Engineering Of A Software Model For Esrkgs Cipher

Uwazie Emmanuel Chinanu, Okonkwo Samuel Ogechukwukama, Joy O. Okah-Edemoh
[1]PhD student, [2]MSc Student, [3]Associate Professor,
[1]Computer Science Department,
[1]Nasarawa State University, Keffi, Nigeria

***Abstract*—Asymmetric-key (or public-key) cryptography with RSA provides some amount of confidentiality and authenticity in data exchange. Several revisions to RSA have been made since its inception, to make the technique more secured. An Enhanced and Secured RSA Key Generation Scheme (ESRKGS) uses the product (N) of four large random prime numbers to generate the Public and Private keys used for Encryption (E) and Decryption (D) respectively. With ESRKGS, encrypting and decrypting a message is more complex than with traditional RSA. This makes ESRKGS secure under the RSA assumption which refers to the hardness of decrypting a cipher text. In this paper, an overview of the ESRKGS is done, and a software is engineered, implementing its functionality. The software was created using C Sharp programming language. C Sharp is an Object Oriented Programming language built for the .Net Framework.**

***Index Terms*—Cryptography, Software, Encryption, Decryption, ESRKGS.**

• Introduction

The term Cryptography comes from two Greek words: Crypto (meaning 'secret') and Graphy (meaning 'writing') [1].
Cryptography is the analysis, design and implementation of mathematical solutions to keep information private from everyone except those whom the information is intended for [6].
Modern cryptography provides Privacy, Authenticity, Integrity and Non-repudiation.

Application of Cryptography

-Crypto currency
-Encrypted Wi-Fi
-Disk Encryption
-Password Encryption
-Secure internet transactions with credit cards
-Digital signatures

Components of Cryptography

Plain text + Algorithm + Key = Cypher text
Optionally, it is:
Plain text + Initialization vector (IV) + Algorithm + Key = Cypher text
Plain text is the raw message you want to secure.
The IV is a pseudo random number that adds randomness to the beginning of the processes so that we get a cypher text that is more unpredictable.
Computer generated random numbers are not truly random, because they are based on some variables like time, date, temperature, keyboard input, etc. so they are pseudo random.
Keys are also called crypto variables-crypto because it is part of cryptography, and variable because it changes randomly. It is usually a number or set of numbers that the algorithm operates on.
A key is the instruction on how to use the algorithm: instructions like which algorithms to use, how many algorithms should be used, how should the algorithm be used, in what order should the algorithms be used.
In a set of functions used for encryption, the key contains the information on which functions to use, the order in which they are used, and the randomness at which they are used.
Keys should have these three attributes: Long, Random and Secret.
A long key does not necessarily mean a secured encryption. A complex algorithm is necessary to make a secured system.
Longer keys demand more processing power, hence lead to less efficiency.
Algorithm is a collection of math functions which substitute one data with another, or adds extra data to an initial data to produce a secure data. It is also called cypher.
When we talk about algorithms, we are looking for three things: Confusion, Diffusion and Openness.
Confusion is concerned with complex mathematics for substitution.
Diffusion deals with the number of permutations (rounds).

Openness is the Kirckhoff's principle which states that algorithms should be open for others to understand it in order to allow for its further development. This principle is not mandatory. The US government does not practice this as she believes the knowledge of an algorithm and its key are the tools needed to eavesdrop a conversation.
Cryptographic Algorithms(Ciphers) are divided into two parts, namely: Symmetric-key cryptography and Asymmetric-key cryptography [2].
Symmetric means same, so for this cryptography, the same key is used both to encrypt and decrypt messages. It is also known as private key, shared key, secret key or session key cryptography. It is secret key because this same key is kept secret between the users in order to avoid attackers making use of it. It is shared because it is used by two parties. It is private key because it is used by just two parties (not everybody) who use it for encryption. It is session key because it is limited based on time to avoid the same key being used over and again. After a given time, the key is replaced with a new one.
Symmetric-key algorithms can either be *block* or *stream.*
Asymmetric-key algorithm is divided into *Discrete Logarithm*s and *Factorization.*

