A Fast Modular Arithmetic Algorithm Using a Residue Table (EXTENDED ABSTRACT)

Shin-ichi KAWAMURA and Kyoko HIRANO

TOSHIBA CORPORATION
RESEARCH AND DEVELOPMENT CENTER

1. INTRODUCTION

Many public key cryptosystems and key distribution systems have been developed making use of a one-way (trap door) function $x \mid\!\!\rightarrow y$ such that $y=a^x \bmod p$ or $y=x^e \bmod n$. Modular multiplication is indispensable for computing these functions. In other words, fast multiple precision modular arithmetic will become increasingly useful for realizing an efficient security system using a public-key cryptosystem, like RSA[1], Rabin's scheme[2], and so on.

Several methods using a pre-computed residue table have been proposed for the efficient computation of A*B modulo a large integer N. In these methods, the size of the number to be processed is successively reduced in each stage of the computation by using a congruent relation over the modulo N. The method proposed in this paper is also included in this category. It achieves further table size reduction by recursively applying the same table to different digits of the number to be processed.

2. BASIC RULES

The basic idea for table lookup is very simple. If one wants to know the value of X mod N for a fixed N frequently for various X, then it is helpful for him to compute and store the value of X mod N for many X in advance. However, the pre-computed residue table must be reduced to a reasonable size because a full-scale exhaustive pre-computation is impossible in principle. (Note

C.G. Guenther (Ed.): Advances in Cryptology - EUROCRYPT '88, LNCS 330, pp. 245-250, 1988.

that the security of the RSA scheme is based on this fact.) So the following rules are applied for the table reduction. Bold printing represents pre-computed terms.

(1) $(A*2^{u} + B) \bmod N \equiv (A*2^{u} \bmod N) + B \pmod N$

(2) $(A1*2^{b} + A2) \bmod N \equiv (A1*2^{b} \bmod N) + (A2 \bmod N) \pmod N$

(3) $(A*2^{u} + B) \bmod N \equiv (A \bmod N)*2^{u} + B \pmod N$

Rule (1) means that in making the table, one may ignore the lower portion of X which is less than N. Rule (2) means that the table should be divided into some segments. The table for $(A1*2^{b} + A2) \bmod N$ is always greater than the summation of the two tables, $(A1*2^{b} \bmod N)$ and $(A2 \bmod N)$. The self-evident rule (3), which is introduced in this paper, enables the repeated use of one table to any digit. The method in [3]-[5] is derived by applying the above two rules, (1) and (2). The next section describes our method based on the additional rule (3).

3. TABLE-LOOK-UP

In order to formulate the problem, it is assumed that Xj, the number to be processed in the j-th stage, is divided into lj blocks and that each block consists of b bits. Then

$$X_0 = A \times B \qquad \text{Eq.(1)}$$

$$X_j = \sum_{i=1}^{\ell_j} x_i^{(j)} \cdot 2^{b\cdot(i-1)} \qquad \text{Eq.(2)}$$

Now, Xj+1 should be so defined that it satisfies the following reduction condition;

$$\begin{cases} X_j \equiv X_{j+1} \pmod N \\ X_j > X_{j+1} \end{cases} \qquad \text{Eq.(3)}$$

Two alternative definitions for Xj+1 are derived;

Definition 1:

$$X_{j+1} = \sum_{i=k+1}^{\ell_j} \underline{(x_i^{(j)} \cdot 2^{b\cdot(i-1)} \bmod N)} + \sum_{i=1}^{k} x_i^{(j)} \cdot 2^{b\cdot(i-1)} \qquad \text{Eq. (4)}$$

where k is an integer which satisfies

$$2^{k\cdot b} - 1 < N \leq 2^{(k+1)\cdot b} - 1 \qquad \text{Eq. (5)}$$

and Definition 2:

$$\begin{cases} X_{j+1} = \sum_{i=1}^{\ell_j - 1} x_i^{(j)} \cdot 2^{b\cdot(i-1)} + \underline{(x_{\ell_j}^{(j)} \cdot 2^{u} \bmod N)} \cdot 2^{t_j} \\ t_j = b \cdot (\ell_j - 1) - u \end{cases} \qquad \text{Eq. (6)}$$

where u is the number of bits of modulo N.

Definition 1 can be called a parallel table lookup method and Def. 2 is named a recursive table lookup method. The underlined terms in the above equations have 2^b values each. They can be pre-computed and stored in memory if b is a modest value. As a result, modular arithmetic is executed not by division, but by table-lookup and addition. Definition 1 appears in some of the former papers. As described in section 2, the main idea of this method is that the memory size is reduced by dividing the number into blocks. Definition 2 is our proposal. The table in this method is independent of the block number (i) . The same table is applied to any portion of the number to be processed. Accordingly, the size of the table is reduced by a large factor. Furthermore, Def. 1's idea that the table size is reduced by block division is also applicable to Def. 2. The underlined

portion of the Def. 2 can be divided into small segments, each of which consists of s bits. Thus a third definition is derived.

