

Comcrypt : An Encryption Algorithm based on Vernam Cipher

¹Maulik Kothari, ²Manthan Shah, ³Meet Malde, ⁴Ashutosh Wad

^{1,2,3}K. J. Somaiya College of Engineering, Mumbai, India

⁴Dept. of Computer Engineering, Thadomal Sahani Engineering College, Mumbai, India

Abstract

In today's information age, information sharing and transfer has increased exponentially. Security, integrity, non-repudiation, confidentiality and authentication services are the most important factors in information security. Of all the methods of encryption ever devised, only one has been mathematically proved to be completely secure. It is called the Vernam cipher or one time pad. The worth of all other ciphers is based on computational security. If a cipher is computationally secure this means the probability of cracking the encryption key using current computational technology and algorithms within a reasonable time is supposedly extremely small, yet not impossible. In theory, every cryptographic algorithm except the Vernam Cipher can be broken given enough cipher text and time. This is where COMCRYPT comes into picture. COMCRYPT is an encryption algorithm, which has been formulated on the lines of Vernam cipher. When a passphrase is taken from the user, a scrambling algorithm is implemented on it, which generates two more random keys. These keys are superimposed on each other and then XOR to the text to produce the cipher text. This algorithm was monitored on different plaintexts, and it was found that this method was almost unbreakable. This method supports multiple encryption and multiple decryption. A minor change in the text key will change the cipher text quite a lot.

Keywords

Scrambling Encryption, Symmetric Key Encryption, Vernam Cipher, Comcrypt

I. Introduction

The data security and data encryption are two important terms in Computer network. Few years back it was not a serious issue but now it is the most important issue. It is a big challenge for a sender to send confidential data from one computer to another computer or from one computer to a remote computer through a server. Imagine a situation when the entire banking information is hacked. Probably the entire Banking Industry will collapse. It means the security of confidentiality of data has now become a very important issue or challenge in data communication network. The confidential data cannot be sent from one computer to another computer in original form as the intruder can intercept the data and can do any kind of danger for sender or for the receiver. Sometimes we send bank transaction report, question paper, suggestions over the mail. The hackers can intercept those mails. All these things are happening because of free network access. If someone applies some common sense can access any data from any machine. Imagine that the hacker somehow break secured key of e-banking and intercept all data. The data must be protected from any unwanted intruder otherwise any massive disaster may happen all on a sudden [5]. The disaster may happen in any business house. This may be further worse if some intruder steals the confidential data of one business house and pass it to some rival company. That implies now in every place there must be some security of data. There must be various levels of security in data management. Because of this hacking problem network security and cryptography is an

emerging research area where the people are trying to develop some good encryption algorithm so that no intruder can intercept the encrypted message. These cryptographic algorithms can be classified into two categories: (i) symmetric key cryptography where one key is used for both encryption and decryption purpose. (ii) Asymmetric key cryptography: where two different keys are used one for encryption and the other for decryption purpose. The advantage of symmetric key cryptography is that the key management is very simple as one key is used for both encryption as well as for decryption purpose and this key must be secret and it should be known to sender and the receiver only and no one else. On the other hand in public key cryptography there are two keys one key is called public key, which may be available to anyone who wants to encrypt the message and the other one which is called secret key or the private key that must be kept only with the receiver. Because of factorization problem from encryption key no one can construct the decryption key. The problem of Public key cryptosystem is that one has to do massive computation for encrypting any plain text. Due to massive computation the public key crypto system may not be suitable to encrypt short message, in sensor networks, mobile networks etc. In the present work we are proposing a symmetric key method called Comcrypt which is an updated vernam cipher method[4]. In forward vernam cipher method we add the ASCII code of the plain text and the keypad and we take modulo with 256 and take the result as feed back in the next column. After finishing forward pass we apply slightly different way the vernam cipher method from last character of the encrypted text. Here instead of adding the ASCII code of the two characters we apply the XOR operation and the result we make modulo with 256 and take the same result as feedback to the next column. In the second pass we apply XOR operation instead of addition of ASCII codes. First we choose a block of 256 characters in a block. The last few characters which will be <256 we apply same method but now the size of key will be changed according to length of the residual characters. We have tested this method on various types of known text files and we found that the results were satisfactory. The present method may be applied in Defense network, mobile network, ATM network, Short message service etc.

