CLOUD STORAGE BASED LIGHT-WEIGHT CRYPTOSYSTEM FOR IOT

15BCE0082-Voleti Ravi, 15BCE0093-Samudra Pratim Borkakoti, 15BCE0076-Vignesh Vaidyanathan

Co-author and Corresponding Author- Prof. Madhu Viswanatham V (Email – vmadhuviswanatham@vit.ac.in) of Vellore Institute of Technology, Vellore, Tamil Nadu – 632014

With the progress in data exchange by electronic system, the need of information security has become a necessity. Due to growth of multimedia application, security becomes an important issue of communication and storage of data. This paper is about a cryptosystem using a one-way function to generate the secret-key block cipher designed to increase security and to improve performance.

We aim in this project to encrypt message of any size using an original encryption algorithm. We encrypt the message using the algorithm using various methods. Our project aims to improve the performance by our method of encryption technique which helps in speeding up the whole encryption and decryption process than the already existing ones. We implement a custom make random key generator to improve the security of the encryption algorithm. The project is realized using Python.

***KEYWORDS* — Cloud computing, Privacy, Cyber security, Encryption, Decryption, One-way function**

I. INTRODUCTION

Because of the increasing demand for information security, data encryption-decryption has become an important research area and it has broad application prospects. The field of encryption is becoming very important in the present era. Data security is of utmost concern as web attacks have become more and more serious. Data encryption-decryption has applications in internet communication, multimedia systems, medical imaging, telemedicine, military communication, etc.

Encryption is the process of transforming the information for its security. With the huge growth of computer networks and the latest advances in digital technologies, a huge amount of digital data is being exchanged over various types of networks. It is often true that a large part of this information is either confidential or private.

Many encryption algorithms have been proposed. To make the data secure from various attacks and for the integrity of data we must encrypt the data before it is transmitted or stored. Government, military, financial institution, hospitals and private business deals.

II. LITERATURE REVIEW

Cloud computing has brought up major advancements to the IT industry. Building on its predecessors, namely, grid and utility computing, this new evolutionary model is witnessing a rapid expansion and proliferation. Today, clients are capable of running their software applications in remote computing clouds where data storage and processing resources could be acquired and released, almost, instantaneously. The virtualization layer on top of the commodity hardware in computing clouds is the driving force that allows cloud providers to “elastically” and promptly respond to client resource demands and requirements.

In spite of all the advantages delivered by cloud computing, several challenges are hindering the migration of customer software and data into the cloud. On top of the list is the security and privacy concerns arising from the storage and processing of sensitive data on remote machines that are not owned, or even managed by the customers themselves. With cloud computing, all the customer can see is a virtual infrastructure built on top of possibly non-trusted physical hardware or operating environments.

So there is a necessity for a secure way to store and retrieve data especially in IoT devices which are predominantly succeptable to attacks.

III. SYSTEM ARCHITECTURE

A. Problem Definition

i. Performance: We will be addressing the speed of encryption using a customized method. As compared to our Reference papers, we are trying to minimize the time complexity required for the encryption while simultaneously requiring less processing power.

ii. Security: This is still under research but our goal will be mostly to provide a randomised 256 bit key. Furthermore when we encounter problems within the project additional security measures will be taken.

B. System Approach

i. There are three locks in the encryption process.
   a. Choosing a random value C (128-bits),
   b. Creating a C’ (128-bits), and
   c. Encrypting C’ to get α (128-bits).

Therefore, to find the plain-text through cryptanalysis, the programmer has to go through 2128*2128*2128 which is extremely big for computation.

ii. The encrypted message is stored in multiple clouds. So, the hacker has to have access to all cloud storage systems to begin cryptanalysis.

iii. System is very light-weight which makes it possible to be used in systems like IoT devices.

