秘密分散に基づく秘匿全文検索

中川 佳貴[1,a)]　大畑 幸矢[2,b)]　清水 佳奈[1,c)]

概要：秘密分散法に基づき，クエリと文書を互いに秘匿したまま全文検索する手法を提案する．計算には，クエリ保有者，文書保有者，結託の恐れのない二人の委託計算者が参加する．クエリ保有者と文書保有者のそれぞれが事前にクエリと文書のシェアを作って委託計算者に送り，委託計算者は検索結果のみをクエリ保有者に返す．提案手法では，FM-index と呼ばれる全文索引により文書を検索しやすい構造に変換する．これにより，委託計算で必要となる計算量，通信量，通信ラウンド数の全てが文書の長さに依存せず，クエリの長さのみに比例する．提案手法を実装し，文書とクエリの長さがそれぞれ 1000 万と 100 のゲノム配列検索を行ったところ，委託計算者の計算時間は 1CPU の利用でわずか 0.30 秒，ラウンド数は 202，通信量は 7.13KB であった．文書保有者の事前計算は文書長とクエリ長の積に線形だが，並列化により効率的に計算できるため，実装上の工夫をせずとも 10CPU の利用で 15 分以内に留まる．提案手法は文字列検索の他に，オートマトンによるパターンマッチや木構造の走査等にも応用でき，多様なデータ検索の高速化に貢献することが期待される．

キーワード：全文検索，秘密分散，FM-index，マルチパーティ計算，プライバシ保護

Privacy-Preserving Full-Text Search Based on Secret Sharing

Yoshiki Nakagawa[1,a)]　Satsuya Ohata[2,b)]　Kana Shimizu[1,c)]

Abstract: We propose a secure full-text search protocol based on secret sharing. We consider a case in which a database holder and a querier send shares of the database and the query to two non-colluding computing nodes and they return the result to the querier. By using an efficient full-text index called FM-index, all the time, communication and round complexities of the protocol become linear to the length of query and do not depend on the length of database. In the experiment using a genomic sequence of length 10 million and a query sequence of length 100, the CPU time, communication round, and data transfer size for a single query were only 0.3s, 202 and 8.69KB. The building block of the protocol can also be applied to other applications such as a deterministic finite automaton matching and a tree traversal, and is expected to contribute to various real word problems.

Keywords: Full-text search，Secret sharing，FM-index，Multi-party computation，Privacy preserving

1. はじめに

複数の文書から特定の文字列を検索するタスクを**全文検索**と呼ぶ．全文検索の応用は広範囲に及び，自然言語による文書のみならず，ゲノム配列，時系列データ，プログラムの実行バイナリ等，文字列として表現することのできる多種多様なデータの解析において重要な役割を果たしている．実社会においては，検索文字列（以下，クエリ）と文書の双方が秘匿性の高い情報であることはしばしばあり，クエリと文書が別々の持ち主のものであった場合は，いずれかの秘密を犠牲にして検索を実行するか，検索の実行を諦めるより他はない．本研究ではこのような問題を解決するため，クエリと文書を互いに秘匿したまま全文検索を行

[1] 早稲田大学 (Waseda university)
[2] 産業技術総合研究所 (Nationla institute of advanced industrial science and technology)
[a)] nakapandayo@fuji.waseda.jp
[b)] satsuya.ohata@aist.go.jp
[c)] shimizu.kana@waseda.jp

う**秘匿全文検索**の開発に取り組む．

詳細は 2 章にて述べるが，秘匿全文検索と同様，もしくは類似の問題に対する手法は，主として準同型暗号に基づくものが主流であった．準同型暗号は，検索者と文書保有者の二者間のみで秘匿検索を実現できる利点があるが，一演算当たりの計算コストが非常に高く，巨大な文書を実時間で検索することが難しい．これに対して本研究では，二者間の秘密分散法と高速な全文索引法として知られる FM-Index を組み合わせる手法を開発し，過去の研究と比較して，クエリ投入から検索終了までの時間を 800 倍以上高速化することに成功した．以下に，本研究の貢献を記す．

(1) FM-Index では，あらかじめデータを検索しやすいように索引化した参照テーブルに対して，要素の参照と，さらに，参照した要素の値の順位に相当する要素を参照する必要がある．本研究では，参照テーブルを秘密分散したまま，中間結果を復元することなく上述の機能を実現する**秘密分散型再帰的紛失通信**を開発した．この技術はテーブルの設計を変えることによって，木構造の検索やオートマトンなど，多様なデータ構造の検索に応用することができる．詳細は 4 章を参照されたい．

(2) 秘密分散型再帰的紛失通信に基づき，秘匿全文検索を実現するプロトコルを開発した．我々のプロトコルの特徴は，秘匿全文検索のオンライン計算量及び通信量がクエリの長さ ℓ に依存し，文書の大きさ N に依存しないことである．また，通信ラウンド数も ℓ にのみ依存し N には依存しない．詳細は 4 章を参照されたい．

(3) 提案手法を実装してゲノム配列の実データを用いた実験を行い，実際の計算量と理論性能がよく一致することを示した．詳細は 5 章を参照されたい．

本稿の構成は以下の通りである．まず 2 章で関連研究を示す．3 章では本研究で用いる暗号要素技術である二者間加法秘密分散とそれに基づく秘匿計算，FM-Index 法を紹介する．4 章では我々が提案する秘密分散型再帰的紛失通信とそれを用いた秘匿全文検索アルゴリズムを説明する．5 章ではゲノム配列の実データを用いた実験結果を示す．6 章において本研究のまとめと今後の方向性について述べる．

