Reduction of Large Integers by Random Modulus in Public-Key Cryptosystems

Plamen Stoianov[1]

Abstract – **Public-key cryptography is often considered to be too computationally expensive for devices if not accelerated by cryptographic hardware. The most asymmetric cryptographic algorithms used modular operationals $X = A^E \bmod M$ for large integers. These operations determine the data processing speed. The paper presents algorithm for calculating modular reduction without division and multiplication. These operations replaced with rotation and subtraction.**

Keywords – ***P*ublic-key cryptosystems, modular reduction, pre-calculation, integer arithmetic.**

I. INTRODUCTION

The need for information security has grown steadily over the years. Users require protection of information from unauthorized access and alteration. Essential tool for achieving these objectives is the use of cryptography. In simplified terms, there are three types of data in encryption technology. The first is plaintext, which is unencrypted data. Encrypted data is referred to as ciphertext. The third is a key, one or more of which is required for encryption and decryption. These tree types of data are processed by an encryption algorithm. Cryptology can be split into two areas of activity, namely cryptography and cryptanalysis. Cryptography is the study of the methods used for encrypting and decrypting data. Goal of cryptanalysis is to develop methods and tools for the revealing of cryptographic systems and evaluate their security.

Modern cryptographic algorithms are generally based on Kerckhoff's principle. This principle says that the entire security of an algorithm should be based only on the on the secrecy of the key, and not on the secrecy of the cryptographic algorithm. The opposite of Kerckhoff's principle is the principle of security by concealment. With this principle, the security of a system is based on the idea that a would-be attacker does not know how the system works. Up to now, every system based on this principle alone has been broken, usually in a very short time[6].

Cryptographic techniques are fundamental to the implementation of security services and may be divided into two classes: symmetric-key and public-key cryptography.

Symmetric-key cryptography requires a single secret key that is used for both encryption and decryption hence the designation 'symmetric'. The exchange of this secret key forms part of the key management problem, that is concerned with the secure distribution of key to the communicating parties. The two types of symmetric-key algorithms are block ciphers and stream ciphers. Block ciphers operate on a block of data while stream ciphers encrypt individual bits. Block chippers are typically used when performing bulk data encryption and the data transfer rate of the connection typically follows the encryption/decryption throughput of the implemented algorithm. The most widely used symmetric cryptographic algorithm (know as Feistel's ciphers) are Triple DES, AES, IDEA etc [11].

A major advance in cryptography came in 1976 with the publication by Diffie and Helman (New Directions of Cryptography) of the concept of public-key cryptography. This new concept that would revolutionize cryptography as it was known at the time. The primary feature is that it removes the need to use a single key for encryption as well as decryption. Pair of matched keys is used, termed 'public' and 'private' keys. The public part of the key pair can be distributed publicly without compromising the security of the private key, which must be kept secret by the receiver. A message encrypted with the public key can only be decrypted with the corresponding private key. The key management problem is greatly simplified by the use of public-key cryptosystems.

Most public-key cryptosystems used today are based on the difficulty of factorizing large integers as well as the difficulty to compute the discrete logarithm of a large integer. The implementation of these public-key cryptosystems requires modular exponentiations.

II. OVERVIEW OF ALGORITHMS FOR MODULAR REDUCTION

The operational speed of public-key cryptosystems is largely determined by the modular exponentiation operation of the form $X = A^E \bmod M$ where X is the remainder, A is the base, E is the exponent and M is the modulus. The modular multiplication operation is accomplished using two steps. It first computes a large-integer multiplication step followed by a modular reduction step. The required modular exponentiation is computed by a series of modular multiplications [8]. The RSA cryptosystem uses modular arithmetic algorithms with large integers in the range of 512 to 2048(more than 600 decimal digits) bits.

