International Journal of Pure and Applied Mathematics
Volume 118 No. 20 2018, 3245-3255
ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
Special Issue

ijpam.eu

PLUTO: Cloud Assist Parallel Encryption for Data in Mobile Cloud Computing

V.Meena, S.Akshhaya Lakshmi, S.R.Aiswarriya

School of Computing, SASTRA Deemed to be University, India

Abstract:

Nowadays Mobile devices rule the entire world, if its batteries go down then whole device become useless. Basically mobile devices are resource constraint devices. If any heavy weight that is computation intensive applications are getting executed, the battery power will be drained very fast. Volume of the data is allowed to store very minimal in order to maintain efficiency. In some situations there will be chance to execute computation intensive, communication intensive and data intensive application. In turn it creates battery drainage problem and then this leads to reduce the lifetime of the battery. In order to overcome the limitations in mobile devices and pave the path to development of a new algorithm called "PLUTO" which reduces the battery drainage in the mobile phones by dividing the large files into small chunks and process them parallelly. Therefore, it increases the efficiency of resource constraint devices. For sharing the secret data among users, there will be need to have secured transmission. In this framework, before transmission the data will be converted to encrypted form. So that, these Mobile devices uses light weight encryption algorithm and a asymmetric key encryption to enhance more security to the data in mobile devices. For key generation they utilize the resourceful remote server by using pseudo random number generator algorithm. This framework implemented the encryption process in nearby resourceful clusters inorder to save the battery power. Finally this framework provides secured transmission of mobile data by saving the energy in resource constraint devices.

Keywords: Cloud computing, Parallel Processing, light weight, cluster, encoding and Decoding.

I. INTRODUCTION

As Mobile devices become the greatest demand, it overcomes the physical machines due to its compactness and handy. These devices store many confidential details like password, pin, and huge information. But the problem arises when we consider about security, battery drainage and memory. These are hurdle for the Mobile devices to attain its full utilization in the cloud infrastructure.

In order to share confidential information between two users it has to be encrypted with strong key to prevent from brute force attack, eavesdropping, hacker, third party member. PLUTO uses a XOR encryption method which takes place in the mobile devices. In this encryption, the key is generated in the cloud infrastructure by pseudo random number generator .Since it is difficult to hack sequence of many numbers and these keys are encrypted with RSA algorithm, a public key encryption is better than the private key i.e. no need to share the key. In this algorithm only public key is visible, the private key is generated based on function and only the receiver who having that key can be decrypted. By this protocol if any intruder or hacker tries to hack the information means that he cannot decrypt the text without this private key.

This encryption in mobile devices may consume lot of power in devices. For a better performance in the mobile devices, encrypting the text parallelly by splitting the file into small chunks based on user defined size .The major challenge of this protocol is that it can deal with even huge files. No limitation in size eventually it split and executes them simultaneously. By encrypting and decrypting them parallelly reduces the fast battery drainage.

At the same time, there is a problem in storing large huge information in mobile devices. This can be overcome by using the cloud server where some files can be offloading into the cloud which exceeds in the mobile phone; it can be retrieve from other user while decrypting it.

The next challenge in this protocol is incase if any or both mobile devices fails, there is a great chance to retrieve the data. Since the cloud server stores the encrypted data there is no chance for data loss.

II. Related Work

Security level between a channeling of information based not only on how strong the encryption algorithm but also on how securely transferring key. Sometimes generating one key for encode may not be sufficient for more confidential data. Hence, [1][2] proposed a system that uses a technique called fine grained. This method uses a bunch of key value pairs that each pair associates with the cipher text along with the private key. In our protocol, with the help of pseudo random number generator gives sequence of random numbers based on some value. There is huge number of loop holes in provide security. To provide a strong security [3][4] suggesting a new approach. In which private encrypted data having a access control that only authenticate person can access the data by providing limited amount of data to thwart from leakage. This can be provided by CIA triads in cloud architecture, it forms a relationship among the attributes to increase Confidentiality, Integrity and Availability. These triads will protect the data against the unauthorized usage [5]. Should provide security not only in the cloud architecture but also during transmission for that [6]provides sequence of random numbers which is uniformly distributed generated by one seed value, used as a key for the encryption algorithm .In this protocol random numbers used as a key for XOR encryption. As XOR is a light weight algorithm convenient to mobile phone. After encrypting data with this algorithm the size of the data reduced is one of the benefits of using this encryption method Random number is a key to this algorithm generated in cloud server. These random numbers are encrypted with RSA algorithm [7] approach a asymmetric keys by prime factorization i.e. encrypted with public key can be decrypt only by the private key.

