Vol.2021-AL-182 No.4
2021/3/17

二分探索が可能な対称秘匿データ構造とその簡潔化

小倉 拳[1]　定兼 邦彦[1]

概要：ビッグデータ時代のマルチパーティー計算でよくあるシナリオが，大量のデータを持つオーナーが，外部のサーバーにデータを保管し，クライアントがそのデータに対してクエリをかけるシナリオである．本研究では，データオーナーが持つ配列 V に対して，クライアントが V における x のランク，つまり V の中で x 以下の要素の個数をクエリするランク決定クエリに応答するプロトコルで，(i) サーバーはクライアントがクエリした x の情報を何も得ず，(ii) クライアントは V に関して x のランク以外の情報を何も得ず，(iii) サーバーは V についても x についても情報を何も得ないようなプロトコルを提案した．本プロトコルは複数回クエリしても，余計な情報がクライアントには漏れないようになっている．また，提案したデータ構造内で利用する ORAM に関して，空間計算量をクエリ数に依存しない量に抑えるための手法も提案した．

1. 序論

複数のパーティーが互いに通信しながら計算を行うことを**マルチパーティ計算**という．秘密計算の分野においては，各パーティーが持っている漏らしたくない情報を漏らさないように計算結果を共有するプロトコルの研究が多くなされてきた．

1.1 外部委託シナリオ

ビッグデータ時代によくあるマルチパーティーのシナリオが，大量のデータの持ち主であるオーナーが，潤沢な計算資源を持つ外部サーバーにデータを保管し，それに対してクライアントが検索のクエリをかけるシナリオである．このシナリオで秘匿したい情報は２つある．

１つめはクライアントがどのデータにアクセスしたかという情報である．これはオーナーとサーバーに対して秘匿したい情報である．具体例をあげると，大量の医療データを持つ医療機関をオーナーとして，それにアクセスする医者，患者，医療研究者というシナリオのときにこの情報を秘匿したくなる．どの医療データにアクセスしたかという情報はそれ自体が個人情報となりうるからである．

２つめはクライアントがクエリした以上の情報である．これはクライアントとサーバーに対して秘匿したい．上で挙げた医療データの例だと，患者が自分のカルテにアクセスしたとき，関係ない患者の情報は秘匿したい．他にも，患者の個人情報は漏らしたくないが，統計処理のために各項目の平均，分散などはアクセスできるようなプロトコルの時，平均の情報を得る際に一部の患者の情報のヒントが多少なりとも漏れるのは好ましくない．

本稿では，これら２つの情報のうち片方のみを秘匿するものを**非対称秘匿**，両方秘匿するものを**対称秘匿**と呼ぶことにする．従来の検索可能暗号はアクセスパターンの情報がサーバーに漏れるため非対称秘匿であった．後述する ObliviousRAM(ORAM) はサーバーへの秘匿の要件は満たすが，クライアントに対して何かを秘匿する機能を持たないため，やはり非対称秘匿であった．このシナリオで役に立ち，かつ対称秘匿性を実現するマルチパーティー計算プロトコルに紛失通信 (Oblivious Transfer) [1] や Symmetric private information retrieval [2] があるが，どれも通信量や計算量がデータ数 N に対して多項式時間[*1]かかる．二分探索のようにデータに入った順序関係を利用して高速 (例えば N の対数多項式時間) に探索クエリに応答できて，かつ対象秘匿であるものは現在までに無かった．

1.2 本研究の貢献

本研究では対称秘匿な設定のもとで, 以下で定義するランク決定クエリにデータ数 N の対数多項式時間 ($(\log N)^{\Theta(1)}$ 時間) で応答できるプロトコルを提案した．またプロトコル内で使う ORAM の空間計算量を削減するため，SuccinctORAM [3] に改良を加え，クエリ数に依存せず簡潔である PKUORAM を提案した．

[1] 東京大学
The Uniersit of Tokyo

[*1] ここで多項式時間は $N^{\Theta(1)}$ の意味で使っている，つまり $\Theta(\sqrt{N})$ なども含まれる

2. 準備

2.1 記号

$\mathrm{Enc}_K(p)$ により暗号鍵 K によって平文 p を暗号化した暗号文を表わす．どの鍵を用いて暗号化しているかが明らかな場合は $\mathrm{Enc}_K(p)$ を $[[p]]$ と書く．$\mathrm{Dec}_K(c)$ により暗号鍵 K によって暗号文 c を復号して得られる平文を表す．整数 a と正の数 m に対して $a \mod m$ で $a = km + r(k, r \in \mathbb{Z}, 0 \leq r < m)$ を満たす唯一の r を表す．

2.2 ランク決定問題

定義 2.1. (ランク決定問題) 要素数 N の整数列 $V = (V_1, \ldots, V_N)$ に対する**ランク決定問題**とは，与えられた整数 x に対して，整数列 V の中での x のランク $\mathrm{rank}(V, i)$, すなわち $V_i \leq x$ となるインデックス $i \in [1, N]$ の個数を求める問題である．[*2]

2.3 マルチパーティー計算の用語

マルチパーティ計算において実現したい**機能 (functionality)** は，各パーティーの入力を所望の各パーティーの出力に移す関数として表す． k パーティーからなる機能 f について，パーティー $i(1 \leq i \leq k)$ の入力を x_i としたときの機能 f の出力を $f_i(x_1, \ldots, x_k)$ と書く．

k パーティーからなるプロトコル π について, パーティー $i(1 \leq i \leq k)$ の入力を x_i としたときの各パーティーの出力を $\pi_i(x_1, \ldots, x_k)$ と書く．プロトコル π の実行中にパーティー i が得る情報を $\mathrm{VIEW}_i(x_1, \ldots, x_k)$ と書く．