The RSA cryptosystem, named after its inventors Rivest, Shamir and Adleman, is the most widely used public-key cryptosystem[4]. Its very simple operating principle is based on the arithmetic of large integers. The two keys are generated from two large prime numbers [10]. The encryption and

[1]Plamen Stoianov is with the Technical University of Varna, Telecommunications Department, Studentska 1 , Varna, Bulgaria, E-mail: pl63@abv.bg.

decryption processes can be expressed mathematically as follows:

encryption: $y = x^e \bmod n$

decryption: $x = y^d \bmod n$

where x = plaintext
y = ciphertext
e = public key
d = private key
n = p.q =public modulus
p,q = secret prime numbers

Before being encoded, the plaintext block must be padded to the appropriate block size, which varies in the RSA algorithm according to the length of the key used. Encryption itself is performed by exponentiation of the plaintext followed by a modulus operation. The result of this process is the ciphertext. This can only be decoded if the private key is known. The decryption process is analogous to the encryption process. The security of the algorithm is based on the difficulty of factoring large numbers. It is quite easy to compute the public modulus from the two prime numbers by multiplication, but it is very difficult to decompose the modulus into its two prime factors, since there is no effective algorithm for this operation. Way to increase the speed of the RSA algorithm is to use the Chinese Remainder Theorem. Prerequisite for using the CRT is that both of the secret prime number p and q are known, which means that it can only be used for decryption [3].

A basic operation in public-key cryptosystems is the modular reduction X=AmodM of large numbers. An efficient implementation of this operation is the key to high performance. In many cases the modulus M is fixed. The fact that M is constant makes it feasible to precompute some values ahead of time which typically results in avoiding divisions and replacing them by multiplications [9].

The Classical, Barrett and Montgomery algorithms are well known modular reduction algorithms for large integers used in public-key cryptosystems. Each algorithm has its own unique characteristics resulting in a specific field of application.

Classical algorithm is a formalization of the ordinary t-n step pencil and paper method, each step of which is the division of a (n+1)-digit number M by the n-digit divisor M, yielding the one-digit quotient Q and n-digit remainder R. Each remainder R is less than M, so that it can be combined with the next digit of the dividend into the (n+1)-digit number Rb+(next digit of dividend) to be used as the new X in the next step[7]. The algorithm is as follows:

```
Input  : A = $\sum_{i=0}^{t-1} a_i b^i$ , M = $\sum_{i=0}^{n-1} m_i b^i$
Output : X = $\sum_{i=0}^{n-1} x_i b^i$ = A mod M
1. X $\leftarrow$ A
2. While X $\geq$ M$b^{t-n}$ do X$\leftarrow$X - $b^{t-n}$
3. for i = t-1 to n-t+1 step -1 do
   if $r_i = m_{n-1}$ then q = b – 1
                  else q = $r_i$ b+$r_{i-1}$ div $b^{t-n}$ $m_{n-1}$
   3.2 While q($m_{n-1}$b+$m_{n-2}$) > $a_i b^2 + a_{i-1} b + a_{i-2}$
           do q $\leftarrow$q – 1
   3.3 X $\leftarrow$ X - q$Mb^{i-n}$
   3.4 if X < 0 then X $\leftarrow$ X + $Mb^{i-n}$
```

Step 3.2 can be modified to :

qm_{n-2} > ($a_i b + a_{i-1} - q m_{n-1}$)b + a_{i-2} .

Since $a_i b + a_{i-1} - q m_{n-1}$ < m_n, this step can be done in two multiplications (plus one comparison of two-digit numbers). Thus this algorithm requires n(n+2) multiplications and n divisions for 2n-bit dividend [8].

P. Montgomery introduced an efficient algorithm for modular multiplication without explicitly carrying out the classical modular reduction step[5]. By representing the residue classes modulo m in a nonstandard way, Montgomery's method replaces a division by m whit a multiplication followed by a division by a power of b. The m-residue with respect to R = b^k of an integer x < m is defined as xR mod m. The Montgomery reduction of x is defined as xR^{-1}mod m, where R^{-1} is the inverse of R modulo and is the inverse operation of the m-reside transformation. It can be shown that the multiplication of two m-residues followed Montgomery reduction is isomorphic to the ordinary modular multiplication. The rationale behind the m-residue transformation is the ability to perform a Montgomery reduction xR^{-1}mod m for 0 $\leq$ x < Rm in almost the same time as a multiplication. If x is the production of two m-residues, the result is the m-residue of the remainder, and the remainder itself is obtained by applying one additional Montgomery reduction. Instead of computing all of t at once, one can compute one digit t at a time, add $t_i m b^i$ to x, and repeat[7]. This change allows the computation of

m'_0 = - m_0^{-1} mod b instead of m'. The algorithm is as follows:

```
for i=0 ; i < k ; i++ do {
  $t_i$ = (X* $m'_0$ ) mod b
  x = x + $t_i m b^i$ }
  x = x div $b^k$
  if (X$\geq$m) then
  x = x - m
```

