完全準同型暗号を用いた暗号化データベースにおける秘匿検索の分散処理による高速化

山田　優輝†　　小口　正人†

† お茶の水女子大学理学部情報科　〒 112–8610　東京都文京区大塚 2-1-1
E-mail: †{yuki,oguchi}@ogl.is.ocha.ac.jp

あらまし　近年ヒトゲノムの解析と応用が可能になり，様々な分野でゲノムデータ利用の実用化が注目されるようになった．ゲノムデータをはじめとするビッグデータを活用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，利用者の問い合わせを受けてクラウドで演算を行う委託システムが今後普及していくと考えられる．この際，クラウドは安全であるとは言えず，またゲノムデータは重要な個人情報であることから，プライバシ保護のための暗号化処理が必須となる．しかし従来の共通鍵暗号方式でこのシステムの実装を試みた場合，クラウド上でゲノムデータを復号することが可能となるため，データの秘匿には適さない．これに対して，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検索演算についてこのシステムを実現するために，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）を用いる秘匿検索手法が提案されているが，実用に際して計算量が課題となっている．そこで本研究では，先行研究のクラウド上で行われる FHE 演算にデータベースの分割による分散処理を適用し，高速化を試みる．

キーワード　並列/分散システム，完全準同型暗号，ゲノム秘匿検索，クラウドコンピューティング

Speeding Up Genome Secure Search on Encrypted Data Base Implemented with Fully Homomorphic Encryption by Distributed Processing

Yuki YAMADA† and Masato OGUCHI†

† Department of Information Sciences, Ochanomizu University
2–1–1 Otsuka, Bunkyo-ku, Tokyo, 112–8610 Japan
E-mail: †{yuki,oguchi}@ogl.is.ocha.ac.jp

1. はじめに

2003 年にヒトゲノム解析計画 [1] と呼ばれる人間のゲノム情報全てを解析するプロジェクトが完了して以降，日本国内でも 2015 年に厚生労働省によるゲノム情報を用いた医療等の実用化推進タスクフォース [2] が編成されるなど，様々な分野でゲノムデータ利用の実用化が注目されるようになった．ゲノムデータをはじめとするビッグデータを統計処理する際には，データを格納する大型のストレージと処理能力の高い計算機が必要になるため，各医療機関や研究機関の内部でデータの処理を行うことは困難である．そこで，処理したい大量のゲノムデータをクラウドに預け，利用者の問い合わせを受けてクラウドで演算を行う，とい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．この際，クラウドはインターネットに接続され不特定多数からのアクセスを受けるため安全であるとは言えず，またゲノムデータは情報が漏洩したとしても付け替えの出来ない重要な個人情報であることから，プライバシ保護のための暗号化処理が必須となる．

特にバイオインフォマティクスの研究においてはゲノムデータに特定の文字列が含まれるか否かの検索が良く行われるため，大量のゲノムデータに対して安全にこの検索を行うことが出来るシステムが求められる．クライアント・サーバ型のシステムで従来の共通鍵暗号方式を用いたゲノム秘匿検索を行うことを考えると，演算を行うためにはクラウドに秘密鍵を渡さなければならず，データが漏洩した際にゲノムデータを復号することが可能となってしまう．暗号文同士での加法演算が成立する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が漏洩することを防ぐことは難しいと考えられている [3].

これに対して先行研究 [4]- [5] では，暗号化方式として暗号化されたデータ同士での加算及び乗算が可能な完全準同型暗号（以下 FHE: Fully Homomorphic Encryption）を用いる秘匿検索手法が提案されている．離散データ構造 Positional-Burrows Wheeler Transform (PBWT) [6] を生かした工夫や計算手順の最適化などによるアルゴリズムの高速化を進めているが，計算量の大きい FHE の演算を行うことから，依然としてサーバ側での負荷が大きくなりやすく課題は解決されていない．そこで本研究では，先行研究 [4] のクラウド（サーバ）上で行われる FHE 演算にデータベースの分割による分散処理を適用し，高速化を試みる．

2. 完全準同型暗号 (FHE)

以下の式 (1) のように暗号文同士での加算が成立する性質を加法準同型性，また式 (2) のように暗号文同士での乗算が成立する性質を乗法準同型性と言うが，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 の概念自体は公開鍵暗号が考案された当初である 1978 年に Rivest らによって提唱されていた [7] 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 [8]．この実装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，読解困難性を保つために，暗号文は平文を暗号化したものにランダムなノイズを加えた形式で表現される．しかし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．近年では Lu らにより，加算や乗算だけではなく比較演算についても高速化及び高精度化が進められる [9] など研究が進められており，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てしまい，閾値を越えると復号することが出来なくなる，というものが挙げられる．特に乗算を行った際のノイズの増加が著しいため， 暗号文に対する加算・乗算操作の演算回数が限定される 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分散や相関，ベクトルの内積等の演算を行うことも研究されている [10]．この問題は bootstrapping 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，計算量が非常に大きくなるなど難点は残る．

