2008 International Symposium on Nonlinear Theory and its Applications
NOLTA'08, Budapest, Hungary, September 7-10, 2008

RSA Encryption / Decryption Using Repeated Modulus Method

Hasan Amin Oseily & Ali Massoud Haidar
Electrical Department, Faculty of Engineering
Beirut Arab University
Beirut-Lebanon
E-mail: sasoha@yahoo.com, ari@bau.edu.lb

Abstract- This paper proposes the implementation of RSA encryption / decryption algorithm [1] using the simplification of repeated modulus in order to improve the performance. This algorithm will be compared with that based on Ancient Indian Vedic Mathematics [4] and the traditional algorithm [3] based on the straight division algorithm (sequential subtraction, multiplication, shifting..). The recently proposed hierarchical overlay multiplier architecture is used in the RSA circuitry for multiplication operation; the Carry save adder is used. The most significant aspect of the paper is the development of a division architecture based on straight division algorithm but with repeated operation of identical modulus and embedding it in RSA encryption / decryption circuitry for efficiency improvement. The examples show that RSA circuitry implemented using repeated identical modulus algorithm with division and multiplication is efficient in terms of speed compared to its implementation using conventional multiplication and division architectures.

1. Introduction

The standard techniques for providing privacy and security in data networks include encryption / decryption algorithms such as Advanced Encryption System (AES) (private-key) and RSA (public-key) [1,2,3]. RSA is one of the safest standard algorithms, based on public-key, for providing security in networks. While hardware implementation of this algorithm tends to be faster compared to its software counterpart, there is a scope for further improvement of performance of RSA hardware. One of the most time consuming processes in RSA encryption / decryption algorithm is the computation of (M^e *mod N)* where M is the text, (*e,N*) is the key[4]. This paper examines how this computation could be speeded up on the repeated modulus.

2. RSA Algorithm

The RSA cryptosystem, designated after its inventors Ron Rivest, Adi Shamir, and Len Adleman. The following section describes how to use the RSA cryptosystem.

2.1 Generating Keys

Let Alice selects two random prime numbers p and q and computes the products:

$$N = p \cdot q \tag{1}$$

$$K = (p-1).(q-1) \tag{2}$$

Additionally Alice selects a natural number e with

$$1 < e < K \quad \& \quad gcd\,(e, K) = 1 \tag{3}$$

And computes a natural number d with

$$1 < d < K \quad \& \quad d \cdot e = 1 \bmod K \tag{4}$$

Since gcd $(e,K) = 1$, there is actually such a number d. It can be computed with the extended Euclidean algorithm. We also consider that e is always odd. The public key consists of the pair (e, N). The private key is d. The number N is called RSA modulus, e is called encryption exponent and d is called decryption exponent.

2.2 Encryption

The finite set of plain text (M) consists of all natural numbers. A plaintext M is encrypted with

$$C = M^e \bmod N \tag{5}$$

Where C is the cipher-text

2.3 Decryption

The decryption is the reverse operation decryption. To decrypt a cipher text C using RSA algorithm

$$M = C^d \bmod N \tag{6}$$

Where M is the plain-text, d is the decryption key.

2.4 Security of RSA

To find out the secret RSA key is as difficult as factorizing the RSA modulus. However, the attacker can also have the intention to find the plain text from the cipher-text. It is not known whether it is therefore necessary to find the secret RSA key. But even if it could be proved that breaking RSA is as difficult as factorizing the RSA modulus, this would not automatically signify that RSA is totally secure, since there is no knowledge about the difficulty of factorizing natural numbers.

2.5 The Selection of *p* and *q*

To make the factorization of the RSA modulus N as difficult as possible, both prime factors p and q should be chosen the same size.

2.6 The Selection of *e*

The exponent e should be chosen in such a manner that the encryption is efficient, without decreasing the security. The

choice $e = 2$ is always excluded because $K = (p-1)\cdot(q-1)$ is even and $gcd(e, K)=1$ must be matched. Therefore, the smallest exponent would be three. With $e = 3$ a low exponent attack is going to be successful.

2.7 The Selection of *d*

The decryption exponent d has to be greater than 292; otherwise the RSA cryptosystem can be broken.

3. Algorithms to Compute Modulus

We will make here a quick revision for two algorithms to determine the quotient and the remainder (or the modulus). The first is the classical algorithm with repeated operation multiplication /subtraction for dividend by divisor. The other algorithm is based on the Vedic division algorithm of ancient Indian mathematics. After, we will introduce our algorithm with a brief comparison to other methods.

