルで，一方のパーティから入力の一部としてランダムビットを受け取り，もう一方に別のランダムビットを返す積集合演算の大きさを計算するための安全な 2 パーティプロトコルである．D_1, D_2 は当事者 P1,P2 それぞれに関する（整数値の）集合とする．M は 2 つの集合の要素が存在する値域の最大値とする．このとき，集合 D_1, D_2 に対応する以下のビット文字列表現を s_1, s_2 とする．$j = 1, 2$ において, $|s_j| = M$(ビット文字列の長さ)， $i \in D_j$ ならば $s_j[i] = 1$, そうでないなら $s_j[i] = 0$ が成り立つ．

SSI_t プロトコルの入出力を $SSI_t((s_1, t)(s_2, t, b_2)) \rightarrow (b_1, \perp)$ と定義する．概要を図 5 に示す．ここで，t は公に知られている閾値，b_2 はランダムビット（一般性を失うことなく b_2 が P2 から入力の一部として提供されていると仮定）とする．

・(s_1, t) :P1 の入力
・(s_2, t, b_2):P2 の入力

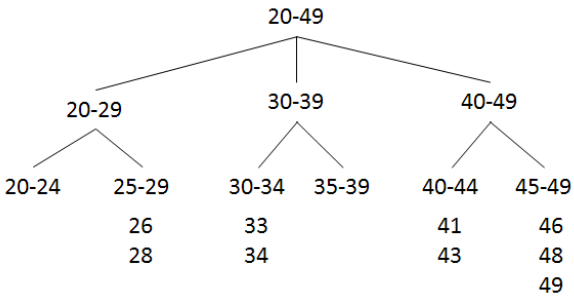

図 4: 年齢の値一般化階層

表 2: 3-匿名データセット

ID	勤務地	年齢	給与
1	中国	25-29	16k-25k
2	関東	30-34	36k-45k
3	中国	45-49	26k-35k
4	中国	25-29	16k-25k
5	中国	45-49	26k-35k
6	北海道・東北	40-44	26k-35k
7	北海道・東北	40-44	26k-35k
8	中国	25-29	16k-25k
9	関東	30-34	36k-45k
10	中国	45-49	26k-35k
11	関東	30-34	36k-45k
12	北海道・東北	40-44	26k-35k

SSI_t の出力を以下に示す.
P1 側:ビット b_1 を出力 ($|D_1 \cap D_2| < t$, ならば $b_1 \oplus b_2 = 1$; そうでないなら $b_1 \oplus b_2 = 0$)
P2 側:記号 $\perp$ で示される空の文字列を出力（出力なしの意味）

P1 と P2 が互いに $b1$ と $b2$ を教えあうことで $b_1 \oplus b_2 = 1$ なら $|D_1 \cap D_2| < t$， $b_1 \oplus b_2 = 0$ なら $|D_1 \cap D_2| > t$ を知る．
例：D_1={1,4,8,10}, D_2={1,2,6,8}, $t = 3, b = 0$ のとき SSI_tの出力は以下となる．
・$D_1 \cap D_2$={1,8}, $|D_1 \cap D_2| = 2 < t$, ・M=10
・P1 の入力:s_1=1001000101
・P2 の入力:s_2=1100010100
・P1 の出力:b_1=1

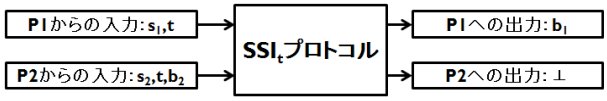

図 5: SSI_t プロトコルの概要

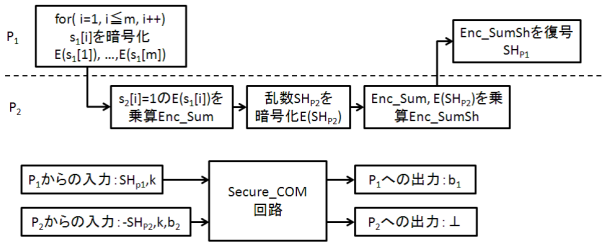

図 6: 2 者間でのプロトコルとしての SSI_t の概要

3.1 SSI_t プロトコルに用いる確率的公開鍵暗号化関数の特性

SSI_t では以下のような確率的公開鍵暗号化関数を用いる．$E : R \times X \to Y$ は確率的公開鍵暗号化方式とし，R, X と Y は整数からなる有限の値域とする．そして $D : Y \to X$ は秘密鍵復号関数とし，$\forall(r, x) \in R \times X$ において $D(E(r, x)) = x$ となる．さらにこれらの暗号化関数は次の性質を満たす．

