ISSN NO. 2320-5407

Journal homepage:http://www.journalijar.com
Journal DOI:10.21474/IJAR01

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH

RESEARCH ARTICLE

RSA Cryptography Algorithm Using linear Congruence Class.

Purna Chandra Sethi[1] and Prafulla Kumar Behera[2].

1. PhD Scholar, Dept. of Computer Science, Utkal University, Bhubaneswar, India.
2. Reader, Dept. of Computer Science, Utkal University, Bhubaneswar, India.

Manuscript Info *Abstract*

Manuscript History:

Received: 18 March 2016
Final Accepted: 19 April 2016
Published Online: May 2016

Key words:
Linear congruence, Congruence class, Modulation Arithmetic, Cryptography, RSA

Corresponding Author

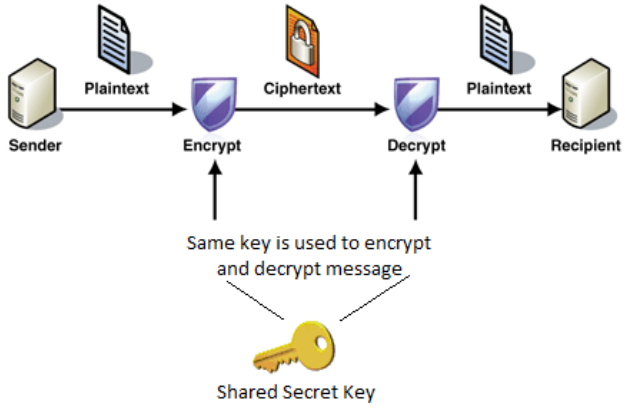
Purna Chandra Sethi

Due to the enormous demand for networking services, the performance and security of information has to be improved. To provide information security, numerous cryptographic algorithms were proposed by various researchers, out of which RSA algorithm is one the most popular algorithm. RSA algorithm uses linear congruence method which restricted the operation to specific class of values. RSA algorithm needs exponential time for decryption of message. By extending the RSA algorithm using congruence class and selecting the key in random, the security of algorithm can be increased. Higher the congruence class index, higher will be its level of security. For each of the congruence class element, complexity of algorithm will be same but there will be increase in the level of security. The basic idea behind this implementation is that by converting the given linear congruence into congruence class and solving them algebraically, actual information can be produced. This paper contains the comparison between linear congruence and congruence class using RSA algorithm. Finally we contend that, due to congruence class implementation, the complexity of algorithm will remain same as that of regular RSA algorithm with enhancement in information security by random congruence class key selection.

1. Introduction

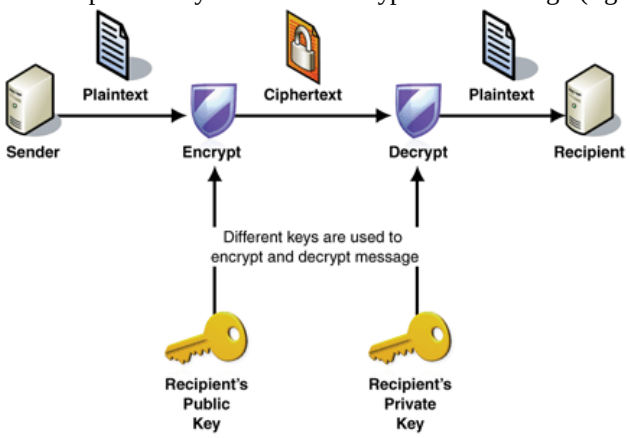
For any type of computer operations, the processing speed must be high with reduced cost for storage. Performance of the system depends on many parameters such as processing time, searching time, on chip memory space allocation, security of information, etc. The growth of the Internet and electronic commerce gives rise to many issues of data privacy. The information need to be protected when it is being transmitted over Internet which leads to the concept of cryptography. Cryptography is a field of Computer Science that makes communications unintelligible to all users except authorized parties which can access the required information. Cryptography secures the information by protecting its confidentiality and can also protect information for the integrity and authenticity of data.

There are two different cryptography systems present. They are private key cryptography and public key cryptography. In private key cryptographic system, the sender and the receiver agree on a single unique secret key called private key, which is shared among both sender and receiver (fig–1). So, in private key cryptography, the same key is used for encryption as well as decryption of the message. There are number of private key cryptography algorithms present which are explained in [1]. Private key cryptographic algorithms are simple in nature and easier to be cracked. Simple guess can be made for cracking of encrypted message (Cipher text). E.g. In case of Caesar Cipher, a simple arithmetic addition can be used for message encryption and subtraction of same key value can be used for message decryption. By Diffie–Hellman "Man in the middle attack", an intruder can access the information during data transmission and can also corrupt the information by changing the message during transmission [2].



[Fig–1: Private Key Cryptography]

Hence, the other cryptography algorithms called Public key cryptography concept is introduced. In public key cryptosystem, pair of public and private keys is used for encryption as well as decryption of data. The public key is used for message encryption and the private key is used for decryption of message (fig–2).



[Fig–2: Public Key Cryptography]

There are number of public key cryptography algorithms present which are explained in [1, 2]. Public key cryptographic algorithms are difficult to crack. The commonly used public key cryptographic algorithms are: RSA, Blowfish, Elliptic Curve Cryptosystem, etc. All these public key cryptographic algorithms have exponential time complexity to crack the encrypted data. Using brute force approach, number of years isneeded for cracking the encrypted message.

