International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Implementation and Analysis of Enhanced Obfuscation Technique for Data Security

Tanuja, Meenakshi

Abstract- With the fast development of digital exchange of data in an electronic way, data and information security are becoming more important in both transmission and data storage. Cryptography is used as a solution which plays an important role in data and information security systems against malicious attacks. The encryption technique is used to provide confidentiality to the data during transmission because security threats are more on data during transmission than data at rest. One can also use encryption to secure user's data at data storage (i.e., data at rest). But an encryption algorithm consumes a more amount of computing resources such as processing power, memory and computation time. Obfuscation technique is a very lightweight technique that comes into a picture to protect the data at storage from malicious attacks. There are many obfuscation techniques are available to ensure the confidentiality of the data. In this paper, an obfuscation technique has been proposed and implemented which uses a 128-bit key to improve the security of the data. The experimental results show that the time taken for obfuscation and de-obfuscation is less and also from the security point of view it provides high avalanche effect.

Keywords- Obfuscation, De-Obfuscation, Avalanche effect, Data security, information security, Cryptography, Confidentiality

I. INTRODUCTION

In today's business world, data is one of the most valued resources that are required for maintaining a competitive edge. So, businesses must often be able to maintain secrecy of data, readily control the authenticity of the data, and strictly control access to data. Data systems commonly consist of different types of computer systems that are interconnected through many different electronic data networks. As the Internet becomes the basis for e-commerce and most of the businesses automate their data processing procedures, the possibility for disclosing private data to unlawful person increases.

Our main concern is data or information security whether the data or information is in transit mode or is at rest. To secure data in transit mode like data being transmitted through networks (e.g., the Internet, e-commerce), mobile, telephones, wireless intercom systems and wireless microphones, there are several cryptographic methods used to protect the data from the attacker. Conversion of plaintext into ciphertext is called encryption, and the reverse process of conversion of ciphertext into plaintext is called decryption [1]. And both encryption and decryption are controlled by a cryptographic key or keys.

Data can be tampered or modified by the outsiders as well as insiders when the data is not in transit mode, i.e., data stored in the cloud storage or computer systems. Today's main concern is to provide data confidentiality so that user can feel free to store to their data to the systems like cloud storage, computer systems, etc. To secure data at rest, we can use encryption technique, but it is not suitable for encrypting a large amount of data as it consumes lots of resources like CPU (Central Processing Unit), memory, etc. So, the technique data obfuscation is introduced which consumes less resources, computation time and computation power as compared to encryption technique.

Data Obfuscation is a process of transforming original data into data in which data is purposely jumbled to prevent it from unauthorized access. The result of obfuscation is unreadable data. Data obfuscation techniques are mainly used to prevent the intrusion of any sensitive data stored in the personal computer or cloud. The primary aim is to convert the data into the format which cannot be understood by a human being. Many research studies on obfuscation proved that data obfuscation made the attackers harder to understand and even impossible in some cases [2].

Significance of using obfuscation over encryption illustrated below:

- Obfuscation is relatively similar to encryption, but it incorporates simple programming logics or mathematical and binary operations like X-OR, two's complement, rotation whereas encryption includes complex fiestel structure comprising of many rounds of processing of the plaintext, each round comprising of a "substitution" step followed by a permutation.
- In obfuscation one can or cannot use key means it is not necessary to use the key but in encryption use of the key is must and plays an important role to secure data.
- Data in transit mode has more security threats than when data is at rest, so we need some strong cryptographic mechanism like encryption to secure the data in transit. Use of cryptographic mechanism to secure the data at rest will be time-consuming as this mechanism consists of complex structures.

Manuscript published on 30 September 2019
* Correspondence Author
Tanuja*, Department of Computer Science and Technology, Central University of Punjab, City Campus, Bathinda, India. Email: shalus384@gmail.com
Meenakshi, Department of Computer Science and Technology, Central University of Punjab, City Campus, Bathinda, India. Email: meenakshi.cup@gmail.com

So, when data is at rest, one can use an obfuscation which is less time consuming and provides high security as compared to encryption.

- Obfuscation is a lightweight mechanism as it consists of simple operations and programming logics. It uses fewer CPU resources, computing power and computational time as compared to encryption that is why encryption is called heavyweight mechanism.

II. LITERATURE REVIEW

Some important findings given by some authors in the direction of the obfuscation technique and cloud computing.

For cloud computing, a privacy manager is defined by **S. S.Pearson, Y.Shen and Mowbray,** which decrease the threat to the cloud storage user of their private data being altered or stolen[3]. Obfuscation technique is used by Privacy Manager. The users' private data are sent to the cloud in an obfuscated form. A key is used in the obfuscation method, chosen by the users and known by the privacy manager and not known to the Cloud Service Providers (CSPs). Therefore, the users' data can't de-obfuscate by the CSPs, and the data are not stored on the CSPs' machines. It decreases the threats of data theft from the unauthorized uses of the data and cloud.

A framework is proposed by **A. Rehman and M. Hussain** to store private data by using obfuscation and encryption techniques [4]. The cloud users store the keys used for encryption in the data storage. A mechanism is proposed to query over encrypted and obfuscated data on the server side. When the required data is filtered on the server side, then data is transmitted to the client side where de-obfuscation and decryption are implemented. Three approaches are presented by the authors such as separating infrastructure and software service providers, hiding data owner's information in the cloud and, data obfuscation technique for security. The authors presented a framework with a two-step encryption process used to entirely protect the encrypted private data.