3.1 Integer Division Classical Algorithm

In general, we divide a number A with $2n$ bit by a number N with n *bit*. We get from this operation a quotient Q and remainder R. we can write:

$$A = Q * N + R \tag{7}$$

The remainder could be written by notation as:

$$R = A \bmod N \tag{8}$$

For example to divide 32 bits dividend by a 6 bits divisor, we use the same approach as multiplication where we make use of positional representation of binary integers to avoid simple repeated subtraction algorithm. But in higher valued logic, the matter will be different and much difficult.

3.2 Unsigned Binary Division Algorithm

To apply the division operation, we need one iteration per bit by subtracting the divisor from a partial remainder and test it to see if less than 0.

- If less than zero: divisor did not fit once, so left shift in a 0 in quotient and add divisor back to partial remainder.
- If >= zero: it fit, left shift a 1 in quotient and shift divisor register right.

For example when dividing two binary numbers with 32 bit dividend (11011101010101010101010101010111) and 6 bits for divisor (110110) here a lot of subtraction and shifting operation will be applied up to get the final results.

3.3 Vedic Division Algorithm

The Division architecture takes N bits of dividend and N bits of divisor to generate the quotient and the remainder. The architecture is based on Straight (At Sight) Division algorithm of Ancient Indian Vedic Mathematics [6]. To simplify the understanding of the algorithm, it is explained for 3 digits number ($X_2X_1X_0$) by 2 digits number (Y_1Y_0).

Steps:

- First do X_2/Y_0 (divide) to get Z_1 as quotient and C_1 as remainder.
- Call Procedure $ADJUST (Z_1, C_1, X_1, Y_1, Y_0)$.
- Now take the next dividend as

$$K = (C_1 * 10 + X_1) - (Y_1 * Z_1) \tag{9}$$

- Do K/Y_0 (divide) to get Z_0 as quotient and C_0 as Remainder,
- Call procedure $ADJUST (Z_0, C_0, X_0, Y_1, Y_1)$.
  Now our required remainder,

$$RD = (C_0 * 10 + X_0) - (Y_1 * Z_1) \tag{10}$$

And hence the Quotient

$$Qt = Z_1.Z_0 \tag{11}$$

The algorithm can be generalized for N digit/bit by N digit/bit but for numbers with great length, the number of equation will be also great.

4. Repeated Modulus Algorithm

Based on the arithmetic properties of modulus, we can develop an algorithm to simplify the operations of division when encrypting a plain text in order to get a cipher text. Let A, B, C and D are natural numbers then,

$$(A*B*C*D \bmod N) = [(A \bmod N) * (B \bmod N) * (C \bmod N) * (D \bmod N) * .....] \bmod N \tag{12}$$

This reduces the intermediate results to modulo N and makes the calculation practical.

The division architecture takes $2N$ bits for dividend and N bits for divisor to generate the quotient and the remainder. The architecture is based on the straight division algorithm of repeated modulus. To simplify the understanding of algorithm, it is explained in the following steps.

Based on the RSA algorithm, Let M be a plain text where M is a binary digit with n bits and e is the encryption exponent. Based on the equation 5, the cipher text will be:

Cipher = $M^e \bmod N$

Where,

(N = p . q) and e < N

Then M is multiplied by itself e times. Applying the equation (12),

$$Cipher = M*M*M*M*M... \bmod N = [(M \bmod N)*(M \bmod N)*...] \bmod N \tag{13}$$

To analyze this equation, we have two cases:

4.2 For M > N

In this case, the following steps are required

- Step 1: We compute only one cell $(M \bmod N) = R_1$ (remainder) since all cells are repeated or identical.
- Step 2: Replacing all cells by the remainder R_1 in the equation (13), we get

$$(R_1 * R_1 * R_1 * R_1 ....) \bmod N = R^e \bmod N \quad (14)$$

- Step 3: Verify if $R^e < N$
  -If yes, then the final remainder is $RD = R^e$
  -If no, then we go to the next step.
- Step4: Splitting the term R^e into sets as,

$$x * z + y = e \quad (15)$$

  Where,
  $R^x >= N$.
  $R^y < N$, then R^y is a remainder
  z is the number of term R^x
- Step 5: Replacing the term R^e by $R^{(x * z + y)}$ in the equation 14, we get

$$R^e \bmod N = R^{(x * z + y)} \bmod N = (R^{(x * z)} * R^y) \bmod N$$
$$= [(R^x * R^x * R^x * ....) \bmod N * R^y] \bmod N \quad (16)$$

- Step 6: We repeat this computation until getting all terms less than the number N.