SYMMETRIC CIPHERS
BLOCK CIPHERS: They chunk data into blocks i.e. 64bit block. They are slow but more secure.
Examples are: ECB, CBC, DES, 3DES and Advanced Encryption Standard (AES). AES is the most common and is used by the US government for sensitive but unclassified information. It is efficient. It has a 256bit key. AES is applied to WPA2, SMIME, etc. 3DES, which is a predecessor to AES. It has a 168bit key. It put its data through 3 times the permutations of DES, so it did 48 permutations. This means the encryption process was performed 48 times. It is very processor intensive so it is not efficient, that's why it is not common used. DES chunk its data into 64bit blocks. It put its data through 16 permutations.
STREAM CIPHERS: It works bit by bit, encrypting a single bit at a time. Stream ciphers are fast to encrypt and decrypt but they are less secure. Traditional stream ciphers are Transposition (which involves scrambling the characters) and Substitution (replacing a character with another). Substitution ciphers are either **Monoalphabetic** or **Polyalphabetic**. Monoalphabetic ciphers replace a character with a particular character each time the original character occurs in a message, while polyalphabetic ciphers can replace a character with different characters at different occurrences in a message. A modern stream cipher technique is XOR (exclusive or), which is applied in the most common stream cipher RC-4. RC-4 is used by WEP and WPA.

Merits of Symmetric Ciphers
-Difficulty of key distribution among parties to use the same key
-It is not scalable as each pair of nodes must have unique primary and secondary keys for each party
-Weak Authenticity as proof of the source of an encryption is not clear
-No Integrity of message is guaranteed
-Non Repudiation is not feasible since authenticity and integrity cannot be guaranteed.

Demerits of Symmetric Ciphers
-Symmetric Ciphers are usually fast

Asymmetric CIPHERS
Examples of Asymmetric-key algorithms are RSA and Diffie-Hellman algorithms.

Merits of Asymmetric-key Cipher
-Provides an efficient key exchange
-There is privacy as a sender encrypts with a receiver's public key and the receiver decrypts with their private key which is known to them alone.
-Integrity is provided through the use of a hash (a value used to represent the message sent) through hashing algorithms like MD5, SHA-1 or SHA-256. If the hash decoded by the receiver is different from that of the sender, the message might have been tampered with.
-Authenticity occurs when the sender uses their private key to encrypt a text added to the message. If the receiver successfully decrypts it with the sender's public key, then it is confirmed that the message is from the sender with an exclusive access to the private key.
-It is scalable as every node needs just a private key and a collection of the public keys of other nodes.
-It provides non-repudiation as the sender encrypts a hash (producing a digital signature) which the receiver decrypts with the sender's public key. If the decrypted hash matches the original hash, then the sender of the message is non-deniable.

Demerits of Asymmetric Cipher
-It is slow

- **Review of Literature**

The most popular Asymmetric-Key cipher is the RSA developed at MIT by Rivest, Shamir and Adleman.
The RSA algorithm can be described as follows [5]:

Algorithm 1. RSA GENERATION
KEY GENERATION

INPUT:
Two large prime numbers p and q

OUTPUT:
Public Key Component: {e, n}
Private Key Component: {d, n}

PROCEDURE:
$n = p * q$
/*Compute Euler phi value of n*/
$\varphi(n)=(p-1)*(q-1)$
Find a random number e, satisfying $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$
Compute a random number d, such that,
$d = e^{-1} \bmod (\varphi(n))$

ENCRYPTION
INPUT:
Plain text message, M(<n)
OUTPUT:
Cipher text, C
PROCEDURE:
-A transfers a message (or plain text) M (<n) along with the public keys to B
-B encrypts the message using A's public key e to generate a Cipher text, C.
Cipher Text, $C = M^e \bmod n$

DECRYPTION
INPUT:
Cipher text message, C

OUTPUT:
Decrypted plain text, P

PROCEDURE:
-B transfers the Cipher text C, to A
-A decrypts the Cipher text using its private key d to derive the plain text P.
Plain Text, $P = C^d \bmod n$

RSA cipher based on four prime numbers
Ivy and others modified the RSA by using four prime numbers p, q, r and s [3].
Their algorithm is as follows:
Algorithm 2. RSA CRYPTOSYSTEM BASED ON 'N' PRIME NUMBERS
Keygeneration
INPUT:
Four large prime numbers p, q, r and s.