Providing security for the wired devices is different for providing for the wireless devices [9] recommended a unique way of authentication with the help of using digital certificates. Wireless devices have more threats than the wired devices. Mobile cloud computing become the emerging technique because there is no limitation in storage, platform and infrastructure. In providing storage to the client devices it encrypt all the data and store it in the cloud server [8] includes a technique called offloading. Offloading helps in mobile devices as it cannot store huge amount of data. So some important data can be offload into the server and can be retrieved with proper authenticate user.[10] suggests a procedure to decrease the battery drainage by offloading some functions to the cloud server there it will do the operations. Hence it reduces the consumption of battery and increases the utilization. [11] Introduces a way of secured transmission using a light weight encryption algorithm using a cloud server like AWS. PLUTO uses asymmetric encryption instead of symmetric key encryption because there is a need to transfer the key to other user. [12] Gives a overall analysis of public key encryption over security level by estimating its efficiency results in public key encryption way better than private key. [13][14] Explains as the need of rapid computation sequential execution cannot deliver leads to a marked place for parallel execution. By parallel execution decreases the execution time thus increases the performance.

III. Implementation

Implementation part of PLUTO deals with mainly four modules. This section tells how PLUTO deals with all the security issues and overcomes all conflicts.

A. Framework for PLUTO

PLUTO mainly deals with three participants. They are Cloud Server (CS) and two client devices (CD). Here, CS can be any cloud server like AWS (Amazon Web Server), Windows Azure, Google Computer Engine then CD can be any wireless devices like Mobile Phones, Laptop and Tablet. The purpose of PLUTO is to enhance the security between communications and increase the response time by improving the overall utilization.

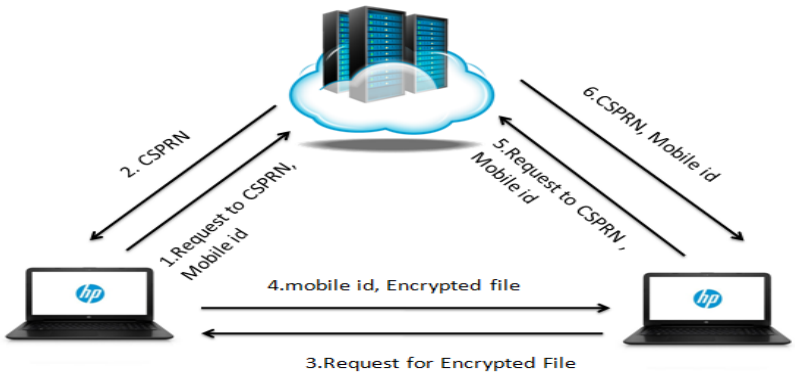


Figure.1 Framework of PLUTO.

Figure.1 describes the basic functionalities of PLUTO. Initially, CD's have to register with CS. After registering the key will be generated in the server using pseudo random number generator and for a secure transmission of key, the key will be stored in the server which is encrypted with RSA algorithm [7]. The following steps will happen during registration process.

a) Registering with Cloud Server: Initially clients have to register with the server for authentication purpose.

b) CSPRN generation: Encryption key is a random number which is generated by Pseudo Random Number Generator in the server.

c) RSA Encryption:Encrypting the key with RSA algorithm *to tranfer key from server to client.Decryption takes place in the mobile devices.*

d) Requesting CSPRN: Clients requests the random number from the server.the server check whether the request is frim the authenticate user or not. After authentication it provide the key to the clients.

