3　暗号理論の応用
3　Applied Cryptography

3-1　高速秘匿共通集合計算プロトコルの設計方法について
3-1　On the Construction of Fast Secure Set-Intersection Protocols

野島　良
NOJIMA Ryo

要旨

本稿では、秘匿共通集合計算プロトコルについて考える。秘匿共通集合計算プロトコルは、二者間のプロトコルであり、それぞれが保持している集合の共通集合を秘密裏に得ることを可能にする。本稿では、この問題を解決する三つの方式を紹介する。一つ目の方式は、通信計算量の下界にあたる方式、つまり通信量の観点から最適な方式である。二つ目の方式は、フリードマンらが考えた秘匿共通集合計算プロトコルに若干の修正を加えたものであり、特に安全性の高い方式である。三つ目の方式は、近似アルゴリズムを応用しており、時間計算量が非常に小さい。本稿では、これら利点の異なる三つの方式について詳しく述べる。

In this paper, we consider a two-party secure set-intersection protocol. In this protocol, there are two parties, a server and a client, and they have secret-sets Q_S and Q_C, respectively, where $|Q_S| = |Q_C| = n$. The goal of the protocol is the client obtaining $S_A \cap S_B$ while preserving the secret-sets secret. We introduce three protocols which implement this functionality in this paper. In the first protocol, we concentrate on the communication complexity and show that the protocol matches the lower bound. In the second protocol, we design the protocol based on th(e protocol proposed by Freedman et al. Compared to the other efficient protocols, this protocol is more secure and is adequate for the practical use. In the last protocol, we concentrate on reducing the time-complexity. In other existent protocols, at least one party needs to perform O (n log n) operations. However, in our protocol, the time-complexity is reduced to O (n). The essential idea behind the protocol is the employment of the approximation algorithm for the Jaccard's distance. We examine these three protocols in detail in this paper.

[キーワード]
秘匿計算，準同型暗号，共通集合
Secure computation, Additive homomorphic cryptosystems, Set-Intersection

1　はじめに

暗号というとSSHやSSLといった通信の安全を確保するためのものであると考えられがちであるが、本稿では、それとは異なった秘匿計算プロトコルと呼ばれる暗号の斬新な応用方法について述べる。そもそも秘匿計算プロトコルは、2000年にチューリング賞を取ったヤオが提案したものであり、下記の問題提起から始まった[7]。

二人の金持ちアリスとボブが道ばたで出会ったとする。彼らは、どちらが金持ちであるかを知りたい。しかしながら、相手に所持金を教えることだけは避けたい。

もちろん、相手に所持金を教えてもよいなら、この問題は簡単に解決する。ここで重要なことは、相手に所持金を教えずにこれを実現する手段が存在するかどうかである。驚くことにヤオは、暗号関連技術を駆使し、この問題を解いてみせた。本稿で述べる話題は、この問題と密接に関係しており、秘匿共通集合計算プロトコルと呼ばれるものである。考える問題を上と同じように記述すると次のようになる。

アリスとボブは、それぞれ秘密の集合 S_A, S_B を保持している。彼らは、S_A, S_B の共通集合のみを知りたい。しかしながら、相手にそれ以外の一切の集合の要素を教えることは避けたい。

例えば、$S_A = \{1, 345, 787, 88\}$, $S_B = \{9893, 3232, 89, 345\}$ とする。$S_A \cap S_B = \{345\}$ であるため、アリスは $\{1, 787, 88\}$ を漏らさずに、ボブは $\{9893, 3232, 89\}$ を漏らさないように、$S_A \cap S_B = \{345\}$ のみを取得可能にしたい。この問題に対する解決方法は、フリードマンらにより提案された[3]。我々は、このプロトコルを改良し、スピア型攻撃を判定するシステムを設計した[8]。

