完全準同型暗号によるゲノム秘匿検索の分散処理

山本　百合† 　　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: †{yuri,oguchi}@ogl.is.ocha.ac.jp

あらまし　バイオインフォマティクス研究におけるゲノムデータの活用方法として，ゲノムデータ委託計算システムが提案されている．しかし個人ゲノムのプライバシ保護の観点から，暗号化によるデータの秘匿が必要となる．先行研究では，完全準同型暗号をクライアント・サーバ型のゲノム秘匿検索に適用し，将来的に複雑な演算にも対応可能なアルゴリズムの高速化を進めている．しかしながら，完全準同型暗号演算の計算量が大きいために，サーバ側での計算量が大きくなりやすい．本研究では，従来手法のサーバ側での演算に対してマスタ・ワーカ型の分散処理を適用することで高速化を行い，クラウドコンピューティングへの適用が可能な実装を目指す．
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

Distributed System for Genome Secret Search Implemented with Fully Homomorphic Encryption

Yuri YAMAMOTO† and Masato OGUCHI†

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

1. はじめに

バイオインフォマティクスの研究において，研究機関や各病院が所持するゲノムデータの統計的処理による活用が求められている．一般的にヒトゲノムは塩基数にして約 30 億個に及ぶ [1] ため，ゲノムデータを取り扱う統計処理には処理能力の高い計算機での演算が必要となる．そのため各医療機関が所有する個人の大量のゲノムデータを，大型の計算機とストレージを所有する機関に委託し，利用者が問い合わせを行うことで統計処理が可能なゲノムデータ委託システムが今後広がっていくと考えられる．しかしヒトゲノムは個人の識別子となるため，プライバシ保護の観点から暗号を適用した秘匿検索手法によるデータ活用が必要である [2].

クライアント・サーバ型のゲノム秘匿検索を行う際に適用する暗号方式として，従来の共通鍵暗号などによる暗号化も考えられるが，暗号化したゲノムデータ同士の複雑な演算を行うためには，クライアントの秘密鍵をサーバ側に渡す必要が生じてしまう．この場合，サーバ側での暗号化データの復号作業が必要となるため，サーバ側へのクライアントのデータの秘匿が困難となる [3]．また関連研究 [4] として，暗号文同士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ための演算処理が難しいと考えられている [5].

先行研究 [5] では，暗号文同士の加法と乗法が成立する完全準同型暗号をゲノム秘匿検索に適用し，サーバ側が復号することなく統計処理などの複雑な演算が可能な手法を目指し，アルゴリズムの高速化を進めている．またその際に，再帰的紛失通信手法や離散データ構造を生かした工夫を用いることでデータの秘匿性を高めている．しかしながら，完全準同型暗号演算などの演算に関する処理は，計算量が大きいためにサーバ側での計算負荷が大きくなりやすい．本研究では，先行研究が用いている手法のサーバ側での演算に対してマスタ・ワーカ型の分散処理を適用することでゲノム秘匿検索の高速化を行い，ゲノム委託システムのクラウドコンピューティングに向けた分散処理の実装を目指す．

2. 完全準同型暗号

完全準同型暗号とは式 (1)，(2) のように暗号文同士の加算と乗算の演算が成立する性質を持ち，暗号化した状態で平文と同様の多項式演算が可能な暗号である．完全準同型暗号は公開鍵暗号方式の機能を持つが，秘密鍵を用いることなく暗号文同士の演算から平文同士の演算を暗号化した値を導くことが可能

となる．そのため，ゲノム秘匿検索に完全準同型暗号を適用することで，秘密鍵を渡すことなくサーバがデータ同士の統計処理を行えると期待できる [6]．また完全準同型暗号の概念自体は，1970 年代後半に公開鍵暗号が考案された当初より提唱され，2009 年に Gentry [7] が実現する手法を提案した．提案当時は計算量の大きさから実用性が乏しかったが，その後も様々な研究によって高速化や改良が進められ，簡単な計算であれば十分な性能レベルを示している [8]．しかし，暗号の解読困難性を保つためのノイズの付加と解読時のノイズの除去のための工夫の必要性などから，依然として複雑な計算や大きなデータに対する演算では，暗号文サイズが大きくなる傾向にあり，計算量が大きくなる難点を持つ [9]．