e) XOR Encryting: After decrypting key ,the plain text will XORed with key which produse cipher text.The main advantage of using XOR is that after encrypting the file sizeis reduced from the original size.

f) Splitting Data: Since, it is hard to encrypt the whole data at a time in mobile devices because of their ram speed and memory size it will lead to exauction of battery in less time.To avoid this situation, the whole data is splitted into small chunks depends on user defined size and encrypting them simultaneously.By this process we can reduce the overhead inmobile devices.

g) Offloading in Server: After parallel encrytion,due to storage limitation in the mobile devices some important files will be moved to the server and data are securely stored there.

h) Merging Data: For decryption the data first we have to merge the data .The other client request the server by sending its id for the CSPRN and the encrypted data.Then the merge take place in the device.

i) Decrypting Data: After merging the cipher text will be decryptd with the key and the plain text is retrived.

B. Cryptographically Secured Pseudo Random Number(CSPRN):

In all the secured algorithm random numbers ensure that no one can hack that algorithm. A Pseudo Random Number Generator (PRNG) is an algorithm which is used to generate a sequence of Random Numbers based on a Seed value. Numbers generated are not actually Random, because these sequence of numbers are generated based on the seed value generated. These PRNG are suitable for cryptographic applications .Such a PRNG is called cryptographically secure Pseudo Random Number Generator (CSPRN).They are secured as these number are purely random and difficult to predict. This PRNG is widely used because it generate the same set of values based on some seed value given in the algorithm.

In Cryptography it provides Security by offering confidentiality, authentication. For cryptographic purposes rather than using a random number as key, we can use pseudo random number generator .Because its results are very hard to guess. Such CSPRN generator have lot of cryptographic applications. To define a CSPRN there are several ways. In one way we can say, given a bound value the CSPRN generator uses it as seed and generate certain Sequence of number .This Sequence is very difficult to predict even by a algorithm which has polynomial time complexity.

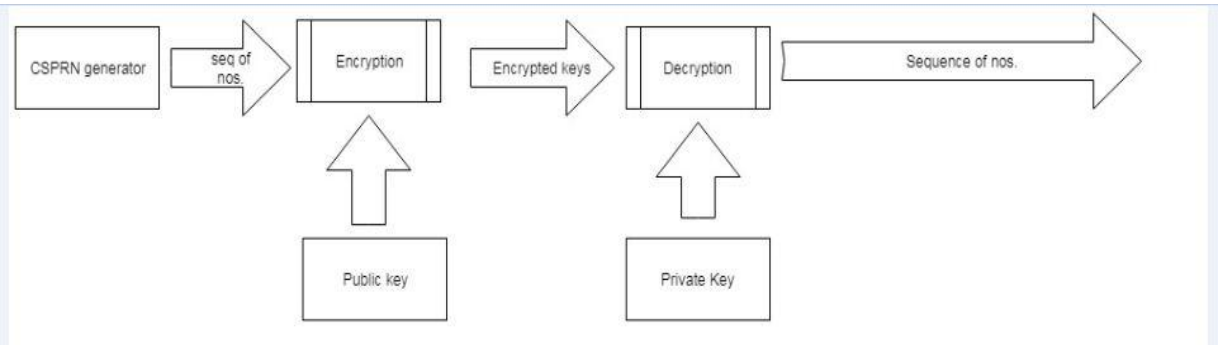


Figure 2: Key Generation in cloud service provider

There are three ways to generate random numbers in JAVA. We should import certain packages to make use of it.
Math.Random ()
util.Random.nextInt ()
util.Random.ints() [Java 8]
These methods will give randomly generated Integer numbers. For using this we should import "java.util.*"
and "java.Math.*" packages in out source code. Specifically "java.util.Random" class is used in most of the applications.Here CSPRN is generated using these functions in Java 1.8 Version. The algorithm for CSPRN generation is as follows;

```
Algorithm CSPRN ()
{
Random r =new Random ();
Loop:  Bound = input the bound;
  Seq = r.negeneratedby the xtInt(bound)//generate random numbers
End of loop
}
```

The above algorithm produces a sequence of numbers as output based on the seed value in a uniform distribution.