タイトルにあるとおり、本稿ではこの秘匿共通集合計算プロトコルの高速化について考える。高速な方式を得るためには、もちろんアルゴリズム論的に効率の良い方式を構成することが必要となるが、その際に考えなければならないものとして、時間計算量、通信計算量がある。本稿では、三つの秘匿共通集合計算プロトコルの構成方法を紹介し、それぞれの時間計算量、通信計算量、そして安全性について述べる。三つの方式にはそれぞれ異なる性質がある。通信計算量の観点からすると、方式 1 は、最適な方式となっている。方式 2 は、フリードマンらが考えた方式に若干の修正を加えたものであるが、安全性が格段に向上している。この方式は、三つの方式の中で最も安全な方式となっている。方式 3 は、時間計算量に注目し、他の方式と比べてより高速に動くように設計されている。

2 準備

2.1 加法に関して準同型性を有する公開鍵暗号方式

まず、公開鍵暗号方式に関する簡単な説明を行う。公開鍵暗号方式においては、暗号化する（公開）鍵 pk と復号する（秘密）鍵 sk が異なる。秘密鍵 sk を保有するユーザ（受信者）は pk のみを公開する。メッセージ送信者は pk を使いメッセージを暗号化して、受信者におくる。受信者は sk を使い暗号文を復号し、メッセージを得る。ここで、メッセージ m の暗号文を $\mathrm{Enc}(m)$ と記述することにする。準同型性を有する暗号系においては、秘密鍵 sk なしで、$\mathrm{Enc}(m_1)$, $\mathrm{Enc}(m_2)$ から $\mathrm{Enc}(m_1 + m_2)$ を得ることが可能である。このような性質を持つ暗号系として、例えばパエリア暗号[6]やエルガマル暗号[2]などがある。

2.2 問題設定

アリスとボブが秘密裏に保持している集合を、それぞれ S_A, $S_B \subseteq U = \{1, 2, ..., N\}$ と記述する。ここで U は普遍集合である。また、簡単のため、$|S_A| = |S_B| = n$ とする。

秘匿共通集合計算プロトコルにも様々な形式のものが存在する。本稿では、特に下記の問題について考える。

問題 1

入　　力：アリスの入力 S_A、ボブの入力 S_B
ボブの出力：S_A と S_B の共通集合

問題 2

入　　力：アリスの入力 S_A、ボブの入力 S_B
ボブの出力：S_A と S_B の共通集合の数

プロトコルの「効率」というものを考えた場合、以下の 2 点について考える必要がある。

1. 時間計算量 ― 各ユーザがプロトコルを終わらせるために必要となる時間の総和
2. 通信計算量 ― 通信路を流れるデータのサイズの総和

効率の良い方式を構成するためには、上記二つ

を可能な限り少なくすることが必要である。

明らかに、問題 1 を時間 O(t)、通信 O(c) で解くプロトコルが存在するならば、問題 2 を時間 O($t + n\log n$)、通信 O(c) で解くプロトコルが存在する。しかしながら、一般に逆は成り立たない。したがって、問題 2 を解決するプロトコルを構成する方が簡単であると推測できる。

3 決定的秘匿共通集合計算プロトコル

本稿では、初めに (秘匿化していない) 共通集合計算プロトコルの構成方法について考え、続けてその方式の秘匿化を行う。

集合に対するベクトル表現を考える。すなわち、S を集合、V を長さ N のベクトルとすると、$x \in S$ ならば $V[x-1]=1$, $x \notin S$ ならば $V[x-1]=0$ と定義する。例えば、$U=\{1, 2, 3, 4, 5\}$, $S=\{1, 3, 5\}$ ならば、S のベクトル表現 V は、$V=[1, 0, 1, 0, 1]$ となる。