完全準同型暗号

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

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

3. 先 行 研 究

本章では，先行研究 [5] が実装した完全準同型暗号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．サーバにゲノム配列データをサンプルごとに並べたデータベースを設置する．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ているため，今回のゲノム秘匿検索は 4 種に限定された文字列検索とみなすことができる．また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．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にすることで，列ごとではゲノム配列の特定の箇所における各サンプルの違いを比較することができる．

ゲノム秘匿検索の問い合わせを行うクライアントは，一致判定を行いたいクエリ配列と検索の開始点（ポジション）をサーバに伝えると，サーバがデータベース上のデータとの検証を行い，クエリとマッチする最長の長さが伝えられる．この検索がサーバとクライアントの双方のデータが秘匿されながら，できるだけ高速に行われることが先行研究の目的である．

3.2 手 法 概 要

石巻ら（2015）は従来のゲノム秘匿検索手法に対して完全準同型暗号を用いることで，複雑な演算処理が可能なゲノム委託計算の整備と，複数の平文を一つの暗号文にまとめて並列計算を行う暗号文パッキングによる計算量の削減を行った [5]．石巻らの手法は，サーバとクライアントが 1:1 で問い合わせを行うゲノム秘匿検索システムにおいて実験が行われている．システムの概要は，サーバはクライアントからゲノム検索文字列を受け取り，自身が所有するゲノムデータとのマッチングを行い，クライアントに結果を返す．このときサーバとクライアント双方のデータを互いに秘匿するために，サーバがノイズを加える再帰的紛失通信を利用する．そのため，クライアントは検索文字列を 1 文字ずつ暗号化した上で送信し，返ってきた結果を利用して次の文字に対する問い合わせを作成する．また，クライアントは受け取った演算結果同士の比較によって演算結果を得ることができる [5] [10]．

またゲノム秘匿検索を高速化するために，先行研究ではゲノムデータベースを Positional Burrows-Wheeler Transform (PBWT) と呼ばれる離散データ構造に変換している．ここでは PBWT に関する詳細な説明は割愛するが，概要としては列ごとにソートを行うことで，データベースを全て読み取ることなくゲノムデータのマッチ判定を行う計算量を大幅に削減する工夫である．他にもクライアントがサーバに伝える検索ポジションの情報にダミーポジションも加えることで，サーバ側にクライアントが検索している箇所を秘匿する工夫など，高速化や秘匿性の向上のための工夫がなされている．

4. 提 案 手 法

本研究では，以下の完全準同型暗号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．提案システムの概要を図 1 に示す．

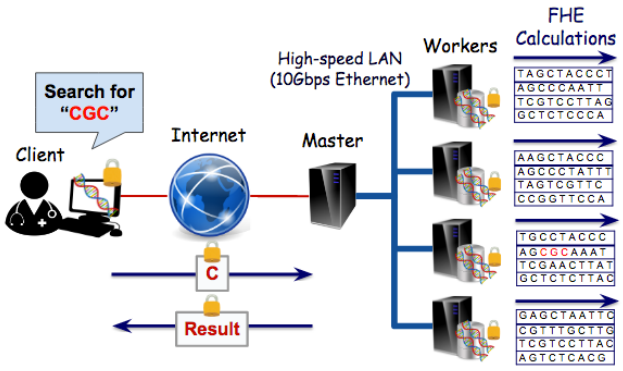

図 1　提案手法概観

(1)　クライアントはクエリの 1 文字を暗号化し，公開鍵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)　各ワーカは完全準同型暗号を用いた演算を行い，結果をマスタへ転送する．

