MODIV: A Proposed Method for Data Encryption

مودف: طريقة مقترحة لتشفير البيانات

Qusay M. J. Alsaadi
Alrafidain University College
Department of Computer Communications Engineering

Abstract

Starting from the principle of computer security that branched to many fields, whereas the encryption is one from, therefore this paper focused on a new idea of data encryption, it depends on modular and integer division operations, and then will produce two numbers, first one represents the result of (MOD operation), second one represents the result of (DIV operation), and then these two numbers and the secret key will be used within the two functions to produce an encrypted file. The proposed method will use one symmetric key to encrypt the plaintext, also there is another proposed elementary level of encryption depends on substitution technique, Finally this paper includes explanation of how the data will be encrypted and decrypted using the idea of (MODULAR and DIVISION), also include some results, computations, the proposed algorithm. This technique will be named as (MODIV) method

المستخلص

ابتداءاً من مبدأ امنية الحاسوب الذي يتفرع الى عدة فروع حيث ان التشفير هو واحد من تلك الفروع، لذلك هذا البحث يركز على فكرة جديدة لتشفير البيانات والتي تعتمد على العمليات المعيارية والقسمة الصحيحة ، ومن ثم يتولد رقمين الاول يمثل ناتج العملية المعيارية ، والثاني يمثل ناتج عملية القسمة الصحيحة، من ثم هذين الرقمين مع المفتاح السري وباستخدام الدالتين لتوليد فايل مشفر. الطريقة المقترحة سوف تستخدم مفتاح سري متناظر واحد لتشفير النص الصريح، كذلك يوجد مستوى ابتدائي مقترح اخر من التشفير يعتمد على مبدأ التعويض. اخيرا هذا البحث يتضمن توضيح كيف للبيانات ان تشفر باستخدام فكرة المعياري والقسمة الصحيحة، كذلك يحتوي على بعض النتائج والحسابات اضافة الى الخوارزمية. يمكن لهذه الطريقة ان تسمى (MODIV).

Key words

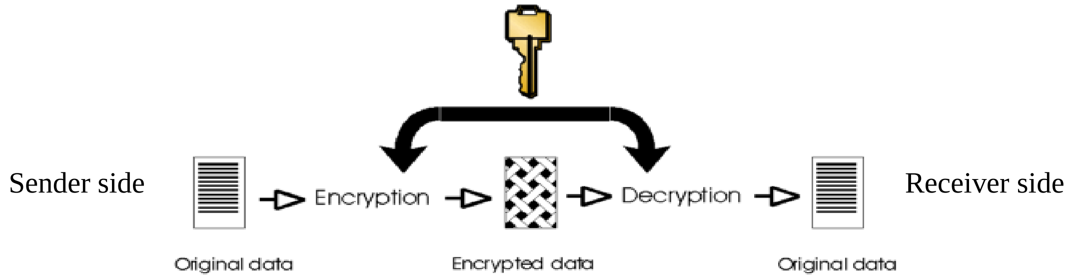
Computer Security, Data Encryption, MOD, DIV, Symmetric Key, Substitution, MODIV.

1-Introduction

One of the first well-known and well-documented encryption algorithms is the “Caesar Code”, a simple code where each letter of the alphabet is associated with another letter of the alphabet, rotated by n positions. Here, n is the secret key that is necessary to both encrypt the human-readable message to the cipher text and to decrypt the cipher text to the human-readable message [1].

1-1 Symmetric and public keys:

Symmetric-key encryption is an encryption method that uses the same key for encryption and decryption, as shown in Fig (1). This type of encryption is quick and well suited when the data does not need to be shared [5], this paper will discuss this type of encryption.



Figure(1) Symmetric Key Encryption

A public-key algorithm (also called an asymmetric algorithm) uses a key pair. As shown in Fig (2) [2], the key used for decryption is different from the key used for encryption.

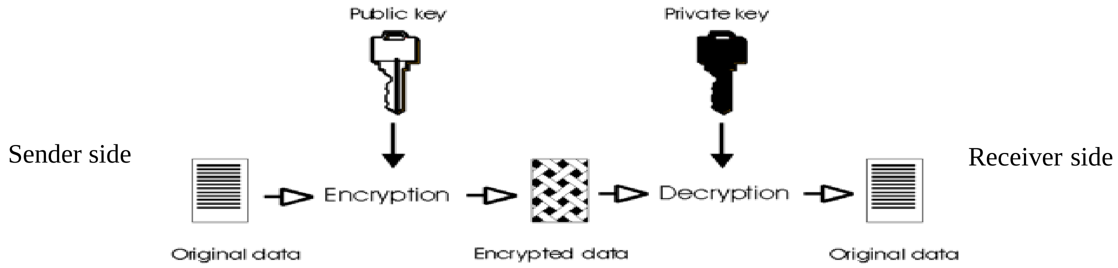


Figure (2) Public Key Encryption

1-2 RSA method:

In 1977, shortly after the idea of a public key system was proposed, three mathematicians, Ron Rivest, Adi Shamir and Len Adleman gave a concrete example of how such a method could be implemented. To honour them, the method was referred to as the RSA Scheme. The system uses a private and a public key. To start two large prime numbers are selected and then multiplied together; *n=p*q* [10].

If we let *f(n) = (p-1) (q-1)*, and *e>1* such that *GCD(e, f(n))=1*. Here *e* will have a fairly large probability of being co-prime to *f(n)*, if *n* is large enough and *e* will be part of the encryption key. If we solve the Linear Diophantine equation; *ed congruent 1 (mod f(n))*, for *d*. The pair of integers *(e, n)* are the public key and *(d, n)* form the private key. Encryption of *M* can be accomplished by the following expression; *Me = qn + C* where *0<= C < n*. Decryption would be the inverse of the encryption and could be expressed as; *Cd congruent R (mod n)* where *0<= R < n*. RSA is the most popular method for public key encryption and digital signatures today [10].

Let see the algorithm of RSA [10]

RSA Algorithm

Key generation:
Select random prime numbers *p* and *q*, and check that *p != q*
Compute modulus *n = pq*
Compute phi, phi= *(p - 1)(q - 1)*
Select public exponent *e*, 1 < e <phi such that *gcd(e,phi) = 1*
Compute private exponent $d = e^{-1} \mod phi$
Public key is *{n, e}*, private key is *d*
Encryption: $c = m^{e} \mod n$, Decryption: $m = c^{d} \mod n$
Digital signature: $s = H(m)^{d} \mod n$, verification: $m' = s^{e} \mod n$,
if *m' = H(m)* signature is correct. H is a publicly known hash function.

Who read this section will ask if it was necessary to mention this algorithm in this paper, the answer about this question is yes, it is necessary because this algorithm (RSA) was gave me a starting point to search about a method more easy than RSA. The new method (MODIV method) represented as symmetric encryption technique, and also the key used in the proposed method (MODIV) is in any length of bytes. Any way it is difficult to say the MODIV is more efficient than RSA, but don’t forget RSA was hacked [1].

