A New PDAC (Parallel Encryption with Digit Arithmetic of Cover Text) Based Text Steganography Approach for Cloud Data Security

Mayanka Gaur, Manmohan Sharma
Department of Computer Science and Engineering, Mody University of Science and Technology, Lakshmangarh, Sikar (distt.), Rajasthan, India
Email: gaur.mac1305@gmail.com, manmohan.manu@gmail.com

***Abstract*---** Internet Computing provides dynamic virtualization, resource pools, services and high availability servers. With rapid growth of internet computing technology, there is a high demand for data storage security on cloud. In this paper we are presenting a useful new approach of text based steganography for cloud data security. In our approach, simple addition, subtraction and multiplication of digits of ASCII code of each character of cover text is done and these new generated numeric values are used to encrypt ASCII values of our plain text. Since, in our approach, there are three basic airthmetic operations that are performed on every character of cover text, therefore, after airthmetic calculations each and every character will generate three numeric values such that, each and every numeric value will encrypt two ASCII values of plain text parallely. One from beginning of array of ASCII values of plain text and another one from ending of the very same array. In our approach, one character of cover text hides at most six characters of plain text. Thus memory allocation problem for cover text and execution time both are reduced. Using parallelism performance of our approach is enhanced.

Keywords- *Information, Hiding, Cryptography, Steganography, Text Steganography, Arithmetic Operation, Mathematical calculations, Encryption, Parallelism.*

******

I. INTRODUCTION

Cloud computing become an IT fuzzword from past few years. Cloud computing accumulates all the computing resources and manages them by some software. There are billions of users using cloud services synchronously. Users need not to worry about how to buy the servers or softwares for a long time perspective instead they can directly use or buy computing resources from the cloud using internet. Users generally worry that the cloud computing providers can misuse their important data present on cloud. The only way is to use some encryption method to hide our plain text such that it cannot be figured out with intrusive eyes. Currently there are two ways of encryption of our plain text one is Cryptography and another one is Steganography. These two approaches are somewhat different from each other. Cryptography is used to hide the contents of the message using either symmetric or asymmetric encryption method whereas Steganography hides the existence of our primitive message using some cover media. Steganography is the art of writing message or data in such a way that no one else except the sender and destined recipient, presume the existence of the message.

Steganography is the science of writing hidden messages in such a way that no one apart from the sender and intended recipient even realizes there is a hidden message. The main goal of Steganography is to lurk the communication. Steganography transmits a message in some cover media i.e. text, image, audio or video files such that to effectively conceal the presence of message over communication channel. Due to some limitations in Cryptography method as the third party is always aware of ongoing communication, Steganography is used more for concealing the original message. British and US government banned the use of cryptography after arrival of 9/11 and from then onwards Steganography gained the importance. Steganography overcomes the limitation of Cryptography by hiding the message in an innocent looking object called cover media which is not identifiable by human eyes. Thus, we can say that the third party is not aware of hidden message in an ongoing communication. It totally conceals the fact that some important communication is going on.

Generally, there are four types of cover media i.e. image, audio, video and text. Among these, text file requires less memory storage so hiding message/data in text cover media and then storing it on cloud server will take less memory as compared to other cover media's. So, we are taking text cover media. Text medium is comparatively difficult with other cover media's as there is lack of available redundant information in text data. Message or data hidden using encryption in text cover media is referred to as text steganography. In this paper, we are presenting an overview of existing text steganography approaches and our proposed approach of Text Steganography. The main problem with many existing methods of Text Steganography is using large bytes of cover text for encryption and decryption for hiding small bytes of plain text and also it takes too much time in encryption and decryption. We introduce new approach based on Text Steganography which

easily hides plain text or original data in the cover text and also takes very less time for encryption.

This paper presents an efficient Text Steganography approach based on simple mathematical calculations and parallelism. We are doing addition, subtraction and multiplication of digits of ASCII code of each character of cover text and then these digits are used for encrypting our plain text in parallel. Thus performance is enhanced using parallelism. In this approach one character of cover text hides six characters of plain text thus memory required and execution time both will be less therefore performance is increased. This approach will generate three digits for one character of cover text in which one digit is used to encrypt two characters of plaintext parallely one from starting and another one from ending.

