秘密分散法による秘匿データベース検索の設計と実装

森田 啓[1,a)]　Nuttapong Attrapadung[1]　Jacob C. N. Schuldt[1]

概要：機械学習などのデータ処理技術の進展やクラウドへのデータ保存の利用拡大により，新たな情報サービスの構築が可能となった．複数のユーザー達の機微情報をこうしたシステム（サーバー）に秘匿したまま保存し，必要に応じて秘匿したままデータ処理を施した後に，各ユーザーが検索語を秘匿したまま検索することで，個人の機微情報を無用に漏らすことなく有用な知識を得ることが可能となる．多者間秘匿計算はこうした機微情報が不特定多数の他者に漏洩しないようにするための解決策の一つだが，効率面では実用に十分ではなかった．本論文では，サーバー間で秘密分散された秘匿データベース検索を簡便に実現する手法を記し，有用性を議論する．

キーワード：秘匿計算，秘匿データベース検索，秘密分散共有法，マルチパーティ計算

Design and Implementation of Private Search over Shared Database

Hiraku Morita[1,a)]　Nuttapong Attrapadung[1]　Jacob C. N. Schuldt[1]

Abstract: Data processing techniques based on machine learning and cloud storage are now widely used, which has enabled us to develop novel application services. For example, users can store sensitive information such as DNA data in cloud storage, and cloud servers process the data securely; ultimately, each user obtains useful information out of the servers by sending a certain query without leaking any information beyond the output. Multi-party computation is one solution that can be used to prevent sensitive information from being leaked. However, its efficiency has not been good enough for practical use. In this paper, we propose a private search over a shared database and discuss its efficiency.

Keywords: Secure Computation, Private Search, Secret Sharing, Multi-party Computation

1. はじめに

秘匿計算は，データを秘匿したまま任意の関数を計算することが可能であり，研究および実利用の両面から今注目を集めている暗号技術の一つである．一般に，秘匿計算では計算量や通信コストを度外視すればいかなる関数も計算できるが，実用的なプロトコルを構成する上で，そうしたコストを実用に即して削減することが望まれる．コスト削減を達成し実用化が進むと，例えば，個人の DNA などの機微情報を秘匿したまま病気の発症リスク通知や体質改善指導のできるシステムの構築が可能となる．これは単に，機微情報を秘匿できるユーザー側が嬉しいだけでなく，システム提供側がユーザーの個人情報を保管しなくて済むという利点がある．

[1] 産業技術総合研究所　サイバーフィジカルセキュリティ研究センター
AIST CPSEC
[a)] hiraku.morita@aist.go.jp

1.1 背景

秘匿計算の応用先の一つとして，上述したようにプライバシー保護データを用いた統計分析や機械学習などの演算処理の秘匿化が挙げられる．このような高度で複雑な処理を秘匿する場合，それぞれの処理に必要な演算を加算・乗算・大小比較・ビット処理などの基本的な単位に分解し，その基本的な演算を秘匿したまま実行するプロトコルを組み合わせれば良い．

秘匿計算においては大きく分けて，

- 秘密分散ベース
- ガーブルド回路ベース
- 準同型暗号ベース

のプロトコル構成法が考えられる．本稿では，秘匿計算のプロトコル構成を秘密分散ベース，特に算術的秘密分散 [7] に基づいて，行う．ガーブルド回路ベース [11] や準同型暗号ベースなどのアプローチと比べて，秘密分散ベースのアプローチは比較的，計算コストや通信に係るバンド幅が小さくて済むという利点がある．ただし，一般に秘密分散ベースの方式は通信のラウンド数が多くなる傾向にあり，ラウンド数を削減したプロトコル構成が望まれている．

秘匿データベース検索． 統計分析や機械学習など高度な処理を実現するために，基本的な演算を組み合わせて構成される，より重要な処理の一つに，テーブル引きやテーブルの結合などのデータベースに関する処理がある．テーブル引きは複雑な秘匿計算を近似する用途に利用でき，また複数の参加者から集めたデータベースを秘匿したまま結合することでより豊富なデータベースを用いたデータ分析を行うことができる．

本稿では，テーブル引きをその特殊例に含む，秘匿データベース検索を扱う．ここで考えるデータベースは，キーワードとそれに対応するレコードの組からなり，秘匿データベース検索においては，クエリに対応するキーワードがデータベース内に含まれていた場合に対応するレコードをクエリした人だけが得ることができる．