2. 関連研究

本章では，要素技術としての秘匿計算とその応用（特にゲノムデータ処理）に関して，それぞれ関連研究を紹介する．

秘匿計算：データを隠したまま計算を行う技術である秘匿計算 [12], [26] は古くから研究されてきたが，計算量及び通信量が非常に多く，実用とは程遠かった．しかし近年，プロトコルの改善が進んだことから現実的な処理に耐えうるものも出現している．秘匿計算の要素技術としては（完全）準同型暗号 [10], [20]，ガーブルド回路 [26]，秘密分散 [22]，セキュアハードウェア [6] などが挙げられる．ハードウェア支援型の秘匿計算は非常に高速で，確保できるメモリ量の制限などはあるものの，平文での計算と遜色ない処理性能を達成できることも多い．一方で，ハードウェア設計の正しさ（とその製造工程）を信頼する必要があり，また現状ではサイドチャネル攻撃に対して脆弱であるという報告もある．ハードウェア支援型以外では，実装の工夫などによって高いスループットを出せることがわかってきた秘密分散に基づく秘匿計算に注目が集まっており，本研究でもこのアプローチを採用する．設定としては，2-out-of-3 秘密分散を用いた三者間秘匿計算に関する研究 [1], [5] が近年では多い．これは後述する事前計算などの点で二者間秘匿計算と比較して性能面の優位性があるからであると考えられるが，必要なハードウェア資源が二者間計算と比べて多く，また結託しないよう監視する必要があるペアも増えるため，運用のコストは二者間秘匿計算と比較して増大する．その他の研究結果としては，事前計算が効率的なモデルの提案 [15], [17]，異なるタイプのシェアを行き来して効率化を図った研究 [7], [16]，具体的なプロトコル（回路）の効率化 [4], [18], [19] などが挙げられる．

秘匿ゲノムデータ処理 [27]：ゲノム情報は個人を識別する ID と見なすことができると同時に，遺伝によって近縁者のプライバシにも影響が及ぶ，今後解析が進むにつれて重要な情報が逐次明らかになっていくという性質を持つ．それ故にゲノムは究極の個人情報と呼ばれることがあり，取り扱いが難しい．その結果，ゲノム情報は特定の組織内に留まり，その利活用が十分に進んでいるとは言えないことが問題視されている．例えば，ゲノム研究に関する情報の共有と活用を目的とした国際組織である Global Alliance for Genomics and Health (GA4GH) が 2013 年に発足し，その中で Beacon プロジェクト（ゲノム上の位置と塩基の種類を指定するとデータベース側がその保有状況を返答する仕組み）が進められているなど，情報開示に向けた努力がなされている．しかし，このような情報開示の取り組みはプライバシ保護の重要度や営利組織の利害も関係してくるため，常に成功するわけではない．このような背景から，ゲノムデータに関して，そのプライバシ保護と利活用の両立を目的とした研究開発が数多く進められている．用いられているツールは圧倒的に準同型暗号が多いが，秘密分散やガーブルド回路を用いている研究も少しずつ増えてきている．テーマとしてはカイ二乗検定などの一般的な統計計算を行った研究（例えば [14]）や，ゲノム文字列の（近似）編集距離を秘匿計算する研究（例えば [2], [21]）が多い．本研究で扱う文字列検索にもいくつか既存研究がある．2009 年には最長共通部分列を求める秘匿計算アルゴリズムが提案された [9], [13] が，実装評価はなく具体的な性能は不明である．2016 年以降に発表された文字列検索に関する研究成果 [23], [24], [25] では具体的な性能評価がなされてい

る．しかし，これらの結果は（完全）準同型暗号に基づくものであり，その計算コストの高さから巨大文書の検索に耐える性能が出ているとは言えない．本研究では秘密分散を用いた再帰的紛失通信の設計により，この課題の克服を試みる．

3. 準備

3.1 加法秘密分散に基づく秘匿計算

データの情報を秘匿する技術として加法秘密分散を用いる．秘密分散はデータを複数に分割し別の場所で保管することでデータを秘匿する手法である．本研究では二者間秘匿計算を想定し，2 つに分割したデータを 2 つの計算サーバーにそれぞれ保存し，分割データの状態で計算を行う．以降，分割データのことをシェアと呼び，データ x に対する 2 つのシェアを $[\![x]\!]_0, [\![x]\!]_1$ と表す．本稿で登場するのはほとんどの場合は算術シェアであるため，この記法で算術シェアを表すことにする．論理シェアの場合は $[\![x]\!]_0^B, [\![x]\!]_1^B$ と書くことにする．

二者間秘匿計算を行うにあたり，2 つのアルゴリズム Share 及び Reconst が必要になる [12]．分散アルゴリズム Share は n ビットで表される値 $x \in \mathbb{Z}_{2^n}$ を入力とし，2 つのシェア $([\![x]\!]_0, [\![x]\!]_1)$ を出力する．再構成アルゴリズム Reconst は，$([\![x]\!]_0, [\![x]\!]_1)$ を入力とし，x を出力する．それぞれのアルゴリズムの詳細は次のようになる．

- Share(x): ランダムな $r \in \mathbb{Z}_{2^n}$ を選び，$[\![x]\!]_0 = r$ と $[\![x]\!]_1 = x - r$ を出力する．
- Reconst$([\![x]\!]_0, [\![x]\!]_1)$: $[\![x]\!]_0 + [\![x]\!]_1$ を出力する．