Our approach requires maximum n bytes of cover text for hiding 6n bytes of plain text since one character of cover text can hide six characters of plain text.

We are presenting some existing text steganography techniques. Then we will describe our proposed approach with it's implementation algorithm of encryption and decryption. We are describing our approach with an example for easy understanding. This method may also have some merits and demerits which we will try to remove in future.

II. RELATED WORK

The aim of Steganography is to hide our plain text or data through the communication channel using some cover media. It is necessary to use some other redundant data as a cover medium for the existing plain text/data. The probable media that can be used as a cover can be image, audio, text or a movie clip/video file. Out of these different cover media files, a text data less bytes and occupies lesser memory storage as compared to other media files [1]. Text steganography is used more in comparison to other media as it would encrypt more bytes of plaintext and can communicate more information with less storage and execution time. Text steganography is used more in comparison to other media as it would encrypt the plaintext and merge with cover text which is generating random character sequences. Text steganography is believed to be the trickiest due to deficiency of redundant information which is present in image, audio or a video file. Text data storage requires less memory and it's faster to read as well as easier communication makes it preferable technique to other types of Steganography methods [11]. Text steganography can be broadly classified into three types : Format based, Random Statistical generation and Linguistic methods.

A. Format Based Methods

It changes the text content to hide secret information. This method involves changing color, size or type of font and adding white spaces to the text content. This method has some demerits. If suppose third party opens and reads the text stegano file in word processor then it will have misspelled words and white spaces included in it will be removed. Similarly changing size, font type, font color are normal changes(are not much effective in hiding information) and are easily identifiable by intrusive eyes [3]. One major fault of this method is that if suppose third party also has original files then it would be very easy to get original text by simply comparing original file with stegano file.

B. Random and Statistical Generation

In this method first generating random sequence of characters known as cover text [4]. On this generated random cover text statistical properties are applied. This method is based on character sequences and word sequences. Firstly hiding information within character sequences appears as a random sequence of characters or like a random cover text thus random generation. Secondly applying statistical properties in order to generate "words" (without lexical value) having similar statistical properties as those of real words in given language is what statistical generation method is.

C. Linguistic Steganography

In this method, focus is mainly on linguistic properties of random generated text. Syntax of text can be used as a linguistic structure to hide secret message or information [6]. Sometimes spaces are used as a linguistic structure to hide information.

Existing Approaches:

In this sub-section, we present some of the popular existing approaches of text steganography.

A. Open Space Method

In this method, white spaces are used to hide secret message. Adding white spaces in the text is very common or general and thus, cannot be recognizable through intrusive eyes. This method does not change the meaning of text document. White spaces can be added to hide secret message in three ways namely Inter-Sentence Space method, End-of-line Space method and Inter-word Space method. In Inter-Sentence Space method as the name suggests, space is added between characters (i.e. after every terminating character) to hide message. In End-of-line Space method, white space is added between lines (i.e. at the end of every line) to hide message. In Inter-word Space method, space is added between words (i.e. after every word and before every word) to hide secret data. Open Space method has one fault if third party or intruder rewrites the text then the secret data will be destroyed. So that is why it is not used.

B. Line Shift Method

In this, as the name suggests, text lines are shifted vertically to some degree. To hide 0 bit line is shifted down vertically to some degree. Similarly, to hide 1 bit line is shifted upwards vertically to some degree. Shifting lines up or down to some degree would not be identifiable by intruder. On the basis of distance between centroid of marked line and it's control lines, whether to shift line upwards or downwards is chosen. This method has some fault that distance can be easily observed through human eyes or measured using some special tools or using OMR sheet.

C. Word Shift Method

It is similar to line shift the only difference is that in this method the text words are shifted horizontally either from left side or from right side. White spaces are automatically added between words in the document while justifying the text. To casual readers this method is not easily identifiable as variable spaces between words are very common. In this method, a bit is hided as a white space by horizontally shifting a word within a line. This method has a fault that for decoding algorithm original document is required and if intruder or third party gets this original document then this method is easily noticeable to him or her and he or she will be able to get the hidden message by making comparison of original document with the modified one. Another defect of this method is that rewriting of document will remove the hidden data. This method is very time consuming.