OUTPUT:
Public Key Components: {e, n}
Private Key Components: {d, n}

PROCEDURE:
$n=p*q*r*s$

/*Compute Euler phi value of n*/
$\varphi(n)=(p-1)*(q-1)*(r-1)*(s-1)$
Find a random number e, satisfying $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$
Compute a random number d, such that,
$d=e^{-1} \bmod (\varphi(n))$
Its Encryption and Decryption processes are similar to those of the traditional RSA.
The scheme is more secured because it makes it more difficult to find the four prime inputs, compared to the ease in fing the two prime inputs in the traditional RSA.

Algorithm 3. ESRKGS KEY GENERATION

INPUT

Four prime numbers p,q,r and s.

OUTPUT
Public Key Components: {E,n}
Private Key Components: {D,n}

PROCEDURE:
```
n=p*q
m=r*s
N=n*m

/*Compute Euler phi value of n and m*/
φ(n)=(p-1)*(q-1)
φ(m)=(r-1)*(s-1)
/*Compute Euler phi value of N*/
φ(N)=φ(n)*φ(m)

Find a random number $e_1$, satisfying $1 < e_1 < φ(n)$ and $gcd(e_1, φ(n)) = 1$
Find a random number $e_2$, satisfying $1 < e_2 < φ(m)$ and $gcd(e_2, φ(m)) = 1$
Compute $E1 = e_1$^$e_2$ $mod$ N
Find a random number E, satisfying $1 < E < φ(N)*E_1$ and $gcd(E, φ(N)*E_1) = 1$
Compute a random number D, such that,
$D=E^{-1} \bmod (φ(N)*E_1)$
```

ESRKGS ENCRYPTION & DECRYPTION
ENCRYPTION
INPUT:
Plain text message, M(<n)
Public Key Components{E,n}

PROCEDURE:
$C=M^E \bmod n$

OUTPUT:
Cipher text, C

DECRYPTION
INPUT:
Cipher text message, C
Private Key Components: {D, n}

PROCEDURE:
$P=C^D \bmod n$

OUTPUT:
Decrypted plain text, P

Mathematical proof of ESRKGS
Mathematically, the ESRKGS is proven like this:
The Cipher text C is obtained as:
$C = M^E \bmod n$ (1)
where M is the plain text, E is the Encryption key and n is a variable generated from the multiplication of two random numbers.
The plain text P can be obtained from the cipher text using:
$P = C^D \bmod n$ (2)
where C is the cipher text, D is the Decryption key and n is a variable generated from the multiplication of two random numbers.
So, how is the plaint text gotten from C^D mod n?
$C^D \bmod n = M^{ED} \bmod n$ (3)
From Algorithm 3,
$D = E^{-1} \bmod (φ(N)*E_1)$.
Hence,
$ED = 1*\bmod(φ(N)*E_1) = 1*\bmod((p-1)(q-1)(r-1)(s-1)*E_1)$
$=1+k((p-1)(q-1)(r-1)(s-1)*E_1)$ (4)

(k is any positive integer)
Substitution (4) in (3)
C^{D} mod n = $M^{1+k((p-1)(q-1)(r-1)(s-1)*E1)}$ mod n
=$(M*M^{k((p-1)(q-1)(r-1)(s-1)*E1)})$mod n
=$M*(M^{(p-1)})^{k(q-1)(r-1)(s-1)*E1}$mod p*q
=$M*(M^{(p-1)})^{k(q-1)(r-1)(s-1)*E1}$mod p*q
=$M*M^{(p-1)k(q-1)(r-1)(s-1)*E1}$mod p*q
(By Fermat's Little Theorem, M^{p-1} =(approximately) 1 mod p)
= $M*1^{k(q-1)(r-1)(s-1)*E1}$ mod n = M mod n = M (Since M < n)

