MILITARY TECHNICAL COLLEGE
CAIRO EGYPT

Fast Calculation of the Modular Exponential Function in the RSA Cipher

*M.N. Elsherbiny, **M.N. Saleh, ***A. Elosmany, ***S. Elhabiby

ABSTRACT

Two different techniques are used to secure communication between military aircraft and control centers; the conventional secret key system and the public key system. In the public key system, the most famous technique is the RIVEST-SHAMIR-ADLEMAN (RSA) method.

In the RSA system, the modular exponential function is used to encipher the plaintext message and to decipher the ciphertext message. The problem considered here is how to speed-up the calculation of the modular exponential function.

An improved binary algorithm based on the binary redundant representa-tion (BRR) is proposed. This algorithm requires the minimum number of basic operations (modular multiplications) among all possible binary redundant representations. Compared to the binary algorithm, the proposed algorithm reduces the number of basic operations by 33%.

Systolic array of Montgomery is used to decrease the needed number of operations in the calculations. Results show that the time needed to compute the modular multiplication became less than 50% of the original time needed to perform the same operation without using systolic array of Montgomery.

The final results show that, the needed time to calculate the modular exponential function will decrease by 66% of the original time.

1- Introduction

Two different techniques are used to secure communication between military aircraft and control centers; the conventional secret key and the public-key systems. The main difference between these two systems, is the way keys are used. With a conventional key system, such as (DES) Data Encryption Standard [1], the same key is used for both encryption and decryption. With a public-key system, two keys are necessary, one is used in the conversion of cleartext to ciphertext, and the other is used to convert the ciphertext to cleartext.

* *Graduate Student,* **Faculty of Enginerring, Ain Shams University.*
*** *Military Technical Coilege, Cairo.*

In the public key system, the most famous technique is the Rivest-Shamir-Adleman (RSA) method. In the RSA algorithm, the public key is (e,n) and the secret key is (d). To encipher a message (m), the ciphertext is $c=(m)^e \bmod n$, and to obtain the cleartext, the algorithm compute $m=(c)^d \bmod n$.

There are different procedures to compute the exponential function. The factor method is based on a factorization of (e) [3]. If $e=p_1 q_1$, where p_1 is the smallest prime factor of (e) and $q > 1$, we calculate x^e by first calculate x^{e-1} and multiply by x. Repeated application of these rules gives a procedure for evaluating x^e, for any given (e), but there are cases where this method is not efficient (e = 33).

Addition chains is another method which gives the minimum number of multiplications for all of the relatively small values of (e) [4]. To calculate x^e, we find the optimal length of the additions of coefficients of the exponent (e), but for large enough (e), this method is not always an optimal procedure. A typical method of computing $(m)^e \bmod n$ is the approach of repeating modular squaring and multiplication [5]. Since it is based on the binary representation of the exponent (e), the algorithm is referred to as binary algorithm. The calculations require at most $2*\log_2(e)$ basic operations (modular multiplications). To reduce the total number of basic operations required by the binary algorithm, this paper introduces the normal form and modified normal form of the binary redundant representations [6]. A corresponding recoding procedure to convert the binary representation into the modified normal form of the binary redundant representation is also presented.

A binary redundant algorithm is proposed which uses the modified normal form of the integer as its input. The proposed algorithm requires a minimum number of basic operations among all possible redundant representations. It reduces the number of basic operations by 33%, on average, compared to the binary algorithm.

Using systolic array of Montgomery [7], we can calculate the modular multiplication without division and in the same time, we can decrease the needed number of multiuplications in the calculation. In section (2) we shall present the binary method to calculate modular exponential function. In section (3) we shall present the improved binary algorithm. In section (4) we shall show how to speed-up the modular multiplication calculation. In section (5) results are given. In section (6) conclusions are summarized.

2- Binary Algorithm

To compute $M^E \bmod N$, let $E = e_{n-1}, e_{n-2}, \ldots, e_0$ be the binary representation of the exponent E, where $e_i \in \{0,1\}$, $i = 0,1,\ldots,n-1$.

The binary algorithm can be described as follows :

Step 1 : $c = 1$

Step 2 : Repeat step 2a and step 2b for i = n-1, n-2, ..., 1, 0;

Step 2a : $c = c^2 \bmod N$

Step 2b : $\begin{cases} c\,M \bmod N & \text{if } e_i = 1 \\ c & \text{if } e_i = 0 \end{cases}$

Step 3 : Halt. c is the result.

