完全準同型暗号を用いたゲノムデータベースにおける秘匿検索システムのデザインの比較と分析

山田 優輝[1,a)]　小口 正人[1,b)]

概要：近年ヒトゲノムの解析と応用が可能になり，ゲノムデータ利用の実用化が注目されるようになった．特にバイオインフォマティクスの研究において頻繁に行われるマッチ判定演算について，各医療機関が保持するゲノムデータに研究者などがアクセス出来るアプリケーションを実現するためには，ゲノムデータのデータベースをクラウドに保持し，利用者の問い合わせに応じてクラウドで演算を行う委託システムを用いることが望ましいと考えられる．この委託システムにおいてクラウドと利用者との間で相互にデータを秘匿するために，従来の共通鍵暗号方式ではなく暗号化されたデータ同士での演算が可能な完全準同型暗号を用いる秘匿検索手法やその高速化が研究されている．完全準同型暗号及び暗号化データベースを用いたシステムではその計算量の削減と復号の保証とのためにクライアント・サーバ間での通信を繰り返す設計が多く提案されているが，一般にクライアントは計算資源に乏しく，また通信コストや性能面での制約もあるため，このような設計は適さない場面が多い．本研究ではサーバ上で一度に全ての秘匿検索演算を行う複数のゲノム秘匿検索プロトコルを実装し，実行時間の比較・分析を行う．

キーワード：完全準同型暗号，ゲノム，プライバシ，クラウド

1. 研究背景

近年ゲノムデータ利用の実用化が注目されるようになったが，これを統計処理する際には大型のストレージと計算機が必要になるため，データをクラウドに預け問い合わせに応じてクラウドで演算を行うゲノムデータ委託システムが今後普及していくと考えられる．特にバイオインフォマティクスの研究ではゲノムデータの特定の位置に特定の文字列が存在するか否かの判定演算が良く行われるため，これを安全に行うシステムが求められる．この際，各機関にとって所持しているゲノムデータを完全に公開することは難しく，クライアントとなる研究者も自身の未発表の研究内容を公にしたくないため，相互にプライバシを保護するための暗号化処理が必須となるが，共通鍵暗号方式を用いると演算を委託するためにクラウドに秘密鍵を渡さなければならず，データの漏洩時にはこの鍵を用いて復号されてしまう．また，加法準同型暗号による暗号化も挙げられるが，複雑な演算が困難なために，演算結果からサーバ側のデータ漏洩を防ぐことは難しいと考えられている [1].

これに対して，従来の共通鍵暗号方式ではなく暗号化されたデータ同士での演算が可能な完全準同型暗号（FHE: Fully Homomorphic Encryption）を用いる秘匿検索手法やその高速化が研究されている．FHE を用いてクラウドに計算を委託するシステムでは，サーバでの計算量の削減と 3 章で述べる暗号文内のノイズの増加への対応のためにクライアント・サーバ間での通信を繰り返す設計が提案されることが多いが，一般にクライアントは計算資源に乏しく通信コストや性能面での制約もあるため，このような設計は適さない場面が多い．先行研究 [2] では，最小限の通信回数でゲノム秘匿検索システムを実現できるプロトコルを提案している．この先行研究ではサーバ上で一度に全ての秘匿検索演算を行うために bootstrap と呼ばれる演算を導入しているが，この演算は計算量が大きく，計算手順の最適化などによる高速化が進められてはいるものの依然としてサーバ側での負荷が大きくなりやすい．また，bootstrap を用いる代わりにパラメータを大きくすることでこのシステムを実装することも可能だが，この場合 FHE の全ての演算の計算量が大きくなる．そこで本研究では，先行研究で提案されたプロトコルと bootstrap を用いないプロトコルとを実装し，クラウド上で行われる完全準同型暗号を用いた検索演算の実行時間の比較とその分析を行う．

[1] お茶の水女子大学
Ochanomizu University, Bunkyo, Tokyo 112–8610, Japan
[a)] yuki@ogl.is.ocha.ac.jp
[b)] oguchi@is.ocha.ac.jp

2. アプリケーション

ゲノム秘匿検索アプリケーションの目的は，クライアントがゲノムデータベースに対して問い合わせを行った際にクエリとデータベース間でのマッチの有無についての検索結果を得られることである [1]．ゲノムデータは四種類のヌクレオチド – A, C, G, T – から構成される配列であるため，ゲノム秘匿検索は 4 種に限定された文字列検索とみなすことが出来る．図 1 は本研究で想定するクライアント・サーバ型のゲノム秘匿検索システムで行われる操作について示している．