IV. OUR PROPOSED METHODOLOGY

This algorithm has three distinct important components:

i. Key generation

ii. Encryption

iii.Decryption

This document will explain these segments in detail below with proper explanation under each component.

i. KEY GENERATION

Every time the code is run for message encryption, a random 256-bit key (K) is generated. This key is stored by the user as secret-key for message decryption later. The 256-bit key is split into two halves (K1 and K2 128-bit each), which are used in a one-way function to form a 128-bit encryption key (E).

ii. ENCRYPTION

The user provides a message plain-text (P) and it is converted to 128-bit blocks. A random 128-bit text (C) is generated which is smaller than plain-text (P) and subtracted to get another 128-bit text (R).

$C; R = P - C;$

The exclusive-OR of C with E and R with E is calculated to get C’ and R’ respectively.

$C' = C \oplus E; R' = R \oplus E;$

Again, similarly, exclusive-OR of C’ with K1 and R’ with K2 is calculated to get α and β respectively.

$\alpha = C' \oplus K1; \beta = R' \oplus K2;$

The values of α and β are stored in different cloud storage systems.

iii. DECRYPTION

The values of α and β are extracted from the cloud storage systems. The exclusive-OR of α with K1 and β with K2 is calculated to get C’ and R’ respectively.

$C' = \alpha \oplus K1; R' = \beta \oplus K2;$

Again, similarly, exclusive-OR of C’ with E and R’ with E is calculated to get C and R respectively.

$C = C' \oplus E; R = R' \oplus E;$

Then, C and R is added to get the original plain-text (P).

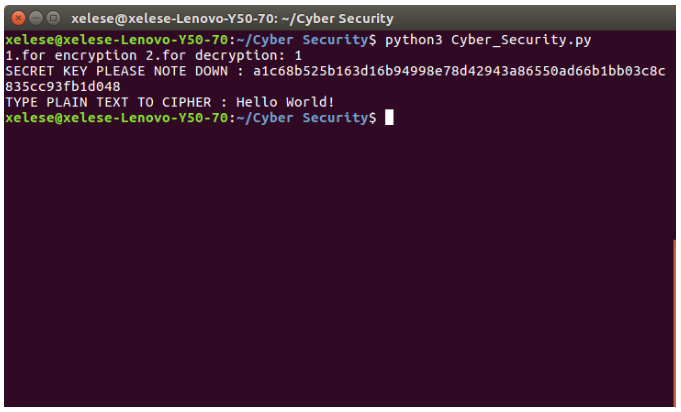
$P = C + R;$

This plain-text can be viewed as the original message by the user.

V. TIME COMPLEXITY

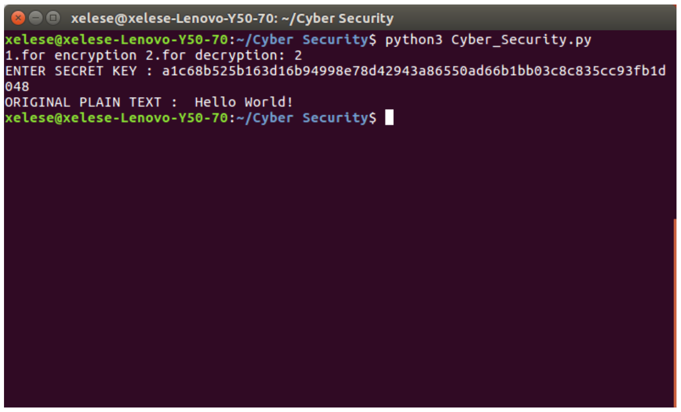
For any given input, the time complexity of this program does not exceed *O(n)*.