The total number of basic operations (modular multiplications) of the binary algorithm depends on [6] :

(i) The length of the binary representation of integer E.

(ii) The total number non-zero digits in the binary representation of integer E.

The number (N) of basic operations during binary algorithm is [8] :

$$N = k + e_0 + e_1 + \ldots + e_{k-1}$$

So, $N < 2k$

The upper bound of (k) is

$$E = 2^k + e_{k-1} \cdot 2^{k-1} + \ldots + e_0$$

Thus, $2^k \leq E$

Taking logarithms, we get the inequality $k \leq \log_2 E$

So, $N \leq 2 \log_2 E$

To calculate the modular exponential function using binary algorithm, the calculations require at most $(2 \log_2 E)$ multiplications and $(2 \log_2 E)$ divisions [5].

3- An Improved Binary Algorithm

Consider computing $M^E \bmod N$, where $N < 2^n (n > 500)$, and both E and M are in the range (0, N-1), [6].

Definition 1 : A binary redundant representation (BRR) of an integer E, $E \geq 0$ is a binary radix polynomial

$$P(E) = b_{n-1} \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + \ldots + b_1 \cdot 2^1 + b_0$$

such that the decimal value of P(E) is equal E, where $b_i \in \{0, 1, -1\}$, $i = 0, 1, \ldots, n-1$.

In the following, $\bar{1}$ denotes (-1). The following algorithm is a modification of the binary algorithm by using (BRR) for the exponent E, $P(E) = b_{n-1} \ldots\ldots b_0$ as its input.

Step 1 : $c = 1$

Step 2 : Repeat step 2a and step 2b for $i = n-1, \ldots, 1, 0$;

Step 2a : $c = c^2 \bmod N$

Step 2b : $\begin{cases} c\ M \bmod n & \text{if} \quad b_i = 1 \\ c\ M^{-1} \bmod n & \text{if} \quad b_i = \bar{1} \\ c & \text{if} \quad b_i = 0 \end{cases}$

Step 3 : Halt. c is the result.

Note that, the value $\left(M^{-1} \bmod N\right)$ can be computed in a linear time, where the time unit is the time to perform a binary addition or subtraction.

Given an integer (E) in the binary representation, it is possible to recode it into (BRR) so that the number of non-zero digits of the (BRR) is less than the number of 1's in its binary representation. The problem will be how can we find a (BRR) of E, such that the corresponding number of basic operations excuted by the binary redundant algorithm will be minimum? First, we introduce a special form of (BRR) called the normal form, which is unique, and has the minimum number of non-zero digits over all possible BRR's of the integer.

Definition 2 : A BRR, $P(E)=\sum_{i=0}^{n-1} b_i 2^i$, $\quad b_i \in \{1,0,\bar{1}\}$, $\quad i=0,1,\ldots,n-1$, is the normal form of the binary redundant representation if and only if it satisfies the following condition.

Condition : For every pair of successive digits of P(E), b_i and b_{i+1} $(i = 0,1,\ldots,n-2)$, at least one of these two digits is zero.

$P_N(E)$ represents the normal form of E.

$W_P(E)$ represents the number of non-zero digits of P(E).

$W_N(E)$ represents the number of non-zero digits of $P_N(E)$.

$P_N(E)$ is unique and $W_N(E)=\min_{\text{all P(E)'s}} \{W_P(E)\}$.

The following procedure is used to convert an integer E from binary representation into normal form. Let $E=e_{n-1},e_{n-2},\ldots,e_0$ be the binary representa-tion of integer E>0. This procedure takes the sequence $e_{n+1},e_n,e_{n-1},\ldots,e_0$ as its input, where $e_{n+1}=0$ and $e_n=0$, assuming n is even.

The basic idea of the procedure NORMAL Fig.(1) can be described as follows :

(1) It checks three digits $(e_{i+2}\ \ e_{i+1}\ \ e_i)$ at a time, with one digit, e_{i+2}, overlapped with the next checking.