クライアント-サーバーモデル． 秘匿計算の実社会での利用シーンの一つとして，クライアント-サーバーモデルと呼ばれる設定が注目されてきた．Araki ら [1] のような研究面のみならず，Cybernetica 社による Sharemind システムのような実利用の場面においても取り上げられている．このモデルは，N 台のサーバーと任意の数の（t 人の）クライアントからなる．それぞれのクライアントは秘密の入力 x_i を持ち，それを N 台のサーバーに秘密分散しており，それらサーバーは入力のシェアに対して互いに協力して計算を行い，その計算結果 $f(x_1, \ldots, x_t)$ をクライアントに返す．このモデルは，N 人の参加者が入力を持たない $(N+t)$ パーティ MPC と考えることもできる．このモデルの特徴は，実社会のサービス事業に適合した設定であることにある．

本稿ではサービス提供者とそのサービスを利用するユーザーをクライアントと捉え，秘匿計算を行う 2 台の計算サーバーをサーバーと捉える 4 者からなるモデルを考える．より具体的には，サービス提供者はオリジナルのデータベースを持ち，それを秘匿したまま 2 台の計算サーバーに保管する．ユーザーは，2 台のサーバーに対してクエリを秘匿したまま送り，サーバーはこのクエリに対応するレコードを秘匿したままユーザーに返す．なおこの設定は，サービス提供者が 2 台のうち片方の計算サーバーの役割を果たす 3 者モデルに読み替えることもできる．ただしその場合は，ユーザーからのクエリがデータベースのエントリーのうちどれを指すのかサービス提供者に情報が漏れないようにするために，秘匿したまま保管されるデータベースに秘匿シャッフルを事前計算として施す必要がある．記述の簡便さのために本稿では 4 者モデルとして記す．

1.2 貢献

提案する秘匿データベース検索は，上述したように 2 台の計算サーバーが秘匿データベースを保持し，ユーザーからのクエリに対応するシェアされたレコードを返す．愚直な方法を用いると，ユーザーからの検索クエリに対して各サーバー自身が保持している秘匿データベースの中から一致するキーワードを探すためには，データベースのエントリー数に関して対数オーダーの通信が必要となる．しかし我々の提案方式においては，サーバー間での通信なしに秘匿データベース検索を構成することができる．

構成のテクニック． 提案するプロトコルは Laur ら [9] のテーブル結合のプロトコルの構成のアイディアに基づいているため，[9] の構成法について簡単に記す．まず，オリジナルのデータベースは $i \in \{1, \ldots, n\}$ に関して $\{(k_i, r_i)\}$ となっているとす

る．ここで n はデータベースのエントリー数とする．[9] では，サーバー $s \in \{1,2\}$ はキーワードもレコードもシェアされたデータベース $\{([\![k_i]\!]_s, [\![r_i]\!]_s)\}$ を持つ．2 サーバーは協力して Oblivious AES を用いて元のキーワードをすべて，ある共通鍵 κ を用いた AES で暗号化した値に変換する．つまり，AES で暗号化されたキーワードと元のレコードのシェアからなる秘匿データベース $\{(\mathsf{AES}_\kappa(k_i), [\![r_i]\!]_s)\}$ を持つ[*1]．ユーザーも共通鍵 κ を知っていれば，クエリ x をこの鍵で暗号化した $\mathsf{AES}_\kappa(x)$ をサーバーに送る．2 サーバーはともに自身の持つ秘匿データベースからこの暗号化キーワードに一致するものがあれば，対応するレコードのシェアをユーザーに送り返す．ユーザーは送られてきた 2 つの値を用いて復元すると，クエリ x に対応するレコードの値が得られる．

上記のプロトコルのうち，Oblivious AES は通信ラウンドが多く，実用性の観点から使い勝手が良くない．そこで，本稿では 2 サーバーが Oblivious AES を走らせる代わりに，もともとのデータベースを持っているはずのサービス提供者が自身でキーワード達だけ AES で暗号化し，レコード達をシェアの形にして 2 サーバーに送信する．安全性は AES に依存することになるが，検索には通信を必要としないプロトコルとなる[*2]．

1.3 関連研究

自然言語処理分野で使われるテキストコーパスのような，あまり構造的でないデータベースに対して，クエリを含む各文書をすべて出力するような秘匿ドキュメント検索と呼ばれる技術があり，これは [8] に詳しい．

本稿で扱うような，法律文書である LexisNexis データベースのように，キーワードとレコードからなる比較的構造的なデータベースに対して，クエリに対応するレコードを返す秘匿データベース検索も研究されてきた [4], [5]．我々の知る限り，[5] が既存の秘匿データベース検索の中で state of the art であるが，準同型暗号もしくは紛失通信という重い処理を必要とする．本稿では，安全性を計算量的な安全性にするかわりに，こうした重い処理をなくし実用的なプロトコルを構成した．

