Journal of Basic and Applied Engineering Research
Print ISSN: 2350-0077; Online ISSN: 2350-0255; Volume 2, Number 2; January-March, 2015, pp. 93-95
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/jbaer.html

A New Approach to Reduce the Length of Keys in One Time Pad Security Schema

Durga Prasad Dwivedi

CSE Deptt. Vishveshwarya Group of Institutions, Dadri G. B. Nagar U.P
E-mail: durgapdwivedi@gmail.com

Abstract—*In this paper I show how the random key stream can be used to reduce the length of key in One Time Pad security schema, Here we provided the new approaches to reduce the length of key up to the sqrt(n), where n is the length of the key. This approach calculates the fraction (p, q), where (m = p x q) and m is the length of the message, such that the difference of fractions (p,q) is minimum. We divide the message into p block, each of size q and generate a random number for first block of size q, which act as a key of length q for first block. The keys for each next block keys can be generated using a previous and a random number of length one.*

Keywords: *Cryptography, Cryptosystem, One-time pad, encryption, auto-key , Key enhancement, Random Number Generation.*

1. INTRODUCTION

Today's networks are seriously threatened by network attacks. Besides the rapid improvement of attacking technologies powered by profits, there are three reasons that cause the present serious status of network security, including internet itself having a weak basis, the current security technologies having respective drawbacks and limitations and the dilemma between security performance and according cost. By considering this three, we can put secure one time pad scheme. But the key size of one time pad schema is very large and is equal to the length of the message. For very large message, it is very complex to encrypt the message using one time pad schema due to the length of the key.

One Time Pad is good secrecy technique which was first decscibed by Gillbert Vernam in 1917 for use in automatic encryption and decryption of telegraph messages. It is interesting that the One-time Pad was thought for many years to be an "unbreakable" cryptosystem, but there was no mathematical proof of this until shannon developed the concept of perfect secrecy over 30 year later.

In this paper I try to reduce the length of key size and can secure the message strongly. Due **to** the reduced size of key length any large message can be encrypted very easily using One Time Pad Security Schema. Also I try to provide the same label of security as with the full length of key in one time pad.

2. ONE TIME PAD

One Time Pad has perfect secrecy because the key used is random and unpredictable as long as the key is transferred to the receiver properly. It is possible to prove that a stream cipher encryption scheme is unbreakable if the following preconditions are met:

- The key must be as long as the plain text.
- The key must be truly random.
- The key must only be used once.

The disadvantage of this cipher is that, it can't be used for longer messages since key-length ≥ msg-length and therefore it gets difficult to manage, store and transfer the key as the size of the plaintext increases. Since, we have seen its disadvantage of excessive key length and hence we can conclude that this advantage is has been proved very costly nowadays, since everybody wants to save his/her space on a

System while One Time Pad does not serves this purpose and hence it's not used for transferring of big messages but they are used as Key Distribution algorithms.

Because the One Time Pad key is completely random and unpredictable, two conclusions can be drawn:

- First, the probability of observing any particular One Time Pad key bit is equal to the probability of observing any other One Time Key bit.

Second, knowing all the previous values of key in a sequence tells us nothing about the next key bit.

3. PROPOSED ALGORITHMS FOR REDUCTION OF KEY LENGTH IN ONE TIME PAD SECURITY SCHEMA:

The proposed algorithm is to maintain the security of the original text which is encrypted by the algorithm and on the other hand also to minimize the various resources used by the algorithm and processing, such as lesser key length which reduces the amount of storage space required as well as easier and efficient for the sender to send the key to the receiver.

3.1 Encryption

For encryption the technique used here is that, the original message which is entered by the user is used by their ASCII values.

3.1.1 Random Number Generation

Since this algorithm depends on the random numbers, so we had to make sure that the generation of the numbers at the run-time must be completely anonymous without any kind of pattern in them just like we found in the Polyalphabetic cipher. Also, in this algorithm, I generate a random number of the range 0-25 for the alphabets (both uppercase and lowercase) and of the range 0-9 for numerals.

3.1.2 Keys

For generation of the keys for the encryption process uses the random number generated at the runtime.

Factoring Process

One of the main process to generate the key here depends on the factors of the length of the plaintext. They can be denoted as f1 and f2. Where these factors are calculated in a way such that the difference between two of them remains minimum.

Difference=min(f1-f2) where, f1>f2 The significance of these two factors of the length of the plaintext serves us a very important purpose i.e., the first factor (f1) will define the length of the blocks in which the plaintext could be divided while the second factor (f2) gives us the number of blocks that combines to form the plaintext. These factors also plays a very important role in the generation of the keys as they helps us to minimize the key for encryption process, which gives us the advantage over the One Time Pad.

Here, in this algorithm the key are basically of two kinds (k1 and k2):

1. Block Key (k1): The one which will be used in the encryption of every block whose length is for plaintext. This key's length will be equal to the first factor (f1). This will be generated only once for whole encryption process.
   f1=k1

2. Single Bit Key (k2): This is the single bit which will be used for every block of the plaintext except the first block.

This key is used to maintain the anonymity of the ciphertext as done in the Polyalphabetic cipher where the whole block which is generated once is used throughout the plaintext. This key or the single bit will be added to the every bit of the key (k1) in order to preserve the randomness of the ciphertext. This key will generated one less than times of the second factor (f2).

k2=(f2)- 1

Now, the encryption process of this cipher becomes very easy similar to the Caeser's Cipher just adding up the key to the plaintext and taking the mod of the resultant generated.

