Computer Security Symposium 2017
23 - 25 October 2017

多機関の秘匿和集合演算

宍戸 克成[1]　宮地 充子[1,2,3]

概要：多様かつ膨大な情報を収集，解析を行い，我々の生活に利活用するビックデータ解析が行われている．情報の処理や解析はプライバシ保護がされている状態，つまり暗号化された状態を維持して行われることが望ましい．各機関ので分散して保管されているデータ解析する手法として Multi-Party Private Set Intersection(MPSI) や Multi-Party Private Set Union(MPSU) と呼ばれる技術が利用されている．要素の重複有り集合 (multiset) に対応している既存手法は効率面で改善の余地がある．本研究は効率的な multiset に対応する MPSU プロトコルの提案をする．

キーワード：Multiparty Private Set Union(MPSU), multiset, counting Bloom filter, deprecated Bloom filter

A study of Multiparty private set union protocol

Katsunari Shishido[1]　Atsuko Miyaji[1,2,3]

Abstract: Many and variety information is collected for big data analysis. It is better to maintain encrypted data to keep privacy. Multiparty Private Set Intersection(MPSI) and multiparty Private Set Union(MPSU) protocol have been proposed as one of the technique for data analysis. Both scalability and flexibility become crucial for these protocols. Previous works responding to not only sets and also multisets have improvement on efficiency. In this work, we propose a scalable and flexibility efficient MPSU.

Keywords: Multiparty Private Set Union(MPSU), multiset, counting Bloom filter, deprecated Bloom filter

1. はじめに

情報化社会の進展により情報量が膨大に増加，情報の多様化が進んでいる．それらの情報を収集し，解析結果を我々の私生活に利活用するビックデータ解析が行われている．各機関が情報を分散して保管しているため，解析するために各機関で情報を共有する必要がある．各機関が情報を共有するとき，プライバシ保護の観点からデータは暗号化される．暗号化されたデータは一般に平文に関係のないデータへ置き換えられるため，暗号文を用いた処理や解析は難しくプライバシ保護を行いながら解析を行う手法が必要になっている．

各機関が持つ情報を秘匿しながら，積集合や和集合を求める Private Set Intersection(PSI) や Private Set Union(PSU) と呼ばれている技術がある．はじめに 2 機関の PSI や PSU に関する研究 [3,4,6,8] が行われ，近年は 3 機関以上で構成される Multiparty PSI(MPSI) や Multiparty PSU(MPSU) プロトコル [7,9–12] に関する研究が行われている．MPSI や MPSU は複数機関が集合を持ち，各機関はそれら集合の積集合 (Intersection) や和集合 (Union) のみ得ることができるプロトコルである．また和集合や積集合を求めるプロトコル以外にも，外部サーバーに積集合や和集合の計算を依頼し，得られた結果が正しく求められているか検証する手法 [9] も提案されている．MPSI や MPSU プロトコルは医療機関同士の機密性の高い患者データや難病研究情報の共有などにも利用が期待されている．

既存の MPSI や MPSU プロトコルは各機関 $\mathbf{P}_i$ の集合を

[1] 大阪大学工学研究科
Graduate School of Engineering, Osaka University
[2] 北陸先端科学技術大学院大学 情報科学研究科
Japan Advanced Institute of Science and Technology
[3] 科学技術振興機構 CREST
Japan Science and Technology Agency (JST) CREST

多項式で表現し，生成した複数の多項式同士を計算する方法 [10,11] と集合を確率的なデータ構造である Bloom filter で表現し，生成した複数の Bloom filter 同士を計算する方法 [7,12] が主流である．また，多項式や Bloom filter をベースとするプロトコルは複数機関の入力集合を秘匿するために秘密分散法を用いて生成したシェアから秘密加算や秘密乗算を行う方法と写像の準同型性を用いて暗号文同士の加算や乗算を計算する方法に分けられる．多項式を利用する手法は $\mathbf{P}_i$ が生成する多項式の次数を合わせる必要があるため，各機関が持つ集合の大きさ (データサイズ)$|S_i|$ を合わせる必要がある．Bloom filter を利用する手法は構築時に使用するハッシュ関数が原因で偽陽性が確率的に発生する．また集合に属する要素の重複を許す multiset を扱うとき，Bloom filter を一般化した counting Bloom filter と呼ばれるデータ構造を利用する必要がある．秘密分散法を用いる方法はシェアを生成するために第三者機関 (TP) が必要となるが，TP を準備するコストが高いため利用しない手法が好ましい．これらの点から Bloom filter を用いて準同型性から和集合や積集合を計算する手法が理想的な MPSI や MPSU に近いプロトコルを構築することができると考えられる．本研究は counting Bloom filter を用いることで各機関が持つ multiset に制限が無く，準同型性を用いて各機関の入力集合を秘匿しながら和集合の計算を行う TP を利用しない MPSU プロトコルの提案を目指す．

2. 準備

2.1 安全性仮定と攻撃者モデル

$\mathbb{F}_p$ を体，$g \in \mathbb{F}_p$ を素数位数 q，$x, y \longleftarrow \mathbb{Z}_q^*$ とする．$(\mathbb{F}_p, q, g, g^x, g^y)$ が与えれたときに，g^{xy} の部分情報を求める問題を DDH(Decisional Diffie-Hellman) 問題という．DDH 問題を効率的に解く多項式時間アルゴリズムが存在しないという仮定を DDH 仮定という．そして任意の多項式時間アルゴリズム A，$\epsilon > 0$ を無視できる値としたとき次の式が成り立つ．$|\Pr[x, y \longleftarrow \mathbb{Z}_q^* : A(g, g^x, g^y, g^{xy}) = 1] - \Pr[x, y, z \longleftarrow \mathbb{Z}_q^* : A(g, g^x, g^y, g^z) = 1]| < \epsilon$