テキスト T に対してパターン p をクエリし，T の中の p の位置を秘匿したまま出力する秘匿テキスト検索は，実用を目指した多くの研究がなされてきた．代表的なものとして，LiLiP [3]，Prisearch [10]，5PM [2]，Frikken [6] などが挙げられる．

2. 準備

本章ではまず，本稿で用いる記法を記し，提案方式で採用する算術秘密分散法について記す．

2.1 記法

n を整数とする．本稿で扱う値は特に断らない限り $\mathbb{Z}_{2^n}$ の要素である．AES は NIST によって標準化された共通鍵暗号方式である．AES の鍵長の選択により，本稿の実装では特に $n = 128$ とする．

$\mathbb{T}_{2^n}$ を完全二分木とし，その葉は整数 0 から $2^n - 1$ に対応する．$\mathbb{S}_{2^n}$ により木 $\mathbb{T}_{2^n}$ の全てのノードを表し，ノード $w_{i,j}$ は i 層目（葉が 0 層目，根が n 層目）の左から j 番目のノードを表す．なお (i,j) は $i \in [0,n]$ および $j \in [0, 2^{n-i} - 1]$ である．

2.2 算術秘密分散法

算術秘密分散法 [7] では，サーバー 1 およびサーバー 2 の間での $x \in \mathbb{Z}_{2^n}$ の算術型シェアは，それぞれ $[\![x]\!]_1 = r$ および $[\![x]\!]_2 = x - r$ と表される．ここで $r \leftarrow_\$ \mathbb{Z}_{2^n}$ は乱数である．$\mathsf{Share}(\cdot)$ をこのようなシェアを出力するアルゴリズムであるとすると，$([\![x]\!]_1, [\![x]\!]_2) \leftarrow \mathsf{Share}(x)$ と書ける．$[\![x]\!]_i = x_i$ および $[\![y]\!]_i = y_i$ であるような $[\![x]\!] = ([\![x]\!]_1, [\![x]\!]_2)$ および $[\![y]\!] = ([\![y]\!]_1, [\![y]\!]_2)$ に対して，加算 $[\![x]\!] + [\![y]\!]$ を計算するには，それぞれのサーバー i が $[\![x]\!]_i + [\![y]\!]_i \leftarrow x_i + y_i$ を計算する．さらに，減算は $[\![x]\!]_i - [\![y]\!]_i \leftarrow x_i - y_i$ を計算し，公開された値 c との乗算は $c \cdot [\![x]\!]_i \leftarrow cx_i$ を計算すればよい．

3. 秘匿データベース検索

本章では我々の提案する秘匿データベース検索について説明する．クライアント C と，2 つのサーバー S1 および S2 がいる状況を考える．以下に，秘匿データベース検索の入出力を記す．

- 入力：
  - (C) 検索クエリ
  - (S1, S2) キーワードとそれに対応するレコードのペア (key, record) からなるデータベース
- 出力：
  - (C) クエリに対応するレコード/$\perp$

[*1] 2 サーバーが持つ秘匿データベースはキーワード部分は同じもの（もとのキーワードの暗号文）となる．

[*2] 暗号化された検索クエリに対応するシェアされたレコードを返すときのみ通信を行う．

– (S1, S2) なし

[9] はデータベースを再構成するのに Oblivious AES を用いており，その処理が比較的重いと考えられる．そこで我々の提案方式では，オリジナルのデータベースを持っているサービス提供者が，再構成したあとのデータベースに相当するデータベースを構成し，それを 2 サーバーに送信しているというのが特徴である．サービス提供者が元のデータベースを持っていると考える今回の設定の場合，ユーザーとこのサービス提供者間で前もって秘密鍵 sk を共有しておく必要があることに注意．

図 1 に提案方式の手続きについて示す．本稿では，サービス提供者がオリジナルのデータベース

$$\begin{pmatrix} k_1 & r_1 \\ \vdots & \vdots \\ k_N & r_N \end{pmatrix}$$

を持っているとする．ここで $i \in \{1, \ldots, N\}$ に対して k_i はキーワード，r_i はそれに対応するレコードを表すとする．サービス提供者は，このデータベースに対して，クライアントと共有している鍵 sk を用いて PRF を実行し，

$$\begin{pmatrix} \mathsf{PRF}_{\mathsf{sk}}(k_1) & [\![r_1]\!]_1 \\ \vdots & \vdots \\ \mathsf{PRF}_{\mathsf{sk}}(k_N) & [\![r_N]\!]_1 \end{pmatrix}$$

