Advancement on Substitution Cipher using Multiple Substitution Table

Aman Roy
GHRCEM
Pune, India
royaman8757@gmail.com

Aadesh Mirajkar
GHRCEM
Pune, India
mirajkaraadesh@gmail.com

Geeta Atkar
GHRCEM, Assistant Professor
Pune, India
geeta.atkar@raisoni.net

***Abstract* - Cryptography is one of the ways to secure data while transmitting from one location to another. There are number of techniques to secure a data by using cryptography. This paper focuses on Substitution Cipher by using Multiple Substitution Table. In ESC (Extended Substitution Cipher), the key has two parts. The first part is used as a seed value and the second part is 'n' which is used to generate n different substitution cipher which will be used for encrypting the plain text. After generating cipher text two attacks are applied one is Brute Force attack and another one is frequency analysis attack. Results are generated. This is fastest and improved method for generating cipher text.**

Keywords - Cryptography, Substitution Cipher, PRNG, Frequency Analysis Attack, Brute Force Attack

I. INTRODUCTION

In the world we are living, everything is interconnected. The need for communication is progressing exponentially. With the need of communication, one thing also comes into the picture. That thing is "secrecy" or "privacy". Even if you are not concerned about these things, still sometimes when we communicate we need to send some data which is having some confidential information. At that time, we need to have some mechanism which keeps your data secure and at that very moment, Cryptography comes into picture.

Cryptography is a process of safeguarding sensitive information by encrypting it using different algorithms. Encrypted data is unreadable and cannot be accessed by any unauthorized person. For making it readable again we have to decrypt it using the same key which is used to encrypt the data. However, the same key doesn't have to be used for encrypting and decrypting the data. One of the fields of cryptography is Public key cryptography in which two keys are there. One is used for encryption and the other one is used for decryption. We are looking at secret-key cryptography(SKC) as of now in which one key is used for encryption as well as for decryption.

Substitution Cipher is a well-known SKC which is very to understand, very easy to implement and unfortunately very easy to break. In this, every letter is mapped to another letter. For encryption, we use to substitute every letter with the letter it is mapped to. For decryption, we do the substitution in the opposite direction. There is one property of plaintext which is conserved even after encrypting it using substitution cipher, i.e. frequency of letter. Using frequency analysis, the ciphertext can be broken easily.

For making it more secure, extended substitution cipher comes for the rescue. Instead of using one substitution table, multiple substitution table is used. It makes the letter frequency close to uniform and the ciphertext becomes harder to break. One more thing which I have tried to resolve is to encrypt the " "(space character) too so that two or three letter words can't be guessed easily. There are multiple design decisions taken for making it non prone to bugs and much more secure.

II. LITERATURE SURVEY

SIT: A Lightweight Encryption Algorithm for Secure Internet of Things [7]
In this paper the need for the lightweight cryptography have been widely discussed for securing an image, also the shortcomings of the IoT in terms of constrained devices are highlighted. In secure systems the confidentiality of the data is maintained and it is made sure that during the process of message exchange the data retains its originality and no alteration is unseen by the system. The IoT is composed of many small devices such as RFIDs which remain unattended for extended times, it is easier for the adversary to access the data stored in the memory. The proposed algorithm gives structure suitable for implementing in IoT environment. Some of the algorithms like AES, 3-Way, Grasshopper PRESENT, SAFER, SHARK, and Square suse Substitution-Permutation (SP) network Several rounds satisfies the Shannon's confusion and diffusion properties that ensues that the cipher text is changed in a pseudo random manner.

[1] proposed the use of modern avatar of Julius Caesar cipher technique to encrypt and decrypt the message into cypher text by choosing primitive root first and then using the encryption technique : Ci =(Mi + Ki) mod 26 and encryption technique : Mi=(Ci - Ki) mod 26

978-1-7281-4042-1/19/$31.00 ©2019 IEEE

where=message words and K's values are used as shift keys with an assumption to determine the prime factors and primitive roots which are used for determining the logarithmic value and create difficulty for an eavesdropper, as he is supposed to practice (n+2)! Attempts for the prime factor and another (n+2)! Attempts for the primitive root. Whereas the Caesar cipher has possible key space of 26! which is 88 bit long which may be get decrypted by using the brute-force. So, their proposed algorithm will make the analysis of interceptor fail because in it values do not follow any sought of a common pattern.