D. Acronyms Method

In this method, abbreviations or full form of words are used to hide secret bit. To hide bit 1 abbreviation of word is used and to hide bit 0 full form of word is used.

E. Syntactic Method

In this method, syntax of text document is used to hide bits 0 and 1 for example full stop (.), semi-colon (;), comma (,), etc are used to hide bits. So, by adding extra punctuation marks to the document, bits can be made hidden [9]. We can use semicolon (;) or any other punctuation mark to hide bit 1 and similarly full stop (.) or any other punctuation mark to hide bit 0. But this method has one fault that adding extra punctuation marks to the document will change the entire meaning of the document and hence the document will be of no use.

F. Semantic Method

In this method, synonyms of actual words are used to hide the bits. To hide bit 1, actual word is used as it is whereas to hide bit 0, synonym of that very actual word is used. In case of rewriting of data or modifying text format the hidden or secret message can still be retrieved back. So, this method can be considered as best among other methods but due to replacement by synonym word, problem can also occur.

G. Feature Coding Method

As the name specifies, features of text are modified to hide secret data. There are two ways to implement this method. One way is to change the certain attributes of text like changing font color, size, type, etc to hide secret data [8]. Another way is either by increasing or decreasing height of text characters or by replacing dots or points with characters 'i' and 'j'. One fault in this method is that if rewriting of text is done or if OCR program is used then the hidden secret message would be ruined.

H. Persian/Arabic Method

In this method [8], points of characters in text document file are used to hide secret bits. To hide bit 0, location of point is not changed whereas to hide bit 1 point is shifted upwards.

I. Quadruple Categorization Method

In this characters are divided into groups on the basis of their features like round curves, one or more straight lines, a straight line in middle, diagonal straight lines, etc. Thus at a time two bits can be hidden in one character.

J. Capital Alphabets Shape Encoding (CASE)Method

In this method [10], two steps are taken, firstly encoding all characters of secret data on the basis of shape of alphabets. Secondly, hiding this message with random cover text. This will reduce the memory required for storage. Therefore, this method is considered as best as compared to above approaches. In this method, if re-formatting or re-typing of text is done then the secret message will not be lost and it's meaning will also remain same. Most of above existing approaches uses random cover text to hide original message.

K. Encryption with Cover Text and Reordering (ECR) Method

In this method [12], to hide n bytes of plain text, n bytes of random cover text are required. In this, plain text is encrypted with cover text using simple x-or operation and then this generated encipher text is merged with cover text using 8-bit random key re-ordering method. 8-bit random key has four 1's and four 0's at random position. Cipher text is generated by placing encipher text whenever 1 is found in random key and placing cover text whenever 0 is found in random key. Since in this method, only one operation is required for encryption i.e. x-or which is very fast in computation. So, time overhead is less but it has one fault i.e. for large plain text, large cover text is required.

L. Parallel Encryption with Digit Arithmetic of Cover Text (PDAC) Method

In this method [7], three operations are used for encryption of plain text (which are addition, subtraction and x-or). Addition and subtraction are applied on digits of cover text and then this generated new cover text is x-ored parallely with plain text for encryption process. In this method, one character of cover text hides four characters of plain text. Since x-oring is done parallely so time overhead and memory requirement for storage of cover text both are very less. This method is good but in it, to hide plain text of more than four characters at least two characters of cover text are required.

Our idea is to reduce number of bytes of cover text so that the memory occupied by it can be utilized at somewhere else. So this is how we came with our New approach.

In all above approaches, to hide one character of plain text minimum one character of cover text is required. But our approach is on top of above all these approaches as in our approach one character of cover text is required to hide six characters of plain text. Thus, more number of information is hided using small cover text. Also, memory requirement for storage of random cover text and time required for execution of this approach both are very less. Therefore, this approach reduces bytes of cover text that are required to encrypt (hide) message or information. In our approach, characters of cover text is reduced and characters of plain text that cover text hides is increased. The very next section describes our approach along with it's implementation.

