Vol.2019-CSEC-85 No.18
Vol.2019-IOT-45 No.18
2019/5/24

# 素集合判定を利用した二者間秘匿インターバルテスト

宍戸 克成[1,a)]　宮地 充子[1,2]

**概要**：情報通信技術が発達し，機密情報を含むデータ解析や提供サービスが活発化している．一方で，プライバシーの漏洩が問題のひとつに挙げられているため，データ解析等におけるプライバシー保護の実現が必要不可欠である．本研究はクライアントとサーバーの二者間秘匿積集合演算の特殊ケースである二者間素集合判定プロトコルを提案する．本稿では実用的な観点から計算量をオンラインとオフラインに分けて評価し，本方式はオンラインにおけるクライアントの計算量が定数，そしてサーバーからクライアントの通信量も定数を実現する．つまり，オンラインのクライアントの計算量と通信量が集合のサイズに依存しない．したがって，本方式はサーバーが複数のクライアントから同時にリクエストを受ける状況においても，クライアントの待ち時間を最小にするサービスを構築できる．

**キーワード**：0/1 encoding, Bloom Filter，Private Set Intersection，素集合判定

# Secure two–party interval test based on the dis–joint sets check

Katsunari Shishido[1,a)]　Atsuko Miyaji[1,2]

## 1. 序論

### 1.1 背景

コンピューター性能の向上や大量のデータ集積が多種多様なデータを使った処理・解析を可能にした．データ解析から得られた新たな知見の 2 次利用により，研究やビジネスの活発化が期待されている．実際に，日本政府はインターネット上でオープンデータを公開し，社会経済全体の発展を目指している．サービス提供者は自身で大量のデータを処理・解析して得られた情報を使い，サーバー上でデータ処理システムを運用して多数のユーザー (クライアント) にサービスを提供できる．そしてユーザーは自身の入力した情報に関連する有用な情報を容易に得ることができる．しかし，このようなビジネスモデルはユーザー情報やサービス提供者の価値ある情報が漏洩するリスクがある．これまでに提案されてきた二者間の秘匿計算やプライバシー保護技術は「クライアント – 軽負荷，サーバー – 重負荷」の構成である．方式の効率性を議論するとき，サーバーに同時に接続するユーザー数も考慮する必要がある．サーバーの負荷はユーザー数に線形的に依存するため，サーバーの負荷を削減することが重要な課題である．そしてユーザーがサーバーに情報を送信してから結果を得るまでに要する時間はサーバーの処理時間+サーバーがユーザーデータを送信する時間+クライアントがサーバーから受け取ったデータの処理時間 (オンラインフェーズの処理時間) となるため，サーバーがユーザーに送信するデータ数の削減とサーバーからデータを受け取った後のクライアントの負荷の削減が必須である．

### 1.2 二者間秘匿インターバルテスト

本研究は二者間の秘匿インターバルテストを扱う．二者間のインターバルテストは秘匿大小比較と同様に統計解析，データ分類，機械学習の構成要素として用いられるため重要である．一般に数値のビット毎の処理により，大小

[1] 大阪大学大学院 工学研究科
Graduate School of Engineering, Osaka University, Suita, Osaka 565–0871, Japan
[2] 北陸先端科学技術大学院大学 情報科学研究科
School of Information Science, Japan Advanced Institute of Science and Technology, Asahidai, Nomi, Ishikawa, 923–1292 Japan
[a)] shishido@cy2sec.comm.eng.osaka-u.ac.jp

比較やインターバルテストが実現される．2007 年に西出らは線形秘密分散ベースの定数ラウンド秘匿インターバルテスト [6] を提案した．また 2018 年に森田らは線形秘密分散ベースの大小比較プロトコル [5] を提案した．SCIS2019 において，森田らはその方式を拡張した線形秘密分散ベースの定数ラウンド秘匿インターバルテストを提案している．これらの方式は $N$ 台のサーバーと $t$ 人のクライアントから構成される計算モデルをベースとし，算術的な秘密分散をビット表現された秘密分散に変換するビット分解と呼ばれる技術を利用しないことが特徴である．2008 年に Peng らは Paillier 暗号を用いてクライアントとサーバーの二者間秘匿インターバルテスト [7] を提案している．この方式は「$m \in [0,d) \Longrightarrow m\%d = m$」という性質を用いて効率的なインターバルテストを実現しているが，高い偽陽性を有する．本研究ではビット処理を集合演算に変換して大小比較を実現する 0/1 encoding [2] と呼ばれる技術を用いることで，クライアントとサーバーの二者間秘匿インターバルテストを提案する．

### 1.3 貢献

0/1 encoding は 2 つの整数の大小比較を 2 つの集合の空集合判定を行うことで実現する．本研究は空集合判定が積集合計算の特殊ケースであることに着目し，宮地らが提案した Bloom Filter ベースの Private Set Intersection(PSI) [4] を拡張して二者間の素集合判定プロトコルを提案した．提案する二者間の素集合判定プロトコルを用いてインターバルテストを構築することにより，オンライン時におけるクライアントとサーバーの計算量と通信量が定数となる．

## 2. 準備

### 2.1 DDH 仮定

$\mathbb{F}_p$ を体，$g \in \mathbb{F}_p$ を素数位数 $q$ の元とする．$x, y, z \xleftarrow{\$} \mathbb{Z}_q^*$ を一様分布に従って独立にサンプルされた元とする．DDH 仮定とは与えられた DDH tuple $(g, g^x, g^y, g^z)$ が $(g, g^x, g^y, g^{xy})$ か $(g, g^x, g^y, g^z)$ を識別できる確率的多項式時間アルゴリズム $\mathcal{A}$ が存在しない，という仮定である．

### 2.2 0/1 encoding

2005 年に Lin，Tzeng はヤオのお金持ち問題 [8] を解くために 0/1 encoding [2] を提案した．0/1 encoding は 2 つの整数値 $a, b \in \{0,1\}^\ell$ の大小比較を 2 つの集合の共通部分が存在するかどうか確認することで実現する．整数 $s \in \{0,1\}^\ell$ の 2 進表現を $s = s_1 s_2 \cdots s_\ell$ とする．また $s$ に対して，はじめ $h$ ビットの値 $s_1 s_2 \cdots s_h$ を $s$ の $h$–length prefix string とする．