C. RSA Encryption and Decryption

Cryptography has two kinds of key generation. They are symmetric key and asymmetric encryption. In symmetric key cryptography for encryption and decryption we use the similar key. For this key should be shared with the recipient carefully. So to avoid this problem of key sharing public key cryptography is used where public key encrypts the data which may be later decrypted by recipient by generating their own private key which matches with the public key.

The sequence of number generated using Random () function should be secretly saved as it is the key for encoding the files. These generated sequences of numbers are preserved from attackers by using RSA algorithm. RSA is asymmetric cryptographic algorithm. Asymmetric cryptography means using two mathematically related but not identical keys for encoding and decoding. Since RSA algorithm is a public key encryption, it uses to key called public and private key. This algorithm uses two keys one for encryption and another for decryption i.e. the public key using public key. It can decrypt only by the private key which the rightful receiver will have that and it is generated from the public key. To establish this sort of security to key that is the "CSPRN" generated, we use RSA public key cryptography algorithm. Each key generated is encrypted using RSA Encrypt technique. Here the key is changed to an unreadable cipher form from Server and sent to the Client. There client uses RSA Decrypt technique and generate a private key to decrypt the cipher and get back the key for encoding.

The idea of RSA is based on the fact that it is difficult to factorize a large integer. We will assume two large prime numbers. The public key consists of two numbers and produces another large integer by multiplying these two prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the

large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially.

ALGORITHM:

P ← BIG INTEGER PRIME NUMBER;
Q ← BIG INTEGER PRIME NUMBER;
N ← P*Q; //PUBLIC KEY
E ← GCD (E, Φ(n))=1 //1<E<Φ(n);
Φ(n)← (P-1)*(Q-1);
D ← DE≡1 (MOD Φ (n))//PRIVATE KEY
OR
D≡E−1 (MOD Φ (n)); i.e., d is the modular multiplicative inverse of e (modulo Φ(n).

For Encryption: c=m^ (e) mod (n);
For Decryption: m=c^ (d) mod (n);

Here m is message to be sent secretly and c is the cipher text formed after encryption. Here m is nothing but CSPRN generated and c equals the encrypted key from the server. This key is again decrypted securely on the client side.

D. **Splitting and Merging:**

The proposed system deals with secured transfer of file between mobile devices. PLUTO can store and transfer files of any size. Mobile devices have memory constraints and it is difficult to store larger files within it. It may lead to battery drainage and has an adverse effect on the performance of mobile device. The proposed system involves splitting the file of any size into smaller chunks. This makes the protocol suitable for storing even large files within a mobile device. These smaller chunks get merged at the receiver side after successful transmission.

E. **Parallel XOR:**

This part place the major role in PLUTO. It enhaces the efficiency of the algorithm by splitting thewhole large file into small piece of chunks and then process all them parallely.

