# Privacy-Preserving Genetic Algorithm Outsourcing in Cloud Computing

**Leqi Jiang$^{1, 2}$ and Zhangjie Fu$^{1, 2, *}$**

**Abstract:** Genetic Algorithm (GA) has been widely used to solve various optimization problems. As the solving process of GA requires large storage and computing resources, it is well motivated to outsource the solving process of GA to the cloud server. However, the algorithm user would never want his data to be disclosed to cloud server. Thus, it is necessary for the user to encrypt the data before transmitting them to the server. But the user will encounter a new problem. The arithmetic operations we are familiar with cannot work directly in the ciphertext domain. In this paper, a privacy-preserving outsourced genetic algorithm is proposed. The user's data are protected by homomorphic encryption algorithm which can support the operations in the encrypted domain. GA is elaborately adapted to search the optimal result over the encrypted data. The security analysis and experiment results demonstrate the effectiveness of the proposed scheme.

**Keywords:** Homomorphic encryption; genetic algorithm; outsourcing; cloud computing.

## 1 Introduction

Cloud computing has been attracting more and more attention as its great storage space and strong computing power. It is quite usual to outsource complex work to cloud server. Cloud server is often considered semi-honest, which means the cloud server will follow the instructions received to complete the corresponding task, but at the same time it will record the data generated in the calculation process [Cao, Wang, Li et al. (2011)]. In order to ensure the user's data privacy, it is very important to encrypt the data before they are outsourced to the cloud server.

GA performs well on many optimization problems. It is used to optimize the problems of traveling salesman [Li, Sun, Zhou et al. (2014)], resource allocation [Li, Zhang and Yu (2012)], siting and sizing of distributed generators [Fu, Lai and Liang (2014)], distribution system reconfiguration [Xue, Jiang, Zhao et al. (2017)], carpool matching [Jiau and Huang (2016)] etc. Because GA could optimize the problem and do not require any expertise related to the problem, it can be applied to any problem that needs to be optimized.

[1] College of Computer and Software, Nanjing University of Information Science & Technology, Nanjing, 210044, China.

[2] Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing, 210044, China.

[*] Corresponding Author: Zhangjie Fu. Email: wwwfzj@126.com.

However, when the genetic algorithm is used to solve the high dimensional problem, the computational complexity is large and the time complexity of solving the problem is high, it is difficult to run on the local computer [Berlanga, Rivera, Jesus et al. (2010)]. In the big data age, the more data, the better effect of genetic algorithm. At the same time, the calculation will take much more cost. And the raw data to be optimized may not be leaked to the provider of the genetic algorithm. Therefore, it is proposed to outsource the genetic algorithm to the cloud server. The user encrypts the raw data and uploads them to the cloud server. The cloud server optimizes the problem while keeping the data private.

**Contribution.** A privacy-preserving genetic algorithm outsourcing in cloud computing scheme is proposed in this paper. A cloud server becomes privacy-preserving genetic algorithm provider. The algorithm user could protect the raw data from revealing to the algorithm provide. The major contributions are summarized as follows:

1) For the first time, this paper proposes a privacy-preserving scheme that outsourcing the genetic algorithm to the cloud server. The heavy computing tasks are outsourced to the cloud server, so that the user only needs to encrypt data and upload them to the cloud server. The data will not be leaked out to the cloud server.

2) An incomplete re-encryption is designed to compare the magnitude of two cipher data encrypted with SH.E, which has never been realized before.

## 2 Related work

A cloud server has a large storage capacity, which provides good storage and extraction services. The datasets usually are encrypted before outsourcing to preserve the privacy. Searchable encryption has been widely used to support data outsourcing [Fu, Wu, Guan et al. (2016b)]. There are different types of ciphertext retrieval method like multi-keyword ranked search [Xia, Wang, Wu et al. (2015)], semantic search [Fu, Ren, Shu et al. (2016a)], and so on. However, the ciphertext retrieval is a method to search the encrypted data. And the cloud server is expected to take on more computing tasks.