機能 f を実現する信頼できる第三者を機能 f の**オラクル (Oracle)** と呼ぶ．オラクルを含むプロトコルを**ハイブリッドプロトコル**と呼ぶ．ハイブリッドプロトコルでは，オラクルとの通信は直列に行われるものとする．つまり，オラクル以外のパーティーがオラクルに情報を送信した場合は，オラクルから返答が来るまで他のパーティー/オラクルとは通信をしないものとする．

2.3.1 安全性モデル

本研究に登場するすべてのプロトコルは **semi-honest** モデルを想定している．semi-honest モデルは，プロトコルの参加者はプロトコルの仕様通りの通信と計算を行うという前提のモデルである．semi-honest モデル上で安全なプロトコルは，攻撃者がプロトコルに従って得られる情報のみを用いて追加の計算を行い秘匿されるべき情報を盗み見ようとしても，情報論的，あるいは計算論的に不可能であるということである．これに対して攻撃者がプロトコルを逸脱した通信，計算を行っても良いモデルを **malicious** モデルと呼ぶ．malicious モデルでは攻撃者は偽の情報を流したり，他者と偽って通信したりするなどの攻撃が許される．

2.3.2 識別不可能性

定義 2.2. (識別不可能性) 添字付けされた 2 つの確率変数列 X_i, Y_i についてこの 2 つが識別不可能であるとは，任意の確率的多項式時間アルゴリズム $\mathcal{D}$ に関して，$|\mathrm{P}[\mathcal{D}(X_i) = 1] - \mathrm{P}[\mathcal{D}(Y_i) = 1]| \leq 2^{-|i|}$ となることを表す．ここで $|i|$ で i を表現するために bit 長を表した．

2.3.3 プロトコルの安全性

k パーティーからなるプロトコル π と機能 f について，プロトコル π が semi-honest モデルのもとで, プロトコル π が機能 f を安全に実現するとは, 各パーティー $i(1 \leq i \leq k)$ について，ある確率的多項式時間アルゴリズム $\mathcal{S}_i$ が存在し， $\{\mathcal{S}_i(x_i, \mathrm{VIEW}_i(x_1, \ldots, x_k)), \pi(x_1, \ldots, x_k)\}_{x_1,\ldots,x_k}$ と $\{f_i(x_1, \ldots, x_k), f(x_1, \ldots, x_k)\}_{x_1,\ldots,x_k}$ が識別不可能であることを表す．

2.3.4 結合定理

semi-honest モデルでは，複数のプロトコルを直列に組合せても安全性は失われないことが知られている．

定理 2.3. (**結合定理** [4] Thm 7.3.3) 機能 f のオラクルを含むハイブリッドプロトコル π^f が機能 g を実現しており，プロトコル ρ が機能 f を実現しているとする．これらから π^f のオラクルへのアクセスをプロトコル ρ で置き換えたプロトコル π^ρ が構成できる．このとき π^ρ は機能 g を安全に実現する．

2.4 既存のプロトコル

2.4.1 秘匿比較可能暗号

定義 2.4. (秘匿比較可能暗号) 暗号化手法の中でも，2 つの暗号文 $\mathrm{Enc}_{K_p}(a), \mathrm{Enc}_{K_p}(b)$ を持つ者と，これらを復号する鍵 K_s を持つ者の 2 者間で通信を行うことで，前者が $\mathrm{Enc}_{K_p}(a < b)$ を入手し，後者は追加情報を得ないような機能を実現するプロトコルが存在するものを秘匿比較可能暗号と呼ぶ．

2.4.2 加法準同型暗号

定義 2.5. (加法準同型暗号) 暗号化手法の中でも，2 つの平文 a, b を暗号化した暗号文 $\mathrm{Enc}_{K_p}(a), \mathrm{Enc}_{K_p}(b)$ のみから a と b の加算を行った結果の暗号文，つまり $\mathrm{Enc}_{K_p}(a+b)$ を容易に得られるものを**加法準同型暗号**と呼ぶ．一般に加法準同型暗号は公開鍵暗号方式である．

Paillier 暗号 [5] は有名な加法準同型暗号である．加法準同型暗号は DGK 比較プロトコル [6], [7] により，秘匿比較可能である．これらは IND-CPA 安全性を満たしている．

2.4.3 DGK 比較プロトコル

提案プロトコルでは, 2 つの暗号文 $\mathrm{Enc}_{K_p}(a)$ と $\mathrm{Enc}_{K_p}(b)$ から，それらを復号せずに，その大小を比較した結果の暗号文 $\mathrm{Enc}_{K_p}(a < b)$ を得る機能が必要となる．そのような

[*2] $V_i \leq x$ ではなく $V_i < x$ となるインデックスの個数を求める流儀もある．

機能を実現するプロトコルとして加法準同型暗号を利用した DGK 比較プロトコル [6], [7] が知られている．DGK 比較プロトコルはパーティが持つ情報，得たい情報，秘匿したい情報についてバリエーションがあるが，本稿内では，秘匿比較可能暗号の定義 2.4 に現れる条件を満たすプロトコルを指すものとする．

2.5 ObliviousRAM

ObliviousRAM (ORAM) はクライアントとサーバーの 2 者からなるプロトコルである．サーバーは N 要素のランダムアクセスメモリ V のように振る舞い，クライアントからの以下の 2 種類のクエリに応答する．