In paper [3], the author proposed a traditional approach for data transmission and guarantee that there will be 100% data transmission. When information is being corrupted, the appropriate corrupted bit is selected and retransmitted again. If such corruption of information is frequently occurs then, the processing time will increase significantly with the decrease in the efficiency of the system. So, the proposed algorithm of this paper can be applied for security of information so that there will be less chance of information being corrupted.

In paper [4], the author proposed an efficient algorithm for optimization of resource utilization applied in paper [3]. UPnP protocol is used for universal plug and play operation. Since each device and resource is not used at each time, UPnP protocol is applied that allows the devices to be plugged into the network first for use. Then it is unplugged from the network so that the resource utilization will be optimum.

In paper [5], the author applied an efficient algorithm using which the on–chip memory space can be reduced efficiently by large extent along with faster searching approach. AES is one of the efficient algorithm that is too difficult to be cracked. In paper [6], the author proposed a new algorithm by combining the feature of paper [5] and SHA–256 algorithm so that less on chip memory space can be used along with the enhancement of

informationsecurity. A proposed three tire model in paper [6]enhanced the security of information significantly. Hence it leads to efficient, secure and faster approach of data transmission. By this algorithm, on–chip memory space is reduced along with increase in searching speed.

In paper [7], the authors proposed a hybrid approach which can be used for real time network traffic analysis and management using a secured clustering approach which provides faster processing as compared to traditional approach. It also needs less on–chip memory space for the implementation of the algorithm.

In papers [6, 7], the encryption and decryption process uses linear congruence method. Linear congruence method plays a very important role in many popular cryptography algorithms. Because of this reason, finding solutions using linear congruence has received remarkable attention in the past several decades and has been studied intensively by numerous authors [8–14]. There are several methods present to solve system of linear congruence equations. In solving linear congruence, Gold et al (2005) used re–modulation method to characterize the conditions under which the solutions exist within the solution space. The solution space reduces the problem size to a finite field called Galois field. This approach relates the solution space of $cx \equiv a \bmod b$ to the Euler quotient function for c which allows developing an alternative approach for encryption and decryption of message. Stein in 2009 proposed an approach in one of his books in Number Theory for translating the given congruence into Diophantine equation $ax + by = c$ to solve linear congruence [13]. Koshy in 2007 also presented an algorithm making use of multiplicative inverses of "a modulo m" in solving linear congruence [11].

The rest of this paper is organized as follows: Section 2 presents the objective of the study and the related works, previous proposed algorithms containing the fundamental features of linear congruence, congruence class and basic modular arithmetic features. Section 3 describes the RSA algorithm in details. Section 4 contains the proposed algorithm implemented using linear congruence and congruence class using RSA algorithm. Section 5 deals with the performance of the algorithm. Section 6 provides the conclusion. Section 7 provides the future work.

2. Objectives of the study

The RSA algorithm is a public key cryptography algorithm which is based on modulo arithmetic operation. It is named after its inventors Rivest, Shamir and Adlemann in 1978. In RSA algorithm, pair of public and private keys is used for encryption and decryption. The main purpose of this paper is to implement RSA algorithm using linear congruence class in a finite field for making the processes faster as well as secure. To validate the developed algorithm, illustrative example is applied using the RSA cryptosystem. In order to effectively understand the concept of linear congruence, it is necessary to familiar with the following definitions, theorems and properties which was used throughout this paper.

Linear Congruence

In general, Linear Congruence method deals with a statement about divisibility. The theory of congruence was introduced by Carl Friedreich Gauss. Gauss contributed to the basic ideas of congruence and proved several theorems related to this theory.

A congruence of the form $ax \equiv b \pmod m$ where x is an unknown integer is called a linear congruence in one variable.

E.g. Let, m be a positive integer. Here, a is congruent to b modulo m if $m \mid (a - b)$, where a and b are integers, i.e. if $a = b + km$ where $k \in Z$. If a is congruent to b modulo m, then it can be written as $a \equiv b \pmod m$.

Example: $19 \equiv 5 \pmod 7$. Similarly $2k + 1 \equiv 1 \pmod 2$ which means every odd number is congruent to 1 modulo 2.

Congruence Class

Let, m is a natural number and let a, b be integers. a is said to be congruent to b modulo m if m divides (a – b). This is denoted by $a \equiv b \pmod m$. The natural number m is called the modulus. The set of all integers congruent to a is called the congruence class of a modulo m (or residue class) and is denoted by $[a]_m = \{ b \in Z : a \equiv b \pmod m \}$. The equivalent class is the set of elements that behave similarly and satisfies the modulo remainder feature.

E.g. The equivalent classes under modulo 5 arithmetic operations are:

[Table–1: Equivalence class for modulo 5 arithmetic operation]

Remainder Value	Equivalence class set						
0	{……	–10	–5	0	5	10	……}
1	{……	–9	–4	1	6	11	……}
2	{……	–8	–3	2	7	12	……}
3	{……	–7	–2	3	8	13	……}
4	{……	–6	–1	4	9	14	……}

Modulus Arithmetic

Modular Arithmetic is a very versatile tool discovered by K. F. Gauss. Two numbers **a** and **b** are said to be equal or congruent modulo **N** written **N|(a–b)**, if their difference is exactly divisible by **N**. Usually **a**, **b** are nonnegative integers and **N** is a positive integer. We write **a ≡b** (**mod N**).