**定義 2.1** (prefix string set $P_s$)**.** *$s$ の prefix string set $P_s$* は次式で定義する．

$$P_s = \{s_1 s_2 \cdots s_h \mid 1 \le h \le \ell\}.$$

**定義 2.2** (0–encoding set $S_s^0$)**.** *prefix string $s_1 s_2 \cdots s_h$* に対して *0–encoding set $S_s^0$* を次式で定義する．

$$S_s^0 = \{s_1 s_2 \cdots s_{h-1} 1 \mid s_h = 0, 1 \le h \le \ell\}.$$

**定義 2.3** (1–encoding set $S_s^1$)**.** *prefix string $s_1 s_2 \cdots s_h$* に対して *1–encoding set $S_s^1$* を次式で定義する．

$$S_s^1 = \{s_1 s_2 \cdots s_{h-1} s_h \mid s_h = 1, 1 \le h \le \ell\}.$$

$a, b \in \{0,1\}^\ell$ に対し，0–encoding set $S_s^0$ と 1–encoding set $S_s^1$ は次の性質を満たす．

**定理 2.1.** *2 つの整数値 $a, b \in \{0,1\}^\ell$ とそれらに対応する 0–encoding set，1–encoding set* に対して次の必要十分条件が成立する．

$$a > b \Longleftrightarrow S_a^1 \cap S_b^0 \neq \phi,\ a \le b \Longleftrightarrow S_a^1 \cap S_b^0 = \phi.$$

定理 2.1 の証明は [2] を参照されたい．$S_a^1 \cap S_b^0 \neq \phi$ が成立するとき，共通要素は次式のようにかける．

$$a_1 a_2 \cdots a_{h-1} 1|_{a_h=1} = b_1 b_2 \cdots b_{h-1} 1|_{b_h=0}.$$

この式の $h$ 番目の値を 0 としても等式は変わらず成立する．

$$a_1 a_2 \cdots a_{h-1} 0|_{a_h=1} = b_1 b_2 \cdots b_{h-1} 0|_{b_h=0}.$$

定義 2.1 から $P_b = \{b_1 b_2 \cdots b_h \mid 1 \le h \le \ell\}$ なので，$b_1 b_2 \cdots b_{h-1} 0|_{b_h=0} \in P_b$ となる．ここで新たに new 1–encoding set $\tilde{S}_s^1$ を次式で定義する．

**定義 2.4** (new 1–encoding set $\tilde{S}_s^1$)**.**

$$\tilde{S}_s^1 = \{s_1 s_2 \cdots s_{h-1} 0 \mid s_h = 1, 1 \le h \le \ell\}.$$

new 1–encoding set $\tilde{S}_s^1$ により，$S_a^1 \cap S_b^0 = \tilde{S}_a^1 \cap P_b$ が成立する．つまり，$\tilde{S}_s^1$ と $P_s$ を用いて定理 2.1 を別の形に変形することができる．

**定理 2.2.** *2 つの整数値 $a, b$ とそれらに対応する prefix encoding set，0–encoding set，new 1–encoding set* に対して次の必要十分条件が成立する．

$$a > b \Longleftrightarrow \tilde{S}_a^1 \cap P_b \neq \phi \ or\ P_a \cap S_b^0 \neq \phi,$$
$$a \le b \Longleftrightarrow \tilde{S}_a^1 \cap P_b = \phi \ or\ P_a \cap S_b^0 = \phi.$$

定理 2.2 より，整数値 $d$ とインターバル $[w_L, w_H]$ に対して $d \in [w_L, w_H]$ が成立するとき次の命題が成立する．

$$d \notin [w_L, w_H] \Longleftrightarrow d < w_L \ and\ d > w_H$$
$$\Longleftrightarrow P_d \cap \tilde{S}_{w_L}^1 \neq \phi \ and\ P_d \cap S_{w_H}^0 \neq \phi.$$

### 2.3 Bloom Filter

1970 年に Bloom が提案した空間効率の良い確率的なデータ構造 [1] であり，ある要素が集合 $S$ に含まれているかどうか確認するために利用される．Bloom Filter は集合の要素を長さ $m$ のブール型配列に格納する生成アルゴリズムと要素が集合に含まれているかどうか確認する検証アルゴリズムで構成される．生成アルゴリズムと検証アルゴリズムでは共通の $k$ 個の独立したハッシュ関数が利用される．検証アルゴリズムは $\mathcal{O}(1)$ で要素が集合に含まれるか確認することができる．しかし，$y \notin S$ を満たす要素に対して，誤って $y \in S$ と結果を出す偽陽性を有する．一方で $x \in S$ を満たす要素に対して，必ず $x \in S$ と結果を出す．つまり，偽陰性は有さない．

Bloom Filter の具体的なアルゴリズムについて説明する．ここから，$w$ 個の要素を持つ集合 $S = \{s_1, \cdots, s_w\}$ に対する処理を前提とする．Bloom filter の生成アルゴリズムを Algorithm 1 に示す．はじめに $m$ ビットの配列全てに 0 を格納する．要素 $s_i \in S$ に対して $k$ 個のハッシュ関数 $h_j(s_i)$ $(1 \leq j \leq k)$ を計算し，$\mathsf{BF}[h_j(s_i)]$ に 1 を格納する．ハッシュ関数の計算結果が指す配列のインデックスに 1 が格納されている場合はその値を保持する．

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

**Require:** A set $S$, $k$ hash function $\mathcal{H} = \{h_1, \cdots, h_k\}$
**Ensure:** A Bloom filter $\mathsf{BF}$

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

要素の検証アルゴリズムを Algorithm 2 に示す．要素 $x$ が集合 $S$ に属しているかどうか確認するためには $\mathsf{BF}[h_j(x)]$ $(1 \leq j \leq k)$ の配列すべてに 1 が格納されているかどうか検証する．