- **READ(** i **)**： V_i を返す．
- **WRITE(** i **,** v **)**： V_i に v を代入し，書き込んだ値を返す．

ここで，V は仮想的なものであることに注意せよ．サーバー上のある連続したアドレスに V の要素が $V_1, \ldots, V_N$ の順に格納されている必要はない．

本稿では ORAM は以下の情報論的安全性を満たすもののみを考える．

定義 2.6. (ORAM のクエリ識別不可能性) ORAM のサイズを N，M を正の数とする．任意の 2 つの長さ M のクエリ列 $P^1 = [\mathrm{op}_1^1, \mathrm{op}_2^1, \ldots, \mathrm{op}_M^1]$, $P^2 = [\mathrm{op}_1^2, \mathrm{op}_2^2, \ldots, \mathrm{op}_M^2]$ に対して，ORAM サーバーにはどちらの列を選択したかを隠してクエリした時，ORAM サーバーはアクセスパターンからどちらの列が選択されたかを情報論的に区別できない．

PathORAM [8]，OnionORAM [9]，SuccinctORAM [3] などの TreeBaseORAM はこの情報論的識別不可能性を満たす．OptORAMa [10] など，別のモデルの ORAM は識別不可能性が計算量的安全性になる．

ORAM はクライアントに対して何か秘匿する要件はないことに注意されたい．

2.5.1 ORAM のデータの暗号化

ORAM はクエリ識別不可能性を達成するために，データはすべて暗号化されてサーバー上に保管される前提である．とくに，V 上の値を変更しない READ と変更する WRITE の区別がつかないようにするため，一度でもアクセスしたデータは必ず一度復号し，再び暗号化し直して，内容が変わっているかどうかがサーバーにわかないようにする必要がある．よってデータの暗号化手法として，同じ平文でも暗号化するたびに違う暗号文になるようなものを選ばなければならない．これを実現する方法として，任意の疑似ランダム関数 $F_K(\cdot)$ を用いて，平文 m を $(i, m \oplus F_k(i))$ に暗号化する枠組みがある．ここで i は今まで暗号化を行った回数であり，$\oplus$ はビットごとの排他的論理和演算子である．この枠組みの暗号文として AES 暗号のカウンターモードが有名である．実際に，SuccinctORAM [3] では，この暗号化方式を用いてデータを暗号化している．本稿では，ORAM 上のすべてのデータは上記の枠組みの暗号化方式を用いて暗号化しているものとする．

本章ではランク決定クエリに対数多項式時間で応答できる対称秘匿なプロトコルを定義する．

3. 問題設定と定義

定義 3.1. (外部委託対称秘匿ランク決定機能) 以下を満たすオーナー，サーバー，クライアントの 3 パーティからなる機能を**外部委託対称秘匿ランク決定機能**と定義する．オーナーは N 要素からなる整数列 V のデータを所有しており，外部のサーバーに保存している．クライアントはランク決定問題を m 回クエリする． i 回目のクエリではクライアントは x_i のランクをクエリする．3 パーティー間で通信を行った結果クライアントは $rank(V, x_i)$ を得るが，V に関してそれ以上の情報を得ない．またオーナーはクライアントがクエリした x_i に関しての情報を何も得ない．サーバーは， V と x_i と $rank(V, x_i)$ のいずれの情報も得ない．なお， V の要素数 N については公開情報とする．

通常のランク決定問題は，あらかじめ整数列を RAM 上に昇順に格納しておけば， $V_i \leq x$ となる最大の i (そのような i が無い場合は 0) を求める問題に置き換えることが出来る．この問題は以下で定義する典型的な二分探索によって $\Theta(\log N)$ 時間で応答可能である．

3.1 二分探索アルゴリズム

簡単のため， $\mathrm{rank}(V, x) \neq 0$ とする[*3]． $L = \lceil \log(N) \rceil$ として，二分探索は L ラウンドの比較からなる．初め $l_0 = 1, r_0 = N + 1$ とする．k ラウンド目 ($1 \leq k \leq L$) では，$i_k = \lfloor \frac{l_{k-1} + r_{k-1}}{2} \rfloor$ として V_{i_k} と x の大小比較を行う．この結果を c_k とする．すなわち，$V_{i_k} \leq x$ ならば $c_k = 1$, そうでないならば $c_k = 0$ とする．この c_k の値に応じて (l_k, r_k) の値を決める．もし $c_k = 1$ ならば $(l_k, r_k) = (i_k, r_{k-1})$ とし，そうでないなら $(l_k, r_k) = (l_{k-1}, i_k)$ とする．各 k ($0 \leq k \leq L$) において， $\mathrm{rank}(V, x) \in [l_k, r_k)$ が保たれ，また $r_k - l_k \leq \lceil \frac{r_{k-1} - l_{k-1}}{2} \rceil (1 \leq k \leq L)$ が成り立つため，$k = \lceil \log(N) \rceil$ においては $[l_k, r_k)$ は要素を 1 つしか持たない．この唯一の要素 l_k が $\mathrm{rank}(V, x)$ となる．

詳細を Algorithm 1 に示す．なお，後の秘匿性の解析のために，ランクだけでなく比較したインデックス $i := [i_1, \ldots, i_{|i|}]$ ，比較結果 $c := [c_1, \ldots, c_{|c|}]$ ，変数 l_k, r_k のそれぞれの列 $l := [l_1, \ldots, l_{|l|}], r := [r_1, \ldots, r_{|r|}]$ も返すようにしている．