The set of numbers congruent modulo **N** is denoted $[a]_N$. If $b \in [a]_N$then, by definition, **N|(a–b)** or, in other words, **a** and **b** have the same remainder on division by **N**. Since there are exactly **N** possible remainders upon division by **N**, there are exactly **N** different sets $[a]_N$.

Quite often these **N** sets are simply identified with the corresponding remainders**:** $[0]_N = 0, [1]_N = 1,..., [N–1]_N = N–1$. Remainders are often called *residues*; accordingly, the **[a]'s** are also known as the *residue classes.*

It is easy to see that if **a ≡b (mod N**) and **c ≡d (mod N)** then **(a+c) ≡(b+d) (mod N).** The same is true for multiplication. This allows us to introduce an **algebraic structure** into the set $\{[a]_N : a=0,1,...,N–1\}$.
By definition:

1. $[a]_N + [b]_N = [a + b]_N$

2. $[a]_N \times [b]_N = [a \times b]_N$

Subtraction can also be defined similarly: $[a]_N – [b]_N = [a – b]_N$ and it can be verified that the set $\{[a]_N: a=0,1,...,N–1\}$ becomes a **ring** with **commutative** addition and multiplication.

E.g. $[5]_{10}*[1]_{10}=[5]_{10}*[3]_{10}=[5]_{10}*[5]_{10}=[5]_{10}*[7]_{10}=[5]_{10}*[9]_{10}=[5]_{10}$

Similarly for odd set of elements, $[5]_{10}*[2]_{10}=[5]_{10}*[4]_{10}= [5]_{10} *[6]_{10}=[5]_{10}*[8]_{10}=[5]_{10}*[0]_{10}=[0]_{10}$

Division cannot be always defined. So $[5]_{10}/[5]_{10}$cannot be defined uniquely.

Definition 1. Congruenceis a linear equation involving congruent relations. Let n be a fixed positive number. Two integers a and b are said to be congruent modulo n, symbolized by $a \equiv b \ (mod\ n)$ if n divides the difference $a – b$; that is, provided that $a – b = kn$for some integer *k.*

Congruence may be viewed as a generalized form of equality, in the sense that its behaviour with respect to addition and multiplication is similar to ordinary equality (=). Some of the basic properties of congruence class are:

2.1. In modular arithmetic, if a and b are any integers and n is a positive integers, then the congruence ax≡b (mod n) has a solution for x if and only if the greatest common divisor of a and n (denoted by gcd(a,n)) is a factor of b.

2.2. The congruence ax≡b (mod n), n ≠ 0, with gcd(a,n) = d|b, has d distinct solutions.

2.3. Reflexive Property.

If a is an integer then a≡a (mod n).

2.4. Symmetric Property.

If a≡b (mod n), then b≡a (mod n).

2.5. Transitive Property.

If a≡b(mod n) and b≡c(mod n), then a≡c (mod n).

2.6. Simplification Property.

If k divides a, b and n, then a≡ b (mod n) is congruent to a/k≡b/k (mod n/k).

2.7.Cancellation Property.

If gcd(k,n) = 1, then ak≡bk (mod n) is congruent to a≡b (mod n).

2.8. Addition Property.

If a≡b (mod n), then a + k≡b+k(mod n).

2.9.Subtraction Property.

If a≡b(mod n), then a–k≡b–k(mod n)

2.10.Multiplication Property.

If a≡b(mod n), then ak≡bk (mod n).

Modulus returns the remainder value after division operation. Modulus Calculation for smaller numbers is easier to calculate. E.g. 100 (mod 3) =1. But for large number, the calculations are difficult. To find the result for such operation, modulus arithmetic is used.
E.g. 3^8mod7=6561mod 7=2. For the above problem, the calculations can be simplified 3^8 (mod 7) = 3^4 * 3^4(mod 7) = 81*81(mod 7)≡ (81 mod 7)* (81 mod 7) = 4*4=16≡2 mod 7.

3. RSA Cryptographic Algorithm

RSA (Rivest Shamir Adleman) system is a public key cryptosystem that uses two prime numbers *P*and *Q*as the private key and number *N*(product of *P*and *Q*) and number *E*(number relatively prime to (P–1)(Q–1)) as cipher text $C = M^E \ (mod\ N)$.

This cryptosystem is used in this study for the following reasons:
1. The encryption function used in RSA is a trapdoor function. Trapdoor function is easy to compute in one direction but very difficult in reverse direction without additional knowledge.
2. Encryption direction is very easy because it only requires exponentiation and modulo operations.
3. Decryption without the private key is very hard because it requires prime factorization which adds to the security of the RSA.

The RSA cryptographic algorithm using the encryption (E, N) key can be defined as follows:
1. Choose two large prime numbers P and Q.
2. Calculate N=P*Q
3. Select the public key (encryption key) E such that it is not a factor of (P–1) and (Q–1)
4. Select the private key (decryption key) D such that the following equation is true: (D*E) mod (P–1)*(Q–1) = 1
5. For encryption, calculate the cipher text CT from the plain text PT as follows: $CT = PT^E \bmod N$
6. Send CT as the cipher text to the receiver.
7. For decryption, calculate the plain text PT from the cipher text CT as follows: $PT = CT^D \bmod N$

4. RSA Algorithm Implementation using Linear Congruence

The subsequent sections provide discussion and illustrative examples of the proposed algebraic algorithm for solving linear congruence and an application of the algorithm in cryptography using the RSA system. **Algebraic Algorithm for Solving Linear Congruence** in the form $ax \equiv b \ (mod\ N)$ can be expressed in linear equation as, $x = b + Nq$, where b is a residue, N is the modulus and q is any integer. The basic idea of this method is to express the given linear congruence to equation and solve it algebraically. The algorithm for solving linear congruence is presented below.