入っていれば $x \in S$ を出力する．なお，$x \in S$ であれば常に $x \in S$ を出力する．一方で，$y \notin S$ でも $\forall j \in [1, k]$ に対して $\mathsf{BF}[h_j(y)] = 1$ となる偽陽性が発生する．偽陽性発生率はハッシュ関数がランダムである仮定の下で計算される．偽陽性の発生率 $\sigma$ は集合に入っていない要素の $k$ 個のハッシュ値が指す配列に 1 が格納されている確率を考えれば良い．これは $w$ 個の要素が追加された $m$ ビットの配列から 1 が格納されている配列を $k$ 個選ぶ確率なので次のようになる [3].

$$\sigma = \left(1 - (1 - \frac{1}{m})^{kw}\right)^k \approx \left(1 - e^{-\frac{kw}{m}}\right)^k.$$

Bloom filter のサイズ $m$ と集合サイズ $w$ が与えられたとき，ハッシュ関数の個数が増えるとあるビットが 1 になる確率が増えるため偽陽性発生率が増える．逆にハッシュ関数が減ると偽陽性発生率が減る．偽陽性の発生率を最小にするために最適化すると，最小の偽陽性の発生率は $\sigma = (1/2)^k \approx (0.6185)^{m/w}$ となる．ある偽陽性の発生率 $\epsilon$ を満たすハッシュ関数の個数は $k = \epsilon / \ln(1/2)$ で与えられる．同様の式変形を行うと，ある偽陽性の発生率 $\epsilon$ を満たす Bloom filter のサイズは $m = -(w \ln(\epsilon))/(\ln(2))^2$ となる．

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

**Require:** A Bloom filter $\mathsf{BF}$, $x$, $k$ hash function $\mathcal{H} = \{h_1, \cdots, h_k\}$
**Ensure:** True if $x \in S$, False otherwise

```
for i = 0 to k − 1 do
    j = h_i(x)
    if BF[j] = 0 then
        return False
    end if
end for
return True
```

### 2.4 Exponential ElGamal 暗号

Exponential ElGamal 暗号は離散対数問題に基づく IND-CPA 安全な加法準同型暗号である．鍵生成アルゴリズム，暗号化アルゴリズム，復号アルゴリズムの 3 つ組のアルゴリズムで構成される．

- 鍵生成アルゴリズム $\mathsf{KeyGen}(1^\lambda) \to (params, sk, pk)$
  セキュリティパラメータ $1^\lambda$ を入力し，公開パラメータ $params$，秘密鍵 $sk$，公開鍵 $pk$ を生成する．はじめに公開パラメータを生成するために，体 $\mathbb{F}_p$ から素数位数 $q$ のベースポイント $g \in \mathbb{F}_p$ を選択し，公開パラメータ $params = (p, q, g)$ とする．秘密鍵として $sk \leftarrow \mathbb{F}_q$ を一様分布に従いサンプルし，公開鍵 $pk = g^{sk} \bmod p$ を計算する．公開パラメータと公開鍵は公開し，秘密鍵は秘密裏に管理する．
- 暗号化アルゴリズム $\mathsf{Enc}(pk, m) \to (u, v)$
  公開鍵 $pk$ と暗号化の対象である平文 $m \in \mathbb{F}_q$ を入力し，暗号文 $(u, v)$ を出力する．暗号化アルゴリズムは乱数 $r$ を $\mathbb{F}_q$ 上から一様分布に従い独立にサンプルし，$u = g^r \bmod p$ と $v = g^m \cdot pk^r \bmod p$ を計算する．最後に $(u, v)$ を暗号文として出力する．
- 復号アルゴリズム $\mathsf{Dec}(x, (u, v)) \to g^m \bmod p$

暗号化に利用した公開鍵 $pk$ と対となる秘密鍵 $sk$ と暗号文 $(u, v)$ を入力し，復号結果 $g^m \bmod p$ を出力する．復号アルゴリズムは $g^m \leftarrow v \cdot u^{-sk} \bmod p$ を計算する．最後に復号結果として $g^m \bmod p$ を出力する．

Exponential ElGamal 暗号の復号結果は $g^m \bmod p$ の形で得られるため，平文 $m$ を得ることは離散対数問題を解くしかない．つまり，復号結果から平文を復元することは困難であることに注意されたい．また，本研究では公開鍵を $pk = g^{-sk} \bmod p$ を計算して求めることで，復号アルゴリズムにて逆元計算を不要にしている．

## 3. 関連研究

本節では関連研究として Lin らが提案した 0/1 encording ベースの二者間大小比較プロトコル [2] と Peng らが提案した二者間インターバルテスト [7] を説明する．

### 3.1 0/1 encoding ベースの二者間大小比較プロトコル

2005 年に Lin らが提案した 0/1 encoding ベースの二者間大小比較プロトコル [2] について説明する．このプロトコルはアリスとボブが持つ整数 $x, y \in \{0,1\}^\ell$ の大小比較を互いに $x, y$ を秘匿しながら行う．アリスとボブはそれぞれ $x, y$ に対して 0/1 encoding を適用して $S_x^1$ と $S_y^0$ を計算する．アリスは乗法準同型暗号である Exponential ElGamal 暗号の設定を行い，秘密鍵と公開鍵を得る．アリスは次の手順に従い $2 \times \ell$ の行列 $T[i, j]$ を生成する．ここで $i, j$ の定義域はそれぞれ $i \in \{0,1\}$，$j \in [1, \ell]$ である．アリスの入力を $x = x_1 x_2 \cdots x_\ell \in \{0,1\}^\ell$ とし，$\overline{x}_j$ は $\overline{x}_j = 1 - x_j$ と定義する．

・$2 \times \ell$ 行列の生成手順

(1) 乱数 $r_j \in \mathbb{F}_q$ をサンプルする．

(2) $T[x_j, j] \leftarrow \mathsf{Enc}(pk, 0), T[\overline{x}_j, j] \leftarrow \mathsf{Enc}(pk, r_j)$ を実行する．