(4)　マスタは結果を収集し，クライアントへ結果を送信する．

(5)　クライアントは復号を行うことで結果を得る．また，その結果を用いてクエリの次の 1 文字を暗号化した後に再びマスタに送信する．

(6)　(2)〜(5) をクエリの長さの回数分繰り返し，クライアントは結果同士の比較でマッチを判定する．

以上のプロトコルを C++で実装した．このシステムの完全準同型暗号計算は，GitHub 上で公開されている準同型暗号計算ライブラリである HElib [11] で実装されている．また，分散化における各マシンの制御のための規格である，Message Passing

Interface(MPI) を利用するライブラリの Open MPI [12] を適用し，さらに C++拡張ライブラリの Boost MPI [13] によって制御を行うプログラムを実装した．

5. 分散方法

今回適用するアプリケーションの分散化手法として，個体のデータごとでデータベースを分割するデータの分散，独立性の高い計算部位に適用する分散処理，また独立性の高い手順に適用する分散処理などが考えられる．先行研究の秘匿計算手法は，クエリとデータベースの要素に対して完全準同型暗号演算を行う手法を適用している．そのため，分散化した各ワーカマシンにデータベースを設置することにより，クエリとデータベースの要素間の暗号演算が可能となり，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る．したがって今回の提案システムでは，もっともシンプルなデータベースの分割による分散処理を適用した．

6. 実験

6.1 実験環境

実装したプログラムを複数のマシンにおいて実行した．各マシンの環境は，Intel® Xeon ® Processor E5-2643 v3 3.4 GHz，6 コア，12 スレッド，メモリ容量 512 GB，ストレージは RAID 0 の SSD が 480 GB，HDD が 2 TB であり，同スペックのマシンを 4 台使用する．1 台をマスタの機能を持ったマシンとし，同時にワーカとして 1 スロット分の演算も行う．また他 3 台をワーカとして最大 2 スロット稼働させた．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 2,184 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．さらに秘匿検索の秘匿性を高めるダミー検索を加えることにより，データベース上の 50 箇所を始点とした文字列検索を行った．

6.2 ワーカ数ごとのマスタとクライアントにおける実行時間の評価

クエリとデータベースとのマッチングの有無の判定を行うプログラムを分散化した環境上で稼働させた．この実験を 3 回試行し，ワーカ数ごとのマスタとクライアントの平均実行時間のグラフをそれぞれ図 2，図 3 に示す．

マスタ側の実行時間において，ワーカ数が増加するにつれて計算時間を減少させることができた．しかし，計算手順の中にデータベースの大きさに依存しない計算が含まれることから，ワーカ数が増えるにつれて計算時間における分散化効果は徐々に横ばいになっていることがわかる．またワーカ数の増加に対して，クライアントにおける復号計算にかかる時間とと通信時間を含めた待機時間はほとんど変化しないため，クライアントとの相互通信におけるオーバヘッドは小さいと考えられる．

クライアント側の実行時間においては，現在マスタ側の計算が終了するまでクライアントは待機する仕様となっているため，マスタの計算時間の分だけ通信時間がかかる．クライアントにおける復号計算時間はワーカ数による大きな変化は見られなかった．

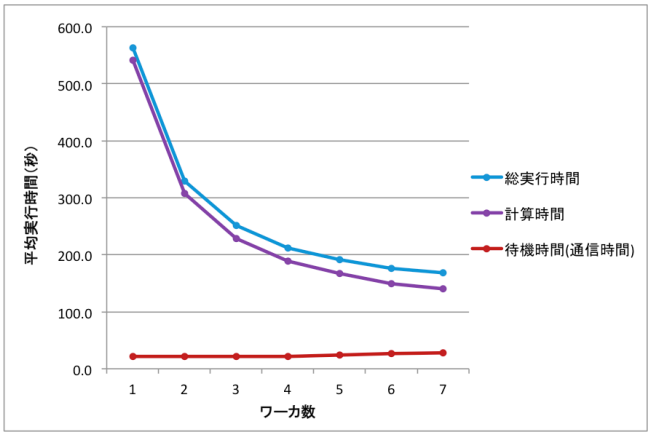

図 2　ワーカ数ごとのマスタにおける平均実行時間 (秒)