III. PROPOSED APPROACH

In our approach, we are performing simple addition, subtraction and multiplication of digits of ASCII code of each character of cover text. This process would generate three decimal values for each character of cover text, one from addition, one from subtraction and one from multiplication operation. Due to subtraction operation there are chances of occurrence of negative decimal values. So, to avoid negative values, add 10 to each decimal values. These new values generated are used for encryption of plain text using X-OR operation. But before encryption, generating equivalent ASCII code for our plain text. Next performing encryption by parallely x-oring ASCII of plain text and ASCII of generated values from cover text. Since in our approach there are three basic airthmetic operation performed on every character of cover text, so therefore, every character will generate three numeric values each such that each and every numeric value will encrypt two ASCII values of plain text parallely. One encryption is done from beginning of array of ASCII values of plain text and another one is done from the end of array of ASCII values of plain text. Therefore, these three numeric values generated from one character of cover text are encrypting total six ASCII values of plain text. So, therefore, we can say that one character of cover text is able to hide at most six characters of plain text. Thus reducing memory allocation problem for cover text and also reducing the time required for executing this approach by making use of parallelism technique. Since, we know that basic airthmetic operations and X-OR operation are fast to compute and we had also used these so we can say that our approach executes faster than any other existing approach of text based steganography. So, if we want to send n bytes of plain text through communication channel or if we want to store our important commercial n bytes of plain text on public cloud then we would require maximum of n/6 bytes of cover text.

IV. IMPLEMENTATION

We have developed two algorithms for hiding and retrieving our plaintext message.

A. Steps for hiding plain text message/data

1) Let your plain text is of n bytes.

2) Now, n bytes of plain text requires ceiling (n/6) bytes of cover text.

3) Generate random cover text on the basis of step 2.

4) Get ASCII code for each character of cover text.

5) Perform addition, subtraction and multiplication of digits of ASCII code of each character of cover text.

6) Due to subtraction operation, there are chances of occurrence of negative decimal values, so to avoid negative decimal values, add 10 to each of the resulting decimal values.

7) After this we will have three digits/characters of cover text.

8) Get ASCII code for each character of the plain text.

9) Now, perform X-OR operation between ASCII codes of plain text and ASCII codes of cover text parallely.

10) One character of cover text will encrypt two characters of plain text parallely i.e. one from starting and another one from ending.

11) Result of this X-OR operation will have equal number of characters as that in our original plain text/message/data and is known as encipher text.

12) Hiding/merging this obtained encipher text with random cover text. From later example it would be clear how we are merging/hiding it.

 ISSN: 2321-8169
1344 - 1352

13) Now after this, our final encipher text is ready but it is in ASCII codes.

14) Get equivalent characters from this ASCII codes.

15) Now, after this we are ready with our final encipher text to send to destination through communication channel or to store it on the cloud sever.

B. Steps for getting back our original plain text message/data

1) Get final encipher text in an array format. This will be array containing sequence of characters.

2) Get equivalent ASCII codes from final encipher text in an array format. This will be array containing ASCII code

3) Now, fetch cover text from this array of encipher text using modulo 7 operation i.e. fetching every character at position 0, 7, 14, 21, 28, ... (i.e. 0 and multiple of 7) to get our random cover text.

4) Next, perform addition, subtraction and multiplication of digits of ASCII code of each character of cover text.

5) To avoid negative decimal values due to subtraction operation, add 10 to every resulting decimal values. After this we will have three digits/characters of cover text which is our final cover text that would be used for x-or operation later.

6) After fetching out cover text, remaining text is known as encipher text or encrypted message which is then x-ored with the final cover text. This is explained clearly with later example.

7) Result of X-OR operation will have equal number of characters as that was in our encipher text and this result is our original plain text message/data. But it is in ASCII code.

8) Get equivalent sequence of characters from sequence of ASCII code.

9) After step 8, we have our original message/data.

