ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 9, September 2015

Multilevel Encryption using Ribbon Based Technique for Symmetric Key Cryptography

Milit Betai[1], Neha Katre[2]

B.E. Student, Information Technology, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India[1]

Assistant Professor, Information Technology, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India[2]

Abstract: Symmetric key encryption is not given much importance when it comes to cryptography. Substitution ciphers have been used since a long time. Caesar Cipher is a basic substitution cipher that forms the basis of ROT13 and Vigenere cipher. It is the basic symmetric key cryptographic technique with a very simple implementation. However, there arose vulnerabilities to this technique, as the key was just a number and predicting that number was not difficult at all. Keeping this in mind, the goal of our research was to develop a symmetric encryption technique that encrypts the data in a fashion similar to Caesar Cipher and provides a minimal probability of guessing the key and allowing multiple levels of encryption, thus, harboring more security. Here we offer a multiple character keyset that would enhance the number of possible keys and which would in turn reduce the chances of guessing it.

Keywords: Ribbons, Referral Ribbon, Substitution Cipher, Symmetric, Encryption.

I. INTRODUCTION

A symmetric key encryption technique as shown in Figure 1 makes use of one key that is used for encrypting the plain text as well as decrypting the cipher text. The key has to be known by the sender (the one who encrypts the message) as well as the receiver (the one who decrypts the message).

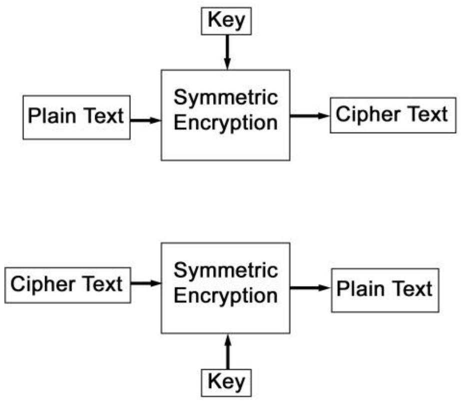


Figure 1 - Symmetric Key Cryptography

A substitution cipher is a cipher in which a letter in the plaintext is replaced by another letter. The function that defines the replacement of a letter by another is known as the key in such ciphers. Caeser Cipher [1] is one of the oldest known symmetric ciphers.

Caeser Cipher:
In cryptography, a substitution cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. The method is named after Julius Caesar, who used it in his private correspondence.

Figure 2 shows the arrangement of letter in Caeser Cipher. It shows how the second array of alphabets is shifted by 3.

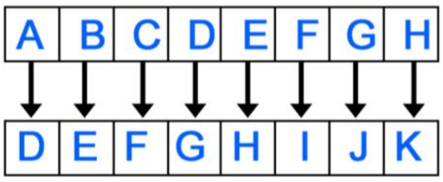


Figure 2 - Caeser Cipher

The main drawback of such substitution ciphers is that once we know the substitute of a letter in the plaintext, the substitutes of the other alphabets can be guessed easily. Figure 2 shows that A is replaced by D, B by E, C by F, and so on. Thus, if an outsider knows that C is replaced by F, the offset (i.e., 3) can be easily calculated and would allow the outsider to find the substitution of every alphabet. Another disadvantage of symmetric encryption is that the key must be known to the sender and the receiver. Our research works on providing symmetric encryption by disguising the key as the data.

II. APPROACH

The encryption method that we offer is a substitution cipher that replaces one character with another. It induces an array of all the characters of the characters sets like utf-8[2], windows-1252[3]. This array is called as ribbons. Think of each ribbon as an array of alphabets in Caesar Cipher only with a series of symbolic characters which include alphabets, numbers and all the special characters. The symbols in ribbons are placed in distinct order; thus, no two ribbons have the same order of the characters. Thus, the number of combinations formed will be equal to the factorial of the number of characters in it. The number of the ribbons determines the key length that can be added or removed as required. Since the number of the ribbons

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 9, September 2015

determines the length of the key, new ribbons can be added or the old ones can be deleted, not affecting the working of the encryption logic. One letter from a string is selected at a time to substitute a character in the plain text.