・暗号化関数は二番目のパラメータに関して単射である．

例:$\forall(r_1, x_1), (r_2, x_2) \in R \times X$,

$E(r_1, x_1) = E(r_2, x_2) \Rightarrow x_1 = x_2$

・暗号化関数は意味的な機密性をもつ．

例:$\forall(r_1, x_1), (r_2, x_2) \in R \times X$,

$\Pi\ (E(r_1, x_2), E(r_2, x_2)) = E(r_3, x_1 + x_2)$

ここで r_3 は多項式時間で r_1, r_2, x_1, x_2 から計算することができる．(Πは２つの暗号化された値を“ 加算 ”するための関数である．ここでは２つの暗号の内積を表す)

・暗号化関数の値域と定義域は適切である．

3.2 ２者での SSI_t プロトコルの概要

文献 [1] に示されている SSI_t プロトコルの手続きを以下に示す．また，その概要を図 6 に示す．

Algorithm SSI_t Protocol

Require: $(s_1, t), (s_2, t, t_2), |s_1| = |s_2| = M$

1:Party 1 (P1):

(a). generate a public and private key pair E, D for the probabilistic key homomorphic encryption system.

(b). encrypt s_1:

$x_i \leftarrow E(r, s_1[i])$,for $i = 1, \cdots, M$; r is randomly chosen for each $s_1[i]$

(c). send $x_1, \cdots, x_M$ in order and E to P2

2:Party 2 (P2):

(a). compute $Enc\_Sum \leftarrow \Pi_{\forall i \wedge s_2[i]=1} x_i$

(b). compute $Enc\_SumSh \leftarrow \Pi\ (Enc\_Sum, E(r\ ', SH_{P2}))$,where $r\ '$and SH_{P2} are randomly chosen

(c). send Enc_SumSh to P1

3:Party 1:

(a). compute $SH_{P1} \leftarrow D(Enc\_SumSh)$

4:Party 1 & Party 2

(a). $Secure\_COM((SH_{P1}, t), (SH_{P2}^{-1}, t, b_2))$

(b). P1 $\leftarrow b_1$ & P2 $\leftarrow \perp$

ステップ 4(a) の $Secure\_COM$ は，2 者からの入力 SH_{P1}, SH_{P2}^{-1} の加算結果を閾値 t と比較し，$SH_{P1} + SH_{P2}^{-1} \geqq t$ ならば $b_1 \oplus b_2 = 1$，そうでないなら $b_1 \oplus b_2 = 0$ を満たす b_1 を P1 に出力する秘匿加算比較回路である．よって，その入力および加算・比較している途中の計算結果は両者に共有されない．

3.3 SSI_t プロトコルの実現に用いる暗号化関数

ElGamal 暗号を SSI_t に使うための暗号化関数が満たすべき特性に反さないような形に以下のように関数の変換を行う．ElGamal 暗号方式で用いられる，素数を p，原始元を g，x を秘密鍵，y を公開鍵 $(y = g^x \bmod p)$ とする [2].

$E(r, M) = (g^r \bmod p, 2^M y^r \bmod p)$

(原型のままでは $E(s_1, t_1) \times E(s_2, t_2) = (g^{s_1+s_2}, t_1 t_2 y^{s_1+s_2}) = E(s_1 + s_2, t_1 t_2)$ となり $t_1 t_2$ の部分が加法となっていないので 2 を底と

し指数を平文 M とする．)
上記より，変更した暗号化に対応するために復号では 2 の対数を取る．
よって $D(a,b) = \log(ba^{p-1-x} \bmod p)$
$= \log(2^M y^r a^{p-1-x} \bmod p)$
$= M$

4 提案プロトコル

SSI_t プロトコルでは 1.(b) でパーティ P1 は $i = 1, ..., M$ に対する $s_1[i]$ で表現されるすべてのビットを各々に暗号化しているので M 回の暗号化が行われている．

