ESSAO: Enhanced Security Service Algorithm using Data Obfuscation Technique to Protect Data in Public Cloud Storage

S. Balamurugan¹, S. Sathyanarayana² and S. S. Manikandasaran³*

¹Department of Computer Science, Bharathiar University, Coimbatore - 641046, Tamil Nadu, India
²Department of Computer Science, Sreekanta 1st Grade College, Mysore - 570004, Karnataka, India
³Department of MCA, CICA, Christhu Raj College, Trichy - 620012, Tamil Nadu, India; moni.tamil@gmail.com

Abstract

Background/Objectives: Cloud provides virtual space for users to storage and maintains their data. It mainly helps small and medium scale industries to easy come up and achieve their IT requirements with minimum cost. Apart from this, it has many security related issues. To address the security issues, this paper proposes an enhanced security service algorithm using data obfuscation technique namely ESSAO. Methods/Analysis: The proposed algorithm is provided as a service to user from the cloud. It is based on data obfuscation technique and it obfuscates only the numerical data in the original data. It uses different mathematical methods to convert the original data into unintelligible data. Findings: Cloud provides this proposed ESSAO algorithm as a service to the users based on the demand. This algorithm is coded in Java and hosted in the cloud server for evaluation. ESSAO reduces the time taken for obfuscation and de-obfuscation. It also reduces the size of data after obfuscation. Each numerical value in the original data is converted into a single character value. Hence, the size of data is reduced. It minimizes the latency of data uploaded to cloud. Hence, it is easily adopted in the cloud environment. Obfuscated data in the cloud is not accessible to the cloud service providers or other users of cloud services. Hence, it maintains the security of data in the cloud. Novelty/Improvements: Addressing security to the cloud environment is difficult task. This proposed algorithm provides security to the data in the cloud. This work is more applicable to education, medical, govt. and etc. to secure their data in the cloud.

Keywords: Cloud Computing, De-obfuscation, Obfuscation, Security Service Algorithm

1. Introduction

Cloud computing is an environment consists of huge amount of computing infrastructure to serve their users based on their demand. It is an unlimited service provisioning system in which users have to pay for what they are consumed from the cloud services. It mainly helps small and medium scale industries for their computing infrastructure needs. There is a big wall when adopting cloud is security of data. Cloud has many challenges and issues, but among that security is main concern in the cloud. Mostly, cryptography techniques are preferred for securing the data and most of the techniques are outdated. Instead of using the traditional technique, data obfuscation is the better choice to secure data in the cloud.

Data Obfuscation is one of techniques widely used nowadays for securing the data in the cloud. Data obfuscation is a method of data hiding where data is purposely scrambled to prevent from unauthorized access. The result of obfuscation is an unintelligible or confusing data. Data obfuscation techniques are used to prevent the intrusion of sensitive data stored in the cloud. However, issues have stemmed from an inability to vigorously prevent security attacks.

It reduces the risks of theft of the data from the cloud and unauthorized uses of the data. Moreover, the obfus-
It is responsible for cloud service providers to maintain the security of the data in cloud. But in public cloud environment, there is a possibility that service provider can access the data without the knowledge of data owner. To avoid this data being accessed by the service providers, users should handle the obfuscation and de-obfuscation. It is necessary to develop better security mechanism to washout the security issues in the cloud.

3. Methodology of Proposed Work

The proposed algorithm ESSAO is executed before the data are uploaded to the cloud storage. ESSAO uses different mathematical function to obfuscate the users' data. The procedure of the ESSAO is as follows,

Procedure of ESSAO

- Users’ data as Plaintext (PT).
- Consider the numerical values in the PT.
- Find the length (N) of PT.
- Determine the position (POS) of each value in PT.
- Multiple (MUL) PT with its positional value.
- Find Square (SQU) of each value in the MUL.
- Get the key K from KaaS cloud service
- Apply the Key K on SQU, for each value in the SQU, key K is incremented by 1.
- Find the Reverse (REV) of SQU
- Modulus (MOD) REV by 256, co-efficient value is the Secret Key (SK).
- Convert the MOD value into ASCII code
- ASCII converted value is the Ciphertext (CT).