外部委託モデルの場合，RAM の代わりに ORAM 上に V を格納し同様のアルゴリズムを実行すれば，サーバーに

[*3] $\mathrm{rank}(V, x) = 0$ であるかどうかは V_1 と x を比較することで判定出来るため，アルゴリズムの序盤にこの確認を行えばこの仮定を外すことが出来る．

Algorithm 1 二分探索によるランク決定アルゴリズム

Input: サイズ N のソート済み整数列 V ，整数 x
Output: rank (V,x) ， i ， c ， l ， r

if $x < V_0$ **then**
  return $0, [], [], [], []$
end if
$(l_0, r_0) \leftarrow (0, N)$
$(i_0) \leftarrow \lfloor \frac{N}{2} \rfloor$
for $k \leftarrow 1$ から L **do**
  if $V[i_k] \leq x$ **then**
    $c_k \leftarrow 1$
    $(l_k, r_k) \leftarrow (i_k, r_{k-1})$
  else
    $c_k \leftarrow 0$
    $(l_k, r_k) \leftarrow (l_{k-1}, i_k)$
  end if
  $i_{k+1} \leftarrow \lfloor \frac{l_k + r_k}{2} \rfloor$
end for
return l_L ， $[i_1, \ldots, i_{L+1}]$ ， $[c_1, \ldots, c_L]$ ， $[l_1, \ldots, l_L]$ ， $[r_1, \ldots, r_L]$

対してアクセスパターンを秘匿しながらランク決定問題を解くことが出来る．しかし，二分探索の過程でクエリされた整数 x と ORAM 上のデータの比較をせねばならない．外部委託対称秘匿ランク決定プロトコルの設定ではこれら２つを同時に知って良いパーティはいないので，工夫が必要である．

4. 4パーティー対称秘匿ランク決定プロトコル

目標となるプロトコルはオーナー，クライアント，サーバーからなる 3 パーティーからなる対称秘匿なプロトコルであるが，円滑な説明のため，まずは探索者を追加した 4 パーティーからなる，対称秘匿なプロトコルを導入する．

4.1 構成

本プロトコルに登場するパーティは**データオーナー**，**クライアント**，**サーバー**，**探索者**の 4 パーティである．このうちクライアントと探索者は同一パーティが兼任しても構わない．

サーバーは w_s bit のデータを読み書きできる ORAM サーバーである．充分な容量を持つとする．

データオーナーは ランク決定問題でクエリされる整数列 V を持っている．整数列 V の要素数を N とする．事前にサーバーにこれらの情報を書き込むことが出来るが，探索者が一度でもクエリを出したらそれ以降はサーバーからデータを読み書きすることはできない．

クライアントは m 個の整数 $x_1, \ldots, x_m$ を持っており，各 x_i についてデータオーナーが持つ整数列 V における x_i のランクを知りたい．

探索者は事前情報を何も持たないパーティである．データオーナーがサーバーにデータを格納し終わった後, ORAM サーバーに READ をクエリする権限を持つ．

4.2 SBC(swap before comparison) プロトコル

4 パーティー対象秘匿ランク決定プロトコル内では，通常の二分探索における大小比較の部分を秘匿した状態で行う必要がある．そのためにまず，以下で定義するの対称秘匿比較機能を実現する SBC(swap before comparison) プロトコルを導入する．

定義 4.1. (対称秘匿比較) 2 数 x, y を加法準同型暗号によって暗号化した暗号文を $[[x]], [[y]]$ と表し，その秘密鍵を K_s と表す．2 パーティーからなる機能で $f(([[x]], [[y]]), K_p) = (x < y, \lambda)$ を満たすものを**対象秘匿比較**と呼ぶ．ここで λ で空の情報を表した．

SBC プロトコルは，秘匿比較プロトコルの実行前に比較対象の 2 数を $\frac{1}{2}$ の確率でプロトコル実行前に両者の値を入れ替えるようにするプロトコルである．具体的には，$[[x]], [[y]]$ を比較する前に一様乱数 $r \leftarrow \{0, 1\}$ を生成し，$r = 1$ なら $x < y$ かどうかの比較を，$r = 0$ なら $y < x + 1$ かどうかの比較を秘匿比較プロトコルを用いて行い，その後オーナーにその秘匿結果を複号してもらう．復号してもらった値を c'_k とすると $c_k = rc'_k + (1 - r)(1 - c'_k)$ とすれば $c_k = (x < y)$ となる．このときオーナーは c'_k を知るが， c'_k は c_k とは独立な一様乱数になるため，何の情報も得ない．

補題 4.2. 以下の機能 g を安全に実現する秘匿比較プロトコルが与えられるとする．$g(([[x]], [[y]]), \lambda) = ([[x < y]], \lambda)$

SBC プロトコルは，安全な秘匿比較プロトコルの元，以下の機能 f を安全に実現する．

$f(([[x]], [[y]]), \lambda) = (x < y, \lambda)$

ここで， λ で空の入出力を表すとする．

Proof. 機能 g を安全に実現する信頼できるサードパーティを組み込んだハイブリッドモデルの上で，機能 f が安全に実現されることを示す．

$f(\cdot, \cdot)$ の戻り値の第 1,2 成分を $f_1(\cdot, \cdot), f_2(\cdot, \cdot)$ で表すとする．

SBC プロトコルにおいて比較したい 2 数の暗号文を持つ者を第 1 のパーティ，秘密鍵を持つ者を第 2 のパーティーと呼ぶ． $\mathrm{VIEW}_i([[x]], [[y]])$ ， $\mathrm{OUTPUT}_i([[x]], [[y]])$ で第 1 のパーティが比較したい 2 数の暗号文が $[[x]], [[y]]$ であったときの，第 i のパーティがプロトコル内で得る情報，出力を表すとする． $\mathrm{OUTPUT}([[x]], [[y]])$ で $(\mathrm{OUTPUT}_1([[x]], [[y]]), \mathrm{OUTPUT}_2([[x]], [[y]]))$ を表すとする．