次に攻撃者としては Semi-honest model と Maliciou model の 2 つのモデルを定義する．

定義 1 Semi-honest model. Semi-honest model はプロトコルで定められた処理を行い，得られた情報を利用して結果から推測できる情報以外の追加情報を得ようと試みる攻撃者である．

定義 2 Malicious model. Malicious model はプロトコルで定められた処理以外の任意の振る舞いを行い，各機関の秘密情報を得ようと試みる攻撃者である．

2.2 Bloom filter

Bloom filter [1] は集合 S に要素 x が属しているかどうか確認するために使われる空間効率の良い確率的なデータ構造である．Bloom filter は m ビットのブール値を格納する配列と k 個の独立したハッシュ関数 $h_i : \{0,1\}^* \longrightarrow \{0, \cdots, m-1\}$ を利用する．集合 S を表現する Bloom filter を $\mathsf{BF}(S)$ と表記する．

Algorithm 1 const.$\mathsf{BF}(S)$

Require: A set S, k hash function $H = \{h_0, \cdots, h_{k-1}\}$
Ensure: A Bloom filter BF
1: **for** $i = 0$ to $m-1$ **do**
2: $\quad \mathsf{BF}[i] \longleftarrow 0$
3: **end for**
4: **for all** $x \in S$ **do**
5: $\quad$ **for** $i = 0$ to $k-1$ **do**
6: $\quad\quad j = h_i(x)$
7: $\quad\quad$ **if** $\mathsf{BF}[j] = 0$ **then**
8: $\quad\quad\quad \mathsf{BF}[j] \longleftarrow 1$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**

Bloom filter の構築アルゴリズム (const.BF) について説明する．はじめに配列のすべての値を 0 に初期化する．集合 S に属する要素 x を Bloom filter に格納するとき，k 個のハッシュ関数からハッシュ値 $h_i(x)$ を求め，値が指す配列のインデックスに 1 を格納する．つまり，要素 x のハッシュ値が指すインデックスに格納されている値は $\mathsf{BF}[h_i(x)] = 1, (0 \leq i \leq k-1)$ となる．

Algorithm 2 check.$\mathsf{BF}(\mathsf{BF}(S), x)$

Require: A Bloom filter BF, x, k hash function $H = \{h_0, \cdots, h_{k-1}\}$
Ensure: True if $x \in S$, False otherwise
1: **for** $i = 0$ to $k-1$ **do**
2: $\quad j = h_i(x)$
3: $\quad$ **if** $\mathsf{BF}[j] = 0$ **then**
4: $\quad\quad$ return False
5: $\quad$ **end if**
6: **end for**
7: return True

次に Bloom filter を利用して集合 S に要素 x が属しているかどうか確認するアルゴリズム (check.BF) について説明する．候補の要素 x と Bloom filter の構築時に利用した k 個のハッシュ関数からハッシュ値 $h_i(x)$ を求め，値が指すインデックスに格納されている値を確認する．x が S に属しているとき，ハッシュ値が指すインデックスに格納されている値はすべて 1 となる．Bloom filter を使用すると，集合 S に属さない要素 y を確率的に属していると判定する偽陽性が発生する．偽陽性の発生確率は $p = \left\{1 - (1 - \frac{1}{m})^{kw}\right\}^k \approx \left\{1 - e^{-\frac{kw}{m}}\right\}^k$ となることがわかっ

ている [13]．関係式から偽陽性を最小にするときのハッシュ関数の個数を $k = \frac{m}{w} \cdot \ln 2$ で求めることができる．対して Bloom filter を使用しても偽陰性は発生しない．よって，Bloom filter を使用すると必ず集合 S に属する要素 x は属していることを判定できる．

2.3 countnig Bloom filter

counting Bloom filter は Bloom filter が利用する m ビットのブール値を格納する配列を m ビットの整数値を格納する配列に変形したデータ構造である．つまり，集合の要素の重複数のカウントが可能であるため，要素の重複が許される集合 (multiset) の表現が可能である．multiset S を表現する counting Bloom filter を $\mathsf{CBF}(S)$ と表記する．

Algorithm 3 const.$\mathsf{CBF}(S)$

Require: A multiset S, k hash function $H = \{h_0, \cdots, h_{k-1}\}$
Ensure: A counting Bloom filter CBF

```
for i = 0 to m − 1 do
    CBF[i] ⟵ 0
end for
for all x ∈ S do
    for i = 0 to k − 1 do
        j = h_i(x)
        if CBF[j] < #x then
            CBF[j] ⟵ #x
        end if
    end for
end for
```

counting Bloom filter の構築アルゴリズム const.CBF について説明する．はじめに配列のすべての値を 0 に初期化する．multiset S の要素 x を格納するとき，k 個のハッシュ関数からハッシュ値 $h_i(x)$ を求め，値が指すインデックスに格納されている値が要素 x の重複数 $\#x$ よりも小さいとき $\#x$ を格納する．

Algorithm 4 check.$\mathsf{CBF}(\mathsf{CBF}(S),x)$

Require: A counting Bloom filter CBF, x, k hash function $H = \{h_0, \cdots, h_{k-1}\}$
Ensure: the number of $x \in S$ $\#x$

```
return min{CBF[h_j(x)]}  (0 ≤ j ≤ k − 1)
```