ここで，シェア上での計算を考える．x と y に対して，パーティ P_0 と P_1 がそれぞれ $([\![x]\!]_0, [\![y]\!]_0)$, $([\![x]\!]_1, [\![y]\!]_1)$ を持っているとする．加算 ADD(x, y) は単にシェア同士を足すだけで実現可能である．乗算 MULT(x, y) については様々な実現方法があるが，本研究では Beaver triple を用いた手法 [3] を採用する．Beaver triple とは $a_0 + a_1 = a$，$b_0 + b_1 = b$，$c_0 + c_1 = c$，$ab = c$ を満たす 3 つ組乱数 (a_0, b_0, c_0) 及び (a_1, b_1, c_1) のことであり，これらの値を事前に共有しておく．P_0 は $[\![x]\!]_0 - a_0$ と $[\![y]\!]_0 - b_0$，P_1 は $[\![x]\!]_1 - a_1$ と $[\![y]\!]_1 - b_1$ をそれぞれローカルで計算して送りあう．その後，お互いにローカルで $x' = \text{Reconst}([\![x]\!]_0 - a_0, [\![x]\!]_1 - a_1)$ 及び $y' = \text{Reconst}([\![y]\!]_0 - b_0, [\![y]\!]_1 - b_1)$ を実行し，P_0 は $[\![z]\!]_0 = x'y' + x'b_0 + y'a_0 + c_0$，$P_1$ は $[\![z]\!]_1 = x'b_1 + y'a_1 + c_1$ を計算する．すると，この $([\![z]\!]_0, [\![z]\!]_1)$ は xy のシェアとなる．この計算は加算を排他的論理和とすることで論理ゲート（XOR および AND）の計算も可能となる．論理否定 NOT は片方のパーティが自身の論理シェアの値を反転することで，論理和 OR は入力を両方否定した状態で AND を計算し，出力をさらに否定することで実現できる．本研究ではこれらの基本的なゲートの他に，一致判定プロトコル Equality(x, y) を用いる．これは $x = y$ であれば 1，そうでなければ 0 の論理シェアを出力するプロトコルである．これは [19] によると，多入力の AND ゲートを使用することで $n \leq 8$ の場合は 1 ラウンド，$8 < n \leq 64$ の場合は 2 ラウンドの通信で計算が可能である．誌面の都合上，本稿では具体的な構成法は割愛する．

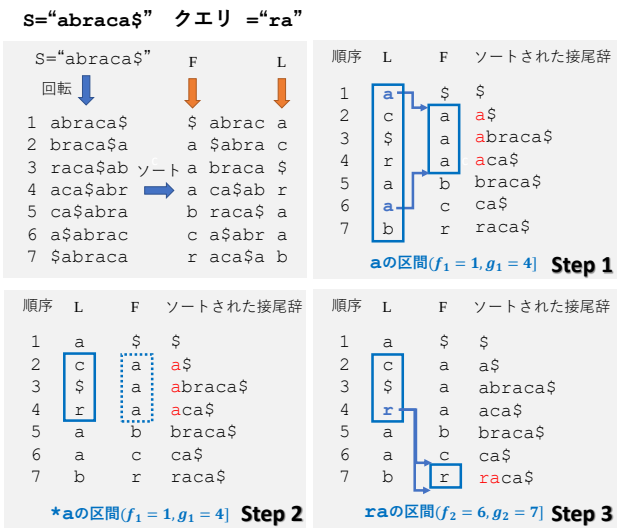

図 1 FM-Index 法の概要

本稿では，秘匿計算を実行する委託計算者はプロトコルから逸脱しないものと考える．これを semi-honest モデルと呼ぶが，上で説明したゲートの計算アルゴリズムは semi-honest 安全性を持つことが証明されている [3], [12]．また，これらのゲートの組み合わせで構成されるプロトコルの semi-honest 安全性は Composition Theorem [11] より成立するため，本稿ではこれ以降プロトコルの安全性に関して詳細には議論しない．

Beaver triple は秘匿計算を実行する委託計算者間で準同型暗号や紛失通信を用いて生成する手法（例えば [7]）が知られている．しかし，この手法は計算量および通信量が多い．そこで Mohassel ら [17] は，計算を委託するクライアントが Beaver triple を生成し，委託計算者にそれぞれ送信するモデル（Client-aided モデル）を提案した．このモデルはクライアントの計算時間が増大すること，秘匿計算を実行する前にクライアントから委託計算者へ送信するデータ量が増加すること，クライアントが委託計算者のどちらかと結託すると安全性が保てなくなるなどの問題がある．一方で，Beaver triple の生成に準同型暗号などを用いないため非常に事前計算が高速化されるメリットがある．本稿ではこの Client-aided モデルの採用を前提に議論を進める．

3.2 FM-Index による検索と参照テーブルの構築

全文検索の高速化には文書の索引化が有効である．索引法の一つとして知られる FM-Index [8] では，文書を一文字刻みで回転させた文字列の集合を辞書順にソートし，各文字列の先頭の一文字を並べた文字列 F と最後尾の一文字

を並べた文字列 L を得る．L と F の長さはそれぞれ元の文書のそれと等しい．誌面の都合上，詳細は割愛するが次式により L の i 番目の要素 $L[i]$ と対応する F の要素の位置 j を計算することができる．

$$j = \mathsf{CF}_{L[i]}(L) + \mathsf{Rank}_{L[i]}(L,\, i) \tag{1}$$

ただし $\mathsf{Rank}_c(T,\, i)$ は文字列 T において位置 i までに現れる文字 c の個数，$\mathsf{CF}(T)$ は文字列 T において c よりも辞書順で小さい文字の総数を出力する関数とする．また，位置 i と元の文書における $F[i]$ の出現位置の対応表は別途用意されており，i が分かれば $F[i]$ を先頭とする接尾辞を特定できるとする．FM-Index では式 1 により，文書から任意の文字列のマッチを計算することができる．マッチはクエリの最後の文字から順に絞り込むように計算され，計算結果は F 上の左開，右閉区間であらわされる．図 1 の例に従い，クエリ ra と文書 $abraca$ のマッチを計算する手順を示す．式 1 により，L で最初に現れる文字 a と最後に現れる文字 a の F 上の区間 $(f, g]$ を計算することができる．$L[i]$ と $F[i]$ は元の文書中で隣り合う文字のため，$(f, g]$ は L では$*a$（ただし，*は任意の文字）のマッチに対応する区間を探り当てたことになる．この区間の中で最初に現れる文字 r と最後に現れる文字 r から F 上の区間 $(\hat{f}, \hat{g}]$ を計算すれば，ra の出現位置を絞りこめたことになる．上記検索における区間の更新は以下のように一般化することができる．次式では k 文字のマッチに対応する区間から $k+1$ 文字のマッチに対応する区間を計算する．

