Parallelization of Cryptographic Algorithm & Key Identification Using Genetic Algorithm Approach

Ghanshyam Gagged, Krishnakant Pandey, Shubham Asati, Jaisankar N
School of computer science & Engineering, VIT University, Vellore-632014, Tamilnadu, India

Abstract

Now days, the amount of transfer of data over large network is increasing day by day. As there is increase in data transfer, simultaneously there are security threats that are arising with it. In this paper we are proposing cryptographic systems that make use of genetic algorithm to securely transfer the data over large network. This approach helps us to provide high achievability to transfer data securely. The techniques that we are using for encryption & decryption are based on genetic algorithm which will help us to efficiently encrypt the data which is resistible for any kind of external attacks. Then we will parallelize it using Open Mp language which accelerates the transformation of data to achieve high security. Then analyze the performance for both code serial and parallel execution.

Keywords: The genetic algorithm, Linear generation Equation, Genetic operator, Open MP

I. Introduction

We know that, Cryptography is the study of encrypting the information & producing incomprehensible data which is unreadable by the third person who is accessing data without prior permission of sender or receiver. Now a day various application like the social networking, business applications, E-commerce, in military or satellite communication etc need lot of data transmission over large data network. Here is the main concern for privacy comes into picture. As the above said areas are very much vulnerable to attacks many researchers are working on it. We already have many algorithms for encrypting & decrypting the data. Encryption is a process of transferring simple text into cipher text and decryption is a process transferring cipher text into simple text which is readable and generate a key which is a shared between processes of cryptography [9]. Encryption and decryption are opposite to each other for transforming data. The genetic algorithm that we are using takes combination of features of cryptography and genetic operator. The genetic operators that we are using are crossover & mutation. In crossover we combine two strings to get the new better string. The mutation is the process of changing any bit from the given set of bits. So here we can use the concept of Open Mp to parallelize the code. The generation of random number can also be parallelized so we will always get different randomized number from different-different threads. Here we are using the concept of symmetric key.

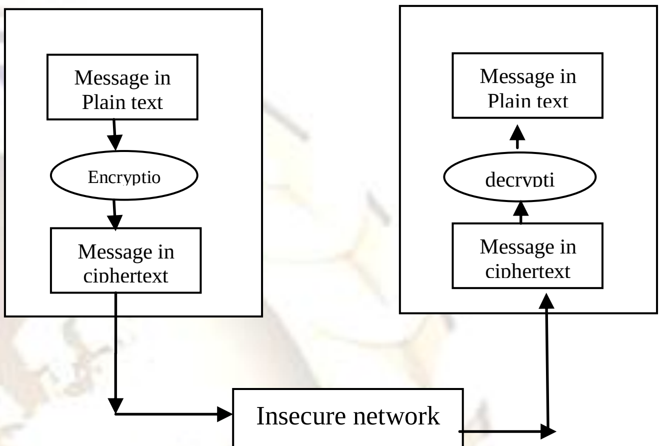

Fig.1 Symmetric key encryption & decryption

In the cryptography there are two types by which we can encrypt and decrypt the text i.e. symmetric and asymmetric key. The cryptographic system in which sender & receiver use the same key for encryption & decryption data is called symmetric key cryptography [1]. The Cryptographic system in which sender and receiver use the different key for encryption & decryption data is called asymmetric key cryptography. As we know the data to be transfer is very large we need very efficient & fast algorithm to encrypt or decrypt it [9]. The parallelization of genetic algorithm yields better performance that will be very useful for transferring large data securely using this parallelized code. Now we will discuss all the issues in detail.

II. Linear generator Equation

Firstly here we a creation a sequence of random number by using Congruential method then next generation number random number with the help of Crossover and Mutation which is a operator for genetic algorithm.

$$\mathbf{X_{n+1} = (a * X_n + C) \bmod m;} \qquad ......(1)$$

Xn= a randomly generated value initially it is assume.
a= multiplier which randomly generated
c= increment which randomly generated
m= modulus (0<m<10)

III. Generic Operators:

From the three operators of genetic algorithm i.e. CROSSOVER, SELECTION and MUTATION [6] here we will be using only two operators i.e. Crossover and Selection for our purpose.

A. Crossover

It is a genetic operator which is used to join two strings to get an efficient string. In this process we take more than one parent process and generate a child solution for that process [2] this crossover genetic operator generate random number which perform in parallel.

