2004

Efficient signature system using optimized elliptic curve cryptosystem over GF(2(n)).

Xiaoguang Wang
University of Windsor

Efficient Signature System using Optimized Elliptic Curve Cryptosystem over $GF(2^n)$

By

Xiaoguang Wang

A Thesis
Submitted to the Faculty of Graduate Studies and Research
Through School of Computer Science
In partial fulfillment of the requirements for
The Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2004

Your file Votre référence
ISBN: 0-612-92455-6
Our file Notre référence
ISBN: 0-612-92455-6

Abstract

Elliptic curve cryptography was proposed independently by Neil Koblitz and Victor Miller in the middle of 80's. The security of Elliptic Curve Cryptography depends upon the elliptic curve discrete logarithm problem. For providing the same strength, it uses a smaller key size than that for RSA. This advantage makes it particularly suitable for some devices and applications, which have a resource constraint.

Digital Signature Systems are one of the most important applications of cryptography. In Y2K IEEE has included two Elliptic Cryptography based methods in its new standard P1363. The elliptic curve cryptosystem uses "point" operations like point doubling and addition. As a consequence, optimization of point operations plays a key role in determining the efficiency of computation. Today's technology easily permits the fabrication of multiple simple "processors" on a single chip. For such devices, a serial-parallel computation has been proposed by Adnan and Mohammad [AM03][AM03a] for a faster computation of elliptic algorithms. This thesis presents a new optimized point operations algorithm for elliptic curve cryptosystems over $GF(2^n)$. We have designed and implemented the new algorithm for a more efficient digital signature system.

Keyword: Elliptic Curve Cryptosystem, Point Addition, Point Multiplication, Point Doubling, Projective Coordinate, Jacobian Coordinate, Chudnovsky-Jacobian Coordinate, DSA, ECDSA, ECDLP

Acknowledgements

The work presented in this thesis would not be possible without the help of many people. I would first like to acknowledge the support of my thesis supervisor and Master degree supervisor, Professor Dr.Akshai Aggarwal for his invaluable advice and ideas on the research. His support and expertise led me in the right direction, whenever we were faced with hard problem in the past two years.

I thank Dr. Huapeng Wu for his advice which enriched my knowledge in elliptic curve cryptography.

I would also like to thank my colleagues in our Research Group . In particular, I would like to show my appreciation to Mr. Lu Xin and Mr. Marmagna Desai for their kind and warm-hearted help. I thank the members of our Research Group for providing a great atmosphere to work

Finally, I would like to give special thanks to my family. Specially, I thank my wife, Mrs. Yuhong Meng for her support, encouragement and understanding .

To all of you thank you very much.

Table of Contents

List of Figures

List of Tables

1 INTRODUCTION

Elliptic Curve Cryptosystem (ECC) has begun to be paid more attention, because the EC Discrete Logarithm problem seems hard to be cracked. So a much smaller key size with an equal security in encryption can be used in ECC [Wi99]. Point addition and doubling of ECC require inversion operation. Inversion is the most expensive operation over $GF(2^n)$. To eliminate inversion operation, several coordinate systems had been considered in elliptic curve point operation. The coordinate system is chosen such that point addition and doubling can be implemented with the smallest number of field multiplications. To make the point operations faster, Adnan and Mohammad proposed a serial-parallel computation architecture (**J-P-ECC**) for point addition and doubling in ECC [AM03]. After exploiting the inherent parallel mechanism at both algorithmic level and the arithmetic level of ECC using Jacobian coordinate system, it uses serial-parallel computation in point addition and doubling. This requires computation of 3 field multiplications with 3 digital serial multipliers in parallel instead of sequential computation with only one digital serial multiplier.

In my thesis, we propose an optimal serial-parallel computation architecture of ECC (**CC-P-ECC**) over $\mathbf{GF}(2^n)$. It chooses the coordinate system that has the least multiplication instruction cycles with 3 digital serial multipliers.

Computation efficiency of elliptic curve cryptosystem is a major research field in that it is an important factor in implementing ECC in some devices with constrained environment. Computation efficiency involves scalar multiplication optimization, mixed coordinate systems and serial - parallel computation in point operation.

1.1 Elliptic Curve Cryptography: Market Application

The strength of elliptic curve (EC) cryptosystem is based on EC Discrete Logarithm problem. [AMR+02]. " The brute force method to solve the EC Discrete Logarithm

problem is (computationally) infeasible." [Wi99]. In other words, this problem is so hard to crack that its key reduction in size is highly considered when compared to the key used by other cryptosystems. The typical example is that it is able to challenge RSA, one of the most popular public key cryptosystems. Although some critics are still skeptical about the reliability of this method, recently some companies have developed several encryption techniques, using properties of elliptic curve. A good example is as follows: "February 16, 2004 – Certicom Corp. (TSX: CIC), a leading provider of wireless security solutions, has licensed its Elliptic Curve Cryptography (ECC) to Neopost, the leading European and number two worldwide supplier of mailing solutions. Neopost is using the Security Builder® Crypto™ toolkit to embed ECC-based security into its mailing systems to create a secure, cost-effective way to generate Digital Postage Marks (DPMs) that meet the stringent requirements of the North American postal officials. In addition to being the only technology providing digital signatures that meet the small footprint requirements for DPMs, ECC enables Neopost's systems to perform smaller signatures at a faster speed than competing systems" [Certi04].

1.2 The methods of performance Improvement for ECC

As we know performance of elliptic curve cryptosystem is determined by computation efficiency on point operations including point multiplication, point addition and doubling. So the techniques for performance improvement of ECC focus on optimization on Point Multiplication or Point addition/doubling in Serial and Serial-Parallel computation architecture, respectively.

1.2.1 Methods of optimization on Point Multiplication

"Point Multiplication is a special case of the general problem of exponentiation in abelian groups and it is related to the *shortest addition chain* problem for integers." [BSS99]

Definition: The *shortest addition chain problem for integers is defined as follows:*
"Let k be a positive integer (the input), Start from the integer 1, and computing at each step the sum of two previous results, what is the least number of steps required to reach k?" [BSS99].

Due to point multiplication's central role in public key cryptography, efficient algorithms for *short addition chain* have continued to receive much attention.

In this thesis, we will focus on the case of ECC over finite field of characteristic two. Some general methods can be used to compute Point Multiplication. Typically, there are several efficient algorithms stated by Blake and Serroussi [BSS99] as follows:

- Binary method
- m-ary method
- Sliding window method
- Signed m-ary Window method

Chapter 4 will describe the above method.

1.2.2 Methods of Optimization on point operation using mixed coordinate systems

The computation of point addition and doubling in ECC includes field division arithmetic operation. The ratio of time taken for Inversion and Multiplication is the order of 3~10 [BBS99], If their ratio is over 1:10, eliminating inversion operation is essentially required in ECC. So the task is to find out a coordinate system representing point that uses the least quantity of field multiplications in point addition and doubling for implementing an inversion operation.

In general, the following coordinate systems are used in elliptic curve cryptosystem as follows [CMO98]:

- Affine coordinate

  P(x, y) represent point over Elliptic curve

- Projective coordinate

  $P(x, y) \rightarrow P(X, Y, Z)$

  $x = X/Z;\ y = Y/Z;$

- Jacobian coordinate

  $P(x, y) \rightarrow P(X, Y, Z)$

  $x = X/Z^2,\quad y = Y/Z^3;$

- Modified Jacobian coordinate

  $P(x, y) \rightarrow P(X, Y, Z, aZ^4)$

  $x = X/Z^2,\quad y = Y/Z^3$

- Chudnovsky Jacobian coordinate

  $P(x, y) \rightarrow P(X, Y, Z, Z^2, Z^3)$

  $x = X/Z^2,\ y = Y/Z^3;$

Chapter 2 will give a further explanation about the coordinate systems used in elliptic curve cryptosystems.

1.2.3 Optimization on Serial-Parallel computation of ECC

As discussed in Section1.2.2, we know that field inversion operation can be eliminated by mixing various coordinate systems. If ECC operations are put in a sequence on the basis of the time consumed for executing it on a computing node, field inversion will be the first and field multiplication operation will be the second costliest operation. With exploiting inherent parallel mechanism in point addition and doubling, there exist a way

to speedup field multiplications with serial-parallel architecture as described by Adnan and Mohammad [AM03].

1.3 Thesis Statement

Although Serial-Parallel computation of ECC proposed by Adnan and Mohammad [AM03], compared with serial ECC architecture on point operation, has an improvement on performance, we found it is possible to achieve furthermore optimization.

After carefully analyzing dependency relation inside point operation, this thesis gives a solution with optimal normal base over $GF(2^n)$ on serial-parallel ECC architecture. Not only theoretically is it proved to improve on performance, experimental results show that it obtains considerable improvement over the system proposed by Adnan and Mohammad [AM03] on serial-parallel ECC architecture over $GF(2^n)$.

1.4 Structure of thesis document

The thesis is organized as follows: Chapter 2 discusses Literature Review and Chapter 3 will give related works. Chapter 4 explains new proposed algorithms. Chapter 5 analyzes the experimented data and Chapter 6 summarizes the thesis and points out future work.

2 REVIEW OF THE LITERATURE

In this Chapter, an introduction to elliptic curve cryptography is given. The architecture designed for elliptic curve cryptosystem with a digital serial multiplier is given.

2.1 Cryptographic Systems

In this Section, a brief review of RSA and the comparison between RSA and Elliptic Curve Cryptosystem is described.

2.1.1 Elliptic Curve Discrete Logarithm Problem(ECDLP)

ECDLP is the discrete logarithm problem applied to elliptic curves over a finite field , which is defined as follows.

Given Q and Y, find x for which

$$Q = [x]Y$$

where $x \in \{1,...,\#Y-1\}$ and Q, Y are points on elliptic curve $E(K)$, K is a finite field.

So far , there is no known sub-exponential time algorithms to compute x given Q and Y [BSS99]. Although the index-calculus method is a sub-exponential time algorithm for solving the discrete logarithm problem, it is not applicable to multiplicative groups in a finite field [BSS99] such as the elliptic curve group. The most efficient algorithm known is the Pollard- ρ method [Pol75]. It is parallelized and the expected running is $\sqrt{\pi n}/(2r)$ with r processors [OW99]. However, the running time is still exponential in n. Therefore the methods for computing ECDLP are much less efficient than those for factoring or DLP . As a result, ECC provides shorter key sizes than others public key cryptosystems with the same security level.

2.1.2 Comparison between RSA and ECC based Public key system

RSA is the most widely used public key cryptosystem. It was first proposed by Ron Rivest, Adi Shamir and Leonard Adleman in 1977 [RSA78]. The algorithm mainly depends on the difficulty of working out the factorization of a large integer n, where n is a product of two prime numbers p and q, of about the same size. If p and q are known, it is relatively easy to calculate $n = p*q$. However it is intractable to compute p and q if n is known. (This problem is called the problem of factorization of *n, where n is a very large integer.*)

Elliptic curve cryptography can be also used for public key cryptosystem . It offers secure communication mechanism. The difference is that the strength of RSA is based on the integer factorization problem while that of ECC is based on elliptic curve discrete logarithm problem (ECDLP) [Odl84]. ECC has one obvious advantage over RSA in that ECC always has a smaller key size than RSA with equivalent strength of security. Strength of security is in terms of the time to break the cryptosystem. In other words, ECC provides a more secure cryptosystem than RSA for the same key length. [Wi99]

2.2 Elliptic Curve Cryptography

Elliptic curves were proposed for cryptographic purposes by Koblitz [Ko87] and Miller [Mi86] in 1985. The discrete logarithm problem over the group of points on an elliptic curve over finite field is a one-way function because there is no sub-exponential attack known for solving this problem[Wi99]. This makes elliptic curve more attractive than other public key cryptosystem. The section gives an introduction to elliptic curves cryptography. It is followed by a discussion of the curve addition and curve doubling operations on elliptic curve over real numbers. Afterwards, elliptic curve over finite fields as well as their operations are described.

2.2.1 Elliptic Curve over R^2

This section introduces elliptic curve over real numbers. Elliptic curves $\boldsymbol{E}$ over the real numbers ($\boldsymbol{R}^2$) are sets of points in the form of (x,y), $x, y, a_4, a_6 \in R$ that satisfy the equation

$$y^2 = x^3 + a_4 x + a_6 \qquad (2.3.1)$$

together with a special point $\boldsymbol{O}$, called the point at infinity which is an identity element. The variables x and y represent a point on elliptic curve and cover a two dimensional (affine) coordinate plane $\boldsymbol{R}$ x $\boldsymbol{R}$. Elliptic curve $\boldsymbol{E}$ over $\boldsymbol{R}^2$ is said to be defined over $\boldsymbol{R}$, denoted by $\boldsymbol{E(R)}$. Elliptic curve over real numbers can be used to form a group $(\boldsymbol{E}(\boldsymbol{R}),+)$ consisting of the set of points $(x, y) \in R \times R$ together with an addition operation + on $\boldsymbol{E(R)}$

Figure 2.1.1 shows a plot of an elliptic curve over $\boldsymbol{R}$.

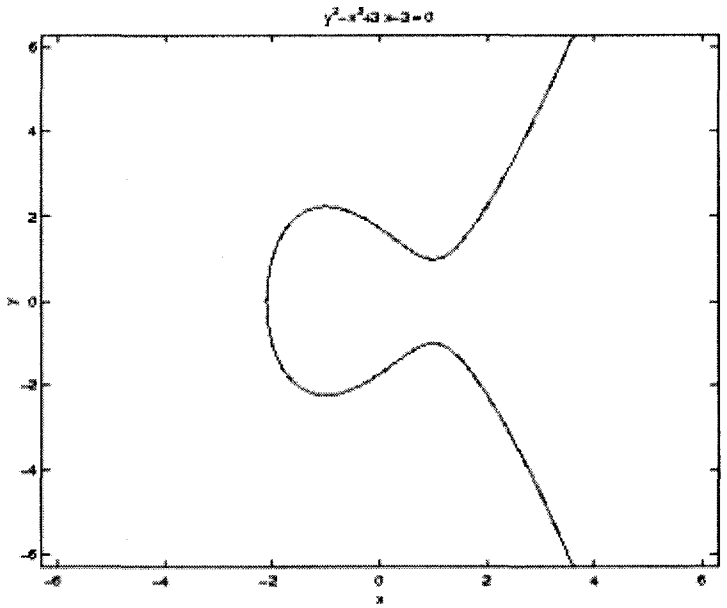


Figure 2.2.1 Plot of Elliptic Curve

Curve Addition and Doubling

The point addition (ESUM, also known as curve addition) operation + is defined on the set $\boldsymbol{E(R)}$ of points (x,y). By the rule of identity, the point at infinity $\boldsymbol{O}$ is the point that

added to any point on the elliptic curve, gives the same point. Therefore for all $P = (x, y) \in E(R)$,

$$P + \boldsymbol{O} = \boldsymbol{O} + P = P$$

For each point $P(x, y) \in E(R)$, the square root of Equation 2.3.1 gives

$$\pm y = \sqrt{x^3 + a_4 x + a_6}$$

So two y-coordinate values are given by each unique value of x. The point $(x,-y)$, denoted $-P \in E(R)$, is called the negative of point P and specified as

$$P + (-P) = (x, y) + (x, -y) = O \qquad (2.3.2)$$

Addition on $\boldsymbol{E(R)}$ is defined geometrically. Suppose there are two distinct points P and Q, $P, Q \in E(R)$. The law of addition in the elliptic curve group is $P+Q=R$, $R \in E(R)$. The geometric relationship is shown in Figure 2.2.2.

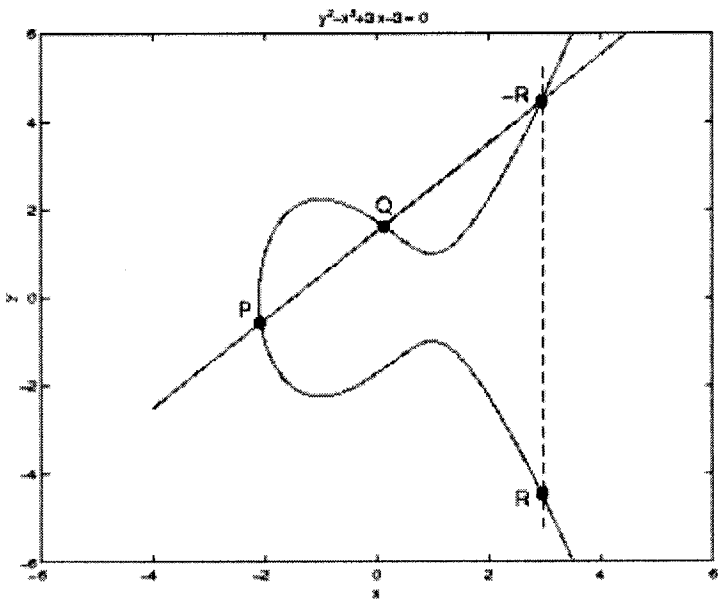


Figure 2.2.2 Addition of EC points

In order to find the point R, first connect the points P and Q by a line L. By simultaneously solving equations L and E, an equation of degree three is derived with exactly three solutions. Therefore the line L is guaranteed to intersect the curve E on a third point, say $-R \in E(R)$. The point R can be obtained by negating the y-coordinate of $-R$.

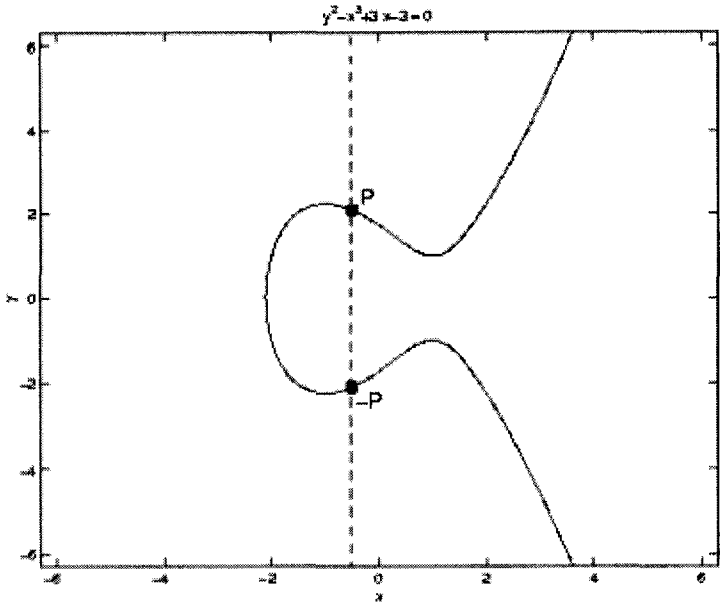


Figure 2.2.3 Addition of points P and -P EC

In the case that the two points are P and $-P$ **Figure 2.2.3**, the line connecting P and $-P$ intersects the elliptic curve at a third point which is the special point O lying on every vertical line in the coordinates plane.

In an operation of point addition, if points $P, Q \in E(R)$ are added where $P=Q$, then the tangent line to the elliptic curve at point P is taken instead (shown in Figure 2.2.4). For this case, it is a point doubling (EDBL) operation where $R=2P$.

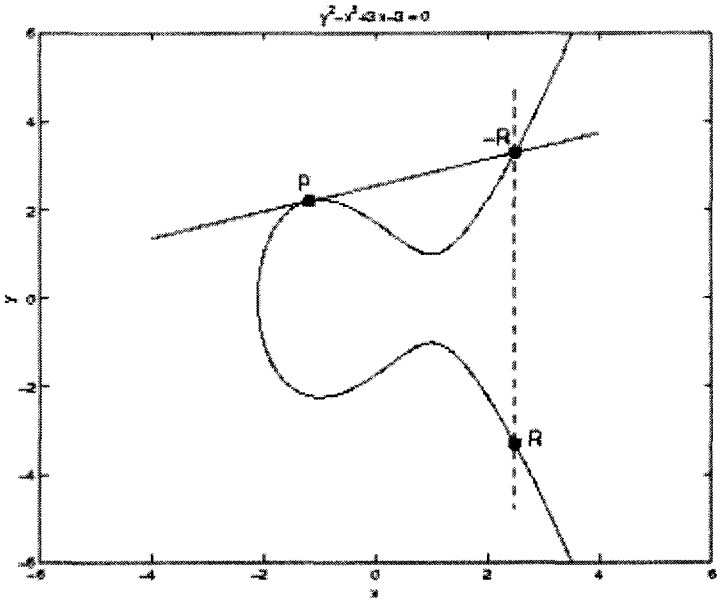


Figure 2.2.4 Doubling of EC Point

The following formulae express the definition of point addition and doubling mathematically

$$P = (x_1, y_1)$$
$$Q = (x_2, y_2)$$
$$R = P + Q = (x_3, y_3)$$

where $P, Q, R \in E(R)$, and

$$x_3 = \theta^2 - x_1 - x_2$$
$$y_3 = \theta(x_1 + x_3) - y_1 \qquad \text{if} \qquad P \neq Q$$
$$\theta = \frac{y_2 - y_1}{x_2 - x_1}$$

or

$$\theta = \frac{3x_1^2 + a_4}{2y_1} \qquad \text{if} \qquad P = Q$$

2.2.2 Elliptic Curve over finite fields

This section introduces elliptic curve over finite fields, the prime fields (F_p) and the binary finite fields (F_{2^n}).

2.2.2.1 Elliptic Curve over F_p with $p>3$

A prime field F_p is generated by using a large prime p [LN94]. The operations of elliptic curve over F_p is similar to $E(R)$. Instead of calculations on real numbers, the calculations modulo a large prime are taken. Therefore an elliptic curve E is defined over F_p, denoted by $E(F_p)$, if $x, y, a_4, a_6 \in F_p$ and $4a_4^3 + 27a_6^2 \neq 0$ satisfying the equation

$$y^2 = x^3 + a_4 x + a_6$$

Points on this curve form a group. Therefore the elliptic curve group $(E(F_p),+)$ is set of points (x,y) (for $x, y \in F_p$) and an operation + (addition) which satisfies the axioms in Section 2.2.2.

The order of a point A on $E(F_p)$ is the smallest positive integer r such that

$$\underbrace{A + A + ... + A}_{r} = O$$

The order of the curve is the number of points of $E(F_p)$, denoted by $\#E(F_p)$. By Hasse's theorem [Kob87a] [Men93], $\#E(F_p) = p + 1 - t$, where $|t| \leq 2\sqrt{p}$.

2.2.2.2 Elliptic Curve over F_{2^n}

A non-supersingular [AMS95] elliptic curves E defined over a finite field F_{2^n} (characteristic = 2), denoted by $E(F_{2^n})$, is the set of solutions, (x,y) for $x, y \in F_{2^n}$, to the simplified forms of the *Weierstrass equation*

$$y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6 \qquad (2.3.3)$$

where $a_1, a_2, a_3, a_4, a_6 \in F_{2^n}$, namely

$$y^2 + xy = x^3 + a_2 x^2 + a_6 \qquad (2.3.4)$$

where $a_6 \neq 0$ and $x, y, a_2, a_6 \in F_{2^n}$. Again, an identity element (point of infinity)$\boldsymbol{O}$, is included in both curve. Elliptic curve E over F_{2^n} also forms a group $(E(F_{2^n}),+)$ that satisfies the axioms in Section 2.2.2.

The curves of Equation 2.3.4 are called non-supersingular curves and are suitable for cryptographic applications [BSS99]. From a hardware implementation perspective, ECs over F_{2^n} are thought to be very practical. The advantages of using ECC are

- Owing to ECC offering the highest security per bit of any known public key cryptosystem, therefore a smaller memory can be used
- ECC hardware implementations use less transistors, as an example, a VLSI implementation of a 155-bit ECC processor has been reported which uses only 11,000 transistors [AMV93], compared with an equivalent strength 512-bit RSA processor which used 50,000 transistors [PID92].