3. 先 行 研 究

本章では，先行研究 [4]- [5] について問題設定とそれぞれが実装した FHE によるゲノム秘匿検索手法の概観を述べる．

3.1 問 題 設 定

ゲノム秘匿検索を適用するモデルについて述べる．サーバにデータベースを設置し，ゲノム配列データをサンプルごとに並べる．このゲノムデータはゲノム配列全体を用いるのではなく，個体差が現れやすい特定の位置の塩基を取り出した SNP（一塩基多型）を並べた SNP 配列を用いている．ゲノムデータは A, G, C, T の 4 種の塩基配列から構成されるため，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．サンプルそれぞれの長いゲノム配列データを行ごとに並べて二次元配列状のデータベースとすることで，各サンプルについての文字列検索を行うことが出来る．

ゲノム秘匿検索システムはサーバ・クライアント形式で実装される．サーバはクライアントから一致判定を行いたいクエリ文字列を FHE により暗号化したものとゲノム配列上の検索開始点（以下 ポジション）を受け取ると，データベース上のデータと FHE 演算によるマッチング判定を行い，その結果を返す．この検索を，サーバとクライアントの双方のデータが秘匿された状態で，可能な限り高速に行うことが先行研究の目的である．

3.2 手 法 概 要

石巻ら（2016）は従来の FHE を用いたゲノム秘匿検索手法に bootstrapping を導入し，その演算の最適化を行うことで，計算量の削減を行った [4]．石巻らのシステムはサーバとクライアントが 1:1 で問い合わせを行うもので，サーバはクライアントから検索したい文字列を暗号化処理したものとその文字列を検索したい配列上のポジションを受け取り，秘匿検索を行ってマッチしたか否かの結果を返す．

また，山本ら（2017）は従来のゲノム検索手法に分散処理を導入することによる高速化を試みている [5]．山本らのシステムには bootstrapping が導入されておらず，問い合わせごとの演算回数が限られる．このため，クライアントはクエリを一文字ずつ暗号化して問い合わせを行うが，サーバから受け取った演算結果同士の比較によって，最終的な結果を得る事が出来る．このサーバ側での演算に対して，ゲノム配列データベースの分割によるマスタ・ワーカ型の分散処理を適用することで，高速化が行われている．

また，先行研究 [4]- [5] ではゲノムデータの秘匿検索を高速に行うために，ゲノム配列のデータベースを離散データ構造である Positional-Burrows Wheeler Transform (PBWT) [6] の形に変換している．これはゲノムデータに対して列ごとのソートを行ったもので，クエリとデータベースに含まれるサンプルとのマッチング判定の計算量を大幅に削減することが出来る．また，クライアントがサーバにダミーを含めた複数の検索開始ポジションを伝えることで，実際に利用するポジションを秘匿することが出来る等，秘匿性向上のための工夫も為されている．

4. 提案手法

4.1 提案システム概要

本研究では先行研究に基づき，以下の FHE を用いたゲノム秘匿検索のマスタ・ワーカ型の分散システムを提案する．図 1 に提案システムの概要を示す．

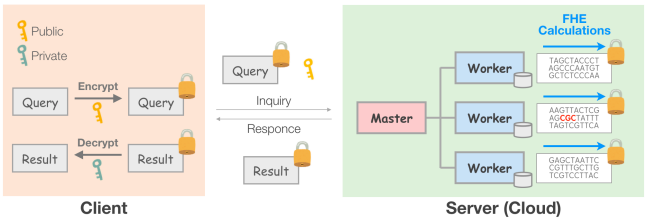

図 1　提案手法概観

(1) クライアントはクエリ全文を暗号化し，
公開鍵などと共にマスタへ送信する．

(2) マスタは受け取ったデータを各ワーカに転送する．

(3) 各ワーカは FHE を用いた秘匿検索演算を行う．
bootstrapping によるノイズのリセットを適宜行う．

(4) 秘匿検索演算終了後，ワーカは検索結果を
マスタへ転送する．

(5) マスタは結果を収集し，クライアントへ送信する．

(6) クライアントは復号を行い，結果を得る．

以上のプロトコルを C++ で実装した．

また，完全準同型暗号計算には GitHub 上で公開されている HElib [11] を，分散時における各マシンの制御のための MPI(Message Passing Interface) を利用するライブラリとしては，Open MPI [12] を用いた．