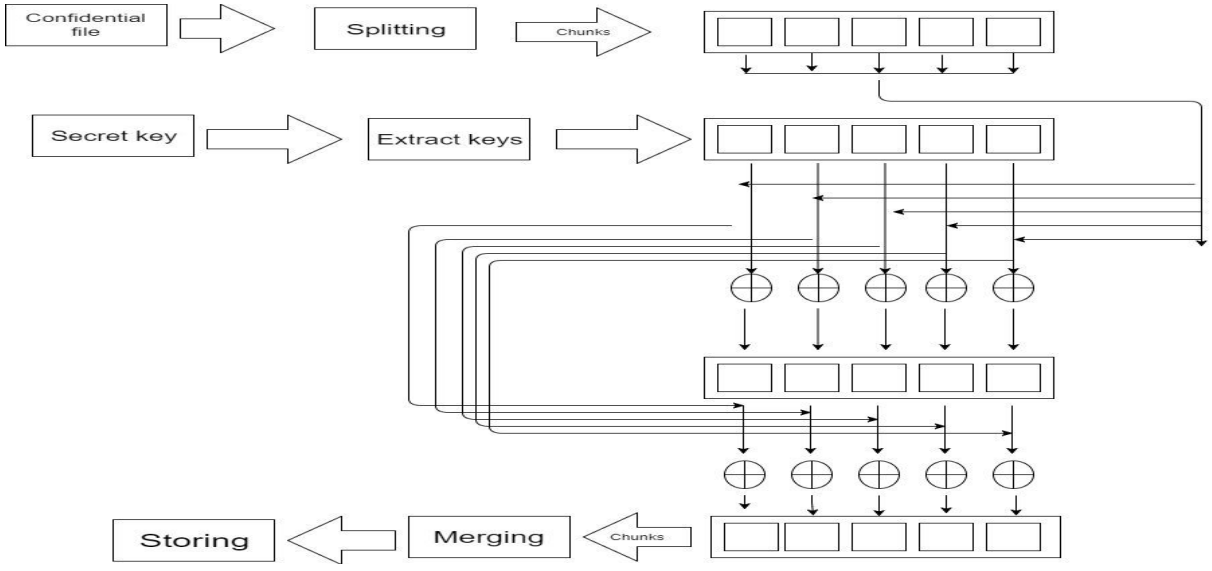


Figure 3. Parallel Encoding and Decoding

Figure 3 explains the whole process behind the parallel processing. Initially we will get the huge amount of input data and then we partition the data with some size and the whole set of partition all are Xored with the key. The files to be transferred between the clients should be made secure. To ensure such sort of security the file should be encoded. Each chunk file created after splitting should be encoded parallelly to improve the computational capability. For encoding a file the proposed system performs simple xoring of file with the CSPRN generated. Each number in the sequence generated becomes a key for Xoring each chunks created after splitting the file. These xoring is done parallel so that each chunk is xored with a key simultaneously. It improves performance of mobile devices and saves power thereby prevents battery drainage.

These encoded splits are transferred to the client who needs access to that file. Later all encoded chunks are again Xored with the same key to retrieve the original file. All these decoded chunks are again merged to get back the original file. The proposed system ensures more security by encrypting the key which is used for Xoring the file. PLUTO helps in secured transfer of confidential files among the users.
The proposed system ensures more security by encrypting the key which is used for Xoring the file. PLUTO helps in secured transfer of confidential files among the users.

Algorithm for PLUTO framework:

1. for I = {1,2.....,bound}do
   1.1 Random r =new random ();
   1.2 Print r.nextInt (bound);
2. End for
3. Establish Server Socket connection.
4. Generate_probable_prime (P,Q);
   4.1 KeyShare (P,Q);
5. Generate_public_key (N);
   5.1 N=P Multiply (Q);
   5.2 Compute (phi (N));
6. Compute_public_key (E);
   6.1 if(1<E<phi(N)) then
      6.1.1 KeyShare (E);
   6.2 end if
7. Encrypt_key (random_no,E,N)
   7.1 C=E pow (random_no) mod(N);
8. Establish Client Connection.
9. Compute_private_key (E,Phi(N));
   9.1 D=inversemod (E,Phi(N));
10. Decrypt_key(C,D,N);
    10.1 random_no =D pow (C) mod(N);
11. Read_input_file(Filename);
12. Get input (Partsize,File length);
13. While (filelength>0)
    13.1 CreateChunk(File length/Part size);
    13.2 Store it in a output stream.
14. End while
15. Assign Split1 as master process;
16. Split2, split3....splitn as slave process;
17. While (chunks!=Null)
    17.1 For i={1,2...Partsize}do
    17.2 output[i]=Chunk(i)(Xor)key(i)
    17.3 end for
18. End while

19. Establish ServerSocket connection
20. Share_encoded_chunks(output);
21. Getkey(random_no);
22. for i={1,2...Partsize}do
    22.1 result[i]=Chunk(i)(Xor)key(i)
23. end for

Figure 3.1 Algorithm for PLUTO protocol

F. **Experimenting Results**

For the analyses, We use different input file size like 38kb, 5mband 17mb files with these files first we split and calculated the computation time is nothing but time taken to encode the files .Next is communication time which is time taken between the processors. Last one is the execution time which is the addition of both the communication and the computation time.