There a three types of crossover [6]:

- Trade of uniform crossover
- one site Crossover
- Cut and splice

In this paper we will be concentrating on one site Crossover. This method is selected because of its advantage over traditional methods that we can easily exchange information through it very effectively as per our requirements.

We know that the data while transferring is converted into bits and then sent over the network so our next method i.e. MUTATION changes any bit randomly from the set of bit to achieve high security. As the bits are changed its very difficult for intruder to withdraw the information from it. As we are going to parallelise the code the different thread will play very vital role in changing the bits.

IV. Cryptography: Genetic Algorithm Approach

3.1. Algorithm

3.1.1 Generation Linear Equation of First order using Random Number [7]

In this we use Linear Equation for generating first order variables [5]. Initially we will consider four random variable a, c, m. $X_{n.}$ Now as per our formula of generating linear equation we will substitute this assumed 4 values to calculate next value. After that iteratively we will put the value of X_{n+1} that is generated from the previous calculation. This iterative process will terminate only when it will complete the iterations equal to the length of the message. Here we can see that the initial parameters are unchanged throughout the process. A good randomization function must be used to get better initial values [6].

3.1.2 Crossovers and Mutation:

After the above defined process we have some random numbers generated that are equal to the length of message. Now this numbers are converted into binary formats and length of each binary number is equated by padding some zeros at the beginning of that number[4]. Now here we will be using the crossover operation of genetic algorithm in which we will swap some bits of one string with the other string. At this stage two numbers are differently changed from their initial values to the new values. Let's illustrate it with an example.

Consider two numbers of 8 bit,

187 $10\underline{\mathbf{1111}}00 \Longrightarrow 10\underline{\mathbf{0001}}00$

132 $10\underline{\mathbf{0001}}00 \Longrightarrow 10\underline{\mathbf{1111}}00$

As of now we have just performed crossover now we will deal with mutation. Mutation is the process of changing the bit from the available set [3]. Here is the main function of mutation operator to change the bit any the selection of bit which is to be changed is done probabilistically so security increases. Finally after performing Crossover & mutation for several iterations we will get series of randomized number that we will again convert into decimal format for our further use.

Now the numbers which we will get finally will be in decimal format, so we will select the smallest number from it & the specific digit that is mutually decided between two parties will be subtracted from the ASCII value of our text message and the number which we will get after that is sent over the network along with the key.

3.2. For Example

3.2.1 Encryption

In this method the simple text converted into cipher text, using above algorithm

- Let, message is ABCDEFGHIJ its length is 10.
- Now applying linear Congruential equation, we will get the generated numbers that are similar in length of text message i.e.10.

Let, numbers are: 106, 239, 538, 1211, 2725, 6132, 13798, 31046, 69854 and 157172. Now, Select two numbers from a group that starting (106,239), (538, 1211), (2725, 6132), (13798, 31046), (69854, 157172).

Then apply crossover and mutation iteratively:

106= 0000000001101010

239=0000000011101111

After performing crossover

0000000101101010 =362

0000000111101111=495

Now we identify that some binary digits have been changed. Performing mutation

0000000110001010 =394

0000000100001111= 271

After applying mutation, numbers are (394, 271).

Now we easily identified how group pair is changed from (106,239) to (394,271). However we find that both crossover and mutation increases security and after performing some iteration the whole data become secure.

Now after first iteration of crossover and mutation number will be obtained

(394, 271, 1018, 1371, 2885, 5652, 13318, 30886, 35215, 39069)

Now took a smallest number i.e. 271 and subtract second right digit from ASCII values of each character of message.

Input Data	Ascii Values	Generated number	Number to be subtracted	Cipher text
A	65	394	9	56
B	66	271	7	59
C	67	1018	1	66
D	68	1371	7	61
E	69	2885	8	61
F	70	5652	5	65
G	71	13318	1	70
H	72	30886	8	64
I	73	35215	1	72
J	74	39069	6	68

Table 1 Encryption procedure

The encrypted message is > @CA=@D@GJ

The Cipher text :{ 56,59,66,61,61,65,70,64,72,68 {key of {a, c, m, X_n }}

3.2.2 Decryption

At a user side we get a whole encrypted data along with key {a, c, m, X_n}. As we know decryption is an opposite process of encryption for that a receiver side inverse of encryption will be perform [8]. After that many iteration perform in decryption as they performed in encryption using cipher key.

