Available online at www.sciencedirect.com

ScienceDirect

Applied Mathematics and Computation 185 (2007) 31–44

APPLIED MATHEMATICS AND COMPUTATION

www.elsevier.com/locate/amc

An efficient Montgomery exponentiation algorithm by using signed-digit-recoding and folding techniques

Der-Chyuan Lou [a], Jui-Chang Lai [a], Chia-Long Wu [b,*], Te-Jen Chang [a]

[a] *Department of Electrical Engineering, Chung Cheng Institute of Technology, National Defense University, Tahsi, Taoyuan 33509, Taiwan*

[b] *Department of Aviation and Communication Electronics, Chinese Air Force Institute of Technology, Kaohsiung 82047, Taiwan*

Abstract

The motivation for designing fast modular exponentiation algorithms comes from their applications in computer science. In this paper, a new CSD-EF Montgomery binary exponentiation algorithm is proposed. It is based on the Montgomery algorithm using the canonical-signed-digit (CSD) technique and the exponent-folding (EF) binary exponentiation technique. By using the exponent-folding technique of computing the common parts in the folded substrings, the same common part in the folding substrings can be simply computed once. We can thus improve the efficiency of the binary exponentiation algorithm by decreasing the number of modular multiplications. Moreover, the "signed-digit representation" has less occurrence probability of the nonzero digit than binary number representation. Taking this advantage, we can further effectively decrease the amount of modular multiplications and we can therefore decrease the computational complexity of modular exponentiation. As compared with the Ha–Moon's algorithm 1.261718m multiplications and the Lou-Chang's algorithm 1.375m multiplications, the proposed CSD-EF Montgomery algorithm on average only takes 0.5m multiplications to evaluate modular exponentiation, where m is the bit-length of the exponent.

Keywords: Montgomery algorithm; Modular exponentiation; Exponent-folding technique; Algorithm analysis; Canonical-signed-digit recoding

1. Introduction

Many public-key algorithms [1–3] require the implementation of modular multiplication for operands of 1024 bits or more in length. Taking the RSA cryptosystem [1] for example, the public and private keys are functions of a pair of large prime numbers. The encryption and decryption operations are accomplished by modular exponentiation and can be described as follows. Given M (plain text), E (public key), D (private

* Corresponding author.
E-mail addresses: dclou@ccit.edu.tw (D.-C. Lou), chialongwu@seed.net.tw (C.-L. Wu).

0096-3003/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2006.06.089

key), and N (modulus), compute ciphertext $C \equiv M^E \bmod N$ for encryption and $M \equiv C^D \bmod N$ for decryption. These operations are realized by multiple modular multiplications based on the value of the exponents E and D, where $D \times E \bmod \phi(N) = 1$ and $\phi(N)$ is an Euler's totient function [4].

As efficient computation of the modular exponentiations is very important and useful for many cryptosystems, we need fast multiplication designs or novel exponentiation algorithms such as the Montgomery reduction method [5], high-radix method [6], addition chains method [7], square-and-multiply (binary) method [8], exponent-folding (EF) method [9,10], residue number conversion method [11], key size partitioning method [12], and signed-digit-recoding method [13]. Moreover, a detailed survey of fast exponentiation techniques has been given in [14].

The rest of the paper is organized as follows. In Section 2, we first review and introduce some famous works of the modular exponentiation. Then, we introduce the concept of canonical-signed-digit (CSD) arithmetic and Montgomery algorithm and propose the CSD-EF Montgomery algorithm for fast modular exponentiation in Section 3. The computational complexity of the proposed algorithm is detailed analyzed in Section 4. Finally, we conclude our work in Section 5.

2. The modular exponentiation

Modular exponentiation and modular multiplication of large integers with large exponent and modulus (usually longer than 1024 bits) is one of the most important operations in several well-known cryptographic algorithms. The modular exponentiation can be implemented using a series of modular squaring and modular multiplication operations. Therefore, modular exponentiation can be time-consuming, and is often the dominant part of modern cryptographic algorithms for key exchange, electronic signature, and authentication.

There are two ways to reduce the execution time of the modular exponentiation operation. One approach is simply to reduce the numbers of modular exponentiation. The other approach is to reduce the execution time of each modular multiplication. In this paper, we concentrate on the first approach to effectively reduce the number of modular multiplications required in modular exponentiation operation.

2.1. The binary modular exponentiation algorithms

The binary modular exponentiation method is also known as the "repeated square-and-multiply" method [8]. There are two commonly used algorithms (with different exponent-scanning patterns) that can convert the modular exponentiation into a sequence of modular multiplications, that is, the LSB (least significant bit) binary algorithm and the MSB (most significant bit) binary algorithm [5]. Assume m denotes the bit-length of the exponent E, the exponent E can be expressed in binary representation, i.e., $E = \sum_{i=0}^{m-1} e_i \times 2^i$, where $e_i \in \{0,1\}$. The LSB binary algorithm computes the exponentiation starting from the least significant bit of the exponent E and proceeding to the left, which is depicted as follows.

LSB binary modular exponentiation algorithm

```
Input: Message: M, Modulus: N, Exponent: E is an m-bit integer
Output: Ciphertext: C ≡ M^E (mod N)
C = 1; S = M;
begin
  for i = 0 to m − 1 do                      /*scan from right to left*/
   begin
     if (e_i = 1) then C ≡ C × S (mod N);   /*multiply*/
     S ≡ S × S (mod N);                     /*square*/
   end;
end.
```

Different from the LSB binary modular exponentiation algorithm, the MSB binary modular exponentiation algorithm computes exponentiation starting from the most significant bit of the exponent and proceeding to the right, which is depicted as follows.