VI. RESULT



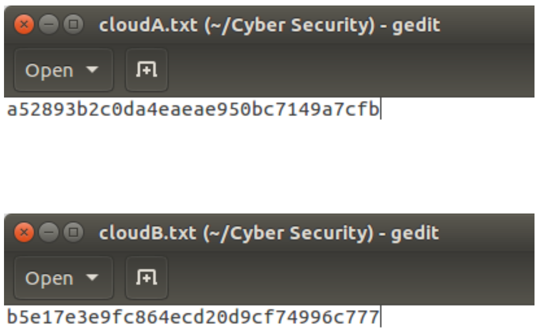
```
xelese@xelese-Lenovo-Y50-70: ~/Cyber Security
xelese@xelese-Lenovo-Y50-70:~/Cyber Security$ python3 Cyber_Security.py
1.for encryption 2.for decryption: 1
SECRET KEY PLEASE NOTE DOWN : a1c68b525b163d16b94998e78d42943a86550ad66b1bb03c8c
835cc93fb1d048
TYPE PLAIN TEXT TO CIPHER : Hello World!
xelese@xelese-Lenovo-Y50-70:~/Cyber Security$ 
```

Fig 1. ENCRYPTION



```
xelese@xelese-Lenovo-Y50-70: ~/Cyber Security
xelese@xelese-Lenovo-Y50-70:~/Cyber Security$ python3 Cyber_Security.py
1.for encryption 2.for decryption: 2
ENTER SECRET KEY : a1c68b525b163d16b94998e78d42943a86550ad66b1bb03c8c835cc93fb1d
048
ORIGINAL PLAIN TEXT :  Hello World!
xelese@xelese-Lenovo-Y50-70:~/Cyber Security$ 
```

Fig 2. DECRYPTION



cloudA.txt (~/Cyber Security) - gedit
Open

```
a52893b2c0da4eaeae950bc7149a7cfb
```

cloudB.txt (~/Cyber Security) - gedit
Open

```
b5e17e3e9fc864ecd20d9cf74996c777
```

Fig 3. CLOUD STORAGE

VII. CONCLUSION

Inferring from above, we can conclude this program is reliably fast, uses low memory, and has a fairly low time complexity making it especially useful in IoT devices where “things” are constrained by low memory and computation complexity. Since the storage mechanism is split, analysing the cipher-text becomes increasingly difficult as the attacker needs to have access to all the cloud systems.

VIII. REFENRENCES

[1] RM Best, “Preventing Software Piracy with Crypto-Microprocessors”, in Proceedings of IEEE Spring COMPCON 80, pp 466-469.

[2] L C. Guillou, M. Ugon, and JJ. Quisquater, “The smart card: A standardized security device dedicated to public cryptology”, In Gustavus J Simmons, editor, Contemporary cryptology: The science of information integrity, IEEE Press, Piscataway, NJ, 1992.

[3] J. D. Tygar and B. Yee, “Dyad: A system for using physically secure coprocessors”, In Proc. of IP Workshop, 1994.

[4] S.R. White and L. Comerford, “ABYSS: An Architecture for Software Protection”, IEEE Transactions on Software Engineering, vol. 16, No. 6, June 1990, pp. 619-629.

[5] S.R. White, S.H. Weingart, W.C. Arnold, E.R. Palmer, Introduction to the Citadel architecture: security in physically exposed environments, Technical Report RC 16672, Distributed Systems Security Group, IBM T.J. Watson Research Center, March 1991.

[6] S.H. Weingart, Physical security for the mABYSS system, IEEE Computer Society Conf. on Security and Privacy, 1987.

[7] Theodoropoulos, D., et al., “Cproc: An efficient Cryptographic Coprocessor”, in Proceedings of the 16th IFIP/IEEE International Conference on Very Large Scale Integration, 2008.

[8] S. Weingart, “Physical security for the mABYSS system”, in Proc. of the IEEE Computer Society Conference on Security and Privacy, pp. 52-58, 1987.

[9] J. Dyer, M. Lindemann, R. Perez, R. Sailer, S. Smith, L. van Doorn, and S. Weingart, “Building the IBM 4758 secure coprocessor”, IEEE Computer, 34:57–66, October 2001.

[10] T.W. Arnold, L.P. Van Doorn,”The IBM PCIXCC : A new cryptographic coprocessor for the IBM eServer”, IBM Journal of Research and Development, Vol 48, May 2004.