Computing outsourcing is an important application of cloud serves. More and more companies want to outsource their work to the cloud server. Especially those problems with large amount of data could be solved by using the cloud computing. Li et al. put forward to use emerging cloud-computing technologies to solve data-intensive geospatial problems in urban traffic systems [Li, Zhang and Yu (2011)]. Balamurugan et al. provide a detailed investigation on solving NP-hard problems like scheduling and task allocation by using cloud computing [Balamurugan, Visalakshi, Prabhakaran et al. (2014)]. Cloud computing could be applied to solve problems in many fields like Industry [Yu and Wang (2018)], Economic [Coyle and Nguyen (2019)] etc.

Genetic algorithm also has a wide range of applications. It can be well linked to cloud computing. Task scheduling could be optimized by genetic algorithm, which may greatly improve the performance of cloud computing [Habibi, Motameni and Ramezani (2008); Zhan, Liu, Gong et al. (2015)]. Huang proposed a framework for solving the problem of carpool service problems, by using cloud servers to collect data and genetic algorithms to optimize the program [Huang, Jiau and Lin (2015)]. All the data in this framework is public, and private preserving is not considered. However, facing the problem that the data may be leaked is unavoidable in the process of using the cloud server. Gennaro has shown the theoretical possibility of a general result of secure computation outsourcing [Gennaro, Gentry and Parno (2010)] based on Yao's garbled circuits [Yao (1982)] and Gentry's breakthrough work on fully homomorphic encryption scheme [Gentry (2009)]. Based on that, an encrypted genetic algorithm outsourcing in cloud computing is proposed in this paper. The algorithm user could process the data in cloud servers without worrying about data leakage

## 3 Preliminariesns

As it could enable the genetic algorithm in the cloud server to support the data being processed in the encrypted domain, the homomorphic encryption algorithm is selected to encrypt the data. The specific parts are introduced as follows.

### *3.1 Homomorphic encryption*

Encryption will make data hard to be utilized. Some methods that support encrypted data operations have been proposed. Homomorphic encryption is a form of encryption that allows computation on ciphertexts, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext. [Jiang, Xu, Wang et al. (2015)]. For example, the plaintext corresponding to the product of two ciphertexts encrypted with Paillier (one of somewhat homomorphic encryptions) is equal to the product of the plaintexts corresponding to those two ciphertexts. An encryption algorithm that can support multiple alternative operations at the same time is called full homomorphic encryption, others are called somewhat homomorphic encryption.

However, the fully homomorphic encryption is difficult to be used because of the high computational complexity. A somewhat homomorphic encryption(S.HE) is mentioned in [Hu, Wang, Wang et al. (2016)]. This scheme is parametrized by the ring $R_q \triangleq \mathbb{Z}_q[x]/\langle x^n+1\rangle$ and a discrete Gaussian error distribution $\chi = D_{\mathbb{Z}^n,\sigma}$ with standard deviation $\sigma$, where $n$ is a power of two, $q$ is an odd prime number, $\sigma$ is a parameter that reflects the degree of dispersion. The message space of the scheme is defined by a prime $t$ as $R_t \triangleq \mathbb{Z}_t[x]/\langle x^n+1\rangle$. The tuple $(SH.Gen, SH.Enc, SH.Dec)$ is used to represent the key generation, the encryption and the decryption procedures in the scheme, respectively.

$SH.Gen$: Sample a ring element $s \leftarrow \chi$ and define the private key $sk \triangleq s$. Sample a uniformly random ring element $a_1 \leftarrow R_q$ and an error $e \leftarrow \chi$ and compute the public key $pk \triangleq (a_0 = -(a_1 s + te), a_1)$;

$SH.Enc(m,pk)$: For a message $m \in R_t$, the encryption algorithm samples $u \leftarrow \chi$, and $f, g \leftarrow \chi$, and compute the ciphertext $ct = (c_0, c_1) \triangleq (a_0 u + tg + m, a_1 u + tf)$.

$SH.Dec(ct,sk)$: To decrypt $ct = (c_0, c_1, \dots, c_\delta)$, compute $\widetilde{m} = F(ct,s)\sum_{i=0}^{\delta} c_i s^i$. Output the message as $\widetilde{m} \bmod t$.

This scheme can support homomor- phic addition and multiplication, its meaning is shown in formulas 1 and 2.

$$m + m' = SH.Dec\Big(\big(SH.Enc(m) \oplus SH.Enc(m')\big), sk\Big) \tag{1}$$

$$m \times m' = SH.Dec\left(\left(SH.Enc(m) \otimes SH.Enc(m')\right), sk\right) \quad (2)$$

In SH.E, $\oplus$ denotes addition of vectors in the plaintext domain and $\oplus$ is named addition operation in ciphertext domain. In particular, if there is a vector with less elements, the missing positions will be filled with zeros. And $\otimes$ denotes convolution in the plaintext domain and $\otimes$ is named multiplication in ciphertext domain.