以下を満たす多項式時間アルゴリズム $\mathcal{S}_1, \mathcal{S}_2$ の存在を示せば良い．

- 任意の $[[x]], [[y]]$ に対して， $(\mathcal{S}_1(([[x]], [[y]]), f_1([[x]], [[y]])), f([[x]], [[y]]))$ と $(\mathrm{VIEW}_1(([[x]], [[y]]), \lambda))$, $\mathrm{OUTPUT}([[x]], [[y]])$ が識別不能である．
- 任意の $[[x]], [[y]]$ に対して， $(\mathcal{S}_2(f_2([[x]], [[y]]))$

, $f([[x]],[[y]]))$ と $(\mathrm{VIEW}_2(([[x]],[[y]]),\lambda))$, $\mathrm{OUTPUT}([[x]],[[y]])$ が識別不能である．

ここで $\mathrm{VIEW}_1([[x]],[[y]])$ は生成した乱数の値 r と，スワップ後の 2 数の秘匿プロトコルにおける比較結果の暗号文 $[[c']]$ ($r=0$ なら $[[x<y]]$ で， $r=1$ なら $[[x\geq y]]$)，その復号結果 c' と最終的な比較結果 c である．まとめると， $(r,[[c']],c',c)$ である．

また， $\mathrm{VIEW}_2([[x]],[[y]])$ は秘匿プロトコルにおける比較結果の暗号文の復号結果 c' である．

$\mathcal{S}_1$ を以下のようなアルゴリズムとする．まず $([[x]],[[y]])$ と $f_1([[x]],[[y]]))$ を受け取り $c=f_1([[x]],[[y]])$ とする．$\{0,1\}$ から一様ランダムに $\tilde{r}$ を生成する． $\tilde{c}=\tilde{r}\oplus\tilde{c}$ とし，$\tilde{c}$ を暗号化し暗号文 $[[\tilde{c}]]$ を得る．最後に $(r',[[\tilde{c}]],\tilde{c},c)$ を返す． r と $\tilde{r}$ は他の変数の値に依存しない独立な一様乱数であるため情報論的に区別がつかない． $c'=c\oplus r$ と $\tilde{c}=c\oplus\tilde{r}$ は c と $\tilde{r}$ から決定される値なので，同じ確率分布に従い，情報論的に区別がつかない． $[[c']]$ と $[[\tilde{c}]]$ は同じ確率分布に従う確率変数から同じ確率的アルゴリズムで算出した値であるため，おなじ確率分布に従う．よって区別がつかない．以上より，このアルゴリズムの出力は $\mathrm{VIEW}_1(([[x]],[[y]]),\lambda)$ と識別不能である．

$\mathcal{S}_2$ を以下のようなアルゴリズムとする． $\{0,1\}$ から一様ランダムに $\tilde{c}$ を生成し， $\tilde{c}$ を返す． c' は一様乱数と同じ分布に従うので, $\tilde{c}$ と c' は情報論的に区別がつかない．よって，このアルゴリズムの出力は $\mathrm{VIEW}_2(([[x]],[[y]]),\lambda)$ と識別不能である． □

4.3 4 パーティー対称秘匿プロトコルの詳細

4 パーティー対称秘匿プロトコルでは，まずオーナーは V の各要素を比較可能暗号を用いて暗号化して ORAM 上に昇順に格納する[*4]．暗号化に用いた公開鍵を K_p ，対応する秘密鍵を K_s とする．クライアントは探索者に x をクエリするとき，公開鍵で暗号化した暗号文 $\mathrm{Enc}_{K_p}(x)$ を探索者に伝える．探索者は前節で説明したアルゴリズムと同様， L 回の比較を行いランクを決定する．

k ラウンド目 ($1\leq k\leq L$) では V_{i_k} と x とを大小比較を行いながら二分探索を行う．ここでの大小比較には前節で導入した SBC プロトコルを用いる．これにより探索者は $c_k=(V_{i_k}<x)$ を得ることができ，二分探索の次のラウンドに進むことができる．

これを L 回繰り返すことで，探索者は $\mathrm{rank}(V,x)$ を求めることができる．最後に探索者はクライアントに $\mathrm{rank}(V,x)$ を伝えてプロトコルは完了する．

[*4] この暗号化は節 2.5.1 で述べた暗号化とは独立のものということに注意せよ．つまり，比較可能暗号で暗号化したデータはさらに AES 暗号などを用いて暗号化されて ORAM 上に格納されることになる．

4.4 安全性の解析

上述のプロトコルが完了した時点で各パーティが知っている情報をまとめる．

サーバーが行う通信は，オーナーによる V の書き込みと，探索者による V のあるインデックスの読み込みである．探索者は Rank クエリに一度応答する間に ORAM に対して L 回の Read クエリを発行するため，ORAM サーバーは Lm 回のクエリされたことを知る．定義 2.6 から，ORAM は何を読み書きされたかの情報は得ない．

データオーナーが行う通信は，サーバーへの V の書き込みと，探索者との秘匿比較プロトコルと，その結果の復号のためのやりとりである．設定上，オーナーは V の値はあらかじめ知っている．二分探索の k ラウンド目 $(1\leq k\leq L)$ では，秘匿比較プロトコルにより V_{i_k} と x そのものの値は秘匿されるが，その後の復号のやり取りによって $V_{i_k}<x$ の結果，すなわち c_k の情報を得る．

