Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 192 (2007) 399–404
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

Fast simultaneous scalar multiplication

P. Balasubramaniam [a], E. Karthikeyan [b,*,1]

[a] Department of Mathematics, Gandhigram Rural University, Gandhigram, Dindigul 624 302, Tamil Nadu, India
[b] Department of Computer Science and Applications, Gandhigram Rural University, Gandhigram, Dindigul 624 302, Tamil Nadu, India

Abstract

Scalar multiplication is a very expensive operation in elliptic curve based cryptographic protocol. In this paper, we propose a modified simultaneous elliptic curve scalar multiplication algorithm using complementary recoding. The signed binary representation of an integer is obtained by simple arithmetic operations such as complement and bitwise subtraction. This is an efficient method to obtain the signed binary representation of scalar when compared to other standard methods such as NAF, MOF and JSF.

Keywords: Elliptic curve cryptography; Scalar multiplication; Non-adjacent form; Joint-sparse form; Complementary recoding

1. Introduction

In the year 1895, Miller [1] and Koblitz [2] proposed an efficient public key cryptosystem called elliptic curve cryptography (ECC) that relies on the difficulty of elliptic curve based discrete log problem (ECDLP). ECC has become a vital technology for cryptography because of its high security with shorter key-length and faster computation than other existing cryptographic schemes. For example, 160-bits of ECC and 1024-bits of RSA [3], DSA [4] offer the same level of security. These advantages are particularly beneficial in applications where bandwidth, processing capacity, power availability and storage is limited (see [5]).

The dominant computation of all elliptic curve based cryptographic algorithms for encryption/decryption as well as the signature generation/verification, is the scalar multiplication. It is of the form $k * P$ for a point P and an integer k. The speed of scalar multiplication plays a vital role in deciding the efficiency of the whole system. This computation is done by representing the integer k in binary form as $k = \sum_{j=0}^{l-1} k_j 2^j$, where $k_j \varepsilon \{0,1\}$ and repeated point addition and doubling operations. Scalar multiplication using binary representation is called binary method [7]. The efficiency of this method is determined by the hamming weight of the scalar representation, that is the number of 1's in this representation. There has been extensive research to compute scalar multiplication efficiently [6].

* Corresponding author.
E-mail addresses: pbalgri@rediffmail.com (P. Balasubramaniam), e_karthi@yahoo.com (E. Karthikeyan).
[1] The author is also working in Karpagam Arts, and Science College, Coimbatore, India.

0096-3003/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2007.03.017

In this paper we discuss the complementary recoding model, an efficient method to obtain signed binary representation of a scalar. It is obtained by the simple computation $k = \sum_{i=0}^{l-1} k_i 2^i = (100\ldots0)_{(l+1)}\text{bits} - \bar{k} - 1$. Here we require few basic arithmetic operations such as complement and bitwise subtraction, which does not require more computational resources when compared to other standard methods.

The paper is organized as follows. In Section 2, the first alternative scalar representation – *booth recoding* – is discussed. The Section 3 is devoted to the computation of simultaneous scalar multiplication using Shamir method and it is illustrated with example and followed by another method to obtain the signed binary representation called *joint sparse form* which is explained in Section 4. Finally our proposed method to obtain the signed binary string is explained with necessary examples and has given simulated result in the Section 5.

2. Signed binary representation

In 1951, Booth [8] proposed a new scalar representation called *signed binary representation*and later Rietweisner [9] proved that every integer could be uniquely represented in this form. The property of this representation is that, of any two consecutive digits, at most one is non-zero. Here the integer k is represented as $k = \sum_{j=0}^{l-1} k_j 2^j$, where $k_j \in \{-1, 0, 1\}$. Rietweisner's canonical representation is called as *non-adjacent form* (NAF) [9]. Fortunately, NAF (k) is at most one digit longer than the binary representation of k. The signed binary representation of scalar is obtained by the following algorithm.

Algorithm 1. Computation of NAF (k)

```
Input: k (Positive integer)
Output: s (NAF representation of k)
  c = k; l = 0
  While (c > 0)
    If (c is odd)
      s[l] = 2 − (c mod 4)
      c = c − s[l]
    Else
      s[l] = 0
    EndIf
    c = c/2; 1 = l + 1
  End While
Return s
```

The average hamming weight of signed binary representation is $n/3$ which has lower hamming weight than binary representation of the same. For example, the binary representation of 2927 is $(101101101111)_2$, the hamming weight is 9 and NAF of 2927 is $(0110\bar{1}00\bar{1}000\bar{1})_2$, the hamming weight is just 5. This reduction saves 4 ECADD operations and it leads to an improvement in the scalar multiplication. Signed binary representation of scalar is meaningful in elliptic curve scalar multiplication because inverse of a point can be computed virtually free.

3. Shamir method

Some public key cryptographic protocols such as the verification of digital signature, self-certified signature scheme requires the computation of powers of two, three, or more. ECDSA verification requires the computation of $aP + bQ$, where a and b are integers and P and Q are elliptic curve points. Brumley proposed [10] a modified self-certified signature scheme which requires the computation of three points.

Shamir proposed a method [11] to compute $aP + bQ$ simultaneously by writing signed binary representation of pair of integers one by one. The number of non-zero columns is defined as the *joint hamming weight* which determines the speed of the computation. For example, the joint weight of 57 and 22 in binary expansion is 6, since $57 = (111001)_2$ and $22 = (010110)_2$. The signed binary representation of the same is $(100\bar{1}001)_2$