C. Pseudo code for message/data hiding

```
Procedure Hiding (String msg)

msg -> contains plain text

k and j are integer variables.

len -> message length

len_ct -> contains length of cover text

temp, sum, sub and mul are integer variables.

encrypted_mess -> contains message/data after encryption.

BEGIN

k=0, j=0;

SET len to msg.length();

SET len_ct to ceil ( len/6 );

FOR i=0 to i<(3 * len_ct)

temp= ASCII ( cover_text[j] );

sum= ( temp/10 + (temp modulo 10) ) +10 ;

sub= ( temp/10 - ( temp modulo 10 ) ) +10 ;

mul= (temp/10 * ( temp modulo 10 ) ) + 10 ;

IF (op is sum) THEN

BEGIN

        SET encrypted_mess[k] to cover_text[j] ;

        k= k+1;

        SET encrypted_mess[k] to char(xor(sum, ascii( msg[i])));

        k=k+1;

        IF ( [len-i-1] >= len/2 ) THEN

                SET encrypted_mess[k] to char( xor(sum, ascii ( msg[len-i-1])));

                k=k+1;

        END IF

END IF

ELSE IF (op is sub) THEN

BEGIN

        SET encrypted_mess[k] to char(xor(sub, ascii( msg[i])));

        k=k+1;

        IF ( [len-i-1] >= len/2 ) THEN

                SET encrypted_mess[k] to char(xor(sub, ascii(msg[len-i-1])));

                k=k+1;
```

 ISSN: 2321-8169
1344 - 1352

```
        END IF
END ELSE IF
ELSE (op is mul) THEN
BEGIN
        SET encrypted_mess[k] to char(xor(mul, ascii(msg[i])));
        k=k+1;
        IF ( [len-i-1] >= len/2 ) THEN
                SET encrypted_mess[k] to char(xor (mul, ascii (msg[len-i-1])));
                k=k+1;
        END IF
END ELSE
END FOR LOOP
j=j+1; // for loop now executes for next digit of cover text
```

D. Pseudo code for message/data retrieval

```
Procedure Retrieval (String encrypted_msg)
encrypted_msg -> contains recieved encrypted message
cover_text -> contains fetched cover_text
len_ct -> contains length of fetched cover text
len -> contains length of received encrypted message
i, j, k are integer variables
temp, sum, sub and mul are integer variables.
BEGIN
j=0;
SET len to encrypted_msg.length()
SET olen to len-ceil(len/7)
Fetching of cover text from received encrypted message
FOR i=0 to i < len
Begin
IF i%7==0 THEN
        cover_text[j]=encrypted_msg[i];
        j=j+1;
END IF
ELSE
        encrypted_message[j]=encrypted_msg[i];
        j=j+1;
END ELSE
End for loop
Fetching original message from encrypted message making use of fetched cover text
FOR i=0 to i<(3 * len_ct)
temp= ASCII ( cover_text[j] );
sum= ( temp/10 + (temp modulo 10) ) +10 ;
sub= ( temp/10 - ( temp modulo 10 ) ) +10 ;
mul= (temp/10 * ( temp modulo 10 ) ) + 10 ;
IF (op is sum) THEN
BEGIN
        SET org_message[k] to char(xor(sum, ascii( encrypted_message[i])));
        i=i+1;
        IF ( [olen-k-1] > k ) THEN
                SET org_message[olen-k-1] to char( xor(sum, ascii ( encrypted_message[i])));
                k=k+1;
        END IF
END IF
ELSE IF (op is sub) THEN
BEGIN
        i=i+1;
        SET org_message[k] to char(xor(sub, ascii( encrypted_message[i])));
        i=i+1;
```

 ISSN: 2321-8169
1344 - 1352

IF ([olen-k-1] > k) THEN

SET org_message[olen-k-1] to char(xor(sub, ascii(encrypted_message[i])));

k=k+1;

END IF

END ELSE IF

ELSE (op is mul) THEN

BEGIN

i=i+1;

SET org_message[k] to char(xor(mul, ascii(encrypted_message[i])));

i=i+1;

IF ([olen-k-1] > k) THEN

SET org_message[olen-k-1] to char(xor (mul, ascii (encrypted_message[i])));

k=k+1;