In this paper, we have encrypted the plaintext such as that if in the plaintext somewhere we have lowercase letter then to the corresponding ciphertext will also have the lowercase letter. And this goes for all Uppercase, Lowercase and numbers. But obviously, we can generalise this encryption process by simply putting up the complete range of the ASCII chart's values. We have described this so that the mod that we take during the encryption process could be understood since the mod taken uses the ranges of the corresponding ASCII ranges of the type (Lowercase, Uppercase and numbers) of the plaintext. Now, generalized encryption is as follows,
Ciphertext ,c(x)=(p1+(k1))mod m + (p2 +(k1+k2,1))mod m + (p3+(k1+k2,1+k2,2))mod m + …… + (pn
+ (k1+….k2,n-1+k2,n))mod m

where, p1, p2….pn are the blocks in which the plaintext has been divided, k2,1, k2,2…..k2,n are the single bit keys generated for the corresponding plaintext n-1 blocks and n is the number of plaintext blocks which are divided on the basis of the factors generated.

3.2 Decryption
Decryption process of this cipher is simply the opposite of the encryption process. To convert the ciphertext back to the plaintext the algorithm needs the key generated at the time of the encryption process should be entered by the user at the time of the decryption. As far as the key is concerned the similar process is followed as in the encryption process. All has to be done is to subtract the key from the ciphertext and therefore taking the mod of the resultant according to its ASCII ranges. It can be done as:

Plaintext, p(x)=(c1-(k1))mod m + (p2 - (k1+k2,1))mod m + (p3-(k1+k2,1+k2,2))mod m + …… + (pn - (k1+…k2,n-1+k2,n))mod m

3.3 Example
It can be explained by an example shown in fig. 1. Suppose plaintext is: "HellomynameisSagar".

And hence its Ciphertext is: **JobyyaupzioaqNshjg** as calculated in fig. 1.

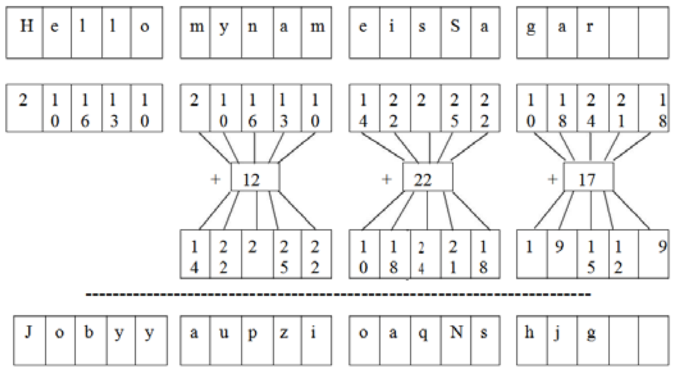

Fig. 1

In the fig. 1 all the additions are mod by their appropriate ranges and the key is mod by 26 for the alphabets and by 9 for all the numerals.

3.4 Strength of this Algorithms

As we've seen the two conclusions from One Time Pad can be drawn:

- First, the probability of observing any particular One Time Pad key bit is equal to the probability of observing any other One Time Key bit.
- Second, knowing all the previous values of key in a sequence tells us nothing about the next key bit.

Now firstly, from the first point of the conclusion which was observed in One Time Pad also applies to this algorithms since, as we've seen that except the first block of the key all the other block are added up with a randomly generated key which separately generated for each block of the key. And hence, similarly the probability of observing any key bit of this algorithms is equal to the probability of observing any other key bit of this algorithms.

Secondly, which is pretty obvious since every bit of the key is random so the previous value if figured by the attacker tells nothing about the next 40 bits in the sequence, which also implies that there is no pattern because all the keys are random unlike Polyalphabetic Cipher.

The 'Brute Force' attack

Brute force attacks use exhaustive trial and error methods in order to find the key that has been used for encrypting the plain text. This means that every possible combination of key bits must be used to decrypt the cipher text. The correct key would be the one that produces a meaningful plain text.

Unlimited computing power is useless

Let's assume an eavesdropper has intercepted a encrypted message and that he has unlimited computing power and time. A brute force attack would be very expensive for a plain text of reasonable size. For example, typical e-mail messages are at least 200 bytes long, requiring the testing of 1.600 bits. Even if the eavesdropper is both willing and able to do this, the following paragraph will describe why unlimited computational power will not ompromise the system.

Attackers must try every possible key

Since all Keys are equally likely and come from a completely unpredictable noise source that is

provablyrandom, the attacker has to test all possible key strings.

Impossible to guess the right plain text

If he used every possible key string to decrypt the cipher text, all potential plain text strings with the same length as the original plain text data would appear. As illustrated on the left-hand side, most of these potential plain text strings would not make sense; however, every meaningful string the same length as the original plain text data would also appear as a potential plain text string. Without knowing the applied Stochastic Cipher, the eavesdropper has no way of finding out which meaningful string is the original plain text. Thus, trying all possible keys doesn't help the attacker at all, since all possible plain texts are equally likely decryptions of the cipher text.

4. CONCLUSION

This algorithm is developed for fully meets the objectives of the system. The system is in a state where all the bugs are eliminated and it is working perfectly. It can be operated at high level of secrecy and efficiency and all the users associated with it shall understand its advantages over the other methods.

REFERENCES

Books

[1] Cryptography and Network Security by William Stallings.
[2] Software Engineering Strategies by Roger S. Pressman.
[3] Cryptography and Network Security by Behrouz A. Forouzan.
[4] Handbook of Applied Cryptography by Alfred Menezes.

Websites:

[1] www.wikkipedia.com
[2] www.stackoverflow.com
[3] www.creately.com
[4] www.tutorialspoint.com