$$\begin{aligned} f_{k+1} &= \mathsf{CF}_{S[k]}(L) + \mathsf{Rank}_{S[k]}(L, f_k) \\ g_{k+1} &= \mathsf{CF}_{S[k]}(L) + \mathsf{Rank}_{S[k]}(L, g_k) \end{aligned} \tag{2}$$

$S[k]$ はクエリ文字列の後ろから k 番目の文字とする．区間の大きさ $g_k - f_k$ はクエリの最後の文字から数えて k 文字目までの部分文字列の出現回数を示す．そのため，$g_k - f_k = 0$ であるならば，クエリと文書の最長一致の長さは $k-1$ であることがわかる．

さて，ここで注意してほしいのは，式 2 は全ての文字の位置について事前計算可能なことである．つまり，式 2 の計算結果を参照テーブルに保存しておけば，検索の計算量は文書の長さに依存せずクエリの長さのみに依存する．そこで，次の参照テーブル（ベクトル）V_c を考えると

$$V_c[i] = \begin{cases} \mathsf{CF}_c(T) & (i = 0) \\ \mathsf{CF}_c(T) + \mathsf{Rank}_c(L, i) & (1 \le i \le N) \end{cases} \tag{3}$$

次式で示す V_c の参照により区間の更新を実現できる． ただし N は文書の長さとする．

$$f_{k+1} = V_{S[k]}[f_k], \qquad g_{k+1} = V_{S[k]}[g_k] \tag{4}$$

つまり，初期区間を $(f_0 = 0, g_0 = N]$ として，クエリ中の文字に対応するテーブルを参照していき，更新の過程で $f = g$ を検知したら，最長一致接部分文字列を知ることができる．なお，FM-Index と類似の索引法で木構造やグラフを扱う方法などが知られており [28]，これらもテーブル化できる．よって，4.1 章で解説する秘匿検索法は，様々な構造の検索に応用可能である．

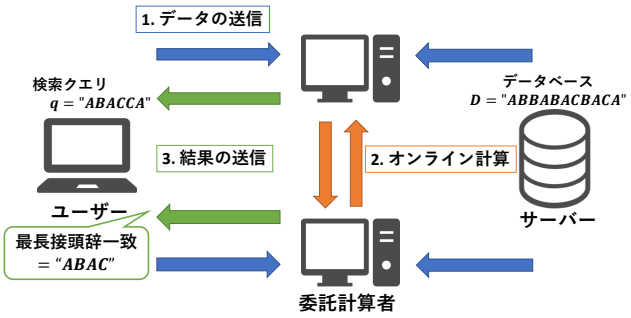

図 2　本研究の設定

4. 提案手法

問題設定: 本研究では，以下のシナリオ想定した二者間秘密分散プロトコルを開発する．図 2 により要件を模式的に示す．

要件 1　プロトコルには，検索クエリ保有者 $\mathcal{A}$ と文書保有者 $\mathcal{B}$ に加え，結託の恐れのない二人の委託計算者 P_0 と P_1 が参加する．$\mathcal{A}$ はクエリ文字列 S を持ち，$\mathcal{B}$ は文書 T を持つ．$\mathcal{A}$ は S を誰にも開示することなく，S と T の最長一致接部分文字列とその出現位置をのみを知ることができる．ただし，最長一致接部分文字列は S の接頭辞であるとする．P_0，P_1 は T に関するいかなる情報も得ない．

提案手法の概要: 3.2 章で示したように，文書 T から構築した参照テーブル V によりクエリのマッチを計算できる．そこで提案手法では，まずはじめに $\mathcal{B}$ が文書 T から V を構築して委託計算者である P_0 と P_1 に送り，P_0 と P_1 が V により文書とクエリのマッチを計算して，計算結果を $\mathcal{A}$ に返却する．この際，委託計算者は中間結果を知ることなく再帰的に V を参照する必要があり，それが本提案手法の中核となる．そのため，まずは 3.2 章にて当該問題設定とそれを実現するプロトコルの詳細を説明する．

4.1 秘密分散型再帰的紛失通信

本研究では，次に述べる問題を秘密分散型再帰的紛失通信と呼称する．

要件 2　データ保有者 $\mathcal{B}$ は長さ N のベクトル V を持つ．結託の恐れのない二人の委託計算者 P_0 と P_1 は，V を知ることなく次式で与えられる x のシェアを計算する．

$$x = V[\ldots V[V[p_0]]\ldots]$$

ただし，再帰の深さ L と検索の起点 p_0 は予め与えられていることとする．

4.1.1 データ保有者 $\mathcal{B}$ の事前準備

まずはじめに，$\mathcal{B}$ は V を秘匿するために，V を乱数でマスクしたベクトルを L 個生成する．以下のように $1 \leq j \leq L$ に関して乱数 r^j でマスクされたベクトルを R^j とする．

$$R^j[i] = \begin{cases} (V[i] + r^j) \bmod N & (j = 1) \\ (V[(i - r^{j-1}) \bmod N] + r^j) \bmod N & (2 \leq j \leq L-1) \\ (V[(i - r^{j-1}) \bmod N]) \bmod N & (j = L) \end{cases}$$