2- Idea of the proposed method

In the beginning this proposed method will be named as (MODIV) method, where this name was gushed from the two operations of division (mod and div); mod: divide a number (n1) on a second number (n2) to keep of the remainder and leave the result, ex: 35 mod 4 = 3; div: divide (n1) on (n2) to keep of the integer result and leave the remainder, ex: 77 div 5 =15. These two operations are the core of this paper, also it is known to every programmer or mathematician, also it is important to mention this technique will use only one private key so that this proposed method correspond to be as private classification.

From the above example it is possible to say, this technique will be used in encryption where n1 is the plaintext and n2 is the key.

The following equations will be used to encrypt the data

X= Number MOD Key ……..(1)

Y= Number DIV Key ……. (2)

Where the

Number is the original data (plaintext).

Key is secret key must be hidden for every one except authorized sender and receiver.

X, Y is the encrypted data.

The two equations above will be used in the proposed method to encrypt the data. So that from one number (data in bytes as shown in the next sections) will produce two numbers saved in encrypted file, Let us see the following example:

34778 Mod 33= 29

34778 Div 33 = 1053

Number= 34778, key=33

X=29, Y=1053 is the encryption of the original data (34778).

Figure (1) and figure(2) explain how the data will be encrypted and decrypt.

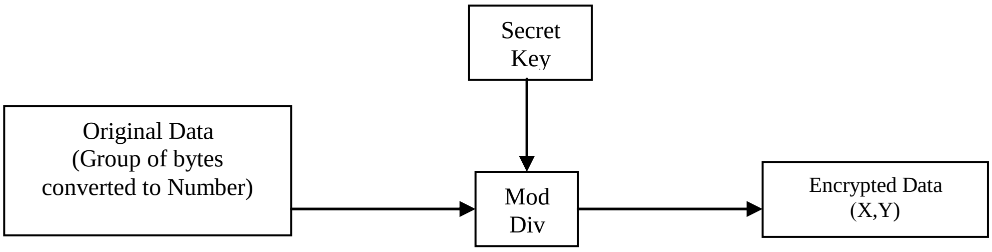


Figure (3) Block diagram of how the data will be encrypted

Where the following equation will be used in decryption:

Number = Y * Key + X ……(3)

1053 * 33 + 29 = 34778, compare this result with the number before encryption.

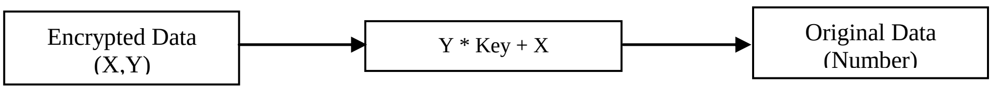


Figure (4) Block diagram of how the data will be decrypted

The number before MODIV equations is (34778), after MODIV equations (encryption) will be (1053, 29), after decryption will be back to the original (34778), and so on to all the data of the original message after some preprocessing operations.

The above equations (Eq 1,2,3) are the core idea of the proposed method; also there are some traditional techniques to complete the principle of encryption.

This technique corresponds to be as symmetric encryption, the key is secret and given to the sender and receiver only, original data is the plain text.

As shown above the proposed method depends on some computations to get the encrypted data and then decrypted data, but if this proposed system provided to any beneficiary may be there are some suspicions, for this reason there is another level of encryption to make the overall system more secure and guaranteed, therefore it is explained as follows:

In encryption there are two techniques to change the plaintext to another form; 1- substitution: is one in which the letters of plaintext are replaced by other letters or by numbers or symbols, if the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with cipher text bit patterns, 2- transposition: the mapping is achieved by performing some sort of permutation on the plaintext letters [12].

So that there is another proposed method possible to be corresponded to the first type (substitution) which complete the overall system. It depends on the mechanism of shifting the bits of the plaintext file by (n) locations to the left, where (n) is the number that generated from the first bits of each byte in the secret key, let see the following example:

Suppose the key bits pattern is as follows:

00010010,01110111,00011001,11100000,00011101

In this case the key is five bytes only, in reality the key length is more than five bytes, anyway, it will take the first bit from each byte to get the following binary number 01101 in decimal is (13) so that it will: 1- (type one) shifts all the bits of the plaintext 13 locations to the left as one block, or 2- (type two) shift each five bytes of the plaintext 13 bits locations to the left. The mechanism of the plaintext partitioning (file of any format) must agree between the sender and the receiver to decide if it is of type 1 or 2. After some theoretical testing it seemed that the type two is better than type one.

Let the plaintext is:

1000110101111100010100011110101100001000

For the above example the shifting key bits is 13 so that the plaintext will be as follow:
100010100011110101100001000**1000110101111** (first level -type one)

This string of bits will represent the first level of encryption in this paper, but as mentioned before it is not the core of the proposed method but it is possible to give the proposed method a new name as (Shift MODIV)

The phase of decryption with same computations to get the number (13) from the secret key and then shift the encrypted file 13 bits locations to the right, this level of encryption also correspond to be as symmetric encryption.

The following diagram (figure (5)) explains the mechanism of substitution technique that proposed in this paper (encryption using shifting by n bits to the left and then decryption using shifting by n bits to the right):

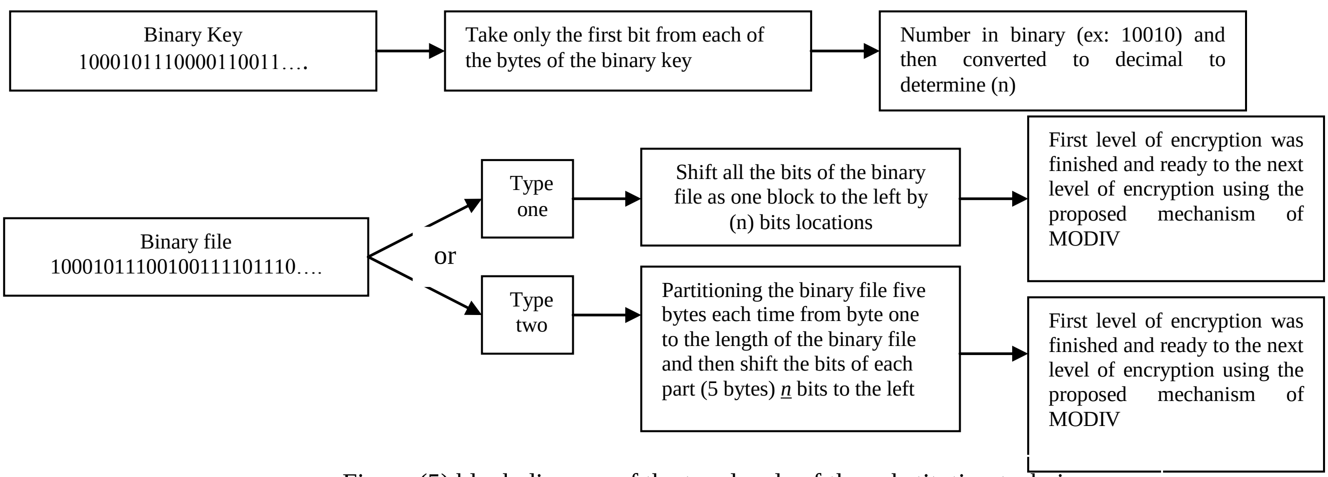


Figure (5) block diagram of the two levels of the substitution technique

The above diagram will be used as a first level of encryption and then the second level of encryption (MODIV) will be ready to encrypt the result of the first level.