ESSAO gets the plaintext from the users for obfuscating the text. ESSAO converts all the characters in the plaintext into ASCII decimal code and these values are computed with the following mathematical functions such as multiply(), pow(), reverse(), modulus() and ascii(). This algorithm uses a key K for obfuscation and this same key used for de-obfuscation. Pseudo code of the proposed ESSAO is as follows,

Pseudo Code of proposed ESSAO

Declarations:

- PT ← Plaintext
- N ← length of PT
POS $\rightarrow$ Positional value
MUL $\rightarrow$ Multiplied value
SQU $\rightarrow$ Squared value
K $\rightarrow$ Key from KaaS
REV $\rightarrow$ reversed value
MOD $\rightarrow$ Mod value
SK $\rightarrow$ Secret key value
CT $\rightarrow$ Ciphertext

ESSAO(PT)

- start
- PT $\leftarrow$ plaintext
- N $\leftarrow$ length(PT)
- for i $\leftarrow$ 0 to N-1
  - POS(i) $\leftarrow$ position(PT)
- end for
  //Multiple POS(i) into PT(i)
- for i $\leftarrow$ 0 to N-1
  - MUL(i) $\leftarrow$ PT(i)*POS(i)
  //find square SQU value for MUL(i)
  - SQU(i) $\leftarrow$ pow(MUL(i),2)
  //Reverse the SQU(i)
  - Get a key K and apply it on SQU(i)
  - SQU(i) $\leftarrow$ SQU(i)+(K+j), j=0,1,2,3,…N
  - REV(i) $\leftarrow$ reverse(SQU(i))
  //Find the module MOD for REV by 256
  - MOD(i) $\leftarrow$ REV(i)%256
  - SK(i) $\leftarrow$ REV(i)/256
  //Convert the MOD into ASCII code to produce Ciphertext CT
  - CT(i) $\leftarrow$ ascii(MOD(i))
- end for
  - Ciphertext $\leftarrow$ CT
- end

4. Experimental Results

Experiment is conducted with sample data. ESSAO uses a key for obfuscation de-obfuscation. The key is generated from a cloud service called KaaS and communicated to the user directly. Users are responsible to keep the key safe. Students' marks details records are taken as sample data for experiment.

4.1 Obfuscation Procedure with Sample Data

| Table 1. Consider the students marks table with four subjects |
|-------------------------------------------------------------|
| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | 56   | 78   | 76   | 87   |
| Stud 2  | 82   | 37   | 54   | 74   |
| Stud 3  | 45   | 77   | 56   | 65   |
| Stud 4  | 56   | 76   | 84   | 37   |

Step 1: Plaintext taken from the student mark table
PT $\leftarrow$ 56 78 76 87 82 37 54 74 45 77 56 65 56 76 84 37

Step 2: Length of the PT
N $\leftarrow$ 16

Step 3: Determine the positional POS value of PT

| POS(0) | 1 $\rightarrow$ 56 |
|------|------------------|
| POS(1) | 2 $\rightarrow$ 78 |
| POS(2) | 3 $\rightarrow$ 76 |
| POS(3) | 4 $\rightarrow$ 87 |
| POS(4) | 5 $\rightarrow$ 82 |
| POS(5) | 6 $\rightarrow$ 37 |
| POS(6) | 7 $\rightarrow$ 54 |
| POS(7) | 8 $\rightarrow$ 74 |
| POS(8) | 9 $\rightarrow$ 45 |
| POS(9) | 10 $\rightarrow$ 77 |
| POS(10) | 11 $\rightarrow$ 56 |
| POS(11) | 12 $\rightarrow$ 65 |
| POS(12) | 13 $\rightarrow$ 56 |
| POS(13) | 14 $\rightarrow$ 76 |
| POS(14) | 15 $\rightarrow$ 84 |
| POS(15) | 16 $\rightarrow$ 37 |

Step 4: Multiple (MUL) PT with its positional value POS(i).
MUL(i) $\leftarrow$ POS(i)*PT(i)

| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | 56   | 156  | 228  | 121104 |
| Stud 2  | 410  | 222  | 378  | 592   |
| Stud 3  | 405  | 770  | 616  | 780   |
| Stud 4  | 728  | 1064 | 1260 | 592   |

Step 5: Find Square (SQU) of each value in the MUL(i).
SQU(i) $\leftarrow$ pow(MUL(i),2)

| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | 3136 | 24336| 51984| 121104|
| Stud 2  | 168100| 49284| 142884| 350464|
| Stud 3  | 164025| 592900| 379456| 608400|
| Stud 4  | 529984| 1132096| 1587600| 350464|
Step 6: Get the sample key from KaaS
\[ K \rightarrow 1282 \]

Step 7: Apply the Key K on SQU(i)
\[ SQU(i) \leftarrow SQU(i) + (K + j), j \in \{0, 1, 2, \ldots, N-1\} \]

| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | 4418 | 25619| 53268| 122389|
| Stud 2  | 169386| 50571| 144172| 351753|
| Stud 3  | 165315| 594182| 380748| 609693|
| Stud 4  | 531278| 1133391| 1588896| 351761|