探索者が行う通信は，クライアントとのクエリ応答，二分探索の過程でのサーバーの値の読み取り，オーナーとの秘匿比較プロトコルとその比較結果 $V_i<x$ の復号のためのやりとりである．クライアントからのクエリや二分探索の過程でサーバーから読み取った値は暗号化されているため，平文は秘匿される．二分探索の k $(1\leq k\leq L)$ ラウンド目では，オーナー同様，秘匿比較プロトコルにより V_{i_k},x の値を知ることはなく，その後の復号のやり取りによって c_k を知る．また二分探索の過程で出てくる各変数の値 (i_k , c_k , l_k , r_k) を知る．最終的に $\mathrm{rank}(V,x)$ を得る．

クライアントが行う通信は，探索者とのクエリ応答のみである．最終的に知っている情報は m 回のクエリに対する，クエリした値の平文 x_i とそのランク $\mathrm{rank}(V,x_i)$ である．

ここで以下の補題により，このプロトコルにおいて，データオーナーや探索者が得ている情報は，実現したい外部委託対称秘匿ランク決定機能の出力 $rank(V,x)$ から確率的多項式時間アルゴリズムによって復元できることが示される．つまりこのプロトコルは所望の機能を実現する．

補題 4.3. 二分探索アルゴリズム（Algorithm 1) の出力において， V のサイズ N を知っている時，以下の 5 つのうちいずれか 1 つが分かれば，残りの 2 つは L の多項式時間で計算することが出来る．

- $rank(V,x)$
- 各ラウンドで x と比較した V の要素のインデックスの列 $i:=[i_1,\ldots,i_{|i|}]$
- 各ラウンドでの比較結果の列 $c:=[c_1,\ldots,c_{|c|}]$
- アルゴリズム内の変数列 $l:=[l_1,\ldots,l_{|l|}]$
- アルゴリズム内の変数列 $r:=[r_1,\ldots,r_{|r|}]$ [*5]

[*5] ランクが 0 かどうかで i,c,l,r のサイズは変わることに注意せよ．

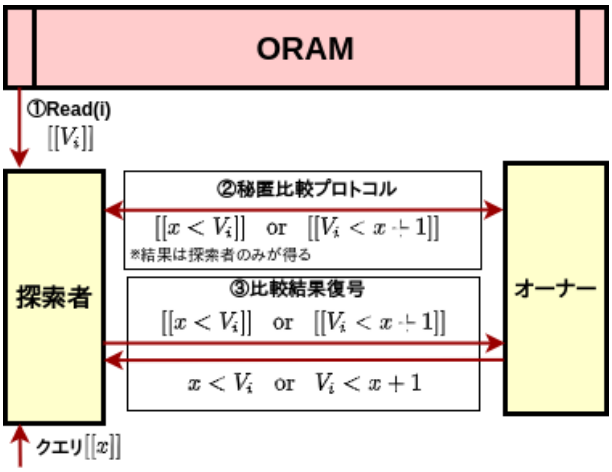

図 1 4 パーティー対称秘匿ランク決定プロトコルの二分探索内の比較 1 ラウンド分の概略図.

Proof. まず，$rank(V, x)$ から他の情報が得られることを示す．$x' = rank(V, x)$ ，$V' = [1, 2, \ldots, N]$ とする．このとき，Algorithm 1 を用いて V' 上での x' のランクを求めたときの出力を $\mathrm{rank}', i', c', l', r'$ とする．このとき (i', c', l', r') は (i, c, l, r) と一致する．なぜならば，Algorithm 1 では x, V_i はその大小比較の結果のみが挙動に影響しており，$x' < V'_i$ が同値であるため，(x, V) を x', V' に置き換えても挙動は変わらないからである．，実際には V' を作る必要はなく Algorithm 1 内の $V[i]$ を i に置き換えるだけで良いため，これらは $\Theta(L)$ 時間で計算可能である．

次に，l，r，i のいずれかのみから $rank(V, x)$ が計算可能であることを示す．l, r, i が空であるとき $rank(V, x) = 0$ であり，そうでないときは $rank(V, x) = l_L = r_L - 1 = i_{L+1} + 1$ である．

最後に，c から l, r が計算可能であることを示す．c が空であるとき l も r も空である．そうでないとき，以下の漸化式で定義される数列 l'，r' を考える．

$$(l'_0, r'_0) = (0, N)$$

$$l'_k = \begin{cases} \left\lceil \frac{l'_{k-1} + r'_{k-1}}{2} \right\rceil & (c_k = 1) \\ l^{k-1} & (\text{otherwise}) \end{cases} \quad (1 \le k \le L)$$

$$r'_k = \begin{cases} r'_{k-1} & (c_k = 1) \\ \left\lceil \frac{l'_{k-1} + r'_{k-1}}{2} \right\rceil & (\text{otherwise}) \end{cases} \quad (1 \le k \le L)$$

(l', r') は (l, r) と一致する．この漸化式は k が小さい順に計算することで $\Theta(L)$ 時間で計算可能である． □

5. 3 パーティー対称秘匿ランク決定プロトコル

本節では本章の初めに導入した外部委託対称秘匿ランク決定機能を実現する 3 パーティのランク決定プロトコルを紹介する．

定理 5.1. semi-honest モデルで，オーナー，クライアント，サーバーの 3 パーティからなり，ORAM, 対象秘匿比較のオラクルを使ったハイブリッドプロトコルで以下の機能を実現するランク決定プロトコルが存在する．