[2] provides bit level conversion of inconsistent block length characters for encryption. Here block of sixteen characters /28 bits is taken. Substitution technique is being followed on the block of characters along with transpositions using multidimensional array. The block is being operated with one-time sub key which will produce intermediate result of similar length. The previous text block is combined with consecutive 8 characters/64 bits of the plain text and gives a block containing 192 bits. This is used as present block of text which produces a new text block containing 24 characters with same technique. Next 8characters are considered with previous block and same technique is applied to give a block of 256 bits. If there are more than 32 characters in plain text i.e. 256 bits then every 256 bit block is XORed with previous 256 bits block other than the first block. At the end bits are being chosen from MSB position and chosen bits are processed through a special substitution technique to give final encrypted block. [7] proposed a substitution technique which is being followed on block of characters with transposition. They considered a block of 16 Characters /128 bits. Intermediate result is being produced by using the block along with one sub key. Furthermore, the last text-block gets combined with consecutive 8/64-characters bits of plain-text and returns a 192 bits block. Similarly, it gets continued which produces 24 characters then next 8 Characters gets in consideration with the previous block and same technique is applied to give a 256 bits. If in the plain-text there are more than 256 bits then every 256 bits block gets XORed with last block except the first one. At last, MSB positional bits chosen and gets processed through a special substitution technique to give a final encrypted block.

III. PROPOSED MODEL

The classical substitution cipher is one of the earliest cipher known to humankind. Just due to being the oldest, it makes it very vulnerable. There are many attacks that can be performed. In this paper, I have tried a new method which I have called "extended substitution cipher" which is based on substitution cipher but has an additional layer of security with the help of multiple substitution table.

Before starting **Extended Substitution Cipher**, let's have a look at classical **Substitution cipher**.

A. Substitution cipher

This cipher is made by **pairing each alphabet to a random alphabet**. Each letter corresponds to one and only one letter. So, for encrypting any text data you substitute every letter with its paired up partner.

Let's see an example to have a clear view: -

Table No. 1 Substitution table

Alphabets	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
Cipher	t	g	j	v	h	z	i	r	p	b	f	e	o
Alphabets	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
Cipher	a	s	w	c	q	n	u	m	y	d	l	x	k

Using the above table, let's encrypt some text to see how it works.

Plaintext - THIS IS A SAMPLE TEXT
Ciphertext - urpn pn t ntoweh uhlu

As you can see this is very easy to do. You can do this with pen and paper also. Decryption is also very easy to do. For that, you only have to go backward. For every ciphertext, you have to substitute it with its corresponding alphabet. So if it is very easy then what is the problem with it and what kind of attacks that can be performed.

B. Attacks that can be performed: -

The **brute force attack** can be done but it is **very slow and impractical** to use. It will take *26!* key search to guess the right set of keys. The complexity of this attack is so high to do it in practice.

Letter frequency is preserved in this method and that feature welcomes easy to decipher without the key.

(" ") Space is also a character and is **not encrypted** which reveals the gap between the words which makes it easy to crack. So we are at the point to see how to overcome these things to make it more complex to crack.

C. Extended Substitution Cipher

As we have seen that in traditional substitution cipher we are having only **one substitution table** that let people easily do frequency analysis. In this version of substitution cipher, we are going to use **multiple substitution table** and all the table are randomly generated each time with the help of password.

If we look at this cipher abstractly, we will see that it is similar to substitution cipher. It takes **KEY** and **PLAINTEXT** as input and outputs **CIPHERTEXT**.

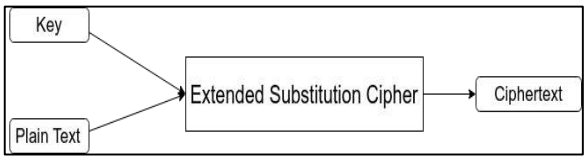

Fig No 1 Extended Substitution Cipher

The only thing which is different is the format of the key. As we have seen in substitution cipher we have to pass 26 letter key but in **ESC (Extended Substitution Cipher)** the key has **two parts**. The first part is used as a **seed value** and the second part is **'n'** which is used to generate n different substitution cipher which will be used for encrypting the plain text. Both parts are **separated using a period (".").**

KEY FORMAT
<seed:[int/float/string]>.<no_of_lists:[int]>

Fig No. 2 Key Format

After understanding it on a broader level, let's dive a little close to the implementation.