2.2.2.3 Point Operations of Elliptic Curves over F_{2^n}

A non-supersingular [AMS95] elliptic curve E over F_{2^n}, $E(F_{2^n})$ was selected for the implementation of elliptic curve cryptosystem. $E(F_{2^n})$ is the set of all solutions to the Equation 2.3.4 with coordinates in the algebraic closure of $\boldsymbol{E}$ [Men93], where $a_2, a_6 \in F_{2^n}$ and $a_6 \neq 0$. Such an elliptic curve is a finite Abelian group [Men93]. The number of points in this group is denoted by $\#E(F_{2^n})$. The ECC can be implemented by affine coordinates or by other coordinate system. For different coordinates systems, the computation of the curve operations are also different. In Chapter 3, other coordinate systems are specified.

Elliptic Curve Operations in Affine Coordinates

In this section, the elliptic curve cryptosystems that will be used are based on the discrete logarithm problem over $E(F_{2^n})$ and the basic computation which must be made is curve multiplication (Point multiplicaiton). Curve multiplication is expressed as a sequence of point additions and point doublings. Similar to EC over real numbers, point addition and point doubling are defined geometrically. It is hard to represent an elliptic curve over a finite field graphically, however, the method of finding the point of addition and doubling are the same as shown at Section 2.3.1.

Assume a non-supersingular elliptic curve E over F_{2^n} given in affine coordinates and P, Q are two points on $E(F_{2^n})$. Let $P=(x_1,y_1)$, $Q=(x_2,y_2)$, then negative of P is $-P=(x_1,y_1+x_1)\in E(F_{2_n})$. From [SOOS95], if $Q\neq -P$, then $P+Q=R=(x_3,y_3)\in E(F_{2^n})$.

If $P\neq Q$

$$\left.\begin{aligned}\theta &= \frac{y_1+y_2}{x_1+x_2}\\ x_3 &= \theta^2+\theta+x_1+x_2+a_2\\ y_3 &= (x_1+x_3)\theta+x_3+y_1\end{aligned}\right\} \qquad (2.3.5)$$

Otherwise if $P=Q$

$$\left.\begin{aligned}\theta &= \frac{y_1}{x_1}+x_1\\ x_3 &= \theta^2+\theta+a_2\\ y_3 &= x_1^2+(\theta+1)x_3\end{aligned}\right\} \qquad (2.3.6)$$

Iin affine coordinates, point addition (ESUM) and point doubling (EDBL) require three and two multiplications, respectively and one field inversion that is far more expensive than field multiplication.)

2.2.2.4 Curve Multiplication

Multiplication (EMUL) is defined by repeated addition, i.e.

$$Q = cP \tag{2.3.8}$$

$$= \underbrace{P + P + \ldots + P}_{c} \tag{2.3.9}$$

This can be computed by using Binary method algorithm that will be described in Chapter 3.

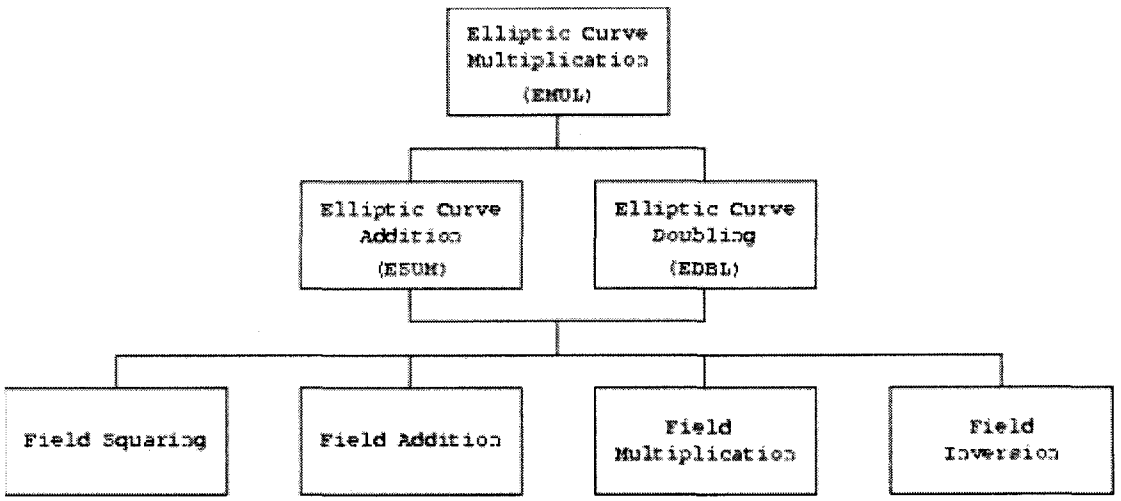


Figure 2.2.5 The hierarchy of elliptic curve operation

The hierarchy of elliptic curve operation is shown in **Figure 2.2.5**. Curve multiplication is computed via point additions (ESUM) and doubling (EDBL) which are in turn computed from field operations.

Summary

In this chapter, the fundemental theory for understanding elliptic curve cryptography is given. The introduction to elliptic curves over real numbers and finite fields was presented. The basic operations, point addition and point doubling of elliptic curve were also described in details. Afterwards, elliptic curve discrete logarithm problem is presented. It specially empasizes that there is no known sub-exponential time algorithm to solve ECDLP.

3 The Related Work

Software Implementations of elliptic curve are presented in this chapter. Point Operations of elliptic curve cryptosystem under different coordinate are introduced. Algorithms on Serial and Serial-Parallel will be given, respectively. Afterwards, cost analysis under different coordinate systems is followed next.

3.1 Introduction

In general, the implementations of efficient elliptic curve cryptosystem are focused on the optimization of the algorithms for point (Curve) multiplication and point doubling/addition. In this Chapter, both aspects will be discussed in details.

3.2 Point Multiplication (Curve Multiplication)

From algebra point of view, Point Multiplication in elliptic curves is a special case of the general problem of exponentiation in abelian groups. Its solutions are techniques available from all the techniques for the general problem and *shortest addition chain* problem for integers. The idea of the shortest addition chain is as follows:

"Let k be a positive integer (the input). Start from the integer 1, and computing at each step the sum of two previous results, what is the least number of steps required to reach k" [BSS99].

To find efficient algorithms for group exponentiation, a lot of efforts have been made by researchers due to point multiplication's central role in public key cryptography.

To obtain faster computation, certain idiosyncrasies of the elliptic curve version of the problem can be considered as follows:

- "Elliptic curve subtraction has virtually the same cost as addition, so the search space is expanded to include addition-subtraction chains and signed representations" [BSS99].

- "In tuning-up algorithm, the relative complexities of general point addition and point doubling have to be considered. This relation depends on the coordinate system used and on the relative complexities of field inversion and multiplication" [BSS99]

- "For certain families of elliptic curves, specific shortcuts are available that can significantly reduce the computational cost of point multiplication" [BSS99].

To make analysis simple on computation, we will only consider the case of finite fields of characteristic two in this thesis. Some typical efficient methods for point multiplication are given in the following sections.

3.2.1 The Binary method

Algorithm of Binary method is described in **Figure 3.2.1**. In the Algorithm 3.2.1, it relies on the binary expansion of k. It requires l -1 point doublings and W-1 point additions (Operations involving infinite point $\boldsymbol{O}$ are not counted), where l is the length and W the weight (number of ones) of the binary expansion of k.[BSS99].

Algorithm 3.2.1 Point Multiplication: Binary method

INPUT: A point P, an l-bit integer $k = \sum_{j=0}^{l-1} k_j 2^j$, $k_j \in \{0,1\}$

OUTPUT: $Q = [k]\,P$.

1. $Q \leftarrow O$.
2. For $j = l$-1 to 0 by -1 do:
3. $Q \leftarrow [2]\,Q$,

4. if $k_j = 1$ then $Q \leftarrow Q + P$.

5. Return Q

Figure 3.2.1 Binary Method Algorithm

3.2.2 The *m*-ary method

In m-ary method, k is represented using m-ary expansion, where $m = 2^r$ for some integer $r \geq 1$. Binary method is a special case in case of $r = 1$. The m-ary method is described in **Figure 3.2.2** as follows:

- **Algorithm 3.2.2: Point Multiplication: *m*-ary method**

  INPUT: A point P, an integer $k = \sum_{j=0}^{d-1} k_j m^j$, $k_j \in \{0, 1, \ldots, m\text{-}1\}$

  OUTPUT: $Q = [k]P$.

  Pre-computation.

  1. $P_1 \leftarrow P$.
  2. For $i = 2$ to m-1 do $P_i \leftarrow P_{i-1} + P$. (We have $P_i = [i]P$)
  3. $Q \leftarrow O$.

  Main loop.

  4. For $j = d$-1 to 0 by -1 do:
  5. $Q \leftarrow [m]Q$. (This requires r doubling)
  6. $Q \leftarrow Q + p_{k_j}$.
  7. Return Q.

Figure 3.2.2 Point Multiplication *m*-ary Method

The algorithm is easily verified, following Honer's rule [Knu81]:

$$[m](\ldots[m]([m]([k_{l-1}]P)+[k_{l-2}]P) + \ldots) + [k_0]P = [k]P$$

From **Algorithm 3.2.2**, the number of doubling in the main loop of the m-ary methods is $(d\text{-}1)r$ (the first iteration is not counted, as it starts with $Q = \boldsymbol{O}$). Since $d = \lceil l/r \rceil$, where l is the length of the binary representation of k, the number of doubling in the m-ary method may be up to r-1 less than the l-1 requires by the binary methods[BSS99].

3.2.3 Modified *m*-ary method

- **Algorithm 3.2.3: Point Multiplication: Modified *m*-ary method**

  INPUT: A point P, an integer $k = \sum_{j=0}^{d-1} k_j m^j$, $k_j \in \{0, 1, \ldots, m\text{-}1\}$

  OUTPUT: $Q = [k]P$.

  Pre-computation

  1. $P_1 \leftarrow P, P_2 \leftarrow [2]P$
  2. For $i = 2$ to $(m\text{-}2)/2$ do $P_{2i+1} \leftarrow P_{2i-1} + P_2$.
  3. $Q \leftarrow O$.

  Main loop

  4. For $j = d$-1 to 0 by -1 do:
  5. If $k_j \neq 0$ then do:
  6. Let s_j, h_j be such that $k_j = 2^{s_j} h_j$, h_j odd.
  7. $Q \leftarrow [2^{r-s_j}]Q$.
  8. $Q \leftarrow Q + p_{h_j}$.
  9. Else $s_j \leftarrow r$
  10. $Q = [2^{s_j}]Q$.
  11. Return Q.

In **Algorithm 3.2.3**, one point doubling and 2^{r-1}-1 point addition in the pre-computation phase are required, and at most n-1 point doublings and d-1 point addition are required in the main loop [BSS99].

3.2.4 Window methods

The m-ary method may be regarded as a special case of the window method, where bits of the multiplier k are processed in blocks of (windows) length r. The algorithm of Window method is as follows.

- **Algorithm 3.2.4 Point Multiplication: Sliding Window Method**

  INPUT: A point P, an l-bit integer $k = \sum_{j=0}^{l-1} k_j 2^j$, $k_j \in \{0,1\}$

  OUTPUT: $Q = [k]\,P$.

  Pre-computation:

  1. $P_1 \leftarrow P$, $P_2 \leftarrow [2]\,P$
  2. For $i = 1$ to 2^{r-1} do $P_{2i+1} \leftarrow P_{2i-1} + P_2$.
  3. $j \leftarrow l\text{-}1$, $Q \leftarrow O$.

  Main loop

  4. While $j \geq 0$ do:
  5. If $k_j = 0$ then $Q \leftarrow [2]\,Q$, $j \leftarrow j\text{-}1$;
  6. Else do:
  7. Let t be the least integer such that
  8. $j - t + 1 \leq r$ and $k_t = 1$,
  9. $h_j \leftarrow (k_j k_{j-1} \ldots k_t)_2$,
  10. $Q \leftarrow [2^{j-t+1}]Q + p_{h_j}$
  11. $j \leftarrow t\text{-}1$.
  12. Return Q.

Figure 3.2.3 Point Multiplication of Window Method

Upon the analysis [BSS99], the benefit using slide window can be given in that there is an effect equivalent to using fixed windows, one bit larger, without increasing the pre-computation cost. The total number of windows processed (and consequently, the number of general point additions in the main loop) behaves like $l(r+1)$ in comparison to l/r for the m-ary method. This is proven in [LH94].

3.2.5 Signed *m*-ary sliding window method

The method that combined *m*-ary and signed methods is described in this section. In this method, a non-redundant signed *m*-ary representation is used, for example, digit set $B = \{-2^{r-1}+1, \ldots, -1, 0, 1, \ldots, 2^{r-1}\}$ with windows of size up to r. The positive multiplier k is decomposed as follows:

$$k = \sum_{i=0}^{d-1} b_i 2^{e_i}, b_i \in B \setminus \{0\}, e_i \in Z \geq 0,$$

where $e_{i+1} - e_i \geq r, 0 \leq i \leq d-2.$

k with the binary representation can be decomposed as **Algorithm 3.2.5a**.

Algorithm 3.2.5a Signed *m*-ary Window Decomposition

INPUT: An integer $k = \sum_{j=0}^{l} k_j 2^j, k_j \in \{0,1\}, k_l = 0$.

OUTPUT: A sequence of pairs $\{(b_i, e_i)\}_{i=0}^{d-1}$

1. $d \leftarrow 0, j \leftarrow 0$.
2. while $j \leq l$ do:
3. If $k_j = 0$ then $j \leftarrow j+1$.
4. Else do:
5. $t \leftarrow \min\{l, j+r-1\}$, $h_d \leftarrow (k_t k_{t-1} \ldots k_j)_2$.
6. If $h^d > 2^{r-1}$ then do:
7. $b_d \leftarrow h_d - 2^r$,
8. increment the number $(k_l k_{l-1} \ldots k_{t+1})_2$ by 1.
9. Else $b_d \leftarrow h_d$.
10. $ed \leftarrow j$, $d \leftarrow d+1, j \leftarrow t+1$.
11. return the sequence (b_0,e_0), (b_1,e_1), …, (b_{d-1}, e_{d-1})

After decomposing the k using Algorithm3.2.5a, the Signed *m*-ary Window method may be implemented as follows:

Algorithm 3.2.5b Signed *m*-ary Window method

INPUT: A point P, and $\{(b_i, e_i)\}_{i=0}^{d-1}$ such that $k = \sum_{i=0}^{d-1} b_i 2^{e_i}$.

OUTPUT: $Q = [k]P$

Pre-computation

1. $P_1 \leftarrow P, P_2 \leftarrow [2]P$.
2. For $i = 1$ to 2^{r-2}-1 do $P_{2i-1} + P_2$.
3. $Q \leftarrow P_{b_{d-1}}$

Main loop

4. For $i = d$-2 to 0 by -1 do:
5. $Q \leftarrow [2^{e_{i+1}-e_i}]Q$.
6. If $b_i > 0$ then $Q \leftarrow Q + P_{b_i}$,
7. Else $Q \leftarrow Q - P_{-b_i}$
8. $Q \leftarrow [2^{e_0}]Q$
9. Return Q.

Figure 3.2.4 Signed *m*-ary Window Method

3.2.6 Cost Analysis

Table 3.2.6.1 lists a more detailed analysis of the cost of computing $[k]P$ in terms of field arithmetic operations. As usual, ***M*** and ***I*** indicate field multiplication and field inversion respectively. The result shows that the signed *m*-ary window method is superior to the unsigned methods.

Table 3.2.6.1 Cost of point multiplication from [BSS99]

			Curve			Total Cost	
Methods	Coordinate	r	ops	***M***	***I***	***I*=3*M***	***I*=10*M***
Binary	affine	n/a	151	302	151	755	1812
Modified *m*-ary	affine	4	128	256	128	640	1536
sliding windows	affine	4	124	248	124	620	1488
signed *m*-ary	affine	5	**122**	244	122	**610**	1464

It is also shown when the ratio ***I:M*** is relatively high over 1:10, total Cost exceeds two times than in *I:M*=1:3 As a consequence, in case of ***I:M*** over 10, field inversion operation should be highly taken into account when implementing elliptic curve cryptosystem.

3.3 Field Inversion and Multiplication

The Algorithms of point multiplication in elliptic curve cryptosystem are analyzed in section 3.2. It is found that when the ratio ***I/M*** is less than 0.1, the cost of field inversion in the point addition and doubling becomes significant.

To eliminate field inversion, other coordinate systems are considered to use in point operations of elliptic curve cryptosystem. In other word, some mixed coordinate systems are intended to use. Generally speaking, several coordinate systems besides affine coordinate system like Projective, Jacobian and Chudnovsky Jacobian coordinate are adopted in elliptic curve cryptosystem over $GF(2^n)$. They will be given in details in the following sections.

3.3.1 The formulae in Affine Coordinate System

This section introduces point operations of elliptic curve cryptosystem using affine coordinate over $GF(2^n)$. Affine coordinate is most easy to use and understand. Point representation is as $P(x,y)$. The procedure of point operations in elliptic curve cryptosystem is as follows [AMR+02].

A nonsingular elliptic curve E over $GF(2^n)$ is given by:

$$y^2 + xy = x^3 + a_2x + a_6 \qquad (3.1)$$

where $a_2, a_6 \in \mathrm{GF}(2^n)$, $a_6 \neq 0$.

- **Point Addition formula in affine coordinate**

Let $P = (x_1, y_1)$ and $Q = (x_2,y_2)$ be points on the elliptic curve E. Then. $P + Q = (x_1, y_1) + (x_2,y_2) = R(x_3, y_3)$, where $P \neq -Q$

$$\lambda = (y_1 + y_2)/(x_1 + x_2)$$

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a_2 \qquad (3.2)$$
$$y_3 = \lambda (x_1+x_3) + x_3 + y_1$$

It requires $1\boldsymbol{I} + 2\boldsymbol{M}$ operations for point addition in affine coordinate.

- **Point Doubling formula in affine coordinate**

The addition of a point to itself (Doubling a point) on the elliptic curve is computed as show below:

$$\boldsymbol{2P}\,(x_1, y_1) = \boldsymbol{R}(x_3, y_3); \text{ where } \boldsymbol{P} = \boldsymbol{Q};$$
$$\lambda = x_1 + y_1/x_1$$
$$x_3 = \lambda^2 + \lambda + a_2 \qquad (3.3)$$
$$y_3 = x_1^2 + (\lambda+1)\, x_3$$

It requires $1\,\boldsymbol{I} + 3\boldsymbol{M}$ operations for point doubling in affine coordinate.

3.3.2 The Formulae in Projective Coordinate System

A non-supersingular curve $E(F_{2^n})$ can be equivalently viewed as the set of all points $E'(F_{2^n})$ in the projective plane $P^2(F_{2^n})$ which satisfy [Men93]

$$y^2 z + xyz = x^3 + a_2 x^2 z^2 + a_6 z^3 \qquad (3.4)$$

By using projective coordinates, the inversion operation which is needed in Point addition and Point Doubling operating using affine coordinates can be eliminated and it is covered in the following sections.

3.3.2.1 Conversion Between Affine and Projective Coordinate Systems

Provided that for any point $(a,b) \in E(F_{2^n})$ in affine coordinates can be viewed as a 3-tuple $(x, y, z) \in E'(F_{2^n})$ in projective coordinates with $x = a$, $y = b$ and $z = 1$. Moreover, a point (tx,ty,tz) in projective coordinates with $t \neq 0$, is regarded as the same point as (x,y,z). Therefore the conversion methods between affine and projective coordinates are given as follows:

$$M(a,b) = M'(a,b,1)$$
$$N'(p,q,r) = N'(\frac{p}{r}, \frac{q}{r}, 1) = N(\frac{p}{r}, \frac{q}{r})$$

3.3.2.2 Curve Operations in the Projective Coordinate System

From procedure point of view, the method of formulating the equations of addition and doubling in projective coordinates is the same as for affine. In fact, conversion is taken on each projective point and then applied to Equation 3.2 and Equation 3.3.

Let $P'=(x_1 : y_1 : z_1) \in E'(F_{2^n}), Q'=(x_2 : y_2 : 1) \in E'(F_{2^n})$ and $P' \neq Q'$ where P', Q' are in projective coordinates. Since $P'=(x_1/z_1 : y_1/z_1 : 1)$, we can apply Equation 3.2 to point $P(x_1/z_1, y_1/z_1)$ and $Q(x_2, y_2)$ for $E(F_{2^n})$ in affine coordinates to find $P'+Q'=R'(x_3' : y_3' : 1)$. Then

$$\begin{aligned} x_3' &= \frac{B^2}{A^2}+\frac{B}{A}+\frac{A}{z_1}+a_2, \\ y_3' &= \frac{B}{A}\left(\frac{x_1}{z_1}+x_3'\right)+x_3'+\frac{y_1}{z_1} \end{aligned} \qquad (3.5)$$

where $A=(x_2 z_1 + x_1)$ and $B=(y_2 z_1 + y_1)$ [Men93]. In order to eliminate the inversion operations, the denominators of the expressions for x_3' and y_3' have to be eliminated. By setting $z_3 = A^3 z_1$ and from the property of projective coordinates, $x_3 = x_3' z_3$ and $y_3 = y_3' z_3$, if $P+Q=(x_3 : y_3 : z_3)$, then

$$\begin{aligned} x_3 &= AD \\ y_3 &= CD + A^2(Bx_1 + Ay_1), \\ z_3 &= A^3 z_1 \end{aligned} \qquad (3.6)$$

where $C=A+B$ and $D = A^2(A + a_2 z_1) + z_1 BC$.

Similarly, the formulae for $2P=(x_3 : y_3 : z_3)$ are,

$$\begin{aligned} x_3 &= AB, \\ y_3 &= x_1^4 A + B(x_1^2 + y_1 z_1 + A), \\ z_3 &= A^3 \end{aligned}$$

where $A = x_1 z_1$ and $B = a_6 z_1^4 + x_1^4$. The resulting point can be converted back to affine coordinates by multiplying each coordinate by z_3^{-1}. Note that there is no inversion

operation when calculating in projective coordinates. Therefore inversion can be eliminated by performing curve multiplication in projective coordinates.

3.3.3 The formulae in Jacobian Coordinate System

Following the Jacobian coordinate [CMO97][BSS99], point (x_1, y_1) are projected to (X,Y,Z) , $Z \neq 0$ where $x = X/Z^2$ and $y = Y/Z^3$. Elliptic curve equation becomes:

$$Y^2 + XYZ = X^3 + a_2X^2Z^2 + a_6Z^6 \qquad (3.7)$$

- **The formulae for point addition using Jacobian Coordinate as follows:**

$P = (X_1, Y_1, Z_1)$; $Q = (X_2, Y_2, Z_2)$; $P+Q = (X_3, Y_3, Z_3)$ where $P \neq Q$ or $-Q$

$$\lambda_1 = X_1Z_2^2$$
$$\lambda_2 = X_2Z_1^2$$
$$\lambda_3 = \lambda_1 + \lambda_2$$
$$\lambda_4 = Y_1Z_2^3$$
$$\lambda_5 = Y_2Z_1^3 \qquad (3.8)$$
$$\lambda_6 = \lambda_4 + \lambda_5$$
$$\lambda_7 = Z_1\lambda_3$$
$$\lambda_8 = \lambda_6X_2 + \lambda_7Y_2$$
$$Z3 = \lambda_7Z_2$$
$$\lambda_9 = \lambda_6 + Z_3$$
$$X_3 = a_2Z_3^2 + \lambda_6\lambda_9 + \lambda_3^3$$
$$Y3 = \lambda_9X3 + \lambda_8\lambda_7^2$$

It requires 20 ***M*** for point addition in Jacobian coordinate

- **The formulae for point doubling of *P* using Jacobian Coordinate is given by:**

$P = (X_1, Y_1, Z_1)$; $P + P = (X_3, Y_3, Z_3)$

$$Z_3 = X_1Z_1^2$$
$$X3 = (X_1 + a_6^2)^4$$
$$\lambda = Z_3 + X_1^2 + Y_1Z_1 \qquad (3.9)$$
$$Y_3 = X_1^4Z_3 + \lambda X_3$$

It requires 10 ***M*** for point doubling in the Jacobian coordinate.

