A Hybrid Cryptographic Scheme of Modified Vigenère Cipher using Randomized Approach for Enhancing Data Security

Sazzad Hossain Saju
Dept. of Computer Science and Engineering
Hajee Mohammad Danesh Science and Technology University, Dinajpur, Bangladesh

Sayed Mahmudul Haque
Dept. of Computer Science and Engineering
Hajee Mohammad Danesh Science and Technology University, Dinajpur, Bangladesh

Liakot Hossain Lingcon
Dept. of Computer Science and Engineering
Hajee Mohammad Danesh Science and Technology University, Dinajpur, Bangladesh

ABSTRACT
Vigenère cipher is an ancient elementary method that uses a series of Caesar shifting for encrypting plaintext to ciphertext protecting it from adversaries. However, a frequency analysis attack is vulnerable in this type of cipher technique. This paper aims to enhance the security of the Vigenère cipher in a hybrid cryptosystem using a randomized approach including pseudorandom substitution, rearranging, padding, etc. In this approach, a complex key generation algorithm is used that generates a pair of subkeys from an input key. One of the subkeys, Subkey1 is used to generate an intermediate ciphertext. The other key, Subkey2 is used to further scramble the ciphertext. The slightest change in the input symmetric key produces an entirely different key pair. Hence, different encrypted results. It also increases the ranges of characters that Vigenère cipher can encrypt by including all ASCII and extended ASCII. Finally, the symmetric key is encrypted with the public key of RSA solving the key distribution problem of the symmetric cipher. This enhanced and modified Vigenère cipher in the hybrid cryptosystem overcomes all limitations of classical cipher and acts as a bridge between classical and modern cryptography.

General Terms
Cryptography, Security, Algorithm, Encryption

Keywords
Caesar Cipher, Vigenère Cipher, Encryption, Decryption, Key, Plaintext, Ciphertext, Cryptanalysis, RSA, Security, Symmetric, Asymmetric

1. INTRODUCTION
Cryptography is "the science and study of secret writing." as defined by Yamen Akdeniz. The most significant reason for using cryptography is to preserve confidentiality. So that, anyone except the authorized person cannot read the message. Other importances are preserving integrity, non-repudiation, and authentication [1]. From ancient times to the modern era, cryptography is a big deal. Around 100 BC, Julius Caesar developed the Caesar cipher to send commands to his generals in the field. Ancient ciphers were concerned with only alphabets that could provide a minimum level of security at that period. In 1553 Vigenère cipher was invented that encrypt alphabetic text by using a series of interwoven Caesar cipher. Historically, many similar classical ciphers were invented that can be practically computed and solved by hand. Cryptography gets its most attention, especially in the modern era. During WWII, the cryptanalysis of Enigma and Lorenz SZ machines lead to the development of modern electronic computers. Because it required very fast simultaneous calculation to perform a brute force attack and theoretically cannot be computed with the human hand in a measurable time. At that period, Claude Shannon proved that a one-time pad cipher is unbreakable when the key material is truly random and of equal or greater length than the message [2]. Then with the invention of the internet, cryptography is not only a concern for the government and scientists but all of us. From mail, cash transactions, file encryption, password manager, multilayer password protection, many protocols like HTTPS, WPA2, TCP/IP are using cryptography on daily basis. Due to a wide range of applications, modern cipher owns a very complex infrastructure because the attacks from the intruder were stronger than ever before with the increasing computational power of computers. So classical ciphers must have their modifications and many researchers have been working on that. Modern ciphers like AES, DES, or Blowfish have output ranges from ASCII 0 to 255. But these are symmetric-key ciphers. Symmetric ciphers are faster than asymmetric ciphers in encrypting a large volume of data. But they have a key distribution problem. While asymmetric ciphers like RSA have two keys: public and private. A public key is publicly announced, so anybody can encrypt using it. On the other hand, a private key is kept secret and used to decrypt. Also, the keypair is very large and must have to store somewhere while the symmetric key is usually simple enough to remember. So Many hybrid cryptosystems are now combining these symmetric-key and asymmetric-key ciphers where the symmetric-key is encrypted with the asymmetric-key before transmitting [3]. This paper presents a new hybrid cryptosystem with the combination of the enhanced and developed version of Vigenère cipher and RSA. With asymmetric-encrypted key sharing and randomized symmetric approach, the cryptosystem is convenient to consider as a fast and reliable method.