END IF

END ELSE

END FOR LOOP

j=j+1; // for loop now executes for next digit of cover text

END

E. Explanation for Encryption with example

1) Let us assume that our original plain text (original data) is "Portal" which is shown in Fig. 1

Fig. 1 Plain Text

2) Generate random cover text on the basis of ceiling(n/6) where n is the number of characters in the plain text. So for our plain text only one character of cover text is required. Let us assume that our cover text is "A" as shown in Fig. 2

Fig. 2 Random Cover Text

3) Get equivalent ASCII code for each character of cover text. For our random cover text equivalent ASCII code would be as shown in Fig. 3

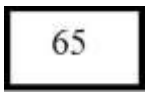

Fig. 3 ASCII of cover text

4) Perform addition, subtraction and multiplication of digits of ASCII code of each character of cover text. Due to subtraction operation, there are chances of occurrence of negative decimal values, so to avoid negative decimal values, add 10 to each of the resulting decimal values as shown in Fig. 4. After this we will have three digits/characters of cover text as shown in Fig. 5

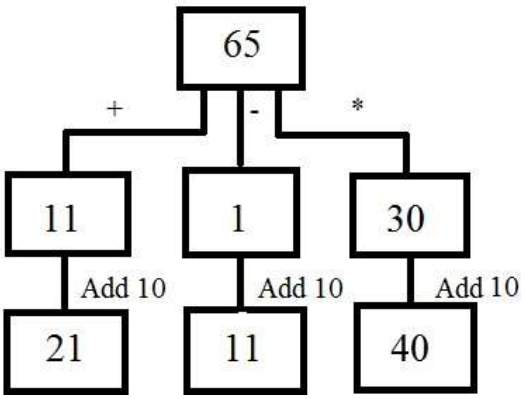

Fig. 4 Performing mathematical calculations on digits of each character of cover text

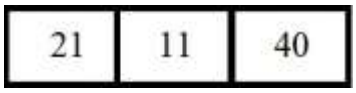

Fig. 5 Final cover text that would be used for xor operation later

5) Get equivalent ASCII code of plain text. For our plain text equivalent ASCII code would be as shown in Fig. 6

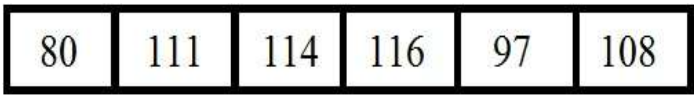

Fig. 6 ASCII code of Plain text

6) Perform X-OR operation between ASCII codes of plain text and ASCII codes of cover text parallely. One character of cover text will encrypt two characters of plain text parallely i.e. one from starting and another one from ending. Result of this X-OR operation will have equal number of characters as that in our original plain text/message/data and is known as encipher text as shown in Fig. 7

 ISSN: 2321-8169
1344 - 1352

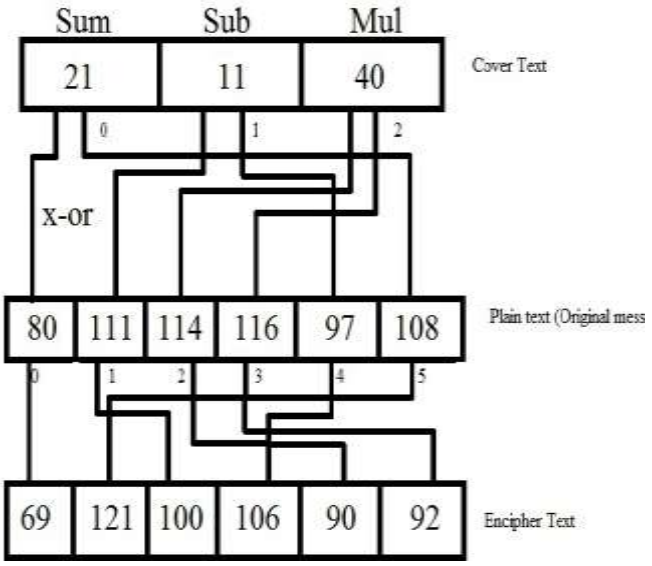

Fig. 7 Performing X-Or operation between cover text and plain text

7) Hiding/merging this obtained encipher text with random cover text as shown in Fig. 8

65	69	121	100	106	90	92

Fig. 8 Merging obtained encipher text with our random cover text

8) Get equivalent characters from this ASCII codes as shown in Fig. 9 which is our final encipher text/encrypted message ready to be sent through communication channel to destination or to be stored on cloud server.