Here is a simple example to introduce those homomorphic operations. In the introduction to the decryption process, in order to simplify the formula, all the variables multiplied by $t$ are recorded as $e_{error}$.

For $ct = SH.Enc(m, pk)$and $ct' = SH.Enc(m', pk)$, taking an additional operation will get a new ciphertext $ct'_{add} = (c_{a0}, c_{a1}) = (c_0 + c'_0, c_1 + c'_1)$.

To obtain the message, the following operation is needed.

$$\begin{aligned} F(ct'_{add}, s) &= \textstyle\sum_{i=0}^{1} c_{ai} s^i \\ &= \textstyle\sum_{i=0}^{1} (c_i + c'_i)\, s^i \\ &= te_{error} + m + m' \end{aligned} \quad (3)$$

The message $m + m'$ will be obtained by modular operation after that.

Taking multiplicative operation will get a new ciphertext $ct'_{mul} = (c_{b0}, c_{b1}, c_{b2})$. To determine $c_{bi}$ an unknown variable $k$ needs to be introduced. Considering the expression as follows

$$\left(\textstyle\sum_{i=0}^{1} c_i k^i\right)\left(\textstyle\sum_{j=0}^{1} c'_j k^i\right) = c_0 c'_0 k^0 + (c_0 c'_1 + c'_0 c_1) k^1 + c_1 c'_1 k^2 \quad (4)$$

So, the output ciphertext is $ct'_{mul} = (c_{b0}, c_{b1}, c_{b2}) = (c_0 c'_0, (c_0 c'_1 + c'_0 c_1), c_1 c'_1)$.

$$\begin{aligned} F(ct'_{mul}, s) &= \textstyle\sum_{i=0}^{2} c_{bi} s^i \\ &= c_0 c'_0 s^0 + (c_0 c'_1 + c'_0 c_1) s^1 + c_1 c'_1 s^2 \\ &= te_{error} + m \cdot m' \end{aligned} \quad (5)$$

Similarly, we can get the ciphertext by multiple additional or multiplicative operations. Brakerski et al. [Brakerski and Vaikuntanathan (2011)] has given a detailed introduction about SHE in his paper.

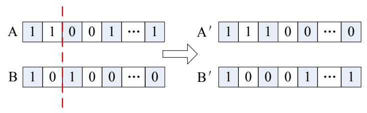
## *3.2 Genetic algorithm*

Genetic algorithm is a kind of evolutionary algorithm. It treats a feasible solution as an individual. Each individual is represented by a gene chain, and each gene is the value of the solution on the corresponding dimension. By simulating the genetic process of natural organisms, new individuals are generated through gene crossover or mutation. Then eliminate some bad individuals according to the principle of survival of the fittest, and the rest in the population will become better and better. To solve a M-dimensional problem, GA has the following main operations:

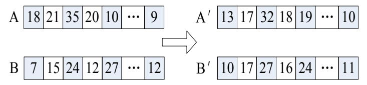
1) Initialization: initialize a population $P$ of $N$ individuals, each individual has a gene chain with *M*-length, $t$ represents the number of generation.

2) Selection: Select random individuals with a certain generation gap $G$. Generate $G \cdot N$ new individuals by gene recombination and variation.

3) Crossover: Point crossover or arithmetic crossover is done for two individuals. The two operations are shown in Fig. 1.



(a)



(b)

**Figure 1:** Recombination. (a) One-point crossover (b) Arithmetic crossover

4) Mutation: Individuals will mutate with a small probability. mutation is a random change of a gene value, for binary code that is 0 turns to 1 or 1 turns to 0.

5) Evaluation: A fitness function is used to evaluate an individual is a good one or not. For some optimization problems, the fitness function could be the objective function.

6) Update: According to the fitness, $N \cdot G$ new and $N$ old individuals are sorted. $N$ better individuals will be retained and form new population $P$.

With the operations listed above, the data is continually optimized. And these operations show that GA has the following advantages:

1) Search using the evaluation function to inspire the process, fast and random search capabilities that are independent of the problem.