L. Arockiam et al. proposed a new technique to attain confidentiality to resolve the issue of data security [5]. They provide confidentiality and also increase security using obfuscation and encryption techniques. Encryption key is kept a secret with the user and decryption key is used to access of this data. Results showed that the proposed technique could offer better security to cloud data than the existing techniques that are using obfuscation, encryption alone.

L. Arockiam and S. Monikandan proposed an encryption technique by combining substitution cipher and transposition cipher [6]. At first, the plain text is transformed into an equivalent ASCII code of each character in the plaintext and the further operations of proposed technique are performed on the same. In traditional encryption method uses key ranges from 1 to 26 and key can be a string (character combination). But in proposed algorithm, they increased a key range from 1 to 256 which provides more security.

T. Nie et al. proposed an algorithm which considers the consumption of power by the encryption algorithm [7]. Their aim is to perform and analysis of methods of power consumption of encryption algorithms. The experimental result reflects that software profiling and external measurement is more precise than those of the uninterruptible power system battery. Power consumption progress is 27.44 and 33.53% which shows software profiling and external measurements are more effective in evaluation of power consumption.

C. P. Dewangan and S. Agrawal implemented Advanced Encryption Standard (AES) by using MATLAB software. Their research aim is to increase the security level [8]. They proposed a technique, in which instead of giving plaintext and Encryption key directly to the AES algorithm, first, plaintext and encryption key are mapped in the different binary codes and then applied to the input of the AES algorithm. Avalanche Effect is calculated by changing one bit in plaintext keeping the key constant and vice-versa. Experimental results show that the proposed technique show high Avalanche Effect which increases security level.

A symmetric cipher model proposed by **S. Anbazhagan and K. Somasundaram**, which offers security to user's data in Cloud Computing [9]. This algorithm involved character to binary conversion then reverses the binary number and then perform all the further steps on that reverse binary number. This concept of an algorithm offers more security for securing the users' data. If the amount of data is less, it executes the faster computation of the data. If the amount of data is high, executes slow computation of data.

L. Arockiam and S. Arul Oli proposed an improved obfuscation technique to resolve the concern of data security [2]. They offered confidentiality using a technique that known as obfuscation and a key is used for obfuscation which is kept secret with the user and access of this data is only allowed by using a corresponding de-obfuscation key. This proposed technique provides high security level and has taken minimum time for obfuscation and de-obfuscation process compared to the existing techniques.

L. Arockiam and S. Monikandan offered the data confidentiality technique known as AROMO [10]. Authors considered the issue that insiders attack cloud storage (CS) from the cloud service provider (CSPs). With the help of proposed framework, user's data are protected in the Cloud Storage. Obfuscation and Encryption two techniques are used to protect the data. The data is encrypted and obfuscated before it is uploaded to CS. Metadata is preserved in the client. It holds the obfuscation and encryption details and applied to the data. A request is generated to run on obfuscated and encrypted data at the server side to recover data from the CS. The required data is retrieved from the CS. based on the request from the client and it could be de-obfuscated and decrypted at client side based on the metadata details. Integration of obfuscation with encryption ensures confidentiality and reduces the size of the data sent to Cloud Storage (CS).

III. OBJECTIVE

The objectives of this research are:

1. To propose an Obfuscation technique to protect both the numerical and non- numerical data.

2. To compare and analyze the proposed obfuscation technique with existing technique in terms of Obfuscation time, de-Obfuscation time and security level, i.e., (Avalanche effect).

IV. EXISTING OBFUSCATION TECHNIQUE

The Existing technique is an obfuscation technique which is proposed by L. Arockiam and S. Arul Oli [2]. Generally, obfuscation is a technique like encryption, but it uses different mathematical methods or some programming logics to hide original data. The Existing technique uses different methods like ASCII, binary, rotate, two's complement and decimal to obfuscate the original data. It also uses a key which is used to rotate the bits.

A. Existing Obfuscation Algorithm [2]

The steps in the existing obfuscation technique are as follows:

- *The given plain text is converted into ASCII codes*
- *ASCII codes are converted into binary values 0s' and 1s'*
- *Generate a key k*
- *Rotate the bits k numbers of time from right to left*
- *Calculate 2's complement*
- *Divide the bits into 8 bits*
- *Convert the binary into decimal*
- *Convert the decimal into ASCII character codes*

V. PROPOSED OBFUSCATION ALGORITHM

In the existing technique, simple mathematical methods and some programming logics are used as described in section IV. Key is also used to hide or secure the original data. The key is used to rotate the binary string by key number of times.

The main aim of this proposed research work is to design an Obfuscation technique to increase the security of data at virtual data storage.

In proposed obfuscation technique, numerical and non-numerical data is transformed in to binary. All the further operations are performed at bit-level because its effects are visible to the character level also.