A	E	y	d	j	Z	\

Fig. 9 Final encipher text/encrypted message

F. Explanation for Decryption with example

1) Get final encipher text in a string array format as shown in Fig. 10

A	E	y	d	j	Z	\

Fig. 10 Encipher message

2) Get equivalent ASCII code of this encipher message or encrypted message as shown in Fig. 11

65	69	121	100	106	90	92

Fig. 11 ASCII code of encipher message

3) Now, fetch cover text from this array of encipher text using modulo 7 operation i.e. fetching every character at position 0, 7, 14, 21, 28, ... (i.e. 0 and multiple of 7) to get our random cover text as shown in Fig. 12

65

Fig. 12 Fetched cover text

4) Next, perform addition, subtraction and multiplication of digits of ASCII code of each character of cover text. To avoid negative decimal values due to subtraction operation, add 10 to every resulting decimal values as shown in Fig. 13.

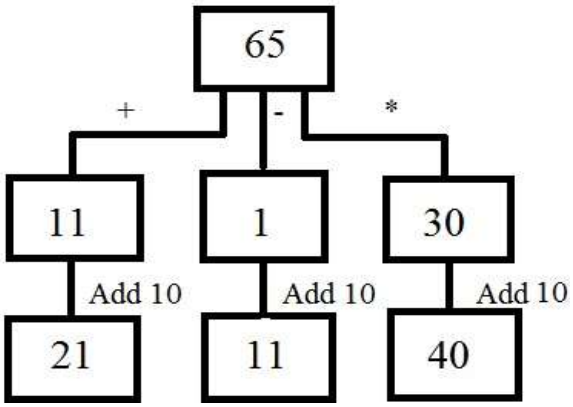

Fig. 13 Performing mathematical calculations on digits of ASCII code of each character of cover text

5) Final cover text that would be used for x-or operation later is obtained after performing mathematical calculations on each character of random cover text (as we have done in above step). Our final cover text is as shown in Fig. 14

21	11	40

Fig. 14 ASCII code of final cover text

6) After fetching out cover text, remaining text is known as encipher text or encrypted message which is then x-ored with the final cover text. Perform X-OR operation between final cover text and encrypted message or encipher text parallely. One character of cover text will encrypt two characters of encrypted message or encipher text parallely i.e. one from starting and another one just next to it. Result of this X-OR operation will have equal number of characters as that in our encipher text(encrypted message) as shown in Fig. 15 and would be our original message but in ASCII codes.

 ISSN: 2321-8169
1344 - 1352

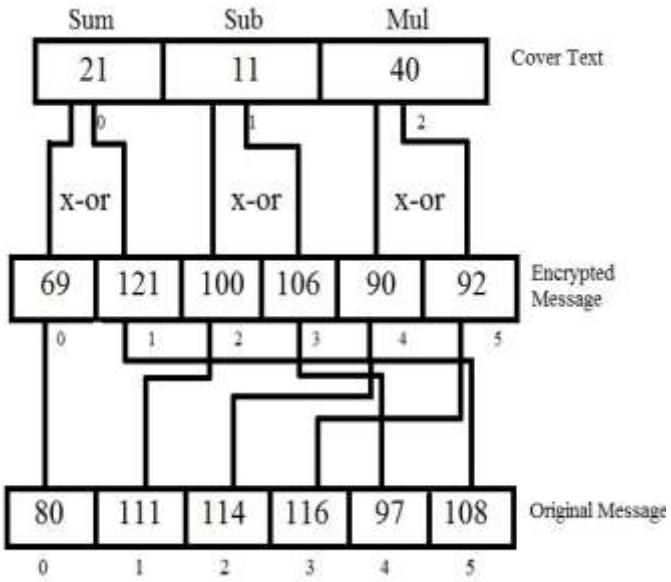

Fig. 15 Performing x-or operation between cover text and encipher text (encrypted message)