ESRKGS SAMPLE ENCRYPTION AND DECRYPTION
Take four prime numbers p = 89, q = 97, r = 73, s = 113.
Compute, n = p * q and m = r * s. n = 8633 and m = 8249
Compute, N = m * n. N = 71213617
Compute Euler phi value of n and m,
φ(n)=(p-1)*(q-1)
φ(m)=(r-1)*(s-1)
φ(n) = 8448 and φ(m) = 8064
Compute φ(N)=φ(n)*φ(m)
φ(N)= 68124672
Find a random number e_1, satisfying 1 < e_1 < φ(n) and gcd(e_1, φ(n)) = 1.
e_1 = 52560929
Find a random number e_2, satisfying 1 < e_2 < φ(m) and gcd(e_2, φ(m)) = 1.
e_2 = 5
Compute $E_1 = e_1$^e_2 *mod* N. E_1 = 57571842
Find a random number E, satisfying 1 < E < φ(N)*E_1 and gcd(E, φ(N)*E_1) = 1.
E = 1.08280691141358E-08
Compute a random number D, such that,
D = E^{-1} mod (φ(N)*E_1).
D = 92352569
Input message, M = Hello World!
Encryption, C = M^{E} mod n. C =
1.00000004630804,1.00000004997285,1.00000005069844,1.00000005069844,1.00000005099512,1.00000003752723,1.00000004835716,1.00000005099512,1.00000005128389,1.00000005069844,1.0000000498651,1.00000003786043
Decryption, P = C^{D} mod n. P = Hello World!

Comparism of the three schemes
After a brute-force attack on the RSA by Rivest [5] (represented with RSA1), and RSA by Ivy [3] (represented with RSA2) and the ESKGS, it is observed that the time taken to find the value of 'n' in ESRKGS is greater than that of RSA1 and RSA2. The comparism is depicted in Table 1.

Table 1 - Brute-force attack time

Length of p, q, r and s (in bits)	Time to find a 'n' by Brute-force (in s)		
	RSA1	RSA2	ESRKGS
6	1.083344	2.645654	7.855621
8	1.574581	5.383928	12.383894
10	4.047046	18.146578	32.492047
12	75.706760	123.498494	234.950321
16	14,961.466536	61,928.389156	126,473.290471

Comment on ESRKGS
Erkam [7] examined the ESRKGS and prove that if n is factorized, an alternative private key D_1 can be used to break the ESRKGS system without the use of brute force attack.
They used Message (M) = 59 and Cipher Text (C) = 2883.
When they factorize n = 7979 and obtain p=79 and q = 101, next they compute φ(n) = 7800. Knowing that Public Key E = 4425692186722853, they obtained $E^{-1} = D_1 \equiv 3317$(mod7800).
They computed $2883^{3317} \equiv 59$ (mod7979), which is M.
Erkam [7] noted that traditional RSA is a strongly secure system as obtaining a private key in RSA would demand factorizing integer n, which is computationally infeasible. They also proved that when n is factorized, ESRKGS can also be broken, so they claim that ESRKGS is just as secure as traditional RSA.

Breaking ESRKGS and breaking RSA
In traditional RSA, the values of Encryption *e* and Decryption *d* keys are based on the product of two large prime numbers, *n*. While in ESRKGS, the values of Encryption *E* and Decryption *D* keys are based on the product of four large prime numbers, *N*, which is a multiple of n.
This makes the the Encryption key E in ESRKGS very lenghty and as such, it would take a longer time for a computer to perform E^{-1} to obtain the decryption Key D in an ESRKGS cryptosystem, than it would take a computer to perform the same operation when using a shorter-length key provided by traditional RSA. This greater complexity with ESRKGS increases its security level, when compared to traditional RSA.

Motivation for using C Sharp
Hao [8] compared the performance of four of the most popular programming languages: C, C++, C# and Java by measuring the time in milliseconds it takes to run bat files for each of these languages in executing the following operations:
Int arithmetic: Math operation using ints.
Double arithmetic: Math operation using doubles.
Long arithmetic: Math operation using longs.
Trig: Calculating sin, cos, tan, log and square root for all numbers up to a maximum value.
I/O: Write message to a file, then read message from a file.