次に counting Bloom filter を利用して multiset S に属する要素 x の重複数 $\#x$ を確認するアルゴリズム check.CBF について説明する．要素 x と k 個のハッシュ関数からハッシュ値 $h_i(x)$ を求め，値が指すすべてのインデックスに格納されている値の中の最小値を x の重複数とする．counting Bloom filter も Bloom filter と同様，偽陽性が $p = \left\{1 - (1 - \frac{1}{m})^{kw}\right\}^k$ で発生する．counting Bloom filter の偽陽性は集合に属していない要素を属していると判定するケースと集合に属している要素の重複数が本来の重複数よりも多く判定されるケースがある．

2.4 加法準同型暗号

加法準同型暗号は暗号化されているデータを処理するために非常に重要な性質を持つ．本研究は加法準同型暗号のひとつである exponential ElGamal 暗号 (Ex-ElGamal) [2] を利用する．Ex-ElGamal は以下に示す加法準同型性を満たす．

加法準同型性
(1) $m_1, m_2 \in \mathbb{Z}_q : \mathrm{Enc}(m_1) \cdot \mathrm{Enc}(m_2) = \mathrm{Enc}(m_1 + m_2)$.
(2) $m, k \in \mathbb{Z}_q : \mathrm{Enc}(m)^k = \mathrm{Enc}(km)$.

本研究は n 機関で構成されるプロトコルを構築するため，(n, n) 分散復号方式の Ex-ElGamal [5] を利用する．(n, n) 分散復号方式は必ずプロトコルに参加している n 機関で協力しなければ復号することができない．(n, n) 分散復号方式の Ex-ElGamal も典型的な公開鍵暗号と同じく 3 つ組 (鍵生成，暗号化，復号) のアルゴリズムで構成される．

鍵生成アルゴリズム．$\mathbf{KeyGen}(1^{\lambda}) = ((p, g, y), x_i)$

素数 p，素数位数 q のベースポイント $g \in \mathbb{F}_p$，各機関の秘密鍵 $x_i \in \mathbb{Z}_q^*$ とする．これらの値から公開鍵 $pk = \prod_{i=1}^{n} g^{x_i}$ を計算する．各機関は秘密鍵 x_i を秘密裏に持ち，(p, g, y) は公開値として共有する．

暗号化アルゴリズム．$\mathsf{Enc}[m] = (u, v)$

暗号化アルゴリズムは平文 $m \in \mathbb{Z}_q$ と公開鍵 pk を入力し，暗号文 (u, v) を出力する．一様ランダムに生成した乱数を $r \in \mathbb{Z}_q^*$ とすると $u = g^r \pmod p$，$v = g^m \cdot y^r \pmod p$ となる．

復号アルゴリズム．$\mathsf{dis.Dec}[(u, v)] = g^m$

各機関 $\mathbf{P}_\mathsf{i}$ は $z_i = u^{sk_i} \pmod p$ を計算することで分散復号を行い，全機関へ送信する．その後，各機関は全機関で共有した z_i から $z = \prod_{i=1}^{n} z_i \pmod p$ を計算し，$g^m = v/z \pmod p$ を求め平文を g^m の形に復号する．

3. 既存研究

3.1 多項式を用いた多機関 PSU

はじめに集合と多項式の対応ついて説明する．集合を $S = \{s_1, ..., s_w\}$ とすると，集合 S に対応する多項式は $f(x) = \prod_{j=1}^{w}(x - s_j)$ となる．つまり，集合の要素を多項式の根とする．次に，集合に属する要素の確認方法について説明する．S を表す多項式は集合に属する要素 $a \in S$ に対し，$f(a) = 0$ が成り立つ．よって，多項式に要素を代入し，零になるかどうか確認することで集合に属するかどうか判定することができる．このプロトコルで利用する加法準同型暗号システム $\epsilon = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ とし，公開鍵を pk，秘密鍵を sk と表記する．以下に示す多項式の計算は加法準同型暗号システムを用いて暗号化されていることを

前提とする．

多機関PSU [11] について説明する．入力は各機関$\mathbf{P}_i$の集合S_i，出力は各機関の和集合$\cup S_i$にt個以上属する要素である．各機関を$\mathbf{P}_1, ..., \mathbf{P}_n$とし，それぞれmultiset $S_1, ..., S_n$を持つ．ここで集合の大きさは$|S_1| = ... = |S_n| = w$とする．また平文の定義域$P \supset S_i$で，一様ランダムに要素を取り出したとき，$a \in P$を選ぶ確率が無視できる確率となる大きな集合をRとする．各機関はプロトコルで利用する準同型暗号システムϵの秘密鍵skを共有し，暗号化に用いる公開鍵をpkとする．結託している機関数は$c < n$と仮定する．

各機関$\mathbf{P}_i$は集合S_iを多項式$f_i = \prod_{j=1}^{w}(x - s_{i,j})$に変換後，暗号化する．$\mathbf{P}_1$は暗号化した多項式$\lambda_1 = f_1$を$\mathbf{P}_2$へ送信する．次に$\lambda_{i-1}$を受信した$\mathbf{P}_{i-1}$は$\lambda_i = \lambda_{i-1} \cdot f_i$を計算し，$\mathbf{P}_{i+1 \pmod n}$へ送信する．$\mathbf{P}_1$は$p = \lambda_n = \prod_{i=1}^{n} f_i$を$\mathbf{P}_2$から$\mathbf{P}_{c+1}$の$c$機関に送信する．$\mathbf{P}_1$から$\mathbf{P}_{c+1}$は多項式$p$の$t-1$階微分$p^{(t-1)}$を計算する．各機関は$nw$次のランダムな多項式$r_i, s_i$と$(t-1)$次のある多項式$F$ ($\forall x \in P,\ F(x) \neq 0$)を選択し，$p \cdot s_i + F \cdot p^{(t-1)} \cdot r_i$を計算し，他の機関に送信する．全機関は受信したデータを用いてグループ復号を行い，多項式$\Phi = F \cdot p^{t-1} \cdot (\sum_{i=1}^{c+1} r_i) + p \cdot (\sum_{i=1}^{c+1} s_i)$を得る．最後に求めた多項式$\Phi$に各機関が持つ集合$S_i$に属する要素を代入し，$t$個以上属する要素とランダマイズされた$t$個未満の要素を全機関に送信する．各機関$\mathbf{P}_i$の通信量は$\mathcal{O}(n^2w^2)$，計算量は$\mathcal{O}(n^2w^2)$となる．

