# A Decentralized System of Genome Secret Search Implemented with Fully Homomorphic Encryption

Yuri Yamamoto, *Ochanomizu University* and Masato Oguchi, *Ochanomizu University*

***Abstract*—An outsourcing system for calculating the statistics of genome data has been proposed as a method for utilizing genomic data in bioinformatics research. Individual genome data need to be encrypted from the viewpoint of privacy protection. In previous research, a client/server system was proposed for the string search of genome sequences using fully homomorphic encryption. The authors also improved the query and calculation algorithms for taking more advanced statistics of the genomic data for use in the future. However, calculations using fully homomorphic encryption are highly complex. In this study, we proposed the implementation of a master/worker decentralized system on the server side of a string search system for the genome using fully homomorphic encryption. We propose a decentralized system that is operated in a cloud computing environment.**

***Index Terms*—Decentralized System, Fully Homomorphic Encryption, Genome Secret Search**

## I. Introduction

In bioinformatics research, it is necessary to utilize the statistics of genomic data possessed by research institutes, hospitals, and other organizations. In general, since the human genome consists of approximately 3 billion bases, it is necessary to calculate the statistics of the genome data on high-performance computers [1]. It is difficult to provide high-performance computers to each organization and utilize the data of other organizations. Therefore, we believe that an outsourcing system, which can possess storage for the large amount of genomic data for each organization and high-performance computers for calculating the genome statistics, so that each organization can query the system and get results, will be widespread in the future. However, since the human genome is an important personal identifier, it is necessary to protect the privacy of the data using encryption.

The use of a common key cryptosystem is the usual method to develop a client/server system for genome string search. However, to calculate the statistics of encrypted data with complex expressions, it is necessary to send the key to decrypt the sensitive data on the server side. It is difficult to protect the privacy of the client's genomic data when it is sent to the server side. Previous research [2] applied additive homomorphic encryption, which is able to perform the addition of ciphertexts to the algorithms of string search for the genome, but there are complex and heavy calculations that are difficult to perform to prevent the leakage of the genomic data from the results on the server [3].

Previous research [3] proposed a secure genome search protocol using fully homomorphic encryption (FHE) that supports both addition and multiplication in encrypted form to calculate the similarity between a query and a database. Then, they protected the data using the recursive oblivious transfer protocol and optimized the algorithm by making use of a discrete data structure. However, the calculation time on the server side tends to be excessive because of large computational complexity of the FHE. In this research, we adopted a master/worker decentralized system for the calculation of the server side to improve the runtime of the genome secret search for an experiment on the system of the cloud computing.

## II. Fully Homomorphic Encryption

FHE is a cipher that supports both the addition and multiplication operations of ciphertexts like expressions (1) and (2). This property enables the polynomial calculation of ciphertexts like plaintexts. Although FHE provides a function of public key cryptography, it is possible to obtain the encrypted results of the operation on the plaintexts from the operation on ciphertexts without using the secret key. Therefore, applying FHE to the genome secret search makes it possible to perform the statistical processing of the genomic data without passing a secret key.

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

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

Public key cryptography was first devised in the latter half of the 1970s, and Gentry [4] proposed the method of the FHE algorithm in 2009. First, it seemed to be difficult to use in applications because of its large runtime, but various improvements have been made to the algorithm of FHE to make it adequate for simple calculations. However, there is difficulty with the large computing complexity of the FHE calculations. The size of FHE

cypher text tends to be large because it is necessary to add noise to maintain the difficulty of decryption of the calculations on the server side as well as a means of removing the noise at the time of the calculation of the client's decryption.

## III. Previous Research

In this section, we provide an overview of the genome secret searching method by FHE as implemented by previous research[3].

### A. Problem Setting

We describe an assumed model case in a two-party genome secret search system. A server holds a set of genome sequences aligned by each sample in a database. Since the genomic data are composed of sequences of 4 different nucleotides, A, G, C and T, the implemented genome secret search is regarded as a 4-kind-character search. The genomic data uses single-nucleotide polymorphism(SNP) sequences, which are specific positions where individual differences are likely to appear, rather than the entire genome sequences. By arranging the long genome sequence of each sample in a row, it is possible to evaluate the differences between samples in a specific position of the genome sequence of each column.

A client making an inquiry transmits it to the server together with the starting point of the search (search position) to calculate the number of matches with the query and database. Then, the server performs calculations on the data and transmits the result, which is the length of the matched characters of the query to the database, to the client. The goal of the previous research was to perform this search as fast as possible while preserving the privacy of the data on both the server side and client side.