Table 2 shows the comparism in performance of the four programming languages: C, C++, C# and Java.

Language	Int Arithmetic	Double Arithmetic	Long Arithmetic	Trig	I/O	Total elapsed time
C	14273.5	18718.6	33110.9	13626.8	6052.1	85781
C++	14275.3	18659	31781.9	13462.9	5563	83742.1
C#	12601.3	17920.8	37974.6	5308.4	4260	78065.1
Java	8916.9	10322.7	27716.4	67401.5	6098.1	120455.6

Table 2

From the result shown in the Table 2, C# is one of the fastest in Int arithmetic, Double arithmetic, Trig arithmetic, I/O and total performance.
This shows that C# is very efficient for mathematical computations, hence the choice of it as the language for the implementation of this system.

- **Research Design and Methodology**

In this paper, the Iterative model of Software Development Life cycle was adopted. This model was adopted because it does not need a list of the full software requirements before the software design and coding process starts. It is iterative because while the functional part of the project is in use, additions can be made when necessary, giving rise to new versions of the system.
The stages of the iterative model include: Analysis; Design; Coding; Testing; Deployment.
Analysis: Various RSA-based ciphers are surveyed to know their security capability, and speed of execution. The ESRKGS was discovered to be the most secured and one of fastest RSA schemes today. Therefore, a system whose features can meet the ESRKGS has been specified for this project.

Design: The algorithm (pseudocode) based on the system design specification is shown below:
Pseudocode for ESRKGS key Generation

```
Input first random prime number
Input second random prime number
Input third random prime number
Input fourth random prime number
Generate Private key, Public key and Modulus N
```

Pseudocode for Encryption

```
Input Plain text
Input Public key of receiver
Input Modulus N of receiver
Generate Cipher text
```

Pseudocode for Decryption

```
Input Cipher text
Input Private key of receiver
Input Modulus N of receiver
Generate Plain text
```

Coding: The software has been developed as a Windows Application in C Sharp programming language.

Testing: The various units -Key Generation, Encryption and Decryption methods- developed in the coding phase have been integrated into a system after testing each unit. After their integration the entire system was tested against bugs and for validity with the system requirements and specifications.