Step 1: Check the solvability of the given linear congruence.
Step2: Convert the given linear congruence into linear equation in terms of the unknown variable.
Step 3: Find the smallest positive integer solutions to the linear equation that will make the unknown variable a whole number.
Step 4: Evaluate the linear equation using the integer solution. The result will be the smallest positive integer that is a solution to the given linear congruence. The general solution is given by the congruence $x\ b \ (mod\ n)$ where *b* is the smallest positive integer solution and n is the given modulus. To show the validity of this algorithm, an illustrative example is provided in this section.

Illustrative Example ***Solve the linear congruence 16x≡ 22(mod 26).***
Step 1. Check the solvability of the given linear congruence. *In modular arithmetic, if a and b are any integers and n is a positive integers, then the congruence ax≡b (mod n) has a solution for x if and only if the greatest common divisor of a and n (denoted by gcd (a , n)) is a factor of b.* Since the greatest common divisor of 16 and 22 is 2 which is a factor of 26, the linear congruence ***16x≡22(mod 26)*** has solutions.

Step 2. Convert the given linear congruence into linear equation in terms of the unknown variables. The linear congruence ***16x≡22(mod 26)*** when converted to linear equation is 16x = 22 + 26q. In terms of x, it will become $x=(22+26q)/16$ or in a more simplified form $x=(11+13q)/8$.

Step 3. Find the smallest positive integer solutions to the linear equation that will make the unknown variable a whole number. Given, $x=(11+13q)/8$, the smallest positive integer value of q that will make x a whole number is 1.

Step 4. Evaluate the linear equation using the integer solution. The result will be the smallest positive integer that is a solution to the given linear congruence. The general solution is given by the congruence $x \equiv b \pmod{n}$ where b is the smallest positive integer solution and n is the given modulus.
If q = 1, then evaluating $x=(11+13q)/8$ will be :
$x=(11+13(1))/8=(11+13)/8 =24/8 = 3$. Thus, the solution to linear congruence $\boldsymbol{16x \equiv 22 \pmod{26}}$ is **3(mod 26).**

Application of the Developed Algebraic Algorithm for Solving Linear Congruence in Cryptography using the RSA System
In this section, the developed algorithm on linear congruence will be applied in some parts in the encryption and decryption of the message using the RSA system. RSA (Rivest Shamir Adleman) system is a public key cryptosystem using prime numbers *P*and *Q* as the private key and number N(product of *P* and *Q*) and number E(number relatively prime to (P–1)(Q–1)) as well as the cipher text $C = M^E \pmod{N}$.

***RSA* encryption algorithm**:
1. Represent the message as an integer between 0 and (N–1) by considering the ASCII equivalent of the characters. Large numbers can be broken up into number of smaller blocks in a specified range. Each block would then be represented by an integer in the same range.
2. Represent the message using the integers produced sequentially.
3. Encrypt the message by raising it to the E^{th} power modulo N.
4. For each of the result, set of congruence class will be produced. Any of the elements among a congruence class can be considered to generate the cipher text from the message (plain text) C.
5. Send the Cipher text to the receiver end.

RSA Encryption algorithm implementation using example:-
1. Let the message "PCSNPKB" to be encrypted using RSA with E = 3 and N = 85.
2. The message is represented by the ASCII equivalent of the characters.

P=80, C=67, S=83, N=78, P=80, K=75, B=66

3. So, the message sequence by considering the block of two digits for each of the character.

M = 80678378807566

4. Encrypt each block as $C = M^E \pmod{N} = M^3 \pmod{85}$

The encrypted message for each block of two characters can be evaluated as given in the table:

[Table–2: Encryption Process]

Character	$M^3 \pmod{85}$	Cipher Text
P	$80^3 \pmod{85}$	45
C	$67^3 \pmod{85}$	33
S	$83^3 \pmod{85}$	77
N	$78^3 \pmod{85}$	82
P	$80^3 \pmod{85}$	45
K	$75^3 \pmod{85}$	20
B	$66^3 \pmod{85}$	26

By combining the individual evaluated values, the cipher text produced is: 45337782452026. Hence the cipher message generated by considering the ASCII equivalent of the numbers as: "–!MR–¶→"

RSA Decryption Algorithm
The decryption algorithm for the message will be:
1. Compute D by the inverse of E mod (P–1)(Q–1)
2. Apply algebraic algorithm to solve the linear congruence for the smallest integer value so that D will produce whole number.
3. Decrypt cipher text message C by raising it to another power modulo N with the cipher text which returns the plain text.

RSA decryption algorithm implementation using example:
The above decryption algorithm for the cipher text "–!MR–¶→" represented in its ASCII equivalent form as "45337782452026" with P = 5, Q = 17 and E = 3 is given below.
1. Compute D, the inverse of E mod (P–1)(Q–1).
(P–1)(Q–1) = 4 (16) = 64. Thus, we will have 3D ≡ 1(mod 64)
2. Use the algebraic algorithm to solve the linear congruence
3D ≡ 1(mod 64)
=>3D = 1 + 64R
=> D = (1 + 64R)/3
Where, 1 is the smallest integer R that will make number D whole number. Thus, by substituting
R = 2, D will be equal to 43.
3. To decrypt the cipher text message, raise it to D modulo N.