3.3.4 The formulae in Chudnovsky Jacobian System

Upon the Chudnovsky-Jacobian coordinate described by DV Chudnovsky, GV Chudnovsky[CC86] and Cohen[CMO97], point (x_1, y_1) are projected to (X,Y,Z,Z^2,Z^3) , $Z \neq 0$ where $x = X/Z^2$ and $y = Y/Z^3$. Elliptic curve equation is:

$$Y^2 + XYZ = X^3 + a_2X^2Z^2 + a_6Z^6 \qquad (3.10)$$

- **The formulas for point addition using Chudvosky-Jacobian coordinate is as follows**[BSS99]:

$P = (X_1, Y_1, Z_1, Z_1^2, Z_1^3)$; $Q = (X_2, Y_2, Z_2, Z_2^2, Z_2^3)$; $P+Q = (X_3, Y_3, Z_3, Z_3^2, Z_3^3)$ where $P \neq Q$ or $-Q$

$$\lambda_1 = X_1Z_2^2$$
$$\lambda_2 = X_2Z_1^2$$
$$\lambda_3 = \lambda_1 + \lambda_2$$
$$\lambda_4 = Y_1Z_2^3$$
$$\lambda_5 = Y_2Z_1^3$$
$$\lambda_6 = \lambda_4 + \lambda_5 \qquad (3.11)$$
$$\lambda_7 = Z_1\lambda_3$$
$$\lambda_8 = \lambda_6X_2 + \lambda_7Y_2$$
$$Z_3 = \lambda_7Z_2$$
$$Z_3^2 = Z_3^2$$
$$Z_3^3 = Z_3^3$$
$$\lambda_9 = \lambda_6 + Z_3$$
$$X_3 = a_2Z_3^2 + \lambda_6\lambda_9 + \lambda_3^3$$
$$Y3 = \lambda_9X3 + \lambda_8\lambda_7^2$$

It requires 20 ***M*** for point addition in the Chudnovsky Jacobian coordinate.

- **The formulae for point doubling of *P* using Chudnovsky-Jacobian coordinate is given:**

$P = (X_1, Y_1, Z_1, Z_1^2, Z_1^3)$; $P + P = (X_3, Y_3, Z_3, Z_3^2, Z_3^3)$

$$Z_3 = X_1Z_1^2$$
$$Z_3^2 = Z_3^2$$
$$Z_3^3 = Z_3^3 \qquad (3.11)$$
$$X_3 = (X_1 + a_6^2)^4$$

$$\lambda = Z_3 + X_1^2 + Y_1 Z_1$$

$$Y_3 = X_1^4 Z_3 + \lambda X_3$$

It requires 10 ***M*** for point doubling in the Chudnovsky Jacobian coordinate

We summarize several coordinate conversions in this section and give a comparison on Cost of Computation in **Table 3.3.4.1**.

Table 3.3.4.1 Cost Comparison by mixing different coordinates EC point operation

	Doubling	Addition
Affine coordinate	1 ***I*** + 2 ***M***	1 ***I*** + 3 ***M***
Projective Coordinate	13***M***	7***M***
Jacobian coordinate	10 ***M***	20 ***M***
Chudnovsky Jacobian	10 ***M***	20 ***M***

In case of the ratio ***I/M*** less than 0.1, of all coordinate systems, cost in affine coordinate system is highest. As a consequence, in that case, other coordinate systems except for affine coordinate are recommended in the implementation. And the implementation using Jacobian or Chudnovsky-Jacobian coordinate is a better choice.

3.4 Serial-Parallel computation in ECC

3.4.1 Serial-Parallel computation of ECC using Jacobian Coordinate System (J-P-ECC)

The costs of point addition and doubling, for elimination of inversion are discussed in section 3.3. The computation uses only a digital serial multiplier. The computation using Jacobian and Chudnovsky coordinate systems are also described in Section 3.3. By exploring the inherent parallelism that exists in point addition and doubling of elliptic curve E over $GF(2^n)$, the architecture using 3 digital serial-parallel multipliers in elliptic curve cryptosystem has been proposed by Adnan [AM03]. The architecture makes point

addition and doubling faster. Efficiency over the algorithms in section 3.3 is significantly improved. The detail of serial-parallel computation on point addition and doubling in elliptic curve cryptosystem is given next.

The algorithm describes computation of point addition and doubling using Jacobian coordinate system (J-P-ECC) as given in section 3.3.4

In this algorithm, field addition in point operation is exclusive-OR, which is much faster than field multiplication and inversion. Due to field inversion being eliminated by mixing coordinate system, field Multiplications becomes the costliest operations in point operation. Finding least field multiplication in ECC becomes important.

Its principle is in that three field multiplications are paralleled to execute using 3 digital serial parallel multipliers within an instruction cycle. Computation cost is reduced to $7M$ in point addition and $6M$ in point doubling. The flow charts are given in **Figure 3.4.1** and **Figure 3.4.2**

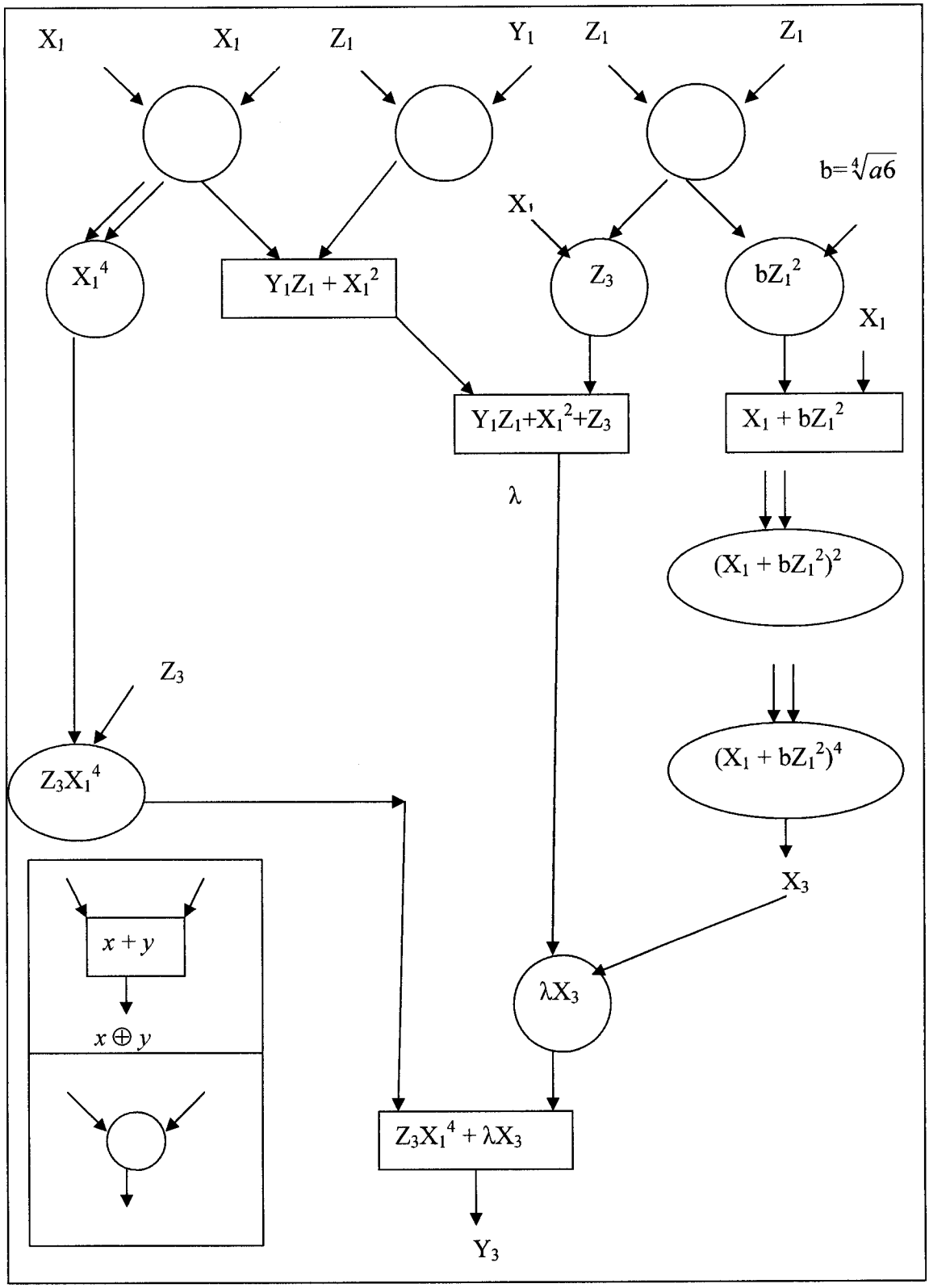


Figure 3.4.1 Data flow graph of point doubling with J-P-ECC

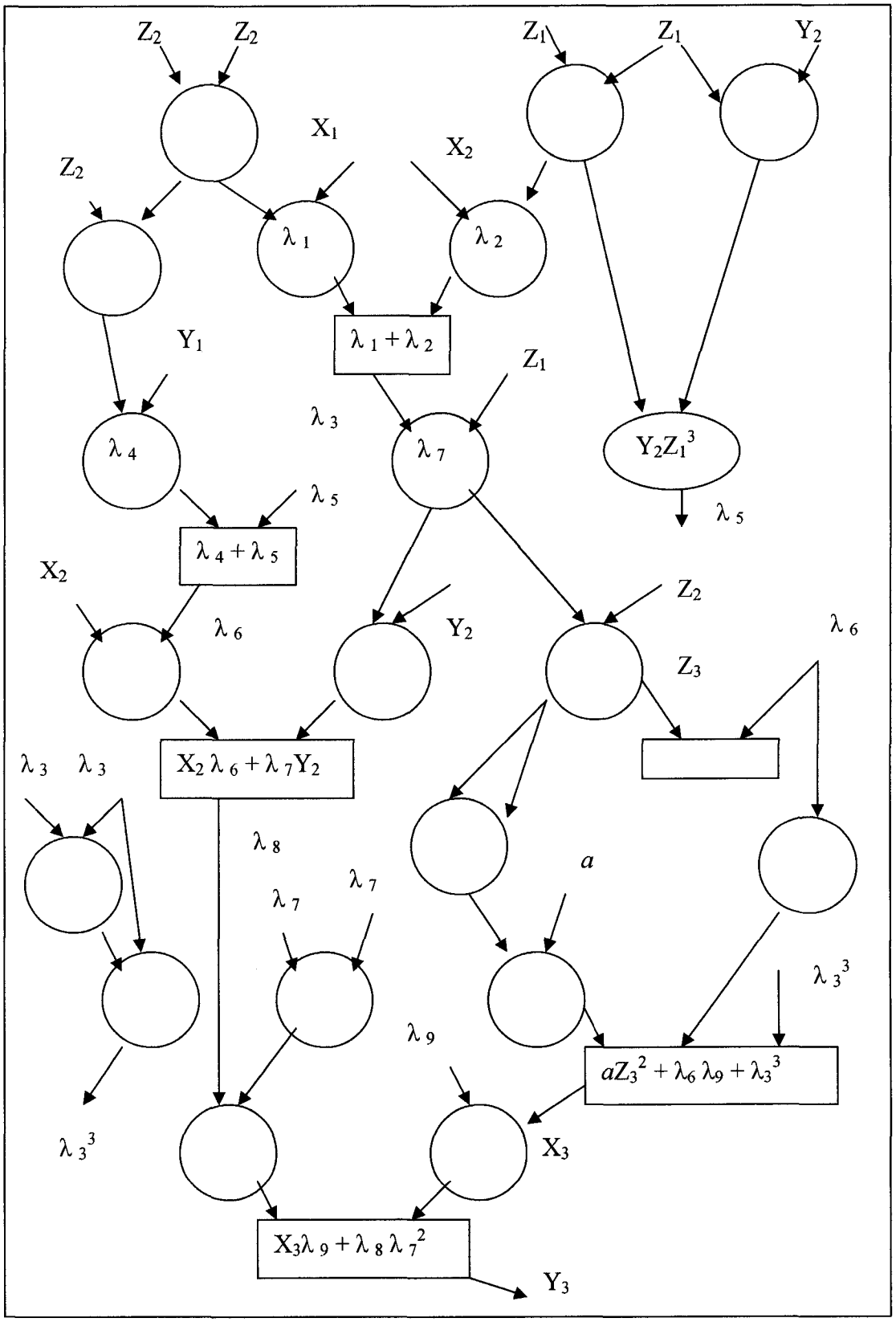


Figure 3.4.2 Data flow graph for adding two points with J-P-ECC

Cost Analysis

In this section, J-P-ECC is described in detail. Comparison between Serial and Serial-parallel computation is given in **Table 3.4.1.1**. Although serial-parallel computation may occupy more space in hardware, the efficiency is increased quite noticeably. From efficiency point of view, serial parallel computation used in ECC is worth doing.

Table 3.4.1.1 Cost comparison on curve operation between Serial and Serial-parallel computation

Cost Comparison	Doubling	Addition
Affine coordinate	$1I + 2M$	$1I + 3M$
Projective Coordinate	13M	7M
Jacobian coordinate	$10M$	$20M$
Chudnovsky Jacobian	$10M$	$20M$
Serial-Parallel EC with Jacobian coordinate system	$6M$	$7M$

Summary

This chapter gives some solutions for implementing efficient elliptic curve cryptosystems including the aspect of Point Multiplication and Point Addition/Doubling. The former focuses on the techniques for the mathematics problem "*Shortest add chain*". The latter uses field multiplication rather than inversion for cost reducing. It adopts the methods by mixing other coordinates with affine (2-Dimension) coordinate to eliminate field inversion. At last, upon further cost reducing, the algorithms J-P-ECC using serial-parallel computation is introduced. Comparing with conventional sequence computation in **Table 3.4.1.1**, this algorithm can reduce over half of field multiplication operations than in conventional ECC. It is proven that serial-parallel computation of ECC is an efficient way of software implementation of elliptic curve cryptosystem.

4 The optimized Serial-parallel computation of ECC over $GF(2^n)$

Software implementation of elliptic curve cryptosystem over $GF(2^n)$ has been introduced in Chapter 3. For reducing computational cost, field inversion is eliminated by mixing coordinate systems, as shown in Section 3.2, 3.3 and 3.4. **Table 3.4.1.1** shows that serial-parallel computation by using three digital serial parallel multipliers is superior to the conventional approaches described in section 3.2, 3.3, which use only serial computation using a digital serial multiplier in elliptic curve cryptosystem. This serial parallel system [AM03] is called J-P-ECC.

After analyzing the inherent parallel mechanism of section 3.3, we propose two new algorithms, P-P-ECC (described in section 4.1) and CC-P-ECC (described in section 4.2). Of these three algorithms, CC-P-ECC, that uses Chudnovsky Jacobian coordinate in ECC over $GF(2^n)$, is found to have a better computational efficiency than J-P-ECC and P-P-ECC.

4.1 First iteration of Serial-Parallel computation of ECC using Projective Coordinate (P-P-ECC) System

Elliptic curve E over $GF(2^n)$ using projective coordinate is represented in equation 3.4. The formula of point addition and doubling using projective coordinate is given from equation 3.5, 3.6.

To make the operation faster, serial-parallel computation is used in point addition and doubling of elliptic curve cryptosystem with projective coordinate system (P-P-ECC). The flow chart of its point addition and doubling are represented in **Figure 4.1.1** and **Figure 4.1.2**.

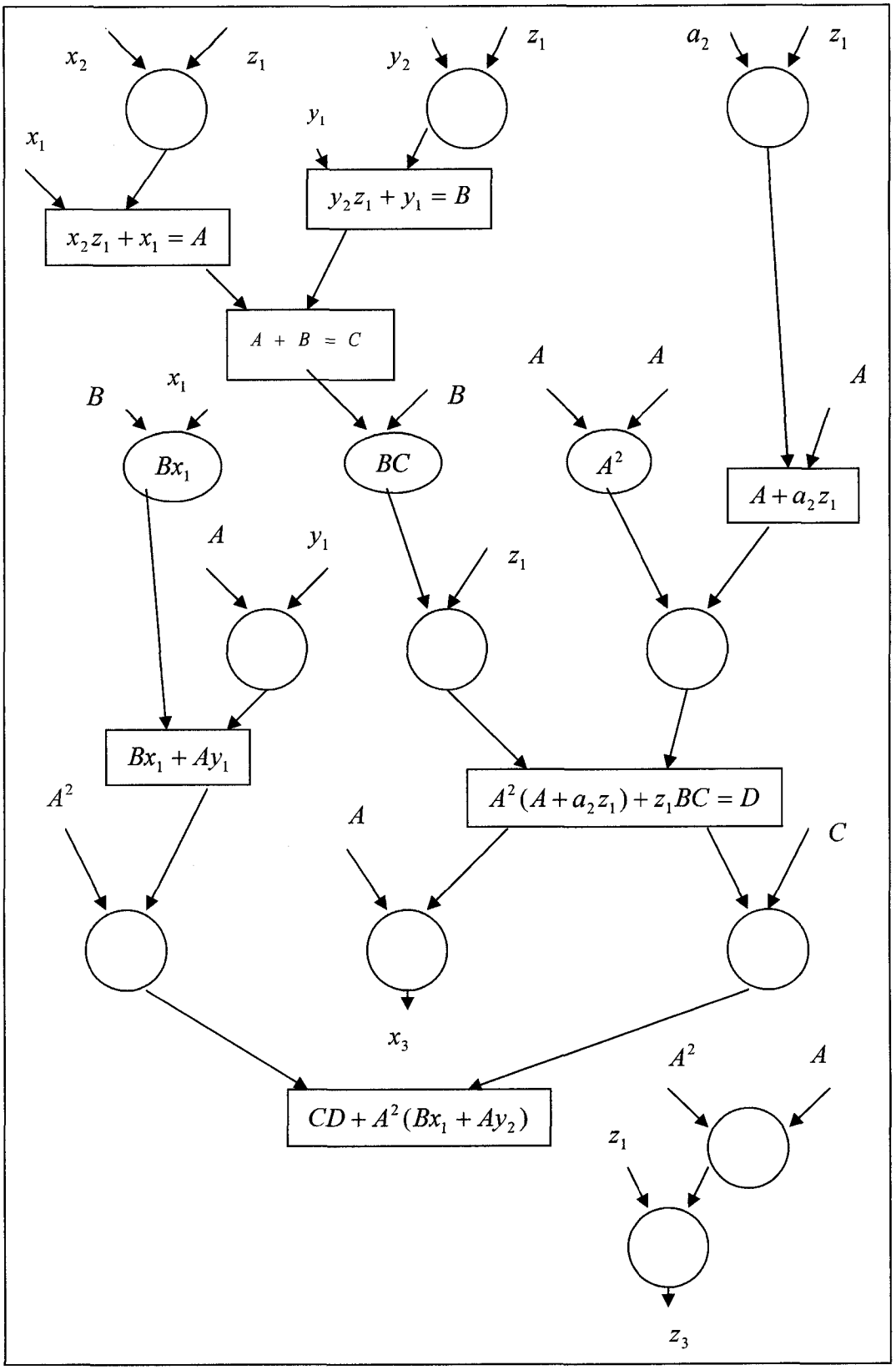


Figure 4.1.1 Flow Chart of Point Addition with P-P-ECC

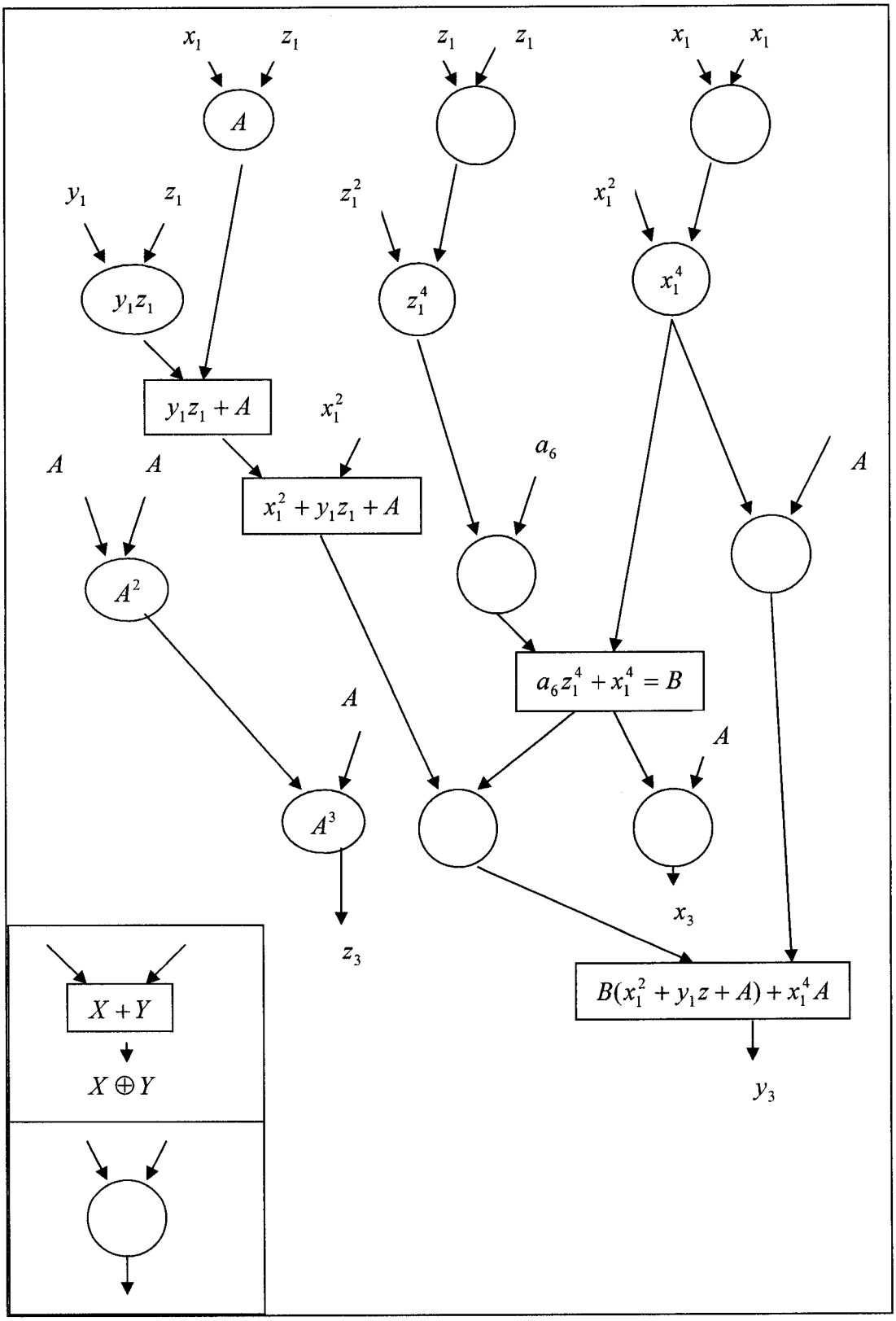


Figure 4.1.2 Flow Chart of Point Doubling with P-P-ECC

From **Figure 4.1.1, 4.1.2**, it is concluded that **P-P ECC** requires $6M$ and $4M$ in point addition and doubling, respectively.

