マルチパーティ計算による Oblivious RAM のコスト削減と文字列検索の秘匿化への応用

数理情報学専攻 48-226203　諫山 航太
指導教員　定兼 邦彦 教授

1　背景

Oblivious RAM (ORAM) はデータの所有者であるクライアントが外部サーバへデータを安全に委託する手法のである．ORAM によって外部サーバへアクセスパターンを秘匿したまま委託データへアクセスすることができ，クラウドサービスへの応用も期待されている．その一方，ORAM はクライアント・サーバ間の通信コストの大きさが課題と言われている．特にマルチパーティ計算 (MPC) においては通信ラウンド数（通信回数）の削減が重要であり，ORAM においても同様である．この解決策としてマルチサーバ ORAM が提案された [7]．マルチサーバ ORAM は，クライアントの処理を複数サーバによる MPC が肩代わりすることでクライアント・サーバ間の通信コストを削減する．シングルサーバの ORAM の複数の方式のうち，ツリーベース型と呼ばれる方式はプロトコルがシンプルであり，最悪コストも他の方式と比べて実用性が高い．一方で最新のツリーベース型マルチサーバ ORAM [3] はサーバ間の 1 アクセスあたりの通信ラウンド数は $O(\log N)$ を要する．そこで本研究では，ツリーベース型マルチサーバ ORAM のサーバ間の通信ラウンド数の削減に取り組む．またその応用として，秘匿二分探索や秘匿パターンマッチングについても研究を行う．

2　準備

本研究の提案方式は 2 パーティの秘密分散法に基づいたマルチパーティ計算を利用する．本節では秘密分散法（特に加法的秘密分散法）とマルチパーティ計算を説明する．

2.1　加法的秘密分散法

位数 n の 2 パーティの加法的秘密分散法とは，以下の 2 つのプロトコルの組のことを言う．

- $\mathrm{Share}_n(x)$: $x \in \mathbb{Z}_n$ に対して，$[\![x]\!]^n \xleftarrow{\$} \{(x_0, x_1) \in \mathbb{Z}_n^2 \mid x_0 + x_1 = x\}$ を出力する．ただし $[\![x]\!]^n = ([\![x]\!]_0^n, [\![x]\!]_1^n)$ と記述する．
- $\mathrm{Reconst}_n([\![x]\!]^n)$: $[\![x]\!]^n$ を入力とし，$[\![x]\!]_0^n + [\![x]\!]_1^n$ を出力する．

$[\![x]\!]^n$ を x の加法的シェアと呼ぶ

2.2　マルチパーティ計算

マルチパーティ計算 (MPC) は複数の計算者が通信を行いながら計算を実行することである．ここで各計算者はそれぞれ秘密の入力を所持しており，計算の過程にて入力に関する情報を他者に漏らさない．本研究ではパーティがサーバであり，サーバの入力は加法的シェアとして与えられる．

3　本研究で扱う問題と成果

本研究ではツリーベース型のマルチサーバ ORAM のラウンド数削減を含む，以下の 3 つの問題を扱う．

3.1　ツリーベース型のマルチサーバ ORAM と Distributed ORAM (DORAM) のラウンド数削減

ここでは 1 クライアントと 3 サーバ（正確には 2 サーバと 1 ヘルパ）によるマルチサーバ ORAM を扱う．マルチサーバ ORAM は以下の機能を実現する確率的なプロトコルである．

定義 1 (マルチサーバ ORAM (Informal))**.** クライアントは 2 つのサーバにデータ $\boldsymbol{v} \in \mathbb{Z}_P^N$ の加法的シェア $[\![\boldsymbol{v}]\!]^P$ を委託する．サーバはヘルパの協力を受けながら，クライアントからの以下のクエリに応答する．

Access: サーバは $([\![\mathrm{op}]\!], [\![a]\!]^N, [\![x]\!]^P)$ をクライアントから受け取り，何らかの MPC を行って $[\![\boldsymbol{v}_a]\!]^P$ をクライアントへ送信する．op = write の時は，$[\![\boldsymbol{v}_a]\!]^P \leftarrow [\![x]\!]^P$ と更新する．

マルチサーバ ORAM の安全性は以下で定義される．

定義 2 (マルチサーバ ORAM の安全性（Informal）)**.** クライアントはクエリ $([\![\mathrm{op}_i]\!], [\![a_i]\!]^N, [\![x_i]\!]^P)$ を任意の回数呼び出す．この時，任意のサーバ $\mathcal{S}$ に対して，プロトコルの実行中にそのサーバ $\mathcal{S}$ が他の計算者から受け取るメッセージの分布が，クエリに依存しないとき，ORAM プロトコルを安全と定める．

以下でツリーベース型フレームワークと，提案方式の概要とを示す．

3.2 ツリーベース型 ORAM

サーバのデータ構造が完全二分木に基づく ORAM を総称してツリーベース型 ORAM と呼ぶ．各データには二分木の葉が一様ランダムに一つ割り当てられていて，常にその葉へのパス上に存在する．例として図 1 に提案方式のデータ構造を示す．