2. CLASSICAL METHODS

2.1 Caesar Cipher
Caesar Cipher or Caesar shift uses a constant for shifting letters to evolve the encrypted text. It is a monoalphabetic substitution i.e. the replacement is the same for the entire text to be encrypted. To encrypt a letter X, it uses encryption formula: En (X) = (X + n) mod 26 and for decryption, the algorithm uses decryption formula: Dn (X) = (X – n) mod 26. Here, X is the position of a letter and n is constant. The operation is performed on the overall 26 letters of the English

alphabet. Here is an example of classical Caesar Cipher using a constant n=7,

Plain text:
MANYCIPHERSHAVEBEENDEVELOPEDTOPROVIDED ATASECURITY

Cipher text:
THUFJPWOLYZOHCLILLUKLCLSVWLKAVWYVCPKL KHAHZLJBYPAF

2.2 Vigenère Cipher

Vigenère cipher is a polyalphabetic substitution technique that encrypts alphabetic text. It uses a series of Caesar shifting based on the letters of a keyword. To perform this, the keyword has to repeat to be the same size as the plaintext. An example of encryption is given in Fig.1:

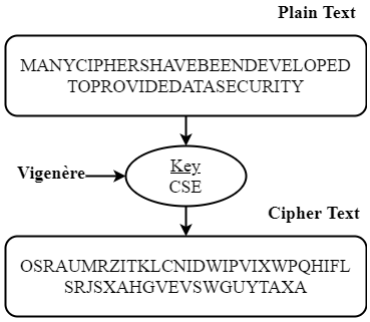

Fig 1: An example of encryption using Vigenère cipher

The ciphertext is produced by using a variable instead of a constant in the Caesar cipher equation. Letters are converted as A=0, B=1, C=2 …. Z= 25 and the variable changes based on the letters of the keyword. So Vigenère cipher must have a minimum of 2 key characters. Otherwise, there will be no difference with Caesar cipher. With a longer keyspace, it was believed at that period that the Vigenère cipher is indecipherable because the key materials are a random choice by the user. But the distribution of letters in a typical sentence of the English language has a very distinct and predictable shape. Therefore, ciphertexts generated with Vigenère cipher are prone to be broken easily using frequency analysis attack, brute force or exhaustive search, etc. Here, in Table-1 letter frequencies in the English language are given. This can be used to perform a frequency analysis attack on the Vigenère cipher.

Table 1. Letter frequency in the English language [4]

Letter	Frequency	Letter	Frequency
a	8.167%	n	6.749%
b	1.492%	o	7.507%
c	2.782%	p	1.929%
d	4.253%	q	0.095%
e	12.702%	r	5.987%
f	2.228%	s	6.327%
g	2.015%	t	9.056%
h	6.094%	u	2.758%
i	6.966%	v	0.978%
j	0.153%	w	2.360%
k	0.772%	x	0.150%
l	4.025%	y	1.974%
m	2.406%	z	0.074%

Here the plaintext shown in Fig.1 is a general English sentence hence the frequencies are corresponding to the table where the letter “E” has the highest frequency. Whereas in ciphertext, the letter “I” has the highest frequency. So “I” corresponds to “E”. Here, E=4 and I=8. So the distance is 4. Again 4=E, so the letter is also "E" in the keyspace. Therefore, it is clear that frequency analysis attack is vulnerable for Vigenère cipher. Also, the frequency analysis can be used as an essential factor in a genetic algorithm for the cryptanalysis of the Vigenère cipher. The genetic algorithm measures fitness for 20 randomly selected keys and searches the keyspace without necessarily knowing the key [5].

3. RELATED WORK

There has been an enormous amount of research on cryptography to gain a higher level of security. Dr. V. Kapoor and R. Yadav proposed a hybrid cryptography technique to support cybersecurity infrastructure. This paper describes various aspects of cryptography as well as a proposed work addressing some of the core issues of cryptography [3]. A randomized approach was made by A. Jain, R. Dedhia, and A. Patil for enhancing the security of the Caesar cipher. It overcomes the limitations faced by the classical Caesar cipher by using a complex key generation technique. Instead of using a single key, this method generates two keys from the input key providing enhanced security [6]. Bhardwaj C. gives a modification of Vigenère cipher by random numbers, punctuations, and mathematical symbols. He used randomly generated numbers instead of the alphabet for the encryption key [7]. Md. P. Uddin developed a cryptographic algorithm based on ASCII conversion and a cyclic mathematical function. In this method, the original message is divided into packets binary matrices and each packet produces unprintable characters. So the encrypted result contains only control characters [8]. R.K. Singh developed a character jumbling text encryption method that varies with an odd or even number of characters. It is a lightweight encryption method that does not require any key from the user. The key is randomly generated inside the cryptosystem [9].