方式 1

入　　力：アリスの入力 S_A、ボブの入力 S_B

ステップ 1：アリスは S_A をベクトル V_A に変換し、ボブに送る。

ステップ 2：ボブは V_A と S_B から共通集合を出力する。

さて、この簡単な方式の通信計算量 (すなわち通信路を流れるビット数) は N となる。ここで考える問題は、この方式よりも通信量が少ない方式が存在するかどうかである。残念ながら、そのような方式は存在しない。この事実は、[4] の結果を少し修正することにより得られる。

定理 1 共通集合計算プロトコルにおいて、通信計算量が N 未満となるような方式は存在しない。

証明：背理法により証明する。

$N-1$ ビットの通信計算量で共通集合を得ることが可能なプロトコルが存在すると仮定する。仮定により参加者間を流れる異なる情報のパターンは高々 2^{N-1} となる。二対の異なる入力パターン (A, A'), (B, B') を考える。ここで、集合 X に対して $X'=U-X$ と記述する。$A, B \subseteq \{1, ..., N\}$ であるので、全部で 2^N の入力パターンが存在する。鳩ノ巣原理により、通信系列が全く同じ入力対 (A, A'), (B, B') が存在しなければならない。ここで、集合 S_1, S_2 に対して、$S_1 \cap S_2 = \phi$ であれば、DJ(S_1, S_2) = 0、それ以外の場合は DJ(S_1, S_2) = 1 と記述する。この記法を利用すると、DJ(A, A') = DJ(B, B') = 0, DJ(B, A') = 1 とできる。 ここで入力対が (A, A') の時と (B, A') の時を考える。アリスが 1 回目のメッセージを送信するとき、入力が A であろうと B であろうと、必ず同じメッセージをボブに送る。ボブの入力は A' であるので、アリスの入力が A であろうと B であろうと同じメッセージを送る。この状態はプロトコルの終了時まで続くので、結局、入力対 (A, A'), (B, A') は全く同じ通信パターンになる。したがって、入力 (A, A') と (B, A') に対して、ボブは同じ出力をする。しかし、DJ(A, A') ≠ DJ(B, A') であるため、プロトコルは間違った出力をすることになる。以上より、通信計算量は $\Omega(N)$ であることが分かる。

したがって、通信計算量や時間計算量が N 未満となるような方式は存在しない。ここで通信量の下界について述べたが、この下界はストリームアルゴリズムの領域計算量の下界と密接に関係している。例えば、DoS 攻撃、ポートスキャンを検知する省スペースのアルゴリズムが存在しないことをこの共通集合の下界から示すことができる [5]。

方式 1 の秘匿化を行う。

秘匿共通集合計算プロトコル (方式 1)

入　　力：アリスの入力 S_A (V_A), pk、ボブの入力 S_B (V_B), pk, sk

ステップ 1：ボブがアリスに Enc($V_B[0]$), Enc($V_B[1]$), ..., Enc($V_B[N-1]$) を送る。

ステップ 2：アリスは各 i に関して、c_i = Enc($r_i(V_B[i] - V_A[i] + i + 1]$) を計算し、$\{(i, c_i)\}_i$ をボブに送る。ただし、r_i は各 i ごとに選ばれる乱数とする。

ステップ 3：ボブは送られてきた暗号文を復号し、その要素が S_B に含まれる場合、

共通集合に含まれるものとして出力する。

定理1において普遍集合のサイズが N であった場合、通信のコストは $\Omega(N)$ となることを述べたが、集合のサイズ n が $n\log N < N$ を満たす場合に特化すると、以下のプロトコルの方が効率的である。

方式2

入　　力：アリスの入力 S_A、ボブの入力 S_B
ステップ1：アリスはボブに S_A の各要素を送る。
ステップ2：ボブは S_A と S_B の共通集合を出力する。

この方式の通信量は $n\log N$ となる。すなわち、$N > n\log N$ ならば、こちらの方式の方が方式1よりも、時間計算量、通信計算量ともに効率が良くなる。