D. Character range

Characters
string.ascii_letters + string.digits + string.punctuation + " " 26 * 2 + 10 + 32 + 1 = 95

Fig No. 3 figure range

In a substitution cipher, the characters which are used is only 26 and it is very short. Another issue is that space (" ") character is escaped and that reveals the length of each word and makes the attacker **easy to guess the two or three letter words**. i.e. - the, is, are, am, etc. The character **range of Extended Substitution Cipher is 95** which is approximately **four times** of classical substitution cipher and **space (" ") character is also encrypted** which makes it more secure by stopping the attacker from knowing each word's length.

E. Generating "n" random substitution table

Using the seed value passed with the key we shuffle all the characters and save it in the list. By repeating these process n times, we make n list to be used further. That's why the key we pass is a **combination of seed and n** so that we don't have to pass n substitution table for encryption. It will be **generated at the time of encryption/decryption**.

F. The final encryption

Now we have plaintext which will be encrypted and *n* substitution table. For every i^{th} letter in plaintext it will be substituted with other letter using the i^{th} substitution table. As we know, the number of substitution table can be less than the length of plaintext. In that scenario, instead of using the i^{th} substitution table we use **(i mod n)th substitution table** that will prevent it from overflowing. let's look at one example to have a more unobstructed view: -

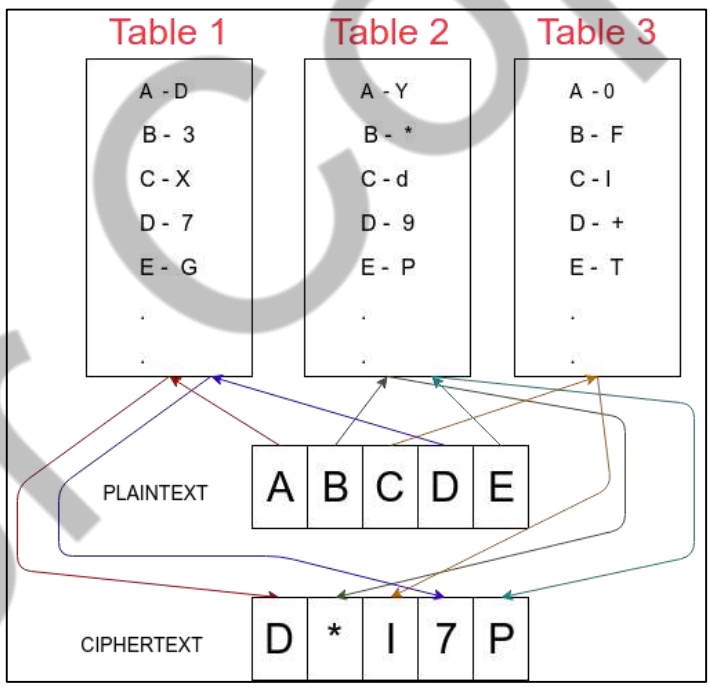

Fig No 4. Proposed Model

Example:

Input
Plaintext–ABCDE
Key – secret.3

Output
Ciphertext - D*I7P

Here the seed value is **'secret'** and the number of substitution table is **3**. Using the seed value all the characters are shuffled in 3 different way making 3 pseudo-random lists.

The first character is **'A'** which goes to **'Table 1'** and substituted with character **'D'**.

The second character is **'B'** which goes to **'Table 2'** and substituted with character **'*'**.

The third character is **'C'** which goes to **'Table 3'** and substituted with the character **'I'**.

The fourth character is **'D'** which goes to **'Table 1'** and substituted with character **'7'**.

The fifth character is **'E'** which goes to **'Table 2'** and substituted with character **'P'**.

G. Decryption

Decryption is as simple as encryption. We will do the same thing but this time instead of substituting the right side of the character using the left side of the character we will do the reverse. We will substitute the left side of the character using the right side of character.