4. PROPOSED ALGORITHM

4.1 Encryption

In the encryption algorithm, a plaintext message M is an input from the user. Mlen is the message length. Mlen becomes an argument to the KeyGeneration function that returns a key pair: Subkey1 and Subkey2. The intermediate ciphertext C is produced from the operation of Subkey1 and M. For padding, the tilde characters are appended with C to be a multiplication of 8. After that, C is rearranged utilizing Subkey2. Finally, C is converted to hexadecimal format and the tilde characters are replaced with random control characters and/or extended ASCII based on the positions. The flowchart of the encryption algorithm is presented in Fig2.

Pseudocode:

1. Input the plaintext message M.
2. Call KeyGeneration(Mlen) function to generate a pair of subkeys: Subkey1, Subkey2.

3. Intermediate ciphertext C is produced using formula: C[i] = (M[i] + Subkey1[i]) mod 94 for i=0 to Mlen.
4. Keep adding tilde character (~) after C until C.size() is not a multiplication of 8.
5. For i = C.size() – 1 to 0
    a. Generate a random number j, using j = Subkey2[i] mod C.size()
    b. Exchange C[i] with C[j]
6. Convert C to hexadecimal format.
7. For i = 0 to C.size()
    a. Substitute tilde character (7E) with random control character (CC) when the position (i/2) is even and substitute with extended ASCII character (EA) when the position is odd.
8. Transmit the ciphertext generated in step 7.

4.2 Decryption

In the decryption algorithm, the user inputs the ciphertext C. A variable pad_char indicates padding characters i.e. number of control characters and/or extended ASCII contained in C. From pad_char the original message length, Mlen is determined which becomes an argument to the KeyGeneration function. The key Generation function returns two subkeys: Subkey1 and Subkey2. Then the padding characters are replaced with tilde characters (7E) and the ciphertext is converted to the base format. Subkey2 inversely scrambles the ciphertext and the result is the intermediate ciphertext C. Finally, the operation of the C and Subkey1 reveals the original message M. The flowchart of the decryption algorithm is presented in Fig3.

Pseudocode:

1. Input the ciphertext C
2. Calculate padding characters or pad_char and get original message length, Mlen = C.size()/2 – pad_char.
3. Call KeyGeneration(Mlen) function to generate a pair of subkeys: Subkey1, Subkey2.
4. Substitute the padding characters with tilde characters (7E) in C.
5. Convert C to base format.
6. For i = 0 to C.size()
    a. Generate a random number j, using j = Subkey2[i] mod C.size()
    b. Exchange C[i] with C[j]
7. If tilde character (~) appears in C
    a. Shrink ciphertext length length by erasing where the tilde character begins.
8. Original message M is revealed using formula: M[i] = (C[i] – SubKey1[i]) mod 94 for i=0 to C.size().