Generation of the key:
The key is a set of characters, one from each of the ribbons, used in this technique. Since every ribbon contains the same characters, every ribbon is shuffled to have distinct arrangement of these characters within itself. On selecting a key, i.e. one character from every ribbon, each ribbon is rotated linearly to bring the corresponding selected character to the beginning of the same, i.e, at index 0 of every ribbon. Thus, selecting the first character of each ribbon forms the key.

Algorithm for generation of the key:

- Set a key that is of length X.
- Initialize a counter variable C to 0.
- Rotate the Ribbon_0 such that Ribbon_0[0]= key[C]. Ribbon_0 is called as the Referral Ribbon.
- Increment C.
- Rotate all the remaining ribbons such that Ribbon_K[0]=X[C].
- Repeat the above steps until C=N where K is the number of the ribbons and N=K-1.

Thus, we have rotated the ribbons as per the key such that every character in the key represents the first element of the corresponding ribbon.

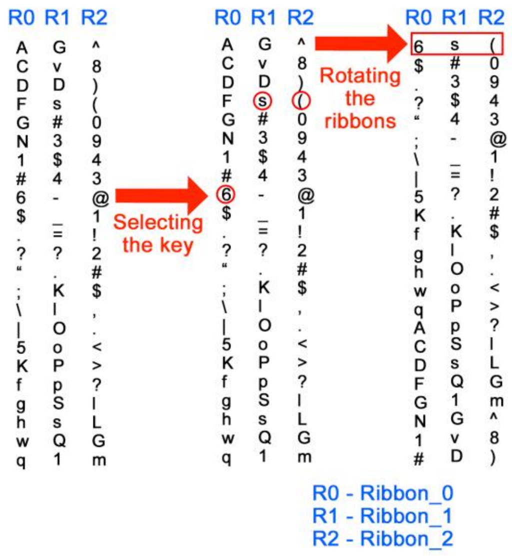


Figure 3 - Generation of the key.

The number of possible combinations generated for C is C!, approximated to: $[\{\sqrt{((2C+1/3)\pi)}\}C^{C}e^{-C}]^{k}$ where C is the number of characters in a ribbon and k is the number of ribbons used. This is done by using Gosper's Factorial Approximation Formula [7]. Thus, for k ribbons, we get $(C!)^{k}$ combinations.

Figure 3 shows the process of generating a key. There are three ribbons, each having the same set of characters but arranged in distinct orders. Let say that the key to be set is "6s(". So, each character is selected from its corresponding ribbon. The entire ribbon is linearly rotated with respect to that character so as to bring it to the first location of the ribbon, i.e., index 0. Thus, "6", "s' and "(" are all brought to the first location of their corresponding ribbons. This arrangement of ribbons so obtained is then used to encrypt the data.

Encryption:

General algorithm for encryption:

- Let L be the length of the plain text.
- For every character "x" in plain text, search for that character in the Referral Ribbon (Ribbon_0) and save its index in i.
- Replace "x" with the element corresponding from Ribbon_1[i].
- Repeat the above process for replacing every character in the plain text with a character from one ribbon at a time from Ribbon_1 to Ribbon_N in a cyclic order.
- Print the encrypted string (cipher text).

Here, we decided to opt for a customized key range. Thus, the key size can be varied as and when required. Here, the encryption takes place by substituting each letter in the input string with a letter from each ribbon. The example below explains that clearly. Depending on the key that is set, the ribbons are rotated to bring the letter from the key corresponding to those in each ribbon to the top. It is like the working of the symbolic wheels of a poker slot machine, aligning them with respect to the key used. Only on getting the right key can the ribbons align themselves to the sequence used while encrypting the data and that the encrypted string can then be decrypted. All the characters from the utf-8 character set can be used in this algorithm, populating each ribbon with all of the utf-8 the characters. The probability of guessing the key is 1/(2048^n), where n is the number of ribbons. [2048: considering the characters in utf-8 character set].