4.2 Example 1

Let a message M with 6 bits M=101000=40 to be encrypted and the encryption key e=5=101, the prime numbers p=5=101 and q=7=111

Then:

- N=5 * 7 =35 = 100011 ($M>N$)
  $M \bmod N$ = 101000 *mod* 100011=5=101=R
- Computing $[R * R * R * R * R] \bmod N = R_1$
  $=[5*5*5*5*5] \bmod 35$.
- Grouping into sets the remainder R
  $R*R=5*5<35=N$
  $R*R*R=5*5*5=125>35$
- Then we have only two sets with $e = z * x + y = 5$
  $y=2$
  $x=3$
  $z=(5-2)/3 =1$
- Compute ($R * R * R \bmod N$ = 125 mod 35 = 20 = R_1)
  The second set $R*R=25<35$ then $R2=25$
  Check $R1*R2= 20*25=500>N$ then repeat the computation
  500 mod 35 = 10
- To prove this result we can apply directly:
  $M.M.M.M.M \bmod N = 40*40*40*40*40 \bmod 35$
  $102400000 \bmod 35 = 10$

Discussion: comparing the biggest cells of modulus (500) with the original number (102400000), we found that division of the number 500 by 35 is much easier than 102400000 by 35.

4.3 For M < N

In this case, we go directly to step 4, but with $R=M$

- Step 4: Splitting the term M^e into $M^{(X*Z+Y)}$
  Where:
  $M^x >= N$.
  $M^y < N$
  z is the number of term M^x
- Step 5: Replacing the term M^e by $M^{(x*z+y)}$, Then

$$M^e \bmod N = M^{(x * z + y)} \bmod N = (M^{(x * z)} * M^y) \bmod N$$
$$= [(M^x * M^x * M^x * ....) \bmod N * M^y] \bmod N \quad (17)$$

  The terms of multiplication $(M^X \bmod N)$ are identical and consisting of identical cells.
  We compute only one of these cells $(M^X \bmod N = R_1$ where R_1 is a remainder with $R_1<N$)
- Step 6: Replacing M^x, M^y in the equation (13)

$$(R_1 * R_1 * R_1 * R_1 ....M^y) \bmod N = R^{e-y} * M^y \bmod N \quad (18)$$

- Step 7: Verify if $R^{e-y} * M^y < N$,
  If yes, then the final remainder is $RD= R^{e-y} * M^y$
  If no, then we do repetition of the algorithm or we refer to step 5 with same sequence and logic.
- We repeat the computation up to get a product of all remainders less than the divisor N.

4.4 Example 2

Let we have a message M to be encrypted with 6 bits M=101000=40 and e=3=11, let p=7=111 and q=7=111

Then:

- $N=7*7=49=110001$ ($M<N$)
- Grouping $M*M*M$ into two sets
  $M*M=40*40>49=N$
  $M=40<49=N$
- Then we have only two sets with $e = z * x + y=5$
  $y=1$
  $x=2$
  $z=(3-1)/2 =1$
- Compute $M*M \bmod N = 1600 \bmod 35 = 32 = R1$
- The second set $M=40<49$ then $R2=40$
- Check $R1*R2$= 32*40=1280>N then repeat the computation
  1280 mod 49 = 6
- To prove this result we can apply directly:
  M*M*M mod N = 40*40*40 mod 49
  64000 mod 49 = 6

5. Flow Chart

The flow chart of figure 1 shows the sequence of operations of division and multiplication. It is very clear that division operations are minimized to the least possible case.