4.2 分散方法

検索アプリケーションの分散化手法としては，データベースの分割による分散処理，独立性の高い計算部位に適用する分散処理，独立性の高い手順に適用する分散処理，などが挙げられるが，山本らによる先行研究 [5] では分散化した各ワーカマシンにデータベースを設置する，というデータベースの分割による分散処理を適用し，同時により多くのクエリとデータベース間のマッチング有無を調べることが可能となっている．また石巻らによる先行研究 [4] のプロトコルでは直前の演算結果を用いた演算を繰り返し行うため，計算部位や処理手順に分散処理を適用するのは非効率的である．したがって本研究の提案システムにおいても，データベースの分割による分散処理を適用した．

5. 実　　験

5.1 実験環境

ワーカとして同スペックのマシンを 4 台用い，プログラムを実行した．以下の表 1 に各マシンの環境を示す．

Server	OS	CentOS 6.9
	CPU	Intel®Xeon ®Processor E5-2643 v3 (3.4GHz) 6 コア × 2 ソケット
	Main Memory	512GB
	SSD	RAID0，480GB
	HDD	2TB

表 1　実験環境

各マシン上で最大 2 スロットのワーカを稼働させ，そのうち 1 スロットにマスタの機能を持たせた．最大で 7 スロット分のワーカを稼働させてワーカ数ごとの実行時間を比較する実験を行った．

実験に使用するゲノムデータは 1 サンプルあたり 10,000 文字のデータを 200 サンプル用意した．また，検索クエリは長さ 5 文字のものを用いた．さらに検索時の秘匿性を高めるためにダミーの検索開始ポジションを加え，1 クエリにつきデータベースの 5〜50 箇所を始点とした文字列検索を行った．

5.2 実験結果

クエリとデータベースとの間でマッチングの有無を判定するプログラムを分散化した環境上で稼働させた．ポジション数を 50 として実験を 3 回ずつ行った際の，ワーカ数ごとのマスタにおける平均実行時間のグラフを図 2 に示す．

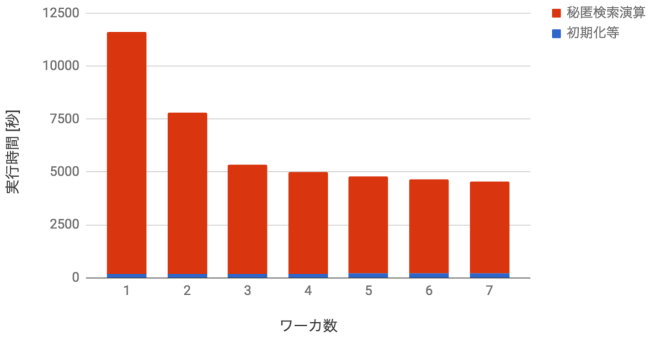

図 2　ワーカ数ごとのマスタにおける平均実行時間 (秒)

マスタでの実行時間において，ワーカ数の増加とともに秘匿検索演算の計算時間が減少することが確認された．初期化の内容としては，クエリ等の送受信やワーカとのデータ共有，データベースの用意等が挙げられるが，これに要する時間はワーカ数に応じて増加することも読み取れる．しかし秘匿検索演算の実行時間と比較するとオーバーヘッドは十分に小さいため，以降は秘匿検索の実行時間のみに注目するものとする．

また，ワーカ数が増えるに連れて，分散処理の導入による計算時間の減少効果は徐々に横ばいになっている．これは計算手順の中にデータベースの大きさには依存しない計算量の処理が含まれているためである．

次に，ポジション数を 5〜30 の間で変化させて，3 回ずつ実験を行った際の，ワーカ数ごとのマスタにおける平均実行時間と高速化率のグラフをそれぞれ図 3，図 4 に示す．この際，分散化の評価となる高速化率は，山本らによる先行研究 [5] と同様に，式 (3) より算出した．

高速化率 = 逐次実行時間 (秒)/並列実行時間 (秒)　　(3)

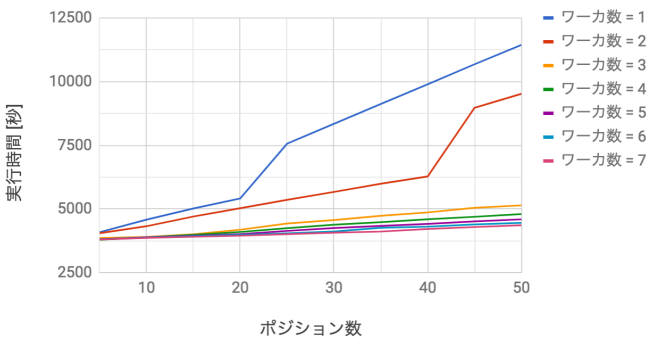

図 3　ポジション数におけるワーカ数ごとのマスタの平均実行時間 (秒)