Consider Ribbon_0 as the Referral Ribbon from which we look up the characters occurring in the plain text. When a character of a plain text is found in Ribbon_0, a character from Ribbon_1, having the index same as the index of the character in Ribbon_0 is used to replace the character in the plain text. Then for the second character in the plain text, on finding the same from Ribbon_0, the subsequent indexed character is then selected from Ribbon_2 and used to replace the second character in the plain text. This is done in a circular fashion until a character from Ribbon N is used to replace the character in the plain text. After this, the substitution starts again from Ribbon_1 and goes on till Ribbon N.

Figure 4 shows the working of the encryption. Let's encrypt the word "CAR". The key decided is "Aq0@". Thus, the ribbons are rotated to bring these characters in the beginning. Now, "C" is looked up in the Referral Ribbon (Ribbon_0) and its location is saved in a variable i. Then, the character at I in the next ribbon, i.e., Ribbon_1 is selected. Thus, "C" is replaced by "e". For the next character in the plain text which is "A", we again look it up in the Referral Ribbon and its location is saved in i. Now, “A” is replaced by Ribbon_2[i] which is “0”.

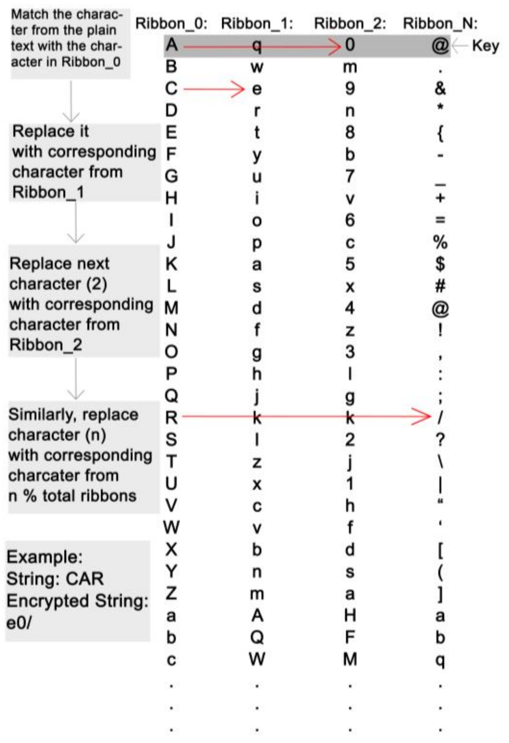

Match the character from the plain text with the character in Ribbon_0
Replace it with corresponding character from Ribbon_1
Replace next character (2) with corresponding character from Ribbon_2
Similarly, replace character (n) with corresponding charcater from n % total ribbons
Example: String: CAR Encrypted String: e0/
Ribbon_0:
Ribbon_1:
Ribbon_2:
Ribbon_N:
Key

Figure 4 - Overview of the encryption.

For the last character, “R”, we look it up in Ribbon_0 and save its location in I and replace it with Ribbon_N[i], i.e., “/”.
Thus, “CAR” is encrypted to “e0/”.

Decryption:
The decryption process is the exact opposite of the encryption process. The first thing to be done, as like in the encryption process, is to set the key so that the ribbons can be aligned. Then, the characters are matched from the Ribbons (1 to N) and are then replaced with the corresponding elements from the Referral Ribbon (Ribbon_0).

General algorithm for decryption:
- Let L be the length of the cipher text.
- For the first character “x” in the cipher text, search for “x” in Ribbon_1 and save its index in i.
- Replace “x” with the character at Ribbon_0[i]. Ribbon_0 is the Referral Ribbon.
- Repeat the above process for replacing every character in the cipher text (by getting the index in i from Ribbon_1 to Ribbon_N) with a character at Ribbon_0[i], one at a time in a cyclic order.
- On reaching Ribbon_N, start again from Ribbon_1.
- Print the plain text (decrypted string).