2) With the potential parallelism, multiple populations could evolve at the same time.

3) It is extensible and easy to combine with other algorithms.

4) Using the probability mechanism to iterate.

**Algorithm 1:** Genetic algorithm

```
Begin
    $p = initialize(N, M)$;
    Repeat
        $p_{parent} = select(p)$;
        $p_{son1} = cross(p_{parent})$;
        $p_{son2} = \text{mutate}(p_{son1})$;
        $f = \text{evaluate}(p, p_{son2})$;
        $p = update(p, p_{son2}, f)$;
    until terminated=true
    output the best individual in $p$;
end
```

Due to the characteristics of genetic algorithm, it could be used to solve various optimization problems. And the storage and computing capability of cloud server can further strengthen the role of genetic algorithm. The explicit steps of genetic algorithm are depicted in Algorithm 1.

## 4 The proposed scheme

### *4.1 Overview of the scheme*

The structure of the outsourcing genetic algorithm consists of three parts: the algorithm provider, the algorithm user and the cloud servers. The general structure is shown in Fig. 2.

First of all the algorithm user informs the algorithm provider of the public key. The algorithm provider uses the public key to encrypt the parameters and make corresponding changes to the algorithm. After that, the genetic algorithm will be uploaded to the cloud calculators and wait to be used.

The algorithm user encrypts the initial data and fitness function, and sends the encrypted data and function to the cloud calculator.

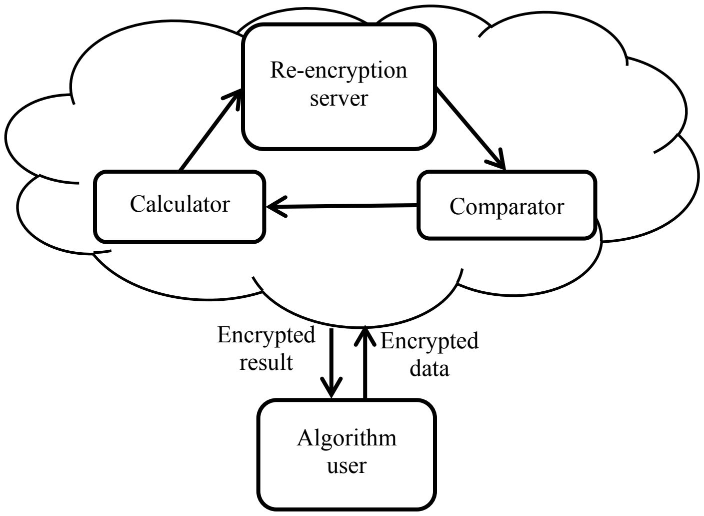


**Figure 2:** The system model

The cloud server consists of calculator, comparator and re-encryption sever. The calculator receives the encrypted algorithm from algorithm user, and it will complete the process of selection, crossover, mutation, and calculating the fitness of the individuals. After one evolution, all the fitness values are sent to the re-encryption server. The re-encryption server re-encrypts the fitness values with re-encrypt key. The comparator will decrypt the re-encrypted data with the key it keeps and returns the sort result to the calculators to make up the new populations. Through the evolution of generations, the calculator will get an optimal solution in the population and return to the algorithm user.

After getting the encrypted optimal result from the calculator, the algorithm user will decrypt it with his private key.

### 4.2 Incomplete re-encryption

Before introducing the details of privacy-preserving genetic algorithm outsourcing scheme, we will first introduce a new conception incomplete re-encryption. Incomplete re-encryption is used for solving the problem of privacy-preserving comparison between two encrypted data. We proposed it by the inspired of re-encryption.

Re-encryption is a method to share information between two users Liu et al. [Liu, Wang and Wu (2014)]. Alice encrypted her massage $m$ with her public key $pk_A$, and send this encrypted massage $c = E(m, pk_A)$ to the cloud server. The cloud server re-encrypted the massage with re-encryption key $rk = (sk_A, pk_B)$ which is consisted of Alice's private key $sk_A$ and Bob's public key $pk_B$. So Bob can decrypt the re-encrypted massage with his own private key $sk_B$. The general process of re-encryption is as follows:

$$\begin{aligned} c' &= E(c, rk) \\ &= E(D(E(m, pk_A), sk_A), pk_B) \\ &= E(m, pk_B) \end{aligned} \tag{6}$$

The plaintexts obtained by Alice and Bob in the re-encryption scheme should be same. However, in the incomplete re-encryption scheme introduced in this paper, Bob can only get the same sorting as Alice, but the decrypted results are different. So we named it as incomplete re-encryption. With the sorting, Bob could find the best solution, and the original data will not be leaked to Bob.