Table 1. Recordings For 38KB

No. of Processors	Computation time	Communication time	Execution time
2	0.015547	0.003756	0.019303
3	0.006116	0.005309	0.011425
4	0.005137	0.005621	0.010758
5	0.003378	0.005239	0.008617
6	0.002384	0.007736	0.010120

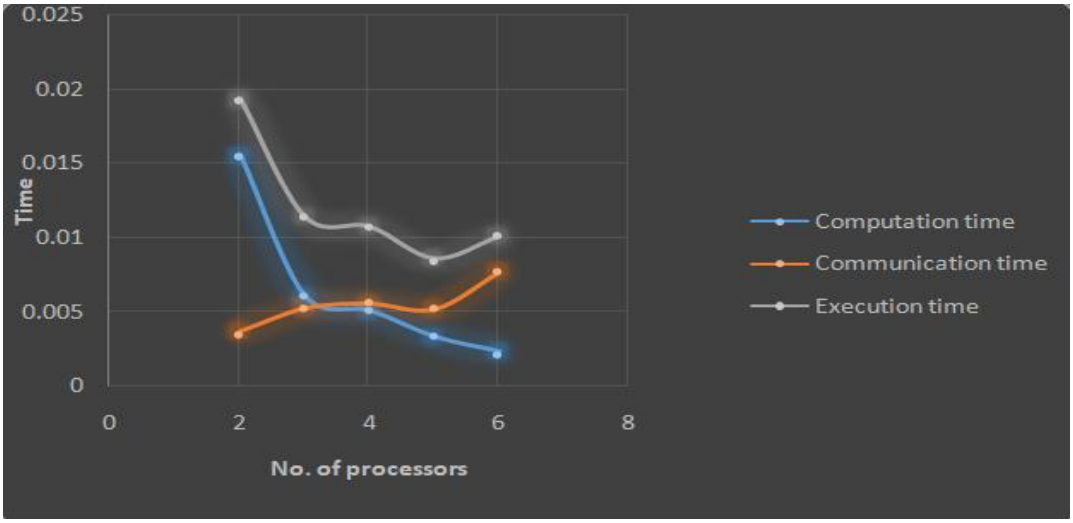


Figure 4. Encryption time for 38kb.

In this graph (fig 4), by comparing the communication, computation and execution time we can say that for smaller file size we can't discuss the differentiation. Though from this we can say that as number of splits increases the total

time is reducing. We case that there is not much any time difference in the computation time but in the execution and communication time since the number of process is increasing and t is processing parallelly execution time is decreasing. Beyond some number of processes the execution time starts to increase than the communication time and in that point we can mention that as a threshold point. After that threshold point there is no efficiency in using this way for 38 kb. Since the file size is too small, after some level the communication starts to increase from lower to higher. Hence couldn't get the exact threshold value. For this reason we used some huge size value.

Table 2.Recordings for 5kb

No. Of Processors	Computation time	Communication time	Execution time
2	0.001246	0.030698	0.031944
3	0.00083	0.003842	0.004672
4	0.000669	0.003028	0.003697
5	0.000634	0.003263	0.003897