この方式を秘匿化した方式として、下記を考えることができる。

秘匿共通集合計算プロトコル（方式2）

入　　力：アリスの入力 $S_A = \{a_1, ..., a_n\}$, pk、ボブの入力 $S_B = \{b_1, ..., b_n\}$, pk, sk
ステップ1：ボブはアリスに $\mathrm{Enc}(b_1)$, $\mathrm{Enc}(b_2)$, ..., $\mathrm{Enc}(b_n)$ を送る。
ステップ2：アリスは各 i, j に関して、$\mathrm{Enc}(r_{ij}(b_i - a_j) + a_j)$ を送る。ここで、r_{ij} は乱数である。
ステップ3：ボブは送られてきた暗号文を復号し、その平文が S_B に含まれる場合、共通集合に含まれるものとして出力する。

この方式の通信計算量は $O(n^2)$ となり、必ずしも効率的であるとは言い切れない。これに対する解決策はフリードマンらにより提案された[3]。しかしながら、この方式を安全にするためには若干の修正が必要であったため、それを修正した方式を示す。

秘匿共通集合計算プロトコル（方式2改）

入　　力：アリスの入力 $S_A = \{a_1, ..., a_n\}$, pk、ボブの入力 $S_B = \{b_1, ..., b_n\}$, pk, sk
ステップ1：ボブは、$f(X) = X^n + c_{n-1}X^{n-1} + ... + c_0 = (X - b_1)(X - b_2) ... (X - b_n)$ の各 c_i を暗号化してアリスに送る。
ステップ2：各 j に関して、アリスは $\mathrm{Enc}(r_j f(a_j) + a_j)$ をボブに送る。
ステップ3：ボブは送られてきた暗号文を復号し、その平文が S_B に含まれる場合、共通集合に含まれるものとして出力する。

この方式の通信量は、$O(n)$ であり、方式2の $O(n^2)$ よりも格段に改良されていることが分かる。

この方式をバケットアロケーションというテクニックを使い実装すると、使わない場合と比べて、時に約20倍の高速化を達成できる。

なお、この方式は、スピア型判定器を構成する際に利用された[8]。

4　近似プロトコル

4.1　近似共通集合計算プロトコル

ここまで共通集合を出力するプロトコルを考えてきた。本章では、高速に動作する秘匿共通計算プロトコルを得るために、近似プロトコルに構成方法について考えていく。ここで言う近似プロトコルとは、$S_A \cap S_B$ を計算する代わりに、$|S_A \cap S_B|$ の近似値を計算するものである。

ここでまず、近似プロトコルを得るための道具として、最小値独立関数族を紹介しておく。

定義［最小値独立関数族[1]］
関数族 $H \subset [N] \to [u]$ が、任意の $X \subset [N]$ と $x \in X$ に対して、
$\Pr_{h \leftarrow H}[h(x) = \min\{h(y) \mid y \in X\}] = 1/|X|$.
を満たすならば、その関数族は最小値独立性を満たすという。

補題：H を最小値独立関数族とすると、任意の $A, B \subseteq [N]$ に対して、
$\Pr_{h \leftarrow H}[\min\{h(A)\} = \min\{h(B)\}] = |A \cap B|/|A \cup B|$
が成り立つ。

match$(A, B) = |A \cap B| / |A \cup B|$, sum$(A, B) = |A| + |B|$とする。matchとsumで$|A \cap B|$を表現すると、

$$|A \cap B| = \mathrm{match}(A, B) \cdot \mathrm{sum}(A, B) \cdot (1 + \mathrm{match}(A, B))^{-1}$$

となる。すなわち、$|A \cap B|$の近似を得ることと、$|A \cap B| / |A \cup B|$の近似を得ることは等価となる。そこで本節では、$|A \cap B| / |A \cup B|$の近似を計算するプロトコルを考える。まず、従来どおり秘匿化してない方式から考えていく。