(2) Substitute the last two digits $(e_{i+1}\ \ e_i)$ according to the combination of $e_{i+2}\ \ e_{i+1}\ \ e_i$ and state of R, such that the normal form condition is satisfied.

(3) R is a binary variable which indicates whether there is a "carry" propagated from the previous substitution.

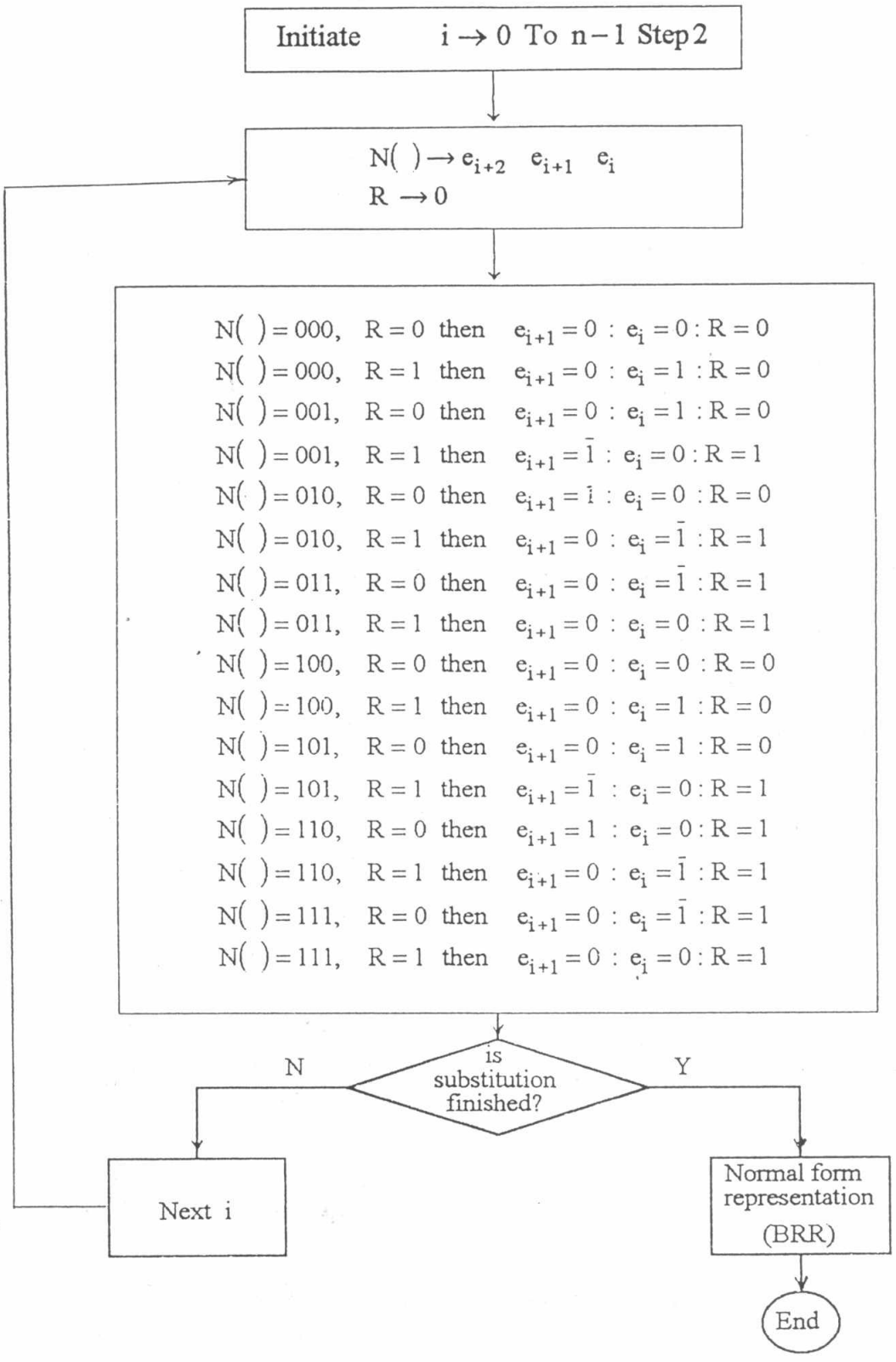

Fig.(1) *Procedure NORMAL*

Since there is only one loop in the procedure NORMAL and the step increment of the loop is two, the total number of steps is $\frac{n+2}{2}$. Each step of the loop is to check the current three digits and to substitute two of them.