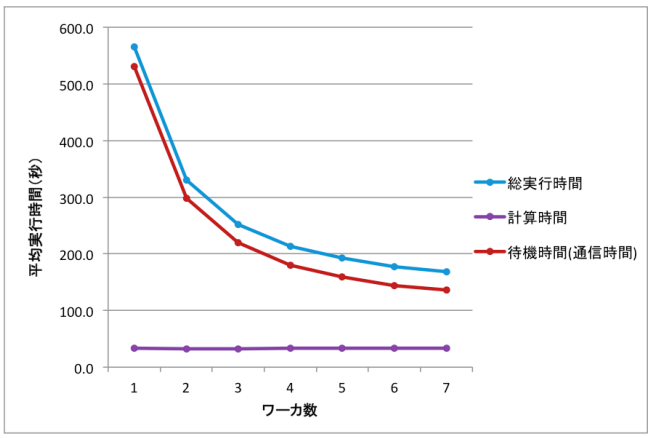

図 3　ワーカ数ごとのクライアントにおける平均実行時間 (秒)

またこの実験のワーカ数 7 における各ワーカの計算時間を比較した結果を表 1 に示す．

表 1　ワーカ数 7 における実験の各ワーカの計算時間

ワーカ数	1	2	3	4	5	6	7
計算時間	120.34	119.02	119.17	119.33	119.52	119.85	119.89

この表より，稼働するワーカ同士の計算時間の差は見られないことがわかる．また他のワーカ数においても，ワーカ間の計算時間の差はわずかなものであった．現在の実装ではマスタと各ワーカのデータの送受信は同期的に行われている．今回適用したゲノム秘匿検索では，単純なマッチングの有無の判定であったため，ワーカ間における計算時間差が見られないと考えられる．したがって，マスタ・ワーカ型の分散処理を適用した際の同期待ち時間のオーバヘッドは非常に小さいと言える．

6.3 ポジション数ごとの分散効率の評価

秘匿性を高めるダミーポジションの付加によって検索ポジションの総数を変化させた実験を行った．ポジション数とワーカ数における分散処理の高速化率を比較した結果を図 4 に示す．また分散化の評価となる高速化率は，式 (3) より算出した [14].

高速化率 = 逐次実行時間 (秒)/並列実行時間 (秒)　　(3)

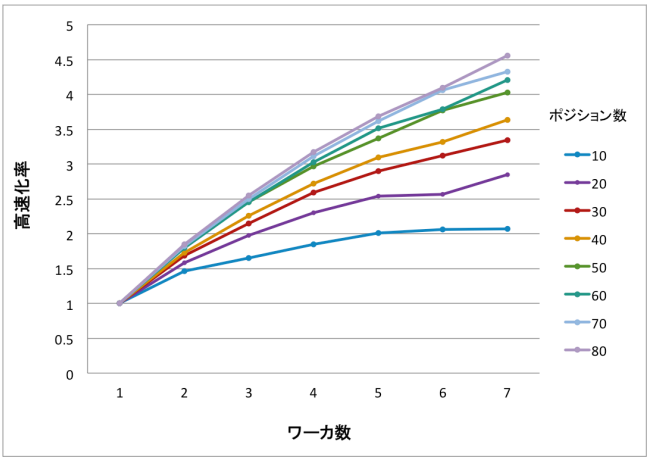

図 4　ポジション数におけるワーカ数ごとの高速化率

ワーカ数とポジション数の増加に伴い高速化率が上昇することが図 4 から読み取れる．3.2 章で先述したように，先行研究はゲノム秘匿検索の高速化のために様々な手法が用いている．特に PBWT 構造のデータベースに再帰的紛失通信を用いることによって，クエリの検索はデータベースを隈なく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高速化手法が行われている．そのためポジション数が少ない実験においては，前処理などの並列分散化できない演算の計算時間の割合が，並列計算可能な演算に対して増加するため，影響が大きくなった結果，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，データベース上の検索を開始する箇所であるポジションがデータベース上に散在して演算範囲が広がるため，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．