Definition 3:

$$X_{j+1} = \sum_{i=1}^{\ell_j - 1} x_i^{(j)} \cdot 2^{b\cdot(i-1)} + \left\{ \sum_{m=1}^{\lceil b/s \rceil} \underline{\left(\xi_m^{(j)} \cdot 2^{s\cdot(m-1)} \bmod N\right)} \right\} \cdot 2^{t_j} \qquad \text{Eq. (7)}$$

This method can be called a recursive parallel table lookup method, which includes two system description parameters b and s. These parameters can be determined from the trade-off between execution time and memory reduction.

4. NUMBER OF ITERATIONS

It is important to evaluate the number of iterations required in reducing the initial value X_0 to a number less than 2^u. In order to evaluate the most critical case, let us consider the model depicted in Fig. 1. S is the number to be processed which is divided into two portions A and Z. A, the higher block, is greater than or equal to 2^u. Z, the lower block, is less than 2^u. If A is greater than 1, another table look up will result in the next value S0=Z0 + R0 which is a u+1 bits number at most. In other words, A1, the higher block of S0, equals 0 or 1. In the case of 0, no further reduction can be achieved by table look up. If A1 equals 1, the next residue from the table is almost always $R1=2^u - N$ except when N is 2^{u-1}. As a result, the k-th summation Sk is represented as

$$Sk = 2^u + (Z0 - k*N). \qquad \text{Eq. (8)}$$

At the moment Sk becomes less than u bits in length, the procedure stops. Considering the range of Z0 and N, K is 2 at most.

According to the above discussion, we can get the upper bound of the iteration by the procedure listed in Fig. 2. The input for this program is b and s, and the output is SS. Assuming that the computation resources available are one 512 bit adder, 1024 bit shift register and residue tables, SS also means the upper bound for the number of additions.

```
PROCEDURE(
   read b,s;
   B  <- b;
    0
   SS <- 0;
   j <- 0;
   WHILE(B  > 1 ){
          J
        T  <- ⌈B /s⌉ ;
         j        j
        B     <- ⌊log T    ⌋ + 1;
         j+1       ⌊   2  j⌋
        j <- j + 1;
   }
   SS <- SS + 2;
   SS <- SS*(u/s);
   write SS;
}
```

Fig.2 ITERATION EVALUATION

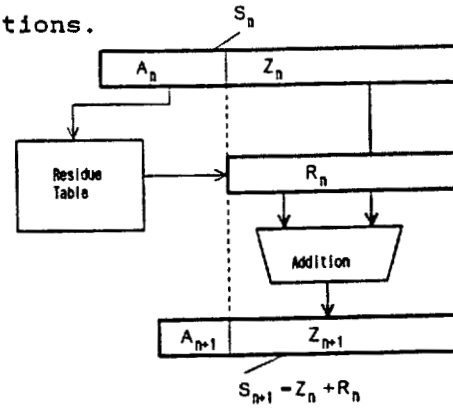

Fig.1 SIMPLE TABLE LOOKUP MODEL

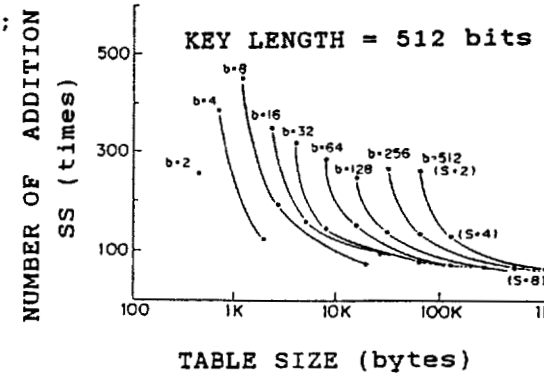

Fig.3 ADDITION VS. MEMORY

5. DISCUSSION

Let A*B and N be 1024 and 512 bits in length, respectively. The total memory capacity Mt is evaluated as follows:

$$Mt(b,s) = 2^{S} * (b/s) * u \qquad \text{Eq.(9)}$$

Reduction of both the number of additions and the memory size can be achieved by choosing appropriate parameters(see Fig.

3). For example, the parameter set (b,s)=(4,4) can reduce the memory size by a factor of 1/64 compared with the (512,4) set, which corresponds to the former method, Eq. (4), in spite of the fact that the two cases require about the same processing time.

6.CONCLUSION

This paper proposes a fast modular arithmetic which can reduce the table size. It also implies the reduction of pre-computation time.

[References]

[1]R.L.Rivest, A. Shamir, L. Adleman:"A method of obtaining digital signatures and public key cryptosystem",Comm. of ACM, pp.120-126(Feb.1978).

[2]M.Rabin:"Digitalized signatures and public-key cryptosystems", MIT/LCS/TR-212,Technical Report MIT (1979)

[3]N.Torii, M. Azuma, R. Akiyama:"A study on RSA parallel processing method"(in Japanese),Proc. of Workshop on cryptography and information security, pp.15-17(Aug.1986).

[4]Y.Nagai,T.Takaragi,F.Nakagawa,R.Sasaki:"Development of trial production for electronic contract authentication system"(in Japanese),Proc. of Workshop on cryptography and information security, pp.109-121(Jul.1987).

[5]Y.Kano,N.Matsuzaki,M.Tatebayashi:"A modulo exponentiation LSI using high-order modified Booth's algorithm"(in Japanese),Proc. of workshop on cryptography and information security, pp.133-142.