To find the value for 45^{43}(mod 85) needs exponential time. So instead of direct calculation, the linear congruence properties are applied. The illustration for finding 45^{43}(mod 85) = 80 can be represented as follows:

45^{43}(mod 85) = 45^{3*14+1} mod 85
= {$(45^3)^{14}$x45} mod 85
= {45^3 x 45^3 x 45^3 x … 14 times x 45} mod 85
= {(45^3 mod 85) x (45^3 mod 85) x…x (14 times) x
(45 mod 85)} mod 85
= {5 x 5 x …14 times x 45} mod 85
= (5^{14} x 45) mod 85
= 80

The decrypted integer values are selected and the plain text was generated by its ASCII equivalent value. Hence the plain text from the cipher text can be generated as shown in the table–3.

[Table–3: Decryption Process]

Cipher Text	C^{43} (mod 85)	Character
45	45^{43}(mod 85) = 80	P
33	33^{43}(mod 85) = 67	C
77	77^{43}(mod 85) = 83	S
82	82^{43}(mod 85) = 78	N
45	45^{43}(mod 85) = 80	P
20	20^{43}(mod 85) = 75	K
26	26^{43}(mod 85) = 66	B

Hence the decrypted message sequence using the evaluated value will be: 80678377807566. Thus the message can be generated by considering the character equivalent of the ASCII values. So, the actual message will be: "PCSNPKB".

Implementation of congruence class for the plain text
Using an encryption (E, N), the ***encryptionalgorithm*** using congruence class can be defined as:
1. Represent the message as an integer between 0 and (N – 1). Large numbers can be broken up into number of blocks. Each block would then be represented by an integer in the same range.
2. Represent the message in using the integers sequentially to represent the block of integers.
3. Encrypt the message by raising it to the E^{th} power modulo N.
4. From the result, set of congruence class element can be produced. Any corresponding set of elements can be selected for generating the cipher text. The result is a cipher text message C.
5. Consider any congruence class value generated as: C = C + i x N, where as i = { –∞, …, –2, –1, 0, 1, 2, …, ∞ }
6. Send the Cipher text to the receiver end.

Implementation of the RSA algorithm using congruence class by example

A. RSA Encryption of Message using Congruence class items:

Encrypt the message "PCSNPKB" using RSA with E = 3 and N = 85.

1. Let the message is represented by the ASCII equivalent of the characters.

P=80, C=67, S=83, N=78, P=80, K=75, B=66

2. So the message sequence by considering the block of two digits.

M = 80678378807566

3. Encrypt each block as $C = M^E \pmod N = M^3 \pmod{85}$

4. Encrypt each block as C = C + i x N, where i = = { –∞, …, –2, –1, 0, 1, 2, …, ∞ }

Modulo 85 will produce 85 set of values out of which any one can be used for data encryption. During decryption of data, initially modulo arithmetic operation will be applied and then decryption algorithm is implemented that will automatically produce the same plain text as result.

The set of equivalent class for modulo 85 is:

[Table–4: Congruence Class for modulo 85]

Remainder Value	Equivalence class set
0	{… …, –170, –85, 0, 85, 170, … …}
1	{… …, –169, –84, 1, 86, 171, … …}
2	{… …, –168, –83, 2, 87, 172, … …}
3	{… …, –167, –82, 3, 88, 173, … …}
4	{… …, –166, –81, 4, 89, 174, … …}
5	{… …, –165, –80, 5, 90, 175, … …}
6	{… …, –164, –79, 6, 91, 176, … …}
7	{… …, –163, –78, 7, 92, 177, … …}
8	{… …, –162, –77, 8, 93, 178, … …}
9	{… …, –161, –76, 9, 94, 179, … …}
10	{… …, –160, –75, 10, 95, 180, … …}
11	{… …, –159, –74, 11, 96, 181, … …}
12	{… …, –158, –73, 12, 97, 182, … …}
13	{… …, –157, –72, 13, 98, 183, … …}
14	{… …, –156, –71, 14, 99, 184, … …}
15	{… …, –155, –70, 15, 100, 185, … …}
16	{… …, –154, –69, 16, 101, 186, … …}
17	{… …, –153, –68, 17, 102, 187, … …}
18	{… …, –152, –67, 18, 103, 188, … …}
19	{… …, –151, –66, 19, 104, 189, … …}
20	{… …, –150, –65, 20, 105, 190, … …}
21	{… …, –149, –64, 21, 106, 191, … …}
22	{… …, –148, –63, 22, 107, 192, … …}
23	{… …, –147, –62, 23, 108, 193, … …}
24	{… …, –146, –61, 24, 109, 194, … …}
25	{… …, –145, –60, 25, 110, 195, … …}
26	{… …, –144, –59, 26, 111, 196, … …}
27	{… …, –143, –58, 27, 112, 197, … …}
28	{… …, –142, –57, 28, 113, 198, … …}
29	{… …, –141, –56, 29, 114, 199, … …}
30	{… …, –140, –55, 30, 115, 200, … …}
31	{… …, –139, –54, 31, 116, 201, … …}
32	{… …, –138, –53, 32, 117, 202, … …}
33	{… …, –137, –52, 33, 118, 203, … …}
34	{… …, –136, –51, 34, 119, 204, … …}
35	{… …, –135, –50, 35, 120, 205, … …}
36	{… …, –134, –49, 36, 121, 206, … …}