本稿で提案する手法ではビット単位ではなくビット文字列単位で暗号化することで暗号化する回数を削減する．

4.1 動機

パーティ P1, P2 それぞれが集合 D_1 と D_2 を個別にもっているときに，D_1, D_2 に対応するビット文字列 s_1, s_2 を暗号化した状態のままで P1,P2 間で操作することにより積集合 $D_1 \cap D_2$ の要素数を求めることが可能か検討した．具体的には，元のビット文字列，もしくはビット文字列の 1 の回数が相手に知られることなく，2 つのビット文字列 x, y の AND 演算結果において 1，つまり $x[i] = y[i] = 1$ となるインデックスの個数を求めることが可能か検証した．上記が可能ならば，SSI_t プロトコルでインデックスの値域の大きさ M と等しい回数行っていた暗号化を定数回で抑えることができる．

そこで，我々はビット文字列 x, y の暗号文 $E(x)$, $E(y)$ に対して演算を適用し，その結果を復元して利用することで AND 演算結果を得ることができるのではないかと考えた．加法の準同型な暗号化関数が SSI_t プロトコルで用いられているので，暗号文 $E(x), E(y)$ に対する加算演算結果から AND 演算結果の導出に焦点を置き考察した．考察の結果，元のビット文字列 x, y 暗号文 $E(x), E(y)$ に加え，x, y の逆順のビット文字列 x', y' の暗号文 $E(x'), E(y')$ も同時に操作することにより，加算演算結果から AND 演算結果を求めることができることがわかった．この加算演算結果から AND 演算結果の導出方法を 4.2 節にて提案する．また，4.3 節では各当事者 P1，P2 が機密性を保つため，ビット文字列 s_1, s_2 からどのようなビット文字列 x, y を作成する必要があるかについて考察する．

4.2 AND 演算結果を求める方法の提案

4.2.1 ビット予測の基礎定理

$x[i]$ と $y[i](i = 1, ..., n)$ を要素とする長さ n のビット文字列 x, y を加算した結果を格納する長さ $n+1$ のビット文字列を a とし，各要素を $a[i](i = 1, ..., n+1)$ とする．このとき，a の各要素は加算の定義より以下を満たす．

・$a[i] = x[i] \oplus y[i] \oplus c_1[i](i = 1, ..., n)$
・$c_1[i+1] = (x[i] \wedge y[i]) \vee ((x[i] \oplus y[i]) \wedge c_1[i])$ $(i = 1, ..., n)$
・$c_1[1] = 0$
・$a[n+1] = c_1[n+1]$

またビット列 x, y の逆順ビット列を x', y' とし，$x'[i] = x[n+1-i]$, $y'[i] = y[n+1-i](i = 1, ..., n)$ を要素とする長さ n のビット列とする．ビット列 x', y' を加算した結果を 長さ $n+1$ のビット列 b に格納するとし，各要素を $b[i](i = 1, ..., n+1)$ とする．b の各要素は加算の定義より以下を満たす．

・$b[i] = x[n+1-i] \oplus y[n+1-i] \oplus c_2[i](i = 1, ..., n)$
・$c_2[i+1] = (x[n+1-i] \wedge y[n+1-i]) \vee ((x[n+1-i] \oplus y[n+1-i]) \wedge c_2[i])(i = 1, ..., n)$
・$c_2[1] = 0$
・$b[n+1] = c_2[n+1]$

また，表 3，表 4 のように $(a, c_1), (b, c_2)$ の要素からそれぞれ x, y の要素がどのようなペアだったか予測することができる．

だが，各々の予測では 2 つの予測が成り立っているので x, y を正確に予測することができない．しかし 2 つの表を表 5 のように結合することで予測を $(1,1), (0,0), (*,*)$ の 3 つのうちどれかに絞ることができる．また表 5 より以下の式を求めることができる．

$(b[n+1-i] \wedge c_2[n+1-i] \wedge c_1[i+1]) \vee$

表 3: $(a)a[i], c_1[i+1]$ による $x[i], y[i]$ ペア予測関数

入力		出力
$a[i]$	$c_1[i+1]$	$(x[i], y[i])$
0	0	(0,0)
0	1	(*,*),(1,1)
1	0	(0,0),(*,*)
1	1	(1,1)