Note: when decrypt the encrypted file, the above diagram must be reversed to get the original data (plaintext), also the receiver must have knowledge of the type one or type two partitioning.

3- Stages and some results of the proposed system

The proposed system will consist of the following stages as shown in figure (6):

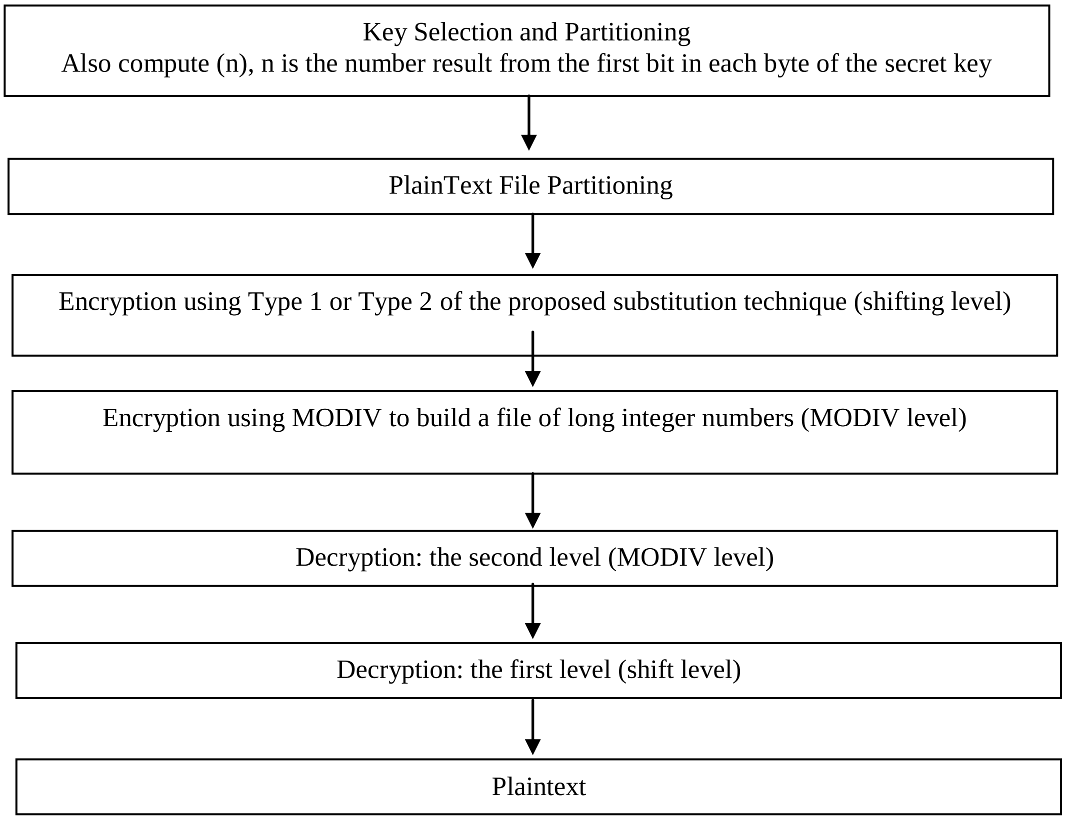


Figure (6) Block diagram of the proposed method stages

3-1 key selection and partitioning:

To understand the proposed system, let the key to be the word (*rafidain*) and the plaintext is (*I want from you to help me*), it is important to mention that the key is a long string or a file of any type but for more explanation text is chosen, there is no problem if it was a file of another format because the mechanism of partitioning depends on bytes.

ASCII of (r)=114 =(01110010)
ASCII of (a)=97 =(01100001)
ASCII of (f,i,d,a,i,n)=102, 105, 100, 97, 105,110

0111001001100001011001100110100101100100011000010110100101101110
The first bit (LSB) from the first five bytes (first level of type two) is:
$(01010)_b = (10)_d$ for this reason the bits of the first five bytes of the text file will shift to the left ten locations and also the next five byte and so on. 110 will be neglected because it is not enough to take

another five bits (first bits of the next five bytes of the key's bytes), otherwise, continues to take another five bits to shift the next five bytes of the plaintext (loop: if key[5bytes]=last five bytes of the key's bytes then go back to the first five bytes of the key). But according to the above example (rafidain) the first bit of the first five characters (bytes) will be used to shift each five bytes of the plaintext ten locations to the left.

In the decryption stage the cipher text will shift to the right ten locations, all of this after the MODIV encryption level finished and send to the receiver.

3-2 Plaintext Partitioning and First Level Encryption (Shifting):

As shown in the section of key selection the string of plaintext was (I want from you to help me), it will be saved in binary form as follow:

010010010010000001110111011000010110111001110100001000000110011001110010011011110110110100
100000011110010110111101110101001000000111010001101111001000000110100001100101011011000111
0000001000000110110101100101

As shown above there are two stages of encryption (shift and MODIV) but before the first level of the encryption (shift) to begin it must to detect if the partitioning (sectoring) is of type one (the file will shift ***n*** bits locations to the left as one block), or, of type two (each five bytes (40 bits) will be shifted ***n*** bits locations to the left), for the above example the result will be as follow:

Type one

N=10 bits locations shift to the left,

100000011101110110000101101110011101000010000000110011001110010011011110110110100100
000001111001011011110111010100100000011101000110111100100000011010000110010101101
1000111000000010000000110110101100101010010010

First level encryption of type one

Type two

N=10

The first five bytes of the plaintext are:

0100100100100000011101110110000101101110 it will shift to the left ten locations, where, the first ten bits will be rotated as follow:

1000000111011101100001011011100100100100

For the next five bytes of the plain text:

0111010000100000011001100111001001101111\1000000110011001110010011011110111010000

And so on,

0110110100100000011110010110111101110101\1000000111100101101111011101010110110100
0010000001110100011011110010000001101000\1101000110111100100000011010000010000001
0110010101101100011100000001000000110110\1011000111000000010000000110110101100101

The last byte will not change because it is not enough to shift it.

Then, the full pattern of the first level encryption (of type two) is as follow

1000000111011101*10000101*10111001*00100100*10000001*10011001*11001001*10111101*11010000
*10000001*11100101*10111101*11010101*10110100*11010001*10111100*10000001*10100000*10000001
*10110001*11000000*10000001*10110101*01100101*

First level encryption of type two also the last byte will not change (shifting of type two)

For good results I advice the sender to leave four spaces in the end of the plaintext to enable the last byte(s) to be encrypted.