A. Proposed Obfuscation Algorithm

The operations performed on the binary data are described as follows:

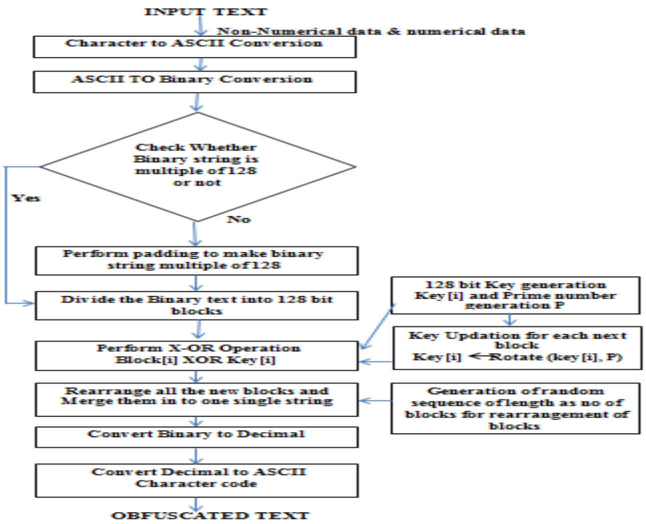

Fig.1. Flowchart of Proposed Obfuscation Technique

Main steps in the implementation of the Proposed Technique are as follows:

Step 1*: Character (Plain Text) to ASCII number Conversion & ASCII to Binary Conversion.*

Step 2: *Check whether the number of bits in the binary string is a multiple of 128 or not.*

Step 3: *If the number of bits in the binary string is already multiple of 128 then go to step 5.*

Step 4: *If the number of bits in the binary string is not multiple of 128 then*

- *Check the last bit of the string; if last bit of binary string ends up with 0 then does padding with 1's else do padding with 0's until binary string becomes multiple of 128. // Padding is performed.*

Step 5: *Division of Binary string into blocks of 128-bits.*

Step 6: *Generation of one 128-bit Key which is the initial key & then generates the list of keys using that initial key. And the number of keys will be the same as the number of 128- bit blocks produced in step 5.*

- *Generation of a random prime number p (it remains the same) and then right rotation of initial key by p times; resulting key is the next key for next block in the list of blocks.*
- *Repeat this process of rotating the bits of the key by prime number p times to generate the next key until a number of keys are the same as the number of blocks produced in step 5.*

Step 7: *Perform X-OR operations on each block in the list of blocks in step 5 with their corresponding key given in the list of keys in step 6.*

Step 8: *Generation of random sequence list of length as the number of blocks in the list of blocks, which contains the sequence numbers for rearrangement of blocks.*

Step 9: *Rearrangement of all the resulting blocks after X-OR operation, according to the sequence numbers given in the sequence list.*

Step 10: *Merge all the blocks into a single binary string.*

Step 11: *Divide binary string into 8-bit binary number blocks.*

Step 12: *Convert binary numbers to corresponding decimal numbers (ASCII values).*

Step13: *Convert decimal numbers (ASCII values) to their corresponding ASCII Character Codes (Obfuscated Text).*

B. Implementation of Proposed Obfuscation Technique with sample data

Plaintext: central university of Punjab, Bathinda

Step 1*: Character (Plain Text) to ASCII number Conversion & ASCII to Binary Conversion.*

Binary string is :

01100011011001010110111001110100011100100110000101101100001000000111010101101110011010010111011001100101011100100111001101101001011101000111100100100000011011110110011000100000010100000111010101101110011010100110000101100010001011000010000001000010011000010111010001101000011010010110111001100100 0110000 1

DOI: 10.35940/ijrte.C5653.098319
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Step 2: *Check whether the number of bits in the binary string is a multiple of 128 or not.*

Length of binary string is: 304, which is not a multiple of 128

Step 3: *If the number of bits in the binary string is already multiple of 128 then go to step 5.*

Step 4: *Number of bits in the binary string is not multiple of 128 then perform padding.*

String after padding :

011000110110010101101110011101000111001001100001
011011000010000001110101011011100110100101110110
011001010111001001110011011010010111010001111001
001000000110111101100110001000000101000001110101
011011100110101001100001011000100010110000100000
010000100110000101110100011010000110100101101110
011001000110000100000000000000000000000000000000
000000000000000000000000000000000000000000000000

Step 5: *Division of Binary string into blocks of 128-bits.*

List of blocks:

['01100011011001010110111001110100011100100110000
1011011000010000001110101011011100110100101110110
0011001010111001001110011011010010111010001111001',
'0111010001111001001000000110111101100110001000000
0101000001110101011011100110101001100001011000100
0010110000100000010000100110000101110100011010000
1101001',
'0111010001101000011010010110111001100100011000010
0000000000000000000000000000000000000000000000000
00000000000000000000000000000000']

Step 6: *Generation of one 128-bit Key, random prime number and list of keys using random prime number.*

Initial random 128-bit key in binary:

['100100111011100101110000001110001110010100101011
001000000110111101100010001101010101110010111100
0101100010001011000100001111000001']

Random prime number: 67

Right rotate the block of key by 67 bits

List of keys:

['100100111011100101110000001110001110010100101011
001000000110111101100010001101010101110010111100
0101100010001011000100001111000001',
'101100010001101010101110010111100010110001000101
100010000111100000011001001110111001011100000111
0001110010100101011001000000110111',
'000001100100111011100101110000001110001110010100
1010110010000001101111011000100011010101011100101
1110001011000100010110001000011111']