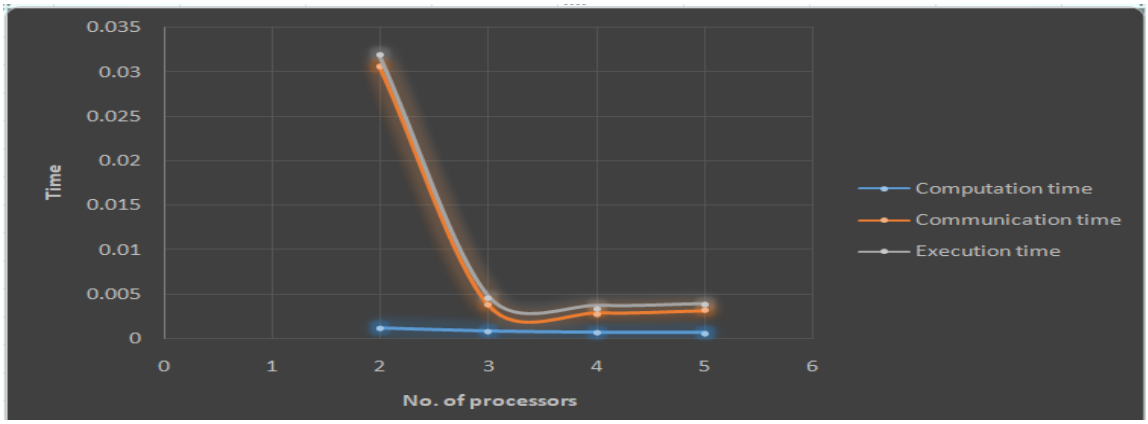


Figure 5:total time taken for encryption of 5kb

In Figure [5] we use the 5kb file which is smaller than the previous file we get the graph like below. From the graph we can say that as the number of process increases and all the process are processing parallelly the total execution time is reduced and in this graph we can get the threshold value. After that threshold point the number of process increases due to large communication time between the process the total execution time also increased.

Table 3.recordings for 17mb

No. Of Processors	Computation time	Communication time	Execution time
2	6.7362	20.8653	27.60159
3	4.9733	20.5737	25.5471
4	3.7116	18.3818	22.0934
5	3.2519	14.9474	18.1993
6	2.3409	15.9865	18.3274
7	2.2124	15.915	18.1275
8	1.76404	15.3945	17.1586
9	1.68943	14.775	16.4645
10	1.5062	13.212	15.262
11	1.5031	14.012	15.011

For better values we are using 17mb file with this file we get around some 11 splits with that data if we plot the data means we get the graph like figure 6. From this figure for no splits to two splits there is decrease in all the three computations, communication and execution time. Further increase in splits there is decrease of around 0.005 second in the total time for execution .After 10 splits the execution time started to increase than the communication time. That point is defined as a threshold value for this 17mb file. The threshold value varies for the different file size. To find a solution for this we take large number of input file with varies size and performs analyses. From that we derive that when number of splits exceeds the threshold it is not more efficient. This algorithm is efficient up to that point alone. We measured this time using time function.