4.2 Motivation of CC-P-ECC

With carefully analysis of Point addition and Doubling in **Figure 3.4.1** and **Figure 3.4.2**, some problems are noticed in that parallel multiplier without full load over $GF(2^n)$ reaches 60% in point doubling and 14% in point addition. It is undoubtedly a resource waste in serial-parallel computation. However, in Section 3.3.3, some operations like Z_2^2, Z_1^2 are required to be computed in advance. If digital serial parallel multipliers are available, it may certainly speed up elliptic curve cryptosystem. Our goal is to choose an optimized Serial-Parallel Computation of ECC (CC-P-ECC) to make computation of ECC more efficient. Of the coordinate systems described in Section 3.2, besides Projective coordinate, Chudnovsky Jacobian coordinate matches our requirement of Serial-Parallel Computation of ECC. Moreover, it has better performance than P-P-ECC. Point doubling and addition of ECC over $GF(2^n)$ **for CC-P-ECC`** is given in the following sections.

4.3 Introduction to CC-P-ECC Algorithm

CC-P-ECC is used to compute point doubling and addition of elliptic curve cryptosystem over $GF(2^n)$. It is qualified on the requirement of efficient ECC as follows:

- In case of the ratio of $I:M$ over 1:10, field inversion is considered as the most costly computation. To eliminate field inversion, Chudnovsky coordinate system can also be used to represent point of elliptic curve in point addition and doubling.
- Moreover, CC-P-ECC may be considered for reducing the number of field multiplications.

Architecture of CC-P-ECC is the same as in **Figure 3.4.1**. It can execute three field multiplications in parallel for a multiplication instruction. The operations are under

optimal normal basis. Exclusive-OR is used to compute field addition operation, which is much faster than field multiplication. Therefore performance of CC-P-ECC is mainly determined by field multiplication. By reducing rounds of 3-field multiplications, the efficiency of elliptic curve cryptosystem must prove.

In section 3.4, J-P-ECC is introduced [AM03]. It uses serial-parallel architecture of elliptic curve cryptosystem. In case of parallelizing computation of field multiplications, different coordinate systems have different performance. On comparing Projective, Jacobian and Chudnovsky Jacobian coordinate systems, this thesis proves that CC-P-ECC has much better performance than J-P-ECC and P-P-ECC. Moreover, as of today, it is the most efficient algorithm known for ECC.

4.3.1 Requirement of Software Implementation in CC-P-ECC

4.3.1.1 Mixed Coordinate System

P-P-ECC and J-P-ECC have been introduced in Chapter 3, In CC-P-ECC, affine and Chudnovsky Jacobian coordinate system is mixed in the computation of point computation of ECC. Point $\boldsymbol{P}$ is represented as $P(X,Y,Z,Z^2,Z^3)$.

4.3.1.2 The Algorithm for Point Multiplication

To simplify software implementation of elliptic curve cryptosystem, the evaluation of ECC algorithm is regarded on binary method of point multiplication described in section 3.1.1.

4.3.1.3 Optimal Normal Basis

Computation of ECC is over binary finite field. There are two kind of number presentation recommended by IEEE: Polynomial Normal basis and Optimal Normal Basis [IEEE1363]. In this theisis, we adopt Optimal Normal Base over $\boldsymbol{GF(2^n)}$.

Why use Optimal Normal Basis?

Optimal normal basis (ONB) is a special case of normal basis. Only AND, XOR and ROTATE operations are required in the normal basis. So it is easier to implementat the operations in hardware or software. Moreover, all of these operations are very fast on a compute node. Hence this thesis uses optimal normal basis as the choice for imlementing elliptic curve cryptosystem [Ros98a. pp-76].

What is Normal Basis?

Assume that β is an element in the field F_{p^m}, the polynomial representation is

$$\beta = a_n x^n + \ldots + a_1 x + a_0$$

where $n<m$.

A normal basis can be formed using the set

$$\{\beta^{p^m}, \ldots, \beta^{p^2}, \beta^p, \beta\}$$

If a finite field of characteristic 2 (i.e. p=2) is chosen, every element A in the field F_2^n can be uniquely represented in the form

$$A = \sum_{i=0}^{n-1} a_i \beta^{2^i}$$

where $a_i \in F_2$ and. $\beta \in F_{2^n}$

There are several operations among the elements over F_{2^n}, which are Addition, Squaring, multiplication and inversion[IEEE1363]. They will be discussed next.

Addition

Given that A, B are elements in the field. F_{2^n}

$$A = \sum_{i=0}^{n-1} a_i \beta^{2^i}$$

$$B = \sum_{j=0}^{n-1} b_j \beta^{2^j}$$

Addition is defined by

$$A + B = \left(\sum_{i=0}^{n-1} a_i \beta^{2^i}\right) + \left(\sum_{j=0}^{n-1} b_j \beta^{2^j}\right) \qquad (4.2.1)$$

Equation 4.2.1 can be rewritten as

$$A + B = \sum_{i=0}^{n-1} (a_i + b_i)\beta^{2^i}$$

where a_i, b_i are added modulo 2. Since there is no carry in finite field arithmetic, the operation of addition can be implemented as a bit-wise exclusive-OR (XOR) operation.

Squaring

Given that element A in finite field $GF(2^n)$, then A to the power 2 is as follows:

$$A^2 = \left(\sum_{i=0}^{n-1} a_i \beta^{2^i}\right)^2$$

$$= \sum_{i=0}^{n-1} a_i \left(\beta^{2^i}\right)^2$$

$$= \sum_{i=0}^{n-1} a_i \beta^{2^{i+1}}$$

Regarding the rules of finite field , there exists an equation $\beta^{2^n} = \beta^{2^0}$. So the squaring equation above can be rewritten as :

$$A^2 = a_{n-1}\beta + \sum_{i=0}^{n-2} a_i \beta^{2^{i+1}} \qquad (4.2.2)$$

In equation 4.2.2, there is a conclusion that squaring an element over F_{2^n} involves shifting each coefficient up to the next term and rotating the most significant coefficient down to the least significant position that is, rotate left operation.

Multiplication

Multiplication over field F_{2^n} is more complex than Addition and Doubling . The detail is described as follows:

Given that A and B in finite field F_{2^n}

$$A = \sum_{i=0}^{n-1} a_i \beta^{2^i}$$

$$B = \sum_{i=0}^{n-1} b_i \beta^{2^i}$$

and

$$C = A \times B = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j \beta^{2^i} \beta^{2^j} = \sum_{i=0}^{n-1} c_i \beta^{2^i}$$

therefore

$$\beta^{2^i} \beta^{2^j} = \sum_{k=0}^{n-1} \lambda_{ijk} \beta^{2^k} \tag{4.2.3}$$

where $\lambda_{ijk} \in \{0, 1\}$. Then multiplication can be written to be

$$c_k = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \lambda_{ijk} a_i b_j \tag{4.2.4}$$

where $0 \le k \le n\text{-}1$. By raising both sides of Equation 4.2.3 to the power of 2^{-l}, then

$$\left(\beta^{2^i} \beta^{2^j}\right)^{2^{-l}} = \beta^{2^{i-l}} \beta^{2^{j-l}} = \sum_{k=0}^{n-1} \lambda_{i-l,j-l,k} \beta^{2^k} = \sum_{k=0}^{n-1} \lambda_{ijk} \beta^{2^{k-l}} \tag{4.2.5}$$

Equating the coefficients of β^{2^0} in the above equation, yields

$$\lambda_{ijl} = \lambda_{i-l,j-l,0} \quad \text{for all } 0 \le i,j,l \le n\text{-}1 \tag{4.2.6}$$

Therefore Equation 4.2.4 can be written as

$$c_k = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \lambda_{i-k,j-k,0} a_i b_j = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \lambda_{ij0} a_{i+k} b_{j+k} \tag{4.2.7}$$

Regarding the description of optimal normal basis [MOVW89], ONB is one with the minimum number of nonzero terms in Equation 4.2.4, or equivalently, the minimum possible number of nonzero terms in λ_{ij} for a specific k.

There are two types of optimal normal bases [MOVW89], Type I and Type II.

The definition of Type I:

An optimal normal basis exists in F_{2^n} if

1. n+1 is a prime
2. 2 is a primitive in F_{n+1}

Rule 2 means 2 raised to any power in the range 0 to n modulo n+1 must result in a unique integer in the range 1 to n.

for a Type II ONB, there exsits an optimal normal basis in F_{2^n} if

1 $2n$+1 is prime, and either

2a. 2 is a primitive in F_{2n+1}, or

2b. $2n+1 \equiv 3 (\mathrm{mod}\, 4)$ and 2 generates the quadratic residues in F_{2n+1}.

Like Type I, rule 2a means that every 2^k modulo ($2n$+1) lies in the range 1 to $2n$ ($0 \leq k \leq 2n$-1). Therefore 2 is called the generator for all the possible locations in the $2n$+1 field. Rule 2b means that even if 2^k mod ($2n$ + 1) does not generate every element in the range 1 to $2n$, however, half of the points in the field formed by rule 2a can be hit. It is because $\sqrt{2^k} \bmod(2n+1)$ can be taken. The points generated by rule 2b are in the form of perfect squares [Rei87][Ros98a].

Inversion

Inversion of a is represented by a^{-1} and is defined as below.

$$aa^{-1} \equiv 1 \bmod n$$

where a and n are elements in field F_{2^n}. The algorithm used for inversion is derived from Fermat's Little Theorem

$$a^{-1} = a^{2^n-2} = \left(a^{2^{n-1}-1}\right)^2 \qquad (4.2.11)$$

for all $a \neq 0$ in F_{2^n}. The method used was proposed by Itoh and Tsujii [IT88], based on the following decomposition which minimizes the number of multiplications (squarings are much cheaper in a normal basis). If n is odd, then

$$2^{n-1}-1 = \left(2^{\frac{n-1}{2}}-1\right)\left(2^{\frac{n-1}{2}}+1\right)$$

and

$$a^{2^{n-1}-1} = \left(a^{2^{\frac{n-1}{2}}-1} \right)^{2^{\frac{n-1}{2}}+1}$$

and can be computed using one field multiplication provided $a^{2^{\frac{n-1}{2}}-1}$ is given. The cost of squaring is ignored because it is insignificant compared with multiplication. On the other hand, if n is even, then

$$2^{n-1}-1 = 2\left(2^{n-2}-1\right)+1 = 2\left(2^{\frac{n-2}{2}}-1\right)\left(2^{\frac{n-2}{2}}+1\right)+1$$

therefore

$$a^{2^{n-1}-1} = a^{2\left(2^{\frac{n-2}{2}}-1\right)\left(2^{\frac{n-2}{2}}+1\right)+1}$$

which takes two field multiplications if $a^{2^{\frac{n-2}{2}}-1}$ is given.

4.3.2 The Equation of ECC over *GF*(2^n) with Chudnovsky Jacobian Coordinate System

As Chudnovsky Jacobian described in [CC86][CMO97], point (x_1, y_1) is projected to (X,Y,Z,Z^2,Z^3) , $Z \neq 0$ where $x = X/Z^2$ and $y = Y/Z^3$. Elliptic curve equation is as follows:

$$Y^2 + XYZ = X^3 + a_2X^2Z^2 + a_6Z^6$$

Where $a_2, a_6 \in Fq$, $q=2^n$, $a_6 \neq 0$

4.3.3 Point addition and doubling using Chudnovsky Coordinate System

- **The formulae for point addition are as follows:**

$P = (X_1, Y_1, Z_1, Z_1^2, Z_1^3)$; $Q = (X_2, Y_2, Z_2, Z_2^2, Z_2^3)$; $P + Q = (X_3, Y_3, Z_3, Z_3^2, Z_3^3)$

where $P \neq Q$ or $-Q$

$$\lambda_1 = X_1Z_2^2$$

$$\lambda_2 = X_2 Z_1^2$$

$$\lambda_3 = \lambda_1 + \lambda_2$$

$$\lambda_4 = Y_1 Z_2^3$$

$$\lambda_5 = Y_2 Z_1^3$$

$$\lambda_6 = \lambda_4 + \lambda_5$$

$$\lambda_7 = Z_1 \lambda_3$$

$$\lambda_8 = \lambda_6 X_2 + \lambda_7 Y_2$$

$$Z_3 = \lambda_7 Z_2$$

$$\lambda_9 = \lambda_6 + Z_3$$

$$X_3 = a_2 Z_3^2 + \lambda_6 \lambda_9 + \lambda_3^3$$

$$Y_3 = \lambda_9 X_3 + \lambda_8 \lambda_7^2$$

- **The formulae for point doubling of *P* are given by:**

$$P = (X_1, Y_1, Z_1, Z_1^2, Z_1^3);\ P + P = (X_3, Y_3, Z_3, Z_3^2, Z_3^3)$$

$$Z_3 = X_1 Z_1^2$$

$$X_3 = (X_1 + a_6^2)^4$$

$$\lambda = Z_3 + X_1^2 + Y_1 Z_1$$

$$Y_3 = X_1^4 Z_3 + \lambda X_3$$

Based on the formulas given above, CC-P-ECC algorithm, an attempt has been made today more field multiplications in 3 digital serial parallel multipliers in every round. Not only does it exploit hardware resource up to maximum, it also saves more rounds of field multiplication. The Flow Chart about point addition and doubling are given in **Figure 4.3.1** and **Figure 4.3.2**, respectively.

In CC-P-ECC, Point Addition requires 5 round field multiplications and 4 in Point Doubling. Compared with J-P-ECC in **Figure 3.4.1** and **Figure 3.4.2**, CC-P-ECC algorithm's computation saves up to 20% and 28.6% on point doubling and addition, respectively.

From the analysis of CC-P-ECC, the advantages are summarized as follows:

- Performance improvement on point computation like Point Addition and Doubling
- Three digital serial Parallel Multipliers are used in CC-P-ECC architecture
- No change in hardware architecture described in J-P-ECC
- Easier software Implementation

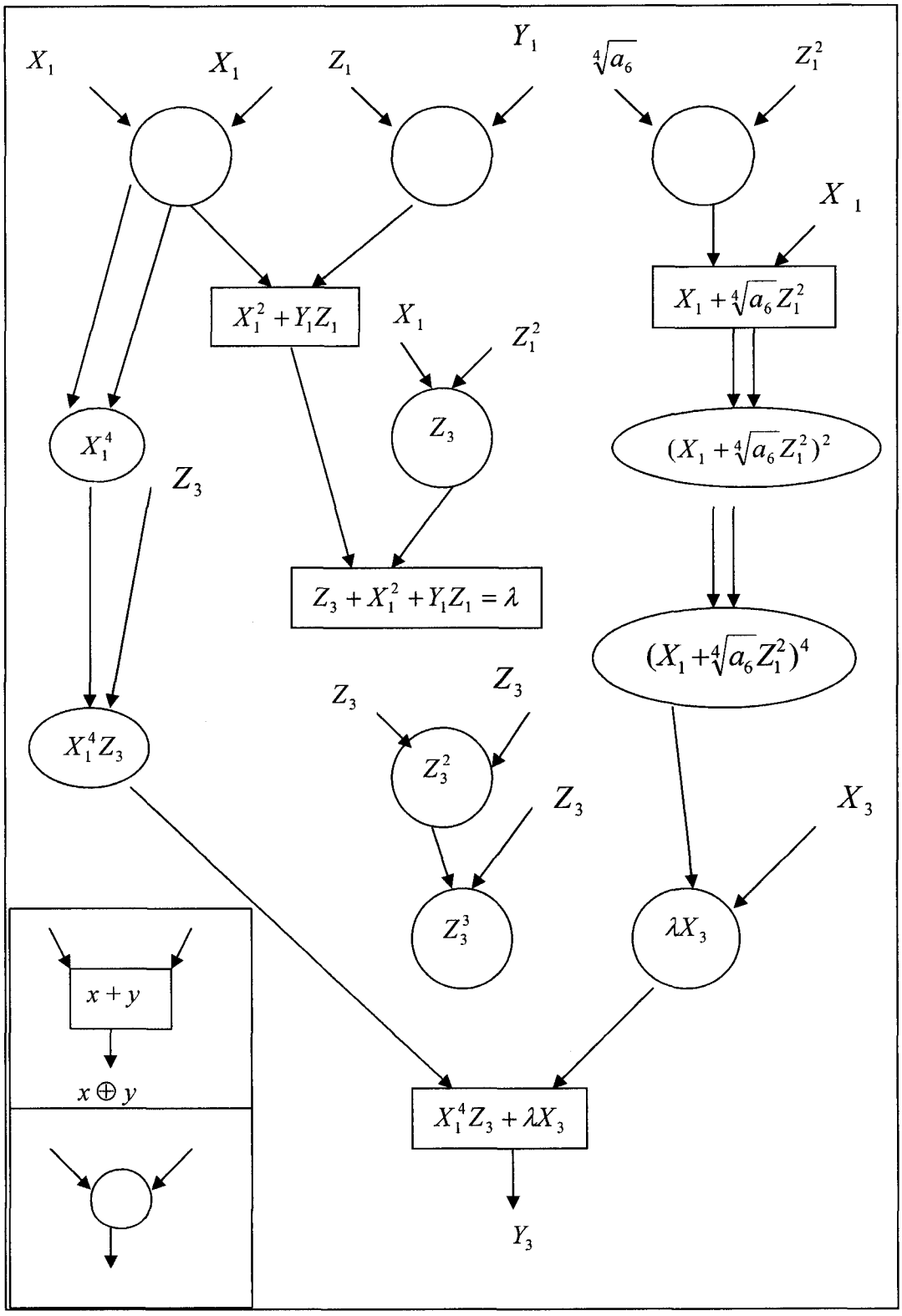


Figure 4.3.1 Data flow on Point Doubling with CC-P-ECC

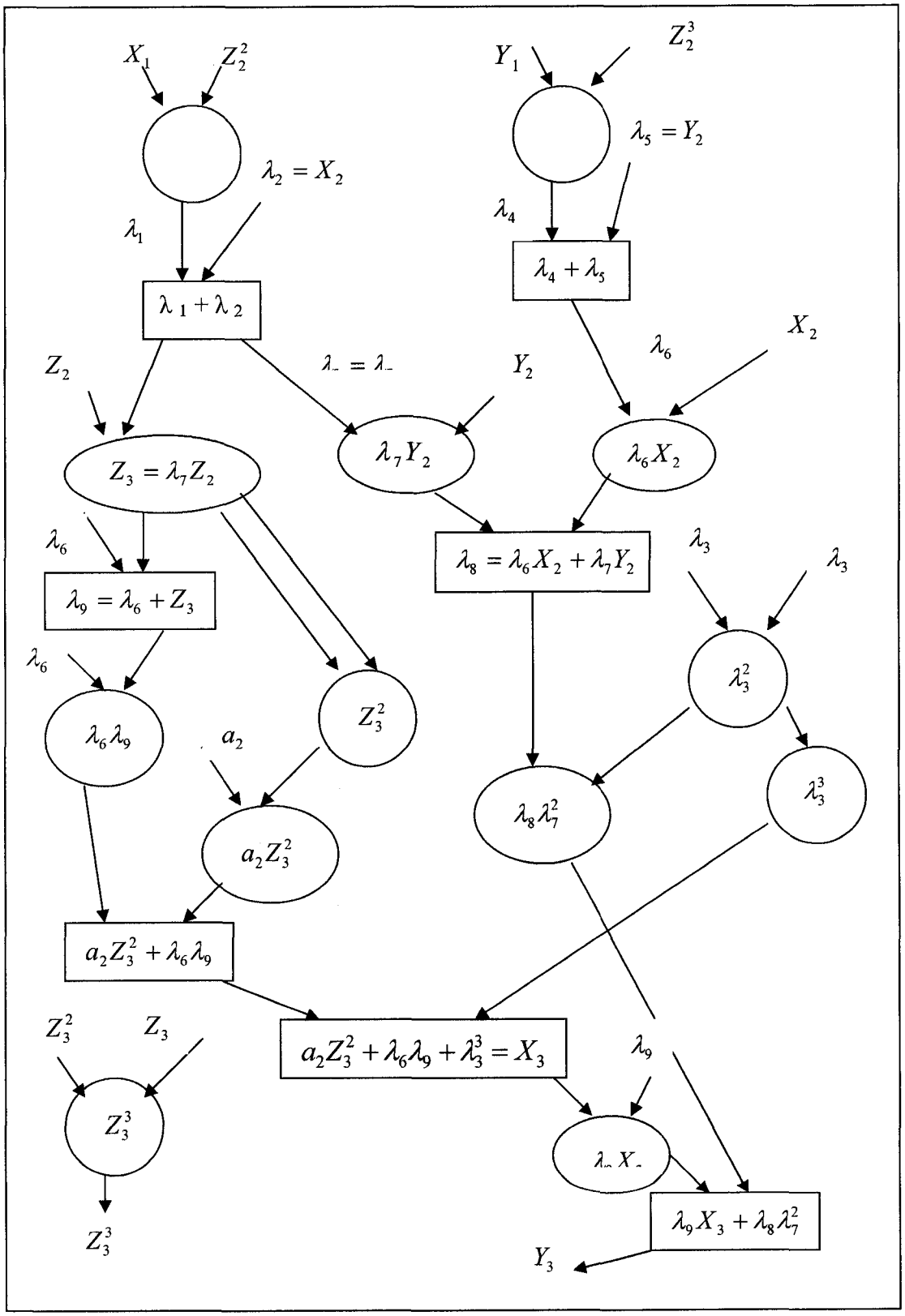


Figure 4.3.2 Data flow on Point Addition with CC-P-ECC

4.4 The application with CC-P-ECC: ECDSA

The description of CC –P-ECC had been given in section 4.2. This section introduces one important application using CC –P-ECC, ECDSA [IEEE 1363].

4.4.1 WHY USE ECDSA IN APPLICATION

Like DSA in public key cryptosystems, ECDSA is also required in some applications using elliptic curve cryptosystem.

ECDSA purpose is to sign on a hashed message with someone's private key to generate a new hash message with a pair of string. Then this message is passed to the person who requires it. The receiver can verify that the message has been signed by the sender.

4.4.2 ECDSA DESCRIPTION

As discussion above, ECDSA is an important application using Elliptic Curve in public key cryptosystem [IEEE1363].

ECDSA uses a random EC_KEYPAIR, along with the signer's private key, to create the signature. In the phase of verification, you use public key to decrypt the signature. If it matches the hash values, it verifies the signature.

As elliptic curve described in section 2.3, let P be the base point with order n on curve $\boldsymbol{E}$, which satisfies equation 2.3.4. We call the signer's private key s and the public key $Q = sP$. Let 's take a random value k and random point $R = kP$. The message hash is e and has been generated to be less than n. The first step in DSA is to take the x component of R modulo the order of the curve to get the first signature component:

$$c = x \bmod n \qquad (4.6.1)$$

The second component is then computed as:

$$d = k^{-1}(e + sc) \qquad (4.6.2)$$

According to the IEEE [IEEE1363], the process of verifying the signature from equation (4.6.1) and (4.6.2) requires computation of three values after computing the hash of the message (called e'):

$$\begin{aligned} h &= d^{-1} \bmod n \\ h_1 &= e'h \bmod n \\ h_2 &= ch \bmod n \end{aligned} \qquad (4.6.3)$$

These values are used to compute a point on the public elliptic curve with the formula:

$$R' = h_1 P + h_2 Q \qquad (4.6.4)$$

If the x component of equation (4.6.4) does not equal equation (4.6.1), the message is assumed to be different from the original signed document. The reason is as follows:

The first equation in equation (4.6.3) can be rewritten with equation (4.6.2):

$$h = k(e + sc)^{-1}. \qquad (4.6.5)$$

With this, the last two equations in equation (4.6.3) expand to:

$$\begin{aligned} h_1 &= e'k(e + sc)^{-1} \\ h_2 &= ck(e + sc)^{-1} \end{aligned} \qquad (4.6.6)$$

Putting the above fully expanded terms into equation (4.6.4) gives:

$$R' = e'k(e+sc)^{-1}P + sck(e+sc)^{-1}P \qquad (4.6.7)$$

where also substituted equation $Q = sP$ .Clearly there are common terms, which reduce down to:

$$R' = k(e'+sc)(e+sc)^{-1}P \qquad (4.6.8)$$

The factors $(e'+sc)(e+sc)^{-1}$ will erased only if the hash of the original message, the signer's key, and the published signature are correct.