The number of operations of the binary redundant algorithm depends not only on the number of non-zero digits, but also on the length of the (BRR) of the integer E, that is, $W_p(E)+L_p(E)$, where $L_p(E)$ is the length of the P(E).

Definition 3 : Suppose $P_N(E)=b_{n-1}\ b_{n-2}\ \ldots\ b_0$ is the normal form of E obtained by procedure NORMAL. We define a modified normal form, $P_M(E)$, as follows :

$$P_M(E)=\begin{cases} P_N(E) & \text{if} \quad b_{n-1}\,b_{n-2}\,b_{n-3}\neq 10\bar{1} \\ 11\,b_{n-4}\ b_{n-5}\ \ldots\ b_0 & \text{if} \quad b_{n-1}\,b_{n-2}\,b_{n-3}=10\bar{1} \end{cases}$$

It is noted that, $P_N(E)$ and $P_M(E)$ have the same value for the integer E, $W_M(E)=W_N(E)$ and $L_M(E)\le L_N(E)$, where $L_M(E)$ is the length of modified normal form of E.

Lemma 2 : For any integer $E>0$, $W_M(E)+L_M(E)=\min_{\text{all P(E)'s}}\{W_p(E)+L_p(E)\}$, [6].

Lemma 2 asserts that the binary redundant algorithm requires the minimum number of basic operations in terms of all possible BRR's of the integer E if it uses the modified normal form of E as its input.

4- Fast Calculation of Modular Multiplication

Modular exponential function will be fast in calculation using systolic array of Montogomery which calculate the modular multiplication of two positive integers. At first, we shall present some short notes about the problem of calculating modular multiplication of two positive integers without trial division [9]. It is based on selecting a radix value ($R > N$), such that R is coprime to N, and the computation modulo R are inexpensive to process. Let R^{-1} and N' be integers satisfying $0 < R^{-1} < N$ and $0 < N' < R$ and $RR^{-1} - NN' = 1$. The rationale behind this selection is to speed-up computation of $(TR^{-1} \bmod N)$ from T if $0 \le T < RN$. The algorithm REDC(T) will compute the value $TR^{-1} \bmod N$ (see Appendix).

If we have two integers x and y between (0 and N-1). Let $Z \equiv \text{REDC}(xy)$, then $t \equiv (xy)R^{-1} \bmod N$, so $\left(xR^{-1}\right)\left(yR^{-1}\right) \equiv zR^{-1} \bmod N$. So, z is the product of x and y, $0 \le z < N$. To convert an integer(x) to an N-residue, compute $\text{REDC}\left((x \bmod N)(R^2 \bmod N)\right)$. At the start of an algorithm, we should compute the constants once such as $N', R^{-1}, R \bmod N$.

Combine the idea of Montogomery on modular multiplication, and of Mc Canny and Mc Whirter [10] on systolic multiplication, the result of this combination will decrease the computational time of modular multiplication which will be at most 50%.
To calculate $(A \times B(\text{mod } N))$, Montogomery shows how to perform the calculation using the following rules :

(i) Reversing the order of treating the digits of the multiplicand, shifting down instead of up.
(ii) Adding multiples of the modulus.

Suppose that, m,n are the number of digits of modulus (M) and integer (A). Normally the integer (B) has at most the same number of digits as the modules (M), or at worst satisfies $B < 2M$. We express A,B,M in a radix (r), since (r) is a fixed power of 2. The radix is chosen so that the operations div r and mod r are trivial. The precondition for this algorithm is that there is no common factor between the radix (r) and the modulus (M) i.e. gcd(r,M) = 1, so we shall compute the multiplicative inverse $(r - M(0))^{-1} \bmod r$.