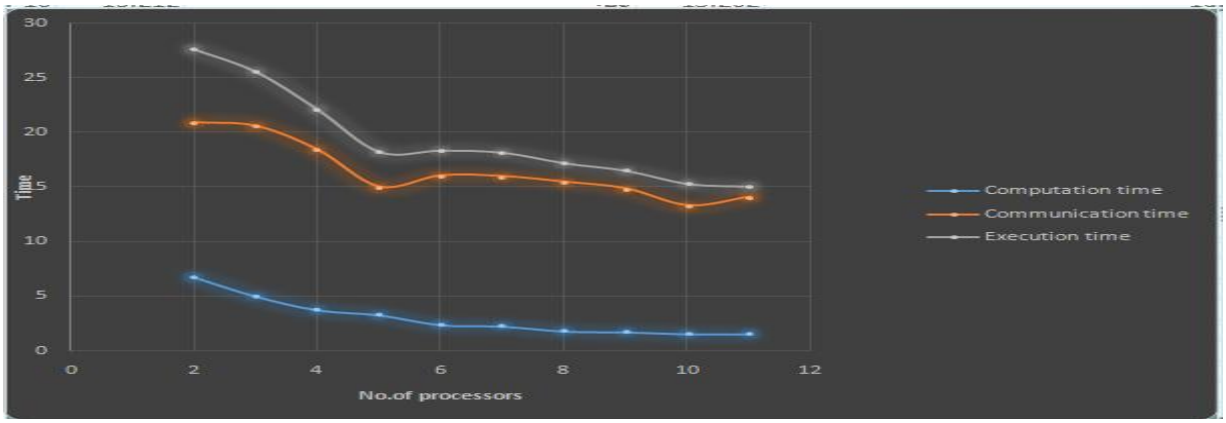


Figure6: total time taken for encrytipon of 17mb

IV.Conclusion

This paper explains about how we can reduce the battery drainage by parallel computation in resource constrained devices. Since we are using this algorithm in mobile phones instead of using heavy weight algorithm we are using light weight encryption method. By using asymmetric key cryptography it will be very difficult to hack the data by the intruders in the middle. PLUTO compresses of all the CIA triads and enhances the authentication only valid receiver cab receiver the data.

References

[1] Y. Baseri, A. Hafid and S.cherkaoui,"Privacy preserving fine-grained location-based access control for mobile cloud." In Computers and Security, 2018,pp249-265.

[2] Vipul goyal, Omkant Pandey, Amit sahai and Brent Waters,"Attribute Based Encryption for Fine Grained Access Control of Encrypted Data" in 2006.

[3] L.T.Phong,Y.Aora,T.Hayashi,L.Wang,S.Moriai,"Privacy-Preserving Deep Learning Via Additive HomomorphicEncryption"in IEEE Transcation of Information and Forensics and Security13(5),8241854, 2018,pp. 1333-1345 .

[4] Zhifeng Xiao and Yang Xiao,"Security and Privacy in Cloud Computing",in IEEE communications Survey and Tutorial ,VOL15,NO 2,Second Quarter 2013.

[5] Sarita Kumari,"A Research paper on Encryption And Compression Techniques .",in Internationa Journal of Engineering and Computer Science ISSN:2319-7242 ,VOL 6, ISSUE 4 April 2017, pp. 20915-20919.

[6] Yevgeniy Dodis, David Pointcheval, Sylvain Ruhault, Damien Vergnaud and Daniel Wichs, "Security Analysis of Pseudo Random Number with Input: /dev.random isnot Robust"in Department of Computer Science , NewYork University.

[7] Xin Zhou, Xiaofei Tng ,"Research And Implementation of RSA Algorithm For Encryption And Decryption"in 6th International Forum on Strategic Technologies , 2011.

[8] Roelof Kemp, Nicholas Palmer, Thilo Kielmann, and Henri Bal, "CUCKOO: A Computational Offloading Framework For Smart Phones" in Vriji University, De Boelelaan 1081A, Amesterdam, Netherland, 2012.

[9] Balachandra Muniyal, Krishna Praksh, Shashank Sharma, "Wireless Public Key Infrastructure For Mobile Phones" in International Jounal of Network Security and its Application(IJNSA),VOL 4,NO 6,November 2012.

[10] Kuang.Z, Guo.S, Liv.J, Yang Y,"A Quick-Response Framework For Multiuser Computation Offloadin in Mobile Cloud Computing" in Future Generation Computer Systems 81,pp. 166-179,2018.

[11] Amit Banerjee,Mahamudu Hasan, Auhidur Rahman, Rajesh Chapagain, "CLOAK: A Strream Cipher Based Encryption ProtocolFor Mobile Computing ",IEEE Access 5,8016572, pp.17678=17691,2017.

[12] David Pointcheval, "Practical Security in Public key Cryptography",in 4th Internationalconference on Information Security and Cryptography , LNCS 2288, pp. 1-17, Verkarg, 2002.

[13] Sundeshna Sinha, Toshinori Monocat and William L. Ditto,"Flexible Parallel Implementaion of Logical Gates on Chaotic Elements", in Computer and Infrormation Departmen, cleveland, 2002

[14] Haoqiang jin, Dennis Jespersen, Piyush Menrotra, Rupak Biswas, Lei Huang, Barbara Chapman, "High Perfomance Computing Using MPI and Open MP on Multi core Parallel Systems", Vol 37, Issue 9, September 2011, pp. 562-575.