をサーバー 1 に送り，

$$\begin{pmatrix} \mathsf{PRF}_{\mathsf{sk}}(k_1) & [\![r_1]\!]_2 \\ \vdots & \vdots \\ \mathsf{PRF}_{\mathsf{sk}}(k_N) & [\![r_N]\!]_2 \end{pmatrix}$$

をサーバー 2 に送る．ここまでが事前計算であり，以下に秘匿データベース検索プロトコルの実際の手順を記す．

クライアントはクエリ x に対し，鍵 sk を用いて $\mathsf{PRF}_{\mathsf{sk}}(x)$ を計算しサーバー 1 および 2 に送信する（手順 (1) および (1′)）．クエリを受け取った各サーバーは，受け取った暗号化クエリに対応するレコードをクライアントに返す（手順 (2) および (2′)）．対応するものがない場合には $\perp$ を返す．サーバーからの返答を受け取ったクライアントは，それらを用いて所望のレコードを復元する．

4. 簡易実装

提案方式を zsc（ZenmuTech Secure Computation Module）ライブラリ [12] を用いて実装した．本ライブラリは，秘密分散された値を用いた秘匿計算を行うライブラリであり，各クライアント・サーバー間の通信も含めた処理を実行可能である．通信における同期処理をバックグラウンドで行うため，開発者は実装の際に通信に係る処理を気にすることなく，主となるコーディングに注力することができる．なお今回は，クライアントおよびサーバーを一台のマシンの上で仮想的に動作させている．zsc について詳しくは [12] を参照のこと．

以下に測定環境を記す．

- OS： macOS 10.14.6
- CPU： 2.5 GHz Intel Core i7
- プログラミング言語：Python 3.7.3
- 利用したライブラリ等：zsc 0.3.0, numpy 1.16.4

使用したデータベースは東京都の 2018 年 8 月 21 日午前 1 時から 2019 年 8 月 22 日午前 0 時までの一時間ごとの気温のデータで，エントリー数は 8784 である．例えば，キーワードとして「2019/8/21 13:00:00」，レコードとしてその時間に対応する「30.2」という数字が記録されている．事前準備としてサービス提供者はオリジナルのデータベースから構成した秘匿データベースをサーバーに送る必要がある．本実験においては一つのエントリーを送信するのに 1–2.5 ミリ秒かかることが観測された．なお，オンラインの秘匿検索について 10 回の施行の平均は約 1.3 ミリ秒であった．WAN の設定においては通信遅延が距離に応じて 30–70 ミリ秒ほどかかるため，秘匿データベースを，エントリーを並列化して送信するのに 30–70 ミリ秒ほどかかること，また秘匿検索には通信 2 ラウンド分の 60–140 ミリ秒ほどかかることが推察される．

5. おわりに

本稿では，統計分析や機械学習の秘匿化に向けて重要となるテーブル関連の処理のうち，データベース検索を対象とし，サーバー間の通信が必要ではない高速な秘匿データベース検索を算術秘密分散ベースで構成した．クエリを受けたサーバーは秘匿一致判定などの複雑な処理を行う必要がなく，平文の検索を行うだけでよい．よって，秘匿検索にかかる時間はクエリの送信と返答の送信にかかる時間が支配的となり，おおむね 60–140 ミリ秒程度と推察される．