2機関の多項式PSIプロトコル [8] をMPSIプロトコルに拡張したプロトコル [10] も提案されている．[8] は$\mathbf{P}_1, \mathbf{P}_2$のうち，$\mathbf{P}_2$が集合$S_2$を加法準同型暗号を用いて暗号化し，$\mathbf{P}_1$へ公開鍵$pk$と暗号化された多項式$\mathsf{Enc}(f_2(x))$を送信する．$\mathbf{P}_1$は$\forall a \in S_1$を受け取った公開鍵で暗号化し，暗号化されている多項式に対して$a \in S_1 \cap S_2$ならば$\mathsf{Enc}(f_2(a)) = 0$を満たすので積集合に属する要素を求めること手法である．この手法を$\mathbf{P}_1 - \mathbf{P}_2, \mathbf{P}_1 - \mathbf{P}_3, ..., \mathbf{P}_1 - \mathbf{P}_n$のようなスターネットワークトポロジーを構築し積集合を求める手法が [10] である．この手法は初めて2機関PSIプロトコルをベースとするMPSIプロトコル示した．$\mathbf{P}_2, ..., \mathbf{P}_n$が持つ集合で最大の集合大きさを$|S_i| = m_2$，$|S_1| = m_1$とすると，計算量は$\mathcal{O}(m_1 m_2)$，通信量は$\mathcal{O}(nm_2)$となる．

3.2 Bloom filterを用いた多機関PSU

n個のmultisetからそれらの和集合の求め方について説明する．$x \in \cup S_i$ならば$x \in S_1 \vee x \in S_2 \vee ... \vee x \in S_n$が成り立つ．従って．それぞれの集合$S_i$を長さ$m$の配列のcounting Bloom filter $\mathsf{CBF}(S_i)$に変換し，対応するインデックス毎の加算$\mathsf{CBF}_{union}[u] = \sum_{i=1}^{n} \mathsf{CBF}_i[u]$を行うことで和集合を求めることができる．つまり，それぞれのmultiset S_iを表現するcounting Bloom filterのインデックス毎の加算を求めることができれば，multisetの和集合に属する要素の重複数を得ることができる．

counting Bloom filterとSEPIAライブラリを利用しPSUを実現する手法 [12] から説明する．各機関$\mathbf{P}_i$が持つmultiset S_iを入力とし，和集合に属する要素xの重複数$\#x$を出力する．SEPIAはシャミアの秘密分散法 [14] を基に秘密計算が可能であり，semi-honest modelに対して安全である．SEPIAは入力のみを実行するn個のInput peer(IP)とシェアを用いて秘密計算を実行するk個のprivacy peer(PP)から構成される．はじめに$\mathrm{IP}_i (1 \leq i \leq n)$はmultiset S_iを長さmの配列のcounting Bloom filter $\mathsf{CBF}(S_i)$に変換する．その後生成した$\mathsf{CBF}(S_i)$からシェアを生成し，各機関が生成したシェアを足し合わせることで和集合を表すcounting Bloom filterのシェアを求めることができる．第三者機関の計算量は$\mathcal{O}(n\omega)$となる．$\#x$を得るために，counting Bloom filterを構築するときに使用したk個のハッシュ関数を使用する．偽陽性の発生率は$p = \left\{1 - (1 - \frac{1}{m})^{kw}\right\}^k \approx \left\{1 - e^{-\frac{kw}{m}}\right\}^k$で与えられる．

Bloom filterを用いたPSU Cardinality(PSU-CA)プロトコル [7] について説明する．このプロトコルは [12] で提案されている手法と秘密分散法のシェアの生成方法が異なる．PSU-CAは各機関$\mathbf{P}_i$が持つ集合S_iを入力とし，和集合の大きさ$|\cup S_i|$の概数を出力するプロトコルである．このプロトコルは各機関の他に秘密分散のシェアを受信し，置換処理を行うα組の機関$\mathbf{P}_{accumulator}$と，置換処理された秘密分散のシェアを受信し，秘密を復号，その後，和集合の大きさの概数を求め出力する機関$\mathbf{P}_{calculaotor}$から構成される．はじめに各機関は集合S_iからBloom filter $\mathsf{BF}(S_i)$に変換する．生成されたBloom filterから秘密分散の複数個のシェアを生成する．シェアの生成方法について説明する．Bloom filterに格納されている0, 1の値をそれぞれ$0 \longmapsto 0 \bmod 2^b, 1 \longmapsto (\text{random element}) \bmod 2^b$のように$b$-bitの値に変換する．この$b$-bitの値を$r = r_1 + ... + r_n \pmod{2^b}$を満たす値$r_i$に分割し，これらをシェアとして$\mathbf{P}_{accumulator\,j}$に送信する．アキュムレータは受信したシェアの総和を求め，結果にある置換$\pi \rightarrow \{0, ..., m-1\}$処理を行う．その後，各アキュムレータが持つ置換処理をしたシェアをカリキュレータに集め，Bloom filterを復元する．復元したBloom filterの0の個数zをカウントし，$|\cup S_i| \approx \{\ln z/m\}/\{k \cdot \ln(1 - 1/m)\}$から集合の大きさの概数の計算を行う．カリキュレータは秘密を復元可能だが，アキュムレータの置換処理により，元のBloom filterを得ることができない．また，Bloom filterの元々の値が1だが，復元したBloom filterで値が0となる場合があり，これが原因による誤差を含んでいる．各機関の通信量はωを各機関の持つ集合の大きさ$|S_i| = \omega_i$の最大値とすると，$\mathcal{O}(n\omega)$となり，各機関の計算量は$\mathcal{O}(\omega_i)$，第三者機関の計算量は$\mathcal{O}(n\omega)$となる．秘密分散法を用いる手法はシェアを生成するために第三者機関 (ディーラー) やシェアを受