Figure 5 shows the overview of the decryption process. Consider the cipher text as “e0/”. Take “e”, the first character from the cipher text and search for it in Ribbon_1. On finding the element, save its index in i and replace “e” with the corresponding element at Ribbon_0[i], i.e., “**C**”. Now take “0” and look it up in Ribbon_2 and save its index in i. Now select the element corresponding to its index from Ribbon_0, i.e., the element at Ribbon_0[i], which is “**A**”. Now for the last character, look up “/“ in Ribbon_N and save its index in i again. Replace it with the element at Ribbon_0[i], i.e., “**R**”.

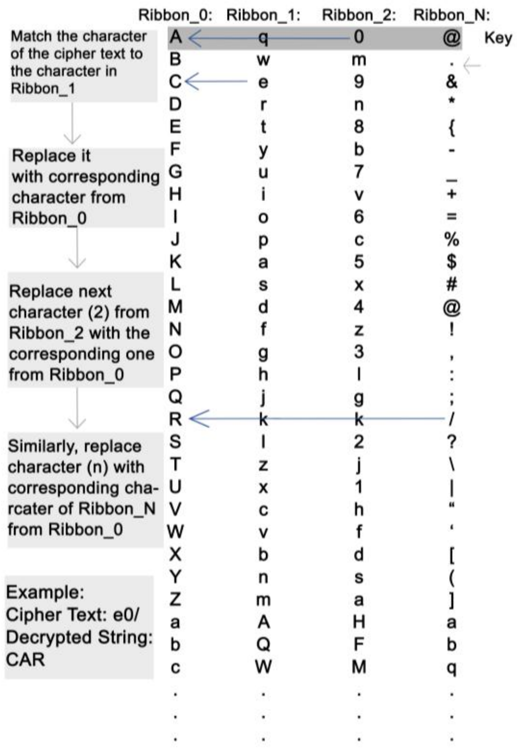

Ribbon_0:
Ribbon_1:
Ribbon_2:
Ribbon_N:
Key
Match the character of the cipher text to the character in Ribbon_1
Replace it with corresponding character from Ribbon_0
Replace next character (2) from Ribbon_2 with the corresponding one from Ribbon_0
Similarly, replace character (n) with corresponding charcater of Ribbon_N from Ribbon_0
Example: Cipher Text: e0/ Decrypted String: CAR

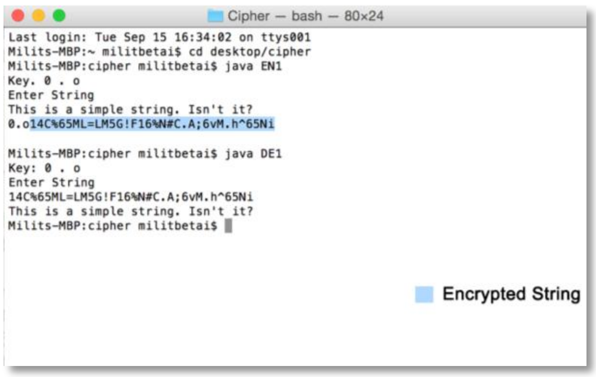
Figure 5 - Overview of the decryption.

Thus, to ensure maximum security, a higher number of ribbons can be used (usually greater than 100) enhancing the possible combinations and reducing the chances of predicting the key.

III.VARIATIONS

Since the number of ribbons and hence the length of the key is variable, numerous variations can be applied to this encryption method to achieve more complex ways of encrypting data like by obfuscating the encrypted string to further reduce the chances of understanding the encryption algorithm by an outsider. It can be done in numerous ways like:

1. The key can be concatenated in the cipher text which allows the key to be disguised as a part of the cipher text as shown in Figure 6. The key would then be unknown and unidentifiable to an outsider. In this variation, the 3-digit key is concatenated towards the beginning of the encrypted string. This allows the key to become a part of the encrypted string that remains unknown to an outsider or to a person with no knowledge of the working of this technique. The key, however, can be concatenated in the end, or in the middle of the encrypted string or after any other character as desired.