アリスは $2\ell$ 回の暗号化によって生成した $2 \times \ell$ の行列 $T[i, j]$ をボブに送信する．ボブはすべての $t = t_1 t_2 \cdots t_\beta \in S_y^0$ に対して，$c_t = T[t_1, 1] \cdot T[t_2, 2] \cdots T[t_\beta, \beta]$ を計算する．また $n = \ell - |S_y^0|$ 個のランダムな暗号文 $c'_1, \cdots, c'_n$ を生成し，計 $\ell$ 個の暗号文 $c_1, \cdots, c_{|S_y^0|}, c'_1, \cdots, c'_n$ をアリスに送信する．ここで $S_x^1 \cap S_y^0 \neq \phi$ を満たすとき，$\exists t \in S_y^0 \ s.t. \ t \in S_x^1$ が成立する．つまり，ボブが生成した $\ell$ 個の暗号文の中に $c_t = \mathsf{Enc}(pk, 0) \cdots \mathsf{Enc}(pk, 0) = \mathsf{Enc}(pk, 0)$ を満たすものが存在する．したがって，アリスは $\ell$ 個の暗号文をすべて復号し，$\mathsf{Dec}(sk, c_t) = g^0 \bmod p$ となる暗号文があれば $x > y$，なければ $x \leq y$ の大小比較結果を得る．

### 3.2 Peng らが提案した二者間インターバルテスト

2008 年に Peng らが提案した二者間インターバルテスト [7] の特徴を説明する．提案方式は加法準同型暗号として知られている Paillier 暗号を用いて構成され，honest–but–curious な敵対者に対して安全なプロトコルである．クライアントは Paillier 暗号の秘密鍵と公開鍵を生成し，サーバーが持つ暗号化された平文 $\mathsf{Enc}(y, m)$ が公開されているインターバル $I = [0, d)$ に含まれているかどうか検証する．クライアントはインターバル $I$ に平文 $m$ が含まれているかどうか結果を得ることができるが，平文 $m$ を得ることができない．提案手法は Paillier 暗号の加法準同型性を利用して任意のインターバルに対して検証を行うことができ，計算量がインターバルに依存しない．しかし，Paillier 暗号で使用する合成数を $n$ とすると，提案方式は $n - 2d + 1 \leq m \leq 3d - 2$ の範囲の平文に対して $m \in I$ という結果を出力する偽陽性を有する．この偽陽性は Bloom Filter のようにパラメータを適切に設定することで下げることができず，高い確率で偽陽性が発生する．

## 4. 二者間秘匿インターバルテスト

### 4.1 二者間秘匿インターバルテストの概要

本研究は素集合判定を利用した二者間の秘匿インターバルテストを提案する．クライアントは整数 $x \in \{0,1\}^\ell$ を持ち，サーバーはインターバル $I = [w_L, w_H] \in \{0,1\}^\ell \times \{0,1\}^\ell$ を持つ．整数 $x$ 及びインターバル $I$ はそれぞれ秘密裏に管理されている．クライアントとサーバーは 2.2 節で説明した 0/1 encoding を適用し，集合 $C \leftarrow P_x$ と $S \leftarrow \tilde{S}_{w_L}^1 \cup S_{w_H}^0$ を生成する．2 つの集合に対して素集合判定を行い，クライアントは $C, S$ が素集合かどうか判定結果を受け取る．一方で，サーバーはプロトコルから結果を得ることができない．定理 2.2 から $C \cap S \neq \phi \iff x \notin I$ が成立するので，素集合判定の結果が $True$，つまり $C \cap S \neq \phi$ ならば $x \notin I$，$False$ ならば $x \in I$ というインターバルテストの結果を得る．敵対者は honest–but–curious model を仮定する．宮地らが提案した Private Set Intersection(PSI) [4] の構成手法を直接用いた方式 (primitive な方式) ではインターバルテストの結果からサーバーが秘密裏に管理するインターバル $I$ の情報が漏れる問題が起こる．本研究では素集合判定が PSI の特殊ケースであることに着目し，二者間の素集合判定プロトコルを構築する．構築した素集合判定プロトコルを用いて，情報が漏れる問題を解決すると同時に，オンラインのクライアントの計算量と通信量を定数に削減した方式を構築する．

### 4.2 二者間秘匿計算プロトコルの安全性

本研究が提案する二者間秘匿インターバルテストの安全性について述べる．本研究はエンティティとして honest–but–curious なクライアントとサーバーの二者を仮定し，クライアントは整数 $x \in \{0,1\}^\ell$，サーバーは $I \in \{0,1\}^\ell \times \{0,1\}^\ell$ を入力する．二者間秘匿インターバルテストにおいて，クライアントは「$x$ がインターバル $I$ の範囲に入っているかどうか」という結果を得ることができ，それ以外の情報を

得ることができない．一方で，サーバーは何も情報を得ることができない．この観点からクライアントプライバシーとサーバープライバシーを次のように定義する．

**定義 4.1** (二者間秘匿インターバルテストの安全性)．二者間秘匿インターバルテストは次の条件を満たすとき安全性を満たす．

*(1) Client Privacy I(CP I):* クライアントから送信された前処理データ列が乱数列と区別できない．

*(2) Client Privacy II(CP II):* クライアントから送信された前処理データ列を得たサーバーは自身の知識を利用しても，乱数列から得られる情報と区別できない．

*(3) Server Privacy I(SP I):* サーバーから送信されたデータが乱数と区別できない．

*(4) Server Privacy II(SP II):* サーバーから送信されたデータを得たクライアントが自身の知識を利用しても，$x \in I$ か　$x \notin I$ の情報以外の情報を入手できない．