け取り秘密計算をする第三者機関が必要となることに注意する．

4. 提案

4.1 表記

提案手法の説明の前に使用する記号の説明をする．

- $\mathbf{P}_\mathsf{i}$: i 番目の機関，$i = 1, \cdots, n$
- $S_i = \{s_{i,1}, s_{i,2}, \cdots, s_{i,w_i}\}$: $\mathbf{P}_\mathsf{i}$ が所持する $|S_i| = \omega_i$ の multiset．
- X: 各機関が持つ集合と $X \supset S_i$ の関係を持つ全体集合
- $\cup S_i$: n 機関の和集合
- $\mathsf{Enc}, \mathsf{dis.Dec}$: (n, n) 分散復号方式の Ex-ElGamal 暗号の暗号化と復号アルゴリズム
- m, k: Bloom filter の配列数とハッシュ関数の数
- $\mathsf{BF}(S_i) = [\mathsf{BF}_i[0], \cdots, \mathsf{BF}_i[m-1]]$: 集合 S_i を表現する Bloom filter
- $\mathsf{CBF}(S_i) = [\mathsf{CBF}_i[0], \cdots, \mathsf{CBF}_i[m-1]]$: multiset S_i を表現する counting Bloom filter
- $\mathsf{DBF}(S_i) = [\mathsf{NBF}_i[0], \cdots, \mathsf{NBF}_i[m-1]]$: multi-set S_i 表現する deprecated Bloom filter
- $\mathsf{agg.CBF}(\cup S_i) = [\sum_{i=1}^n \mathsf{CBF}_i[0], \cdots, \sum_{i=1}^n \mathsf{CBF}_i[m-1]]$: multiset S_i を表現する counting Bloom filter を統合した Bloom filter
- $\mathsf{agg.DBF}(\cup S_i) = [\sum_{i=1}^n \mathsf{NBF}_i[0], \cdots, \sum_{i=1}^n \mathsf{NBF}_i[m-1]]$: multiset S_i を表現する deprecated Bloom filter を統合した Bloom filter

4.2 提案手法

本研究の目的は multiset S_i の和集合を計算し，和集合に属する要素の重複数を求めることである．前提条件として各機関は全体集合 X について既知であるが，求めるべき和集合 $\cup S_i \subset X$ や和集合に属する要素の重複数について既知でないとする．

和集合を求める簡単な手法は各機関が持つ multiset S_i を counting Bloom filter $\mathsf{CBF}(S_i)$ に変換し，それらを統合することで $\mathsf{agg.CBF}(\cup S_i)$ を求める．そして得られた $\mathsf{agg.CBF}(\cup S_i)$ に要素 $x \in X$ が $\mathsf{agg.CBF}(\cup S_i)$ に属するかどうか確認し，重複数 $\#x$ を出力することである．しかし，$\mathsf{agg.CBF}(\cup S_i)[j]$ はハッシュ値が $j = h_s(a_{i,j})$ $(0 \le \forall s \le k-1)$ となる要素 $a_{i,j} \in S_i$ の最大重複数となる．つまり，簡単な手法は正確な重複数よりも大きな重複数を出力することがある．

正確な重複数を出力する確率を大きくするために新しい記号を定義する．

- h_a: 要素 $a \in S_i$ の特定ハッシュ．h_a は $\forall x \in X$ に対して $h_a(a) \neq h_i(x)$ $(0 \le i \le k-1)$ を満たす．
- DBF: S_i に属する各要素をカウントした Bloom filter

提案する MPSU プロトコルの説明をする．

初期設定

(1) 体 $\mathbb{F}_p$，素数位数 q のベースポイント $g \in \mathbb{F}_p$ とする．各機関 $\mathbf{P}_\mathsf{i}$ はランダムに秘密鍵 $x_i \in \mathbb{Z}_q^*$ を選択し，$y_i = g^{x_i}$ を計算後全機関に公開する．

(2) 各機関 $\mathbf{P}_\mathsf{i}$ は公開されている y_i から暗号化に利用する公開鍵 $y = \prod_{i=1}^n y_i \pmod p$ を計算する．

Bloom filter の構築と暗号化

(1) $\mathbf{P}_\mathsf{i}$ は const.BF と const.CBF と実行し，multiset S_i から $\mathsf{DBF}(S_i)$ と $\mathsf{CBF}(S_i)$ を生成する．

(2) $\mathbf{P}_\mathsf{i}$ は公開鍵 y を利用し，生成した $\mathsf{DBF}(S_i)$ と $\mathsf{CBF}(S_i)$ を暗号化する．

$\mathsf{Enc}_y\{\mathsf{CBF}(S_i)\} = [\mathsf{Enc}_y(\mathsf{CBF}_i[0]), \cdots, \mathsf{Enc}_y(\mathsf{CBF}_i[m-1])]$

$\mathsf{Enc}_y\{\mathsf{DBF}(S_i)\} = [\mathsf{Enc}_y(\mathsf{NBF}_i[0]), \cdots, \mathsf{Enc}_y(\mathsf{NBF}_i[m-1])]$