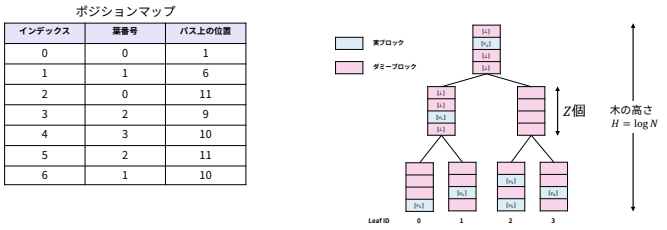

図 1. 提案方式のデータ構造．左側がクライアントのデータ構造であり，右側がサーバのデータ構造である．

ツリーベース型 ORAM のアクセスの流れは以下の通りである: (1) 所望するデータのパスからデータを得る (ReadPath)，(2) 取得したデータを二分木の根に再格納する (WriteBack)，(3) パスを一つ選び，割り当てた葉に矛盾しないようにパス上の全データを下へ移動させる．S3ORAM は逐次的な秘匿行列積によってデータを移動しており，$O(\log N)$ ラウンドの通信が発生していた．

3.3 提案方式の概要

秘匿置換作用，秘匿ソート [1] や秘匿単位ベクトル化 [2] 等のプロトコルを用いて，Evict でのデータの移動を一括処理することでラウンド数を削減する．パス上のブロックのデータの移動を表す置換のシェアを秘匿ソートや秘匿単位ベクトルを用いて並列に計算する．置換を秘匿置換作用でパスのデータ列に作用させ，データを一括移動させる．秘匿ソート [1] を用いる場合は $O(\log \log N)$ ラウンド，秘匿ソートを回避することで $O(1)$ ラウンドの ORAM を構成できる．

3.4 Distributed ORAM (DORAM) への適用

クライアントが存在せず，アクセスするインデックス a を各サーバがシェアとして入力する ORAM を DORAM と呼ぶ．アクセス結果 $\boldsymbol{v}_a$ も各サーバがシェアとして取得する．クライアントの平文計算が必要ないという性質から，前項で構成した Evict プロトコルはツリーベース型 DORAM の Evict としても使用できる．これにより既存方式 [4] の $O(\log^2 N)$ ラウンドを $O(\log N)$ に削減できる．

3.5 ツリーベース型マルチサーバ ORAM に基づく秘匿二分探索のラウンド数削減

単調増加列 $[\![\boldsymbol{v}]\!]$ を共有する 2 パーティと 1 ヘルパを考える．秘匿二分探索では，パーティはクエリ $[\![x]\!]$ に対して $\boldsymbol{v}_i \le x$ を満たす最大の $[\![\boldsymbol{v}_i]\!]$ を計算する．前節の Evict を適用することで，既存方式 [4] の $O(\log^2 N)$ ラウンドを $O(\log N)$ へと削減できる．

3.6 新たな秘匿パターンマッチングプロトコル

文字列 $\boldsymbol{v} \in \Sigma^N$ を加法的シェアとして共有する 2 パーティと 1 ヘルパを考える．秘匿パターンマッチングでは，パーティはクエリ文字列 $\boldsymbol{q} \in \Sigma^M$ の加法的シェアを入力として，$\boldsymbol{t}$ 中で $\boldsymbol{q}$ が出現する位置のシェアを全て求める．本研究では Sadakane [6] の平文アルゴリズムを提案秘匿二分探索プロトコルを用いて秘匿化することで，秘匿パターンマッチングプロトコルを新たに構成した．既存方式 [5] にて検索時に消費される乱数のサイズを大幅に削減すると同時に，乱数の入力への依存を取り除くことに成功した．

参考文献

[1] N. Attrapadung, G. Hanaoka, T. Matsuda, H. Morita, K. Ohara, J. C. N. Schuldt, T. Teruya, and K. Tozawa. Oblivious linear group actions and applications. In G. Vigna and E. Shi, editors, *ACM CCS 2021*, pages 630–650. ACM Press, Nov. 2021.

[2] N. Attrapadung, H. Morita, K. Ohara, J. C. N. Schuldt, and K. Tozawa. Memory and round-efficient MPC primitives in the pre-processing model from unit vectorization. In Y. Suga, K. Sakurai, X. Ding, and K. Sako, editors, *ASIACCS 22*, pages 858–872. ACM Press, May / June 2022.

[3] T. Hoang, A. A. Yavuz, and J. Guajardo. A multi-server oram framework with constant client bandwidth blowup. *ACM Trans. Priv. Secur.*, 23(1):1–35, Feb. 2020.

[4] M. Keller and P. Scholl. Efficient, oblivious data structures for MPC. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 506–525. Springer, Heidelberg, Dec. 2014.

[5] Y. Nakagawa, S. Ohata, and K. Shimizu. Efficient privacy-preserving variable-length substring match for genome sequence. *AMB*, 17(1):1–22, 2022.

[6] K. Sadakane. Compressed text databases with efficient query algorithms based on the compressed suffix array. In G. Goos, J. Hartmanis, J. Leeuwen, D. T. Lee, and S.-H. Teng, editors, *ISAAC 2000*, volume 14369, pages 410–421. Springer, Heidelberg, Dec. 2000.

[7] E. Stefanov and E. Shi. Multi-cloud oblivious storage. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 2013*, pages 247–258. ACM Press, Nov. 2013.