II. Methodology

A. Comcrypt Process

A cipher uses encryption to scramble the text in a unique manner, making the decryption of the data uniquely available to the person who has access to the "key". This key will change the information from the ciphertext to plaintext. For many standard or commonly used encryptions, a password, typically four to eight characters long, acts as a key to view the plaintext. Unfortunately, this exposes inherent weaknesses in the security, as the integrity of the encryption is entirely dependent on the strength of a password and the inability to deduce it. Some products require users to select or accept an assigned password that is usually quite complicated and difficult to remember. Such passwords often contain both

upper and lower case letters mixed with numbers or symbols (e.g. "myPa\$\$wor3!"). Having an uncommon key may delay a digital intruder from immediately discovering its encryption but, unknown to the user, the skilled hacker may be engaging in a "brute force" attack from a distance. The unwanted hacker has the tools and the benefit of computers fitted with impressive power to quickly process algorithms that will "guess" all possible password combinations until the key is discovered. Software-based solutions such as Comcrypt; encryption algorithms, however, can offer greater security against aggressive intrusions due to a password with much greater length, such as 16 characters.

The user may enter a short length key, but a scrambling algorithm will scramble the key in different ways and add it to the original key making it a considerably long key.

B. Scrambling Process

Before Encryption the pass key is parsed through the scrambling process, which consists of following functions.

```
StringRev()
ScrambleKey()
similarDissimilaar()
BurrowWhealer()
```

STEP 1: Start

STEP 2: The first time key is obtained. Let K be the key.

STEP 3.1: Now, three arrays equal to the key length are created. So there are three arrays say k1, k2, k3.

Size(k1)=size(k2)=size(k3)=Length(k)

STEP 3.2: Each element is modded with 500 to keep values from being too high.

STEP 4: The first array is called key and stores the key using toCharArray() function.

char[] a=s.toCharArray().

STEP 5: The second array is called reverse and stores the reverse of the key.

Stringrev():

STEP 5.1: Let b=Char Array where b.length=a.length.

The array b is used to store the reverse string.

STEP 5.2: i=a.length-1, j=0

STEP 5.3: while(j<a.length)

Copy a[i] into b[j]

j++;i--;

STEP 6: The third array stores a rearranged version of the key done by:

Scramblekey():

STEP 6.1: j=key.length-1;

STEP 6.2: k=0, l=0;

STEP 6.3: for each i, from i=0 to i= key.length

STEP 6.4:

if(i=0), go to step 6.5 else go to step 6.6

STEP 6.5:

new[j--]=key[i];

i=1.

STEP 6.6:

new[k++]=key[i];

i=0.

STEP 7: The three arrays are then added to each array creating a fourth array to be referred to as key from the next step.

STEP 8: This new array is now checked such that no two similar characters are next to one another.

similarDissimilar(char a, char b):

STEP 8.1: Let s be a null string.

STEP 8.2: n=(int)a, i=0, j=0.

STEP 8.3: n=n*1000

n=n+b

STEP 8.4: while(n!=0)

s+=n%2;

n=n/2;

STEP 8.5: Length of i=(length of s)/2

STEP 8.6:

If each character != 1

Increment i.

STEP 8.7:

For each value of k, from 0 to j,

n+=(int)Math.pow((double)2, (double)j++)*((int)s.charAt(k)-48)

STEP 8.8:

For each value of k from i+1 to length of the string

n+=Math.pow((double)2, (double)j++)*((int)s.charAt(k)-48)

STEP 8.9:

return new String(ch);

STEP 9: The similar characters are then changed to different characters using bit manipulation. A Burrow Wheeler performs this. The transformation permutes the order of characters. If the original string had several substrings which occurred often, then the transformed string will have several places where a single character is repeated several times in a row. This is useful for compression.