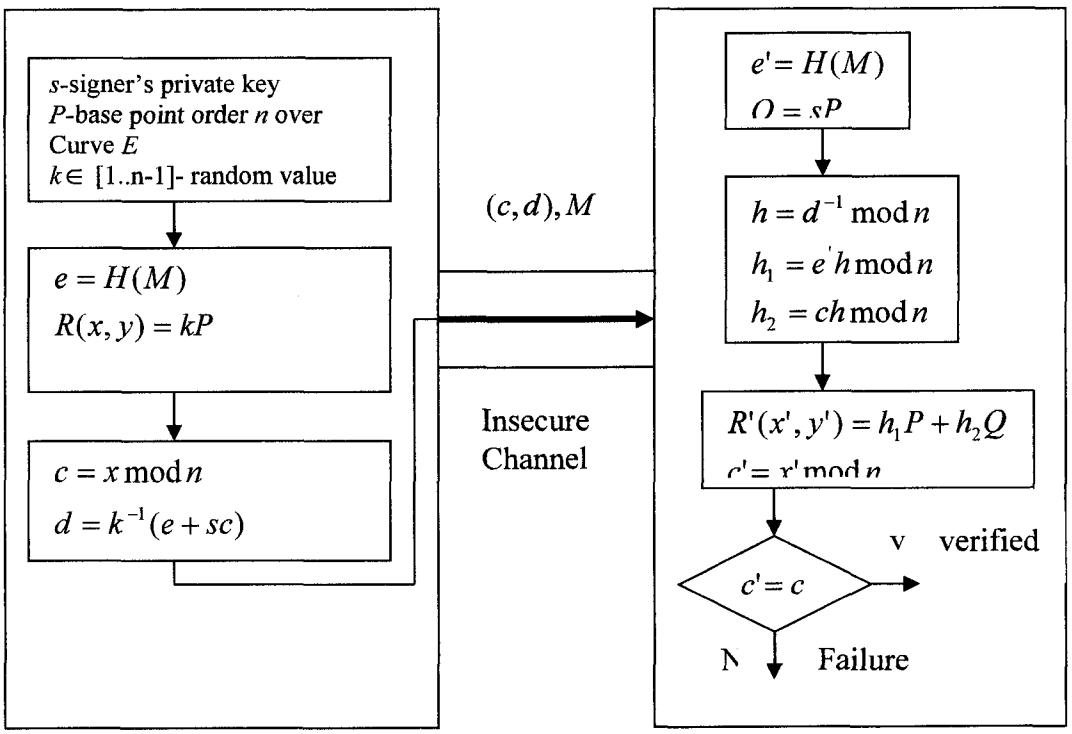


Figure 4.4.1 Digital Signature with message digest in elliptic curve Cryptosystem

4.4.3 ECDSA IMPLEMENTATION

In **Figure 4.4.1**, Implementation of ECDSA is composed of three components, ECC Key Generation, Digital Signature and Message Verification. The three parts in detail are given in the following section.

4.4.3.1 ECC Key Generation

ECKGP is used to compute identity public key Q with specific private key s. Private key is a random number that is modulo order n of Base Point. Then it is multiplied by Base Point to compute counterpart public key. Its flow chart is as **Figure 4.4.2**.

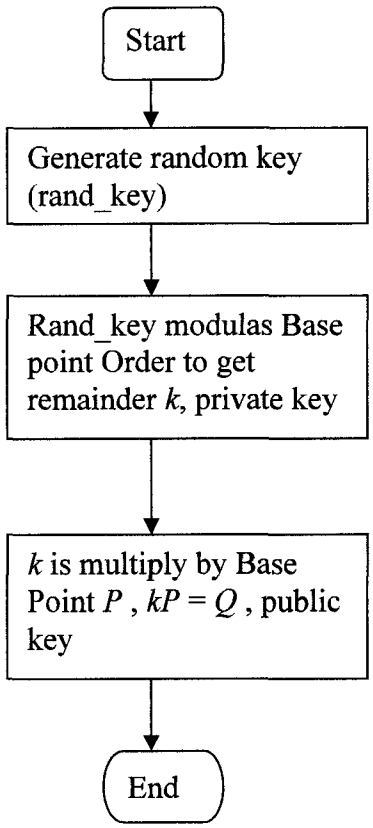


Figure 4.4.2 Elliptic Curve Key Generation Subroutine

4.4.3.2 ECDSA Signature

ECDSA Signature is a function of signing a hashed message with someone's private key to generate a new message with a pair of strings before sending it to the Destination. It first hashes the given message. Then use signer's private key and random point's public key to compute a pair of string. Finally, it passes the original message with generated pair string to destination. The procedure is shown in **Figure 4.4.3**.

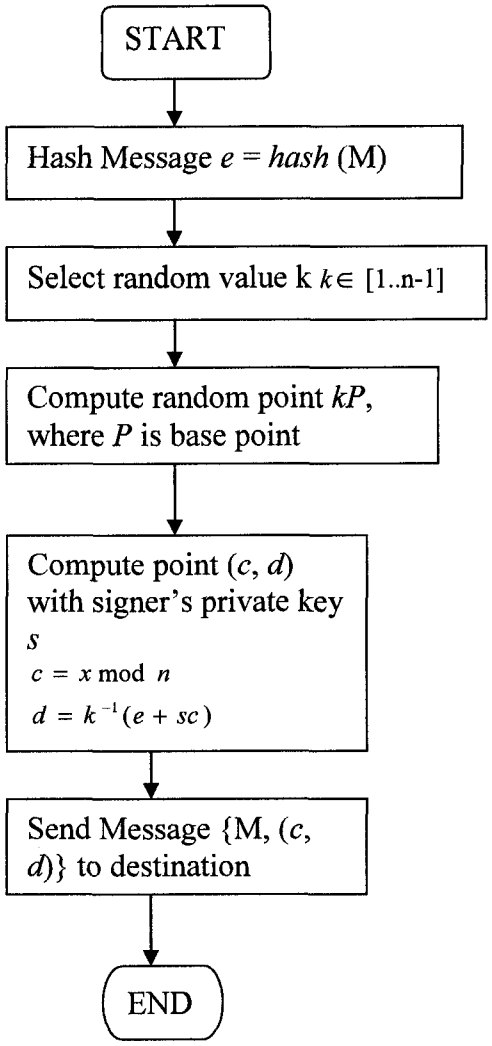


Figure 4.4.3 ECDSA signature subroutine

4.4.3.3 ECDSA Verification

After the message signed by specific private key is passed to the verifier, there requires message check if the message is original. ECDSA verification is designed for this particular mission. From the discussion in section 5.3.2, ECDSA verification subroutine uses signer's public key Q , hashed Message e' and a pair of string (c, d) passed from "signer" to compute a string c'. If c is equal to c', the message is sent from the correct signer or not otherwise. The procedure is as follows:

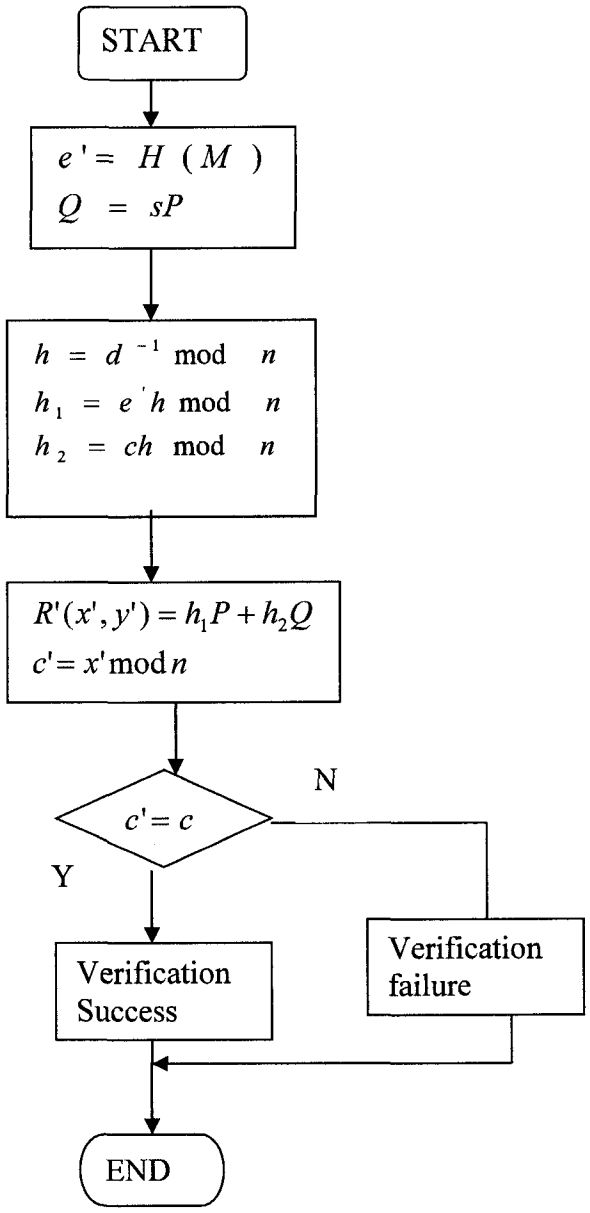


Figure 4.4.4 ECDSA verification subroutine

4.4.3.4 ECDSA with CC-P-ECC

With introduction to Elliptic curve cryptosystem in Chapter 2, we know Point Multiplication composed of Point Addition and Doubling plays an import role on security and performance. In this thesis, because point multiplication with CC-P-ECC is used in ECC key generation, signature and verification subroutine of ECDSA, it undoubtedly can

make ECDAS to be improved on computational efficiency. For performance, Chapter 5 will give some experiments and results.

Summary

In this section, CC-P-ECC Algorithm over $GF(2^n)$ is given in detail. The computational costs for CC-P-ECC are less than those for J-P-ECC by 20% and 28.6% on Point Doubling and Addition respectively. P-P-ECC saves 20% and 14.2% respectively as compared to J-P-ECC. Thus CC-P-ECC is the best candidate for ECDSA. The experiment results of ECDSA for CC-P-ECC and the other two parallel systems are given in Chapter 5.

5 Experiments and Results

This section describes the experiments for testing the performance of CC-ECC, P-ECC and J-ECC. The measurement is done for the Key Generation, Digital Signature and Message Verification processes. Besides cost comparison among Projective, Jacobian

and Chudnovsky Jacobian coordinate system with a digital serial multiplier, it also includes the measurements with multiple serial parallel multipliers for J-P-ECC, P-P-ECC and CC-P-ECC.

5.1 Description of the Experiment

In this section, a description of the experimental platform is given. The values of the selected elliptic curve parameters are also provided.

5.1.1 System Requirement

Hardware

We adopt Davinci Server, which is located in the school of Computer Science of University of Windsor, as my test platform. The reason we use it is that it is suitable for high performance computation. It has 12 CPUs and plenty of storage space for multiple users to use simultaneously. The operating system is UNIX System,

Software

The operating System is Solaris 9 Unix system. Rosing Software Package [Ros98a] is used to measure Digital Signature of elliptic curve cryptosystem. This software is programmed with C Language. So all test subroutines are programmed with C Language. We choose cc.exe as C compiler, which is included in the core package of Solaris UNIX.

5.1.2 Selection of Elliptic Curve and Parameters

Based on DIGITAL SIGNATURE STANDARD [FIPS1862] recommendation, Random Curve and Koblitz Curve may both be adopted in Polynomial [IEEE1363] basis and Normal basis in a practical application. In my experiment, Koblitz curve with normal Basis is chosen. The Koblitz Curve over Binary Field 2^n has the form:

$$y^2 + xy = x^3 + a_2 x^2 + 1 \qquad (5.1)$$

where a_2 = 0 or 1.

Of the Field Representation, optimal normal basis with **Type I** described in Chapter 4 (Page 38 - 40) is chosen in this experiment.

The parameter used in the experiment:

- Galois Field (2^{113})
- Coefficient a_2 =1, a_6=1
- Base Point (G_x,G_y)

  G_x : 1f43c 6942b1a4 9aaaac4a b572fdbf

  G_y : 10145 c084d629 96208f8e 44d9f291

5.1.3 Simulation of Serial-Parallel computation of ECC

Simulation of Serial-Parallel Computation is proposed to execute 3 field multiplication operations with 3 digital serial parallel multipliers once a time. Every field multiplication is individually executed in a separated process. In our experiment, three field multiplications can be executed in three separated processes created almost at the same time.

In this thesis, three digital serial-parallel multipliers are used to parallelize field multiplications in point addition and doubling of ECC. In the UNIX environment, ECC program employs 4 processes. One is to control the main function. The other three processes are used to simulate 3 independent field multiplication operations.

5.2 Measurements

The measurements are divided into two groups. One group is tested in the conventional elliptic curve cryptosystem, which has one digital serial multiplier in elliptic curve system. The other are tested for serial-parallel computation under simulation environment. Execution times of ECDSA including signature and verification are measured. The measured subroutines include Key Generation, Signature and Verification.

5.2.1 Conventional ECC computation on various coordinate systems

ECDSA has already been introduced in Chapter 4. In this section, the tests are done with different coordinate systems. Test parameters are specified in section 5.1.2. The parameters used in the test are given again in every test.

Conventional ECDSA is measured with a digital serial multiplier in various coordinate systems as follow.

5.2.1.1 Conventional ECDSA with Projective Coordinate System

This test is purposed to measure execution time of Key Generation, Signature and Verification with point represented by Projective Coordinate System. The result is shown in Test I. To measure more accurately, a number of readings are taken. The average value is calculated to represent the execution time.

Test I:

```
Koblitz 113
a2 :    1ffff ffffffff ffffffff ffffffff
a6 :    1ffff ffffffff ffffffff ffffffff
random point
x :     7d17 b7423658 5f8dae64 d624d542
y :    100a4 823ec133 6f1d883c a95f43ac
Base point
x :    1f43c 6942b1a4 9aaaac4a b572fdbf
y :    10145 c084d629 96208f8e 44d9f291
Key Generation time=0.340000 seconds using Projective Coordinate
Signer's secret key :     1988 296d3d8c a8d08ecf 91ff45bf
Signers public key
x :     282b a9e578b1  bcd0319 24bdb6fc
y :     2950 906bb03b 415e5ad8 4bd2c559
Signature take time=0.340000 seconds using Projective coordinate
first component of signature :     c0c2 421b5284 57235894 76f9862a
second component of signature :     8ca2 a1dccee5 7cde9114 a74ff0a3
Verify Signature time=0.690000 seconds using Projective Coordinate
Message Verifies
```

Table 5.2.2.1 in the Appendix C shows the other readings. Average Value of Key Generation, Signature and Verification are 0.341, 0.3460 and 0.695 sec, respectively.

5.2.1.2 Conventional ECDSA with Jacobian Coordinate System

The following test is done with the Jacobian Coordinate System. The result is shown as Test II.

Test II:

```
Koblitz 113
a2 :    1ffff ffffffff ffffffff ffffffff
a6 :    1ffff ffffffff ffffffff ffffffff
random point
x :     7d17 b7423658 5f8dae64 d624d542
y :    100a4 823ec133 6f1d883c a95f43ac
Base point
x :    1f43c 6942b1a4 9aaaac4a b572fdbf
y :    10145 c084d629 96208f8e 44d9f291
Key Generation time=0.310000 seconds using Jacobian Coordinate
Signer's secret key :     1988 296d3d8c a8d08ecf 91ff45bf
Signers public key
x :     282b a9e578b1  bcd0319 24bdb6fc
y :     2950 906bb03b 415e5ad8 4bd2c559
Signature take time=0.310000 seconds using Jacobian Coordinate
first component of signature :     c0c2 421b5284 57235894 76f9862a
second component of signature :     8ca2 a1dccee5 7cde9114 a74ff0a3
Verify Signature time=0.610000 seconds Jacobian Coordinate
Message Verifies
```

The average of Key Generation, Signature and Verification are 0.3034, 0.3084, 0.6125 sec. (**Table 5.2.2.2** in **Appendix C**)

5.2.1.3 Conventional ECDSA with Chudnovsky Coordinate System

The following data is measured in the Conventional ECDSA with the Chudnovsky Jacobian Coordinate System.

```
Test III:
Koblitz 113
a2 :    1ffff ffffffff ffffffff ffffffff
a6 :    1ffff ffffffff ffffffff ffffffff
random point
x :     7d17 b7423658 5f8dae64 d624d542
y :    100a4 823ec133 6f1d883c a95f43ac
Base point
x :    1f43c 6942b1a4 9aaaac4a b572fdbf
y :    10145 c084d629 96208f8e 44d9f291

Key Generation time=0.300000 seconds using Chudnovsky Coordinate
Signer's secret key :     1988 296d3d8c a8d08ecf 91ff45bf
Signers public key
x :     282b a9e578b1  bcd0319 24bdb6fc
y :     2950 906bb03b 415e5ad8 4bd2c559
Signature take time=0.320000 seconds using Chudnovsky Coordinate
first component of signature :     c0c2 421b5284 57235894 76f9862a
second component of signature :     8ca2 a1dccee5 7cde9114 a74ff0a3
Verify Signature time=0.620000 seconds using Chudnovsky Coordinate
Message Verifies
```

The average of Key Generation, Signature and Verification are 0.3041, 0.3084, 0.6134 sec (**Table 5.2.2.3** in **Appendix C**).

5.2.1.4 Performance Analysis

Table 5.2.1.1 Comparison on Conventional ECDSA using three coordinate systems

Time (Seconds)	Key Generation	Signature	Verification
P-ECC	0.341	0.3460	0.695
J-ECC	0.3034	0.3084	0.6125
CC-ECC	0.3041	0.3084	0.6134

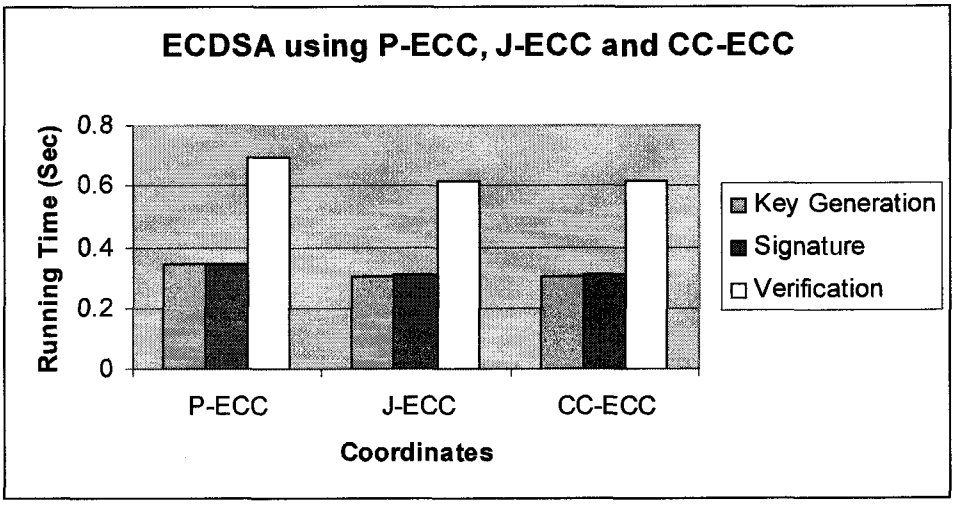


Figure 5.2.1 Comparison of running time using P-ECC, J-ECC and CC-ECC

From **Table 5.2.1.1** and **Figure 5.2.1**, comparing ECDSA the three coordinates of elliptic curve cryptosystem, Jacobian and Chudnovsky coordinate is almost the same and the Projective is slower than Jacobian and Chudnovsky around 11%, 11%, 12% with Key Generation, Message Signature and Message Verification, respectively. So Jacobian and Chudnovsky Jacobian should be the better solution for implementing ECDSA using a digital serial multiplier.

5.2.2 Serial-Parallel ECC computation on various coordinate system

The following section gives the measurement on ECDSA using simulation of serial parallel computation with projective, Jacobian and Chudnovsky coordinate system, respectively.

5.2.2.1 Measurement of ECDSA with serial-parallel multiplier simulator using Projective Coordinate System (P-P-ECC)

The measurement of ECDSA using P-ECC is taken and the data is given in **TEST IV** and **Table 5.2.2.4** in Appendix C

TEST IV:

```
Koblitz 113
a2 :    1ffff ffffffff ffffffff ffffffff
a6 :    1ffff ffffffff ffffffff ffffffff
random point
x :     7d17 b7423658 5f8dae64 d624d542
y :    100a4 823ec133 6f1d883c a95f43ac
Base point
x :    1f43c 6942b1a4 9aaaac4a b572fdbf
y :    10145 c084d629 96208f8e 44d9f291
Key Generation time=11.160000 seconds using Projective Coordinate
Signer's secret key :     1988 296d3d8c a8d08ecf 91ff45bf
Signers public key
x :     282b a9e578b1  bcd0319 24bdb6fc
y :     2950 906bb03b 415e5ad8 4bd2c559
Signature take time=11.120000 seconds using Projective coordinate
first component of signature :     c0c2 421b5284 57235894 76f9862a
second component of signature :     8ca2 a1dccee5 7cde9114 a74ff0a3
Verify Signature time=21.930000 seconds using Projective Coordinate
Message Verifies
```

The Average execution time in the ECDSA using P-P-ECC is 9.99, 9.97 and 19.74 seconds on Key Generation, Signature and Verification, respectively.

5.2.2.2 Measurement of ECDSA with serial-parallel multiplier simulator using Jacobian Coordinate system (J-P-ECC)

ECDSA measurement using J-P-ECC is described in this section. Data about Key Generation, Signature and Verification using J-P-ECC is shown in **Test V** and **Table5.2.2.5 in Appendix C.**

TEST V:

```
Koblitz 113
a2 :    1ffff ffffffff ffffffff ffffffff
a6 :    1ffff ffffffff ffffffff ffffffff
random point
x :     7d17 b7423658 5f8dae64 d624d542
y :    100a4 823ec133 6f1d883c a95f43ac
Base point
x :    1f43c 6942b1a4 9aaaac4a b572fdbf
y :    10145 c084d629 96208f8e 44d9f291
Key Generation time=13.960000 seconds using Jacobian Coordinate
Signer's secret key :     1988 296d3d8c a8d08ecf 91ff45bf
Signers public key
x :     282b a9e578b1  bcd0319 24bdb6fc
y :     2950 906bb03b 415e5ad8 4bd2c559
Signature take time=12.880000 seconds using Jacobian Coordinate
first component of signature :     c0c2 421b5284 57235894 76f9862a
second component of signature :     8ca2 a1dccee5 7cde9114 a74ff0a3
Verify Signature time=26.520000 seconds Jacobian Coordinate
Message Verifies
```

Average execution time in the ECDSA using J-P-ECC is 10.62, 10.57, 20.76 seconds on Key Generation, Signature and Verification, respectively.