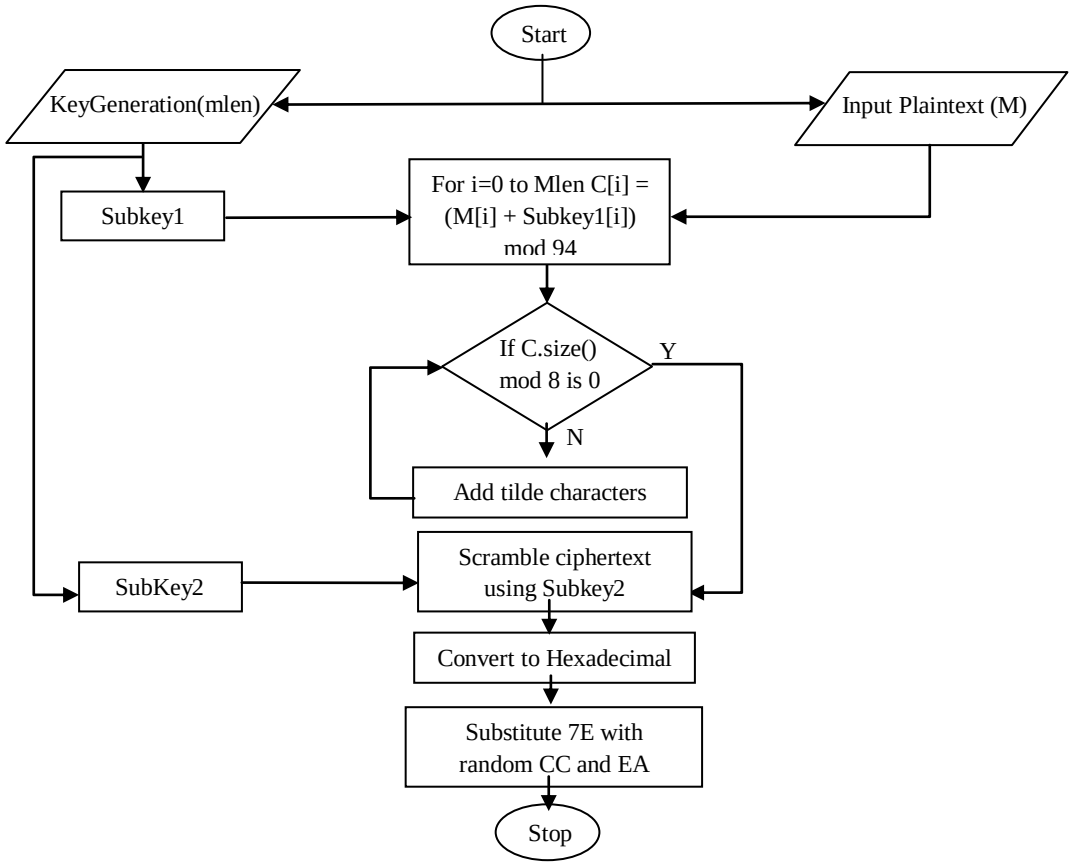

Fig 2: Flowchart of Encryption Algorithm

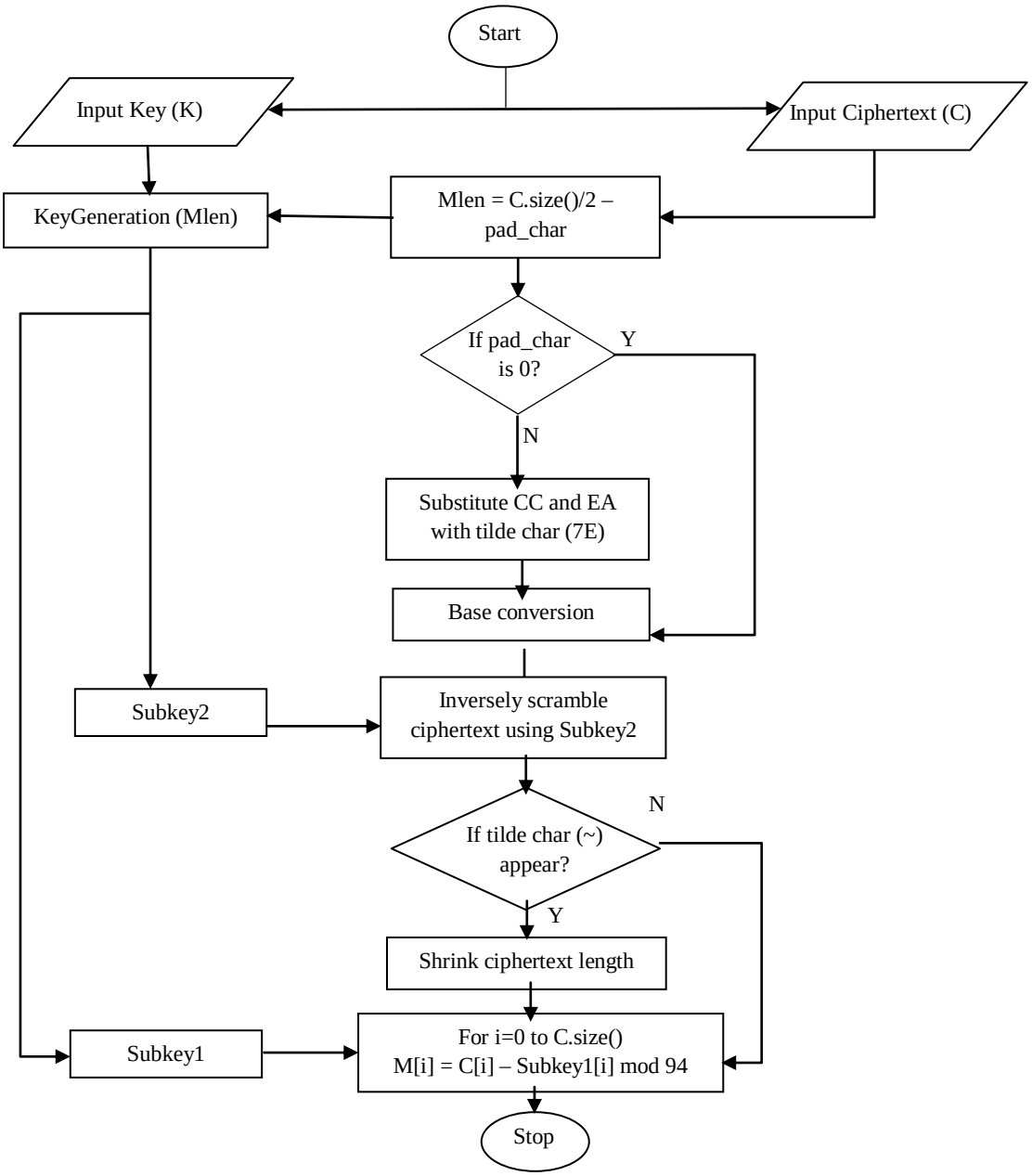

Fig 3: Flowchart of Decryption Algorithm

4.3 KeyGeneration(Mlen)

The KeyGeneration algorithm takes an input key K from the user. Message length or Mlen is an argument to the function. The input key, K becomes an argument to another Txt2Int function. This function returns a random number Seed generated from K. Then the input key is enhanced to be the same size as the message. The enhanced key is substituted using the random number Seed. Then the substituted key is rearranged and that is the Subkey1. If Subkey1 is not a multiplication of 8, then it is further repeated and again substituted with random characters. Finally, Subkey2 is already generated from Subkey1. So, the algorithm returns the keypair.

Pseudocode:

1. Input a key K
2. Get a random number, Seed = Txt2Int(K) + Mlen
3. While (K.size() ≠ Mlen) repeat K
4. For i=0 to K.size() do substitution
   Seed = (K[i] + Seed) Mod 95
   Temp = K[i] + Seed
   if(Temp> 126) Temp = Temp – 95
   K[i] = Temp
5. For i=0 to K.size() do rearranges
   srand(Seed)
   j = rand() Mod K[i] Mod K.size()
   exchange K[i] with K[j]
6. Subkey1 = K
7. While (Subkey1.size() mod 8) is not equal to 0
   Subkey2 = repeate Subkey1
8. For i=0 to Subkey2.size() do substitution
   Seed = (Subkey2[i] + Seed) Mod 95
   Temp = Subkey2[i] + Seed
   if(Temp>126)Temp = Temp – 95
   Subkey2[i] = Temp
9. Return Subkey1, Subkey2.

4.4 Txt2Int(K)

1. Initialize Seed =0, grd=1
2. For i=0 to K.size() do
   Seed = (Seed+K[i]) * grd
   grd *= 2
3. Return Seed

4.5 Existing Method

RSA is an asymmetric key encryption technique where two prime numbers are taken initially and then the product of these values are used to create a public and a private key. The symmetric key is encrypted with the public key at the end of the symmetric encryption process. Similarly, at the beginning of the symmetric decryption, the private is used to reveal the original symmetric key.