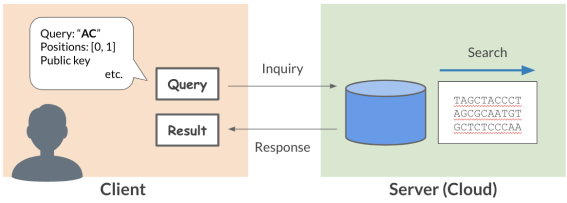

図 1: アプリケーション概要

(1) クライアントは検索したい文字列とその文字列の検索開始位置（ポジション）とを公開鍵などのパラメータと共にクエリとしてサーバに送る．
(2) サーバはクエリとして受信した文字列と保持するゲノムデータベースとの間で，指定された検索開始位置からマッチ検索を行い，クライアントに結果を送る．
(3) クライアントはデータベースとクエリ文字列との間でマッチが存在したか否かの結果を得る．

サーバは保持するデータベース内でゲノム配列データをサンプルごとに並べることによって，各サンプルを特定のポジションから検索することを可能としている．また，クライアントはダミーを含む複数のポジションを指定することで，自身が実際に用いるポジションを秘匿することが出来る．

3. 完全準同型暗号

以下の式 (1) 及び式 (2) のように暗号文同士での加算，乗算が成立する性質をそれぞれ加法準同型性，乗法準同型性と言う．完全準同型暗号 FHE はこの両方の性質を持ち合わせた暗号化手法である．

加法準同型性，乗法準同型性

$$Encrypt(m) \oplus Encrypt(n) = Encrypt(m+n) \quad (1)$$

$$Encrypt(m) \otimes Encrypt(n) = Encrypt(m \times n) \quad (2)$$

FHE を用いることで，ユーザは平文上で行うのと同様に暗号文同士での加法演算・乗法演算を行うことが出来る．

3.1 特徴

FHE の概念は 1978 年に Rivest らによって提唱された [3] が，実装はその 31 年後である 2009 年になってから Gentry によって提案された [4]．これは多項式環やイデアル格子を応用した暗号スキームであり暗号文は読解困難性を保つために平文を暗号化したものにノイズを加えた形式で表現される．この手法を用いた場合，1bit の平文を暗号化するとその暗号文は 1GB 程にもなってしまうなど，当時は計算量の大きさから実用性がないとされていたが，近年ではより効率の良いスキームやその実装について多くの研究がなされており，実用化への期待が高まっている．

問題点としては計算量が大きいことの他に，暗号文に含まれるノイズが演算の度に増加し，*level* と呼ばれる閾値を越えると復号することが出来なくなることが挙げられる．特に乗算を行った際のノイズの増加が著しいため，暗号文に対する乗算操作の回数を限定した *Somewhat Homomorphic Encryption (SHE，SwHE)* の範囲内で，一度の乗算と複数回の加算によって計算することが出来る演算を行うことも研究されている *[5]*．また，*bootstrap* と呼ばれるノイズをリセットする手法の導入を行うことで演算回数の限定は解決することが出来るが，*bootstrap* 自体の計算量が非常に大きいなど難点は残る．

4. 先行研究

PBWT-sec[1] は加法準同型暗号と *Positional-Burrows Wheeler Transform (PBWT)[6]* と呼ばれる離散データ構造とを組み合わせたクライアント・サーバ型のマッチカウントプロトコルであり，クエリの文字列長だけクライアント・サーバ間での通信が必要となるゲノム秘匿検索システムを提案しているが，これに *FHE* を導入したゲノム秘匿検索システムについて複数の先行研究が存在する．*FHE* を用いた場合，加法準同型暗号よりも多くの計算コストを要するが，安全性の向上以外にも，ワイルドカード検索など様々なアプローチが可能となるなどの利点を挙げることが出来る．

石巻らによる先行研究 *[2]* では，*PBWT-sec* に *FHE* を導入し，さらに *bootstrap* を用いるゲノム秘匿検索システムが提案されている．*Bootstrap* の導入によりクライアント・サーバ間での通信回数をクエリの文字列長に関わらず一定とした他，計算アルゴリズムの最適化により実行時間の削減にも取り組んでいる．本研究ではこの先行研究をもとに，ゲノム秘匿検索演算プロトコルについて議論・分析を行う．システムデザインの詳細は 5 章で議論する．

また，本研究では議論しないが，*FHE* を用いたゲノム秘匿検索システムを分散処理の導入によって高速化する研究もなされている *[7]*．*PBWT-sec* のプロトコルでは直前の計算結果を用いた演算が繰り返し行われるため，この先行研究ではデータベースの分割による分散が行われている．