方式 3

入　　力：アリスの入力 S_A、ボブの入力 S_B、共通の入力 l

出　　力：アリスの出力はなし、ボブの出力は $|S_A \cap S_B|$ の近似

ステップ 1：アリスは l 個の最小値独立関数をランダムに選びボブに送る。

ステップ 2：アリスは、$(a_1, \ldots, a_l) = (\min\{h_1(S_A)\}, \ldots, \min\{h_l(S_A)\})$ を計算する。ボブも同様に、$(b_1, \ldots, b_l) = (\min\{h_1(S_B)\}, \ldots, \min\{h_l(S_B)\})$ を計算する。

ステップ 3：アリスは、$(a_1, \ldots, a_l)$をボブに送る。

ステップ 4：ボブは、$|\{1 \le i \le l \mid a_i = b_i\}| / l$ を計算し出力する。

4.2　解析

ここでは、ボブが出力した $|\{1 \le i \le l | a_i = b_i\}| / l$ と $|A \cap B| / |A \cup B|$ との間にどの程度の違いがあるかについて解析する。i 回目がマッチする確率は、最小値独立関族の性質から $|A \cap B| / |A \cup B|$ である。i 回目でマッチした場合が 1、マッチしなければ 0 となるような確率変数 X_i を考える。この確率変数の期待値と分散を求める。期待値は、$E[X_i] = 1 \cdot \Pr[X_i = 1] + 0 \cdot \Pr[X_i = 0] = |A \cap B| / |A \cup B|$ となる。分散はその定義から、$\mathrm{Var}[X_i] = E[X_i^2] - E[X_i]^2$ であった。$E[X_i^2] = 1 \cdot \Pr[X_i = 1] + 0 \cdot \Pr[X_i = 0] = |A \cap B| / |A \cup B|$ であることに注意すると、$|A \cap B| / |A \cup B| - (|A \cap B| / |A \cup B|)^2$ となる。ここで新たに $X = (X_1 + X_2 + \ldots + X_l) / l$ と定義される確率変数を考える。期待値の線形性から、

$$\begin{aligned} E[X] &= E[(X_1 + X_2 + \ldots + X_l) / l] \\ &= E[(X_1 + X_2 + \ldots + X_l)] / l \\ &= (E[X_1] + E[X_2] + \ldots + E[X_l]) / l \\ &= 1_{A \cap B} / |A \cup B| \end{aligned}$$

となる。また分散は、

$$\begin{aligned} \mathrm{Var}[X] &= \mathrm{Var}[(X_1 + X_2 + \ldots + X_1) / l] \\ &= \mathrm{Var}[X_1 / l] + \mathrm{Var}[X_2 / l] + \ldots + \mathrm{Var}[X_1 / l] \\ &= (|A \cap B| / |A \cup B| - (|A \cap B| / |A \cup B|)^2) / l^2 \end{aligned}$$

となる。したがって、チェビシェフの不等式から、

$$\begin{aligned} \Pr[|X - 1_{A \cap B} / |A \cup B|| \ge \varepsilon] &\le (|A \cap B| / |A \cup B| - (|A \cap B| / |A \cup B|)^2) / l^2 \varepsilon^2 \\ &\le 1 / l^2 \varepsilon^2 \end{aligned}$$

が任意の正の ε に対して成立する。以上より、アルゴリズムの出力が $1_{A \cap B} / |A \cup B|$ から外れる確率が求められた。

4.3　方式 3 を秘匿化した方式

本節では、前節で考えた方式 3 の秘匿化を行う。そのためには、準同型暗号を使い等価性判定器を作成しなければならない。このプロトコルではアリスが x をボブが y をもっており、$x = y$ であればボブは 1 を出力し、それ以外のときは 0 を出力する。