Step 7: *Perform X-OR operations on each block in the list of blocks with their corresponding key given in a list of keys.*

List of blocks after x-or operations:

['111100001101110000001111000000010110111000001101
1100101101100111011111110110111011000110111000001
11000000111010111100011000010101000',
'110001010110001110001110001100010100101001100101
110110000000011010101110000011101010011110110110
0000101010110101010001010010101110',
'011100100010011010001100101011111010001101001000
010110010000011011111011000100011010101011100101
11100010110001000101100010000111']

Step 8: *Generation of random sequence list of length as the number of blocks in the list of blocks, which contains the sequence numbers for rearrangement of blocks.*

Sequence list is: [0, 2, 1]

Step 9: *Rearrangement of all the resulting blocks after X-OR operation, according to the sequence numbers given in the sequence list.*

Sequence list is: [0, 2, 1]: place the 0th block at index 0 then 2nd block at index 1 and 1st block at index 2 in a new list of blocks.

Rearranged list is:

['111100001101110000001111000000010110111000001101
1100101101100111011111110110111011000110111000001
11000000111010111100011000010101000',
'011100100010011010001100101011111010001101001000
010110010000011011111011000100011010101011100101
11100010110001000101100010000111',
'110001010110001110001110001100010100101001100101
110110000000011010101110000011101010011110110110
0000101010110101010001010010101110']

Step 10: *Merge all the blocks into a single binary string.*

Merged string is :

111100001101110000001111000000010110111000001101
110010110110011101111111011011101100011011100000
111000000111010111100011000010101000011100100010
011010001100101011111010001101001000010110010000
011011111011000100011010101011100101111000101100
010001011000100001111110001010110001110001110001
100010100101001100101110110000000011010101110000
011101010011110110110000010101011010101000101001
010110

Step 11: *Divide binary string into 8-bit binary number blocks.*

New block list is:

['11110000', '11011100', '00011110', '00000101', '10111000', '00110111', '00101101', '10011101', '11111101', '10111011', '00011011', '10000111', '00000111', '01011110', '00110000', '10101000', '01110010', '00100110', '10001100', '10101111', '10100011', '01001000', '01011001', '00000110', '11110110', '00100011', '01010101', '11001011', '11000101', '10001000', '10110001', '00001111', '11000101', '01100011', '10001110', '00110001', '01001010', '01100101', '11011000', '00001101', '01011100', '00011101', '01001111', '01101100', '00010101', '01101010', '10001010', '01010110']

Step 12: *Convert binary numbers to corresponding decimal numbers (ASCII values).*

Decimal list is:

[240, 220, 30, 5, 184, 55, 45, 157, 253, 187, 27, 135, 7, 94, 48, 168, 114, 38, 140, 175, 163, 72, 89, 6, 246, 35, 85, 203, 197, 136, 177, 15, 197, 99, 142, 49, 74, 101, 216, 13, 92, 29, 79, 108, 21, 106, 138, 86]

Step13: *Convert decimal numbers (ASCII values) to their corresponding ASCII Character Codes (Obfuscated Text).*

Obfuscated text is:

ðÜ│¸7-ý»˛•ˆ0¨r&¯£HYö#UËÅ±xÅc1JeØ\Ol┴jV

Retrieval Number C5653098319/2019©BEIESP
DOI: 10.35940/ijrte.C5653.098319
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

VI. PROPOSED DE-OBFUSCATION ALGORITHM

The operations to perform de-obfuscation on binary data are as follows:

Step 1: *Obfuscated text to ASCII Conversion and ASCII to Binary conversion.*

Step 2: *Merge all the 8-bits binary numbers into a single binary string.*

Step 3: *Divide the Binary String into blocks of 128-bits.*

Step 4: *Rearrange all the resulting blocks according to the sequence numbers in the given sequence list as in step 8 of the obfuscation process.*

Step 5: *Perform X-OR operations on each block in the list of blocks as in step 4, with their corresponding key given in the list of the key as in step 6 of the obfuscation process.*

Step 6: *Merge all the blocks into a single binary string.*

Step 7: *Remove all the padding bits from binary string to get the original binary string.*

Step 8: *Divide the original binary string into 8-bit binary number blocks.*

Step 9: *Convert binary numbers to corresponding decimal numbers.*

Step 10: *Convert decimal numbers to their corresponding Character Codes (Plaintext).*