### B. Method of the Previous Research

Ishimaki et al. (2015) designed a genome secret search method that is built on FHE for obtaining more advanced statistics of the genomic data for use in the future. They reduced the runtime of the system by a packing technique that enables encrypting an integer vector into one ciphertext. The method was tested as a server/client system, which consists of one server connected with one client. The overview of the system is that a server receives a character of the query from a client, performs matching with the genomic data that it possesses, and then returns the result to the client. At this time, to preserve the privacy of both the server and the client from each other, the server uses a recursive oblivious transfer protocol to add noises [2]. In the protocol, the client encrypts a character of the query, transmits it, and creates an inquiry for the next character by using the returned result. The client can obtain the result by comparing the received result with the data prior to the communication.

To reduce the runtime of the method, the genomic database is converted to the arrangement of a positional Burrows-Wheeler transform (PBWT) which is a discrete data structure used to search for a substring match for a set of aligned genome sequences. Because of the PBWT and the recursive oblivious transfer protocol, the system performs its calculations using the data of only a part of the database. Adding dummy search positions to the information of the search position that the client sends to the server also preserves the privacy of the data. As described above, various methods have been used for speeding-up the system and preserving privacy.

## IV. Proposed Method

### A. Overview of Proposed Method

In this research, we propose a master/worker decentralized system of genome secret search using FHE. An overview of the proposed system is shown in Fig1.

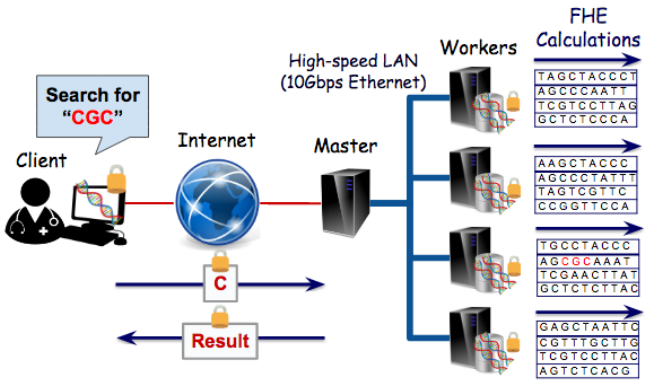


Fig. 1. Overview of the Proposed System

(1) The client encrypts one character of the query and sends it to the master with the public key.
(2) The master transfers the received data to each worker.
(3) The workers calculate the result using FHE and transfer it to the master.
(4) The master collects the result and sends it to the client.
(5) The client gets the result by decoding the received data and then, using the result, encrypts the next character of the query and sends it to the master.

(6) (2)-(5) are repeated as many times as the length of the query, and the client obtains a match by a comparison between the initial result and the final result.

These protocols were implemented in C++. The FHE calculations of this system are implemented with HElib [5], which is a homomorphic encryption calculation public library on GitHub. We also use Open MPI [6], which is one library implementation of Message Passing Interface (MPI) [7], which is a standard control and communication system for the machines in decentralization. To use the MPI library efficiently, we use Boost MPI [8] from the C++ extension library.

### B. Decentralized System

As a method of decentralization of the application, we consider the following three approaches; decentralizing each individual data, decentralizing each independent calculation, or decentralizing independent algorithm. Most of the calculations of the previous research system are FHE calculations between a character of the query and the database, and they are independent for each database. Therefore, in the proposed system, the function of the workers was implemented by considering the division by database of the server side.

### C. Cloud Computing

We did an experiment using the method on a cloud environment like that in Fig.2. When applying this method to an actual cloud environment, it is assumed that a longer communication time will be needed than that in the experiment in a local environment because it will be necessary to include the transmission through the Internet along with the server and client interaction. The master/worker cluster of this system on the Ochanomizu University side communicates on 10 Gbps Ethernet. Therefore, communication between the master and workers is very fast, even if the throughput of communications on the Internet between Ochanomizu University and Waseda University is low.

## V. Experimental Results and Discussion

### A. Experimental Environment

The program of the proposed system was executed on 4 homogeneous machines with an Intel ® Xeon ® Processor E5-2643 v3 3.4 GHz, 6 cores, 12 threads, memory capacity 512 GB, RAID 0 SSD 480GB, HDD 2TB. One machine has the function of master and one slot with the function of a worker at the same time, while the other machines each have 2 slots with the function