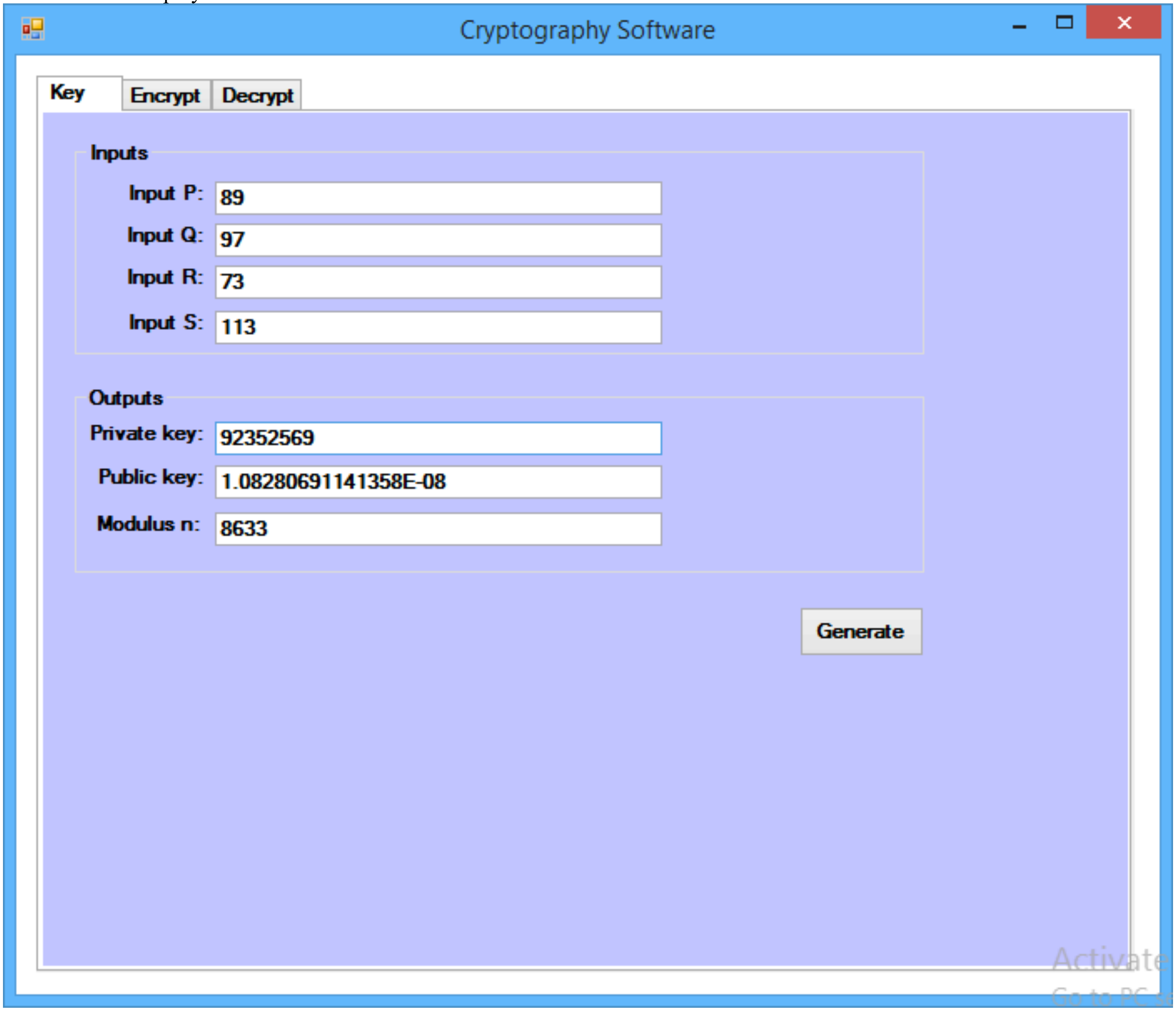
Deployment: The product is ready for use to any interested user.

- **System Implementation**

The software was implemented on a computer system with a 64-bit Intel CORE i5 processor, 4GB RAM, 500GB hard disk drive, 14 inches monitor and Windows 8.1 operating system.
The following are features/interfaces of the Software:

-Key Tab

As shown in figure 1, the Key tab accepts the prime numbers P, Q, R and S. The private and Public keys together with the modulus n are displayed when the user clicks the 'Generate' button.