The details of incomplete re-encryption are introduced as follows:

1. Alice encrypts her massage m with her public key $pk$, and sends her encrypted massage $c = E(m, pk)$ to the re-encryption server.

2. Then Alice samples ring elements $x, y, z \leftarrow \chi$ and sends the re-encryption key $rk = x + tz$ to the re-encryption server and the decryption key $dk = (xs + ty, t)$ to Bob respectively. So the re-encryption server could re-encrypt the ciphertext $c$ as formula (7)

$$Rc_i = c_i \cdot (x + tz)^{\delta - i - 1} \tag{7}$$

3. The encrypted massage is $C = (c_0, c_1, \dots, c_\delta)$. The re-encryption server will send the re-encrypted massage which is $RC = (Rc_0, Rc_1, \dots, Rc_\delta)$ to Bob. Bob could decrypt it with the decryption key $dk$ as formula (8).

$$\begin{aligned} \mathrm{F}(RC, xs + ty) &= \textstyle\sum_{i=0}^{\delta} Rc_i \cdot (xs + ty)^i \\ &= \textstyle\sum_{i=0}^{\delta} c_i \cdot s^i \cdot x^{\delta-1} + te_{error} \\ &= x^{\delta-1} \cdot m_{fitness} + te_{error} \end{aligned} \tag{8}$$

4. Bob can get the by modular operation.

If Alice pass multiple massage to Bob by incomplete re-encryption, Bob can get the magnitude relationship of them without getting the private key of Alice or the raw massages.

### 4.3 Privacy-preserving operators in calculator

The cloud calculator needs to accomplish the missions of selection, crossover, mutation and evaluation. The detailed process will be introduced in this section.

Selection: Selection in genetic algorithm is random. So it is also easy to realize in encrypted genetic algorithm.

Crossover: There are many ways to cross two individuals, but they belong to only two categories: point crossover and arithmetic crossover. Point crossover such as one-point crossover shown in Fig.1(a). The breakpoint between $x$ bit and $x+1$ bit means that the first $x$ bits of the newly generated individual A' are obtained from individual A, and the rest are obtained from individual B. For multiple points crossover, the bits of A' are obtained from A and B alternately. It is easy to complete the point crossover in the ciphertext domain by using the same operation in the plaintext domain. The arithmetic crossover could not be used in binary code. A' is usually obtained by calculating A and B with finite addition and multiplication.

Mutation: The purpose of mutation is to maintain population diversity and prevent population from falling into the local optimal solution. It is a suitable operation of genetic mutation that modify the ciphertext randomly with small probability.

Evaluation: Fitness is the only evaluation standard to evaluate the solution. It is obtained by calculating the object function or fitness function. Due to the limitation of the homomorphic encryption algorithm, the function can only support simple mathematical operations at present.

### *4.4 Privacy-preserving operators in re-encryption server*

The re-encryption server re-encrypts the fitness values into ciphertexts that the comparator can decrypt. The decrypted results are not the same as the fitness values, but the magnitude relationship is not changed.

Re-encryption server will send the re-encrypted data which is $RC = (Rc_0, Rc_1, \dots, Rc_\delta)$ to the comparator.

The comparator will decrypt the re-encrypted fitness values with decryption key, and it will get the $x^{\delta-1} \cdot m_{fitness}$ by modular operation. For all the solutions, the fitness values turned the same times, so the magnitude relationship is unchanged. The comparator will return the sorting to the calculator. The calculator will update the population, and evolution will go on.

### *4.5 The complete process description*

Based on the introduction, the complete process will be described in Alg. 2. The Enc means encrypting the parameters and modifying the normal additional or multiplicative operations to the operations which can fit the encryption algorithm.

## 5 Security analysis

The cloud servers are honest-but-curious. If collusion problem between servers is not considered, the security of the data depends on the security of the encryption algorithm. However, the security of $SH.E$ is analyzed in Hu et al. [Hu, Wang, Wang et al. (2016)]. We only analyze the four situations in which servers collaborate with each other.