次に，$\mathcal{B}$ は R^j のシェア $([\![R^j[0]]\!]_0, \ldots, [\![R^j[N-1]]\!]_0)$ と $([\![R^j[0]]\!]_1, \ldots, [\![R^j[N-1]]\!]_1)$ を作成し，それぞれを委託計算者 P_0 と P_1 に送信する．

4.1.2 オンライン計算

委託計算者 P_0 と P_1 は次の手順により x のシェアを得る．

Step1: P_0 はシェア $[\![R^1[p_0]]\!]_0$ を，P_1 はシェア $[\![R^1[p_0]]\!]_1$ を選び，以下の計算により次の位置 p_1 を得る．

$$p_1 = \mathsf{Reconst}([\![R^1[p_0]]\!]_0, [\![R^1[p_0]]\!]_1)$$

Step2: $2 \leq j \leq L-1$ に関して以下の計算を繰り返す．

$$p_j = \mathsf{Reconst}([\![R^j[p_{j-1}]]\!]_0, [\![R^j[p_{j-1}]]\!]_1)$$

Step3: P_0 はシェア $[\![R^L[p_{L-1}]]\!]_0$ を，P_1 はシェア $[\![R^L[p_{L-1}]]\!]_1$ を $\mathcal{B}$ に送る．

Step4: $\mathcal{B}$ は受け取ったシェアから以下により x を得る．

$$x = \mathsf{Reconst}([\![R^L[p_{L-1}]]\!]_0, [\![R^L[p_{L-1}]]\!]_1)$$

4.1.3 プロトコルの健全性及び安全性

健全性: 計算の正しさを検証する．まず Step1 により

$$p_1 = R^1[p_0] = (V[p_0] + r^1) \bmod N$$

が計算される．次に，Step2 における $j = 2$ の処理では

$$\begin{aligned} p_2 &= R^2[p_1] = (V[(p_1 - r^1) \bmod N] + r^2) \bmod N \\ &= (V[V[p_0]] + r^2) \bmod N \end{aligned}$$

上式で示されるように R^2 の生成に使われた乱数は r^1 と r^2 であるが，このうちベクトル V の回転に用いた r^1 は R^1 の生成時に作用させた $(V[p_0] + r^1) \bmod N$ により打ち消される．R^j と R^{j+1} にも同様の関係があることから，Step2 の処理により x のシェアを正しく計算することができる．

安全性: Step1 で復元される値は，$(V[p_0] + r^1) \bmod N$ であるが，これは委託計算者の知りえない乱数 r^1 によりマスクされているため V に関する情報は漏洩しない．同様に，Step2 で復元される値は

$$\begin{aligned} p_j &= R^j[p_{j-1}] = (V[(p_{j-1} - r^{j-1}) \bmod N] + r^j) \bmod N \\ &= (V[\ldots V[p_0] \ldots] + r^j) \bmod N \end{aligned}$$

となり，r^j によりマスクされるため，V に関する情報は漏洩しない．

4.1.4 計算量，ラウンド数，通信量

事前計算: 検索対象となるベクトル V の長さが N であり，各要素に関して計算が必要なことから $R^1, \ldots, R^L$ の計算量は $O(LN)$ となる．同様に通信量も $O(LN)$ となる．

オンライン計算: Step1 と Step2 では R の大きさに依存した計算は行われず，合計 L 回の Reconst の計算のみが必要とされる．1 回の Reconst の計算量，ラウンド数，通信量はそれぞれ，$O(1)$, 1, $O(1)$ であるため，オンライン計算全体では $O(L)$, L, $O(L)$ となる．

以上より，秘密分散型再帰的紛失通信の委託計算者における計算は，データベースの長さ N に依存せず，参照の回数 L のみに依存することが示された．

4.2 秘匿全文検索

提案手法では，まず $\mathcal{B}$ が文書 T から参照テーブル $V = (V_1, \ldots, V_{|\Sigma|})$ を計算し，委託計算者に計算を依頼する．V の参照には秘密分散型再帰的紛失通信プロトコルを用いる．この際，$\mathcal{A}$ の持つクエリの文字 c に従って，参照の対象となるベクトル V_c を選択しなければならない．詳細な手続きは以降に記載するが，提案手法では $\mathcal{A}$ がクエリの文字を unary 形式で表現し，（つまり，アルファベットのサイズが $|\Sigma| = 4$ で $c = 2$ の場合は $(0, 1, 0, 0)$ と表現する．）委託計算者にそのシェアを送信する．委託計算者は unary とあらかじめ計算しておいた $(V_1[i], \ldots, V_{|\Sigma|}[i])$ のシェアを用いた内積計算により $V_c[i]$ のシェアを得る．以降ではデータ保有者をサーバー，クエリ保有者をユーザーと呼称する．

4.2.1 サーバー及びユーザーの事前準備

秘密分散型再帰的紛失通信プロトコルの手順と同様に，サーバーは V_c から以下の値を計算する．

$$R^j_{c,f}[i] = \begin{cases} (V_c[i] + r^j_f) \bmod N & (j = 1) \\ (V_c[(i - r^{j-1}_f) \bmod N] + r^j_f) \bmod N & (2 \leq j \leq N) \end{cases}$$

上記は区間の下限 f についての式であり，上限 g についても同様の計算をする．したがってサーバーは $N \times L \times |\Sigma|$ の二倍のサイズのテーブルを作ることになる．

また，ベクトル R を生成する上で必要な乱数 r^j_f 及び r^j_g について，それらの差分を計算したベクトル

$$r'[j] = r^j_f - r^j_g$$

を計算しておく．FM-Index の計算では，区間の大きさが 0 となった時に最長一致であることが確認できる．上記の値は，$g - f = 0$ をシェアのまま計算しするために必要となる．ここまでに生成した値は，すべてシェアにして委託計算者 P_0, P_1 にそれぞれ送信しておく．