Step 8: Find the Reverse (REV) of SQU(i)
\[ REV(i) \leftarrow reverse(SQU(i)) \]

| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | 8144 | 91652| 86235| 983221|
| Stud 2  | 683961| 17505| 271441| 357153|
| Stud 3  | 513561| 281495| 847083| 396906|
| Stud 4  | 872135| 1933311| 6988851| 167153|

Step 9: Modulus (MOD) REV(i) by 256, co-efficient value is the Secret Key (SK).
\[ SK(i) \leftarrow REV(i)/256 \]

| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | 31   | 358  | 336  | 3840 |
| Stud 2  | 2671 | 68   | 1060 | 1395 |
| Stud 3  | 2006 | 1099 | 3308 | 1550 |
| Stud 4  | 3406 | 7551 | 27300| 652  |

MOD(i) \leftarrow REV(i) \mod 256

| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | 208  | 4    | 219  | 181  |
| Stud 2  | 185  | 97   | 81   | 33   |
| Stud 3  | 25   | 151  | 235  | 106  |
| Stud 4  | 199  | 255  | 51   | 241  |

Step 10: Convert the MOD value into ASCII code

| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | ♣   | ♦   | ❑   | ❍   |
| Stud 2  | a   | Q   | !    |     |
| Stud 3  | ü   | δ   | J    |     |
| Stud 4  | 00  | 3   | ±    |     |

Obfuscated text:
\[ ♣ ♦ ❑ ❍ a Q ! ü δ j 00 3 ± \]

The obfuscated text contains symbols, alphabets, digits and extended ASCII codes. Numerical values in the plaintext are converted into single character ASCII values. Compare to numerical values, characters are minimum in size. Hence, the size of the data is reduced.

De-obfuscation is the reverse procedure of obfuscation with the same key used for obfuscation. Followings are de-obfuscation procedure of proposed ESSAO.

4.2 De-obfuscation Procedure with Sample Data

Consider the Obfuscated Text for de-obfuscation,

| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | ♣   | ♦   | ❑   | ❍   |
| Stud 2  | a   | Q   | !    |     |
| Stud 3  | ü   | δ   | J    |     |
| Stud 4  | 00  | 3   | ±    |     |

Cipher Text = ♣ ♦ ❑ ❍ a Q ! ü δ j 00 3 ±

Following steps represent the de-obfuscation procedure.

Step 1: length of the CT
\[ N = length (CT) = 16 \]

Step 2: Converts CT(i) value into ASCII decimal value.

| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | 208  | 4    | 219  | 181  |
| Stud 2  | 185  | 97   | 81   | 33   |
| Stud 3  | 25   | 151  | 235  | 106  |
| Stud 4  | 199  | 255  | 51   | 241  |

Step 3: Multiple SK(i) with 256 and sum CT(i)
\[ MUL(i) \rightarrow (SK(i) \times 256) + CT(i) \]

| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | 8144 | 91652| 86235| 983221|
| Stud 2  | 683961| 17505| 271441| 357153|
| Stud 3  | 513561| 281495| 847083| 396906|
| Stud 4  | 872135| 1933311| 6988851| 167153|

Step 4: Reverse MUL(i)
\[ REV(i) = reverse(MUL(i)) \]
Step 5: Use same key which is used for obfuscation
\[ K = 1282 \]

Step 6: Apply the Key K on REV
\[ \text{REV}(i) = \text{REV}(i) - (K + j), \quad j = 0, 1, 2, 3 \ldots N \]

| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | 4418 | 25619| 53268| 122389|
| Stud 2  | 169386| 50571| 144172| 351753|
| Stud 3  | 165315| 594182| 380748| 609693|
| Stud 4  | 531278| 113391| 1588896| 351761|

Step 7: Find Square root (SRT) of each value in the REV.
\[ \text{SRT}(i) = \sqrt{\text{REV}(i)} \]

| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | 56   | 156  | 228  | 348  |
| Stud 2  | 82   | 37   | 54   | 74   |
| Stud 3  | 45   | 77   | 56   | 65   |
| Stud 4  | 56   | 76   | 84   | 37   |

Step 8: Divide (DIV) SRT(i) with its positional value POS(i).
\[ \text{DIV}(i) = \frac{\text{SRT}(i)}{\text{POS}(i)}, \quad j = 0, 1, 2, \ldots N \]

| S. Name | Sub1 | Sub2 | Sub3 | Sub4 |
|---------|------|------|------|------|
| Stud 1  | 56   | 156  | 228  | 348  |
| Stud 2  | 82   | 37   | 54   | 74   |
| Stud 3  | 45   | 77   | 56   | 65   |
| Stud 4  | 56   | 76   | 84   | 37   |

Step 9: Original De-obfuscated value
De-obfuscated text \[ \leftarrow 56\ 78\ \ 76\ \ 87\ 82\ 37\ 54\ 74\ 45\ 56\ 76\ 65\ 56\ 76\ 84\ 37 \]