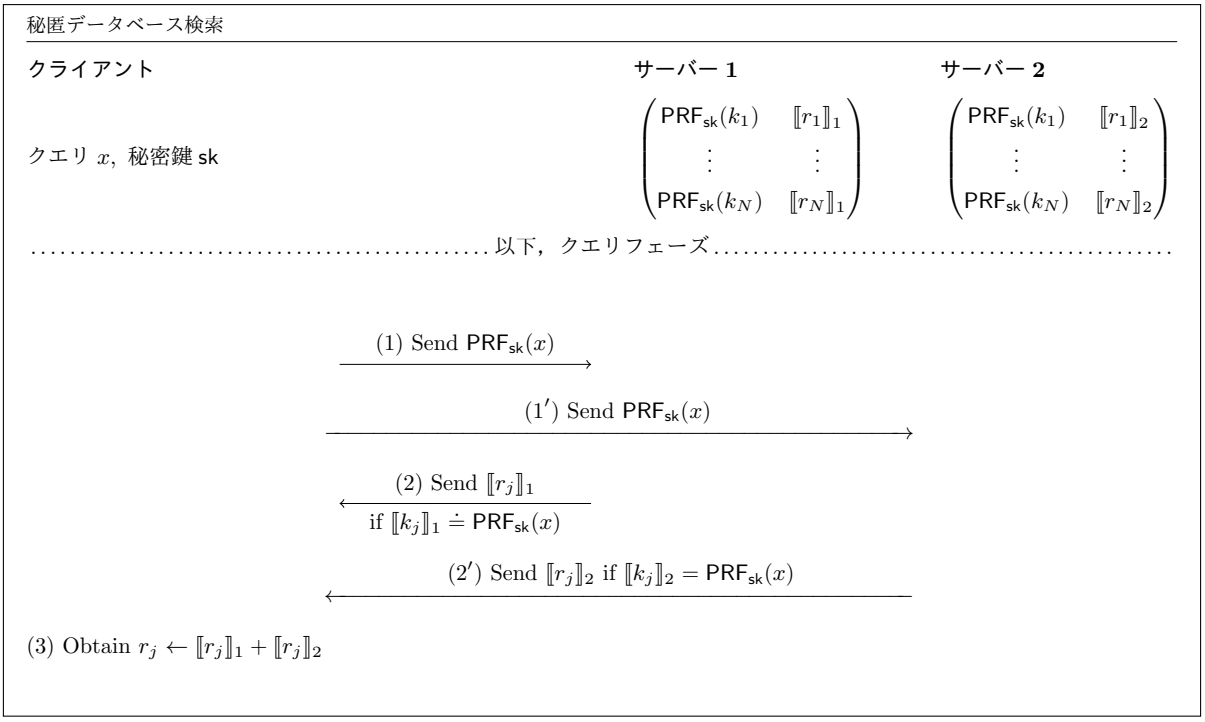

図 1 提案方式の手続きの概略

謝辞

本研究は JST CREST JPMJCR19F6 の支援を受けて行われた．

参考文献

[1] Araki, T., Furukawa, J., Lindell, Y., Nof, A. and Ohara, K.: High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority, *CCS 2016*, pp. 805–817 (online), DOI: 10.1145/2976749.2978331 (2016).

[2] Baron, J., Defrawy, K. E., Minkovich, K., Ostrovsky, R. and Tressler, E.: 5PM: Secure pattern matching, *Journal of Computer Security*, Vol. 21, No. 5, pp. 601–625 (online), DOI: 10.3233/JCS-130481 (2013).

[3] Darivandpour, J. and Atallah, M. J.: Efficient and secure pattern matching with wildcards using lightweight cryptography, *Computers & Security*, Vol. 77, pp. 666–674 (online), DOI: 10.1016/j.cose.2018.01.004 (2018).

[4] Ferragina, P. and Grossi, R.: Fast String Searching in Secondary Storage: Theoretical Developments And Experimental Results, *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, 28-30 January 1996, Atlanta, Georgia, USA.*, pp. 373–382 (1996).

[5] Freedman, M. J., Ishai, Y., Pinkas, B. and Reingold, O.: Keyword Search and Oblivious Pseudorandom Functions, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, pp. 303–324 (2005).

[6] Frikken, K. B.: Practical Private DNA String Searching and Matching through Efficient Oblivious Automata Evaluation, *Data and Applications Security XXIII, 23rd Annual IFIP WG 11.3 Working Conference, Montreal, Canada, July 12-15, 2009. Proceedings*, pp. 81–94 (2009).

[7] Goldreich, O., Micali, S. and Wigderson, A.: How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority, *STOC 1987*, pp. 218–229 (online), DOI: 10.1145/28395.28420 (1987).

[8] Hazay, C. and Lindell, Y.: *Efficient Secure Two-Party Protocols - Techniques and Constructions*, Information Security and Cryptography, Springer (2010).

[9] Laur, S., Talviste, R. and Willemson, J.: From Oblivious AES to Efficient and Secure Database Join in the Multiparty Setting, *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, pp. 84–101 (2013).

[10] Riazi, M. S., Songhori, E. M. and Koushanfar, F.: PriSearch: Efficient Search on Private Data, *Proceedings of the 54th Annual Design Automation Conference, DAC 2017, Austin, TX, USA, June 18-22, 2017*, pp. 14:1–14:6 (2017).

[11] Yao, A. C.: How to Generate and Exchange Secrets (Extended Abstract), *FOCS 1986*, IEEE Computer Society, pp. 162–167 (online), DOI: 10.1109/SFCS.1986.25 (1986).

[12] 石田祐介， 國井淳，桶谷純一，大畑幸也，花岡悟一郎：スケーラブルな二者間秘匿計算のサーバー構成と効率的な通信方法の実装，*PWS 2019* (2019).