表 4: $(b)b[i]c_2[i]$ による $x[i], y[i]$ ペア予測関数

入力		出力
$b[n+1-i]$	$c_2[n+1-i]$	$(x[i], y[i])$
0	0	(0,0),(1,1)
0	1	(*,*)
1	0	(*,*)
1	1	(0,0),(1,1)

表 5: $x[i], y[i]$ ペアの予測関数

入力				出力	
$a[i]$	$c_1[i+1]$	$b[n+1-i]$	$c_2[n+1-i]$	$(x[i], y[i])$	$(c_1[i], c_2[n+2-i])$
0	0	0	0	(0,0)	(0,0)
0	0	0	1	×	×
0	0	1	0	×	×
0	0	1	1	(0,0)	(0,0)
0	1	0	0	(1,1)	(0,1)
0	1	0	1	(*,*)	(1,1)
0	1	1	0	(*,*)	(1,0)
0	1	1	1	(1,1)	(0,1)
1	0	0	0	(0,0)	(1,0)
1	0	0	1	(*,*)	(0,1)
1	0	1	0	(*,*)	(0,0)
1	0	1	1	(0,0)	(1,0)
1	1	0	0	(1,1)	(1,1)
1	1	0	1	×	×
1	1	1	0	×	×
1	1	1	1	(1,1)	(1,1)

×は禁止入力，(*,*) は (1,0) か (0,1) の組合せでを示す.

$(\bar{b}[n+1-i] \wedge \bar{c_2}[n+1-i] \wedge c_1[i+1]) = 1$ のとき，$x[i] = y[i] = 1$.
$(a[i] \oplus c_1[i+1]) \wedge (b[n+1-i] \oplus c_2[n+1-i]) = 1$ のとき, $x[i] \neq y[i]$.
$(\bar{b}[n+1-i] \oplus c_2[n+1-i]) \wedge \bar{c_1}[i+1] = 1$ のとき, $x[i] = y[i] = 0$.
以上の解析により，以下の定理が導出できる.

定理　長さ n のビット列 x, y の加算結果を格納した長さ $n+1$ のビット列 a と x, y の逆順ビット列の加算結果を格納した長さ $n+1$ のビット列 b において，$1 \leqq i \leqq n$ を満たすある整数 i に対して，$a[i], c_1[i+1], b[n+1-i], c_2[n+1-i]$ が与えられたとき，$x[i], y[i], c_1[i], c_2[n+2-i]$ を表 5 よりあいまいに求めることができる．ここで，c_1 はビット列 x, y の加算時のキャリービット列，C_2 は x, y の逆順ビット列の加算時のキャリービット列とする．　□

4.2.2　手続き AND

表 5 を用いて，$1 \leqq i \leqq n$ の整数 i において $x[i] = y[i]$ となる回数 An_1 を計算する手続き AND を図 7 に示す．手続き AND より An_1 を計算できるので分散 k-匿名化フレームワークで結合後 k-匿名性を保つかを調べることができる．また，$x[i] \neq y[i]$ のときどちらが 0 か 1 かは元の情報を漏えいしないため確定できてはいけない.

例：$x = 1001, y = 1010$ とすると a, b はそれぞれ $a = 10011, b = 01110$ である．$a[5] = 1$ より $c_1[5] = 1$，また $c_2[1] = 0$, $a[4] = 0$, $b[1] = 0$，なので $(x[4], y[4], c_1[4], c_2[2]) = (1, 1, 0, 1)$. 次に $(a[3], c_1[4], b[2], c_2[2]) = (0, 0, 1, 1)$ なので $(x[3], y[3], c_1[3], c_2[3]) = (0, 0, 0, 0)$．次に $(a[2], c_1[3], b[3], c_2[3]) = (1, 0, 1, 0)$ なので $(x[2], y[2],$

```
Input  ビット列 a[1,…,n+1], b[1,…,n+1]
Output 整数 An1
    c1[n+1]=a[n+1]; c2[1]=0; An1=0;
    for ( i=n; i>0; i-- ) {
        a[i], c1[i+1], b[n+1-i], c2[n+1-i] の値から表5より、
        ペア( x[i], y[i] ) の組み合わせを予測;
        加算の定義よりc1[i], c2[n+2-i]を求める;
        if ( ( x[i], y[i] ) == (1,1) )  then An1++;
     }
```

図 7: 手続き AND

$c_1[2], c_2[4]) = (1, 0, 0, 0)$．次に $(a[1], c_1[2], b[4], c_2[4]) = (1, 0, 1, 0)$ なので $(x[1], y[1], c_1[1], c_2[5]) = (1, 0, 0, 0)$．上記より $x = 10**, y = 10**$ のとなる．(ここで $*$ は $x \neq y$ となる $1, 0$ の組み合わせである.)