37	{… …, –133, –48, 37, 122, 207, … …}
38	{… …, –132, –47, 38, 123, 208, … …}
39	{… …, –131, –46, 39, 124, 209, … …}
40	{… …, –130, –45, 40, 125, 210, … …}
41	{… …, –129, –44, 41, 126, 211, … …}
42	{… …, –128, –43, 42, 127, 212, … …}
43	{… …, –127, –42, 43, 128, 213, … …}
44	{… …, –126, –41, 44, 129, 214, … …}
45	{… …, –125, –40, 45, 130, 215, … …}
46	{… …, –124, –39, 46, 131, 216, … …}
47	{… …, –123, –38, 47, 132, 217, … …}
48	{… …, –122, –37, 48, 133, 218, … …}
49	{… …, –121, –36, 49, 134, 219, … …}
50	{… …, –120, –35, 50, 135, 220, … …}
51	{… …, –119, –34, 51, 136, 221, … …}
52	{… …, –118, –33, 52, 137, 222, … …}
53	{… …, –117, –32, 53, 138, 223, … …}
54	{… …, –116, –31, 54, 139, 224, … …}
55	{… …, –115, –30, 55, 140, 225, … …}
56	{… …, –114, –29, 56, 141, 226, … …}
57	{… …, –113, –28, 57, 142, 227, … …}
58	{… …, –112, –27, 58, 143, 228, … …}
59	{… …, –111, –26, 59, 144, 229, … …}
60	{… …, –110, –25, 60, 145, 230, … …}
61	{… …, –109, –24, 61, 146, 231, … …}
62	{… …, –108, –23, 62, 147, 232, … …}
63	{… …, –107, –22, 63, 148, 233, … …}
64	{… …, –106, –21, 64, 149, 234, … …}
65	{… …, –105, –20, 65, 150, 235, … …}
66	{… …, –104, –19, 66, 151, 236, … …}
67	{… …, –103, –18, 67, 152, 237, … …}
68	{… …, –102, –17, 68, 153, 238, … …}
69	{… …, –101, –16, 69, 154, 239, … …}
70	{… …, –100, –15, 70, 155, 240, … …}
71	{… …, –99, –14, 71, 156, 241, … …}
72	{… …, –98, –13, 72, 157, 242, … …}
73	{… …, –97, –12, 73, 158, 243, … …}
74	{… …, –96, –11, 74, 159, 244, … …}
75	{… …, –95, –10, 75, 160, 245, … …}
76	{… …, –94, –9, 76, 161, 246, … …}
77	{… …, –93, –8, 77, 162, 247, … …}
78	{… …, –92, –7, 78, 163, 248, … …}
79	{… …, –91, –6, 79, 164, 249, … …}
80	{… …, –90, –5, 80, 165, 250, … …}
81	{… …, –89, –4, 81, 166, 251, … …}
82	{… …, –88, –3, 82, 167, 252, … …}
83	{… …, –87, –2, 83, 168, 253, … …}
84	{… …, –86, –1, 84, 169, 254, … …}

By selecting the congruence class items, the cipher text in its ASCII equivalent form is: 130 118 162 167 130 105 111 (Considering, i = 1). Hence the cipher text is: “évóǫéio”.

B. RSA Decryption Algorithm using Congruence class items:
The decryption algorithm for the message will be:-

1. For i ← 1 to Length(C)
2. While (C_i<0)
3. $C_i = C_i + N$
4. $C_i = C_i$ % N
5. Compute D by the inverse of E mod (P–1)(Q–1)
6. Apply algebraic algorithm to solve the linear congruence for the smallest integer value so that D will produce whole number.
7. Decrypt cipher text C by raising it to another power modulo N with the cipher text which returns the plain text.

Decrypt the cipher text "évóoéio" represented in by their ASCII equivalent as: "130 118 162 167 130 105 111" for the RSA algorithm implementation using P = 5, Q = 17 and E = 3.

1. Since congruence index i=1, so each actual message can be generated by modulus operation. Hence the after modulus operation the message becomes:

M = 45 33 77 82 45 20 26

2. Compute D, the inverse of E mod (P–1)(Q–1).
(P–1)(Q–1) = 4 (16) = 64. Thus, we will have 3D ≡ 1(mod 64)
3. Use the algebraic algorithm to solve the linear congruence
3D ≡ 1(mod 64)
=>3D = 1 + 64R
=> D = (1 + 64R)/3
Where, 1 is the smallest integer R that will make number D whole number. Thus, by substituting

R = 2, D will be equal to 43.

4. To decrypt the cipher text message, raise it to D modulo N.

Since, calculating the value for large exponentiation is difficult, so linear congruence methods are applied to find the modulo output.