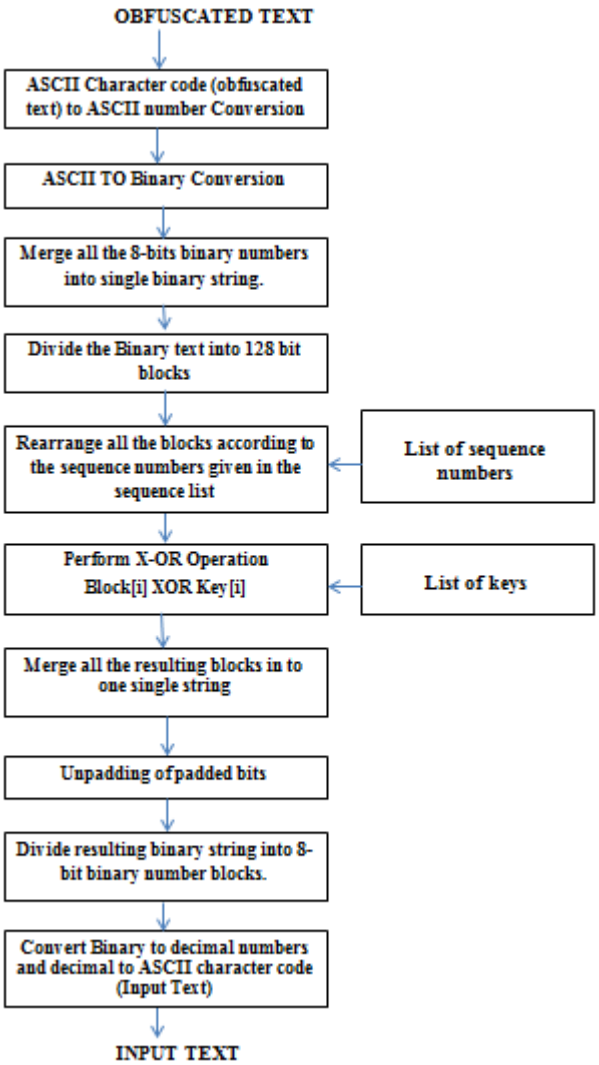

Fig.2. Flowchart of Proposed De-Obfuscation Technique

A. Implementation of Proposed De-Obfuscation Technique with sample data

Obfuscated text is:

ðÜ│¸7-ý»¸•ˆ0¨r&¯£HYö#UËÅ±xÅc1JeØ\Ol┴jV

List of keys:

['10010011101110010111000001110001110010100101011
00100000110111101100010001101010101110010111100
10110001000101100010000111100000 1',
'10110001000110101010111001011110001011000100010
10001000011110000011001001110111001011100000111
00111001010010101100100000110111',
'00000110010011101110010111000001110001110010100
10110010000011011110110001000110101010111001011
11000101100010001011000100001111']

Sequence list is: [0, 2, 1]

Step 1: *Obfuscated text to ASCII Conversion and ASCII to Binary conversion.*

Obfuscated text to ASCII conversion list is:

[240, 220, 30, 5, 184, 55, 45, 157, 253, 187, 27, 135, 7, 94, 48, 168, 114, 38, 140, 175, 163, 72, 89, 6, 246, 35, 85, 203, 197, 136, 177, 15, 197, 99, 142, 49, 74, 101, 216, 13, 92, 29, 79, 108, 21, 106, 138, 86]

ASCII to Binary conversion list is:

['11110000', '11011100', '00011110', '00000101', '10111000', '00110111', '00101101', '10011101', '11111101', '10111011', '00011011', '10000111', '00000111', '01011110', '00110000', '10101000', '01110010', '00100110', '10001100', '10101111', '10100011', '01001000', '01011001', '00000110', '11110110', '00100011', '01010101', '11001011', '11000101', '10001000', '10110001', '00001111', '11000101', '01100011', '10001110', '00110001', '01001010', '01100101', '11011000', '00001101', '01011100', '00011101', '01001111', '01101100', '00010101', '01101010', '10001010', '01010110']

Step 2: *Merge all the 8-bits binary strings into single binary string.*

The merged string is:

11110000110111000001111000000101101110000001101110010110011101111111011011101100011011100000111000001110101111000110000101010000111001000100110100011001010111110100011010010000101100100000110111101100010001000111010111000111000111000110001010010011001011101100000001101010111000000111010100111101101100000101010110101010100010100101010110

Step 3: *Divide the Binary String into blocks of 128-bits.*

List of blocks:

['11110000110111000001111000000101101110000001101110010110011101111111011011101100011011100000111000001110101111000110000101010000',
'01110010001001101000110010101111101000110100100001011001000001101111011000100011010101011100101111000101100010001011000100001111',
'11000101011000111000111000110001010010100110010111011000000011010101110000011101010011101101100000101010110101010100010100101010110']

Step 4: *Rearrange all the resulting blocks according to the sequence numbers in the given sequence list as in step 8 of the obfuscation process.*

Sequence list is: [0, 2, 1]: place the 0th block at index 0 then 1st block at index 2 and 2nd block at index 1 in a new list of blocks.

List of blocks after rearrangement is:

['11110000110111000001111000000101101110000011011
10010110110011101111111011011101100011011100001 1
10000011101011110001100001010100 0',
'11000101011000111000111000110001010010100110010 1
11011000000011010101110000011101010011110110110 0
00010101011010101000101001010110',
'01110010001001101000110010101111101000110100100 0
01011001000001101111011000100011010101011100101 1
11000101100010001011000100001111']

Step 5: *Perform X-OR operations on each block in the list of blocks as in step 4, with their corresponding key given in the list of keys.*

List after x-or operations with key:

['01100011011001010110111001110100011100100110000
10110110000100000011101010110111001101001011101 1
00110010101110010011100110110100101',
'01110100011110010010000001101111011001100010000 0
01010000011101010110111001101010011000010110001 0
00101100001000000100001001100001',
'01110100011010000110100101101110011001000110000 1
00000000000000000000000000000000000000000000000 0
00000000000000000000000000000000']

Step 6: *Merge all the blocks into a single binary string.*

Merged string is :