4-1 Systolic Array

A key property of Montogomery algorithm is that the choice of the modulus multiple which is based on the lowest digit P(0) of the output partial product. When this multiple Q_i is determined, the output partial product P_{i+1} is determined, and this partial output enable the corresponding digits of the next row. This is done by the array illustrated in Fig.(2), where each row performs an iteration of the loop and each column computes successive values for a single digit position.
From Fig.(2) we see that, the right most column has the burden of computing the digits of Q_i

For $i = 0$ To n

$$Q_i = \left((P_i(0) + A_i B(0))(r - M(0))^{-1}\right) \bmod r \qquad (1)$$

Carry entering the array from the right is zero, it is generated as follows

$$c = (P_i(0) + A_i B(0) + Q_i M(0)) \text{ div } r \qquad (2)$$

Suppose that the integer (B) intially satisfies $B < 2M$, then it must add another row to satisfy the above inequality, so there is (n+1) rows in the array and the integer(A) must be paded at the top end with extra digit i.e. A(n) = 0.

For $j = 1$ To $(m+2)$

The output partial product is

$$P_{i+1}(j-1) = \left(P_i(j) + A_i B_j + Q_i M_j + c\right) \bmod r \qquad (3)$$

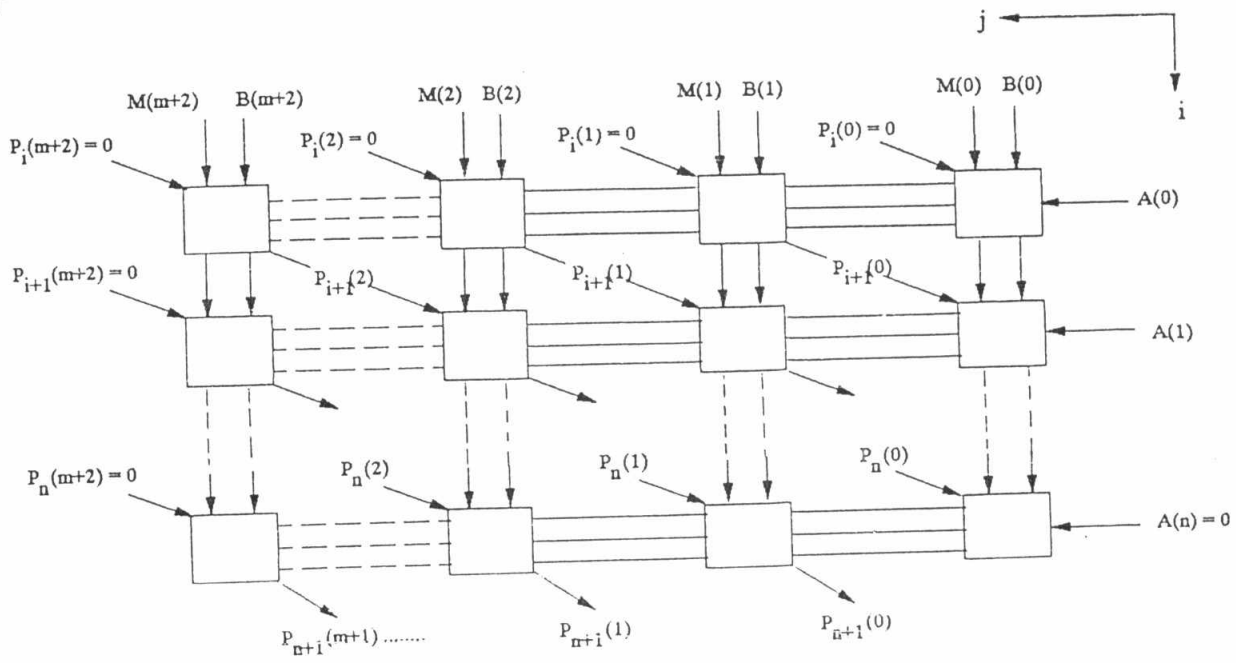

Fig.(2) *Arrangement of modular multiplication array*

5- Results

Lemma 2 asserts that the binary redundant algorithm requires the minimum number of basic operations of the integer E, if it uses the modified normal form of E as its input. Table (1) compares the number of non-zero digits on average in the binary representation with that in the modified normal form, for integers $E<2^{25}$ denoted by n_1 and n_2, respectively, where

$$n_1 = \frac{1}{2^{n-1}}\left(\sum_{i=0}^{2^{n-1}-1} W_B\left(2^{n-1}+i\right)\right) \quad \text{and} \quad n_2 = \frac{1}{2^{n-1}}\left(\sum_{i=0}^{2^{n-1}-1} W_M\left(2^{n-1}+i\right)\right)$$

Where

$W_B(E)$ is the number of non-zero digits of the binary representation of E,
$W_M(E)$ is the number of non-zero digits of the modified form of (BRR) of E, and
n is the length of integers.