BurrowWheeler():

STEP 9.1:

Initialize i=0, j=0.

STEP 9.2:

Initialize string temp to null.

STEP 9.3: If two bits next to each other are similar

STEP 9.4: Concatenate the string

STEP 10: The array is now checked such that no more than 10% of the key are similar characters, if this figure is not achieved the offending characters are grouped and

Go to Step 7.

STEP 11: Save key

STEP 12: End

C. Encryption Algorithm

Comcryptenc(f1, f2):

STEP 1: Start

STEP 2: Pass the key through the Scrambler.

STEP 3: Invoke similarDissimilar(), Burrow wheeler() to generate a new key.

STEP 4: pass=1, times3=1, ch1=0

STEP 5: A block from the input file f1 is taken (<256 characters)

STEP 6: If block size <256, go to step 8.

STEP 9: If block size <256, goto Step 14

STEP 10: Each character of the block is copied into a character array str[]

STEP 11: encryption() function called and str[] is passed as parameter along with the size of the block

STEP 12: if(pass==1)

times=(times+times3*11)%64

pass++

else if(pass==2)

times=(times+times3*3)%64

```

pass++
else if(pass==3)
times=(times+times3*7)%64
pass++
else if(pass==4)
times=(times+times3*13)%64
pass++
else if(pass==5)
times=(times+times3*times3)%64
pass++
else if(pass==6)
times=(times+times3*times3*times3)%64
pass=1
STEP 13: Scrambler is called using the current value of times.
STEP 14: Each character of the last block (residual characters , if
any) is copied into str[]
STEP 15: encryption() called using str[] and the no. of residual
characters
STEP 16: Return

```

```

encryption(str[],n):
STEP 1: Start
STEP 2: ch=(str[0]+key[0]+ch1)%256
STEP 3: ch is wrote into output file
STEP 4: ch1=ch
STEP 5: i=1
STEP 6 :if i≥n, goto Step 11
STEP 7: ch=(str[i]+key[i]+ch1)%256
STEP 8: ch is wrote into the output file
STEP 9: ch1=ch
STEP 10: i=i+1 ,goto Step 6
STEP 11: Return

```

D. Decryption Algorithm

```

Comcryptdec(f1,f2):
STEP 6: if k<1 goto Step 11
STEP 7 :file_rev(outf1,outf2) : p=k%2
STEP 8: if p==1
times=times2;
file_rev(outf1,file1);
decryption(file1,outf1);
times=times2;
else if p==0
comcryptdec(outf2,outf1);
times=times2;
file_rev(outf1,file1);
comcryptdec(file1,outf1);
times=times2;
STEP 9: k=k-1 Goto Step 6
STEP 10 : copy the contents of file „outf1“ into file 1,file2.
STEP 12: Stop

```

```

decryption(str[],n):
STEP 1: Start
STEP 2: ch=(256+str[0]-key[0]-ch1)%256
STEP 3 : ch is wrote into the output file
STEP 4 : i=1
STEP 5: if i≥n, go to Step 9
STEP 6: ch=(256+str[i]-key[i]-str[i-1])%256 STEP 7: ch is wrote
into the output file
STEP 8 : i=i+1 ,goto Step 5

```

```

STEP 9 : ch1=str[n-1]
STEP 10: Return.

```

III. Implementation

The algorithm proposed by us, has been transformed into a working graphical user interface .The software enables us to encrypt text files, word documents ,media files.

- The user is required to enter the passphrase in the password text field.
- The user is allowed to choose between a binary or Unicode output.
- Once the text is entered or file is opened ,encode button is clicked, which implements the algorithm on plain text using passphrase to give cipher text.
- The entered text can be encrypted multiple times to ensure higher level of security, using same or different sets of keys.
- Once the file is encrypted, it can be saved to a desired location. Now nothing but the same passphrase used to encrypt the file, can decrypt it.

The implementation of the software also enables us to utilize one of the several possible applications that is encryption of e-mails. Using the SMTP server of gmail , the software allows us to send encrypted emails which can be decoded by the receiver using 'Comcrypt' Software .