Pseudocode:

1) Select two large prime numbers, p, and q. Let n= p.q
2) Calculate Euler's Totient: $\varphi(n) = (p-1)(q-1)$
3) Find a random number e such that $1<e<\varphi(n)$ and relatively prime with $\varphi(n)$ i.e. $\gcd(e,\varphi(n)) = 1$
4) Calculate a number d such that $d = e^{-1} \bmod \varphi(n)$
5) Encryption: Given a symmetric key K in plaintext then the ciphertext $CK = K^e \bmod n$
6) Decryption: The encrypted symmetric key is decrypted by the formula: $K = CK^d \bmod n$

5. Explanation with Example

5.1 Encryption of the Plaintext

Following the developed symmetric encryption algorithm step by step, the outcome is an encrypted string in hexadecimal-format for the example taken below:

Plaintext M: qwerty is my facebook password

Key K: Qwerty12

1) First step is to get the message length or Mlen from the input message M.

Message length or Mlen = 30

2) In the second step, Mlen is passed as an argument to the KeyGeneration function. The KeyGeneration function takes an input key K from the user. Then it returns the following keypair where Subkey1 has the same size as M and Subkey2 have the size of multiplication of 8.

Subkey1 = 3.,)v`W_v@IhEm75VdVN m;<*D#My

Subkey2 = }LxBYZRRiJ4=#1h>5:1 @N*f1u9'Aa5c

3) Third step produces the intermediate ciphertext C from the operation of M and Subkey1. The ciphertext C contains anything in the readable ASCII characters except the tilde character. The tilde character is reserved for padding. Here is the process for the first character in C:

M[0] = q = 113 (ASCII)

Subkey1[0] = 3 = 51 (ASCII)

Temp = (113+51) mod 94 = 70 = F (ASCII)

C[0] = F

So, q → F, Similarly,

Intermediate C = FG3=.{wj-`X%eu::]hg_-/M??Y<^/&

4) Until the length of the intermediate C is not a multiplication of 8, then tilde (~) characters are appended to it in the 4th step. Here C.size() = 30. The nearest number 32 is a multiplication of 8. So (32-30) = 2 tilde characters are appended to C. This is enhanced C.