5. システムデザイン

本章では石巻らによる先行研究 *[2]* で提案されたシステムデザインについて議論する．先行研究 *[2]* で提案されたシステムの概要を図 *2* に示す．

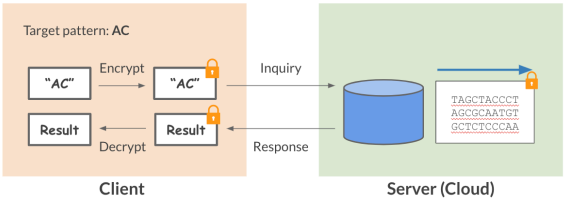

図 2: アプリケーションデザイン

(1) クライアントはクエリを全文暗号化し，その他のパラメータと一緒にサーバに送信する．
(2) サーバはクエリを元に *FHE* 演算を行い，必要に応じて *bootstrap* の演算も実行する．
(3) サーバは最終的な結果をクライアントに送信する．
(4) クライアントはサーバから送られてきたデータを復号し，結果を得る．

PBWT-sec[1] では，サーバは一度に一文字の検索しか行わないため，クライアントはクエリの長さだけ問い合わせを繰り返すことで最終的な結果を得る．これにより一度の問い合わせあたりのサーバ上での計算量を削減することが出来るが，その一方で，クエリ長が増大するとともに，クライアントでの計算負荷も増加してしまうという問題もある．また，毎回の通信では暗号データを始めとする大容量のデータが転送されるため，このデザインは計算資源の乏しいクライアントには適さないと考えることが出来る．

これに対して図 *2* に示されるデザインでは，クエリの文字列長に関わらず一往復の通信で検索を行うことが出来る．このデザインは計算資源の乏しいクライアントにより適しているものの，*3.1* 節で述べたとおり計算のたびに暗号文に含まれるノイズが増えてしまうため，復号を保証するための何らかのアプローチが必要となり，先行研究 *[2]* では *bootstrap* の導入を行うことでこれを解決している．この場合は暗号文のノイズをリセットすることが出来る一方，*bootstrap* 自体が計算量の大きな演算であるため，大きなオーバーヘッドが発生する．

同様のデザインにおいて *bootstrap* を導入しない場合，計算内容に対して十分に大きな値を，復号を保証出来るノイズの量の閾値を表すパラメータ *level* として指定することが考えられる．この場合 *bootstrap* によるオーバーヘッドは発生しないものの，毎回の *FHE* 演算の計算コストが増大してしまうという問題点がある．

6. 実験

先行研究 *[2]* で提案された *bootstrap* を用いるシステムデザインをデザイン *1*，同様のシステムにおいて *bootstrap* を用いずにパラメータ *level* に大きな値を使用する場合をデザイン *2* とし，これらのデザインについてシステムを実装して実験及び分析を行う．

6.1 問題設定

ゲノムデータのサンプルとして，*1000* 人ゲノムプロジェクト *[8]* により提供されるデータから，個体差の現れやすい特定位置の塩基を取り出した一塩基多型 *(SNP: Single-Nucleotide Polymorphism[9])* を並べた *SNP* 配列を *512* サンプル用いる．それぞれのサンプルの長さはは *10,000* 文字とする．検索開始位置であるポジションについて，今回の実験ではダミーを含めずに実際に用いるポジション一つのみを指定する．クエリ長は *1* から *6* まで変化させて実験を行う．

6.2 実験環境

各デザインを *HElib[10]* を用いて *C++*で実装する．いずれにおいても *Smart et al.* による パッキング技術 *[11]* を利用する．

実験を行うマシンのスペックを表 *1* に，また実験に用いたパラメータ *level* の値を表 *2* にそれぞれ示す．このパラメータは各クエリ長での問い合わせ時における *FHE* の演算の内容に対して十分な大きさの値を設定した．デザイン *2-1* については先行研究と同じパラメータを用いた．

表 1: 実験環境

Server	OS	CentOS 6.9
	CPU	Intel®Xeon®Processor E5-2643 v3 (3.4GHz) 6 Cores × 2 Sockets
	Main Memory	512GB
	SSD	80GB
	HDD	2TB