### 4.3 primitive な方式

2017 年に宮地らが提案した PSI [4] を用いた二者間インターバルテストについて説明する．

- 初期設定
  (1) クライアントとサーバーはそれぞれ自身の入力 $x \in \{0,1\}^\ell$ と $I = [w_L, w_H] \in \{0,1\}^\ell \times \{0,1\}^\ell$ に 0/1 encoding を適用して集合 $C, S$ を生成する．
  (2) クライアントは Exponential ElGamal 暗号の鍵生成アルゴリズム $\mathsf{KeyGen}(1^\lambda)$ を実行して $params, sk, pk$ を得る．
  (3) クライアントとサーバーは自身の集合 $C, S$ から Bloom Filter $\mathsf{BF}_C, \mathsf{BF}_S$ を生成する．
- 手順
  (1) クライアントは $\mathsf{BF}_C = [1 - \mathsf{BF}_C[0], \cdots, 1 - \mathsf{BF}_C[m-1]]$ を配列の要素毎に暗号化して，$\mathsf{Enc}(pk, \mathsf{BF}_C) = [\mathsf{Enc}(pk, 1 - \mathsf{BF}_C[0]), \cdots, \mathsf{Enc}(pk, 1 - \mathsf{BF}_C[m-1]])$ を得る．クライアントは $\mathsf{Enc}(pk, \mathsf{BF}_C)$ をサーバーに送信する．
  (2) サーバーは $\mathsf{BF}_S = [1 - \mathsf{BF}_S[0], \cdots, 1 - \mathsf{BF}_S[m-1]]$ を配列の要素毎に暗号化して，$\mathsf{Enc}(pk, \mathsf{BF}_S) = [\mathsf{Enc}(pk, 1 - \mathsf{BF}_S[0]), \cdots, \mathsf{Enc}(pk, 1 - \mathsf{BF}_S[m-1]])$ を得る．サーバーは $\mathsf{Enc}(pk, \mathsf{BF}_C)$ と $\mathsf{Enc}(pk, \mathsf{BF}_S)$ と独立かつ一様ランダムにサンプルされた乱数 $r_0, \cdots, r_{m-1} \in (\mathbb{Z}_q^*)^m$ から下記の式を計算して暗号化された統合 Bloom Filter IBF を求め，サーバーへ IBF を送信する．

$$\mathsf{Enc}(pk, \mathsf{IBF}[i]) = (\mathsf{Enc}(pk, \mathsf{BF}_C[i]) \cdot \mathsf{Enc}(pk, \mathsf{BF}_S[i]))^{r_i}.$$

  (3) クライアントは $\mathsf{Enc}(pk, \mathsf{IBF})$ を復号し，$\mathsf{IBF} = [\mathsf{IBF}[0], \cdots, \mathsf{IBF}[m-1]]$ を得る．得られた統合 Bloom Filter に対して $\forall x \in C$ のメンバーシップ検証を行う．$C \cap S \neq \phi$ のとき $x \notin I$，それ以外のとき $x \in I$ を得る．

$$result = \begin{cases} x \notin I & (C \cap S \neq \phi), \\ x \in I & (C \cap S = \phi). \end{cases}$$

### 4.4 提案方式

- 初期設定
  (1) クライアントとサーバーはそれぞれ自身の入力 $x \in \{0,1\}^\ell$ と $I = [w_L, w_H] \in \{0,1\}^\ell \times \{0,1\}^\ell$ に 0/1 encoding を適用して集合 $C, S$ を生成する．
  (2) クライアントは Exponential ElGamal 暗号の鍵生成アルゴリズム $\mathsf{KeyGen}(1^\lambda)$ を実行して $params, sk, pk$ を得る．
  (3) クライアントとサーバーは自身の集合 $C, S$ から Bloom Filter $\mathsf{BF}_C, \mathsf{BF}_S$ を生成する．
- 手順
  (1) クライアントは $\mathsf{BF}_C = [\mathsf{BF}_C[0], \cdots, \mathsf{BF}_C[m-1]]$ を配列の要素毎に暗号化して，$\mathsf{Enc}(pk, \mathsf{BF}_C) = [\mathsf{Enc}(pk, \mathsf{BF}_C[0]), \cdots, \mathsf{Enc}(pk, \mathsf{BF}_C[m-1]])$ を得る．クライアントは $\mathsf{Enc}(pk, \mathsf{BF}_C)$ をサーバーに送信する．
  (2) サーバーは手に入れた $\mathsf{Enc}(pk, \mathsf{BF}_C)$ と $\mathsf{BF}_S$ と乱数 $r \xleftarrow{\$} \mathbb{Z}_q^*$ を用いて次式の計算し，$c'$ をクライアントへ送信する．

$$c' = (\prod_{i=0}^{m-1} \mathsf{Enc}(pk, \mathsf{BF}_C[i])^{\mathsf{BF}_S[i]})^r.$$

  (3) クライアントは $c'$ を復号し，$\mathsf{Dec}(sx, c') \neq 1$ のとき $x \notin [a, b]$，それ以外のとき $x \in [a, b]$ と結果を得る．

$$result = \begin{cases} x \notin I & (\mathsf{Dec}(sk, c') \neq 1), \\ x \in I & (Otherwise). \end{cases}$$

#### 4.4.1 正当性

**定理 4.1** (提案方式の正当性)．本研究が提案する二者間秘匿インターバルテストは正当性を満たす．つまり，$x \notin I$ のとき提案プロトコルは $x \notin I$ を出力し，それ以外のとき $x \in I$ を出力する．

*Proof.* サーバーの入力を $I = [w_L, w_H]$ とし，クライアントの入力を $x \notin I$ と仮定する．仮定からクライアントとサーバーが 0/1 encoding を適用して生成した集合 $C$ とサーバーの集合 $S$ は定理 2.2 から $x \notin I \iff C \cap S \neq \phi$ を満たす．提案プロトコルは加法準同型暗号である Exponential ElGamal 暗号で暗号化する．したがって，クライアントの Bloom Filter の暗号文 $\mathsf{Enc}(pk, \mathsf{BF}_C)$ は次式で表される．

$$\mathsf{Enc}(pk, \mathsf{BF}_C) = [\cdots, (g^{r_i} \bmod p, g^{\mathsf{BF}_C[i]} pk^{r_i} \bmod p), \cdots].$$

加法準同型性より，クライアントが復号する暗号文は $c' = (g^r \sum_{i=1}^{m-1} r_i \bmod p, g^r \sum_{i=0}^{m-1} \mathsf{BF}_C[i]\cdot\mathsf{BF}_S[i] \bmod p)$ とかける．$C \cap S \neq \phi$ を満たすので，$\sum_{i=0}^{m-1} \mathsf{BF}_C[i]\cdot\mathsf{BF}_S[i] \neq 0$ を満たす．つまり，入力された値が $x \notin I$ を満たすとき，復号結果は常に $\mathsf{Dec}(x, c') \neq g^0$ となる．したがって，提案プロトコルは $x \notin I$ を出力するので正当性を満たす． □