Cipher text	Number to be added	ASCII Values	Input Data
56	9	65	A
59	7	66	B
66	1	67	C
61	7	68	D
61	8	69	E
65	5	70	F
70	1	71	G
64	8	72	H
72	1	73	I
68	6	74	J

Table 2 Decryption procedure

V. RESULTS

As our main aim is to parallelize the program to get fast output according to that we tested it for various number of file size. Here we present the table representing the throughput and the graphical representation with core different core processor.

A) For 2 core machine:

File Size	Serial Execution Time	Parallel Execution Time	Speed Up
1MB	0.33ms	0.19ms	1.67
10MB	1.5ms	0.82ms	1.81
50 MB	11.2ms	5.6ms	2
100 MB	21.4ms	9.5ms	2.25
150MB	37.3ms	13.4ms	2.78

B) For 4 core machine:

File Size	Serial Execution Time	Parallel Execution Time	Speed Up
1MB	0.33ms	0.11ms	3
10MB	1.5ms	0.48ms	3.12
50 MB	11.2ms	2.71ms	4.12
100 MB	21.4ms	3.4ms	6.3
150MB	37.3ms	4.72ms	7.89

C) For 8 core machine:

File Size	Serial Execution Time	Parallel Execution Time	Speed Up
1MB	0.33ms	0.07ms	4.67
10MB	1.5ms	0.28ms	5.2
50 MB	11.2ms	1.49ms	7.51
100 MB	21.4ms	2.4ms	8.9
150MB	37.3ms	3.49ms	10.67

D) Performance Analysis:

File Size	2 Core	4 Core	8 Core
1MB	1.67	3	5.67
10MB	1.81	3.12	6.2
50 MB	2	4.12	8.51
100 MB	2.25	6.3	9.9
150MB	2.78	7.89	11.67

Table 3 Performance analysis

We can see that parallelized program with different core processor which gives better output in terms of time taken for execute.

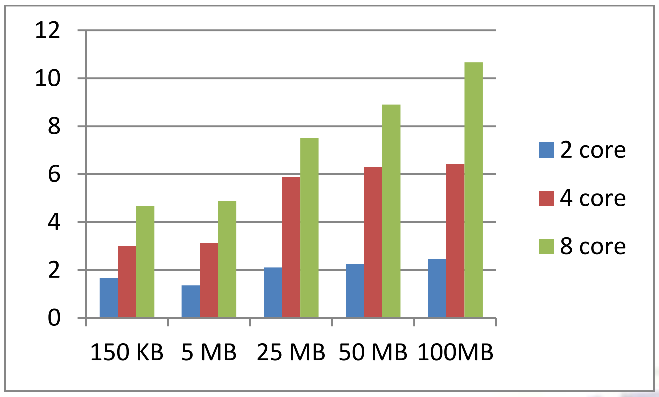

Fig 2 Graph showing the perfomance analysis

As we can see from the performance of our algorithm on different-different cores we can predict the future speed up for given large size of file.

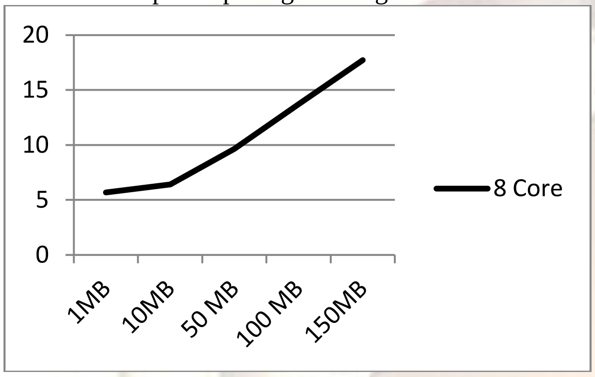

Fig 3 speed up Curve for known file size

By looking in this graph, we say that the speed up factor is seems to be increasing as file size is increase in a constant growth rate. Now we are using logarithms for transfroming this equation into linear equation.
The linear equation is as follows,
Y = m + cX ....(2)
Where,
c = constant
m = speed up factor

Projected values

File Size	8 Core	Projected Values
1MB	5.67	5.67
10MB	6.4	6.4
50 MB	8.51	9.64
100 MB	9.9	13.69
150MB	11.67	17.7
200MB	-	21.81
300MB	-	29.92
400MB	-	38.01
500MB	-	46.13