Enhanced C = FG3=.{wj-`X%eu::]hg_-/M??Y<^/&~~

5) Step-5 scrambled the enhanced ciphertext C using the Subkey2. Process begins from end to the beginning of C. For each character in C of indexed i, a random position j is generated ($j \leq$ C.size()). Here is the first iteration:

i = C.size() – 1 = 32 – 1 = 31

j = Subkey2[i] mod C.size() = Subkey2[31] mod 32 = 99 mod 32 = 3

Exchange C[31] with C[3]. Similarly,

Scrambled C = %~h3~g^?F<`G/Y-/:u?-M]Xwe.{j&_:=

6) The resultant C from step-5 is converted to hexadecimal format giving the following values:

Hexadecimal conversion of C =
257E68337E675E3F463C60472F592D2F3A753F2D4D5D58
77652E7B6A265F3A3D

7) The hexadecimal converted C contains 7E that is the tilde character. Here are two tilde character at position 2/2 = 1 and 8/2 = 4. So the first tilde character is replaced with random extended ASCII for the odd position and the second tilde character is replaced with random control character for the even position.

Final ciphertext C =
25A3683301675E3F463C60472F592D2F3A753F2D4D5D58
77652E7B6A265F3A3D

At the end of the symmetric encryption process, the asymmetric encryption begins to encrypt the symmetric key. Here is the process for the taken symmetric key K:

1) Two big prime numbers (p and q) and an auxiliary number (e) are chosen where $\gcd(e,\varphi(n)) = 1$. Then the public key is (n,e). Here n = p.q and $\varphi(n) = (p-1) * (q-1)$. Consider,

p = 759902534011993492390886979244737626978083

q = 14624287353165479746453426 09

e = 789063169

Public key (n,e) =
(111130330177899972597479708618738371024191819140641274576834686903854 7, 789063169)

2) Now the symmetric key is encrypted using formula $CK = K^e \bmod n$. K is the symmetric key.

Symmetric Key, K = Qwerty12

Encrypted Symmetric Key, CK =
4526994CD2223D17221EDF1FF71F0E65973CFA2DF87DD
65C02733698D

Table 2. The encrypted output produced by the cryptosystem

Encrypted Symmetric Key (CK)	4526994CD2223D17221EDF1FF71F0E65973 CFA2DF87DD65C02733698D
Encrypted Message (C)	25A3683301675E3F463C60472F592D2F3A75 3F2D4D5D5877652E7B6A265F3A3D

5.2 Decryption of the Ciphertext

The process begins with the asymmetric decryption of the encrypted symmetric key CK. Decryption isn't possible without the private key and each public key and private key are unique pairs in RSA.

1) Along with prime number p and q and the auxiliary number e, a new auxiliary number d is determined where d = e-1 mod φ(n). Then the private key is (n,d). Here d is calculated as:

d = 383280002165879030198753522455249095344531737286646454646311124952265 7

Private key (n,d) = (111130330177899972597479708618738371024191819140641274576834686903854 7, 383280002165879030198753522455249095344537372866464546463111249522657)

2) Now the encrypted symmetric key is decrypted using the formula $K = CK^d \bmod n$.

Encrypted Symmetric Key, CK = 4526994CD2223D17221EDF1FF71F0E65973CFA2DF87DD65C02733698D

Symmetric Key, K = Qwerty12

After the symmetric key is retrieved, the symmetric decryption can begin to reveal the encrypted message C. Here is the step by step decryption processes explained for the taken example:

1) User inputs the ciphertext C.

Ciphertext, C = 25A3683301675E3F463C60472F592D2F3A753F2D4D5D5877652E7B6A265F3A3D

2) In step-2, the number of padding characters are determined. From this, the original message length or Mlen is calculated as follows:

pad_char = 2 (A3, 01)

Mlen = C.size()/2 – pad_char = 62/2 – 2 = 30

3) Mlen becomes an argument to the KeyGeneration function and it takes the input key, K from the user. Then returns the same keypair as encryption:

Subkey1 = 3.,)v`W_v@IhEm75VdVN m;<*D#My

Subkey2 = }LxBYZRRiJ4=#1h>5:1 @N*f1u9'Aa5c

4) Step-4 replaces the padding characters of ciphertext C with the tilde character (7E).

Substitute C = 257E68337E675E3F463C60472F592D2F3A753F2D4D5D5877652E7B6A265F3A3D

5) The ciphertext C is then converted in base format and that is scrambled ciphertext.

Scrambled C = %~h3~g^?F<`G/Y-/:u?-M]Xwe.{j&_:=

6) Step-6 rearranges the characters of the ciphertext in proper position using Subkey2. Process starts from beginning to the end of C. After the end of iteration the result is enhanced C.

First, i = 0 and

j = Subkey2[i] Mod C.size() = Subkey2[0] mod 32 = 125 mod 32 = 29

So, C[0] exchanged with C[29]. Similarly,

Enhanced C = FG3=.{wj-`X%eu::]hg_-/M??Y<^/&~~

7) Step-7 removes the tilde character from C if there is any. The result is intermediate ciphertext C.