5.2.2.3 Measurement of ECDSA with serial-parallel multiplier simulator using Chudnovsky Jacobian Coordinate System (CC-ECC)

The following test is for the measurement of ECDSA with CC-P-ECC. The result is shown in Table VI and **Table 5.2.2.6** in **Appendix C**.

TEST VI:

```
Koblitz 113
a2 :    1ffff ffffffff ffffffff ffffffff
a6 :    1ffff ffffffff ffffffff ffffffff
random point
x :     7d17 b7423658 5f8dae64 d624d542
y :    100a4 823ec133 6f1d883c a95f43ac
Base point
x :    1f43c 6942b1a4 9aaaac4a b572fdbf
y :    10145 c084d629 96208f8e 44d9f291
Key Generation time=8.710000 seconds using Chudnovsky Coordinate
```

```
Signer's secret key :      1988 296d3d8c a8d08ecf 91ff45bf
Signers public key
x :      282b a9e578b1  bcd0319 24bdb6fc
y :      2950 906bb03b 415e5ad8 4bd2c559
Signature take time=9.080000 seconds using Chudnovsky Coordinate
first component of signature :      c0c2 421b5284 57235894 76f9862a
second component of signature :      8ca2 a1dccee5 7cde9114 a74ff0a3
Verify Signature time=17.880000 seconds using Chudnovsky Coordinate
Message Verifies
```

In the measurement of ECDSA with CC-P-ECC, Average execution time are 8.73, 8.67 and 17.18 for Key Generation, Signature and Verification, respectively.

5.2.2.4 Comparison of Execution Time with simulation of Serial Parallel Computation

In **TEST IV, V, VI**, Execution time of ECDSA is measured using simulation of Serial-Parallel computation using different coordinate systems. The comparison on **P-P-ECC**, **J-P-ECC** and **CC-P-ECC** are made in the **Table 5.2.2.1.** The result shows that ECDSA with CC-P-ECC achieves best performance among the three algorithms. It is faster than J-ECC by 17.8%, 17.8% and 17.2% on Key Generation, Signature and Verification. It is faster than P-ECC by 12.6%, 13.0% and 13.0% as well on Key Generation, Signature and Verification respectively.

Table 5.2.2.1 Running Time of ECDSA with P-P-ECC, J-P-ECC and CC-P-ECC

Coordinate Type	Time of Key Generation (Second)	Time of Signature (Seconds)	Time of Verification (Seconds)
J-P-ECC	10.62	10.57	20.76
P-P-ECC	9.99	9.97	19.74
CC-P-ECC	8.73	8.67	17.18

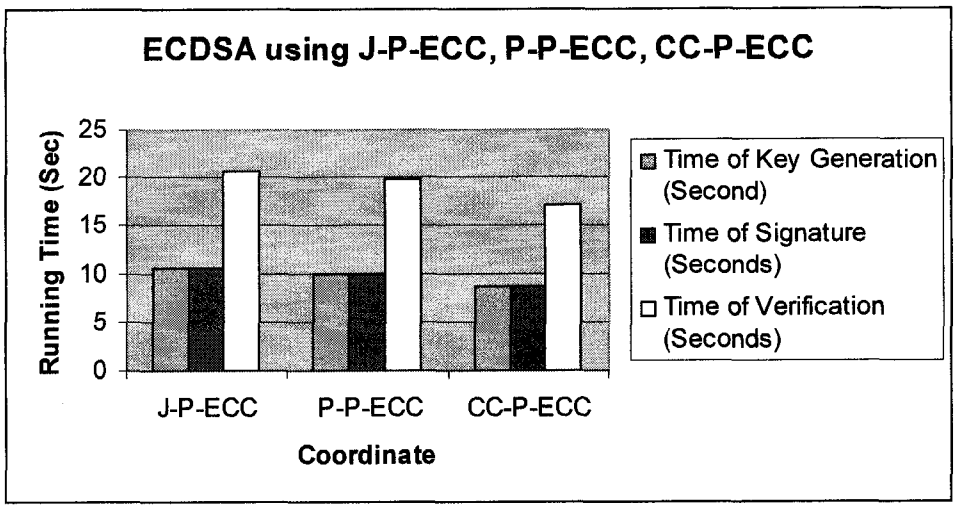


Figure 5.2.2 Comparison of ECDSA with P-P-ECC, J-P-ECC and CC-P-ECC

Summary

The experiments on ECDSA using a digital serial multiplier and Simulation of Serial Parallel computation using 3 digital serial multipliers are given in this Chapter. The comparison in both conditions is made. For ECC with a serial multiplier, Jacobian and Chudnovsky Coordinate gives almost same performance but both of them are better than Projective Coordinate System. In case of Serial-Parallel Computation, ECC with Chudnovsky Jacobian coordinate had the best performance among the three algorithms.

6 Conclusion and Future Work

As elliptic curve discrete logarithm problem (ECDSLP) makes elliptic curve cryptosystem to be noticed as an important alternative in the public key cryptosystem family. The Cryptosystems based on elliptic curve cryptography is going to be used widely for the application in the future [Certi04]. As a consequence, the methods that make its implementation on software or elliptic curve processors more practical and efficient are of importance. In this thesis, our goal is to find an efficient algorithm of elliptic curve cryptography that can be implemented using digital serial parallel multipliers.

6.1 Conclusion

The CC-P-ECC algorithm over $GF(2^n)$ using 3 digital serial parallel multiplier is proposed in this thesis. This new algorithm results in considerable reduction in execution time. While it exploits the inherent parallelism in the computation of doubling and addition of points over elliptic curve $GF(2^n)$, the new algorithm explores a new way that reduces rounds of field multiplications in point addition and doubling operation to save execution time. The evaluation of ECDSA is done by using a series of experiments. The result shows that the two proposed method (CC-P-ECC and P-P-ECC) have a considerable improvement over J-P-ECC, in terms of Serial-Parallel computation of ECC with digital serial parallel multipliers. It is also found that CC-P-ECC is the best algorithm out of the three.

In Chapter 4, the methods for point multiplications had been introduced. Binary method used in this thesis is the simplest among them. As a matter of fact, CC-P-ECC can also be tuned by using m-*ary* method, window method, or by signed *m-ary* window method described by blake[BSS99].

6.2 Future work

The **CC-P-ECC** algorithm proposed in this thesis requires 3 digital serial parallel multipliers in hardware. If we decrease the quantity of digital serial parallel multipliers, **CC-P-ECC** performance may not be the best.

Because point multiplication of elliptic curve cryptosystem is essential in public key calculation and key exchange, faster methods for Point Multiplication of elliptic curve cryptosystem may be devised in the future.

The key is to find new FASTER methods of point multiplication better than those available. It is the same problem as finding algorithms for solving "*shortest addition chain problem for integer*" defined in Chapter 4. With its special role in elliptic curve cryptography, exploration of faster methods is still an active and challenging research field.

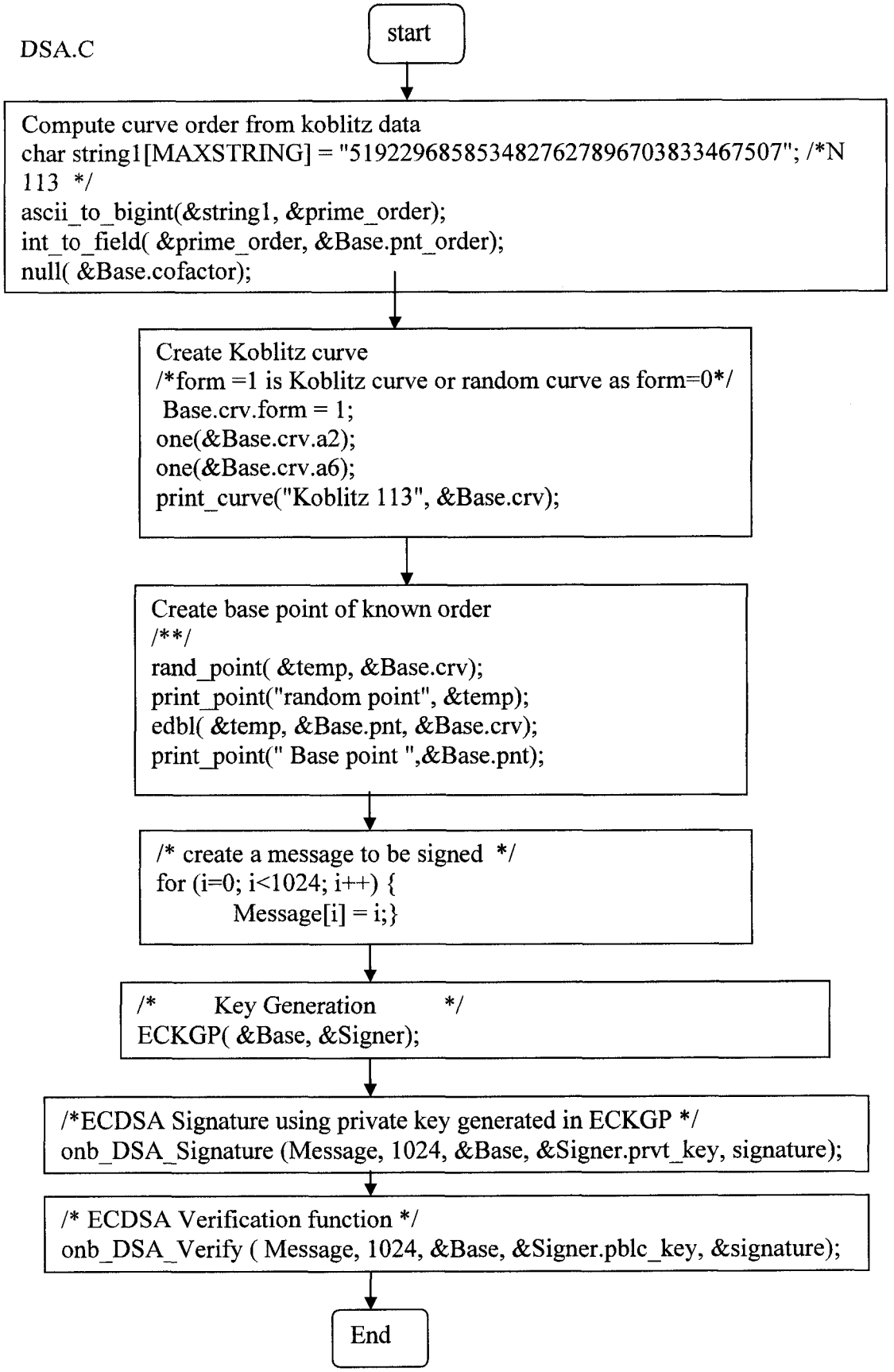
Appendix A Implementation of CC-P-ECC

The implementation of CC-P-ECC consists of several components mentioned in [Ros98a]. It uses some important files as follows:

DSA.c, onb_integer, elliptic.c

Major functions includes: ECDPK(), onb_DSA_signature(), onb_DSA_verify(), c_j_eliptic_mul(), c_j_esum(); c_j_edb(), parallel_mul(), opt_mul(); We will give the detail in the following section.

DSA.C



Specification

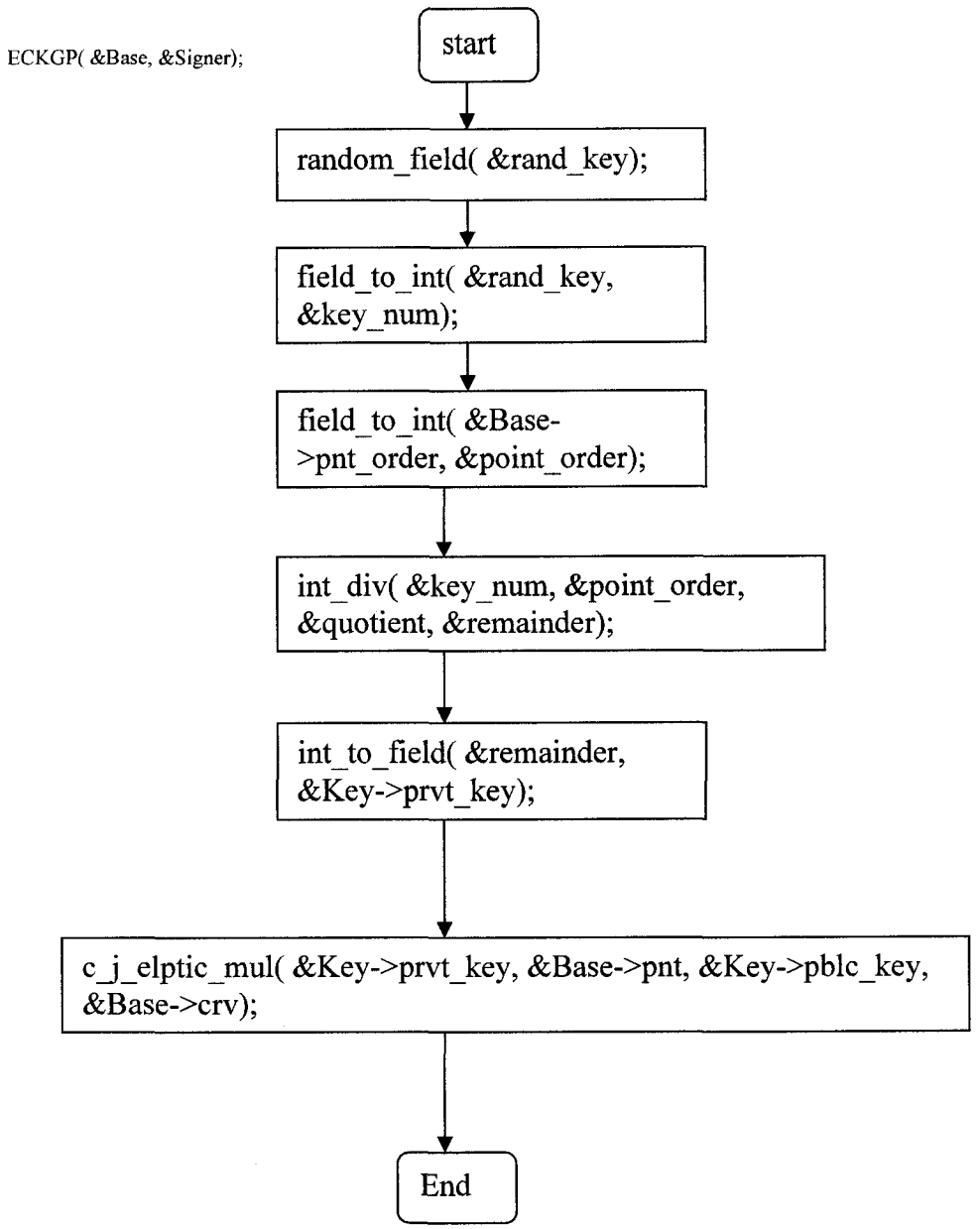
Program Name: DSA.C [Ros98a]

IEEE P1363 includes ECDSA as one of the standards for digital signatures. So we have used ECDSA module for testing our algorithm ECDSA module is to test signature of a hashed message using a private key and verify it with its public key in Elliptic Curve Cryptosystem.

This function presets elliptic curve as Koblitiz Curve over $GF(2^{113})$, Point Order is set as 5192296858534827627896703833467507. Cofactor =2.

Using the parameters above, it generates the random point in the range of specified curve. It then uses point doubling operation to generate a base point for Elliptic curve point operation .

ECKGP generates a pair of keys: private & public key. DSA signature uses a hashed message SHA-1(FIPS 180). The next step is to sign this hashed message using Signature function, onb_DSA_signature () and verify the encrypted message with its public key in onb_DSA_verify() function.



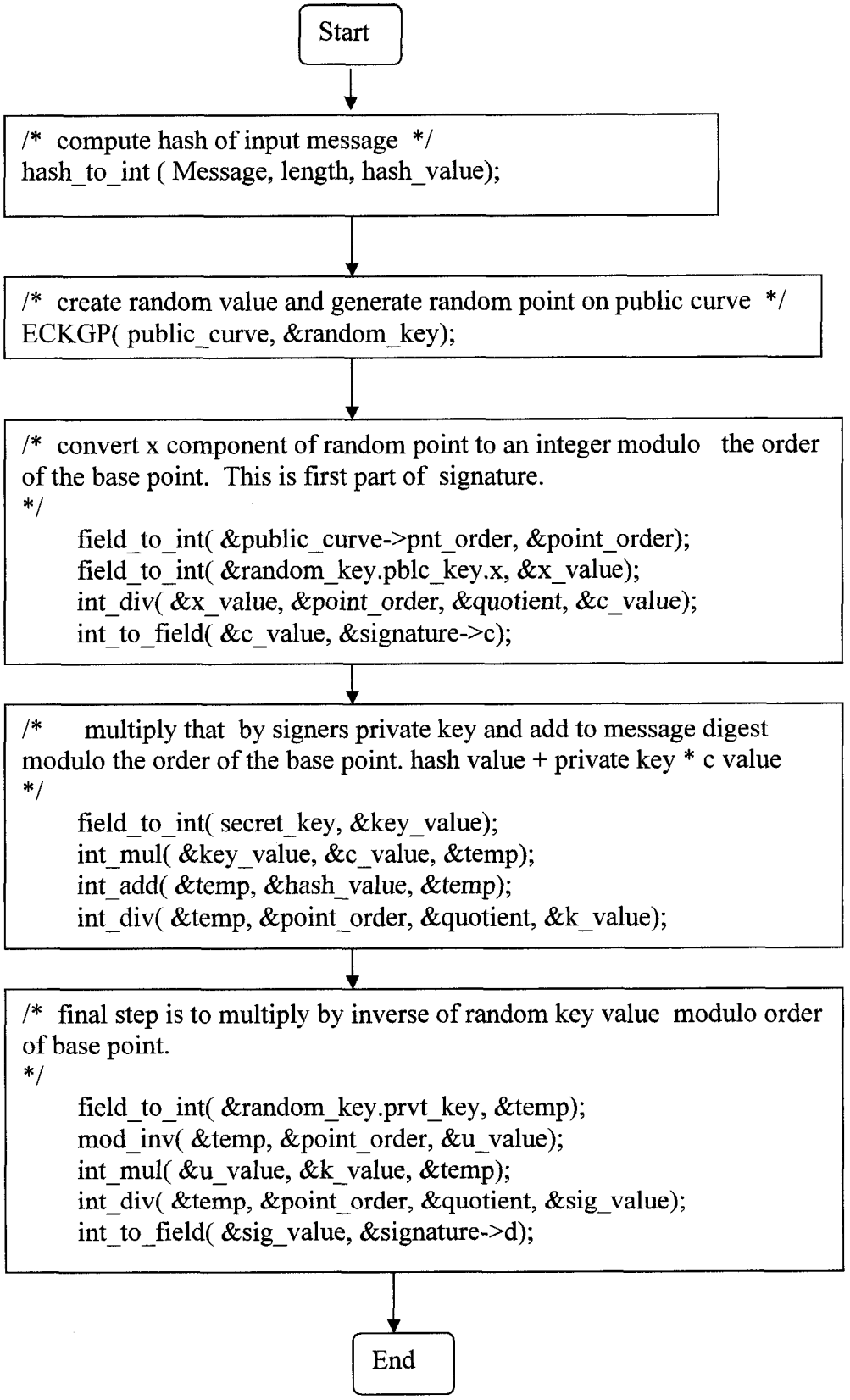
Specification

Program Function: ECKGP (&Base, &Signer) [Ros98a]

This is a part of onb_integer.c. This function uses random number modulo order of base point. The remainder is the signer's private key. In CC-P-ECC, the public key is the product of signer's private key and base point.

The 6^{th} module c_j_elptic_mul(&Key->prvt_key, &Base->pnt, &Key->pblc_key, &Base->crv) is designed as a part of the thesis work. This replaces eliptic_mul (k,p,n,curve) in ECKGP () of [Ros98a].

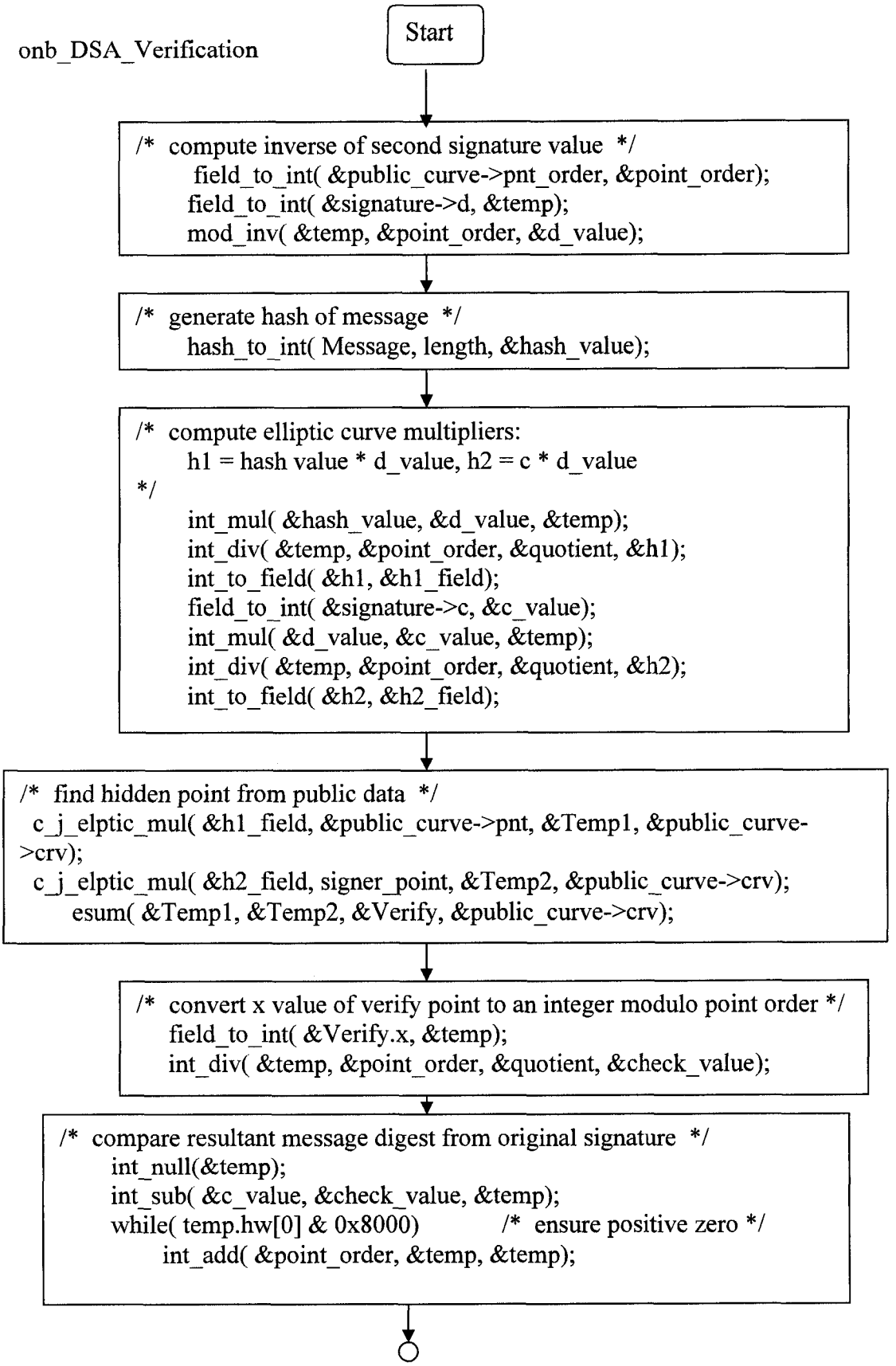
onb_DSA_signature()