IV. Conclusion

In the present work we use a scrambler algorithm to make the encryption process too hard. From the method also it is very clear that our encryption method is very hard. We apply our method on some known text where the same character repeats number of times and we found that after encryption in the output pattern there is no repetition of pattern in the output string. We have tested this feature closely and we found in almost all cases we have got satisfactory result. This is possible as we have used modified vernam cipher method with feedback character. The overhead of the present method is very less and hence this method may be applied specially in encryption of bio-informatics data where the same pattern is repeated or to encrypt short message, password etc. There is lot of scope to modify the present method. The merit of this method is that it is almost impossible to break the encryption algorithm without knowing the exact key matrix. We propose that this encryption method can be applied for data encryption and decryption in banks, in defense, in government sectors for sending confidential data. The present algorithm may be used for database encryption also.

V. Acknowledgements

We are very much grateful to Department of Computer Engineering to give us opportunity to work on symmetric key Cryptography. We sincerely express our gratitude to Prof. Deepak Sharma and Prof. Rohini Nair for providing great insight in the field of System Security and giving constant inspiration to carry out research work.

References

- [1] Debanjan, D., Joyshree, N., Megholova, M., Neha, C., Asoke, Nath., "An Integrated symmetric key cryptography algorithm using generalized vernam cipher method and DJSA method: DJMNA symmetric key algorithm", Proceedings of IEEE International conference : World Congress WICT-2011 held at Mumbai University 11-14 Dec, 2011, pp. 1203-1208, 2011.
- [2] A, Nath., S, Das., A, Chakrabarti., "Data Hiding and Retrieval", Proceedings of IEEE International conference on Computer Intelligence and Computer Network held at Bhopal from 26-28 Nov, 2010.
- [3] Dripto, C., Joyshree, N., Suvodeep D., Asoke, N., "A new Symmetric key Cryptography Algorithm using extended MSA method :DJSA symmetric key algorithm", IEEE conference CSNT-2011 held at SMVDU 03/06/2011 to 05/06/2011.
- [4] Dripto, C., Joyshree, N., Sankar, D., Shalabh, A., Asoke, N., "Symmetric key Cryptography using modified DJSSA symmetric key algorithm", WORLDCOMP-2011 held at Las Vegas, USA 18-21 Jul 2011.
- [5] Trisha, C., Tamodeep, D., Joyshree, N., Shayan, D., Asoke, N., "Symmetric key cryptosystem using combined cryptographic algorithms- generalized modified vernam cipher method, MSA method and NJSSAA method: TTJSA algorithm R", Proceedings of IEEE International conference: World Congress WICT-2011 held at Mumbai University 11-14 Dec, 2011, pp. 1179-1184, 2011.
- [6] A, Nath., S, Ghosh., M, Mallik., "Symmetric key cryptography using random key generator", Proceedings of International conference on SAM-2010 held at Las Vegas(USA) 12-15 July, 2010, Vol. 2, pp. 239-244.
- [7] Dripto., "Advanced Symmetric key Cryptography using MSA method: DJSSA symmetric key algorithm".



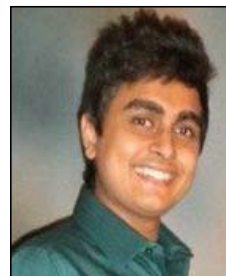
Manthan Shah is pursuing his B.E. degree in Computer Engineering from K.J. Somaiya College of Engineering, Mumbai, India. His research interests include cryptography, e-learning, open graph protocols and cloud computing.



Ashutosh Wad, is a student of Computer Engineering pursuing his B.E. degree at Thadomal Sahani Engineering College. His areas of interests include System Security, Neural Networks and Distributed Computin.



Meet Malde is pursuing his B.E. degree in computer engineering from K.J.Somaiya College of Engineering. His areas interests include cloud computing, artificial intelligence, software engineering and computer security.



Maulik Kothari is a final year student of Computer Engineering at K.J Somaiya College of Engineering. His areas of interest include System Security, Software Engineering, Web Development, Artificial Intelligence