01100011011001010110111001110100011100100110000 1
01101100001000000111010101101110011010010111011 0
01100101011100100111001101101001011101000111100 1
00100000011011110110011000100000010100000111010 1
01101110011010010110000101100010001011000010000 0
01000010011000010111010001101000011010010110111 0
01100100011000010000000000000000000000000000000 0
00000000000000000000000000000000000000000000000 0

Step 7: *Remove all the padding bits from binary string to get the original binary string.*

Unpadded string is:

01100011011001010110111001110100011100100110000 1
01101100001000000111010101101110011010010111011 0
01100101011100100111001101101001011101000111100 1
00100000011011110110011000100000010100000111010 1
01101110011010010110000101100010001011000010000 0
01000010011000010111010001101000011010010110111 0
0110010001100001

Step 8: *Divide the original binary string into 8-bit binary number blocks.*

List of 8-bit blocks: ['01100011', '01100101', '01101110', '01110100', '01110010', '01100001', '01101100', '00100000', '01110101', '01101110', '01101001', '01110110', '01100101', '01110010', '01110011', '01101001', '01110100', '01111001', '00100000', '01101111', '01100110', '00100000', '01010000', '01110101', '01101110', '01101010', '01100001', '01100010', '00101100', '00100000', '01000010', '01100001', '01110100', '01101000', '01101001', '01101110', '01100100', '01100001']

Step 9: *Convert binary numbers to corresponding decimal numbers (ASCII values).*

Decimal list is: [99, 101, 110, 116, 114, 97, 108, 32, 117, 110, 105, 118, 101, 114, 115, 105, 116, 121, 32, 111, 102, 32, 80, 117, 110, 106, 97, 98, 44, 32, 66, 97, 116, 104, 105, 110, 100, 97]

Step 10: *Convert decimal numbers (ASCII values) to their corresponding ASCII Character Codes (Plaintext).*

Text list is : ['c', 'e', 'n', 't', 'r', 'a', 'l', ' ', 'u', 'n', 'i', 'v', 'e', 'r', 's', 'i', 't', 'y', ' ', 'o', 'f', ' ', 'P', 'u', 'n', 'j', 'a', 'b', ',', ' ', 'B', 'a', 't', 'h', 'i', 'n', 'd', 'a']

De-obfuscated text is: central university of Punjab, Bathinda

VII. EXPERIMENTAL RESULTS AND COMPARISONS

The Existing and Proposed algorithms are implemented using Python 3.6.3(64-bit). The obfuscation and de-obfuscation is done on the user's machine. The results are compared and analyzed with the existing technique on the basis of below-mentioned parameters in Table I:

Table I: Parameters to be evaluated

Obfuscation Technique	Inputs	Parameters
Existing proposed	Input Size (in Bytes) • 519 • 1,019 • 2,038 • 4,076 • 6,114 • 8,152 10,190	Obfuscation Time De-Obfuscation Time
	Different Input Texts • Text1 • Text2 • Text3 • Text4 • Text 5	Avalanche Effect

Above mentioned parameters are described as follows:

Obfuscation time: It is the time consumed by the obfuscation algorithm to convert plain text to obfuscated text [2].

De-Obfuscation time: It is the time consumed by the de-obfuscation algorithm to convert obfuscated text to the plain text [2].

Avalanche effect: Avalanche effect is a powerful factor for measuring the security of a cryptographic algorithm [11]. It is defined as a small change in either the plaintext or the key should generate a significant change in the ciphertext. A one bit change in the key or one bit change in the plaintext should yield a change in many bits of the ciphertext. More changes in the bit positions denote more randomness and difficulty in deciphering the text, and therefore the security level is more.

A. Experiment: To calculate the obfuscation time of existing and proposed technique.

Table II represents the different obfuscation times taken by the existing and proposed obfuscation techniques. It signifies that for small input data size the difference between obfuscation time for both the techniques is very small but as the size of the input data increases, the difference between obfuscation time taken by both the techniques is considerably more, and proposed obfuscation technique consumes very less time in obfuscation process as compared to existing technique.

Table II: Comparison of Existing and Proposed Techniques in terms of Obfuscation Time

Input size (in Bytes)	Obfuscation Time Existing (in seconds)	Obfuscation Time Proposed (in seconds)
519	0.1247	0.0936
1,019	0.1404	0.0936
2,038	0.1560	0.1092
4,076	0.3900	0.3588
6,114	0.9672	0.4836
8,152	1.2480	0.5304
10,190	1.7004	0.5772

Analysis of Obfuscation Time:

Figure 3 shows the obfuscation time taken by both the techniques Existing and Proposed.