#### 4.4.2 安全性

**定理 4.2** (提案方式の安全性)．本研究が提案する二者間秘匿インターバルテストは *DDH* 仮定の下で *honest–but–cuirous* な敵対者に対して安全な二者間秘匿計算プロトコルである．

*Proof.* 定義 4.1 で示した内容を示すことで，提案方式の安全性を証明する．

- CP I の証明:
  $(\mathsf{Enc}(pk, \mathsf{BF}_C[0]), \cdots, \mathsf{Enc}(pk, \mathsf{BF}_C[m-1]))$ と $(r_0, \cdots, r_{m-1})$ を確率 $\epsilon$ で識別できる確率的多項式時間アルゴリズム $D$ を仮定する．ここで確率的多項式時間アルゴリズムは DDH 問題を解くシミュレーター SIM を下記の手順で構成する．
  (1) SIM は DDH tuple $(g, g^\alpha, g^\alpha, g^\zeta)$ を受け取る．
  (2) SIM は一様分布に従って $\mathbb{F}_p$ から独立に $m$ 個の乱数 $\alpha_0, \cdots, \alpha_{m-1}$ と $m-1$ 個の乱数 $\gamma_1, \cdots, \gamma_{m-1}$ をサンプルする．
  (3) $\delta = ((g^\alpha, \alpha_0 g^\zeta), ((g^\alpha)^{\gamma_1}, \alpha_1 (g^\zeta)^{\gamma_1}), \cdots, ((g^\alpha)^{\gamma_{m-1}}, \alpha_1 (g^\zeta)^{\gamma_{m-1}}))$ を確率的多項式時間アルゴリズム $D$ に送信し，結果を受け取る．
  DDH tuple が $z = xy$ を満たすとき，$\zeta$ は $(\mathsf{Enc}(pk, \mathsf{BF}_C[0]), \cdots, \mathsf{Enc}(pk, \mathsf{BF}_C[m-1])$ の分布と一致する．したがって，$D$ は確率 $\epsilon$ で 1 を出力する．一方で，DDH tuple が $z \neq xy$ を満たすとき，$\zeta$ は $(\mathsf{Enc}(pk, \mathsf{BF}_C[0]), \cdots, \mathsf{Enc}(pk, \mathsf{BF}_C[m-1])$ の分布と異なるので，アルゴリズム $D$ は確率 $1/2$ で正しい結果を返す．よって，確率的多項式時間アルゴリズムは構築した SIM シミュレーター SIM を使って DDH 問題を確率 $1/2+\epsilon$ の確率で解くことができる．したがって，DDH 仮定の下で $(\mathsf{Enc}(pk, \mathsf{BF}_C[0]), \cdots, \mathsf{Enc}(pk, \mathsf{BF}_C[m-1])$ は任意の確率的多項式時間アルゴリズムに対して $(r_0, \cdots, r_{m-1})$ と識別が不可能である．Exponential ElGamal 暗号で暗号化された Bloom Filter は乱数列と区別できないため，提案方式は CPI を満足する．
- CP II の証明:
  サーバーが受け取るデータ列はすべて Exponential ElGamal 暗号で暗号化されている．上記の証明から任意の確率的多項式時間アルゴリズムは暗号化された Bloom Filter と長さ $m$ の乱数列を区別することができない．また公開鍵に対応する秘密鍵を持たないため，暗号文を復号することもできない．したがって受け取ったデータから得られる情報はない．
- SP I の証明:
  サーバーがクライアントへ送信するデータはサーバーによってランダマイズされた Exponential ElGamal 暗号で暗号文である．したがって，任意の確率的多項式時間アルゴリズムは暗号文と乱数を区別することができない．
- SP II の証明:
  クライアントはサーバーから受け取った暗号文 $(u, v)$ を復号して $g^d \bmod p$ を得る．正当性の証明から $x \notin I$ のとき $d = 0$ となり，$x \in I$ のとき $d$ は乱数となる．クライアントは Bloom Filter の元データを持つが，暗号文 $(u, v)$ はサーバーによってランダマイズされている．したがって，クライアントはインターバルテストの結果 $x \in I$，または，$x \notin I$ のみ得ることができ，サーバーの秘密インターバル $I$ の情報を得ることができない．

□

## 5. 評価

### 5.1 素集合の判定に適した Bloom Filter の設定

本研究では Bloom Filter を 2 つの集合の素集合判定に利用する．2 つの集合 $S_1, S_2$ に共通要素が存在するとき，$\forall j \in [1,k], \exists x \in S_2 \ s.t.\ \mathsf{BF}_{S_1}[h_j(x)] = 1$ が成立する．したがって，2 つの集合 $S_1, S_2$ に対応する Bloom Filter を $\mathsf{BF}_{S_1}, \mathsf{BF}_{S_2}$ とすると，$\mathsf{BF}_{S_1}, \mathsf{BF}_{S_2}$ の配列の要素毎に論理積を計算すれば，$S_1 \cap S_2$ を求めることができる．つまり補題 5.1 が成立する．

**補題 5.1.**

$$S_1 \cap S_2 \neq \phi \Rightarrow \exists i \in [0, m) \ s.t.\ \mathsf{BF}_{S_1}[i] \wedge \mathsf{BF}_{S_2}[i] = 1.$$

提案方式では 2 つの集合の素集合判定に補題 5.1 を利用する．2 つの集合に共通要素が存在すること，つまり，$S_1 \cap S_2 \neq \phi$ を仮定すると常に「素集合でない」と判定する．一方で，$S_1 \cap S_2 = \phi$ を満たす集合に対して「素集合でない」となる偽陽性が発生する．素集合判定の偽陽性発生率 $\mu$ は Bloom Filter の偽陽性発生率 $\sigma$ から算出できる．集合 $S$ の要素数を $w$，ハッシュ関数の個数を $k$，Bloom Filter の長さを $m$ とする．$w$ 個の要素を $k$ 個のハッシュ関数を利用して長さ $m$ の Bloom Filter を構築したとき，$y \notin S$ に対して $y \in S$ と結果を出す確率は $\sigma = \left(1 - (1 - \frac{1}{m})^{kw}\right)^k$ で与えられる．これは $y \notin S$ と $\forall j \in [1,k]$ に対して $\mathsf{BF}_S[h_j(y)] = 1$ となる確率である．