**Algorithm 2:** Encrypted genetic algorithm

```
At user:
1) (pk,sk)=KeyGenerate;
2) Send the public key pk to the provider, the re-encryption key rk to the re-encryption
   server and the decryption key dk to the cloud comparator re- spectively;
3) $E.Data_1 = \mathrm{SH.Enc}(original\ data, pk)$;
4) $E.Fitnessfunction = \mathrm{Enc}(fit, pk)$;
5) Upload $E.Data_1$ and $E.Fitnessfunction$ to the cloud calculators;
At provider:
1) $E.GA = \mathrm{Enc}(GA, pk)$;
2) Upload E.GA to the cloud calculators;
At cloud server:
1) Repeat
   Calculator:
           $[E.MidData_t, E.Fitness] = \mathrm{E.GA}(E.Data_t, E.Fitnessfunction)$;
     Send the E.Fitness to re-encryption server;
   Re-encryption server:
     Re-encryped the $E.Fitness$ with $rk$ to get $RE.Fitness$
     Send the E.Fitness to re-encryption server;
   Comparator:
     $Fitness = \mathrm{SH.Dec}(E.Fitness)$;
     $[Sorting] = \mathrm{sort}(Fitness)$;
     Send the Sorting to calculator;
   Calculator:
     $E.Data_{t+1} = \mathrm{Update}(E.MidData_t, Sorting)$;
     $t = t + 1$
   until terminated=true
   end
2) Output the final results $E.Data_t$ and E.Fitness to user;
At user:
1) $[Data, Fitness] = SH.Dec((E.Data_t, E.Fitness), sk)$;
Get the best solution by sorting the Fitness.
```

**Situation A:** Calculator colludes with re-encryption server

The re-encryption key $x$ is a random number that independent of any information which can be obtained by the calculator. So there is no help that calculator colludes with re-encryption server.

**Situation B:** Comparator colludes with re-encryption server

The comparator owns the decryption key $dk = (xs + ty, t)$, and the re-encryption server owns $rk = x + tz$. There are four variables in two equations. In general, there are innumerable solutions to this problem. So if comparator colludes with re-encryption server, the private key *s* and *x* still safe.

**Situation C:** Calculator colludes with comparator

The calculator could send the encrypted original data without re-encryption to the comparator. The comparator decrypts the data with the decryption key *dk*. The decryption result as follows:

$$\begin{aligned}\widetilde{m} &= F(ct, xs + ty)\\ &= \textstyle\sum_{i=0}^{\delta} c_i (xs)^i + te_{error}\\ &= c_0 + c_1 xs + c_2 x^2 s^2 + \cdots + c_\delta x^\delta s^\delta + te_{error} + m \end{aligned} \tag{9}$$

Obviously, $\mathrm{m} \neq \widetilde{m}\ mod\ t$, it is hard to recover *m* from $\widetilde{m}$.

**Situation D:** All the parties of the cloud servers colludes with each other

As described in situation B, *s* and *x* are unavailable. All the parties of the cloud servers collude with each other. So the comparator could get the re-encrypted original data, and try to decrypt it. The decryption result as follows:

$$\begin{aligned}\widetilde{m} &= F(RC, xs + ty)\\ &= \textstyle\sum_{i=0}^{\delta} Rc_i (xs)^i + te_{error}\\ &= \textstyle\sum_{i=0}^{\delta} c_i s^i x^\delta + te_{error}\\ &= te_{error} + x^\delta m \end{aligned} \tag{10}$$

The cloud servers could only get $x^\delta m$ by $\widetilde{m}\ mod\ t$.

## 6 Experiment results

The purpose of the privacy-preserving genetic algorithm outsourcing proposed is to reduce the cost of the algorithm users. At the same time, it hopes the cloud server could return a correct global optimal solution which is not worse than the optimization result in the plaintext domain. Two indicators are summarized to verify the performance of the algorithm. The two indicators are the difference of the optimal values and speed increase rate between using privacy-preserving genetic algorithm and standard genetic algorithm.

However, the types of arithmetic operations that can be supported by the current homomorphic encryption algorithm are limited, and only integers can be correctly encrypted and decrypted. Although we can multiply the float-point numbers by $10^k$ to amplify them, many decimals have to be abandoned which leads to the result being an approximation.