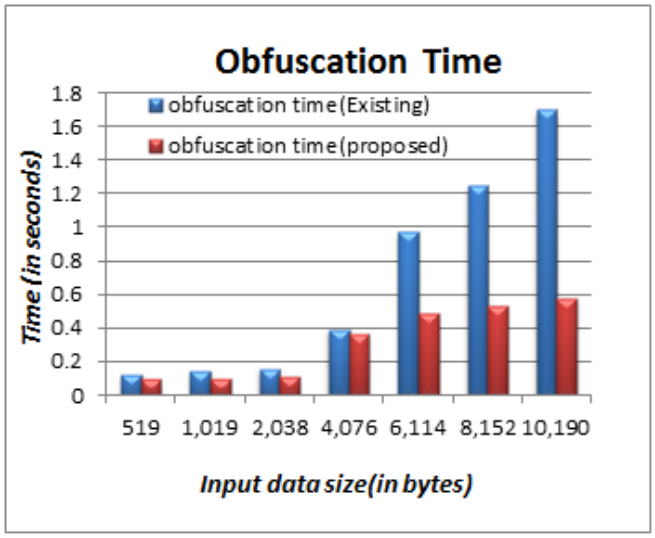

Fig.3. Obfuscation time of Existing and Proposed

The considerable change in obfuscation time of proposed technique is due to the division of long binary data into 128-bits blocks and then performs the operations on them which will reduce the time consumption by proposed obfuscation technique. In the existing technique, operations are performed on the complete long binary data at a time which will raise the time consumption by existing obfuscation technique.

B. Experiment: To calculate the de-obfuscation time of existing and proposed technique.

Table III represents the different De-obfuscation times taken by the existing and proposed obfuscation techniques, and it signifies that De-obfuscation time taken by the proposed approach is considerably less as compared to the existing approach.

Table III: Comparison of Existing and Proposed Techniques in terms of De-Obfuscation Time

Input size (in Bytes)	De-Obfuscation Time Existing (in seconds)	De-Obfuscation Time Proposed (in seconds)
519	0.0155	0.0155
1,019	0.0780	0.0467
2,038	0.1404	0.0936
4,076	0.2184	0.2184
6,114	0.4524	0.3120
8,152	0.7332	0.3900
10,190	0.8580	0.4368

Analysis of De-Obfuscation Time:

Figure 4 shows the De-obfuscation time taken by both the techniques (Existing and Proposed).

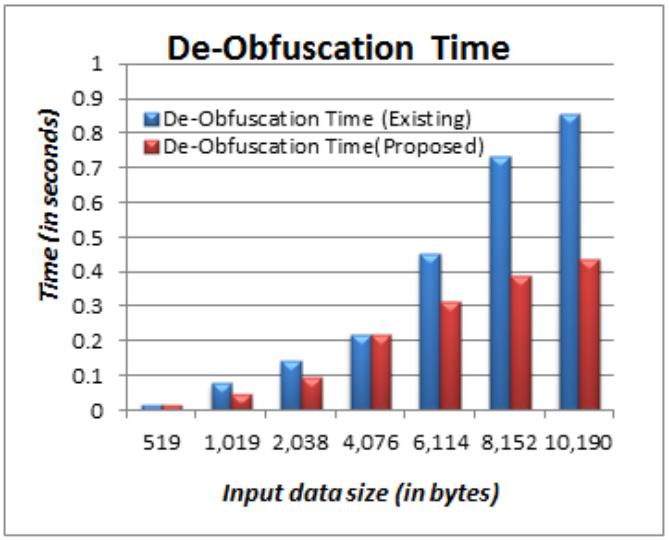

Fig.4. De-Obfuscation time of Existing and Proposed

Proposed de-obfuscation technique consumes very less time in the de-obfuscation process as compared to the existing technique, and the reason is the same as in obfuscation time reduction. We can also observe that De-obfuscation Time of the existing and proposed technique is less as compared to Obfuscation Time of existing and proposed technique because we haven't generated the list of keys and list of sequence numbers again for de-obfuscation. We used the same list of keys and list of sequence numbers for de-obfuscation which are used in the obfuscation process.

C. Experiment: To calculate the Avalanche Effect of the existing and proposed technique.

To calculate the Avalanche Effect, first, we obfuscate the input data as it is and gets the obfuscated text (O1). And then modify one bit in the input data, obfuscate it and get the obfuscated text (O2). Now we compare the two obfuscated texts and calculate the number of bits flipped in the second obfuscated text. More the number of flipped bits more will be the avalanche effect.

$$Avalanche\ effect = \frac{flipped\ bits\ in\ obfuscated\ text(O2)}{Number\ of\ bits\ in\ obfuscated\ text(O1)}$$ [12]

Different text inputs taken for the calculation of the avalanche effect are given below:

Text1: With the fast progression of digital data exchange in electronic way

Text2: information technology

Text3: central university of pun jab

Text4: cyber security

Text5: security

Table IV: Comparison of Existing and Proposed Techniques in terms of Avalanche Effect

Input Texts	Avalanche Effect Existing (in %)	Avalanche Effect Proposed (in %)
Text 1	0.73	51.40
Text 2	2.27	46.48
Text 3	0.89	49.60
Text 4	3.57	13.28
Text 5	1.56	50.78

Table IV indicates the difference between Avalanche Effect for both the techniques and proposed obfuscation technique shows very high Avalanche as compared to the existing technique.

Analysis of Avalanche Effect:

Figure 5 shows the Avalanche Effect of existing and proposed techniques.