Table 4 projected values

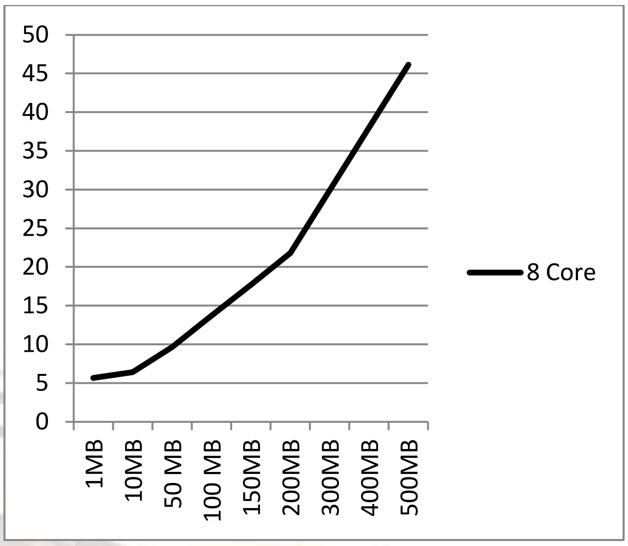

Fig 4 Speed up curve for projected values

Proof:

We can prove it by using mathematical formula. Lets consider the speed up on 8 core machine for a given size file then it will follow the equation (2) i.e.
Y = c + mX
Solution: We will use mathematical induction for proving the above statement.
For 8 core machine,
Lets take initial values of X & Y
Y = 5.67 for file size of X = 1MB
Y= 6.4 for file size of X = 10MB
Putting this value in our equation,
We will get,
c = 5.58, m = 0.0811.
so now extending it to size of 100MB, 150 MB it comes true.
For all values it will come true so we can say that this equation holds for all given size of file.

VI. Conclusion

We can see that the genetic algorithm operator that we have used gives better performance as compared to other algorithm. If we write the same alphabet in our message like ZZZZZZ still it will generate different key and the value for each alphabet. The number of times we iteratively repeat the process of crossover & mutation we would be keep getting more secure encryption. Even successful parallelization yields better results in terms of execution time. As we have not yet used all the operations of genetic algorithm in future we can used it get better security.

References

[1] Benjamin Arazi, "Vehicular Implementations of Public Key Cryptographic Techniques", IEEE TRANSACTIONS ON VEHICULAR

TECHNOLOGY, VOL. **40,** NO. 3,pp 646-653, AUGUST 1991.

[2] Murat Kantarcioglu, Wei Jiang, Ying Liu, and Bradley Malin, "A Cryptographic Approach to Securely Share and Query Genomic Sequences", IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE, VOL. 12, NO. 5,pp 606-617, SEPTEMBER 2008

[3] Jong-Bae Park, Young-Moon Park, Jong-Ryul Won, and Kwang Y. Lee, "An Improved Genetic Algorithm for Generation Expansion Planning", IEEE TRANSACTIONS ON POWER SYSTEMS, VOL. 15, NO. 3, pp 916-922,AUGUST 2000.

[4] K. F. Man, K. **S.** Tang, and **S.** Kwong, "Genetic Algorithms: Concepts and Applications", IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 43, NO. 5,pp 519-534, OCTOBER 1996.

[5] R. H. Torres, G. A. Oliveira, J. A. M. Xexéo, W. A. R. Souza and R. Linden, "Identification of Keys and Cryptographic Algorithms Using Genetic Algorithm and Graph Theory", IEEE LATIN AMERICA TRANSACTIONS, VOL. 9, NO. 2, APRIL 2011.

[6] Thang Nguyen Bui and Byung Ro Moon, "Genetic Algorithm and Graph Partitioning", IEEE TRANSACTIONS ON COMPUTERS, VOL. 45, NO. 7,pp 841-855 JULY 1996.

[7] Yao-xue zhang, Kaoru Takahashi, Nor10 Shiratori, and Shoichi Noguchi, "An Interactive Protocol Synthesis Algorithm Using a Global State Transition Graph", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 14. NO. 3,pp 394-404, MARCH 1988.

[8] Franciszek Seredynski and Albert Y. Zomaya, "Sequential and Parallel Cellular Automata-Based Scheduling Algorithms", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 13, NO. 10, pp 1009-1023,OCTOBER 2002.

[9] Yi-Ta Wu, and Frank Y. Shih, "Genetic Algorithm Based Methodology for Breaking the Steganalytic Systems", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 36, NO. 1, pp 24-31,FEBRUARY 2006.