Figure (7) show how the plaintext "I want from you to help me" was encrypted using the first level encryption of type two:

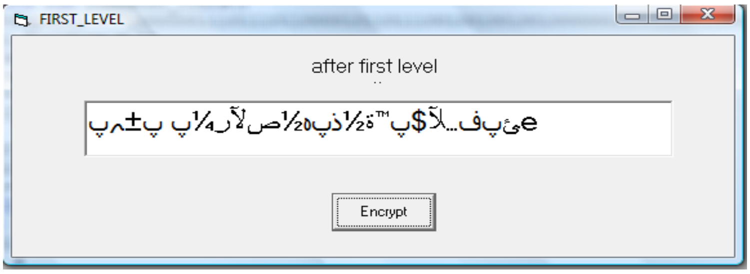


Fig (7) Encrypted data of the plaintext "I want from you to help me"

3-3 Encryption Using the Core Level (MODIV level):

The following level (MODIV) is stronger than the first level (SHIFT substitution) that explained above. It will take the results of the first level and set it to the two equations of this level as shown in section three of this paper, let see how the encryption of this level is working:

The first try was done using three bytes of the plaintext string (shift type two or one) and three bytes key (24 bits key) as shown bellow:

The secret key was ***rafidain***

011100100110000101100110*011010010110010001100001*0110100101101110

The string of the plaintext is (I want from you to help me).

The bits string of the first level encryption (SHIFT level of type two) is:

100000011101110110000101101110010010010010000001100110011100100110111101110100001000000111100101101111011101010110110100110100011011110010000001101000001000000110110001110000000100000011011010101100101

First iteration

011100100110000101100110 = 7496038 first three bytes of the key (24 bits encryption)

100000011101110110000101 = 8510853 first three bytes of the string (shift)

Using Eq(1) and Eq(2) we will get the following results:

8510853 MOD 7496038= 1014815

8510853 DIV 7496038= 1

And continue to the next iteration, where the key will take the next three bytes, also shift the SHIFT string three byte as follow:

The second three bytes of the key is 011010010110010001100001=6906977

The second three bytes of the shift string is 101110010010010010000001=12133505

also MOD and DIV operations to get the results.

The result of the Eq(1) is 5226528

And the result of the Eq(2) is 1

And so on for all the three bytes of the key and shift string.

Decryption of the three bytes string with three bytes key using Eq(3):

1*7496038+1014815=8510853 100000011101110110000101

1*6906977+5226528=12133505 101110010010010010000001

100000011101110110000101.101110010010010010000001

The bits of the first five bytes will shift ten locations (*n*) to the right
01001001.00*100000.01110111.01100001.01101110*

73= I 32=(space) 119= w 97=a 119= n
And so on (decryption) for the next five bytes of the string until the last five bytes.

Note1: if we did not use the first level the shift string will be the plain text it self.
Note2: all the results of the MOD and DIV operations will be saved in a text file as an integer numbers (loop) and then it become ready to send it to the receiver.
Note3: when the key partitioning, and shift string partitioning (or may be the binary plaintext) procedures called it will sectors the first three bytes (or ***L*** bytes) of the key and the first three bytes (or ***B*** bytes) of the shift string in the first iteration of the loop and use the (MODIV) equations to give the new encrypted data, in the second iteration the MODIV will take the next three bytes of the key and the next three bytes or next four or five bytes….n'th byte (it is flexible to select any number of bytes sectoring) of the shift string or (plaintext without first level of encryption), and so on with the loop.

After some tests it is clear that the three byte of the shift string is not enough for the second level of encryption (MODIV level), for this reason let the next try is the first four bytes of the shift string:

10000001110111011000010110111001= 2178778553

2178778553 MOD 7496038
MOD function was programmed using the following subprogram:

```
Sat1=2178778553
Key=7496038
Do
  sat1 = sat1 - key
Loop Until (sat1 < key)
Print sat1
```

Also DIV function was programmed using the following subprogram:

```
sat1 = 2178778553
key = 7496038
i = 0
Do
  sat1 = sat1 – key
  i = i + 1
Loop Until (sat1 < key)
Print i
```

Sat1 and Key are not constants but to explain the results made them as constants, any way: the result of Eq(1) is 4927533 and the result of Eq(2) is 290

As shown in the subprogram above the subtraction operation is before the loop condition to overcome the problem of (sat1< key), because if this problem was not solved it will make the plain characters (bytes value) is the same result of normal MOD.

Ex\
<u>Normal MOD:</u>
133002 MOD 2204432=133002 it is problem because the result is the same with plaintext value (133002).
<u>But in the programmed MOD (paper work)</u>
133002 MOD 2204432= (-2071430) is the solution.

In the second iteration:
The next four bytes of the shift string is: 00100100100000001100110011100100 1= 612473289
And the next three bytes of the key is: 011010010110010001100001=6906977
612473289 MOD 6906977=4659313
612473289 DIV 6906977 =88
(LOOP) until length of the Shift string (or the plaintext it self if the encryption is without first level), each time four bytes of the string and three bytes of the key,
Decryption four bytes string with three bytes key using Eq(3):
In the first iteration the key was 7496038,
((result of DIV operation)*key)+(result of MOD operation)
290*7496038+4927533= 2178778553 \ <u>10000001110111011000010110111001</u>
In the second iteration the key was 6906977
88*6906977+4659313=612473289 \ <u>00100100100000001100110011100100 1</u>
<u>*10000001110111011000010110111001.00100100*10000000110011001110010 01</u>
Each 40 bits will shift to the right ten bits (n) locations
<u>01001001.00100000.01110111.01100001.01101110</u>