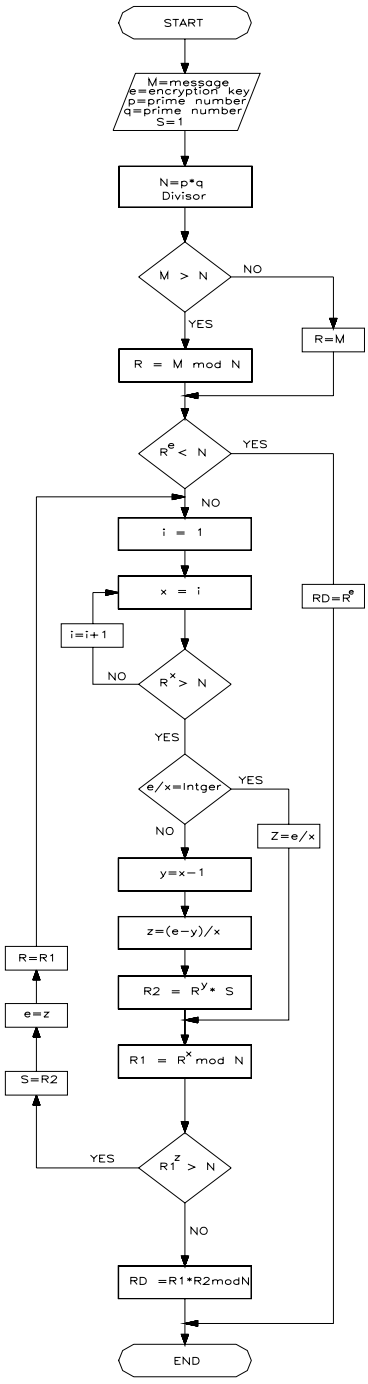

Figure 1. Flow Chart of RSA Repeated Modulus Algorithm

6. Analysis of Algorithms

The RSA encryption/decryption circuitry achieves a significant improvement in performance using the repeated or identical modulus algorithm as reflected by the results shown in examples compared with traditional division algorithm. In cryptography, we care only about the remainder or the modulus to get the cipher and plain text and we do not take any care to get the quotient. Hence, the main advantage of this algorithm is the simplicity comparing with other techniques and the uniqueness of output (remainder instead of remainder + quotient). Then it is found that when implemented with this algorithm, the RSA circuitry has less timing delay compared to its implementation using traditional multipliers and division algorithms. We can see that multiplication and division operations are two of the most important operations in computation of A^B mod N and a high performance multiplication and division algorithm/ architecture will considerably improve the speeds of encryption and decryption. Two known methods of multiplication are array and booth multiplication each with its own limitations [5,6]. From the architecture point of view, division circuits are usually much larger than multiplier circuits for an equivalent data word length and division is generally performed through restoring and non-restoring algorithms [8,9]. A faster and novel hierarchical overlay multiplier has earlier been proposed based on Ancient Indian Vedic Mathematics [10] that performs better than the conventional multiplier architectures [6]. While this paper still utilizes the same concept in computation of A^B *mod N*, but it proposes a novel and optimized algorithm to minimize the length of dividend number.

7. Conclusion

The RSA encryption / decryption implemented with repeated or identical modulus algorithm has improvement in efficiency in terms of speed and area. It has the advantage that as the number of bits increases the execution time of operations increase very slowly as compared to RSA encryption employing traditional multipliers and division algorithms [6].

References

[1] R. Rivest, A. Shamir and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. *Communications of the ACM*, 21 (2), pp. 120-126, February 1978.

[2] Daeman, J., and Rijmen, V. “Rijndael : “The Advanced Encryption Standard.” , *Dr. Dobb's Journal*, March2001.

[3] Daeman, J., and Rijmen, V. “ The Design of Rijndael: The Wide Trail Strategy Explained”, New York,
Springer- Verlag, 2000.

[4] William Stallings, “ Cryptography and Nework Security”, Third Edition, Pearson Education, 2003

[5]Himanshu Thapliyal, R.V Kamala and M.B Srinivas "RSA Encryption/Decryption in Wireless Networks Using
an Efficient High Speed Multiplier", Proceedings of IEEE International Conference On Personal Wireless
Communications (ICPWC-2005) , New Delhi, pp-417-420, Jan 2005.

[6] Himanshu Thapliyal and M.B Srinivas, "High Speed Efficient Hierachical OverlayMultiplier Architecture
Based on Ancient Indian Vedic Mathematics", Proceedings of International Conference on Signal Processing,
ICSP 2004, Turkey, Dec 2004.

[7] M.M. Mano, “Computer System Architecture”, 2nd Ed, Prentice Hall, 1982.

[8] V.C. Hamacher, Z.G. Vranesic, S.G. Zaky, “Computer Organisation”, PP-281-285, 4th Ed, The Mcgraw Hill
Company, 1996.

[9] J.P. Hayes, “Computer Architecture and Organisation”, PP-244-250, 3rd Ed, The Mcgraw Hill Company,1998.

[10] Jagadguru Swami Sri Bharath, Krsna Tirathji, “Vedic Mathematics or Sixteen Simple Sutras From TheVedas”, Motilal Banarsidas , Varanasi(India),1986.