(3) $\mathbf{P}_\mathsf{i}$ は $\mathsf{Enc}_y\{\mathsf{CBF}(S_i)\}$ と $\mathsf{Enc}_y\{\mathsf{DBF}(S_i)\}$ を全機関へ送信する．

和集合の計算と分散復号

(1) $\mathbf{P}_\mathsf{i}$ は $\mathsf{Enc}_y\{\mathsf{CBF}(S_i)\}$ と $\mathsf{Enc}_y\{\mathsf{DBF}(S_i)\}(1 \le i \le n)$ と Ex-ElGamal 暗号の加法準同型性から以下を計算する．

- $\mathsf{Enc}_y\{\mathsf{agg.CBF}(\cup S_i)\} = \prod_{i=1}^n \mathsf{Enc}_y\{\mathsf{CBF}(S_i)\}$
- $\mathsf{Enc}_y\{\mathsf{agg.DBF}(\cup S_i)\} = \prod_{i=1}^n \mathsf{Enc}_y\{\mathsf{DBF}(S_i)\}$

(2) $\mathbf{P}_\mathsf{i}$ は秘密鍵 x_i を用いて計算した $\mathsf{Enc}_y\{\mathsf{agg.CBF}(\cup S_i)\}$，$\mathsf{Enc}_y\{\mathsf{agg.DBF}(\cup S_i)\}$ を全機関で協力することで分散復号し，$\mathsf{agg.CBF}(\cup S_i)$ と $\mathsf{agg.DBF}(\cup S_i)$ を得る．

(3) $\mathbf{P}_\mathsf{i}$ は $\mathsf{agg.CBF}(\cup S_i)$ と $\mathsf{agg.DBF}(\cup S_i)$ の各配列の要素 g^d を d へ変換する．

和集合に属する要素の重複数の確認

$\mathbf{P}_\mathsf{i}$ は和集合に属する要素 $\forall x \in X$ の重複数を以下の手順で確認する．

(1) $\mathbf{P}_\mathsf{i}$ は特定ハッシュ h_a を持つ要素 $a \in X$ の重複数 $\#a \in \cup S_i$ を $\mathsf{agg.DBF}(\cup S_i)[h_j(x)](0 \le j \le k-1)$ の最小値とする．

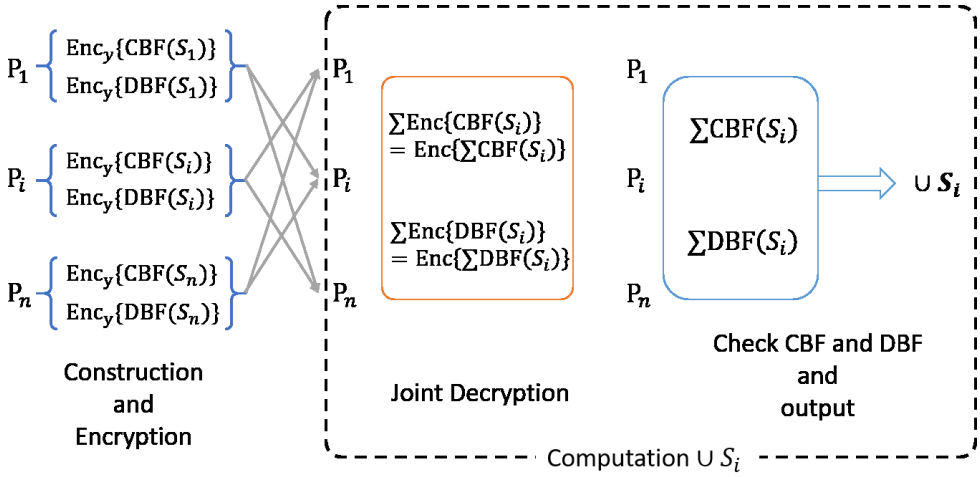
(2) $\mathsf{agg.DBF}(\cup S_i)[h_j(a)] = \mathsf{agg.DBF}(\cup S_i)[h_j(a)] - \#a$ を計算し，要素 a の重複数を $\mathsf{agg.DBF}(\cup S_i)$ に更新する．

(3) 特定ハッシュを持つ要素の重複数の確認後に残りの要素 $x \in X$ の重複数の上界として $\mathsf{agg.DBF}(\cup S_i)[h_j(x)](0 \le j \le k-1)$ の最小値を出力する．

4.3 正当性と安全性

MPSU プロトコルの正当性を証明する．

定理 1. $\cup S_i \ni \{a, \cdots, a\}$ $(\#a = \eta)$ ならば $\{a, \cdots, a\}$ $(\#a = \eta)$ は $\cup S_i$ に属すると出力されるとき MPSU プロトコルは正当性を満たす．



証明.

特定ハッシュを持つ要素 $a(\#a=\eta)$ のハッシュ値 $h_a(a)$ はその他の要素 $\forall x \in X$ の任意のハッシュ値 $h_a(a) \neq h_i(x)$ $(0 \leq i \leq k-1)$ を満たす．つまり，counting Bloom filter，及び，deprecated Bloom filter に要素 a の重複数 η だけ格納しているインデックスが存在する．

従って，要素 a のハッシュ値 $h_i(a)$ が指すインデックスの中で特定ハッシュが指すインデックスに格納されている値は他のインデックスに格納されている値よりも小さくなり，$a \in \cup S_i$ の重複数として η が出力される．また，出力された重複数を $h_i(a)$ が指す deprecated Bloom filter のインデックスに更新する．deprecated Bloom filter はインクリメント処理により生成されている．他の要素が指すハッシュ値と衝突している配列インデックスは各要素の重複数の和が格納されているため，要素の正確な重複数を引くことで値の更新が可能である．また a の重複数が更新されると $h_i(a)$ と衝突しているハッシュ値 $h_i(x)$ が特定ハッシュになる可能性がある．特定ハッシュの重複数を確認し，deprecated Bloom filter を更新する処理を繰り返すことで，最終的に特定ハッシュを持たない要素が残ることになる．