To find the value for 45^{43}(mod 85) needs exponential time. So instead of direct calculation, the linear congruence properties are applied. The illustration for finding 45^{43}(mod 85) = 80 can be represented as follows:

45^{43}(mod 85) = 45^{3*14+1} mod 85
= {$(45^3)^{14}$x45} mod 85
= {45^3 x 45^3 x 45^3 x … 14 times x 45} mod 85
= {(45^3 mod 85) x (45^3 mod 85) x…x (14 times) x (45 mod 85)} mod 85
= {5 x 5 x …14 times x 45} mod 85
= (5^{14} x 45) mod 85
= 80

The decrypted message for each block of two characters from the encrypted message will can be evaluated as given in the table:

[Table–5: Decryption of encrypted information]

Cipher Text	C^{43} (mod 85)	Character
45	45^{43}(mod 85) = 80	P
33	33^{43}(mod 85) = 67	C
77	77^{43}(mod 85) = 83	S
82	82^{43}(mod 85) = 78	N
45	45^{43}(mod 85) = 80	P
20	20^{43}(mod 85) = 75	K
26	26^{43}(mod 85) = 66	B

Hence the decrypted message sequence using the evaluated value will be: 80678377807566. Thus the message can be generated by considering the character equivalent of the ASCII values. So, the actual message will be: "PCSNPKB".

Based on the congruence class table, the possible encrypted message class for the string "PCSNPKB" will be:

[Table–6: Equivalent set of congruence class elements for encrypted message]

Character	Cipher Text	Equivalence class set
P	45	{… …, –125, –40, 45, 130, 215, … …}
C	33	{… …, –137, –52, 33, 118, 203, … …}
S	77	{… …, –93, –8, 77, 162, 247, … …}
N	82	{… …, –88, –3, 82, 167, 252, … …}
P	45	{… …, –125, –40, 45, 130, 215, … …}
K	20	{… …, –150, –65, 20, 105, 190, … …}
B	26	{… …, –144, –59, 26, 111, 196, … …}

There will be number of possible choices present that can be used for data encryption. Other than the actual set of values, if any other set is selected for data encryption then it will return a cipher text which is more secure than that of the actual cipher text. The resultant cipher text under operation will return the same plain text leading to better security than that of the actual ones.

Performance Matrix

The performance of the above process is evaluated by comparing the standard processing with the different congruence class elements. Using the different congruence class values the encryption and decryption process is applied. The performance matrix for the actual process without congruence class application can be represented as:

[Table–6: Total time for encryption and decryption of information using RSA algorithm]

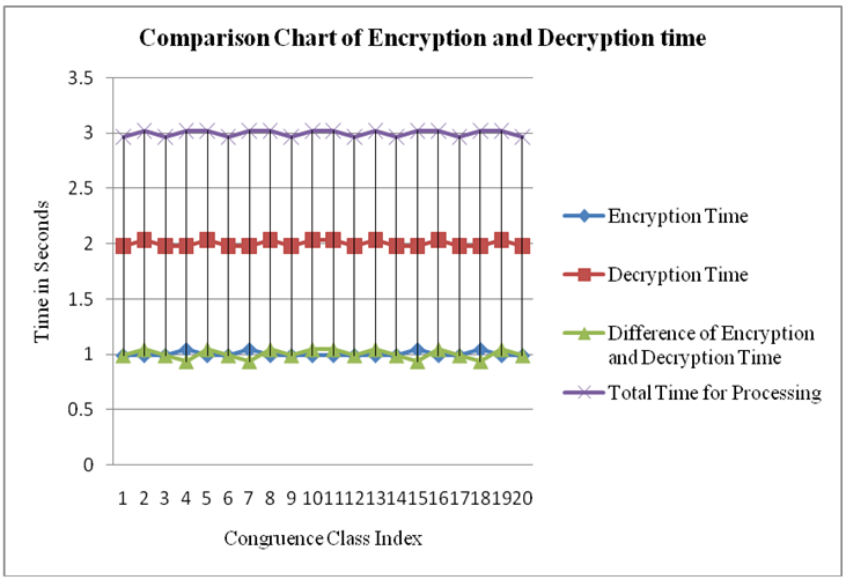
Encryption Time	Decryption Time	Total Time
64.78022	100.384615	165.164835

After congruence class implementation the encryption time, decryption time, difference between the encryption and decryption time and the total time for encryption and decryption time will be:

[Table–7: Total time for encryption and decryption of information using congruence class in RSA algorithm]

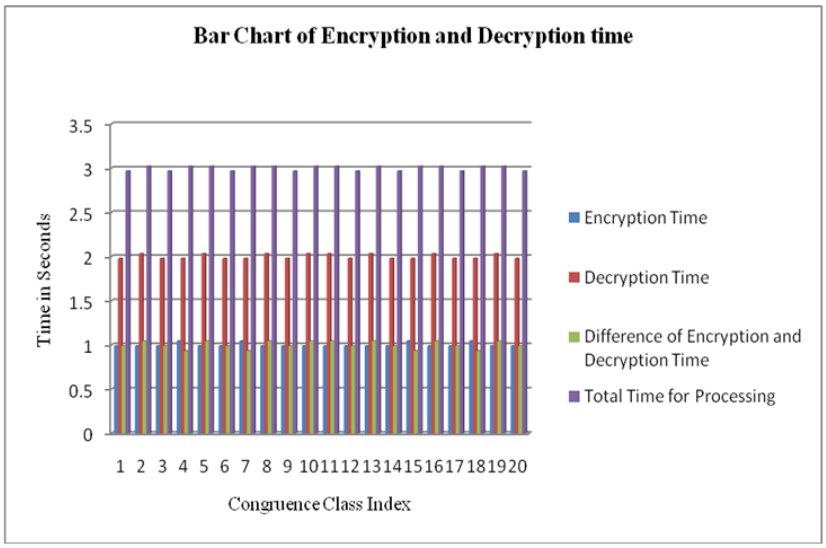
Congruence Class Index	Encryption Time	Decryption Time	Difference of Encryption and Decryption Time	Total Time
1	0.989011	1.978022	0.989011	2.967033
2	0.989011	2.032967	1.043956	3.021978
3	0.989011	1.978022	0.989011	2.967033
4	1.043956	1.978022	0.934066	3.021978
5	0.989011	2.032967	1.043956	3.021978
6	0.989011	1.978022	0.989011	2.967033
7	1.043956	1.978022	0.934066	3.021978
8	0.989011	2.032967	1.043956	3.021978
9	0.989011	1.978022	0.989011	2.967033
10	0.989011	2.032967	1.043956	3.021978
11	0.989011	2.032967	1.043956	3.021978
12	0.989011	1.978022	0.989011	2.967033
13	0.989011	2.032967	1.043956	3.021978
14	0.989011	1.978022	0.989011	2.967033
15	1.043956	1.978022	0.934066	3.021978
16	0.989011	2.032967	1.043956	3.021978
17	0.989011	1.978022	0.989011	2.967033
18	1.043956	1.978022	0.934066	3.021978
19	0.989011	2.032967	1.043956	3.021978
20	0.989011	1.978022	0.989011	2.967033