ユーザーはクエリ文字列 S の各要素 $S[j] (1 \leq j \leq L)$ に関して次式で示す $|\Sigma| \times L$ のテーブル q を生成する．（$(q[i, 1], \ldots, q[i, 1])$ が $S[|\Sigma|]$ の unary となることに注意．）

$$q[j,c] = \begin{cases} 1 & (c = S[j]) \\ 0 & (c \neq S[j]) \end{cases}$$

q の各要素もすべてシェアにして委託計算者 P_0, P_1 にそれぞれ送信する．

Algorithm 1 オンライン計算

- Public input:データベース長 N, アルファベット $\Sigma = 0, 1, \cdots, |\Sigma| - 1$, クエリ長 L,$f_0 = 0, g_0 = N$
- Private input of user:長さ L の検索文字列 S から計算したクエリ q
- Private input of server:事前に計算したベクトル $R^j_{c,f}, R^j_{c,g} (1 \leq j \leq L), r'$

1: (Server preparation) サーバーは $R^j_{c,f}, R^j_{c,g}, r'$ をシェアにして委託計算者 P_0, P_1 へ送信.
2: (User preparation) ユーザーは q のシェアを P_0, P_1 へ送信.
3: (Computation between P_0 and P_1)
4: **for do**$j = 1, \cdots L$
5: すべての $c \in \Sigma$ についてシェア上での内積計算を行う.

$$[\![f_j]\!]_0, [\![f_j]\!]_1 \leftarrow \sum_c \mathsf{MULT}(R^j_{c,f}[f_{j-1}], q[j,c])$$

$$[\![g_j]\!]_0, [\![g_j]\!]_1 \leftarrow \sum_c \mathsf{MULT}(R^j_{c,g}[g_{j-1}], q[j,c])$$

6: $P_i (i = 0, 1)$ は，$[\![f_j]\!]_i, [\![g_j]\!]_i$ を相手に送信.
7: f と g を更新する.

$$f_j \leftarrow \mathsf{Reconst}([\![f_j]\!]_0, [\![f_j]\!]_1)$$

$$g_j \leftarrow \mathsf{Reconst}([\![g_j]\!]_0, [\![g_j]\!]_1)$$

8: **end for**
9: すべての $j (1 \leq j \leq L)$ についてシェア上での f と g の一致判定を行う.

$$[\![eq[j]]\!]_0, [\![eq[j]]\!]_1 \leftarrow \mathsf{Equality}(f_j - g_j - r'[j], 0)$$

10: P_0, P_1 は $[\![eq]\!]_0, [\![eq]\!]_1$ をユーザーへ送信

4.2.2 オンライン計算

委託計算者が行う計算について説明する．委託計算者は前節で述べた R, r', q のシェアをすべて受理済みであるとする．

区間の更新: R の参照により，クエリと文書のマッチを示す区間を更新する．ここでは，$(f_{k-1}, g_{k-1}]$ が既に計算されている場合に，$(f_k, g_k]$ を計算する手順を説明する．

まず，委託計算者は次式で示すように，R と q からそれぞれ計算に必要な $|\Sigma|$ 個の要素を取り出し，その内積のシェアを計算する．

$$[\![f_k]\!]_0, [\![f_k]\!]_1 \leftarrow \sum_c \mathsf{MULT}(R^k_{c,f}[f_{k-1}], q[k,c])$$

上式は区間の下限 f についての計算だが，上限 g についても同様の計算を行う．この計算により，k 文字目の検索文字 $[S]$ を委託計算者に開示することなく，$R^k_{S[k],f}[f_{k-1}]$ 及び $R^k_{S[k],g}[g_{k-1}]$ のシェアを得ることができる．次に，互いのシェアを送り合い，それぞれの手元で Reconst を計算して $R^k_{S[k],f}[f_{k-1}], R^k_{S[k],g}[g_{k-1}]$ を復元し，次の区間更新で使う f_k，g_k とする．以上の操作を繰り返す．ここで，アルゴリズムのポイントが理解しやすいように，更新式を具体的に見ていく．まず $k = 1$ の時，f と g の初期値は $f_0 = 0, g_0 = N$ であるため，次のように更新される．

$$\begin{aligned} f_1 &= R^1_{S[1],f}[f_0] = (V_{S[1]}[0] + r^1_f) \bmod N \\ g_1 &= R^1_{S[1],g}[g_0] = (V_{S[1]}[N] + r^1_g) \bmod N \end{aligned}$$

同様に $k = 2$ については次のようになる．

$$\begin{aligned} f_2 &= R^2_{S[2],f}[f_1] = (V_{S[2]}[(f_1 - r^1_f) \bmod N] + r^2_f) \bmod N \\ &= (V_{S[2]}[V_{S[1]}[0]] + r^2_f) \bmod N \\ g_2 &= R^2_{S[2],g}[g_1] = (V_{S[2]}[(g_1 - r^1_g) \bmod N] + r^2_g) \bmod N \\ &= (V_{S[2]}[V_{S[1]}[N]] + r^2_g) \bmod N \end{aligned}$$

4.1 章での説明と同様に，テーブル R を生成するときに使った乱数のうち，ベクトル V の回転として作用させていた乱数は一つ前の値を代入することで打ち消され，検索に必要な値はもう一つの乱数で守られる．

f=g の判定: 区間の更新をすべて終えた後，$f_k = g_k$ の成立の判定を $k = 1, \ldots, L$ についてすべて計算する．当然，委託計算者に結果を開示できないため，判定結果は Equality プロトコルを用いてシェアの状態で得る．f と g は，別々の乱数でマスクされているため，乱数の差分を解消するために r' のシェアを作用させる．以下に計算式を示す．

$$[\![eq[k]]\!]_0, [\![eq[k]]\!]_1 \leftarrow \mathsf{Equality}(f_k - g_k - r'[k], 0)$$