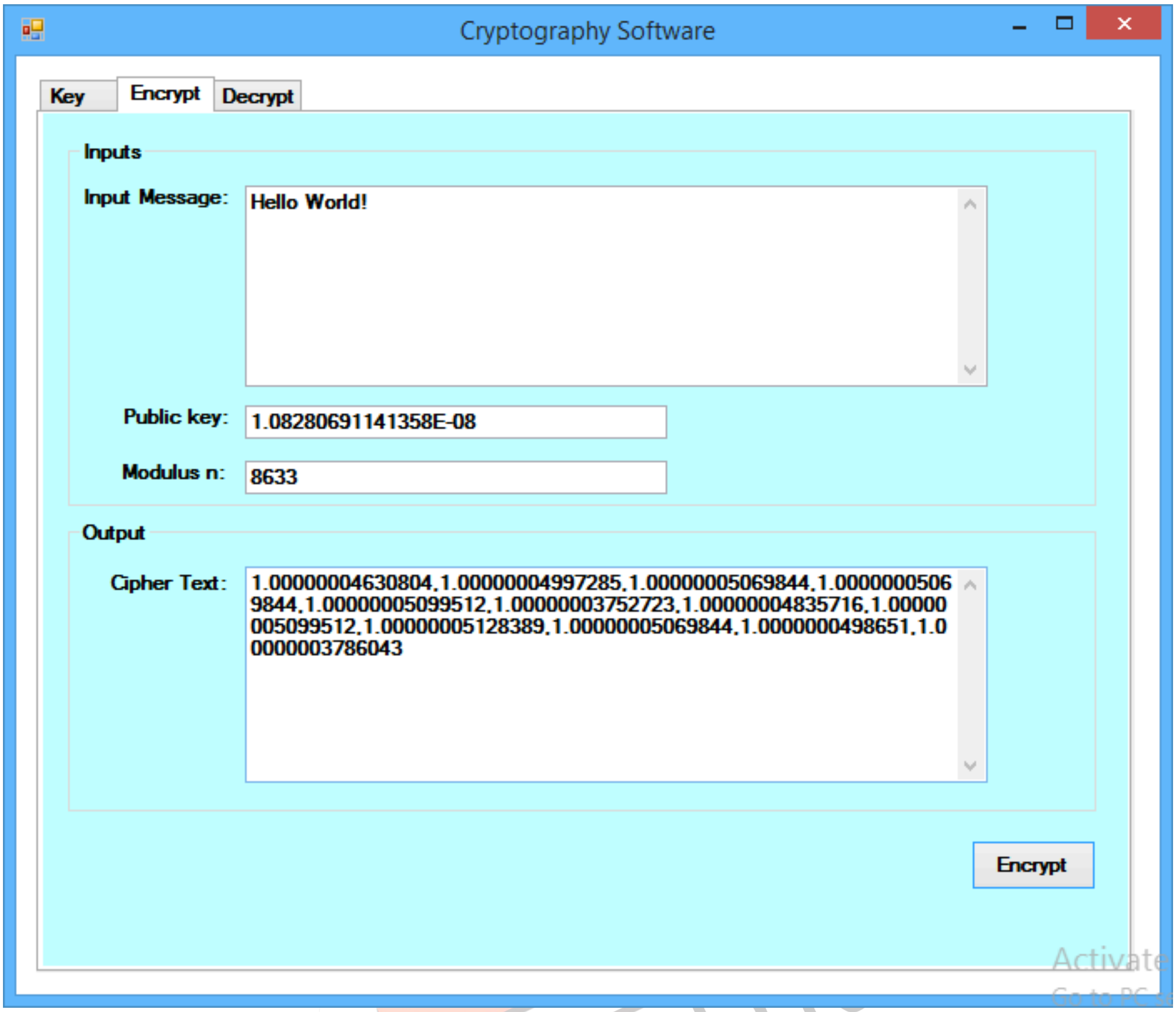


Input P: 89
Input Q: 97
Input R: 73
Input S: 113
Private key: 92352569
Public key: 1.08280691141358E-08
Modulus n: 8633

Figure 1.

-Encrypt Tab

The Entry tab provides the user with text boxes to key in the Plain text, Public Key and Modulus n. When the Encrypt button is clicked, the Cipher text is displayed.



Input Message: Hello World!