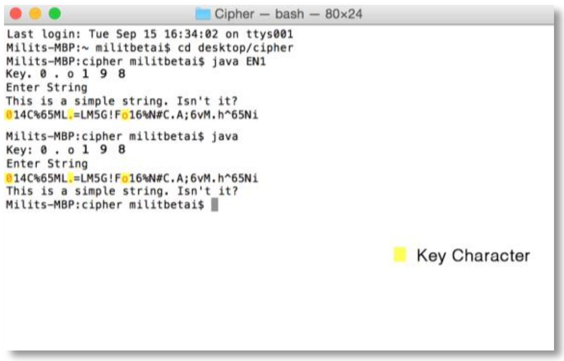
```
Cipher — bash — 80×24
Last login: Tue Sep 15 16:34:02 on ttys001
Milits-MBP:~ militbetai$ cd desktop/cipher
Milits-MBP:cipher militbetai$ java EN1
Key. 0 . o
Enter String
This is a simple string. Isn't it?
0.o14C%65ML=LM5G!F16%N#C.A;6vM.h^65Ni

Milits-MBP:cipher militbetai$ java DE1
Key: 0 . o
Enter String
14C%65ML=LM5G!F16%N#C.A;6vM.h^65Ni
This is a simple string. Isn't it?
Milits-MBP:cipher militbetai$
```

Figure 6 - Concatenation of the key and the cipher text.

This ensures that only a legitimate user with the knowledge of the presence of the key in the encrypted string will be able to cut the key out of it and use the key on the remaining part of the encrypted string to decrypt it.

2. In this variation, the 3-digit key is broken up and spread out across the encrypted string. Figure 7 shows this variation. On observing Figure 7, it is seen that the key actually consists of six characters: 3 characters and 3 numbers. Here, the key is divided such that each of the given number marks the location of the next key character from the current key character. From 1 9 8 we can see that the first key character "0" is at the first location. The next number is 9, i.e. the next key character "." is 9 characters away from the current key character "0". Similarly, the last key character is "*" 8 characters away from the current key character ".



```
Cipher — bash — 80×24
Last login: Tue Sep 15 16:34:02 on ttys001
Milits-MBP:~ militbetai$ cd desktop/cipher
Milits-MBP:cipher militbetai$ java EN1
Key. 0 . o 1 9 8
Enter String
This is a simple string. Isn't it?
014C%65ML.=LM5G!Fo16%N#C.A;6vM.h^65Ni

Milits-MBP:cipher militbetai$ java
Key: 0 . o 1 9 8
Enter String
014C%65ML.=LM5G!Fo16%N#C.A;6vM.h^65Ni
This is a simple string. Isn't it?
Milits-MBP:cipher militbetai$
```

Figure 7 - Distributive embedding of the key in the cipher text.

3. The number of ribbons used can be varied on a regular basis. Thus ensuring a different length key for every new use, harboring more security.

4. Useless ribbons (which are present but never used) can be introduced to complicate the understanding of the encryption system

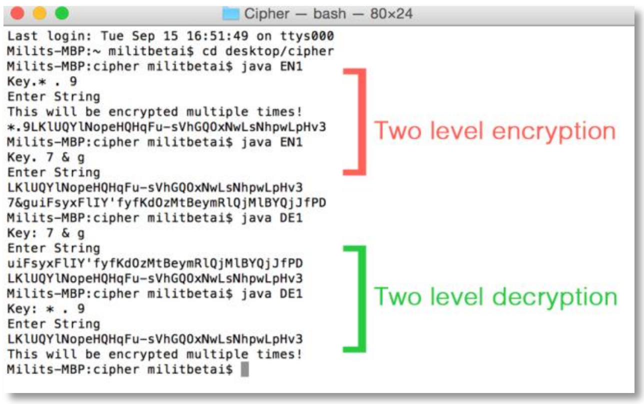
The variations are applied to allow the sharing of the secret key without making it obvious to an outsider. The key, however, can also be shared via other medium and/or algorithms.