ここで参考のために，システムの並列度と期待される高速化率の関係性に関するモデルとして Amdahl の法則に基づいた考察を行う．Amdahl の法則は，処理内容のうち並列実行計算と逐次実行計算の割合と高速化率の限界，つまり理想的な状態において期待できる高速化率の関係性に関する法則である [14]．Amdahl の法則には公式が複数存在するが，今回は最もシンプルな以下の (4) の式を適用したモデルを考える．

$$\text{高速化率} \leqq \frac{1}{(1 - \text{並列実行時間の割合}) + \frac{\text{並列実行時間の割合}}{\text{分散コア数}}} \quad (4)$$

この式に様々な並列実行時間の割合を代入し，高速化率を算出したグラフが図 5 である．

図 4 と図 5 を比較すると，ポジション数を 80 に設定した際の高速化率の変化より，並列実行割合は 90%程度と考えられる．同様にポジション数を 10 に設定した際は，並列実行割合が 50%よりも少し高い程度だと考えられる．また Amdahl の法則では，無限の分散コア数を想定した際に期待できる高速化率も予測できる．例えば 90%の並列実行割合では，無限の分散コア数を適用しても 10 倍以上の高速化は得られないことが予測できるとされている．

したがって，ポジション数にしたがってデータベースとの演算量が変化し，並列実行時間の割合が変わるため，図 4 のような高速化率が変化したと考えられる．またポジション数が多い実験の方がその割合が大きいため高速化率の向上が見られたと考えられる．