最後に，一致判定の結果 $[\![eq[k]]\!]_0, [\![eq[k]]\!]_1$ を全てユーザへ送信する．ユーザーは委託計算者から受け取ったシェアを Reconst すれば最長一致文字列を知ることができる．オンライン計算のプロトコルの詳細は Algorithm1 に示す．

4.2.3 計算量，ラウンド数，通信量

事前計算: 参照テーブル V から構築した R の大きさは $2 \times N \times L \times |\Sigma|$ であり，R の各要素について計算が必要なことから，計算量は $O(LN|\Sigma|)$ となる．また，同様に通信量も $O(LN|\Sigma|)$ となる．なお詳細は割愛するが，参照テーブルを構築する際に，式 2 の結果を単純に保存するのではなく，Wavelet Matrix を用いた索引法を利用することにより，容易に参照テーブルのサイズを $O(LN \log(|\Sigma|))$ に削減することができる．よって，計算量，通信量ともに $O(LN \log(|\Sigma|))$ に抑えることができる．

オンライン計算: オンライン計算では，R の大きさに依存した計算は行われず，合計 $2L$ 回の内積と Reconst の計算のみが必要とされる．そのため，オンライン計算全体の計算量，ラウンド数，通信量は $O(L|\Sigma|)$, $2L$, $O(L|\Sigma|)$ となる．なお，上述の Wavelet Matrix を利用した場合の計算量，ラウンド数，通信量は，$O(L \log(|\Sigma|))$, $2L \log(|\Sigma|)$, $O(L \log(|\Sigma|))$ となる．

表 1　シミュレータによる測定結果

N	L	事前計算	オンライン計算	ラウンド数	通信量
10^3	10	0.09683s	0.01333s	22	0.3564KB
	50	0.4662s	0.05971s	102	1.782KB
	100	0.9325s	0.1203s	202	3.564KB
10^4	10	0.9275s	0.01251s	22	0.3564KB
	50	4.775s	0.06035s	102	1.782KB
	100	9.267s	0.1424s	202	3.564KB
10^5	10	9.772s	0.03016s	22	0.7128KB
	50	49.05s	0.1507s	102	3.564KB
	100	96.12s	0.2842s	202	7.129KB
10^6	10	98.98s	0.03300s	22	0.7128KB
	50	494.0s	0.1409s	102	3.564KB
	100	984.8s	0.3202s	202	7.129KB

5. 提案手法の性能評価

実装: 提案した文字列検索プロトコルのシミュレータを Python3.6(+Numpy1.16.4) を用いて実装した．今回の実装では，通信部分の実装は行っておらず，一つのプログラムの上でサーバーとユーザーの事前準備及び委託計算者によるオンライン計算を記述している．そのため，通信が行われた場合の通信量は，一つのシェアのサイズと交換されるシェアの個数から算出した．また，ラウンド数もプロトコル通りを記しているが，これはプログラムの for 文の回数と一致するため，仮に通信部分を実装したとしても同じ値となる．

実験環境, 実験データ: 実験には, Xeon 3.4GHz の CPU が搭載された計算機を用いた．シミュレータを高速にするための特別な実装は行っておらず, 1CPU コアしか利用していない．今回の実験ではゲノム配列の検索を想定し，アルファベットのサイズ $|\Sigma| = 4$，長さ $N = 1000, 10000, 100000, 1000000$ の文書 T，長さ $L = 10, 50, 100$ のクエリ S のデータセットを生成し，様々なパラメータでの性能を計測することとした．また，これら人工データに加え，実際のヒトゲノム配列を用いた実験データも用意した．一番染色体の先頭から長さ $N = 10000000$ の配列を抜き出して文書 T とし，クエリ S は一番染色体のランダムな位置から抜き出した長さ $L = 100$ の配列とした．シェアの平文空間のサイズは $N = 1000, 10000$ については 16 ビット，$N = 100000, 1000000, 10000000$ については 32 ビットでシミュレーターを実行した．

人工データによる実験結果: 結果を**表 1** に示す．秘密分散プロトコルにおいて事前計算とは，MULT に必要な Beaver triple を生成する時間のみを表し，その他のシェアを生成時間は含めないのが慣例となっている．しかしながら，提案手法は事前計算の比重を増し，オンライン計算のコストを大幅に削減する特徴をもつため，Beaver triple の生成に加え，サーバーがベクトル V からテーブル R と r' を生成し，それらをシェアに変換する計算時間も含めている．

計算量の理論値からも明らかだが，1CPU での実験にも拘わらず，オンライン計算は非常に高速に動作する．また，ラウンド数も少ないため，インターネット上の通信遅延を考慮したとしても，十分に実用的な計算時間を実現しているといえる．一方，事前計算には多くの時間がかかっていることがわかる．実装を精査したところ，シェアの生成に必要な乱数の生成に時間がかかっていることが分かった．このため，乱数生成を高速な実装に切り替えることにより，事前計算量を抑えられる可能性が高い．また，事前計算は並列性能の高い処理であるため，使用する CPU を増やせば高速化が可能である．

実データによる実験結果: 実データの実験では，オンライン計算に 0.3s，事前計算に 8353s かかり，ラウンド数は 202，総通信量は 7.129KB であった．これは 1CPU での実行であるため，例えば 10CPU で並列計算を行った場合には事前計算は 14 分程度で実行可能である．