Public key: 1.08280691141358E-08

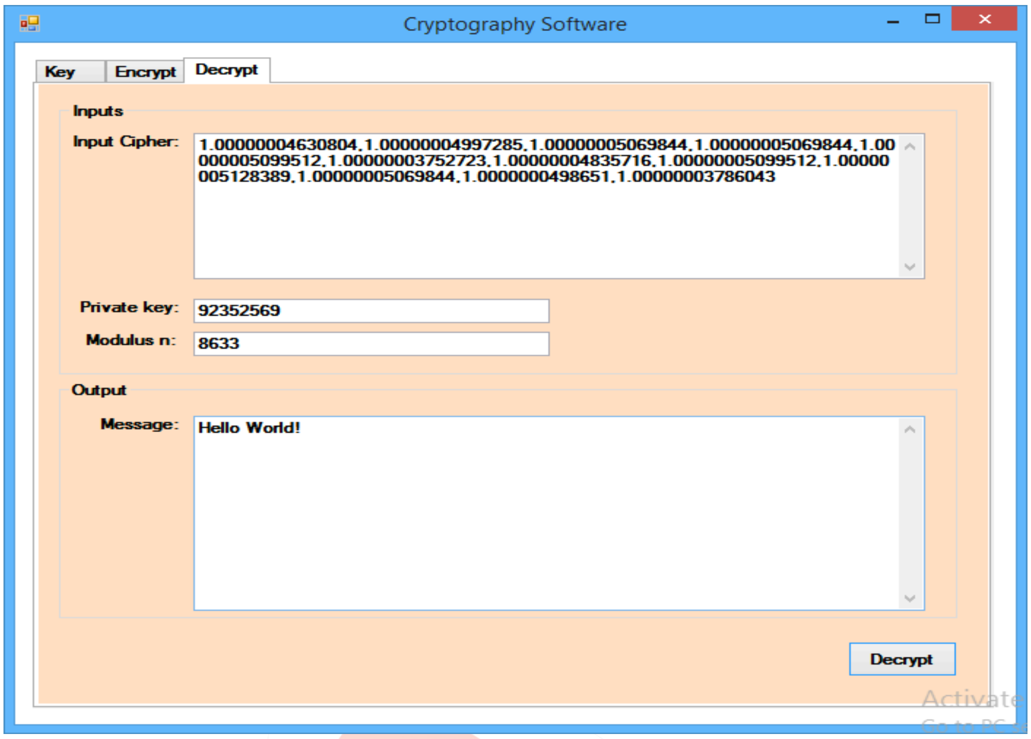
Modulus n: 8633

Cipher Text:

```
1.00000004630804,1.00000004997285,1.00000005069844,1.0000000506
9844,1.00000005099512,1.00000003752723,1.00000004835716,1.00000
005099512,1.00000005128389,1.00000005069844,1.0000000498651,1.0
0000003786043
```

Figure 2.

-Decrypt Tab



Input Cipher: 1.00000004630804,1.00000004997285,1.00000005069844,1.00000005069844,1.00000005099512,1.00000003752723,1.00000004835716,1.00000005099512,1.00000005128389,1.00000005069844,1.0000000498651,1.00000003786043

Private key: 92352569

Modulus n: 8633

Message: Hello World!

Figure 3.

CONCLUSION

In this paper, a software for implementing the Enhanced and secured RSA Key Generation Scheme (ESRKGS) cryptography has been designed and implemented. The new system provides reliable Encryption and Decryption since it is based on ESRKGS which is a reliable form of RSA Cryptographic Cipher. With the use of four random numbers, ESRKGS makes it difficult for hackers to find he primes used and the operations performed on them to obtain the Private and Public Keys. The software was created using C Sharp programming language. It is a Graphical User Interface (GUI) application that is easy to use.

REFERENCES

[1] Anjula Gupta, Navpreet Kaur Walia. Cryptography Algorithms: A Review. International Journal of Engineering Development and Research 2014; Volume 2, Issue 2.

[2] Isha Bhardwaj, Ajay Kumar, Manu Bansal. A Review on Lightweight Cryptography Algorithms for Data Security and Authentication in IoTs. 4th IEEE International Conference on Signal Processing, Computing and Control, 2017, Solan, India.

[3] Ivy PU, Mandiwa P, Kumar M. A modified RSA cryptosystem based on 'n' prime numbers. Int J Eng Computer Sci 2012;1(2):63-6.

[4] M. Thangavel, P. Varalakshmi, Mukund Murrali, K. Nithya. An Enhanced and Secured RSA Key Generation Scheme (ESRKGS). journal of information security and applications xxx (2014) 1-8.

[5] Rivest RL, Shamir A, Adleman LA. Method for obtaining digital signatures and public-key cryptosystems. Commun ACM 1978;21(2):120-6.

[6] S. Sridhar, S. Smys. Intelligent Security Framework for IoT Devices. International Conference on Inventive Systems and Control. 978-1-5090-4715-4/17/$31.00 ©2017 IEEE.

[7] Erkam Lüy, Zekeriya Y. Karatas, Huseyin Ergin, Comment on "An Enhanced and Secured RSA Key Generation Scheme (ESRKGS)", Journal of Information Security and Applications (2016), doi: 10.1016/j.jisa.2016.03.006

[8] Hao Chen. Comparative Study of C, C++, C# and Java Programming Languages. VAASAN AMMATTIKORKEAKOULU UNIVERSITY OF APPLIED SCIENCES, Information Technology, 2010.