7) Get equivalent characters (as shown in Fig. 16) from ASCII codes of original message obtained from x-or operation in above step 6

P	o	r	t	a	l

Fig. 16 Original message/data/plain text

V. CONCLUSION

In this paper, we are presenting a useful new approach of text based steganography for cloud data security. In our approach we are performing simple addition, subtraction and multiplication of digits of ASCII code of each character of cover text. This process would generate three decimal values for each character of cover text, one from addition, one from subtraction and one from multiplication operation. Due to subtraction operation there are chances of occurrence of negative decimal values. So, to avoid negative values, add 10 to each decimal values. These new values generated are used for encryption of plain text using X-OR operation. But before encryption, generating equivalent ASCII code for our plain text. Next performing encryption by parallely x-oring ASCII of plain text and ASCII of generated values from cover text. Since in our approach there are three basic airthmetic operation performed on every character of cover text, so therefore, every character will generate three numeric values each, such that, each and every numeric value will encrypt two ASCII values of plain text parallely. One encryption is done from beginning of array of ASCII values of plain text and another one is done from the end of array of ASCII values of plain text. So, therefore, we can say that one character of cover text is able to hide at most six characters of plain text. Thus reducing memory allocation problem for cover text and also reducing the time required for executing this approach by making use of parallelism technique. Since, we know that basic airthmetic operations and X-OR operation are fast to compute and we had also used these so we can say that our approach executes faster than any other existing approach of text based steganography. Although our approach generates an absurd message or information but still it would be better using this approach to secure your commercial data or personal data or industrial information on public cloud servers

REFERENCES

[1] W. Bender, N. Morimoto, D. Gruhl and A. Lu, “Techniques for data hiding,” IBM Systems Journal, vol.35, pp. 313-336, 1996.
[2] F. A. P. Petitcolas and M. G. Kuhn, “Information hiding- a survey,” In Proceedings of IEEE, vol.87, pp. 1062-1078, 1999.
[3] S. Bhattacharyya, G. Sanyal and I. Banerjee, “A novel approach of secure text based steganography model using word mapping method,” International Journal of Computer and Information Engineering, volume 4, pp. 96-103, 2010.
[4] G. Sanyal, I. Banerjee and S. Bhattacharyya “Novel text steganography through special code generation,” Int. Conf. on Systemics, Cybernetics and Informatics, 2011, pp. 298-303.
[5] K. Rabah, “Steganography the art of hiding data in cover text,” Information Technology Journal, vol.3, pp.245-269, 2004.
[6] D.Ghosh, S. Changder and N. C. Debnath, “A Linguistic approach for text steganography through Indian text,” 2010 2nd Int. Conf. on Computer Technology and Development, 2010, pp. 318-322.
[7] Sahil Kataria, Balvinder Singh, Tarun Kumar and Hardayal Singh Shekhawat, "PDAC (Parallel Encryption with Digit Airthmetic of Cover Text) Based Text Steganography", IEEE, 2014.
[8] M. S. Shahreza and M. H. S. Shahreza, ”A new approach to Persian/Arabic language text steganography,” In 1st IEEE/ACIS Int. Workshop on Component-Based Software Engineering, Software Architecture and Reuse, 2006, pp. 310-315.
[9] M. H. S. Shahreza and M. S. Shahreza, ”A new synonym text steganography,” Int. Conf. on Intelligent Information Hiding and Multimedia Signal Processing, 2006, pp. 1524-1526.
[10] S.Chaudhary, P.Mathur, T.Kumar, "A Capital Shape Alphabet Encoding(CASE) Based Text Steganography” , R. Sharma , Conference on Advances in Communication and Control Systems 2013 (CAC2S 2013) , India.
[11] William Stallings, “Cryptography and Network Security: Principles and Practice 6/e” Prentice hall,2002,pp. 681.
[12] Kataria S. , Singh K. , Kumar T. , Nehra M.S. , "ECR (Encryption with Cover Text and Reordering) based Text Steganography", IEEE Second International Conference on Image Information Processing(ICIIP), 2013.