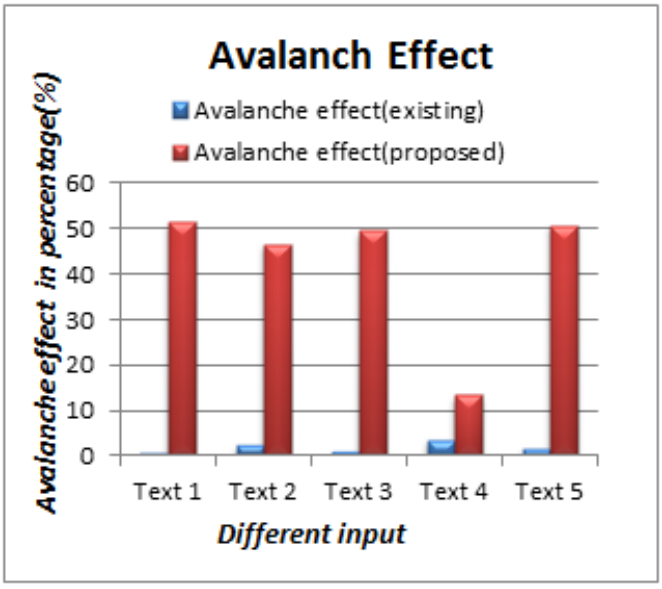

Fig.5. Avalanche Effect of Existing and Proposed Techniques

This extreme change in the avalanche effect of the proposed technique is due to the randomness generated in an algorithm by using different rearrangement of blocks. So, we can say that the proposed approach is more random because of that attacker can't observe the pattern in the obfuscated text and can't easily get the plaintext or original message.

VIII. CONCLUSIONS

In this paper, a new obfuscation approach is developed and implemented to improve the performance and security of the data at rest. This new approach has taken minimum time for obfuscation and de-obfuscation process as compared to the existing technique. This new approach also produced randomness in an algorithm which results in significant changes in obfuscated-texts it means that the proposed algorithm produced very high avalanche effect. Due to this factor attacker can't be able to observe the pattern in different obfuscated texts. In this way, proposed approach provides better security and the data is protected in the storage.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Er.Meenakshi, Assistant Professor, Department of Computer Science and Technology, Central University of Punjab for providing me their invaluable guidance and suggestions. I would specially like to thank her for always motivating me throughout my research work.

I would also like to thanks to all the Authors whose names I mentioned in my research paper, to provide the clarity of the concepts in the direction of my research topic and inspired me to do my research in the same area. I would like to express my deepest gratitude to my family, whose love and guidance are with me in whatever I pursue. Most importantly, I would like to thank my friends, who were of great support in deliberating over our problems and findings, as well as providing happy distraction to rest my mind outside of my research. This research work would have not been possible without their warm love, continued patience, and endless support.

REFERENCES

1. D. E. R. Denning, Cryptography, ADDISON-WESLEY PUBLISHING COMPANY , 1982.
2. S. A. Oli and L. Arockiam, "Enhanced Obfuscation Technique for Data Confidentiality in Public Cloud Storage," in *MATEC Web of Conferences*, 2016.
3. S. Pearson, Y. Shen and M. Bray, "A Privacy Manager for Cloud Computing," in *IEEE International Conference on cloud Computing*, Berlin, Heidelberg, 2009.
4. A. Rehman and M. Hussain, "Efficient Cloud Data Confidentiality for DaaS," *International Journal of Advanced Science and Technology*, pp. 355-359, October, 2011.
5. S. M. P. D. S. K. M. L. Arockiam, "Obfuscrypt: A Novel Confidentiality Technique for Cloud Storage," *International Journal of Computer Applications*, vol. 88, no. 1, pp. 17-21, February 2014.
6. L. Arockiam and S. Monikandan, "Data Security and Privacy in Cloud Storage using Hybrid Symmetric Encryption Algorithm," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 8, pp. 3064-3070, August 2013.

7. T. Nie, L. Zhou and M. Lu, "Power evaluation methods for data encryption algorithms," *IET Software IEEE*, vol. 8, no. 1, p. 12–18, February 2014.
8. C. . P. Dewangan and S. Agrawal, "A Novel Approach to Improve Avalanche Effect of AES Algorithm," *International Journal of Advanced Research in Computer Engineering & Technology*, vol. 1, no. 8, pp. 248-252, 2012.
9. S. Anbazhagan and K. Somasundaram, "Cloud computing Security through Symmetric Cipher model," *International Journal of Computer Science & Information Technology(IJCSIT)*, vol. 6, no. 3, pp. 57-66, June 2014.
10. L. Arockiam and S. Monikandan, "AROMO Security Framework to Enhance Security of Data in Public Cloud," *International Journal of Applied Engineering Research*, vol. 10, no. 9, pp. 6740-6746, 2015.
11. N. P. Singh and S. Rani, "Optimal Keyless Algorithm for Security," *International Journal of Computer Applications*, vol. 124, no. 10, pp. 28-32, 2015.
12. A. K. Mandal and A. Tiwari, "Analysis of Avalanche Effect in Plaintext of DES using Binary Codes," *International Journal of Emerging Trends & Technology in Computer Science*, vol. 1, no. 3, pp. 162-171, September – October 2012.

AUTHORS PROFILE

Tanuja has done her Master of Technology in Computer Science and Technology (Specialization in Cyber Security) from Central university of Punjab, Bathinda, India.

Meenakshi has done her Master of Engineering in Computer Science Engineering from Punjab Engineering College University of Technology, Chandigarh, India. She is currently working as Assistant Professor in Department of Computer Science and Technology, Central University of Punjab, Bathinda. Her main research work focuses on Information Security, Cryptography, and Computer Networks. She has more than 7 years of teaching experience.