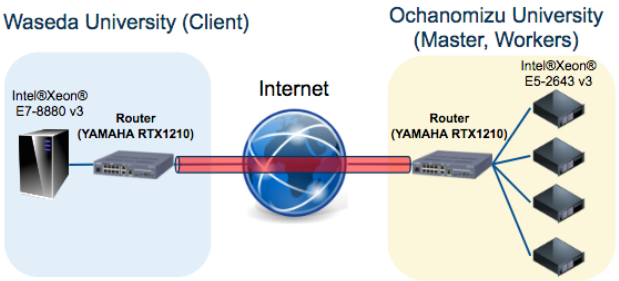


Fig. 2. Experimental Evironment for Cloud Computing

of a worker. We conducted experiments to compare the execution time for different numbers of workers by operating with workers in up to seven slots. The genomic data used for the experiment was 2,184 samples with 10,000 characters per sample. The user's query string had a length of 5. Dummy searches are also conducted to improve the security, starting from 50 locations in the database.

### B. Evaluation of execution time for master and client for each number of workers

We operated a program for calculating the matching between the query and the database on the decentralized system. This experiment was run 3 times, and graphs of the average execution time of the master and client by the number of workers are shown in Fig. 3 and Fig. 4, respectively.

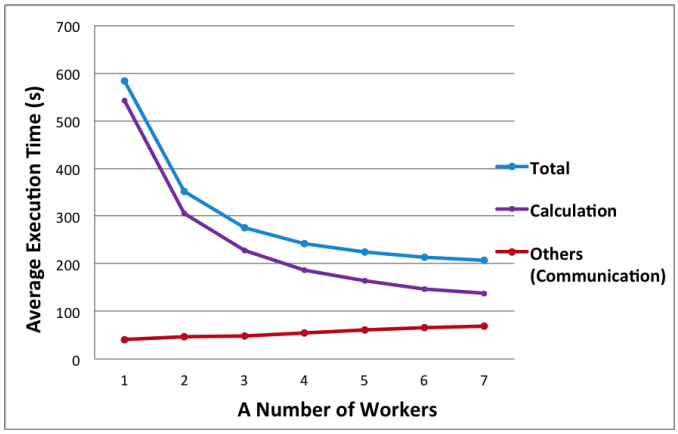


Fig. 3. Average execution time of the master by the number of workers(s)

As the number of workers increases, the calculation time is reduced in the execution time on the master side, as shown in Fig. 3. However, the effect of the decentralization on the calculation time is gradually flattened. In addition, as the number of workers increases, the waiting

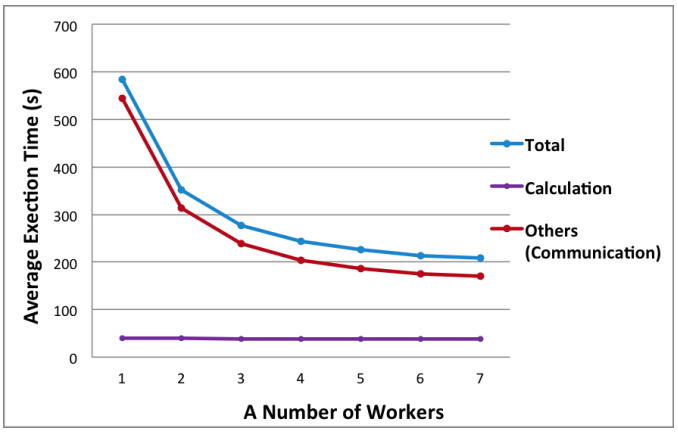


Fig. 4. Average execution time of the client by the number of workers(s)

time, including the decoding time of the results by the client and communication time, does not much change, so the overhead in the mutual communication with the client is considered to be small.

In the execution time on the client side, as shown in Fig.4, the client waits until the calculation on the master side is complete, and the calculation time for decoding the results on the client side does not much change with the number of workers.

The results of comparing the calculation time of each worker in the experiment with 7 workers are shown in Table I.

TABLE I
CALCULATION TIME OF EACH WORKER IN EXPERIMENT WITH 7 WORKERS

| Worker Name | Calculation Time |
|---|---|
| Worker 1 | 120.78 |
| Worker 2 | 119.53 |
| Worker 3 | 119.58 |
| Worker 4 | 119.76 |
| Worker 5 | 119.76 |
| Worker 6 | 120.21 |
| Worker 7 | 120.31 |

From Table I, there is almost no difference in the calculation time for different numbers of workers. Additionally, in the case of a specific number of workers, the difference in the calculation time between workers is very slight. In the implementation of the experiment, the data transmission and reception between the master and each worker are performed synchronously. The system for the genome secret search performs simple matching, so there is no difference in the calculation time between workers. Therefore, the overhead of the synchronization for the waiting time in the master/worker decentralization is minimal.