In general, we can show that the saving can be up to 33% if E is large enough [6]. The modular multiplication output $P_{n+1}(j)$, where $(1 \le j \le m+1)$, is equal to $\left(r^{-n-1} AB \bmod M\right)$, or be (M) more than this because the output is bounded by 2M. The extra factor $\left(r^{-n-1}\right)$ can be removed after each modular multiplication by using the array

again to multiply $\left(ABR^{-n-1} \bmod M\right)$ and $r^{2n+2} \bmod M$. An extra (M) can be removed by subtracting (M) according to the sign of $\left(P_{n+1}(\ ) - M\right)$.

Table (1) *Comparison between the binary and modified normal form*

Number of digits (n)	Average number of 1's in binary representation (n_1)	Average number of non-zero's in modified normal form (n_2)	$\frac{n_1 - n_2}{n_1}$
10	6	4.44	26%
15	8.5	6.11	28.1%
20	11	7.77	29.4%
25	13.5	9.44	30.8%

From equation (3), we see that, in order to compute the modular multiplication of (AB mod M), we do not divide by the modulus (M) but we divide on the radix (r), since this division is linear in time, since unit time is the required to compute one binary additional subtraction. In the same time, if we increase the radix value (r), we can decrease the number of operations, since the number of elements of the systolic array will decrease. As a result, the computational time to calculate the modular multiplication will be at most 50% of the original time needed to perform the calculation without using systolic array of Montogomery. The final result will be, we can decrease the number of basic operations (modular multiplications) by 33% using the improved binary algorithm, and we can decrease time taken to compute the basic operation by at least 50% using the systolic array of Montogomery, so the total computational time of modular exponential function will decrease by 66% of the original time needed to calculate this function without using these improvements.

6- Conclusions

We have presented an improved binary algorithm to support the RSA cryptographic system. The proposed algorithm significantly reduces the number of basic operations by using the modified normal form of binary redundant representations if integers. It needs to convert an integer (E) from binary representation into modified normal form only once, and precompute the value ($M^{-1} \bmod N$) for a message M. We have shown that all of these operations can be done in linear time, where the time unit is the time period required to compute one binary addition/ subtraction or shifting. This algorithm reduces the number of basic operations up to 33%.

The structure of the systolic array of Montogomery reduces the time needed to calculate the modular multiplication, since the division and multiplication is done in linear time (not divided by the modulus M), this reduction is at least 50%.
The total time reduction to calculate the modular exponential function will be at most 66%.

Appendix

The algorithm to calculate (TR^{-1} mod N) is as follows :

FUNCTION REDC(T)

$m \leftarrow (T \bmod R) N' \bmod R$

$t \leftarrow (T + m N) / R$

IF $t \geq N$, then return t - N elase return t.

References

[1] Michael S. "Understanding data encryption standard", Mangement and Adimistration., Nov. 1989.

[2] R.M. Needham and Michael D. Schroeder, "Using encryption for authentication in large networks of computers", Communication of the ACM., Vol.21, No.12, December 1978.

[3] D.E. Knuth, "The art of computer programming", Vol.2, Seminumerical Algorithms, Addison-Wesly, New-York, 1969.

[4] Peter D., Beton. L., Ravi S., "Computing sequences with addition chains", SIAM J COMPUT., Vol.10, No.3, August 1981.

[5] R.L. Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signature and public key cryptosystems", Communication of the ACM, Feb. 1978.

[6] C.N. Zhang, "An improved binary algorithm for RSA", Computers Math. Applic. Vol. 25, No.6, 1993.

[7] Colin D. Walter, "Systolic modular multiplication", IEEE Transactions on Computers, Vol.42, No.3, March 1993.

[8] Denning, "Cryptography and data security", Addition-wesly, 1982.

[9] Peter L. Montogomery, "Modular multiplication without trial division", Mathematics of computation, Vol.44, No.170, April 1985.

[10] J.V. Mc Canny and J.G. Whirter, "Implementation of signal processing functions using 1-bit systolic arrays", Electronic letter, Vol.18, 1982.