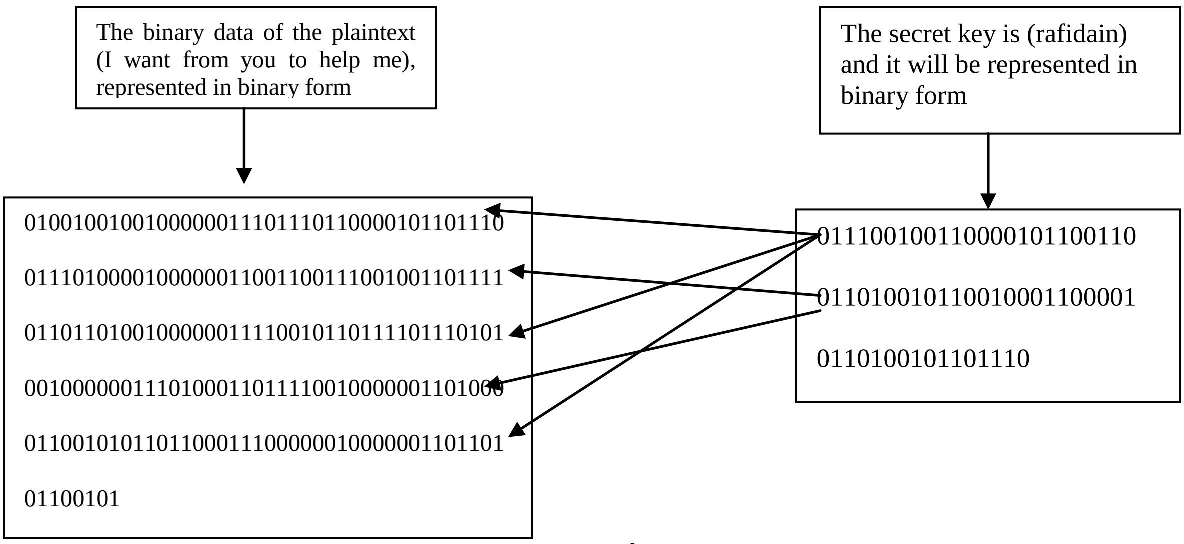
73=I 32=(space) 119=w 97=a 110=n

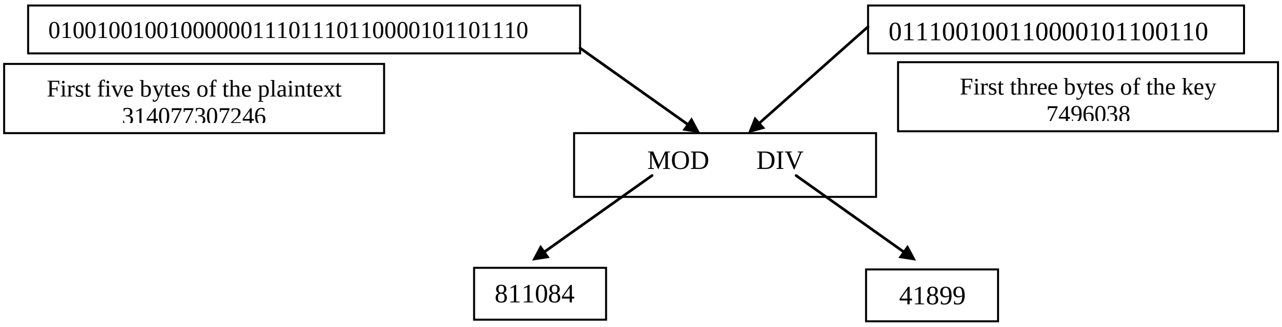
And so on (decryption) for the next five bytes of the string until the last five bytes.

Why five bytes by five bytes, because the shifting level was of type tow, where it was shift each five bytes *n* locations to the left, in decryption it will shift *n* locations to the right.

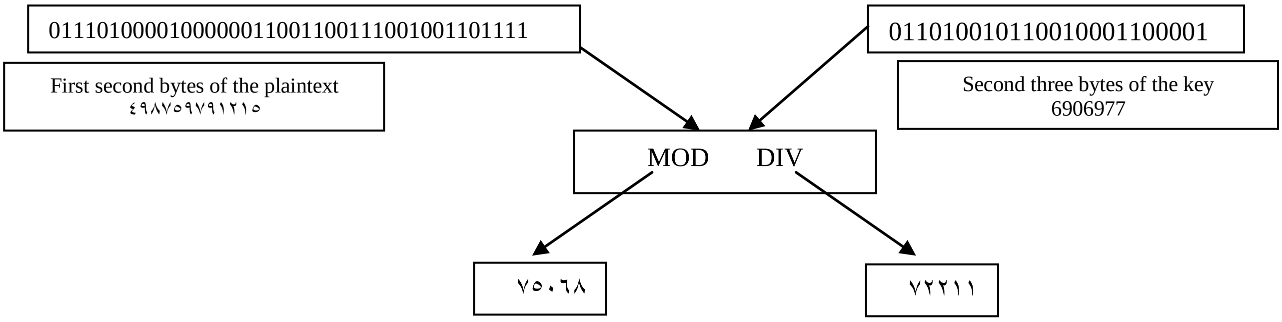
4- Encryption using MODIV level only without first level:

This paper was tried to proof the effectiveness of the functions MODULAR and DIVISION for encryption, for this reason let try to encrypt a file of text type using a key of length (L) and as shown in the previous sections with same mechanism of encryption, but in this section the encryption is without the SHIFT level (with out any type of shifting method(substitution)), it will encrypt any data using the Eq(1) and Eq(2) only, where the decryption using the Eq(3) only, in this section the sectoring of the text string (plaintext) is five bytes each time and the key is three bytes each time.

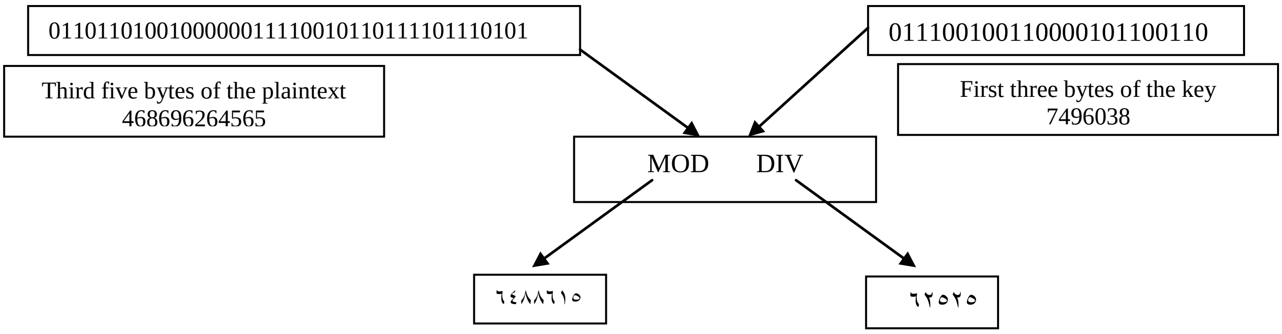




For the second iteration will be as shown bellow:



For the third iteration as shown bellow:



And so on for all the data of the plaintext, the encrypted file will be as follow (Table (1) show the results of encryption for the string (I want from you to help me) using second level (MODIV level) only (by applying Eq(1) and Eq(2)):

MODULAR RESULTS Eq(1)	DIVISION RESYULTS Eq(2)
811084	41899
75068	72211
6488615	62525
2690483	20181
1224301	58112

Table (1)

The results shown in table (1) will save in a text file and it will be sent to the receiver. As shown the encryption of five bytes or more will not need for the first level and it is secure because five bytes plaintext and three bytes key with the mechanism of (Iteration (shift five byte plaintext shift three byte key)) next five bytes of the plaintext each time and three bytes of the key each time) it is very good mechanism for encryption.

Now, let explain how the encrypted data (table 1) will decrypt using the Eq(3).

Number = Y * Key + X …. (3)

41899*7496038+811084=314077307246\ 0100100100100000011101110110000101101110
72211*6906977+75068=498759791215\111010000100000011001100111001001101111
62525*7496038+6488615=468696264565 \0110110100100000011110010110111101110101
20181*75068+2690483=1517637791\01011010011101010101000010011111
58112*7496038+1224301=435610984557\110010101101100011100000010000001101101

The binary results will represent the plain text after converting each 8 bits to a decimal (ascii), and converting to symbols, finally we get the string (I want from you to help me)

5- The proposed Algorithm:

As shown above the proposed system will consist of two levels (Shift level and MODIV level), whereas the first level (proposed Shift level) is to give the proposed system the perfection, the second level (MODIV) is the main idea of this paper.

Encryption Algorithm:

INPUT: Secret Key, Plaintext, *b*, *q*

First level algorithm:

1- Select the secret key (K) as a long string or any file of data.
2- Gather the first bit from each byte of the key bytes to create a number (N_SHIFT) represents how many bits the file bits string will shift to the left.
3- Select the type of Shifting level
    a- If it is of type one then all bits of the file will shift to the left N_SHIFT bits locations as one block.
    b- If it is of type two then shift the bits of each three bytes (or may be ***b*** bytes) of the plain text N_SHIFT bits locations to the left.