### *C. Evaluation of decentralization efficiency by the number of positions*

We conducted an experiment changing the number of search positions by adding dummy positions. Fig. 5 shows the result of the experiment with the number of search positions for each number of workers, and Fig. 6 shows the decentralization efficiency of the experiment. The decentralization efficiency is calculated from the expression (3) [9].

$$\text{Efficiency} = \text{Sequential Execution Time (s)} \;/\; \text{Parallel Execution Time (s)} \qquad (3)$$

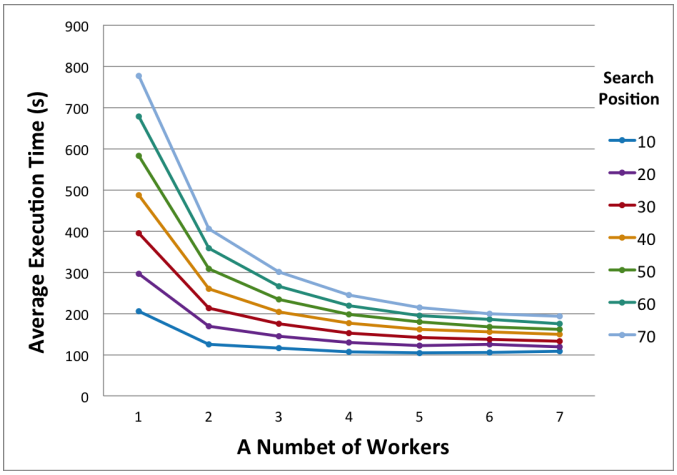


Fig. 5. Average execution time of the master by the number of positions for each number of workers

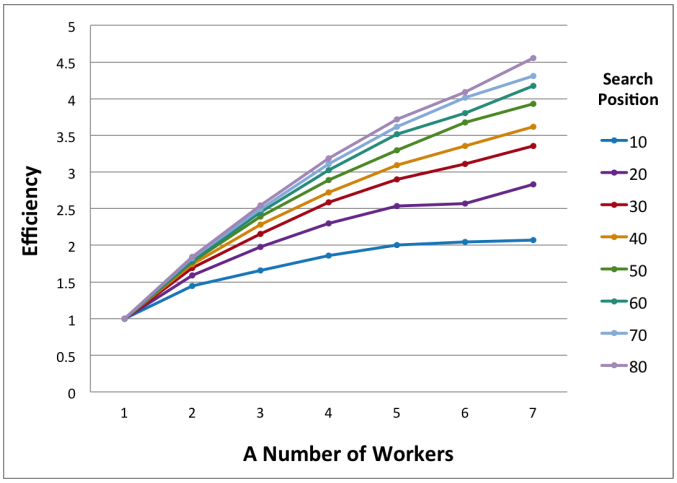


Fig. 6. Decentralization efficiency by the number of positions for each number of workers

From Fig. 6, it can be seen that the efficiency increases with the numbers of workers and search positions. As mentioned in the section on III-B, various methods have

been used in previous research for speeding up the genome secret search. Particularly in the methods of the recursive oblivious transfer protocol and the PBWT structure for the database, the query searching does not need to search the entire range of the database but rather only the range of the length of the query from the search positions. Therefore, in experiments with fewer search positions, the proportion of the calculation time of the sequential execution part, which includes steps such as preprocessing, increases compared with the parallel calculation, so the decentralization does not significantly improve the efficiency. On the other hand, in experiments with many search positions, the effect of decentralization in the database is more likely to appear, because the search positions that are the start bases of the search are covered on the database, and the calculation range expands.

We consider Amdahl's law as a model of the relationship between the degree of parallelism of the system and the expected efficiency of the experiment. Amdahl's law gives an upper bound on the speedup efficiency of the parallel execution of a task [9]. Amdahl's law can be explained by several formulas, but we consider a model that applies the simplest formula(4).

$$\text{Efficiency} \leqq \frac{1}{(1 - pctPar) + \frac{pctPar}{p}} \tag{4}$$

Where pctPar is the percentage of the execution time that will be run in parallel, and p is the number of cores on which to run the parallel application.

Fig. 7 is a graph of the efficiency calculated with various percentages of parallel execution time from formula(4).

When comparing Fig. 6 and Fig. 7, it is considered that the percentage of parallel execution time of the experiment with 80 search positions is approximately 90%. Similarly, when setting the number of positions to 10, the percentage of the parallel execution time is considered to be slightly higher than 50%. We can also calculate the efficiency when assuming an infinite number of workers from Amdahl's law. For example, at a 90% parallel execution ratio, the efficiency of the experiment with 80 search positions will be less than 10 even using an infinite number of workers.