Barrett reduction was inspired by fast division algorithm that multiply the reciprocal of the divisor to emulate division. This reduction technique is advantageous in a modular exponentiation where many reductions are performed with the same modulus[1]. It was the first approach to perform reduction without explicitly using the division step in the loop. P.Barrett introduced the idea of estimating the quotient Q = A div M with operations that either are less expensive than a multiprecision division by M [2]. The estimate for Q' of A div M is obtained by replacing the floating-point divisions in

Q = $\lfloor (A/b^{2k-t})(b^{2k}/M)/b^t \rfloor$ by integer division

Q' = ((A div b^{2k-t}) μ) div b^t where $\mu = \left\lfloor \frac{b^{2k}}{M} \right\rfloor$

The number of multiplications and the resulting error is more or less independent of t. The best choice for t, resulting in the least number of operations and the smallest maximal error is t=k+1. The algorithm is follows:

```
Input  : A = $\sum_{i=0}^{2k-1} a_i b^i$ , M = $\sum_{i=0}^{k-1} m_i b^i$
Output : X = $\sum_{i=0}^{k-1} x_i b^i$ = A mod M
1. Pre-calculation
   1.1 $\mu = \left\lfloor \frac{b^{2k}}{M} \right\rfloor$
2. Calculation of the quotient
   2.1 Q $\leftarrow \left\lfloor \frac{A}{b^{k+1}} \right\rfloor \mu$
   2.2 Q' $\leftarrow \left\lfloor \frac{Q}{b^{k+1}} \right\rfloor$
3. Compute the remainder
   3.1 $R_1 \leftarrow$ A mod $b^{k+1}$
   3.2 $R_2 \leftarrow$ (Q'*M) mod $b^{k+1}$
   3.3 $R \leftarrow R_1 - R_2$
4. Correction of the result
   4.1 if R < 0 then R + $b^{k+1}$
   4.2 while R $\geq$ M do R $\leftarrow$ R - M
```

III. Proposed Algorithm

For computing X=AmodM without division and multiplication the following algorithm is suggested:

```
Input  : A = $\sum_{i=0}^{t-1} a_i 2^i$ , M = $\sum_{i=0}^{n-1} m_i 2^i$ , t > n
Output : X = $\sum_{i=0}^{t-n-1} x_i 2^i$ = A mod M
1. Pre-calculation
   1.1 X $\leftarrow$ A mod $2^n$ , S $\leftarrow 2^n$ - M
   1.2 While X $\geq$ M do X $\leftarrow$ X – M
   1.3 While S > M do S $\leftarrow$ S – M
2. Computation X
   2.1 for i = n ; i < t ; i++ do {
   2.2 if $a_i$ = 0 then step 3
   2.3 X $\leftarrow$ X + S
   2.4 if X $\geq$ M then X $\leftarrow$ X – M
3. Correction S
   3.1 S $\leftarrow$ S + S
   3.2 if S $\geq$ M then S $\leftarrow$ S – M }
4. return ( X )
```

Step 1 involves pre-calculated of X and S. The expression of A may be written in the following way:

A = $\sum_{i=0}^{t-1} a_i 2^i$ mod M = ($\sum_{i=0}^{t-n-1} a_i 2^i * 2^n + \sum_{i=0}^{n-1} a_i 2^i$) mod M = $\sum_{i=0}^{t-n-1} a_i 2^i$ mod M * S + X

where X = $\sum_{i=0}^{n-1} a_i 2^i$ mod M and S = 2^n mod M

In algorithm RSA t $\leq$ 2n because always A < M^2

In step 2, the current bit a_i is checked and if it is =1 the current value of X is corrected.