The communication latency between the user and the cloud servers is ignored since the other operations in this scheme cost much more time, the latency could take few effect to the scheme. De Jong function is chosen to evaluate the performance of the scheme. And the experiment results are shown in Tab. 1.

**Table 1:** Performance Results. Here difference value means the difference between the optimal value of original GA and encrypted GA. The $t_{original}$, $t_{cloud}$, and $t_{user}$ denotes the user-side original problem solving time, cloud-side encrypted problem solving time, and user-side computation time, respectively. The asymmetric speedup captures the user efficiency gain via outsourcing

| Dimension | Difference Value | Original Problem | Encrypted Problem | | Asymmetric Speedup |
|---|---|---|---|---|---|
| | | $t_{original}$(sec) | $t_{cloud}$(sec) | $t_{user}$(sec) | $t_{original}$ /$t_{user}$ |
| 50 | 0 | 67.5607 | 318.3374 | 1.0072 | 67.0777 |
| 100 | 0 | 138.0556 | 734.0666 | 2.0081 | 68.7494 |
| 150 | 0 | 221.7535 | 1268.0280 | 3.0934 | 71.6860 |
| 200 | 0 | 381.1831 | 2278.9867 | 4.0301 | 94.5840 |

As shown in the Tab. 1, encrypted GA could get the same solution as the original GA. The operation in encrypted domain does not have a negative effect on the optimization results. The purpose of algorithm user outsources the calculation process to the cloud server is to reduce the computational cost himself. In spite of $t_{cloud}$ is much larger than $t_{original}$, the time spent by the user is greatly reduced. Because the genetic algorithm needs to process the data repeatedly, the greater the problem dimension is, the longer the calculation time of once evolutional generation costs. And the greater the dimension is, the more the evolutional generation is needed, which leads to much more cost. With the rise of the problem dimension, the value of asymmetric speedup is also rising. That is because the user only needs to encrypt the original data and decrypt the final results, which is less affected by the growth of the dimension than the genetic algorithm.

It is expected that outsourcing encrypted genetic algorithm to the cloud servers can bring more benefits for the user with the rise of the dimension. More experiments need to be done to verify this view, and it is what the author is doing.

## 7 Conclusion

With the help of cloud servers, user's workload is greatly reduced. And genetic algorithm can also be out- sourced to the cloud server. Although the arithmetic operations that can be supported by the recent encryption algorithm, it is hopeful to outsource more optimization to the cloud servers with the deepening of research on encryption algorithms.

**Acknowledgement:** This work is supported by the NSFC (61672294, 61601236, U1536206, 61502242, 61572258, U1405254, 61373133, 61373132, 61232016), BK20150925, Six peak talent project of Jiangsu Province (R2016L13), NRF-2016R1D1A1B03933294, CICAEET, and PAPD fund.

## References

**Balamurugan, S.;Visalakshi, P.; Prabhakaran, V. M.; Sankaranarayanan, S.** (2014): Strategies for solving the NP-Hard workflow scheduling problems in cloud computing environments. *Australian Journal of Basic & Applied Sciences*, vol. 16, no. 8, pp. 345-355.

**Berlanga, F. J.; Rivera, A. J.; Jesus, M. J.; Herrera, F.** (2010): GP-COACH: Genetic programming- based learning of compact and accurate fuzzy rule-based classification systems for High-dimensional problems. *Information Sciences*, vol. 8, no. 180, pp. 1183-1200.

**Brakerski, Z.; Vaikuntanathan, V.** (2011): Fully homomorphic encryption from ring-LWE and security for key dependent messages. *Advances in Cryptology-CRYPTO 2011*, pp. 505-524.

**Cao, N.; Wang, C.; Li, M.; Ren, K.; Lou, W. J.** (2011): Privacy-preserving multi-keyword ranked search over encrypted cloud data. *INFOCOM, 2011 Proceedings IEEE*. pp. 829-837.

**Coyle, D.; Nguyen, D.** (2019): Cloud computing, Cross-Border data flows and new challenges for measurement in economics. *National Institute Economic Review*, vol. 1, no. 249, pp. R30-R38.

**Eldurssi, A. M.; O'Connell R. M.** (2015): A fast nondominated sorting guided genetic algorithm for multi-objective power distribution system reconfiguration problem. *IEEE Transactions on Power Systems*, vol. 2, no. 30, pp. 593-601.