5. Result Analysis

The proposed ESSAO produces different ciphertext for same plaintext.
For example,
Plaintext is 56 78 76 87 82 37 54 74 45 76 56 65 76 84 37

\[ \text{Ciphertext above plaintext produce by ESSAO is} \]
\[ \text{Cipher Text} = \text{╨} \delta \text{╟} \text{003±} \]

Here, in plaintext the value ‘56’ is occurred three times in places of 1\text{st}, 11\text{th} and 13\text{th}. In ciphertext, the values in the same place are “╨ δ ╚”.
The value ‘76’ is also occurred two times in the places of 3\text{rd} and 14\text{th}. The ciphertext value for ‘76’ is different “█ 00”.
It shows that ESSAO produces different ciphertext for same plaintext values. It confuses the attackers to analysis the pattern of the data obfuscation. Hence, it strongly protects attacks like brute force and dictionary attacks on the obfuscated data stored in the cloud environment.

6. Advantages

The proposed ESSAO provides security advantage to the users who wants kept their data in the cloud storage.
- Data are obfuscated before uploaded to the cloud.
- Users no need to trust the cloud service provider for securing their data in cloud.
- Reduces the latency in data upload.
- Data size is minimized.

7. Conclusion

This paper has proposed an enhanced security service algorithm based data obfuscation technique namely ESSAO. It is only applied on numerical values in the users' data. It uses different mathematical function to operate
the original text into unintelligible text. A key is used for obfuscation and de-obfuscation. The key is kept by the users. This key is not communicated to the cloud service providers. ESSAO produces different ciphertext for the same plaintext value occurred more than once in the plaintext. It is confused the attackers to get the original data from the obfuscated data. Hence, the security of the data in cloud increased by using ESSAO. It reduces the size of the data after obfuscation. Hence, the data uploading latency is reduced. Security of the data is ensured by the proposed ESSAO algorithm.

8. Reference

1. Mell P, Grance T. The NIST Definition of Cloud Computing. Technical Report-800-145. Version 15. Gaithersburg, MD, United States: National Institute of Standards and Technology; 2011.
2. Sugumar R, Sheik Imam SB. Symmetric encryption algorithm to secure outsourced data in public cloud storage. Indian Journal of Science and Technology. 2015 Sep; 8(23):1-5.
3. Arockiam L, Monikandan S. Data security and privacy in cloud storage using hybrid symmetric encryption algorithm. International Journal of Advanced Research in Computer and Communication Engineering. 2013; 2(8):3064-70.
4. Rajathi A, Saravanan N. A survey on secure storage in cloud computing. Indian Journal of Science and Technology. 2013 Apr; 6(4):4396-401
5. Subashini S, Kavitha V. A survey on security issues in service delivery models of cloud computing. Elsevier Journal of Network and Computer Applications. 2011; 34(1):1-11.
6. Randy Raymond. Data Obfuscation for Test Environments. 2010. Available from: www.xensight.com
7. Robertson C. PDF Obfuscation - A Primer. TBA. 2012. 1-38. The SANS Institute. Available from: https://www.sans.org/reading-room/whitepapers/engineering/pdf-obfuscation-primer-34005
8. Govinda K, Sathiyamoorthy E. Agent based security for cloud computing using obfuscation. Elsevier, Science Direct Procedia Engineering. 2012; 125-9.
9. Data Obfuscation. 2013. Available from: http://www.techopedia.com/definition/25015/dataobfuscation-do
10. Nguyen M-D, Chau N-T, Jung S, Jung S. A demonstration of malicious insider attacks inside cloud IaaS vendor. International Journal of Information and Education Technology. 2014; 4(6):483-6.
11. Mowbray M, Pearson S, Shen Y. Enhancing privacy in cloud computing via policy-based obfuscation. The Journal of Super Computing. 2012; 61(2):267-91.
12. Arockiam L, Monikandan S. Efficient cloud storage confidentiality to ensure data security. Proceedings IEEE International Conference on Computer Communication and Informatics; 2014. p. 1-5.
13. Monikandan S, Arockiam L. Confidentiality technique to enhance security of data in public cloud storage using data obfuscation. Indian Journal of Science and Technology. 2015; 8(24):1-10.
14. Pearson S, Shen Y, Mowbray M. A privacy manager for cloud computing. Proceedings of International Conference on Cloud Computing: Berlin, Heidelberg: Springer-Verlag; 2009. p. 90-106.
15. Maheshwari V, Nourian A, Maheswaran M. Character-based search with data confidentiality in the clouds. Proceedings of IEEE International Conference on Cloud Computing Technology and Science; 2012. p. 895-9.