Step 3 is related preparation of S for the next cycle i+1

If the checking of i is performed before step 3 , the calculation will be reduced by time for thelast preparation of S.

In base b>2 the following algorithm is suggested:

```
Input  : A = $\sum_{i=0}^{t-1} a_i b^i$ , M = $\sum_{i=0}^{n-1} m_i b^i$ , t > n
         b = $2^k$
Output : X = $\sum_{i=0}^{t-n-1} x_i b^i$ = A mod M
1. Pre-calculation
   1.1 X $\leftarrow$ A mod $b^n$ , S $\leftarrow b^n$ - M
   1.2 While X $\geq$ M do X $\leftarrow$ X – M
   1.3 While S > M do S $\leftarrow$ S – M
2. Computation X
   2.1 for i = n ; i < t ; i++ do {
   2.2 for j = 0 ; j < b ; j++ do {
   2.3 if $a_{i+j}$ = 1 then X $\leftarrow$ X + S
   2.4 if X $\geq$ M then X $\leftarrow$ X – M
3. Correction S
   3.1 S $\leftarrow$ S + S
   3.2 if S $\geq$ M then S $\leftarrow$ S – M }}
4. return ( X )
```

When is selected base b > 2 , he number of the external cycles is reduced.

When there is a larger bulk of operation memory it is possible to reduce the operating time. In step 1 is calculated:

$S_k \leftarrow b^{n+k}$ - M for k=0 to b-1

Calculated values for S_k are used to calculate X $\leftarrow$ X + S without multiplication in step 2. In this case, step 3 is outside of the internal cycle

IV. Conclusion

In the known algorithms for modular reduction pre-calculations are carried out in order to change the module to b^2 of 2 for faster processing of blocks of data. These calculations involve multiplication and division of large integers. The proposed algorithm uses only elementary operations of rotation, addition and subtraction without division and multiplication.

The algorithm can be employed in applications using microcontrollers with smaller computing capabilities without hardware multipliers. In addition, the efficiency and reliability of the algorithm is higher when processing small amounts of data due to the elementary pre-calculations. Therefore, it can be used to exchange session keys for symmetric algorithms.

References

[1]]. W. Hasenplaugh, G. Gaubatz, V. Gopal,“Fast Modular Reduction”. IEEE Symposium on Computer Arithmetic pp.225-229, 2007

[2] P. Barrett. “Implementing the Rivest Shamir Adleman public-key encryption algorithm on a standart digital signal processor”. Advances in Cryptology – CRYPTO’86, pp.311-323, 1987

[3] T.R. Rao. “Aryabhata Remainder Theorem : Relevance to public-key crypto algorithms”. Symposium on Cryptography and Information Security. Japan 2005.

[4] R. Rivest, A. Shamir, L. Adleman. “A method for obtaining for digital signatures and public-key cryptosystems”. CACM, vol.21, pp.120-126, 1978

[5] P. Montgomery. “Modular multiplication without trial division”. Mathematics of Computation, vol.44, pp.519-521, 1985.

[6] R. Wolfgang, E. Wolfgang, “Smart card handbook” 3rd edition, November 2003.

[7] A. Bosselaers, R. Govaerts and J. Vandewalle, “Comparison of three modular reductions”, Advances in Cryptology – Crypto ’93 (LNCS 773), Springer-Verlag, pp. 175-186, 1994.

[8] Chia-Long Wu. “Fast modular multi-exponentiation using modified complex arithmetic”. Applied Mathematics and Computation, pp.1065-1074, 2007.

[9] N. Gura, A. Patel, A. Wander, H. Eberle, S. Shantz, “Comparing Elliptic Curve Cryptogrraphy and RSA on 8-bit CPUs” Proceedings of CHES’2004. pp.119-132

[10] D. Boneh, H. Shacham, :Fast variants of RSA”. Crypto Bytes, vol. 5, No 1, pp. 1-9,2002

[11] T. Wollinger, J. Guajardo, Ch. Paar,:Cryptography in Embedded Systems:An Overview”. Proceedings of the Embedded World 2003, Design & Elektronik, Germany, 2003, pp. 735-744