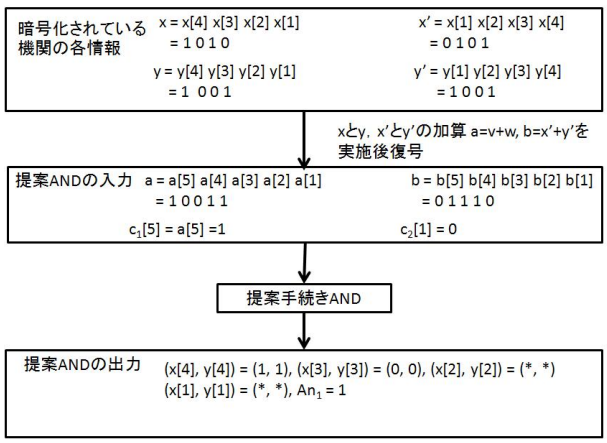

図 8: 加算結果ビット列 a，b による要素数判定の例

4.2.3 正当性の証明

a, b の要素から $x[i]$, $y[i]$ がどのような要素のペアか知ることができることを数学的帰納法によって証明する．上記の式より $x[n-i+1]$, $y[n-i+1]$ の要素のペアは $a[n-i+1]$, $c_1[n-i+2]$, $b[i]$, $c_2[i]$ の値によって求めることができる．そして $x[n-i+1]$, $y[n-i+1]$ の要素のペアより $c_1[n-i+1]$, $c_2[i+1]$ の値を求めることができる．

・基本ステップ

$i = 1$ のとき

$c_1[n+1] = a[n+1]$, $c_2[1] = 0$,$a[n]$, $b[1]$ より，表 5 から $x[n-i+1]$, $y[n-i+1]$ のペアは求めることができる．

また，$c_1[n]$ は $x[n-1]$ と $y[n-1]$ と $c_1[n-1]$ の加算による桁上がりなので $x[n]$, $y[n]$, $a[n]$ から桁上がりの有無つまり，$c_1[n]$ を求めることが可能である．よって，

$c_1[n] = ((x[n] \oplus y[n]) \wedge \bar{a}[n]) \vee ((x[n] \oplus \bar{y}[n]) \wedge a[n])$

また，$c_2[2]$ は $x[n]$ と $y[n]$ と $c_2[1]$ の加算による桁上がりなので

$c_2[2] = (x[n] \wedge y[n]) \vee ((x[n] \oplus y[n]) \wedge c[1])$ である．

・帰納ステップ

$i = k$ のとき，命題成立と仮定し，表 5 より $x[n+1-k], y[n+1-k]$ のペアを求めることが出来とする．

このとき，$x[n+1-k], y[n+1-k]$ のペアがわかるなら，

$c_1[n+1-k] = ((x[n+1-k)] \oplus y[n+1-k]) \wedge \bar{a}[n+1-k]) \vee ((x[n+1-k] \oplus \bar{y}[n+1-k]) \wedge a[n+1-k])$

$c_2[k+1] = (x[n+1-k] \wedge y[n+1-k]) \vee ((x[n+1-k] \oplus y[n+1-k]) \wedge c_2[k])$

により，$c_1[n+1-k], c_2[k+1]$ の値を求めることができる．

$i = k+1$ のとき，上記より $c_1[n+1-k], c_2[k+1], a[n+1-(k+1)], b[k+1]$ の値を求めることができるので $x[n+1-(k+1)], y[n+1-(k+1)]$ がどのようなペアか求めることができる．また $x[n+1-(k+1)], y[n+1-(k+1)]$ のペアがわかるなら