       M=length of the plaintext in bytes MOD 3
       for i= 1 to (length of the plaintext in bytes – M)
           i = i+3(or ***b***),
           Shift (bits string of the plaintext N_SHIFT to the left).
       Loop next three bytes.
    c- first level encryption (file (E1)) will be ready for the second level of encryption (MODIV level)

Second level algorithm

1- Select the key (K) as a long string or any file of data (the same secure key in the first level (SHIFT level)).
2- Detect if you will use the first level to encrypt the plaintext

i- If yes then:
    a- Tell the receiver about this agreement.
    b- (E1) must be the result of the first level encryption.
    c- Go to (steps)

ii- If no then :(steps)

Steps:1-Subdivide the secret key to groups of three bytes(or any length of partitioning) each time to create strings of 24 bits (i'th 24 bit key) this will named (K1), i=1.

Note: the length of the subgroup key is optionally (3 byte or 4 byte ….) but it must less than the length of subdividing the plain text to make the final computations difficult to be concluded.

2- Subdivide the plaintext to groups of (q) bytes each time to create strings of (q*8) bits (j'th q*8 bit plaintext string) this will named (N), j=1.

Note: q must be at least 3 bytes, but it preferred (by experiments) to be 5 bytes or more.

3- Using the following equations to find (M1 and D1):

M1=N MOD K1

D1=N DIV K1

4- (M1,D1) saved in Encrypted File.

5- i=i+[3 or 4,5,…], j=j+[3 or 4,5,…], if i= length of the secret key then i=1.

6- Loop (go to steps), if j= length of the plain text then end the operation of encryption (stop loop).

7- Send the Encrypted file to the receiver.

Note: if the encrypter used the two levels of encryption, (E1) must be computed before, and then second level will begin to compute M1 and D1.

Decryption Algorithm:

In the beginning the sender and receiver must agree if the encryption is using first level and second level or it was using second level only (core level), also must agree about the type of shifting in the first level. Second level decryption must before first level. The algorithm of decryption is as follow:

INPUT: Secret Key, Encrypted file

1- The secret key must be known for both sender and receiver.

2- Subdivide the key to find K1's, i=1, j=1

3- For all M1's and D1's compute N's using the following equation:

N=D1*K1+M1

4- Convert N to binary

5- Check if the encryption was done by using first and second level or by using second level only.

If the encryption was done by using only the second level then jump to the step 6.

If the encryption was done by using first and second level then N's will represent (E1) and then determine if it was of type one or type two

If the type is one then

Shift (E1) N_SHIFT bits to the right.

If the type is two then

Shift each three bytes (N_SHIFT) bits to the right

6- i=i+[3or 4,5,…]   j=j+[3 or 4,5…].

(I) incrementing must less than (J) incrementing to make the results more difficult for analyzing.

7- If i= length of the secret key then i=1.

8- If j= length of the encrypted file then stop the decryption operation, else go to **step 3**.

9- Convert (N's) or (E1) to an equivalent ASCII code.

Note: key length partitioning and plaintext length partitioning (subdividing) possible to be with any length.

6- Results of the Proposed System:

The proposed system originally depends on the idea of encryption using (proposed MODIV) technique, for this reason it is important to see how the algorithm of second level will encrypt a data of text type with a mechanism of partitioning five bytes (Loop will increment by five bytes plaintext and three bytes key in the next iteration), also the mechanism of the first and second levels was explained in the sections above.

The following results were computed using second level to see how the method (MODIV) is strong.

Consider the file (plaintext.txt) contains the following text data:

> Encryption has primarily been used to prevent the disclosure of confidential information ،but can also be used to provide authenticity of the source of the message ،verify the integrity of received data, provide the digital equivalent of a handwritten signature, and nonrepudiation. Nonrepudiation assures that a transacting party cannot deny that the transaction took place

Plain text

The secret key is: "iraq is my country", it is possible to be very long string (may be a file of any data format with unlimited size) (1KB,2KB,…..), the key possible to be an image of type BMP for example.

The following table represents the results of (Eq1) and (Eq2)

Mod Results Eq(1)	Div Results Eq(2)
226281	43152
5754724	65146
1098832	18448
6923724	60839
2762544	67272
7049296	55376
3887484	72996
581270	67452
4898459	57757
6028987	17559
4795052	59040
1432449	62724
4828773	20158
6347923	64553
7471410	57739
2772628	58502
759670	65529
4237715	59338
4836095	6293
7347910	18762
3994150	55458
289204	53237
3015727	59616
5184166	18256
348483	65502
2556203	68043
3723829	66267
812576	65536*
7191933	68445
4260509	62728
4828790	20158
4909152	60477
3479821	65720
965104	23874
1626891	59649
2951695	18265
3267889	65526
4540408	66280
7428692	63413
5503343	53778
6240484	58872
1718226	54688
5785611	69274
3876295	18801
3831865	57159
6161791	52710
3073412	69036
1708250	57085
2772600	63472

Mod Results Eq(1)	Div Results Eq(2)
3123455	56443
2543048	60030
2335058	17341
6817283	60815
1284172	64381
6480495	68608
4328814	63983
3441454	57159
6942059	60289
7025086	65519
6746233	66067
5813097	65528
4196947	66882
485473	65536 *
5666538	62830
156461	67254
5574371	55972
567635	20161
3519684	70170
3567952	62865
5779543	5622
6375464	19082
2251691	11243
4073535	72372
4965693	56418
6097826	62689
2801094	57961
5387860	58442

No problem if there is any repetition because the result of Eq(1) is different

If there is any repetition in Eq1 and Eq2 results at the same time (in the same iteration) that is not mean a big problem because it solved by using a long string of secret key.

Table (2) encrypted data (encrypted file) will send to the receiver

(5 bytes plaintext partitioning 3 bytes key partitioning)

When you decrypt the file (table 2) using the same secret key "iraq is my country" (in the receiver side) the results (decrypted data) will be as shown in plaintext.txt. But, if the secret key is not correct, like to be " hello my friend", the decrypted data will be as follow:

```
D½K`©kتF„¤-ذùVP_   )}Kh-b½PX6b   x١١‘üp¬□j   ZI-2   -,«·^§–,g□٨!ب!¬qT%déh‹dY]١[~ص———————ڷ#•□ll4چ–c-î   س
            ضص¤:
<‹®XWà@~X□¥£Tc•½/□ٹ
ٌve`§ûlcد٩ىm¤"•my,f jذ¢a     )½    «-d□’مm ب   (
%R□^□\Q÷Eo□™@›lk□…ň·چLč!Cچ‘S<□ؤw]ب;□„Z|UKژs·ۂKd□— Xw…1Ŝ÷` ‡r9خ/□_[T…zc8کèW[ة£
_ =p´  ±G
ؚe–±Nمd¤Fh‘j/ʾoeة^ىB^‘‘□5dµۈV‹fl□کfA.éهhbپˆ
            n©€P“my'hab7ع□h□□6EY)”□![¤0su6B̈   bE»»ة   ڨ³xo-eù,`-8±Œ/x؟’¤نX1صحoa   ؛a†\T…¯□`²¢ص”
```

7- Source Code:

The following source code is part of the program that gives the results of table (2).