H. Pseudocode:

```
# space charachter is also included
# in all_char at the end but not visible
all_char<-abcdefghijklmnopqrstuvwxyz+
          ABCDEFGHIJKLMNOPQRSTUVWXYZ+
          0123456789!"#$%&'()*+,-./:;<=
          >?@[\]^_`{|}~

# key Format
READ key
GET seed from key
GET n from key

# seed is set to PRNGs for generating the
# same set of random list everytime
SET seed for PRNG
list_of_rand_list<-GENERATE n Shuffled list
                          using the PRNG

function substitute(array A, array B, char C) {
     try:
          index <- get index of C in array A
          return B[index]
     except:
          return C
}

ciphertext <- ""

for i, letter in enumerate(text):
     temp <- substitute(all_char,
                          list_of_rand_list[i%n],
                          letter)
     ciphertext <- ciphertext+temp

OUTPUT ciphertext


plaintext=""
for i, letter in enumerate(text):
     temp<-substitute(list_of_rand_list[i%n],
                          all_char,
                          letter)
     plaintext<-plaintext+temp
OUTPUT plaintext
```

I. *Installation*

We have implemented it in python. You can use it by installing it via pip.

pip install exsc

J. Usage

For encryption:

```
>> from exsc import Extended SubCipher
>> plain text = "This is a test message."
>> obj = ExtendedSubCipher("secret.323")
>> cipher text = obj.encrypt(plaintext)
>> print (ciphertext)
```

For decryption:

```
>> from exsc import ExtendedSubCipher
>> ciphertext = "ld&shf^$gdfk68df63r)(^)"
>> obj = ExtendedSubCipher("secret.323")
>> plaintext = obj.decrypt(ciphertext)
>> print(plaintext)
```

K. Attacks:

Brute force attack

Even substitution cipher cannot be attacked by this. This is more secure than the classical version. Breaking complexity of Extended Substitution Cipher is **(95!)"** which is quite **impossible to crack.**

Frequency analysis attack

Frequency analysis of plaintext

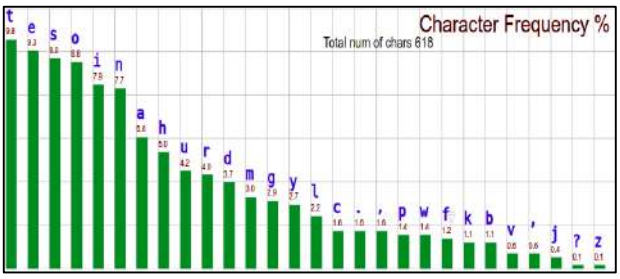

Fig No. 5 Frequency Analysis of Plain Text

Above graph shows the frequency of the plaintext – paragraph from Kafka on the shore

't' is the most frequent character with **9.8%**. Second most frequent character is 'e' with **9.3%**.

Frequency analysis of cipher text encrypted using substitution cipher

Plaintext – paragraph from Kafka on the shore
Key – oljbpxhrstiuvwqfymzceankdg

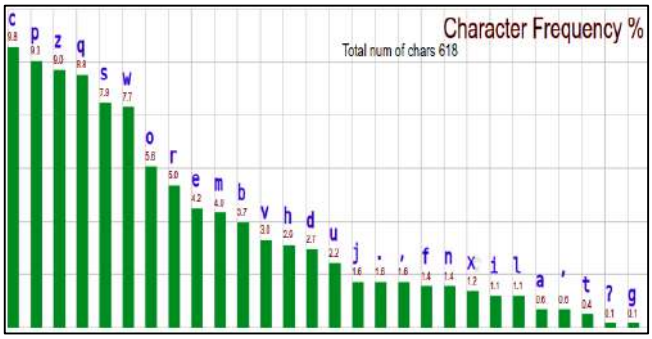

Fig No 6. Frequency Analysis of Ciphertext using Substitution Cipher

In this graph, here the most frequent letter is 'c' with **9.8%**. Second most frequent character is 'p' with **9.3%**. If you see closely, you will notice that the **percentage is identical** only characters are different. That means we can decipher it without the key very easily.