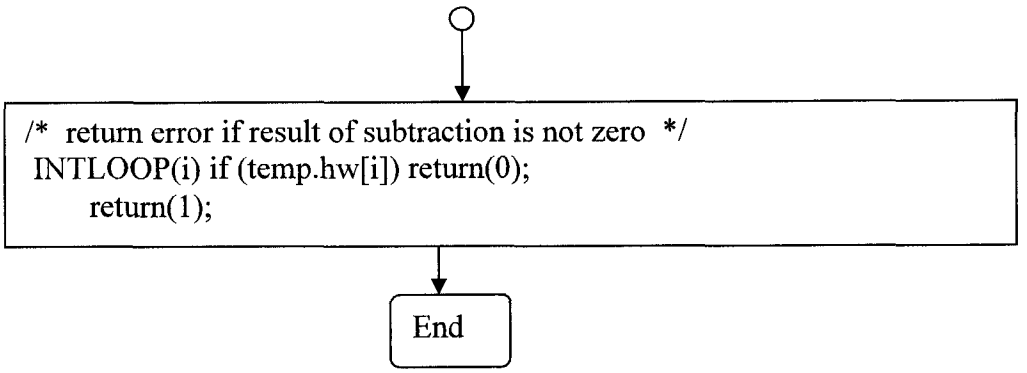


Specification
Function: onb_DSA_signature()[Ros98a]

This function is used for signing a message using ECC to generate a pair of value. It then passes the message to the other party. It is a part of program onb_integer.c [Ros98a]. In the second module, ECKGP has been modified as stated on page 6-76

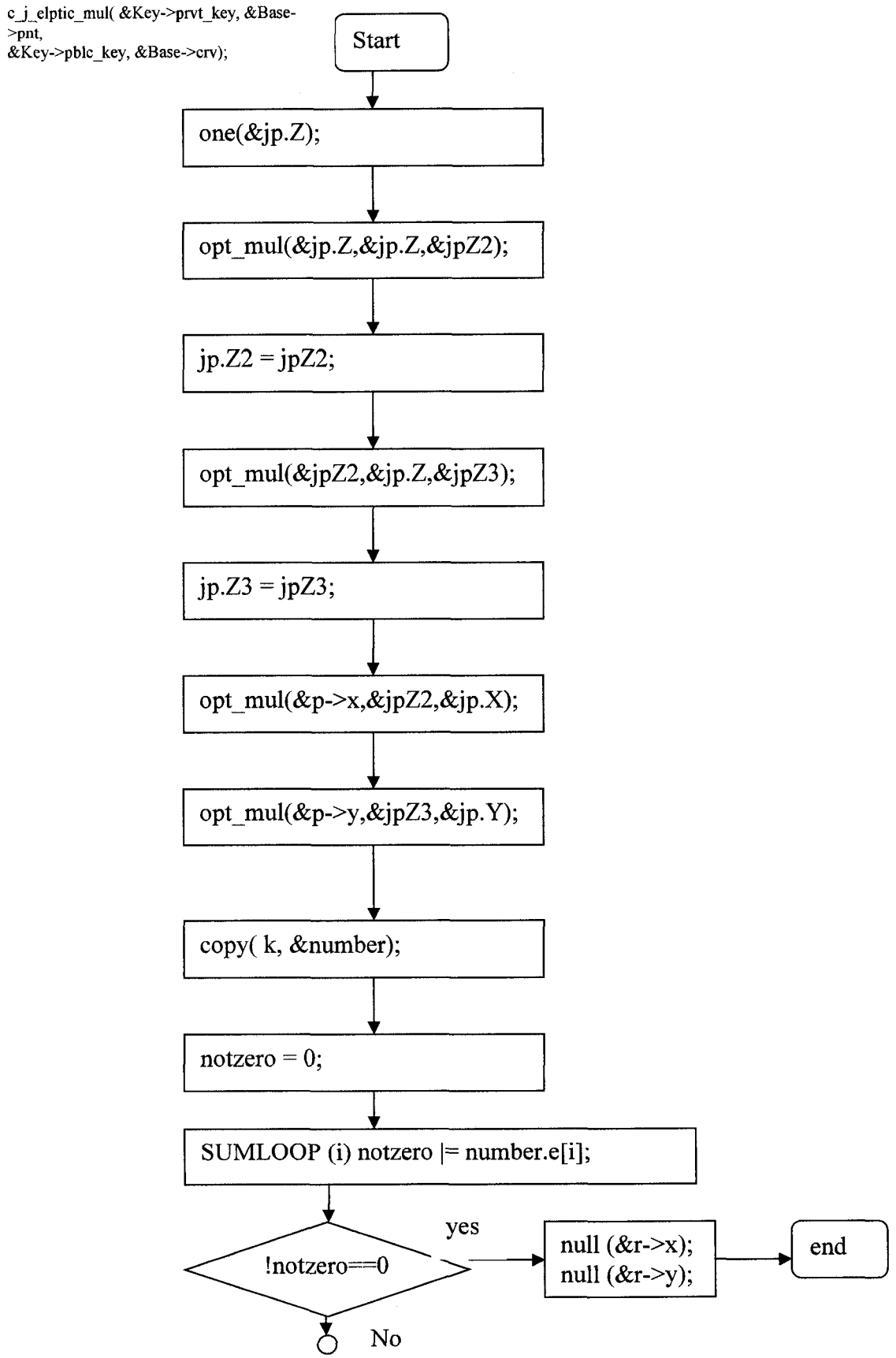
onb_DSA_Verification

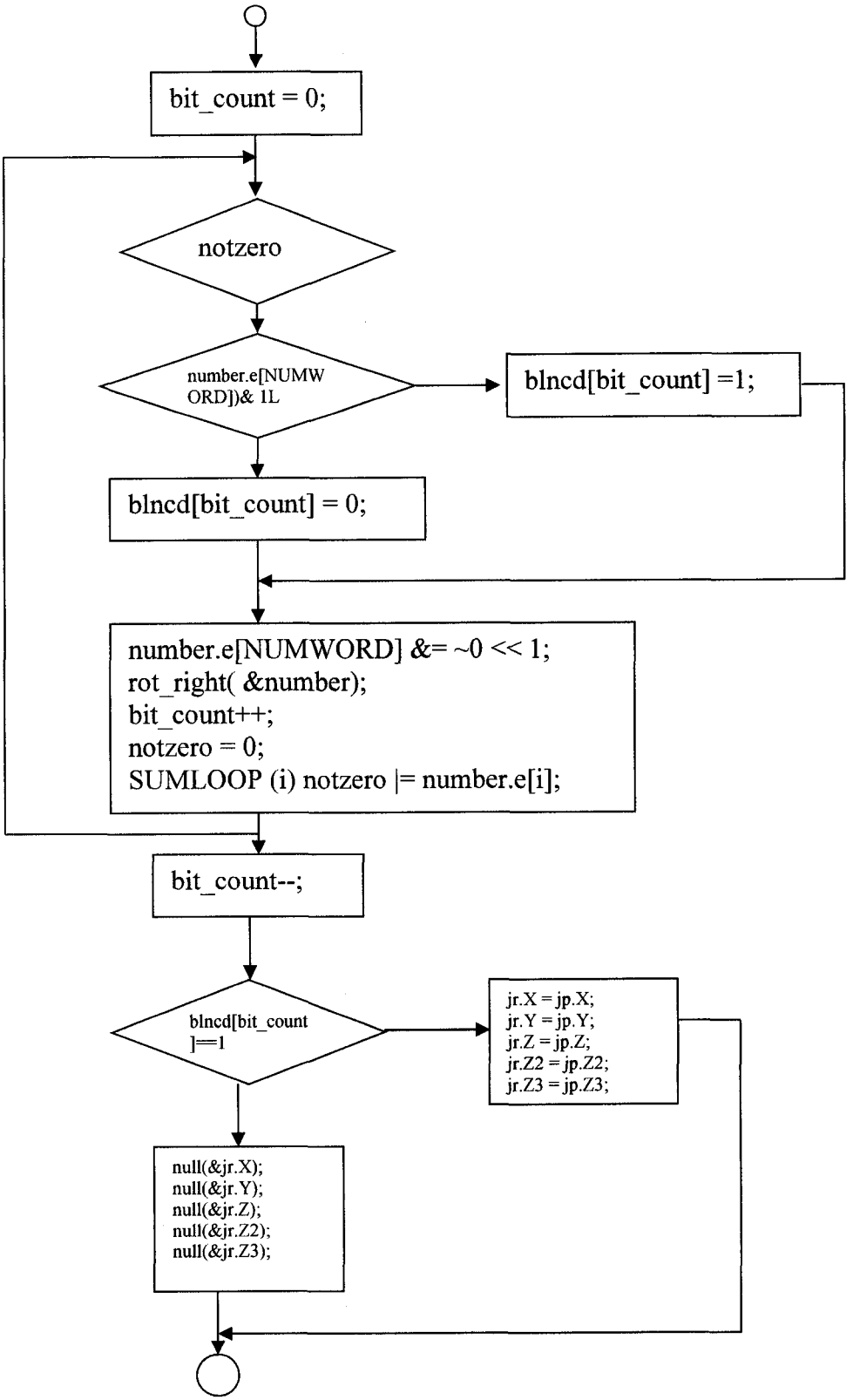


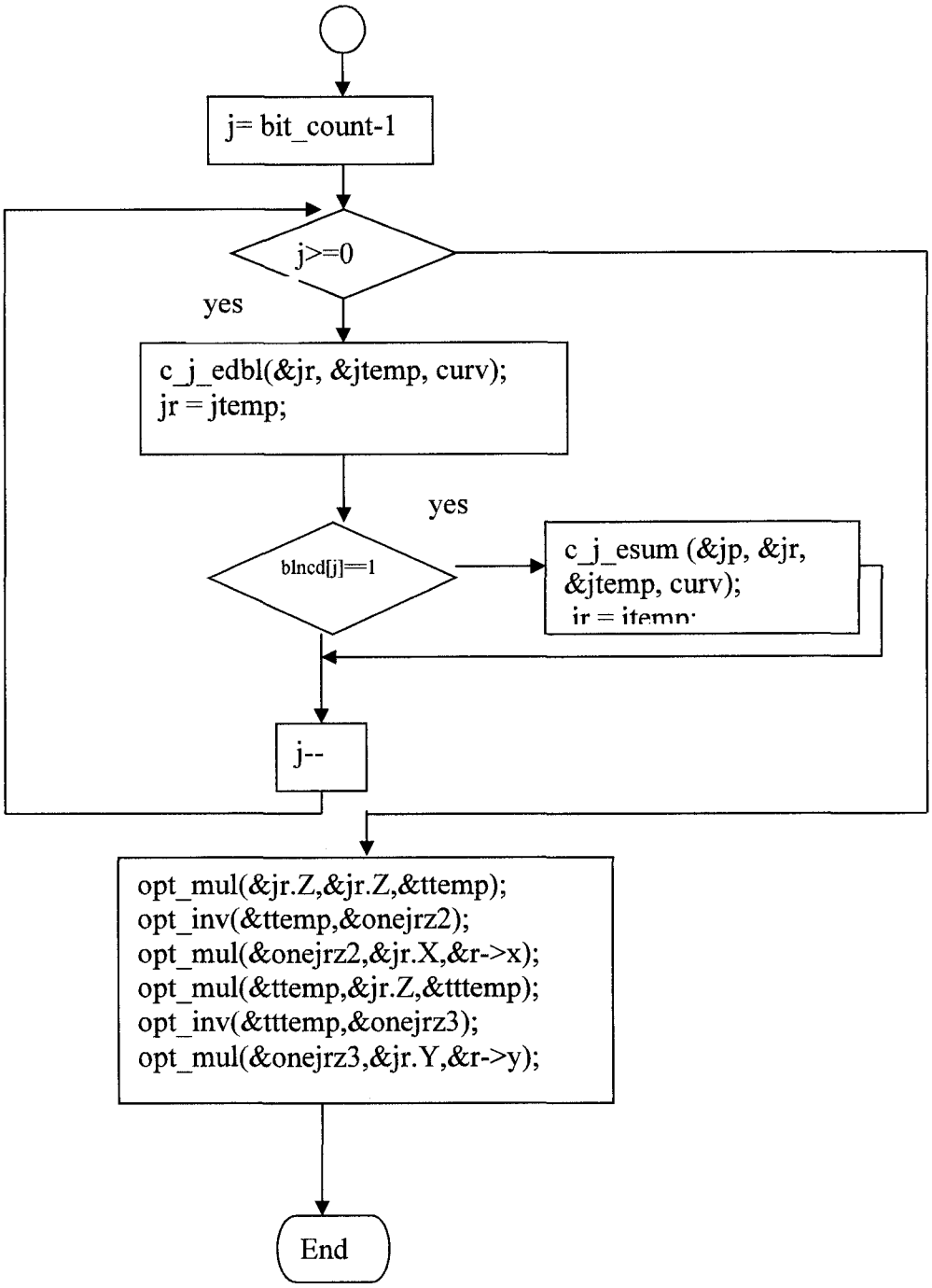


Specification
Function Name: int onb_DSA_Verify(Message, length, public_curve, signer_point, signature) [Ros98a]. This is a part of onb_integer.c.

This function receives signature from signer with Original Message and verifies it by using ECDSA verification procedure. The 4th module has been designed as a part of the thesis work. eliptic_mul() has been replaced by c_j_eliptic_mul() as specified on page:74



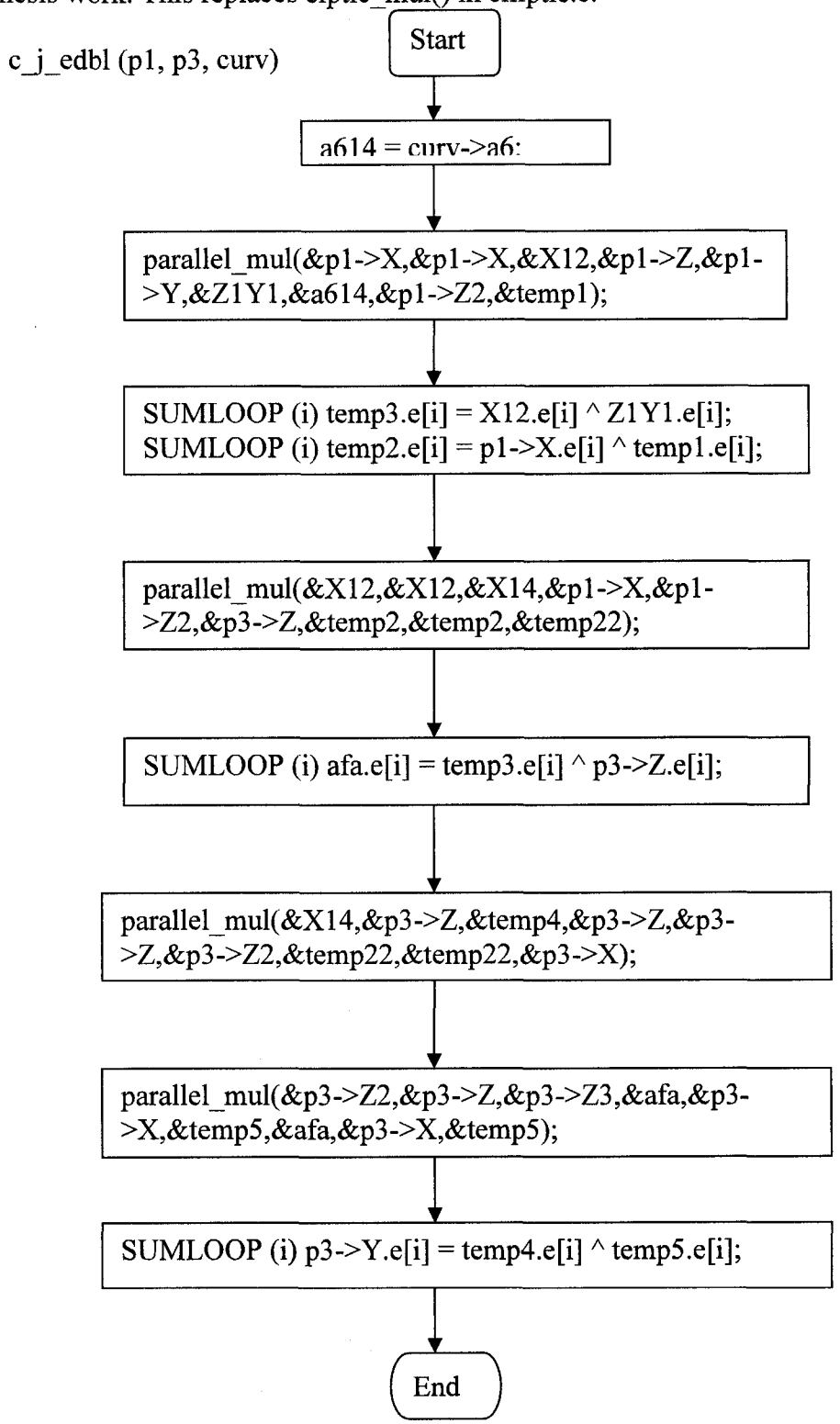




Specification
Function Name: c_j_eliptic_mul (&Key->prvt_key, &Base->pnt, &Key->pblc_key, &Base->crv)

This function implements point multiplication of elliptic curve using Chudnovsky Jacobian system with serial-parallel computation. The data in the range of order of base point is multiplied by base point to get another point. In this function, Chudnovsky coordinate system is used to represent a point on the elliptic curve. Binary method(refer

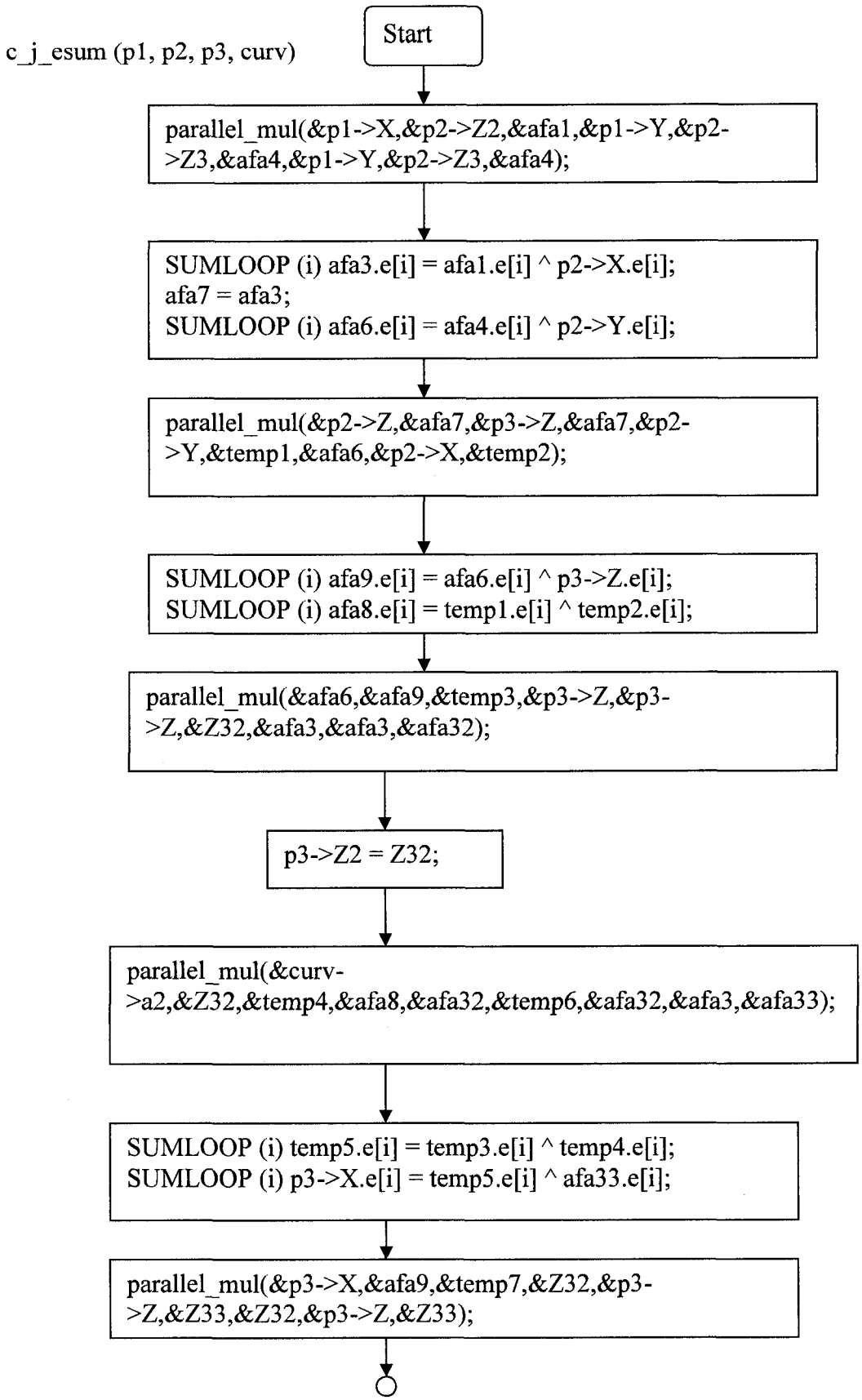
to section 3.2.1) is used as Point Multiplication. This has been designed as a part of the thesis work. This replaces elptic_mul() in elliptic.c.

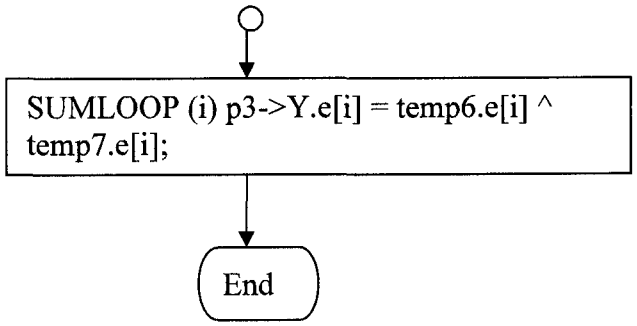


Specification

Function name: c_j_edbl (p1, p3, curv)

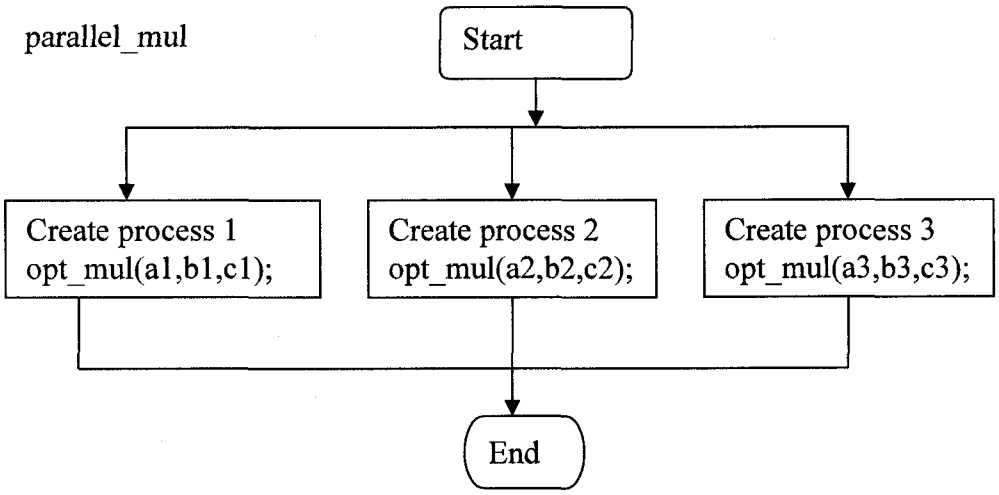
This function is used to calculate point doubling operation using serial-parallel computation. $p_3 = 2p_1$, Chudnovsky coordinate system is used to represent the point. It requires 4 round serial-parallel multiplication operations. This has been designed as a part of this thesis work. It replaces edbl(p1,p3,curv) in elliptic.c [Ros98a].
c_j_edbl calls parallel_mul four times. parallel_mul is described on page 80.