表 2: 実験に用いたパラメータ *level* の値

デザイン	*Level*
デザイン 1	*Level L = 23*
デザイン 2	*Level L = 9 * クエリ長*

6.3 実験結果

各実験を三回ずつ行い，平均の実行時間を算出した．

図 *3* はデザイン *1* 及びデザイン *2* の，各サーバ上での検索演算の平均実行時間をグラフにしたものである．

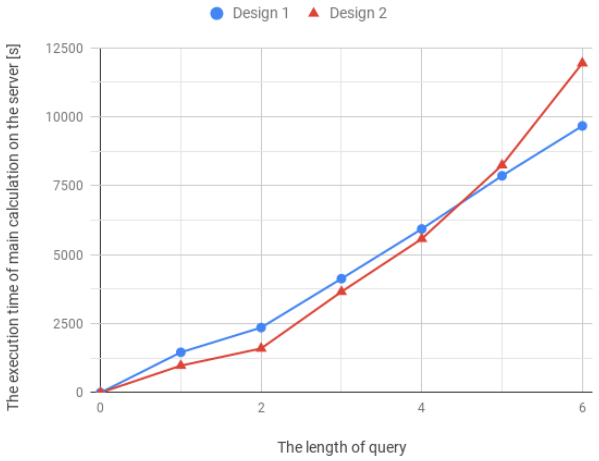

図 3: サーバ上での秘匿検索演算の平均実行時間

6.4 分析

デザイン *1* とデザイン *2* を比較すると，クエリ長が *4* までのときはデザイン *2* が，クエリ長 *5* 以上のときはデザイン *1* のほうが高速となっている．これは表 *2* に示されているパラメータの影響だと考えることが出来る．デザイン *1* では常に同じパラメータを用いるため実験結果はクエリ長に比例した線形グラフになるが，デザイン *2* ではクエリ長によってパラメータが変わるためクエリ長が増えると実行時間も急激に増加する非線形のグラフになる．このため，クエリ長が短い間はデザイン *2* が，長くなるにつれてデザイン *1* がそれぞれより有効なものになるのは妥当であると考えられる．今回の実験ではポジション数は *1* で固定したが，指定するポジション数やその他のパラメータによっても実行時間は左右されるため，クライアントからの問い合わせに応じて *bootstrap* を使用するか否かのデザインをを変更できることが望ましい．

7. まとめと今後の課題

先行研究に基づき，完全準同型暗号を用いたゲノム秘匿検索システムを *bootstrap* を用いる場合と *level* と呼ばれるパラメータを大きくする場合との二種類のデザインで実装し，複数のクエリ長についてサーバ上での秘匿検索演算の実行時間を計測し比較した．その結果，クエリ長やその他のパラメータによって適するデザインが異なることが示された．今後は実装やアルゴリズムの改善による秘匿検索演算の高速化に取り組むとともに，クライアントでの演算時間も含む全体の実行時間の高速化や，クライアント・サーバ間での通信コストについても計測していきたい．

謝辞

本研究は *JST CREST JPMJCR1503* の支援を受けております．

参考文献

[1] K. Shimizu, K. Nuida, and G. Rätsch: “Efficient privacy-preserving string search and an application in genomics.” *Bioinformatics* 32.11 (2016), pp. 1652–1661.

[2] Y. Ishimaki et al.: “Privacy-preserving string search for genome sequences with FHE bootstrapping optimization.” *2016 IEEE International Conference on Big Data (Big Data)*. IEEE. 2016, pp. 3989–3991.

[3] R. L. Rivest, L. Adleman, M. L. Dertouzos, et al.: “On data banks and privacy homomorphisms.” *Foundations of secure computation* 4.11 (1978), pp. 169–180.

[4] C. Gentry et al.: “Fully homomorphic encryption using ideal lattices.” *Stoc*. Vol. 9. 2009. 2009, pp. 169–178.

[5] M. Naehrig, K. Lauter, and V. Vaikuntanathan: “Can homomorphic encryption be practical?” *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. ACM. 2011, pp. 113–124.

[6] R. Durbin: “Efficient haplotype matching and storage using the positional Burrows–Wheeler transform (PBWT).” *Bioinformatics* 30.9 (2014), pp. 1266–1272.

[7] Y. Yamamoto and M. Oguchi: “A decentralized system of genome secret search implemented with fully homomorphic encryption.” *the 1st IEEE International Workshop on Big Data and IoT Security in Smart Computing (BITS2017)*. IEEE. 2017, pp. 1–6.

[8] “The International Genome Sample Resource.” URL: `http://www.internationalgenome.org/`.

[9] “What are single nucleotide polymorphisms (SNPs)? - Genetics Home Reference - NIH.” URL: `https://ghr.nlm.nih.gov/primer/genomicresearch/snp`.

[10] “shaih/HElib: An Implementation of homomorphic encryption.” URL: `https://github.com/shaih/HElib`.

[11] N. P. Smart and F. Vercauteren: “Fully homomorphic SIMD operations.” *Designs, codes and cryptography* 71.1 (2014), pp. 57–81.