特定ハッシュを持たない要素の重複数について説明する．特定ハッシュを持たない要素 $x(\#x=d)$ はある要素 $y \in \cup S_i$ に対して $h_i(x)=h_j(y)$ $(0 \leq i,j(i \neq j) \leq k-1)$ を満たす．deprecated Bloom filter は複数の要素の重複数の和が各インデックスに格納されるため，要素の重複数を得ることができない．対し，counting Bloom filter は上記の条件が成り立つ要素の中で最も大きい重複数がインデックスに格納されている．つまり，要素 x の重複数 d が重複数 $t(t<d)$ の要素とハッシュ値が衝突しているとき，counting Bloom filter に t よりも大きな値 d が格納される．よって，重複数 d の要素 x は正確な重複数を得ることができる．それ以外の要素 x の重複数は $h_i(x)=h_j(y)$ $(0 \leq i,j(i \neq j) \leq k-1)$ を満たすすべての要素 y の中で，最も小さい数 $u(d<u)$ を重複数の上界として得ることになる．

従って，提案した MPSU プロトコルは $\cup S_i \ni \{a, \cdots, a\}$ $(\#a=\eta)$ ならば $\{a, \cdots, a\}(\#a=\eta)$ が $\cup S_i$ に属すると出力するため，正当性を満たす．

次に提案した MPSU プロトコルの安全性を証明する．

定理 2. n 機関よりも少ない機関が結託していても，MPSU プロトコルは semi-honest model に対して DDH 仮定の基で安全である．

証明.

各機関 $\mathbf{P}_i$ が得る値 $\mathsf{Enc}_y\{\mathsf{CBF}(S_i)\}$ $=[\mathsf{Enc}_y(\mathsf{CBF}_i[0]), \cdots, \mathsf{Enc}_y(\mathsf{CBF}_i[m-1])]$ と $\mathsf{Enc}_y\{\mathsf{DBF}(S_i)\}$ $=[\mathsf{Enc}_y(\mathsf{NBF}_i[0]), \cdots, \mathsf{Enc}_y(\mathsf{NBF}_i[m-1])]$ がそれぞれ乱数 $\mathbf{r}=[r_0, \cdots, r_{m-1}] \in \mathbb{Z}_q^m$ と識別不可能であることを証明する．

MPSU プロトコルが生成した値 $\mathsf{Enc}_y\{\mathsf{CBF}(S_i)\}$ と判定したときに 1 を出力し，乱数 $\mathbf{r}$ と判定したときに 0 を出力する多項式時間識別器 $\mathbf{D}$ を仮定する．ここで，DDH 問題を解くシミュレーター $\overline{\mathrm{SIM}}$ を構築できることをで示す．シミュレーター $\overline{\mathrm{SIM}}$ は DDH チャレンジ文 $(\overline{g}, \overline{g}^{\alpha}, \overline{g}^{\beta}, \overline{g}^{\gamma})$ を受け取ると以下の処理を実行する．

(1) 公開鍵を $y = \overline{g}^{\beta}$ と置き，メッセージとして $k_0, \cdots, k_{m-1} \in \mathbb{Z}_q$，乱数として $r_0, \cdots, r_{m-1} \in \mathbb{Z}_q$ を一様ランダムに選択する．

(2) $\overline{\mathsf{Enc}\{\mathsf{CBF}(S_j)\}}$ として $[(\overline{g}^{\alpha}, \overline{g}^{k_0} \cdot \overline{g}^{\gamma}), ((\overline{g}^{\alpha})^{r_1}, \overline{g}^{k_1} \cdot (\overline{g}^{\gamma})^{r_1}), \cdots, ((\overline{g}^{\alpha})^{r_{m-1}}, \overline{g}^{k_{m-1}} \cdot (\overline{g}^{\gamma})^{r_{m-1}})]$ を識別器 $\mathbf{D}$ へ送信する．

DDH 問題のチャレンジ文 $(\overline{g}, \overline{g}^{\alpha}, \overline{g}^{\beta}, \overline{g}^{\gamma})$ が $\gamma=\alpha\beta$ を満たすとき，$\overline{\mathsf{Enc}\{\mathsf{CBF}(S_j)\}}$ は MPSU プロトコルで生成された値と同じ分布となるため，識別器 $\mathbf{D}$ は 1 を出力する．対して，$\gamma \in \mathbb{Z}_q$ が一様なランダムな乱数のとき，$\overline{\mathsf{Enc}\{\mathsf{CBF}(S_j)\}}$ は一様ランダムな分布と一致するため識別器 $\mathbf{D}$ は 0 を出

表 1　既存研究との比較

protocol	[11]	[12]	[7]	our protocol
TTP	No	Yes	Yes	No
input	S_i	S_i	S_i	S_i
input data	$\|S_i\| = \omega$	Any	Any	Any
output	$\#x \in \cup S_i$	$\#x \in \cup S_i$	$\|\cup S_i\|$	$\#x \in \cup S_i$
multiset	Yes	Yes	No	Yes
Computation	$\mathcal{O}(n^2\omega^2)$	$\mathbf{P}_\mathrm{i}$: $\mathcal{O}(\omega_i)$, TP: $\mathcal{O}(nw)$	$\mathbf{P}_\mathrm{i}$: $\mathcal{O}(\omega_i)$, TP: $\mathcal{O}(nw)$	$\mathcal{O}(nw)$
Communication	$\mathcal{O}(n^2\omega^2)$	$\mathcal{O}(n\omega)$	$\mathcal{O}(n\omega)$	$\mathcal{O}(n\omega)$