Therefore, as the amount of calculation with the database changes according to the number of positions, the percentage of the parallel execution time also changes. Then, the decentralization efficiency is changed like in Fig. 6, and the efficiency is high in the experiments with many positions.

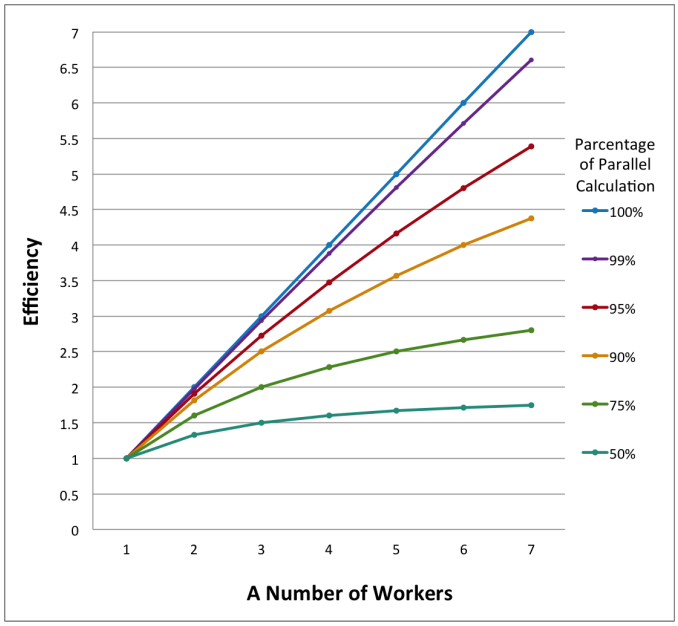


Fig. 7. Theoretical efficiency of the parallel calculation time based on Amdahl's law

## VI. Conclusion

We designed a master/worker decentralized system for genome secret search using FHE on the server side and conducted an experiment in an environment assuming cloud computing. The system applies decentralization by a sample of the database. As a result, the calculation time on the master side decreased according to the number of workers, and the waiting time was almost always short. When evaluating the decentralization efficiency, an experiment with many search positions indicating the starting point on the database showed a high efficiency of decentralization. We compared the result with the ideal efficiency calculated by the Amdahl's law.

In the future, we will design a method for faster decentralization such as by utilizing a data structure and cache. Additionally, a genome secret search system using FHE that does not require character-by-character communication has been proposed, so we also want to adapt decentralization to the proposed system [10].


## Acknowledgment

I would like to thank the members of the Yamana Laboratory of Waseda University and the Yamaguchi Laboratory of Kogakuin University for their valuable advice.

Yu Ishimaki from Waseda University, in particular, provided me with much of advice on the programming of the genome secret search system that I deeply appreciate.

This work was partly supported by JST CREST Grant Number JPMJCR1503, Japan.

## References


[1] 1000 Genomes Project Consortium et al. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, 2012.

[2] Kana Shimizu, Koji Nuida, and Gunnar Rätsch. Efficient privacy-preserving string search and an application in genomics. *Bioinformatics*, 32(11):1652–1661, 2016.

[3] Yu Ishimaki, Kana Shimizu, Koji Nuida, and Hayato Yamana. Poster: Privacy-preserving string search for genome sequences using fully homomorphic encryption. *IEEE Symposium on Security and Privacy*, 2016.

[4] Craig Gentry et al. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.

[5] Shoup v. and halevi s. http://shaih.github.io/HElib/index.html. Accessed: 2017-1.

[6] Open mpi. https://www.open-mpi.org/. Accessed: 2017-1.

[7] Peter S Pacheco. *Parallel programming with MPI*. Morgan Kaufmann, 1997.

[8] Boost. http://www.boost.org/. Accessed: 2017-1.

[9] Clay Breshears. *The art of concurrency: A thread monkey's guide to writing parallel applications*. " O'Reilly Media, Inc.", 2009.

[10] Yu Ishimaki, Hiroki Imabayashi, Kana Shimizu, and Hayato Yamana. Privacy-preserving string search for genome sequences with fhe bootstrapping optimization. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 3989–3991. IEEE, 2016.

[11] Wen-Jie Lu, Yoshiji Yamada, and Jun Sakuma. Privacy-preserving genome-wide association studies on cloud environment using fully homomorphic encryption. *BMC medical informatics and decision making*, 15(5):S1, 2015.

[12] Shai Halevi and Victor Shoup. Algorithms in helib. Cryptology ePrint Archive, Report 2014/106, 2014. http://eprint.iacr.org/2014/106.