The comparison diagram for the above experimental result can be represented as:



[Fig–3: Comparison chart of Encryption and Decryption time]

The bar chart for the above experimental result can be represented as:



[Fig–4: Bar chart of Encryption and Decryption time]

5. Performance of Algorithm

The algorithm can be applied for any type of message encryption. Due to linear congruence implementation, set of possible values can be generated for each character that gives set of possible cipher text. The message encrypted using RSA algorithm is difficult to crack. It needs exponential time for cracking. By considering set of congruence class elements, the complexity of algorithm remains same because instead of selecting the actual value, a random element from the congruence class is selected. Due to the random selection of congruence class element, instead of regular cipher text, the cipher text for the corresponding random congruence class element is produced which increases level of security for the information. Higher the congruence class index, higher will be information security.

6. Conclusion

Aside from the known methods and techniques of solving linear congruence, the algebraic algorithm provides another way of finding solutions to congruence. The algorithm can be implemented for any type of message encryption. Due to linear congruence implementation, set of possible values was generated for each character that produced set of possible cipher text. The message encrypted using RSA algorithm is difficult to crack. It needs exponential time for cracking. By the set of congruence class output, a class of cipher text messages was generated. Due to the set of messages, the security of information increases as compared to the regular RSA algorithm implementation.

7. Future works

The above work can be extended from finite field to complex field. Due to the complex field application, two set of output (real and imaginary filed) will be produced. By using both set of values, the information can be extracted. Single part (either real part or imaginary part) will not be sufficient for accessing the information. By the application of complex field, the security level can be increased to a higher level.

References

1. Behrouz A. Forouzan, "Cryptography & Network Security", 1st Edition, TMH.
2. AtulKahate, "Cryptography and Network Security", 2nd Edition, TMH
3. P. C. Sethi, C. Dash: "High Impact Event Processing using Incremental Clustering in Unsupervised Feature Space through Genetic algorithm by Selective Repeat ARQ protocol", ICCCT– 2nd IEEE Conference – 2011, pp. 310–315.
4. P. C. Sethi, "UPnP and Secure Group communication Technique for Zero–configuration Environment construction using Incremental Clustering", International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 12, December – 2013, ISSN: 2278–0181, pp. 2095–2101
5. Tzu–Fang Sheu, Nen–Fu Huang, and Hsiao–Ping Lee, "In–Depth Packet Inspection Using a Hierarchical Pattern Matching Algorithm", IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 7, NO. 2, APRIL–JUNE 2010
6. P. C. Sethi, P.K. Behera, "Secure Packet Inspection using Hierarchical Pattern matching implemented Using Incremental Clustering Algorithm", December–22–24, ICHPCA–2014 (IEEE International Conference)
7. P. C. Sethi, P.K. Behera, "Internet Traffic Classification for Faster and Secured Network Service", International Journal of Computer Applications (IJCA), Volume 131 – No.4, December2015, pp. 15–20
8. González, H.E., and Carmona, L., J.J., "Solving Simultaneous Linear Equations using Finite Fields On Hybrid GPU–Accelerated Multi–core Systems", International Journal of Engineering and Innovative Technology (IJEIT) Volume 4, Issue 6, December 2014, P153–158
9. Adams, D.G.(2010). Distinct Solutions of Linear Congruence. ActaArithmetica Vol. 141 No. 2. pp. 103–152 [4] Burger, E. B. (2006). Small Solutions of Linear Congruence over Number of Fields. Rocky Mountain Journal of Mathematics Vol. 26 No. 3.pp 875–888
10. Frieze, A. et al. (2006). Reconstructing Truncated Integer Variables Satisfying Linear Congruence. SIAM Journal on Computing. Vol. 17 No. 2. pp 262–280
11. Koshy, T. (2007). Elementary Number Theory with Applications. 2nd Ed. Elsevier Publishing Inc. pp. 211–245
12. Lindahl, L. A. (2003). Number Theory. Retrieved from http://www2.math.uu.se/~ lal/kompendier/Talteori.pdf.
13. Stein, W. (2009). Elementary Number Theory : Primes, Congruence and Secrets. 1st Ed. Springer Publication. pp 21–44
14. Sburlati, G. (2003). Counting the Number of Solutions of Linear Congruence. Rocky Mountain Journal of Mathematics Vol. 33 No. 4.pp 1487–1497