- クライアントは m 回のランク決定クエリに対する応答 $\mathrm{rank}(V, x_i)(0 \le i \le m)$ の情報のみを得る．
- オーナー，サーバーは何の情報も得ない．

□

対象秘匿比較の機能は SBC プロトコルによって実現され (補題 4.2)，ORAM の機能は SuccinctORAM を始めとした多くの既存プロトコルにより実現されているため，結合定理から以下の系が得られる．

系 5.2. semi-honest モデルで，オーナー，クライアント，サーバーの 3 パーティからなるプロトコルで以下の機能を実現するランク決定プロトコルが存在する．

- クライアントは m 回のランク決定クエリに対する応答 $\mathrm{rank}(V, x_i)(0 \le i \le m)$ の情報のみを得る．
- オーナー，サーバーは何の情報も得ない．

ここで，λ を対象秘匿比較プロトコル内で使う加法準同型暗号のセキュリティパラメータとし，ある多項式 poly に対して $m < \mathrm{poly}(\lambda)$ とする．

対象秘匿比較プロトコルで用いる想定の加法準同型暗号 (Pailliar 暗号，GDK 暗号) は IND-CPA 安全性を満たすが，これはセキュリティパラメータ λ に対して多項式時間では，選択平文攻撃に耐えるという内容であるため，$m < \mathrm{poly}(\lambda)$ が必要である．

6. PKUORAM を用いた簡潔化

対称秘匿ランク決定プロトコルではサーバーの実装として ORAM を仮定している．計算資源を抑えるためにも ORAM の空間計算量は小さくしたい．小野寺らによる SuccinctORAM [3] は，n bit の情報を格納する ORAM を簡潔に，つまり $n + o(n)$ bit で構築することに初めて成功したものである．しかし彼らの実装では，暗号化による bit 数の増加を定数と仮定していた．各データを ORAM 上に格納する前に AES 暗号のカウンターモードで暗号化するが，これによってブロックサイズのうちいくつかは冗長化のために使われる．[3] ではこの部分は常に 128 bit であるとして，無視できるサイズであると主張している．

実際にはこの冗長化部分のサイズはクエリ数に依存して大きくする必要がある．漸近評価をして簡潔であることを示す観点では，クエリ数の依存はなくしたい．

本研究では SuccinctORAM をベースとし，定期的に暗号鍵を変える機構を追加することでクエリ数に依存しない簡潔な空間計算量を実現した Periodical Key Update ORAM (PKUORAM) を構築した．このアイデアは SuccinctORAM 以外の ORAM にも適用可能である．

6.1 前提条件

PKUORAM は以下の仮定で挙げられる条件を満たすような ORAM をブラックボックスとして利用する．具体的な実装は指定しないが，少なくとも SuccinctORAM が以下の実装を満たす ([3] の Theorem 1)．

命題 6.1. B , n は $B = \omega(\log(n))$ と $0 < \exists c < 1$, $B = O(n^c)$ を満たす整数をとる変数とする．全データサイズが n bit，ブロックサイズが B bit の ORAM 実装で以下の条件を満たすものが存在する．

- サーバーの空間計算量が $n + o(n)$ bit である．
- 最悪バンド幅が $O(\log^2 n)$ である．

6.2 PKUORAM

本研究では以下の定理を満たす ORAM 実装を与えた．

定理 6.2. B , n は $B = \omega(\log(n))$ と $0 < \exists c < 1$, $B = O(n^c)$ を満たす整数をとる変数とする．全データサイズが n bit，ブロックサイズが B bit の ORAM 実装で以下の条件を満たすものが存在する．

- サーバーの空間計算量が $n + o(n)$ bit である．これはクエリの個数 M に依存しない．
- 最悪バンド幅が $O(\log^2 n)$ である．

命題 6.1 との違いは，空間計算量がクエリ数に依存しない条件が追加された部分である．実際 SuccinctORAM はデータサイズ n に関する寄与は $n + o(n)$ bit だが，クエリ数 M を加味すると少なくとも $\frac{n}{B} \log M$ bit は必要であり，M が充分大きい環境では簡潔性が成り立たなくなる．

PKUORAM のアイデアは，適当な頻度で暗号鍵を更新することで，見たことのある暗号文が出てきたとしても平文が同じと言えない状況を作ることである．まずは以下の補題を示す．

補題 6.3. $\mathcal{O}$ を命題 6.1 を満たす Tree-based ORAM とする． L を $\mathcal{O}$ のベースとなる完全二分木の葉の個数とする．また，左から i 番目 $(0 \leq i < L)$ の葉を l_i とする．$K_0, K_1, K_2, \ldots$ を暗号鍵の列とする．初め $\mathcal{O}$ の各要素は K_0 で暗号化されて格納されているとする． M を $M \geq L$ を満たす整数とする．

$\mathcal{O}$ の j 番目 $(L < j)$ のクエリを処理した後，ある $0 \leq i < L$ に対して $j \equiv i \mod M$ のとき， $\mathcal{O}$ の根から l_i までのパス上に格納されている暗号文を取得し，それらを K_{p-1} あるいは K_p で適切に復号し， K_p で暗号化し，サーバーに送り返すことを行う．ここで $\lfloor \frac{j}{M} \rfloor$ を p と置いた．

