修士論文 要旨

二分探索が可能な対称秘匿データ構造とその簡潔化

数理情報学専攻 48196209　小倉 拳
指導教員　定兼 邦彦 教授

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．このシナリオでは，(i) クエリの内容やクエリの応答の内容をサーバーやオーナーに対して秘匿したく，(ii) オーナーが持つデータに関してクエリの応答以上の情報をクライアントに対して秘匿したい．本研究では (i),(ii) のうち両方が秘匿されるプロトコルを**対称秘匿**プロトコル，片方のみが秘匿されるプロトコルを**非対称秘匿**プロトコルと呼んだ．

既存のプロトコルは，非対称秘匿なプロトコルがほとんである．紛失通信 [1] など対称秘匿なものもあるが，その場合はトレードオフとしてプロトコルの通信量や計算量がデータ数に対して多項式時間のものがほとんどであった．

対称秘匿で，対数多項式時間で二分探索が可能なデータ構造を提案した．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．

2　既存研究

2.1　秘匿比較可能暗号

定義 2.1. (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プロトコルが存在するものを**秘匿比較可能暗号**と呼ぶ．

DGK 比較プロトコル [2, 3] を用いることで，すべての加法準同型暗号は秘匿比較可能である．

2.2　ORAM

ObliviousRAM (ORAM)[4] 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．

ORAM は以下の安全性を満たす．

定義 2.2. (ORAM のクエリ識別不可能性) 任意の 2 つの同じ長さのクエリ列 $P^1 = [\mathrm{op}^1_1, \mathrm{op}^1_2, \ldots, \mathrm{op}^1_M]$，$P^2 = [\mathrm{op}^2_1, \mathrm{op}^2_2, \ldots, \mathrm{op}^2_M]$ 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 $\frac{1}{2}$ より有意な確率で当てることができない．

SuccinctORAM [5] は初めて簡潔な空間計算量 (もとのデータの bit 数を n として $n + o(n)$bit) を達成した ORAM である．しかし，暗号化による冗長化の影響は考慮されていなかった．一般に ORAM に用いられる暗号化手法は，同じ平文が毎回異なる暗号文に暗号化される必要がある．M 回暗号化するのであればすべての平文は対応する暗号文を M 種類は持っている必要があるため，暗号文空間は平文空間の M 倍以上のサイズでなければならない．よって表現に必要な bit 数は自ずと $\log M$bit 増えることになる．SuccinctORAM が持つ簡潔性は，この $\log M$ のファクターを定数とみなした上での簡潔性であった．

3　成果 1：対称秘匿ランク決定プロトコル

本研究ではランク決定クエリに応答できる対称秘匿なプロトコルを提案した．

定義 3.1. (ランク決定クエリ) 要素数 N の整数列 $V = (V_1, \ldots, V_N)$ に対する**ランク決定クエリ**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 $i \in [1, N]$ の個数を求めるクエリである．

情報の秘匿をしなくて良いならば，ランク決定クエリは $O(\log N)$ で応答できる：V を予めソートしておき，二分探索すればよい．二分探索アルゴリズムは内部で数値の比較を行いその結果次第で条件分岐し，行う計算を切り替えている．行う計算を切り替えていることは，

暗号化しようがないので，余計な情報として漏れてしまうことになる．これが原因で今までは対称秘匿なプロトコル内で条件分岐を行うアルゴリズムは用いられなかった．

3.1 提案プロトコルの構成

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**の 3 パーティである．

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 とし，それぞれの w_c bit 整数であるとする．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 w_c bit の整数 x を持っており，データオーナーが持つ整数列 V における x のランクを知りたい．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．

3.2 提案プロトコルのアルゴリズム

3.2.1 前処理

データオーナーは V のデータを加法準同型暗号で暗号化し，昇順に ORAM サーバーに格納する．公開鍵を K_p，秘密鍵を K_s とする．

3.2.2 ランク決定クエリ

クライアントは ORAM サーバー上のデータで通常通り二分探索を行う．二分探索の仮定で x と V_i の比較を行う必要がある．クライアントがアクセスできるのは $\mathrm{Enc}_{K_p}(V_i)$ のみであるため，K_s を知るオーナーと秘匿比較プロトコルを用いて大小結果の暗号文を得る．その後得た結果をオーナーに送り複合してもらう．この際，オーナーに比較結果の平文の情報が漏れるが，比較対象の 2 数を予めランダムに入れ替えることにより情報をマスクする．

3.2.3 秘匿性

サーバーは ORAM サーバーであるため何も情報を得ない．オーナーは DGK 秘匿プロトコルやその後の復号からは何も情報絵を無い．クライアントは，二分探索の過程で行う大小比較の結果の情報を得てしまうが，これはクエリの応答 $Rank(x)$ から復元可能なデータであるため，追加の情報にはなっていない．よって，このプロトコルは対称秘匿である．

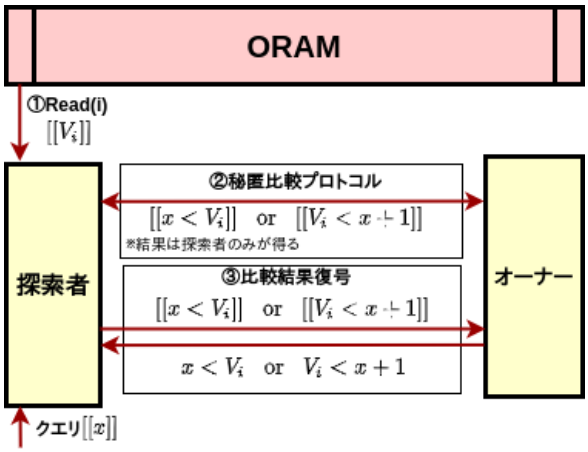

図 1. ランク決定クエリ応答時のアルゴリズム．

4 成果 2：クエリ数に依存しない簡潔な ORAM

定理 4.1. 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．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．

これは，SuccinctORAM [5] をベースとし，$N^{O(1)}$ クエリに一度，サーバー上のデータの暗号化鍵を更新する操作を挟み，暗号化による冗長化を防いで実現した．

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] 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.

[3] Ivan Damgard, Martin Geisler, and Mikkel Kroigard. A correction to 'efficient and secure comparison for on-line auctions'. *Int. J. Appl. Cryptol.*, Vol. 1, No. 4, pp. 323–324, August 2009.

[4] Rafail Ostrovsky. Efficient computation on oblivious rams. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pp. 514–523, 1990.

[5] Taku Onodera and Tetsuo Shibuya. Succinct oblivious ram. In *35th Symposium on Theoretical Aspects of Computer Science*, 2018.