MODIV subprogram for encryption:

```
Private Sub ENCRYPT_Click()
Open "d:\encryfil.txt" For Output As #2
JLOOP = 0
RES = 0
keystr = ""

  For KLOOP = 0 To k - 1
   RES = 0
   S_T_ENC = TEXTARRAY(KLOOP)

   For I_BIN = 1 To Len(S_T_ENC)
       DIGIT_BIN = Mid(S_T_ENC, ((Len(S_T_ENC)) - (I_BIN - 1)), 1)
       If DIGIT_BIN = 1 Then
          RES = RES + 2 ^ (I_BIN - 1)
      End If
   Next I_BIN

    sat1 = RES
    sat22 = RES
    Print RES
    keylen = Len(KEYARRAY(JLOOP))
    keystr = KEYARRAY(JLOOP)
    Print KEYARRAY(JLOOP)
    keysum = 0

    For Ikey = 1 To Len(keystr)
      keybin = Mid(keystr, ((Len(keystr)) - (Ikey - 1)), 1)
        If keybin = 1 Then
         keysum = keysum + 2 ^ (Ikey - 1)
        End If
    Next Ikey

  Print "keysum", keysum

    Do
     sat1 = sat1 - keysum
    Loop Until (sat1 < keysum)

  X = sat1
  ik = 0

  Do
    sat22 = sat22 - keysum
    ik = ik + 1
  Loop Until (sat22 < keysum)

  Y = ik
  JLOOP = JLOOP + 1
  If JLOOP = j Then
     JLOOP = 0
  End If
  Print JLOOP
  Print #2, X, Y
 Next KLOOP

Close #2
End Sub
```

The following source code is part of the program that decrypts the results of table (2).
MODIV subprogram for decryption:

```
Private Sub DECRYPT_Click()
 JLOOP = 0
 z = 0
 e1 = 0
 e2 = 0
 Open "d:\encryfil.txt" For Input As #2

 Do While Not EOF(2(
    sline_st1"" =
    sline_st2"" =
    Line Input #2, sline
    b = 1
    uu = 0

   If Mid(sline, b, 1) = " " Then
     Do
         b = b + 1
     Loop Until (Mid(sline, b, 1(" " <> (
  End If

 If Mid(sline, b, 1) <> " " Then
    Do
     sline_st1 = sline_st1 + Mid(sline, b, 1(
     b = b + 1
    Loop Until (Mid(sline, b, 1(" " = (
 End If

 If Mid(sline, b, 1) = " " Then
    Do
      b = b + 1
      Loop Until (Mid(sline, b, 1(" " <> (
 End If

 If Mid(sline, b, 1) <> " " Then
     Do
      sline_st2 = sline_st2 + Mid(sline, b, 1(
      b = b + 1
     Loop Until (Mid(sline, b, 1(" " = (
 End If

 amod(e1) = Val(sline_st1(
 adiv(e2) = Val(sline_st2(
 e1 = e1 + 1
 e2 = e2 + 1
Loop

 For DEC_KLOOP = 0 To k1 - 1
    keylen = Len(KEYARRAY(JLOOP((
    keystr = KEYARRAY(JLOOP(
    keysum = 0
    For Ikey = 1 To Len(keystr(
       keybin = Mid(keystr, ((Len(keystr)) - (Ikey - 1)), 1(
      If keybin = 1 Then
         keysum = keysum + 2 ^ (Ikey - 1(
      End If
    Next Ikey
   Print "keysum", keysum
   star= " "
   z = 0
   z = (adiv(DEC_KLOOP) * keysum) + amod(DEC_KLOOP(
  JLOOP = JLOOP + 1
  If JLOOP = j Then
    JLOOP = 0
 End If
```

```
 ssnew = Bin(z(
 len_ssnew = Len(ssnew(

 If len_ssnew < 40 Then
    Do
      star = star + "0"
      len_ssnew = len_ssnew + 1
    Loop Until len_ssnew = 40
 star = star + ssnew
 ssnew = star
 End If

 m = 1
 decr1 = Mid(ssnew, m, 8(
 decr2 = Mid(ssnew, 9, 8(
 decr3 = Mid(ssnew, 17, 8(
 decr4 = Mid(ssnew, 25, 8(
 decr5 = Mid(ssnew, 33, 8(
 resdec1 = 0

 For I_BINdecr1 = 1 To Len(decr1(
    DIGIT_BINdecr1 = Mid(decr1, Len(decr1) - (I_BINdecr1 - 1), 1(
    If DIGIT_BINdecr1 = 1 Then
         resdec1 = resdec1 + 2 ^ (I_BINdecr1 - 1(
   End If
 Next I_BINdecr1

 Text1.Text = Text1.Text + Chr(resdec1(
 resdec2 = 0
 For I_BINdecr2 = 1 To Len(decr2(
   DIGIT_BINdecr2 = Mid(decr2, Len(decr2) - (I_BINdecr2 - 1), 1(

    If DIGIT_BINdecr2 = 1 Then
       resdec2 = resdec2 + 2 ^ (I_BINdecr2 - 1(
    End If
 Next I_BINdecr2

 Text1.Text = Text1.Text + Chr(resdec2(
 resdec3 = 0

 For I_BINdecr3 = 1 To Len(decr3(
   DIGIT_BINdecr3 = Mid(decr3, Len(decr3) - (I_BINdecr3 - 1), 1(
   If DIGIT_BINdecr3 = 1 Then
       resdec3 = resdec3 + 2 ^ (I_BINdecr3 - 1(
  End If
 Next I_BINdecr3
 Text1.Text = Text1.Text + Chr(resdec3(
 resdec4 = 0

 For I_BINdecr4 = 1 To Len(decr4(
  DIGIT_BINdecr4 = Mid(decr4, Len(decr4) - (I_BINdecr4 - 1), 1(
  If DIGIT_BINdecr4 = 1 Then
      resdec4 = resdec4 + 2 ^ (I_BINdecr4 - 1(
  End If
 Next I_BINdecr4

 Text1.Text = Text1.Text + Chr(resdec4(
 resdec5 = 0

 For I_BINdecr5 = 1 To Len(decr5(
  DIGIT_BINdecr5 = Mid(decr5, Len(decr5) - (I_BINdecr5 - 1), 1(
  If DIGIT_BINdecr5 = 1 Then
     resdec5 = resdec5 + 2 ^ (I_BINdecr5 - 1(
  End If
 Next I_BINdecr5
  Text1.Text = Text1.Text + Chr(resdec5(

Next DEC_KLOOP
 Open "d:\afterdecry.txt" For Output As #3
 Print #3, Text1.Text
 Close #3
End Sub
```

The following encrypted data was resulted from encryption of five bytes key partitioning (40 bit) and five bytes (40 bit) plaintext partitioning:

The plaintext is as shown above (plaintext.txt)

The secret key is: "Cryptography is the name for the study of procedures, algorithms, and methods to encode and decode information, Where, Cryptanalysis is the study of methods and means to defeat or compromise encryption techniques. Encryption usually"