- Diffie-Helman [4] or RSA [5] algorithm can be used to exchange or share a generated key.
- Steganography [6] can be used to hide the key in an image and share the image, hereby sharing the key with only the legitimate user.

Figure 8 shows two level encryption. This is done by encryption the encrypted data with a new key as well as can be done with the older one. This further enhances the security. The data can be encrypted n times to facilitate n level encryption.

IV. CONCLUSION

Using the approach mentioned in this paper, we believe that it is possible to achieve high levels of security by having multiple levels of encryption. This can be done by further encrypting the encrypted data for as many times as required. Also, many variations like viding the key across the message, disguising the key in the message by concatenating it, sharing the key before hand, obfuscating decryption by introducing dummy ribbons, using another encryption technique for key generation and sharing. These variations when applied enhance the security offered by this technique as one wouldn't know which variation or a policy is in effect. Depending on the policy implemented, the outsider wouldn't know if the key is present in the encrypted data or not.



```
Cipher — bash — 80×24
Last login: Tue Sep 15 16:51:49 on ttys000
Milits-MBP:~ militbetai$ cd desktop/cipher
Milits-MBP:cipher militbetai$ java EN1
Key.* . 9
Enter String
This will be encrypted multiple times!
*.9LKlUQYlNopeHQHqFu-sVhGQOxNwLsNhpwLpHv3
Milits-MBP:cipher militbetai$ java EN1
Key. 7 & g
Enter String
LKlUQYlNopeHQHqFu-sVhGQOxNwLsNhpwLpHv3
7&guiFsyxFlIY'fyfKdOzMtBeymRlQjMlBYQjJfPD
Milits-MBP:cipher militbetai$ java DE1
Key: 7 & g
Enter String
uiFsyxFlIY'fyfKdOzMtBeymRlQjMlBYQjJfPD
LKlUQYlNopeHQHqFu-sVhGQOxNwLsNhpwLpHv3
Milits-MBP:cipher militbetai$ java DE1
Key: * . 9
Enter String
LKlUQYlNopeHQHqFu-sVhGQOxNwLsNhpwLpHv3
This will be encrypted multiple times!
Milits-MBP:cipher militbetai$
```

Figure 8 – Two level encryption.

V. FUTURE SCOPE

Encryption can be made more secure by eliminating the need of having just one character to substitute one character in the plain text. This would replace a character with a string of length greater than one, hereby, making the encrypted string more secure. Along with this, new and more complex variations can also be introduced.

REFERENCES

[1] Caeser Cipher - link: https://en.wikipedia.org/wiki/ Caesar_cipher
[2] Unicode Transformation Format - link: http://www. unicode .org
[3] Windows 125 - link: https: //msdn.microsoft.com/ enin/go global/cc305145.aspx
[4] Diffie-Helman – link: http://cs.unc. edu/~ Fabian / course_papers/diffie.hellman.pdf
[5] RSA Algorithm – link: https://people.csail.mit edu/ rivest/ Rsapaper.pdf
[6] Steganography [Steganography Techniques– A Review Paper, International Journal of Emerging Research in Management &Technology ISSN: 2278-9359(Volume-3, Issue-5) - May, 2014]

link: http://www.ermt.net/docs/papers/Volume_3/5_May2014 V3N5-190.pdf
[7] Gosper's Factorial Approximation Formula. Link: http://mathworld.wolfram.com/StirlingsApproximation. html

BIOGRAPHIES

Mr. Milit Betai is a Bachelor's student pursuing his B.E in Information Technology from Dwarkadas. J. Sanghvi College of Engineering, Vile Parle, Mumbai, Maharashtra, India. His fields of interest are security, data mining, algorithms and analysis, and artificial intelligence.

Mrs. Neha Katre is an assistant professor in the department of Information Technology at Dwarkadas J. Sanghvi College of Engineering, Vile Parle, Mumbai, Maharashtra, India. Her areas of interest range from security, databases to computer graphics and virtual reality.