素集合判定の偽陽性発生率を議論するために，$|S_1| = w_1$，$|S_2| = w_2$ かつ $S_1 \cap S_2 = \phi$ を満たす集合を $S_1, S_2$ とする．集合 $S_1$ に対応する $\mathsf{BF}_{S_1}$ に対して $\forall y \in S_2$ のメンバーシップ検証を考える．$S_1 \cap S_2 = \phi$ から $\forall y \in S_2$ に対するメン

バーシップ検証において，少なくとも 1 度でも $y \in S_1$ と結果を出力すれば「素集合でない」と結果を出力することになる．したがって，$S_1, S_2$ に対して補題 5.1 を利用した素集合判定手法が素集合でないと結果を出力する確率 $\mu$ は次式で与えられる．

$$\mu = 1 - (1-\sigma)^{w_2} = 1 - \Big(1 - \big(1 - (1 - \frac{1}{m})^{kw_1}\big)^k\Big)^{w_2}.$$

ハッシュ関数の個数が増えると Bloom Filter のあるビットが 1 になる確率が高くなるため，素集合判定の偽陽性確率 $\mu$ が大きくなる．提案方式で利用する Bloom Filter は素集合判定に関する偽陽性発生確率を低くするために，使用するハッシュ関数の個数を 1 とする．$k = 1$ のとき，素集合判定の偽陽性発生率は次式となる．

$$\mu = 1 - (1 - \frac{1}{m})^{w_1 w_2}.$$

上記の式で与えられる素集合判定の偽陽性発生率を整理すると，Bloom Filter のサイズは次式で与えられる．

$$m = \frac{1}{1 - (1-\mu)^{1/(w_1 w_2)}}.$$

素集合判定の偽陽性の理論値の有意性を示すために，実験から算出した偽陽性発生率と比較する．使用する Bloom Filter のサイズ $m$ は素集合判定の偽陽性発生率 $\mu$ と $x \in \{0,1\}^\ell$ の prefix string $P_x$ のサイズ $|P_x|$ とインターバル $I = [a, b] \in \{0,1\}^\ell \times \{0,1\}^\ell$ に対応する 0/1 encoding の集合 $\tilde{S_a^1} \cup \tilde{S_b^0}$ のサイズ $|\tilde{S}_a^1 \cup S_b^0|$ に依存する．実験で使用する Bloom Filter のサイズ $m$ は次式を計算して定めた．

$$m = \frac{1}{1 - (1-\mu)^{1/|P_x| \cdot |\tilde{S}_a^1 \cup S_b^0|}}.$$

本研究では $\ell = 16, 32, 64$ に対して $x \in I$ を満たす整数 $x \in \{0,1\}^\ell$ と $I \in \{0,1\}^\ell \times \{0,1\}^\ell$ をランダムに生成し，$P_x \cap (\tilde{S}_a^1 \cup S_b^0) = \phi$ を満たす集合 $P_x, \tilde{S}_a^1 \cup S_b^0$ の素集合判定を行った．この処理を $10^4$ 回繰り返し，失敗回数をカウントして偽陽性発生率を求めた．

**表 1** $w_2 = |S_I|$ とした偽陽性発生率

| $\ell$ | 16 | 32 | 64 |
|---|---|---|---|
| $\mu = 0.2$ | 0.1945 | 0.1962 | 0.1924 |
| $\mu = 0.1$ | 0.104 | 0.1048 | 0.0976 |
| $\mu = 0.05$ | 0.0515 | 0.0515 | 0.0521 |

実験結果を表 1 に示す．表 1 からわかるように，実験から算出した偽陽性発生率が前もって設定した偽陽性確率と誤差 5%程度になる．したがって，5.1 節で示した理論的な偽陽性発生率の有意性が示された．

### 5.2 提案方式で利用する Bloom Filter の設計

本研究が提案する二者間の秘匿インターバルテストではクライアントがテスト実行時に Bloom Filter を構築する必要があり，素集合判定の対象である 2 つの集合サイズが必須となる．しかし，サーバーは安全性を達成するために $|S_I| = |\tilde{S}_a^1 \cup S_b^0|$ をクライアントに対して秘匿しなければいけない．したがって，クライアントはサーバーの集合情報を使わずにサーバーの集合サイズ $|S_I|$ を前もって決めることで Bloom Filter を設計する必要がある．

**補題 5.2.** $\ell$ が与えられたとき，$x$ の *prefix string set* $P_x$ のサイズは $|P_x| = \ell$ となる．

補題 5.2 により，$\omega_1 = \ell$ と設定する．$|S_I| \in [1, 2\ell - 2]$ に対して，$\omega_2 = |S_I|$ となる集合は $S_I$ は一意的に決まらない．このため，集合数を事前に与えることが直接的に範囲 $I$ を決定するとはいえない．しかし，$|S_I|$ は偽陽性発生率に影響を与えることから，理論的に考慮して決定する必要がある．$\omega_2$ の与え方としては 2 通りの考え方がある．1 つは，サーバが $I$ の値を鑑みて決定する．$I$ に依存せずに，$\omega_2$ を決める．以下ではこれらを決定するための解析を行う．

**補題 5.3.** $\ell$ が与えられたとき，$w_2 = |S_I|$ とする．

- $\omega_2 = 2\ell - 2$ となるのは，範囲 $I = [\sum_{i=0}^{\ell-2} 2^i, 2^{\ell-1}]$ の時で *1* 個ある．
- $\omega_2 = 1$ となるのは範囲 $I = [0, \sum_{i=0}^{\ell-2} 2^i], [0, \sum_{i=0}^{\ell-3} 2^i + 2^{\ell-1}], \cdots, [0, \sum_{i=1}^{\ell-1} 2^i]$,$[2^0, 2^\ell - 1], [2^1, 2^\ell - 1], \cdots,$ $[2^{\ell-1}, 2^\ell - 1]$ の時で $2\ell$ 個ある．
- $\omega_2 = \ell$ となるのは範囲は $\sum_{i=0}^{\ell} \binom{\ell}{i}!$ 個ある．