Intermediate C = FG3=.{wj-`X%eu::]hg_-/M??Y<^/&

8) Finally, the original message M is revealed from the operation of the intermediate C and Subkey1. Here is the process for the first character in M.

C[0] = F = 70 (ASCII)

SubKey1[0] = 3 = 51 (ASCII)

Temp = (70-51) mod 94 = 19

Temp < 32, So Temp += 94

M[0] = Temp = 113 = q

So, T → q. Similarly,

Message M = qwerty is my facebook password

Table 3. The decrypted output produced by the cryptosystem

Decrypted Symmetric Key (K)	Qwerty12
Decrypted Message (M)	qwerty is my facebook password

5.3 KeyGeneration(Mlen)

The KeyGeneration function is the same in both encryption and decryption. It returns a key pair generated from an input key chosen by a user. Here the key generation is explained for the key used in the encryption and decryption example above.

1) The user chooses a strong password as the input key. Here an assumption,

K = Qwerty12

2) The Txt2Int function takes K as an argument and returns an integer which is summed up with the message length or Mlen. That is a random number Seed,

$\text{Txt2Int(K)} = \text{int(Q)}.2^0 + \text{int(w)}.2^1 + \ldots + \text{int(1)}.2^{\text{K.Size()}-2} + \text{int(2)}.2^{\text{K.Size()}-1} = 16899$

Seed = Txt2Int (K) + Mlen = 16899 + 30 = 16929

3) The Key is repeated until K.Size() ≠ Mlen. Here K.Size() = 8 and and Mlen = 30. Hence the result from step-3 is,

Enhanced, K = Qwerty12Qwerty12Qwerty12Qwerty

4) Step-4 performs a pseudorandom substitution of K using the random number Seed. Here is the process for the first character in the substituted key

K[0] = Q = 81 (ASCII)

Seed = 16929

Seed = (Seed + K[0]) mod 95 = (16929 + 81) mod 95 = 5

Temp = K[0] + Seed = 81 + 5 = 86 = V (ASCII)

Q → V, Similarly,

Substituted, K = V5)I` h<M, @Wv_3D#v7NmV*;ym.Ed

5) In step-5, the substituted K is shuffled randomly to produce a further scrambled key-stream. The updated value of Seed is 74 from step-4. This is used to indicate a random number in a pseudorandom sequence.

srand(Seed) = srand(74)

For, i = 0

K[i] = K[0] = V = 86 (ASCII)

K.Size() = 30

j = rand() mod K[0] mod K.Size() = 22

So, K[0] exchanged with K[22]. Similarly,

Random indexed K =

3.,)v`W_v@IhEm75VdVN m;<*D#My

6) The result from step-5 becomes SubKey1.

SubKey1 = 3.,)v`W_v@IhEm75VdVN m;<*D#My

7) The length of Subkey1 is not a multiplication of 8. So it is enhanced and becomes an intermediate Subkey2:

Int. Subkey2 = 3.,)v`W_v@IhEm75VdVN m;<*D#My 3.

8) A random substitution is performed for Subkey2 similar to step-4. Here the Seed is initially 74 and this produces the following Subkey2:

Subkey2 = }LxBYZRRiJ4=#1h>5:1 @N*f1u9'Aa5c

9) The generated Subkey1 and Subkey2 get returned from the KeyGeneration function. Table-4 exhibits the results for the taken input key K.

Table 4. The result from the KeyGeneration function for the example taken

Input Key (K)	Qwerty12
Subkey1	3.,)v`W_v@IhEm75VdVN m;<*D#My
SubKey2	}LxBYZRRiJ4=#1h>5:1 @N*f1u9'Aa5c

6. RESULT AND DISCUSSION

The result from the cryptosystem is very sophisticated with fast processing. The symmetric-key algorithm was simulated in the C++ programming language shown in Fig.4 and the RSA algorithm was implemented in Python for performing calculations over large integers. Both algorithms give encrypted results in the hexadecimal format and contain anything from ASCII and extended ASCII. Although it is possible to discriminate between the two from the analysis of the outputs. But the cryptanalysis will require both knowing the cipher technique and the key used.

In the symmetric-key cipher, if the input key changes slightly, the output changes significantly because of the different key pairs. This is called the avalanche effect is shown in Table-5. The minimum input key length takes 8 characters and the minimum encrypted string is 16 digits. Also, the output is a multiplication of 16. Thus it hides the actual message length. The system robustness is also found in the scrambling of the ciphertext using Subkey2. So after base conversion, the result is still scrambled. Instead of using a small repeated key, the applied keypair exhibits more randomness in outputs over some existing systems. An enhanced Caesar cipher produces an encrypted string in only readable ASCII and also the original message length can be found directly from the encrypted output length. No standard password requirement is followed and it applied a fixed substitution using a table [6]. Here in this paper, the enhanced and modified Vigenère cipher overcomes these limitations as well as removes the drawbacks of classical ciphers.