8521384453	1
4512484862	1
-309518595667	1
64779992577	3
17314026496	1
-17348473428	1
56073973932	1
-4116230141	1
-2852848635	1
-287510892626	1
-4458069237	1
13086621692	1
-360660292297	1
61372841220	3
-34208743182	1
32972012023	1
61239080743	3
21647706783	1
-386547183599	1
-339319435849	1
-55733650253	1
4038526469	1
56429224869	2
-351998161101	1
-39007769600	1
30383270907	1
47081259009	1
102059357888	3
4092574714	1
60788048414	3
-296436780784	1
-46091272447	1
78034060059	3
-304949453994	1
4513267992	1
-359368428800	1
25753482485	1
12818515387	1
61032712207	3
-94674547466	1
-42950722543	1
-31176851715	1
43003888672	1
-334722896976	1
-68720002220	1
-86022355963	1
214597169400	1
-42832118017	1
-10078865675	1
288049170	3
-21155610546	1
-302298672395	1
-4139712255	1
-12873170958	1
35495345152	1
47462808323	1
-4459244536	1
12998061541	1
-21373849020	1
86975067412	3
-17012508686	1
64430079478	1
77043678740	3
67566092038	1
61220720126	1
-51270889986	1
-334840594425	1
101669532160	1
95273963331	2
-446689620426	1
-352271733485	1
-334688949331	1
47232124160	1
104544783	3
-22884185354	1
42181987098	3
-8846331716	1

Note: key partitioning and text partitioning (number of bytes selection with loop) possible to be changed according to sender and receiver agreement.

Table (3) (5 bytes plaintext partitioning 5 bytes key partitioning)

The following encrypted data was resulted from encryption of five bytes key partitioning (40 bit) and eight bytes (64 bit) plaintext partitioning:

The plaintext is as shown above (plaintext.txt)
The secret key is: "Cryptography is the name for the study of procedures, algorithms, and methods to encode and decode information, Where, Cryptanalysis is the study of methods and means to defeat or compromise encryption techniques. Encryption usually"

Mod Results Eq(1)	Div Results Eq(2)
282600706540	17270737
261059569856	16781145
281038632033	18373255
117697712096	16740844
108493316064	15410870
108984553824	16617879
309185714072	16251939
306622720536	16500860
388772022166	16335070
405807660102	19649301
334836689760	18430478
54207881982	6817569
303435137736	14323162
95741392672	16779166
57199461155	4977330
363315036781	17602709
19118839775	52528900
410108655972	5420303
325675083201	19342369
292269832677	15381718
120518857180	17544737
21303212542	1801158
138293847578	44259255
177241883925	15171539
389644215376	4755369
448012747523	15218661
220039748324	17890497
54040817120	60176110
216697384476	16911724
54072733147	61232832
87295913168	16888303
352878034752	17406826
25647717026	6747047
486187865344	17115316
96203010439	5419617
342119191392	14506808
279185215766	5450895
58720099795	15118615
25402309254	50467504
57019778278	16082093
325990487539	17425261
19446053226	17736563
322318485772	5356178
37336736624	1981954
426617611328	15043288
82090309548	16633770
282560851436	26226607
347449086256	4885266

Table (4) (8 bytes plaintext partitioning 5 bytes key partitioning)

8-Conclusions:

Information technologies are developing every day, computer security needs to be developed also, but in the security there is no perfect secure system, from this point it is difficult to measure how this proposed method is perfect, from our work we conclude:

1- The method (MODIV) or may be named as (Shift MODIVE) is strong enough to be used as a modern security system to encrypt the data.
2- One of the main characteristics of this proposed system is the simplicity and easy to understand there is no need to complicate the mechanism of encryption.
3- There are two levels of encryption; the first level is optional, while, the second level (which is the core) will use two equations to encrypt the data, and one equation to decryption.
4- The level of substitution (Shift level) it is not the core of the proposed system, it is to make the proposed system more strong, shift level not necessary in long bytes of key and plaintext partitioning.
5- The proposed method provides flexibility in the length of key partitioning and plaintext partitioning.
6- One of the main problems in the encryption methods is repetition of symbols, but according to the results of the proposed method there is no repetition, if there is any repetition it is impossible to be in Eq(1) and Eq(2) at the same time, may be the results of Eq(2) contains repetitions but Eq(1) is not, also the difference between the number of bytes in key partitioning and the number of bytes in plaintext partitioning must not be less than two bytes to avoid the problem of repetition, and to make the analyzing of the cipher text more difficult.
7- The secret key length is flexible.
8- Encryption of 8 bytes plaintext partitioning and 5 bytes secret key partitioning will take a time of 20 seconds for a plaintext file of size 380 Bytes and the size of the encrypted file is 820 Bytes, where encryption of 5 bytes plaintext partitioning and 3 byte secret key partitioning will take a time of 2 seconds for a plaintext file of size 380 Bytes and the encrypted file size is 1.2 KB, the system executed using a computer of 1.6 GHz and RAM of 500 Mb.
9- The main problem of the proposed system is the size of the encrypted file.
10- This proposed mechanism used ***n*** bytes of plaintext and ***m*** bytes of secret key, ***it is good to make the secret key as a string (may be a file) of unlimited length.***
11- MODIV technique corresponds to be symmetric encryption.
12- (Shift MODIV) software is ready for marketing; also it was programmed using Visual Basic 6.

Acknowledgment

I would like to thank and express my gratitude to Dr. Hussein Kettan Alkafaji for his help and support throughout preparing this paper.

References:

[1] A. Menezes, P. van Oorschot, and S. Vanstone, "*Handbook of Applied Cryptography*", CRC Press, New York, 1997.

[2] Bishop, M. " *Introduction to Computer Security*". Boston, Addison Wesley, 2005.

[3] C. Kaufman, R. Perlman, and M. Speciner. "*Network Security: Private Communication in a Public World*", Second Edition. Prentice-Hall PTR, 2002.

[4] Chang, R. " *Defending Against Flooding- Based Distributed Denial of Service Attacks: a tutorial*". IEEE Communication Magazine, October 2002.

[5] E. Fujisaki and T. Okamoto. "*Secure Integration of Asymmetric and Symmetric Encryption Schemes*". Advances in Cryptology -Proceedings of CRYPTO'99, Springer-Verlag, 1999.

[6] G. Lowe. "*Some New Attacks Upon Security Protocols*". In Proceedings of the 9th IEEE Computer Security Foundations Workshop, pages 162–169. IEEE Computer Society Press, June1994.

[7] Katzenbeisser, S. " *Information Hiding Techniques for Steganography and Digital Watermarking*", boston: Artech House, 2000.

[8] Kent, S. " *On the Trail of Intrusions into Information System*" IEEE spectrum, December 2000.

[9] M. Abdalla, M. Bellare, and P. Rogaway. DHAES: "*An Encryption Scheme Based on the Diffie-Hellman Problem*". Submission to IEEE P1363: Asymmetric Encryption, 1998.

[10] Pointcheval, D. " How to Encrypt Properly with RSA". Cryptobytes, winter/ spring 2002.

[11] Wenbo Mao Hewlett-Packard Company " *Modern Cryptography: Theory and Practice*" Publisher: Prentice Hall PTR Pub Date: July 25, 2003, ISBN: 0-13-066943-1

[12] William Stallings "*Cryptography and Network Security*" , 4'th edition, Prentice Hall, 2006.