このとき同じ平文が同じ暗号鍵で暗号化される回数は $O(M \log^2(n))$ 回である．

Proof. ある暗号鍵 K_i を使って暗号化が行われうる期間を考える．$L \geq M$ より，左から j 番目 $(0 \leq j < L)$ の葉に該当するブロックが K_i で暗号化されるのは $Mi + j$ 番目のクエリの直後である．よって K_i で暗号化された暗号文が初めて格納されるのが Mi 番目のクエリの直後であり，最後に K_i で暗号化された暗号文が次の暗号文に書き換えられるのは $M(i+1) + L - 1$ 番目のクエリの直後である．よってこの間のクライアントからのクエリ $M + L$ 回に加えて，暗号を書き換えるための $2L$ 回 ($Mi, \ldots, Mi+L-1$, $M(i+1), \ldots, M(i+1) + L - 1$ 回目の各クエリの直後に鍵更新クエリを一回ずつ) を加えた $M + 3L$ 回のクエリの中で K_i を使った暗号化が行われる．1 回のクエリで暗号化し直す暗号文のブロック数はユーザーに送られてくるブロック数，つまり最悪バンド幅で上から抑えられる．命題 6.1 より，このブロック数は $O(\log^2(n))$ である．よって K_i を使って暗号化される回数は， $M > L$ に注意すると， $O(M \log^2(n))$ 回であることがわかる． □

[3] と同様に暗号化手法として AES 暗号のカウンターモードを用いれば，暗号化のための冗長 bit は暗号化を行う回数の対数の分だけあれば良い．つまり 1 ブロックあたり $O(\log(M) + \log\log(n))$ bit の冗長 bit があれば充分ということになる．

系 6.4. 補題 6.3 において $M = n^{O(1)}$ とすれば，暗号化した後の全データサイズは $n + o(n)$ bit で充分になる．

Proof. 1 ブロックを暗号化することで増える bit 数は $O(\log(n))$ bit になる．またブロック数は $\frac{n}{B}$ である．よって，全データに対して増える bit 数は $O(\frac{n \log(n)}{B})$ あれば充分である．命題 6.1 より， $B = \omega(\log n)$ なので，この値は $o(n)$ である．つまり，暗号化した後のデータサイズは $n + o(n)$ bit で充分になる． □

6.3 PKUORAM のクエリ秘匿性

PKUORAM では READ/WRITE クエリによるアクセスと，UPDATEKEY によるアクセスが発生する．前者は通常の ORAM と同様，サーバーの視点ではクエリの内容にかかわらず，1 クエリ毎にアトランダムに葉 l を選び，$\mathcal{O}$ の根から l のパス上のブロックを通信しているようにしか見えない．UPDATEKEY はランダムな要素こそないものの，過去に実行したクエリの個数でアクセスパターンが決定されるので，その挙動はいままでのクエリの内容には依存しない．よってクエリの内容を漏洩することはないので命題 2.6 の要件をみたすことになる．まとめると，サーバー視点では毎回ランダムな葉までのパスを通信するが，定期的に左の葉から順に走査するようなアクセスパターンが間に挟まるように見えることになる．

6.4 対称秘匿ランク決定プロトコルへの応用

前章で紹介して対称秘匿ランク決定プロトコルのサーバーの実装を PKUORAM にすることが可能である．ここで，ランク決定プロトコルでは DGK 比較プロトコルを実

行するためにデータを加法準同型暗号で暗号化し，さらにAES 暗号で暗号化することになる．よって PKUORAM のデータサイズは平文を加法準同型暗号で暗号化した後のデータサイズに対して簡潔であるという点に注意せねばならない．Paillier 暗号を用いた場合は，暗号の bit 数は平文の bit 数の 2 倍になり，さらに上で議論した冗長 bit が追加されることになる．元のデータサイズを n bit とすると $2n + o(n)$ bit になり，全体としては簡潔ではなく，コンパクトになる．

7. まとめ

本研究では，従来手法では実現されていなかった対数多項式時間で実行できる対称秘匿ランク決定プロトコルを提案した．また，SuccinctORAM に対し，暗号化による bit 数の増加分を加味したときにデータサイズがクエリ数に依存していくらでも大きくなってしまう点を指摘し，定期的に暗号鍵を交換することでクエリ数への依存を無くす解決策を示した．

参考文献

[1] Michael O Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptol. ePrint Arch.*, Vol. 2005, No. 187, 2005.
[2] Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel Rosu, and Michael Steiner. Outsourced symmetric private information retrieval. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 875–888, 2013.
[3] Taku Onodera and Tetsuo Shibuya. Succinct oblivious ram. In *35th Symposium on Theoretical Aspects of Computer Science*, 2018.
[4] O Goldreich. Basic application. foundations of cryptography, vol. 2, 2004.
[5] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pp. 223–238. Springer, 1999.
[6] Ivan Damgard, Martin Geisler, and Mikkel Kroigard. Homomorphic encryption and secure comparison. *Int. J. Appl. Cryptol.*, Vol. 1, No. 1, pp. 22–31, February 2008.
[7] Ivan Damgard, Martin Geisler, and Mikkel Kroigard. A correction to 'efficient and secure comparison for online auctions'. *Int. J. Appl. Cryptol.*, Vol. 1, No. 4, pp. 323–324, August 2009.
[8] Emil Stefanov, Marten Van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path oram: an extremely simple oblivious ram protocol. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 299–310. ACM, 2013.
[9] Srinivas Devadas, Marten van Dijk, Christopher W Fletcher, Ling Ren, Elaine Shi, and Daniel Wichs. Onion oram: A constant bandwidth blowup oblivious ram. In *Theory of Cryptography Conference*, pp. 145–174. Springer, 2016.
[10] Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, and Elaine Shi. Optorama: Optimal oblivious ram. *IACR Cryptology ePrint Archive*, Vol. 2018, p. 892, 2018.