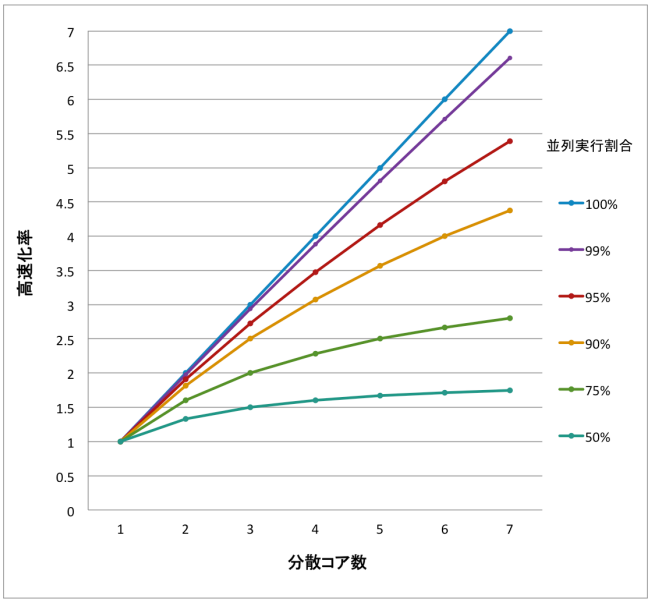

図 5　Amdahl の法則に基づいた並列実行時間の割合別高速化率予測曲線

今後は，データベースの分割による分散処理の効果が得られるような並列実行割合が高いゲノムデータ統計処理に本手法の適用を考えたい．

7.　クラウドコンピューティング化にむけて

現在，クラウドコンピューティングを想定した実験環境での本手法の適用を考えている．今回実施した実験は，全てお茶の水女子大学内で構成された，通信速度 10 Gbps のイーサネット上で通信するクラスタで行った．そのため実際のクラウド環境に本手法を適用する際には，サーバとクライアントのやりとりに対してインターネット回線を通過する必要があることから，現状よりも通信時間がかかると想定される．そのため，図 6 のようにお茶の水女子大学と早稲田大学にマシンを設置し，その間を Virtual Private Network(VPN) による接続を行い，ゲノム委託計算システムのクラウド環境を想定した実験を行う予定である．なお早稲田大学側に設置するマシンスペックは，Intel® Xeon ® Processor E7-8880 v3 2.3 GHz，72 スレッド，メモリ容量 1 TB であり，クライアントとして使用する．またお茶の水女子大学側，早稲田側双方にヤマハ ギガアクセス VPN ルーター RTX1210 を設置し，VPN 環境を構築して実験を行う予定である．

8.　関 連 研 究

今回，石巻らが 2016 年 1 月に提案した完全準同型暗号を用いたゲノム秘匿検索 [5] に関して分散処理を適用した．これは 3.2 章で述べたように，サーバとクライアント間で検索クエリの文字列の長さの回数の通信を行う必要がある手法である．そのため，マスタ・ワーカ側が処理を終えた後に通信を行い，ク

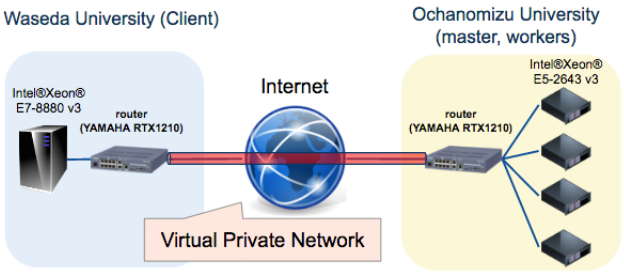

図 6　クラウド環境を想定した実験環境

ライアントが復号演算を行う間マスタは待機し，クライアントから再度クエリを受け取り，統計演算を行うというステップを踏む必要があり，影響は大きくないものの高速化を阻む要因であった．しかしその後，石巻らより 2016 年 12 月にブートストラップ法を使用することで，複数回の通信を用いる必要の無いノンインタラクティブなゲノム秘匿検索の高速化手法 [15] が提案されている．複数回の相互通信を用いないモデルであれば，クライアントの復号演算時間をマスタ・ワーカ側が待つ必要が無いため，分散処理手法を適用する際に通信によるオーバヘッドの影響がより小さくなると考えられる．今後はこのような分散処理の効果が期待されるゲノム秘匿検索手法に対する本手法の適用も検討したい．

また本研究ではゲノムデータを用いた秘匿検索に対する分散処理を行ったが，近年ゲノムデータの検索以外の統計的処理の秘匿計算システムの研究が多く見受けられる．特に Lu らによる完全準同型暗号を用いたゲノムデータに対する統計処理の秘密計算システム [16] では，カイ二乗検定などの様々な統計処理を実現可能にする提案がなされている．これにより，個人ゲノム情報と疾患の統計的な関係性の算出や患者の個人ゲノム情報に対応する個別化医療の発展に役立つことが期待されている．今後も完全準同型暗号によるゲノム統計処理の秘密計算システムの研究が盛んになると考えられる．

9. まとめと今後の課題

完全準同型暗号を用いたクライアント・サーバ型ゲノム秘匿計算のサーバ側の処理に，マスタ・ワーカ型の分散処理システムを適用し実験を行った．今回は分散処理方法としてデータベースの分割を適用した．その結果マスタ側の計算時間が分散台数に応じて減少し，通信時間も含めた待機時間はほぼ変わらない結果となった．また高速化率による評価を行った結果，検索のデータベース上の開始点を示すポジション数が多い方が高速化率が高いことが判明した．これはポジションが増加するほどデータベースとの演算が多くなるため，データベースによる分散の効果が現れやすいからだと考えられる．