Frequency analysis of cipher text encrypted using extended substitution cipher

Plain text – paragraph from kafka on the shore
Key – secret.576

Encrypted using python version – 3.7.3 on Manjaro Linux

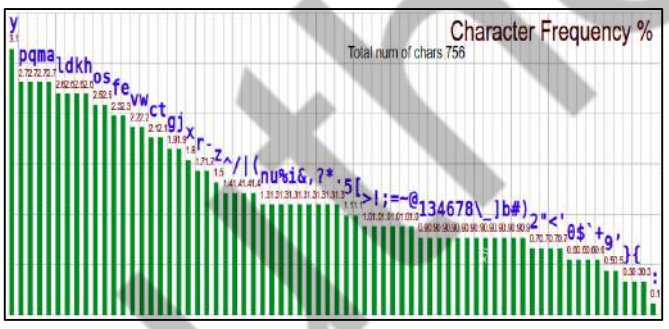

Figure No.7. Frequency Analysis of Cipher Text using Extended Substitution Cipher

In the graph, here the most frequent letter is 'y' with **merely 3.1%**. Second most frequent character is 'p' with **2.7%**. The graph is very **different and spread uniformly**. So frequency analysis attack is useless on this cipher.

IV. CONCLUSION

"A cryptosystem should be secure even if everything about the system, except the key, is public knowledge." - Auguste Kerckhoffs.

As per results, Extended Substitution cipher gives the best result. By using Extended Substitution cipher, it's difficult for an attacker to get seed value and generate plain text as compared to the previous method. Graph generated by ESC is uniformly distributed. As we increase the number of tables, the results will be more and more accurate.

Advantages over classical Substitution cipher

- Space character is also encrypted in Extended Substitution Cipher but not in classical substitution cipher which makes hard to guess the word length.
- Brute force attack in classical one takes *26!* key search but in exsc it takes $(95!)^n$ key search which is much more hard to crack.
- Frequency analysis attack is not possible in exsc.
- Key length is smaller than the classical substitution cipher in spite of being very secure.

Drawback

We have to do process overhead for generating ***n*** substitution cipher which will be used for encryption.

- If ***n*** is very large, then it will make this algorithm very slow but will make it very much secure.
- If ***n*** is very small, then it will make this algorithm very fast but will leave it unsecure.

Here choosing ***n*** cleverly is very important so that you neither make it slow nor unsecure.

Also, if any character which is not alphanumeric or special character or space character then it will be left unencrypted. It can easily be fixed by adding those character in the *all_char* variable.

Some extra caution

If you are using Python for implementation use the same version of python to get the same random value. **Seed generates different value due to version differences.**

Use a PRNG whose **range is very long**. In this implementation we have used **Mersenne Twister** whose range is very large.

Make n (number of substitution table) **as large as possible** that will make it more secure.

REFERENCES

[1] Rajput A.S., Mishra N., and Sharma S.,-Towards the growth of image Encryption and Authentication Schemes‖, (ICACCI), 2013.

[2] Bassem Bakhache, Safwan El Assad,Improvement of the Security of ZigBee by a New Chaotic Algorithm‖, IEEE Systems Journal 2013.

[3] Audio-video Crypto Steganography using LSB substitution and advanced chaotic algorithm‖.Praveen. P1, Arun. Narayana Gurukulam College of Engineering, Kadayiruppu, Ernakulam, Kerala .

[4] Abboud, G.; Marean, J.; Yampolskiy, R.V., "Steganography and Visual Cryptography in Computer Forensics," Systematic Approaches to Digital Forensic Engineering (SADFE), 2010 Binary Images," Innovative Computing, Information and Control,2006, ICICIC '06.

[5] Yogita Verma1, Neerja Dharmale2, 1M Tech Scholar Digital Electronics RCET Bhilai, India 2Assistant Professor (ET&T) RCET Bhilai, India, A Survey Paper Based On Image Encryption and Decryption Using Modified Advanced Encryption Standard, International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Index Copernicus Value (2013)

[6] Sneha Ghoradkar, Aparna Shinde,Review on Image Encryption and Decryption using AES Algorithm, International Journal of Computer (0975 –8887) National Conference on Emerging Trends in Advanced Communication Technologies (NCETACT-2015

[7] Muhammad Usman_, Irfan Ahmedy, M. Imran Aslamy, Shujaat Khan_ and Usman Ali Shahy, Faculty of Engineering Science and Technolog, SIT: A Lightweight Encryption Algorithm for Secure Internet of Things, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 8, No. 1, 2017.

[8] K.Senthil, K.Prasanthi, R.Rajaram Department of Computer Science and Engineering Vickram College of Engineering, Enathi 63056

[9] Jayanta Kumar Pal, J. K. Mandal, "A Novel Block Cipher Technique Using Binary Field Arithmetic Based Substitution (BCTBFABS)" Second International conference on Computing, Communication and Networking Technologies, 2010.