**Foster, J. D.; Berry, A. M.; Boland, N.; Waterer, H.** (2014): Comparison of mixed-integer programming and genetic algorithm methods for distributed generation planning. *IEEE Transactions on Power Systems*, vol. 2, no. 29, pp. 833-843.

**Fu, Y.; Lai, K. K.; Liang, L.** (2014): A robust optimisation approach to the problem of supplier selection and allocation in outsourcing. *International Journal of Systems Science*, vol. 4, no. 47, pp. 1-6.

**Fu, Z. J.; Ren, K.; Shu, J. G.; Sun, X. M.; Huang, F. X.** (2016a): Enabling personalized search over encrypted outsourced data with efficiency improvement. *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 27, pp. 2546-2559.

**Fu, Z. J.; Wu, X. L.; Guan, C. W.; Sun, X. M.; Ren, K.** (2016b): Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. *IEEE Transactions on Information Forensics and Security,* vol. 12, no. 11, pp. 2706-2716.

**Gennaro, R.; Gentry, C.; Parno, B.** (2010): Non-interactive verifiable computing: outsourcing computation to untrusted workers. *Lecture Notes in Computer Science*, vol. 3, no. 6223, pp. 465-482.

**Gentry, C.** (2009): Fully homomorphic encryption using ideal lattices. *ACM Symposium on Theory of Computing, STOC 2009*, pp. 169-178.

**Yu, G.; Wang, L.** (2018): Advantages and paths of developing cloud computing industry in heilongjiang province. IETI Transactions on Social Sciences and Humanities, pp. 37-42.

**Habibi, F., Motameni, H., Ramezani, F.** (2008): The new approach for solving task scheduling in cloud computing environment using combination of genetic algorithm and tabu search. *Networks*, pp. 60-67.

**Hu, S. S.; Wang, Q., Wang, J. J., Qin, Z., Ren, K.** (2016): Securing SIFT: privacy-preserving outsourcing computation of feature extractions over encrypted image data. *IEEE Transactions on Image Processing,* vol. 7, no. 25, pp. 3411-3425.

**Huang, S. C.; Jiau, M. K.; Lin, C. H.** (2015): A genetic-algorithm-based approach to solve Carpool service problems in cloud computing. *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 16, pp. 352-364.

**Jiang, L.; Xu, C.; Wang, X.; Luo, B.; Wang, H.** (2015): Secure outsourcing SIFT: efficient and privacy-preserving image feature extraction in the encrypted domain. *IEEE Transactions on Dependable & Secure Computing*, vol. 8, no. 14, pp. 1-15.

**Jiau, M. K.; Huang, S. C.** (2015): Services-oriented computing using the compact genetic algorithm for solving the carpool services problem. *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 16, pp. 2711-2722.

**Li, J.; Sun, Q.; Zhou, M. C.; Yu, X.; Dai, X.** (2014): Colored traveling salesman problems and solutions. *IEEE Transactions on Cybernetics*, vol. 3, no. 47, pp. 9575-9580.

**Li, Q.; Zhang, T.; Yu, Y.** (2011): Using cloud computing to process intensive floating car data for urban traffic surveillance. *International Journal of Geographical Information Science*, vol. 8, no. 25, pp. 1303-1322.

**Liu, Q.; Wang, G. J.; Wu, J.** (2014): Time-based proxy re-encryption scheme for secure data sharing in a cloud environment. *Information Sciences*, vol. 3, no. 258, pp. 355-370.

**Xia, Z. H., Wang, X. H., Sun, X. M., Wang, Q.** (2015): A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems,* vol. 2, no. 27, pp. 340-352.

**Xue, Y.; Jiang, J. M.; Zhao, B. P., Ma, T. H.** (2017): A self-adaptive articial bee colony algorithm based on global best for global optimization. *Soft Computing*, pp. 1-18.

**Yao, A. C.** (1982): Protocols for secure computation. *Foundations of Computer Science Annual Symposium on*, pp.160-164.

**Zhan, Z. H.; Liu, X. F.; Gong, Y. J.; Zhang, J.** (2015): Cloud computing resource scheduling and a survey of its evolutionary approaches. *Acm Computing Surveys*, vol. 4, no. 47, pp. 63.