今後はクラウドコンピューティングを想定した環境での実験を行うとともに，異なる分散方法による分散処理実装を行い，より高速な分散処理が可能な手法を考案していきたい．また本手法が効果的に作用すると考えられる完全準同型暗号を用いたゲノムデータの統計処理への適用も提案していきたい．

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に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 の支援を受けております．

文　　献

[1] DNA Data Bank of Japan : 「遺伝子とゲノム」, http://www.ddbj.nig.ac.jp/infobio/genegenome-j.html, 2017 年 1 月閲覧
[2] 下山武司 :「暗号を解かずにデータ処理」,「情報処理」, Vol.57, No. 1, pp 44–50, 2016 年 1 月
[3] 草川恵太 :「完全準同型暗号の概要」, 電子情報通信学会誌, Vol. 99, No. 12, pp. 1151–1158, 2016 年 12 月
[4] Kana Shimizu, Koji Nuida and Gunnar Rtsch : “Efficient privacy-preserving string search and an application in genomics,” http://biorxiv.org/content/early/2015/11/27/018267, 2017 年 1 月閲覧
[5] 石巻優, 清水佳奈, 縫田光司, 山名早人 :「完全準同型暗号を用いた高速なゲノム秘匿検索」, SCIS 2016, 2A2-2, 2016 年 1 月
[6] 安田雅哉 :「完全準同型暗号の応用」, 電子情報通信学会誌, Vol. 99, No. 12, pp. 1167–1175, 2016 年 12 月
[7] Craig Gentry : “Fully Homomorphic Encryption Using Ideal Lattices,” STOC, Vol. 9, No. 2009, pp. 169–178, 2009
[8] Tibouchi Mehdi :「整数上完全準同型暗号の研究」, NTT 技術ジャーナル, Vol. 26, No. 3, pp. 71–75, 2014 年 3 月
[9] 佐藤宏樹, 馬屋原昂, 石巻優, 今林広樹, 山名早人 :「完全準同型暗号のデータマイニングへの利用に関する研究動向」, 第 15 回情報科学技術フォーラム 2016, F-002, 2016 年 9 月
[10] Yu Ishimaki, Kana Shimizu, Koji Nuida, Hayato Yamana : “Poster:Privacy-Preserving String Search for Genome Sequences using Fully Homomorphic Encryption,” the 37th IEEE Symposium on Security and Privacy, 2016 年 5 月
[11] Shoup V. and Halevi S. : HElib, http://shaih.github.io/HElib/index.html, 2017 年 1 月閲覧
[12] Open MPI : https://www.open-mpi.org/, 2017 年 1 月閲覧
[13] Boost : http://www.boost.org/, 2017 年 1 月閲覧
[14] Clay Breshears : “The Art of Concurrency,” 2009 年 5 月
[15] Yu Ishimaki, Hiroki Imabayashi, Kana Shimizu, Hayato Yamana : “Privacy-preserving string search for genome sequences with FHE bootstrapping optimization,” Big Data 2016 IEEE International Conference, pp. 3989–3991, 2016
[16] Wen-Jie Lu, Yoshiji Yamada, Jun Sakuma : “Privacy-preserving genome-wide association studies on cloud environment using fully homomorphic encryption,”BMC medical informatics and decision making, Vol. 15, No. 5, pp. S1, 2015
[17] 有田正剛 :「イデアル格子暗号入門」, https://www.iisec.ac.jp/proc/vol0006/arita14.pdf, 2017 年 1 月閲覧
[18] Peter Pacheco : “Parallel Programming with MPI,” 1996 年
[19] Shoup V. and Halevi S. : “Algorithms in HElib,” http://eprint.iacr.org/2014/106, 2017 年 1 月閲覧
[20] The 1000 Genome Project Consortium : “An integrated map of genetic variation from 1,092 human genomes,” Nature, Vol. 491, pp. 56–65, 2012.
[21] 中村蓉子 :「完全準同型暗号の実装」, http://siio.jp/pdf/grad/2010/2010grad55.pdf, 2017 年 1 月閲覧