従来研究との比較: 準同型暗号による手法との比較を行うため，sWM[24] を用いた実験も行った．$N = 10000, 100000$，$L = 100$ とし，通信は localhost で行い，10CPU での並列計算を行った．$N = 10000$ でサーバーの計算時間が 25.48s，$N = 100000$ で 199.7s という結果になった．したがって事前計算の時間を考慮しても提案手法の方が高速であることがわかる．オンライン計算との単純比較では，1CPU 換算で 7000 倍以上の高速化を達成したことになる．また，通信遅延を 10ms と仮定した場合でも 800 倍以上の高速化が予想される．また，sWM ではユーザーも途中計算に参加する必要があり，クエリの文字ごとに $\sqrt{N}$ に比例する数の暗号文を送信する必要があるため，オンライン計算時における通信量についても提案手法が効率的であると言える．ただし，提案手法ではサーバーから委託計算者に事前に送られるテーブル R のシェアのサイズが非常に大きいため，事前計算の結果の送信に工夫が必要となる．

6. まとめと今後の課題

本研究では，秘密分散における二者間秘匿計算を用いてサーバーとユーザー間で効率的に文字列検索を行う手法を提案した．提案手法はのオンライン計算量，通信量，ラウンド数はデータベース文書の大きさに依存せず，クエリの長さに依存する．提案手法を実装し，実データを含むゲノム配列を用いて実験を行ったところ，提案手法のオンライン計算は巨大なデータベースにも対応可能な実用性を持つことが確認された．オンライン計算が高速であるのに対して，事前計算はデータベース長に依存する計算量が必要なため，今後はシェアの生成の高速化や，索引方法の改良などにより参照テーブルサイズの縮小が課題となる．

参考文献

[1] Toshinori Araki, Assi Barak, Jun Furukawa, Tamar Lichter, Yehuda Lindell, Ariel Nof, Kazuma Ohara, Adi Watzman, and Or Weinstein. Optimized honest-majority MPC for malicious adversaries - breaking the 1 billion-gate per second barrier. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pp. 843–862, 2017.

[2] Gilad Asharov, Shai Halevi, Yehuda Lindell, and Tal Rabin. Privacy-preserving search of similar patients in genomic data. *PoPETs*, Vol. 2018, No. 4, pp. 104–124, 2018.

[3] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pp. 420–432, 1991.

[4] Dan Bogdanov, Margus Niitsoo, Tomas Toft, and Jan Willemson. High-performance secure multi-party computation for data mining applications. *Int. J. Inf. Sec.*, Vol. 11, No. 6, pp. 403–418, 2012.

[5] Koji Chida, Daniel Genkin, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Yehuda Lindell, and Ariel Nof. Fast large-scale honest-majority MPC for malicious adversaries. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, pp. 34–64, 2018.

[6] Victor Costan and Srinivas Devadas. Intel SGX explained. *IACR Cryptology ePrint Archive*, Vol. 2016/086, , 2016.

[7] Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, 2015.

[8] Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pp. 390–398, 2000.

[9] Matthew K. Franklin, Mark A. Gondree, and Payman Mohassel. Communication-efficient private protocols for longest common subsequence. In *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009. Proceedings*, pp. 265–278, 2009.

[10] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pp. 169–178, 2009.

[11] Oded Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004.

[12] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pp. 218–229, 1987.

[13] Mark A. Gondree and Payman Mohassel. Longest common subsequence as private search. In *Proceedings of the 2009 ACM Workshop on Privacy in the Electronic Society, WPES 2009, Chicago, Illinois, USA, November 9, 2009*, pp. 81–90, 2009.

[14] Liina Kamm, Dan Bogdanov, Sven Laur, and Jaak Vilo. A new way to protect privacy in large-scale genome-wide association studies. *Bioinformatics*, Vol. 29, No. 7, pp. 886–893, 2013.

[15] Payman Mohassel, Ostap Orobets, and Ben Riva. Efficient server-aided 2pc for mobile phones. *PoPETs*, Vol. 2016, No. 2, pp. 82–99, 2016.

[16] Payman Mohassel and Peter Rindal. Aby^3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pp. 35–52, 2018.

[17] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pp. 19–38, 2017.

[18] Hiraku Morita, Nuttapong Attrapadung, Tadanori Teruya, Satsuya Ohata, Koji Nuida, and Goichiro Hanaoka. Constant-round client-aided secure comparison protocol. In *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part II*, pp. 395–415, 2018.

[19] Satsuya Ohata and Koji Nuida. Towards high-throughput secure MPC over the internet: Communication-efficient two-party protocols and its application. *CoRR*, Vol. abs/1907.03415, , 2019.

[20] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pp. 169–179, 1978.

[21] Thomas Schneider and Oleksandr Tkachenko. EPISODE: efficient privacy-preserving similar sequence queries on outsourced genomic databases. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019*, pp. 315–327, 2019.

[22] Adi Shamir. How to share a secret. *Commun. ACM*, Vol. 22, No. 11, pp. 612–613, 1979.

[23] Kana Shimizu, Koji Nuida, and Gunnar Rätsch. Efficient privacy-preserving string search and an application in genomics. *Bioinformatics*, Vol. 32, No. 11, pp. 1652–1661, 2016.

[24] Hiroki Sudo, Masanobu Jimbo, Koji Nuida, and Kana Shimizu. Secure wavelet matrix: Alphabet-friendly privacy-preserving string search for bioinformatics. *IEEE/ACM transactions on computational biology and bioinformatics*, 2018.

[25] Yuri Yamamoto and Masato Oguchi. A decentralized system of genome secret search implemented with fully homomorphic encryption. In *2017 IEEE International Conference on Smart Computing, SMARTCOMP 2017, Hong Kong, China, May 29-31, 2017*, pp. 1–6, 2017.

[26] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pp. 162–167, 1986.

[27] 清水佳奈. 生命情報科学におけるプライバシ保護検索. シミュレーション, Vol. 35(1), , 2016.

[28] 岡之原大輔. 高速文字列解析の世界. 岩波書店, 2012.