秘匿等価性判定プロトコル

入　　力：アリスの入力 x、ボブの入力 y

ボブの出力：$x = y$ ならば 1、それ以外のときは 0 を出力

ステップ 1：ボブはアリスに $\mathrm{Enc}(y)$ を送る。

ステップ 2：アリスは $\mathrm{Enc}(r(x - y) + 1)$ を計算し、その結果をボブに送る。

ステップ 3：ボブは送られてきた暗号文を復号し、1 であれば 1 を出力し、それ以外のときは 0 を出力する。

さて、秘匿等価性判定プロトコルを応用し、方式 3 の秘匿化を行う。

秘匿共通集合計算プロトコル（方式 3）

入　　力：アリスの入力 S_A、ボブの入力 S_B、共通の入力 l

出　　力：アリスの出力はなし、ボブの出力は $|S_A \cap S_B|$ の近似

ステップ 1：アリスは l 個の最小値独立関数をランダムに選びボブに送る。

ステップ 2：アリスは、$(a_1, ..., a_l) = (\min\{h_1(S_A)\}, ..., \min\{h_1(S_A)\})$ を計算する。ボブも同様に、$(b_1, ..., b_l) = (\min\{h_1(S_B)\}, ..., \min\{h_1(S_B)\})$ を計算する。

ステップ 3：ボブは、$(\mathrm{Enc}(b_1), ..., \mathrm{Enc}(b_l))$ をアリスに送る。

ステップ 3：アリスは、$(\mathrm{Enc}(r_1(a_1 - b_1) + 1), ..., \mathrm{Enc}(r_l(a_l - b_l) + 1))$ をボブに送る。

ステップ 4：ボブは、復号して 1 の数を数える。その合計を sum とすると、彼は、sum/l を出力する。

5 おわりに

暗号プロトコルは主に理論研究として発展してきたため、本稿で述べた秘匿共通集合計算プロトコルのように、実社会での運用に耐え得るようなものは、ほとんど存在しない[8]。この点から考えると、理論研究としてこれまで育ってきた暗号プロトコルにも、実社会での応用を強く意識することにより、まだ、巨大な研究分野を開拓できる可能性がある。

ここでインターネットへの応用を考えるだけでも、暗号プロトコルの応用範囲は広いと推測される。例えば、トレースバック技術は不正行為を行ったユーザを追跡するためのものであるが、残念ながら不正行為を行っていない正当なユーザのプライバシを侵害してしまう可能性がある。暗号技術を使い正当なユーザのプライバシを守りつつ、不正なユーザを追跡する。そのような暗号プロトコルを設計することが筆者の次の大きな研究課題となっている。

参考文献

1 Andrei. Z. Broder, M. Charikar, Alan. M. Frieze, and Michael Mitzenmacher, "Min-Wise Independent Permutations", J. Compute. Syst. Sci. 60(3), pp.630-659.

2 Taher ElGamal, "A Public-Key Cryptosystem and a Signature Based on Discrete Logarithms", IEEE Transactions on Information Theory, Vol.IT-31, No.4, 1985.

3 Michael J. Freedman, Kobbi Nissim, and Benny Pinkas, "Efficient Private Matching and Set Intersection", EUROCRYPT 2004, pp.1-19.

4 Eyal Kushilevitz, and Noam Nissan, "Communication Complexity", Cambridge University Press, 1997.

5 Kirill Levchenko, Ramamohan Paturi, and George Varghese, "On the Difficulty of Scalably Detecting Network Attacks", ACM Conference on Computer and Communication Security 2004, pp.12-20.

6 Pascal Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Class", EUROCRYPT 1999, pp.223-238.

7 Andrew Chi-Chih Yao, "Protocols for Secure Computations", FOCS 1982, pp.160-164.

8 スピア型サイバー攻撃判定システム開発のための共同実証実験を開始 ―特定の組織に限定したサイバー攻撃を早期に検知するシステムの実現に向けて―, http://www.nict.go.jp/news/h19-press.html, 2008

野島 良（のじま りょう）
情報通信セキュリティ研究センタートレーサブルネットワークグループ研究員　博士（工学）
アルゴリズム論、暗号理論