$c_1[n+1-(k+1)] = ((x[n+1-(k+1)] \oplus y[n+1-(k+1)]) \wedge \bar{a}[n+1-(k+1)]) \vee ((x[n+1-(k+1)] \oplus \bar{y}[n+1-(k+1)]) \wedge a[n+1-(k+1)])$

$c_2[k+2] = (x[n+1-(k+1)] \wedge y[n+1-(k+1)]) \vee ((x[n+1-(k+1)] \oplus y[n+1-(k+1)]) \wedge c_2[k+1])$ である．

ゆえに $i = k$ で命題成立ならば $i = k+1$ でも命題成立である．

4.3 両当事者が作成すべきビット列

本研究で使用するデータである x, y は P2 が s_1, s_2 をランダムに並べ替えて長さ h ランダムのビット文字列 w_1, w_2 を加えられた長さ n のビット文字列とする．$(n = M + h)$
$An_2 = |\{j|w_1[j] = w_2[j] = 1, 1 \leqq j \leqq h\}|$ とする．P2 は自分の持つデータ s_2 をランダムに並び替えて空白ビットを作成し，そこにランダムビット文字列 w_1, w_2 の加える．P1，P2 間で x を暗号化状態で演算を行い y と同じ並び替えを行う．そして x と y の 加算結果 a の暗号化データを得る．また，$s'_1[i] = s_1[n + 1 - i]$, $s'_2[i] = s_2[n + 1 - i]$ を要素とする s'_1 と s'_2 にも同様の演算を行い x' と y' の 加算結果 b の暗号化データを得る．

その後，a,b の暗号データを P1 が復号し a, b から $x_1[i], x_2[i]$ がどのようなペアであるか求めることで $x_1[i] = x_2[i] = 1$ となる回数 An_1 を知ることができるので s_1, s_2 の結合後 k-匿名性を満たすか判定することができる．

次節では図 9 の 2 つのビット文字列 x, y の加算結果を算出する方法について考察し従来プロトコルと計算時間を比較する．

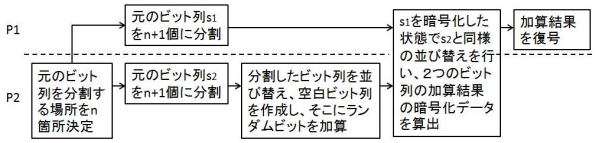

図 9: 提案プロトコルの概要

4.4 実行時間における考察

従来プロトコルと提案プロトコルの各パーティの手続きを OpenSSL ライブラリ [4]，C 言語を用いて計算機上に実現し，シミュレーション実験を行った．計算機環境は 2x2.93Ghz 6-Core Intel Xeon プロセッサ，メモリは 32GB とする．データセットの要素数を変化させながら 3000 回要素数判定したときの計算時間を比較した結果を表 6 に示す．

この結果より，暗号化回数がデータセットの要素数に依存する従来手法ではデータセットの要素数に比例して実行時間が増加していることがわかる．しかし，要素数判定問題を解くための暗号化回数を定数回に抑えた提案手法ではデータセットの要素数に関係なく一定の速度で要素数判定問題を解くことができる．

表 6: 計算時間の比較

データセットの要素数	従来プロトコル (sec)	提案プロトコル (sec)
10	0.54	0.66
30	1.44	0.66
50	2.34	0.66
100	4.59	0.66

5 まとめ

本研究では分散 k-匿名化問題に対する従来研究 [1] で用いられていた SSI_t プロトコルにおいて計算コストがかかっている部分に注目した．各ビットごとに行われていた暗号化をビット列で暗号化し，暗号化関数から得られる結果から必要な情報だけを得る方法を考察した．今後の課題として，2 つのビットの加算結果を作成する流れにおいて機関 P_1, P_2 の情報の漏えいがないかを検討していく必要がある．上記を満たすことが示せたならば本稿で提案したプロトコルは安全であるといえる．

参考文献

[1] W. Jiang and C. Clifton; "A secure distributed framework for achieving k-anonymity," The VLDB Journal , Vol.15, pp.316-333 (2006).

[2] B. Schneier; "Applied Cryptography, Second Edition," John Wiley & Sons Inc. (1996).

[3] L. Sweeny; "k-anonymity: a model for protecting privacy," International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, Vol.10, No.5, pp.571-588 (2002).

[4] http://www.infoscience.co.jp/technical/openssl/docs/