力する．よって，識別器 $\mathbf{D}$ を用いてランダムに推測するよりも有意な確率で DDH 問題を答えるシミュレーターを構成できる．

DDH 仮定が成り立つとすると，DDH 問題を解くシミュレーターの存在が仮定に矛盾する．また，各機関 $\mathbf{P}_\mathrm{i}$ は n 機関より少ない機関で復号処理を実行することができない．つまり，暗号文 $\mathsf{Enc}_y\{\mathsf{CBF}(S_i)\}$ は分散復号されるまで情報を漏らすことがない．従って，n 機関よりも少ない機関が結託していても，MPSU プロトコルは semi-honest model に対して DDH 仮定の基で安全である．

$\mathsf{Enc}_y\{\mathsf{DBF}(S_i)\} = [\mathsf{Enc}_y(\mathsf{NBF}_i[0]), \cdots, \mathsf{Enc}_y(\mathsf{NBF}_i[m-1])]$ と乱数 $\mathbf{r} = [r_0, \cdots, r_{m-1}] \in \mathbb{Z}_q^m$ が識別不可能であることも同じ方法で証明することができる．

5. 比較

既存の MPSU プロトコルと第三者機関の有無，入力の制限，計算量，通信量について比較する．表 1 で既存プロトコルと比較を行う．[11] は DCR 仮定の基で安全な公開鍵暗号システム (Paillier 暗号)，本研究の提案プロトコルは DDH 仮定の基で安全な公開鍵暗号システム (Ex-ElGamal) を用いることで，信頼できる第三者を要することなく semi-honest な攻撃者に対して安全なプロトコルである．しかし，[7,12] は秘密分散を用いるため信頼できる第三者機関が必要となる．多項式を用いる既存プロトコルは生成する多項式の次数を全機関で揃える必要があるため，集合の大きさを $|S_i| = \omega_i$ に合わせる必要がある．また重複数 t の要素を確認するために t 階微分を求める必要があり，各機関の計算量は $\mathcal{O}(n^2\omega^2)$ となる．また通信量はランダマイズ処理に必要なランダムな多項式が含まれるため，$\mathcal{O}(n^2\omega^2)$ となる．本研究で提案したプロトコルは各機関が持つ multiset の大きさに制限がなく，それぞれの集合の大きさは $|S_i| = \omega_i$ とする．各機関 $\mathbf{P}_\mathrm{i}$ が counting Bloom Filter, duplication Bloom Filter を構築するのに要する時間はそれぞれ $\mathcal{O}(\omega_i)$ である．各機関で最も大きな集合の大きさを ω とすると，暗号化されたデータの計算に $\mathcal{O}(n\omega)$ 要する．各機関の通信量は counting Bloom Filter と deprecated Bloom filter の大きさに依存するため，$\mathcal{O}(n\omega)$ となる．

6. まとめ

本研究は効率的な MultiParty Private Set Union(MPSU) プロトコルを提案した．提案は信頼できる第三者機関を必要しないプロトコルで，秘密分散法をベースに利用しているプロトコルよりもコストが低いと考えられる．また，counting Bloom filter を利用したプロトコルはハッシュの衝突が原因で偽陽性がある確率で発生してしまうが本研究は偽陽性の影響を受けにくい，正しい要素の重複数を出力する確率を増やすプロトコルを初めて示した．今後は MultiParty Private Set Intersection(MPSI) に提案した方式の拡張や提案プロトコルの効率化が課題である．

謝辞

本研究の一部は JSPS 科研費基盤 C (JP15K00183) と (JP15K00189) 及び科学技術振興機構（JST）の CREST(JPMJCR1404) と国際科学技術協力基盤整備事業 及び 文部科学省の情報技術人材育成のための実践教育ネットワーク形成事業 分野・地域を越えた実践的情報教育協働ネットワークの助成を受けています．

参考文献

[1] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[2] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European transactions on Telecommunications*, 8(5):481–490, 1997.

[3] E. De Cristofaro, P. Gasti, and G. Tsudik. Fast and private computation of cardinality of set intersection and union. In *CANS 2012*, volume 7712 of *LNCS*, pages 218–231. Springer, 2012.

[4] E. De Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In *FC 2010*, volume 6052 of *LNCS*, pages 143–159. Springer, 2010.

[5] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *CRYPTO 1989*, volume 1462 of *LNCS*, pages 307–315. Springer, 1989.

[6] C. Dong, L. Chen, and Z. Wen. When private set intersection meets big data: An efficient and scalable protocol. In *ACMCCS 2013*, pages 789–800. ACM, 2013.

[7] R. Egert, M. Fischlin, D. Gens, S. Jacob, M. Senker, and J. Tillmanns. Privately computing set-union and

set-intersection cardinality via bloom filters. In *ACISP 2015*, volume 9144 of *LNCS*, pages 413–430. Springer, 2015.

[8] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, 2004.

[9] Esha Ghosh, Olga Ohrimenko, Dimitrios Papadopoulos, Roberto Tamassia, and Nikos Triandopoulos. Zero-knowledge accumulators and set algebra. In *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II 22*, pages 67–100. Springer, 2016.

[10] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Scalable multi-party private set-intersection. In *IACR International Workshop on Public Key Cryptography*, pages 175–203. Springer, 2017.

[11] L. Kissner and D. Song. Privacy-preserving set operations. In *CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257. Springer, 2005.

[12] D. Many, M. Burkhart, and X. Dimitropoulos. Fast private set operations with sepia. *Technical Report*, 345, 2012.

[13] Michael Mitzenmacher and Eli Upfal. *Probability and Computing Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[14] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.