Fig 4: Simulation of the developed symmetric-key encryption and decryption process

In the RSA algorithm, two big prime numbers are taken, and the product of these prime numbers is needed to be at least more than 1 million. Otherwise, factorization can be possible within measurable periods that would reveal the private key. There is no efficient algorithm except Euclid's algorithm for factorization. So much bigger the number provides much more security. But even if the product is bigger but one of the prime numbers is small, then a simple brute force attack reveals the two primes. So both keys are needed to be bigger and they also have to have a bigger distance. Otherwise, a small difference is vulnerable too [10]. However, when a user avoids these vulnerabilities, the RSA algorithm provides reliable security and solves the key distribution problem of the symmetric-key algorithm.

Table 5. Avalanche effect in Key Generation of the developed symmetric cipher when Mlen=30

Input Key (K)	**Qwerty12**	**Qwerty21**
Subkey1	3.,)v`W_v@IhEm75VdVN m;<*D#My	M cBQ-hv*ZH6?!TulK6w-$d?Zm_H9V
SubKey2	}LxBYZRRiJ4=#1h>5:1 @N*f1u9'Aa5c	Ww{^P}'>hcL#b$x/<(^vDhmMHVV?xo]}

7. CONCLUSION

In the ongoing development process of cryptography, a new cipher provides users one more option to choose for encryption. This randomness confuses an attacker in determining the cipher technique. This paper describes the frequency analysis attack on classical Vigenère cipher and proposed an algorithm that uses a randomized approach to produce ciphertext utilizing two keys where the frequency analysis attack has no impact and the key distribution problem is solved with the asymmetric-key cipher i.e. RSA. Also increases the ranges from ASCII to extended ASCII in the ciphertext. A brute force attack would not be efficient because of the avalanche effect. The key has to be the exact same, otherwise there would be no similarities helps in determining possible keys. For password manager or financial data security or any kind of secret writing, the developed cryptosystem can be found to be useful for encrypting small to large text files to store or transmit data securely.

8. REFERENCES

[1] William Stalling. 2017. "Cryptography and network security". (7th ed.). Prentice-Hall. ISBN: 9780134444284

[2] D. Kahn. 1967. "The Codebreakers". The Story of Secret Writing. New York. Macmillan, 1967. ISBN 0-684-83130-9.

[3] Dr. Vivek Kapoor and Rahul Yadav. 2015. "A Hybrid Cryptography Technique to Support Cyber Security Infrastructure" International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 4 Issue 11, November 2015, DOI: 10.5120/ijca2016909863.

[4] Agyepong, Enoch & Buchanan, William & Jones, Kevin. (2018). Detection of Algorithmically Generated Malicious Domain. 13-32. 10.5121/csit.2018.80802.

[5] S. S. Omran, A. S. Al-Khalid and D. M. Al-Saady, "A cryptanalytic attack on Vigenère cipher using genetic algorithm," 2011 IEEE Conference on Open Systems, Langkawi, 2011, pp. 59-64, DOI: 10.1109/ICOS.2011.6079312

[6] A. Jain, R. Dedhia and A. Patil. 2015. "Enhancing the security of Caesar cipher substitution method using a randomized approached for more secure communication", International Journal of Computer Applications, Volume 129, Number 2015, pp. 6-11, DOI: 10.5120/ijca2015907062.

[7] Bhardwaj C. 2012. "Modification of Vigenère cipher by Random Numbers, Punctuations & Mathematical symbols". Journal of Computer Engineering (IOSRJCE) ISSN: 2278-0661 Volume 4, Issue 2 (Sep.-Oct,2012), PP 35-38, DOI: 10.9790/0661-0423538

[8] Md. Palash Uddin, Md. Abu Marjan and N.B. Sadia. 2014. "Developing a cryptographic algorithm based on ASCII conversions and a cyclic mathematical function," 2014 International Conference on Informatics, Electronics & Vision (ICIEV), May 2014, Dhaka, Bangladesh. DOI: 10.1109/ICIEV.2014.6850691.

[9] R. K. Singh, T. Begum, L. Borah, and D. Samanta, "Text encryption: Character jumbling," 2017 International Conference on Inventive Systems and Control (ICISC), Coimbatore, 2017, pp. 1-3, DOI: 10.1109/ICISC.2017.8068691.

[10] Boneh, Dan. (2002). Twenty Years of Attacks on the RSA Cryptosystem. NOTICES OF THE AMS. 46.