補題 5.3 により，$|S_I|$ が最小値と最大値を取る可能性は非常に小さくなることがわかる．したがって，クライアントは $w_1 = w_2 = \ell$ とおいて Bloom Filter を構築し，プロトコルで利用する．$w_1, w_2 = \ell$ とすることによって素集合判定の偽陽性発生率にどれくらい影響を与えるか検証するために次の実験をした．実験で使用する Bloom Filter のサイズ $m$ は次式を計算して定めた．

$$m = \frac{1}{1 - (1-\mu)^{1/\ell^2}}.$$

本研究では $\ell = 16, 32, 64$ に対して $x \in I$ を満たす整数 $x \in \{0,1\}^\ell$ と $I \in \{0,1\}^\ell \times \{0,1\}^\ell$ をランダムに生成し，$P_x \cap (\tilde{S}_a^1 \cup S_b^0) = \phi$ を満たす集合 $P_x, \tilde{S}_a^1 \cup S_b^0$ の素集合判定を行った．この処理を $10^4$ 回繰り返し，失敗回数をカウントして偽陽性発生率を求めた．表 5.2 に構築した Bloom Filter のサイズ $m$ と偽陽性確率を与える．表 5.2 からわかるように，$w_2 = |\tilde{S}_a^1 \cup S_b^0|$ とおいたときより，事前に設定した偽陽性発生率と実験から算出した偽陽性発生率の差が大きるが，誤差率が 10%程度であるため実用上問題ないと考えられる．したがって，本研究の提案方式では $w_1 = w_2 = \ell$ として構築した Bloom Filter を利用する．

表 2　$w_2 = \ell$ とした Bloom Filter のサイズ $m$ と偽陽性発生率

| $\ell$ | 16 | | 32 | | 64 | |
|---|---|---|---|---|---|---|
| | $m$ | 偽陽性確率 | $m$ | 偽陽性確率 | $m$ | 偽陽性確率 |
| $\mu = 0.2$ | 1148 | 0.1848 | 4589 | 0.2019 | 18356 | 0.1920 |
| $\mu = 0.1$ | 2430 | 0.0951 | 9720 | 0.0977 | 38877 | 0.0946 |
| $\mu = 0.05$ | 4991 | 0.0562 | 19964 | 0.0450 | 79855 | 0.0476 |

表 3　計算量と通信量の比較

| | | 計算量 | | 通信量 |
|---|---|---|---|---|
| | | クライアント | サーバー | |
| [2] | オフライン | $6\ell E$ | - | 上り: $4\ell\lvert\mathbb{F}_p\rvert$ |
| | オンライン | $\ell(E+I)$ | $3(\ell - \lvert S_b^0\rvert)E$ | 下り: $2\ell\lvert\mathbb{F}_p\rvert$ |
| primitive な方式 | オフライン | $3mE$ | - | 上り: $2m\lvert\mathbb{F}_p\rvert$ |
| | オンライン | $m(E+I)$ | $5mE$ | 下り: $2m\lvert\mathbb{F}_p\rvert$ |
| 提案方式 | オフライン | $3mE$ | - | 上り: $2m\lvert\mathbb{F}_p\rvert$ |
| | オンライン | $E$ | $2E$ | 下り: $2\lvert\mathbb{F}_p\rvert$ |

### 5.3　各手法の計算量と通信量の比較

本節では提案方式を Lin らの二者間大小比較プロトコル 3.1 と本研究の primitive な方式と比較する．全手法は Exponential ElGamal 暗号を利用し，honest–but–curious な敵対者に対して安全なプロトコルである．クライアントとサーバーの計算量はサーバーにデータを送信する前の処理 (オフライン) とその後の処理 (オンライン) に分け評価する．計算量は法べき乗算 $E$ と逆元 $I$ の計算回数をカウントする．プロトコルの中で法乗算も行われるが，計算コストが法べき乗算より小さく計算回数も少ないので，法乗算の計算回数は無視する．通信量は二者間でやり取りをする $\mathbb{F}_p$ 上の元の個数とし，クライアントからサーバーの通信を上り，サーバーからクライアントの通信を下りとして評価する．ここでインターバルテストで扱う整数の大きさ $\ell$ と Bloom Filter のサイズ $m$ は常に $\ell < m$ を満たすことに注意されたい．表 3 に各手法の計算量と通信量を示す．

[2] とプリミティブな方式は後処理においてクライアントの計算量は $\ell$ や Bloom Filter のサイズ $m$ に依存している．また，下りの通信量も同様に $\ell$ や $m$ に依存している．一方で，提案方式はオンラインにおいてクライアントの計算量と通信量が定数である．これはクライアントがデータをサーバーへ送信後，すぐに結果を得られるというように解釈できる．

## 6.　まとめ

本研究は 0/1 encoding ベースのインターバルテストを提案した．0/1 encoding は 2 つの集合の素集合判定によって 2 つの整数の大小比較を実現する．本研究は素集合判定が PSI の特殊ケースであることに着目し，PSI を拡張することによって，クライアントの計算量と通信量がオンラインフェーズにおいて定数となる方式を構築した．今後の課題はオフラインのクライアントの計算量を削減することである．

### 参考文献

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

[2] Hsiao-Ying Lin and Wen-Guey Tzeng. An efficient solution to the millionaires' problem based on homomorphic encryption. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 456–466, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

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

[4] Atsuko Miyaji, Kazuhisa Nakasho, and Shohei Nishida. Privacy-preserving integration of medical data. *Journal of Medical Systems*, 41(3):37, Jan 2017.

[5] Hiraku Morita, Nuttapong Attrapadung, Tadanori Teruya, Satsuya Ohata, Koji Nuida, and Goichiro Hanaoka. Constant-round client-aided secure comparison protocol. In *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part II*, pages 395–415, 2018.

[6] Takashi Nishide and Kazuo Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, pages 343–360, 2007.

[7] Kun Peng, Feng Bao, and Ed Dawson. Correct, private, flexible and efficient range test. *Journal of Research and Practice in Information Technology*, 40(4):275–289, 2008.

[8] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.