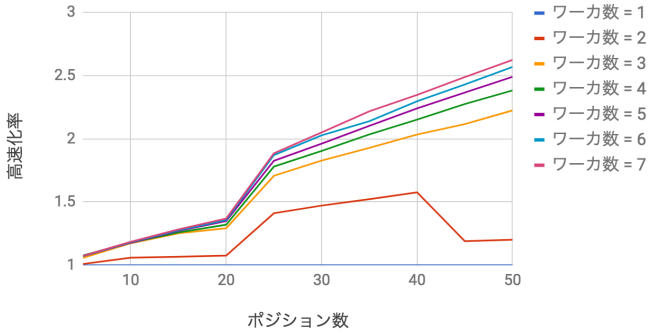

図 4　ポジション数におけるワーカ数ごとの平均実行時間 (秒)

指定するポジション数を増加させるに連れて，高速化率が高くなっていることが分かる．3.2 節で述べたように，先行研究は PBWT 構造のデータベースを用いることによる秘匿検索の高速化を行っている．これによって，クエリの検索はデータベース内のサンプルを全て検索するのではなく，クエリの長さの分のみデータベースと演算することで最長マッチ数を算出する事が出来る．このため，検索ポジション数が少ない実験においては，前処理等の並列分散化できない演算の量を並列分散化可能な演算の量と比較したときの割合がポジション数を増やしたときよりも大きくなり，結果として分散効率が抑えられてしまう．しかしポジション数が多い実験においては，検索するポジションの散在によりデータベース上で参照する箇所が多くなり，データベースによる分散化の効果が現れやすくなったと考えられる．

6. まとめと今後の課題

FHE 及び bootstrapping を用いたゲノム秘匿検索のサーバ側での処理に，データベースの分割によるマスタ・ワーカ型の分散処理を適用し，実験を行った．その結果，分散台数に応じて計算時間が減少すること，また PBWT 構造のデータベースを用いていることにより，指定する検索開始ポジションの数を増やすほど高速化率が高くなることが示された．計算手順の中にデータベースの大きさに計算量が依存しないものも含まれているため，分割数が大きくなると分散処理の導入による計算時間の減少効果は徐々に横ばいになってしまうなどの課題も残っている．

今後は実装やアルゴリズムの改善により更なる高速化に取り組むとともに，実用化を目指して実際によく行われる操作に特化した高速化処理や，秘匿が不要な部分でのプライバシレベルをコントロールすることなども検討していきたい．

謝　　辞

本研究を進めるにあたり，大変有益なアドバイスを頂いた早稲田大学山名研究室及び工学院大学山口研究室の皆様に感謝いたします．

特に早稲田大学山名研究室所属の石巻さん及びお茶の水女子大学小口研究室所属の山本さんからは，ゲノム秘匿検索システムのプログラムと多くの助言を賜りました．深く感謝いたします．

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

文　　献

[1] ヒトゲノム解析センター – ヒトゲノム解析計画とは：http://hgc.jp/japanese/humangenome-j.html, 2017 年 12 月閲覧

[2] 厚生労働省 – ゲノム情報を用いた医療等の実用化推進タスクフォース：http://www.mhlw.go.jp/stf/shingi/other-kousei.html?tid=311652, 2017 年 12 月閲覧

[3] Kana Shimizu, Koji Nuida and Gunnar R ä tsch : ”Efficient privacy-preserving string search and an application in genomics,” http://biorxiv.org/content/early/2015/11/27/018267, 2017 年 12 月閲覧

[4] Y. Ishimaki, H. Imabayashi, K. Shimizu and H. Yamana : ”Privacy-preserving string search for genome sequences with FHE bootstrapping optimization,” 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, 2016, pp. 3989-3991.

[5] Y. Yamamoto and M. Oguchi : ”A Decentralized System of Genome Secret Search Implemented with Fully Homomorphic Encryption,” 2017 IEEE International Conference on Smart Computing (SMARTCOMP), Hong Kong, 2017, pp. 1-6.

[6] R. Durbin : ” Efficient haplotype matching and storage using the Positional Burrows-Wheeler Transform (PBWT),” Bioin- formatics, vol. 30, no. 9, pp. 1266-1272, 2014.

[7] Ronald K. Rivest, Len Adleman, Michael L. Dertouzos : “On Data Banks and Privacy Homomorphisms,” Foundations of Secure Computation, Academia Press, 1978

[8] Craig Gentry : “Fully Homomorphic Encryption Using Ideal Lattices,”Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009

[9] Wen-jie Lu, Shohei Kawasaki, Jun Sakuma : “Using Fully Homomorphic Encryption for Statistical Analysis of Categorical, Ordinal and Numerical Data,” Proceedings of The Network and Distributed System Security Symposium, 2017

[10] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan : ”Can homomorphic encryption be practical?,” Cloud computing security workshop (CCSW), ACM, New York, 2011, pp. 113-124

[11] Shoup V. and Halevi S. : http://shaih.github.io/HElib/index.html, 2017 年 12 月閲覧

[12] Open MPI : https://www.open-mpi.org/, 2017 年 12 月閲覧