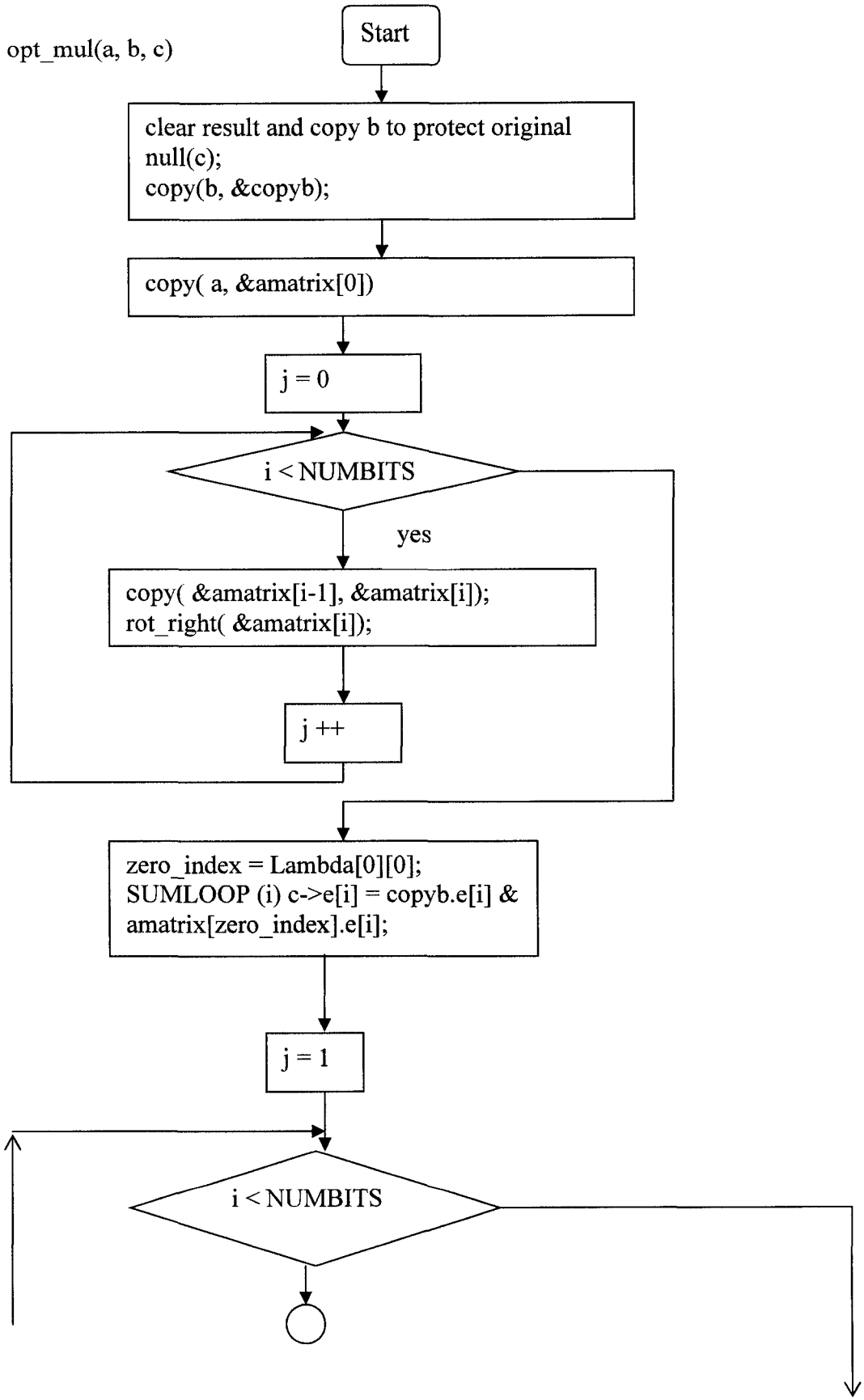
Specification
Function Name: c_j_esum (p1, p2, p3, curv)

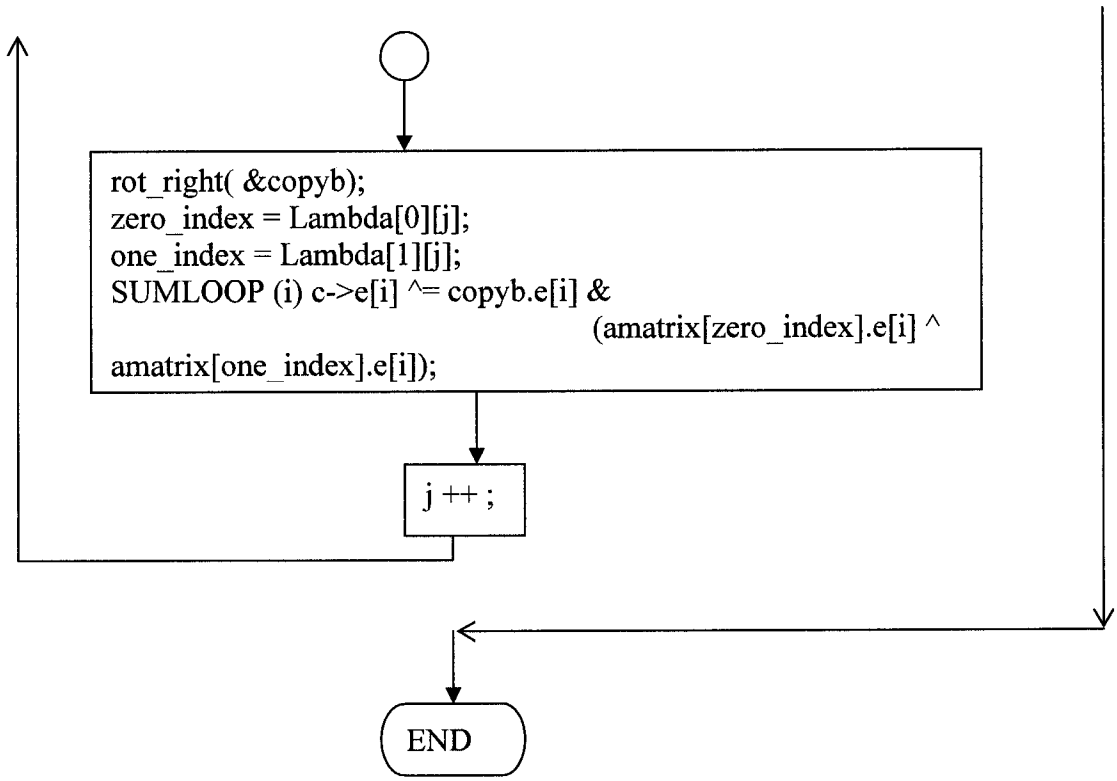
Point Addition is implemented in this function, $p_3=p_1+p_2$. Point representation uses Chudnovsky Jacobian Coordinate system. It requires 5 round serial-parallel multiplication operation. This has been designed as a part of this thesis work. It replaces esum(p1,p2,p3,curv) in elliptic.c [Ros98a].c_j_esum uses the module parallel_mul five times.



Specification
Function Name: parallel Mul(a1,b1,c1,a2,b2,c2,a3,b3,c3)

This function is programmed to parallel 3 processes to calculate 3 optimal normal basis multiplications at a time. Unix fork() is used to create process. Each of three opt_mul Store the resulting data in a file (called file1, file2 and file3 , respectively) so that the data can be called by the main process. This has been designed as a part of this thesis work.





Specification
Function Name: opt_mul()

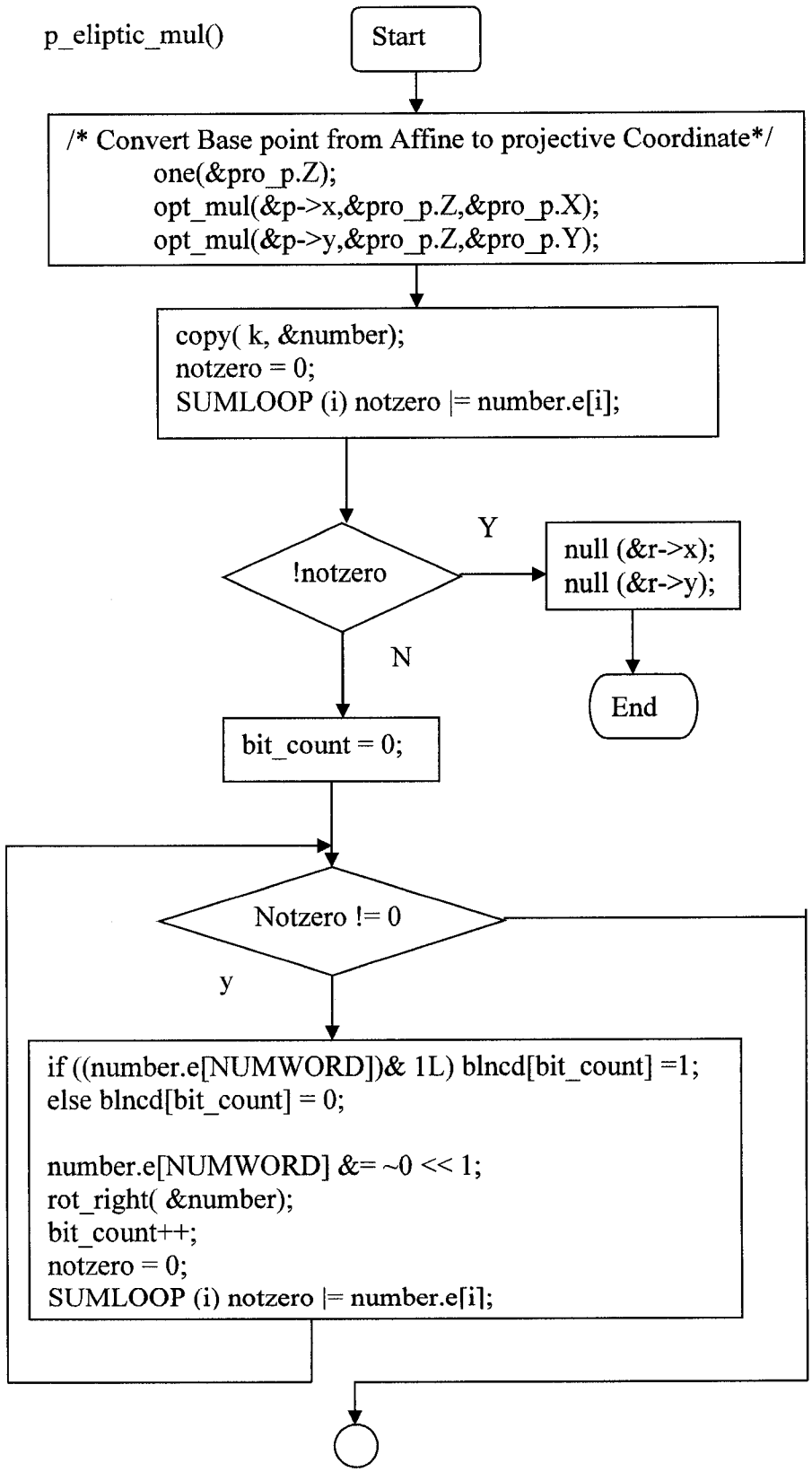
This is a part of onb.c [Ros98a]. It is used to calculate multiplication of optimal normal basis

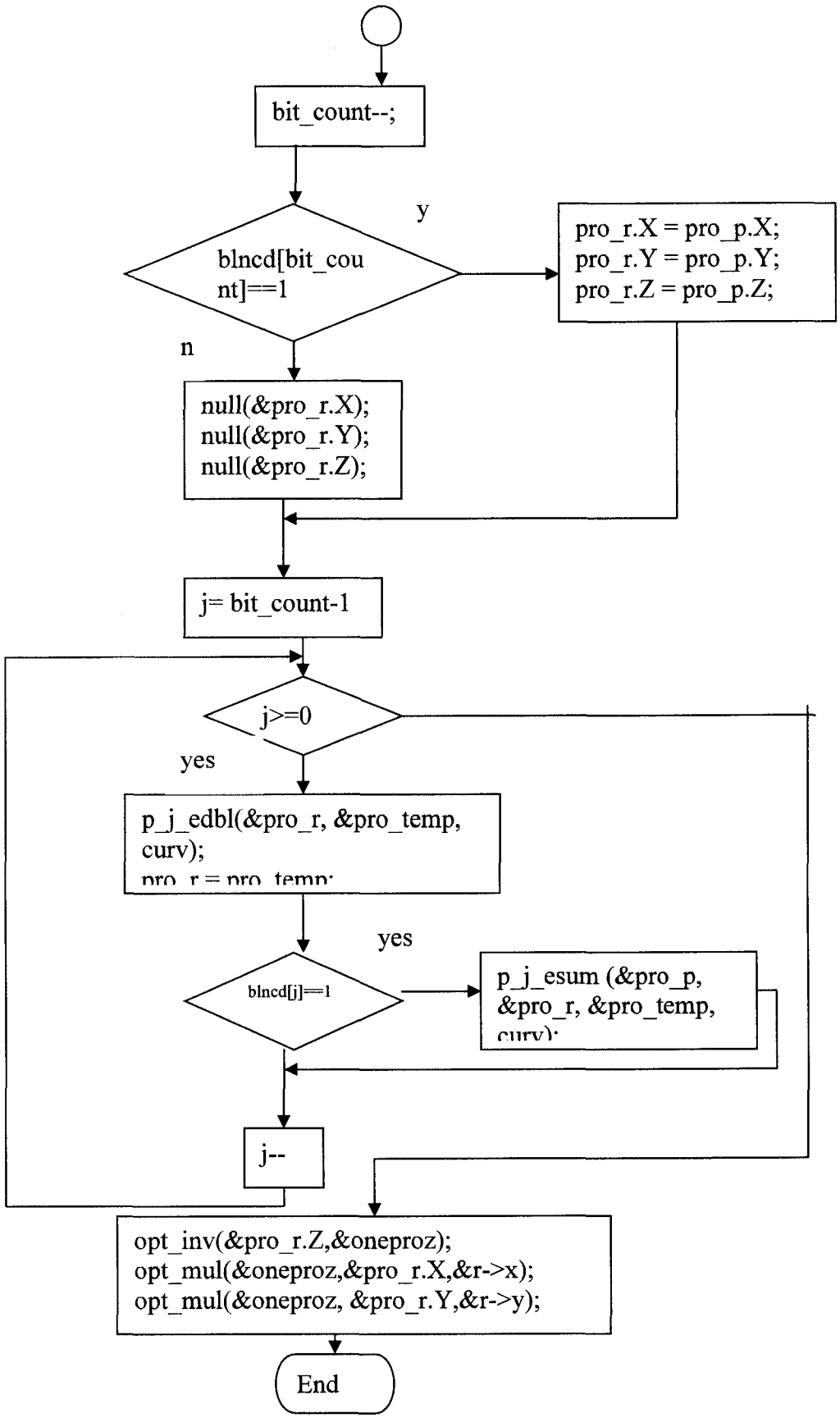
Appendix B Implementation of P-P-ECC/J-P-ECC

Most of the Program is the same as in CC-P-ECC. Only the following parts are different:

For P-P-ECC,
In onb_integer.c, ECGPK() and onb_DSA_verify() call the function p_elptic_mul(&Key->prvt_key, &Base->pnt, &Key->pblc_key, &Base->crv)

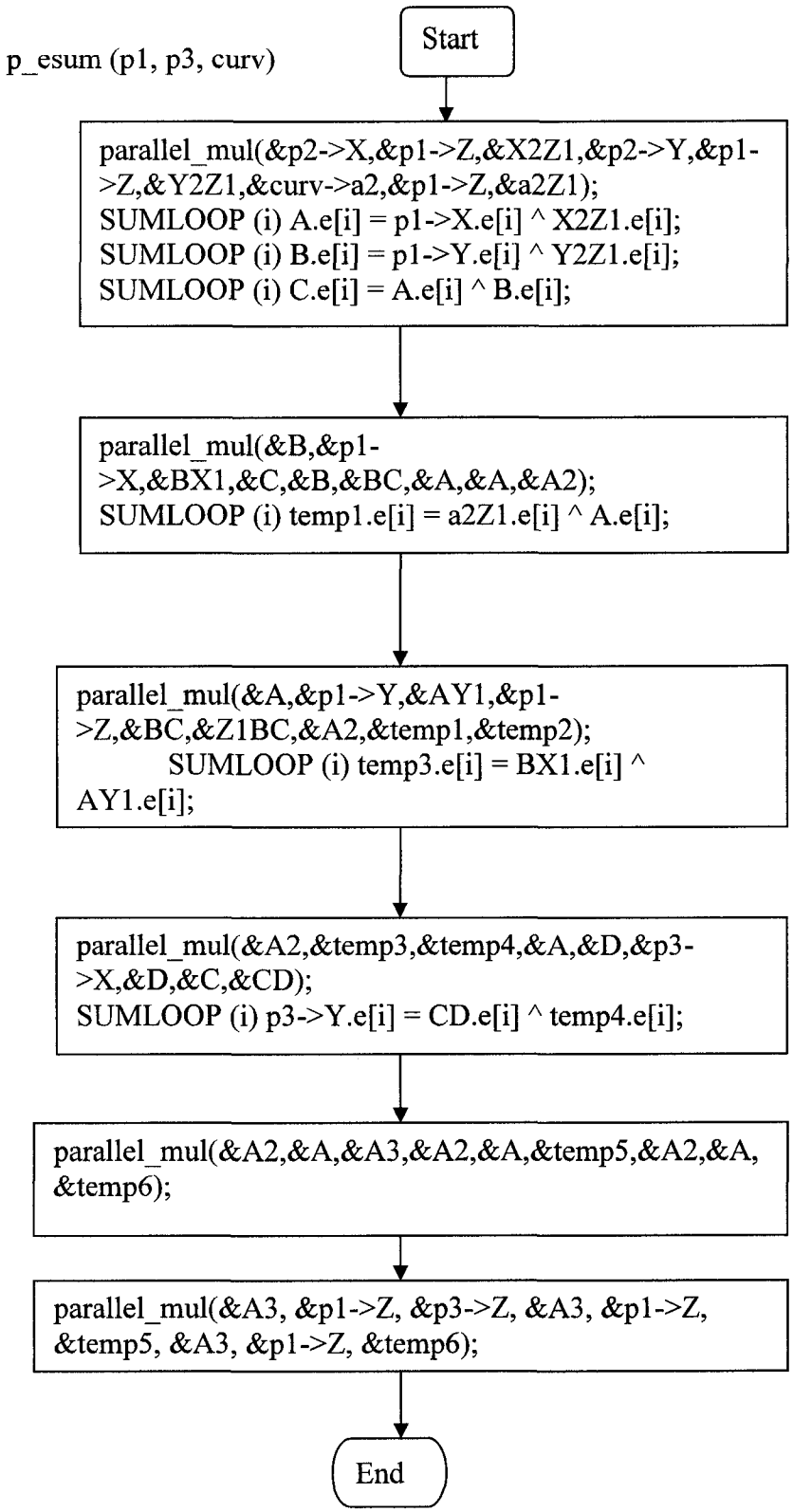
In eliptic.c , p_eliptic_mul(k,p,r,curv) call the functions p_esum(p1,p2,p3,curv) and p_dbl(p1,p3,curv)





Specification
Function Name: p_j_eliptic_mul (&Key->prvt_key, &Base->pnt, &Key->pblc_key, &Base->crv)

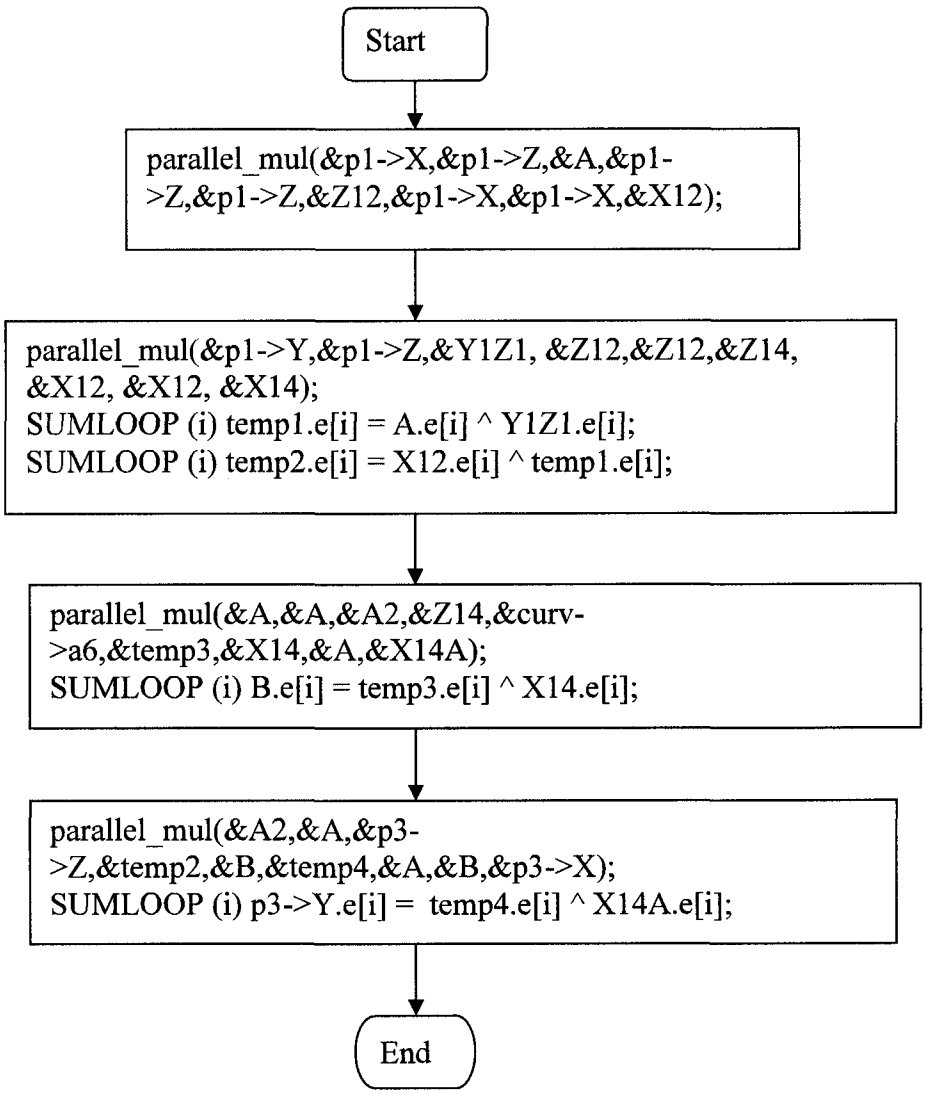
This function implements point multiplication of elliptic curve using Projective system with serial-parallel computation. The data in the range of order of base point is multiplied by base point to get another point. In this function, Projective coordinate system is used to represent a point on the elliptic curve. Binary method (refer to section 3.2.1) is used as Point Multiplication. This has been designed as a part of the thesis work. This replaces elptic_mul() in elliptic.c.



Specification

Function Name: p_j_esum (p1, p2, p3, curv)

Point Addition is implemented in this function, $p_3=p_1+p_2$. Point representation uses Projective Coordinate system. It requires 5 round serial-parallel multiplication operation. This has been designed as a part of this thesis work. It replaces esum(p1,p2,p3,curv) in elliptic.c [Ros98a].p_esum uses the module parallel_mul six times.

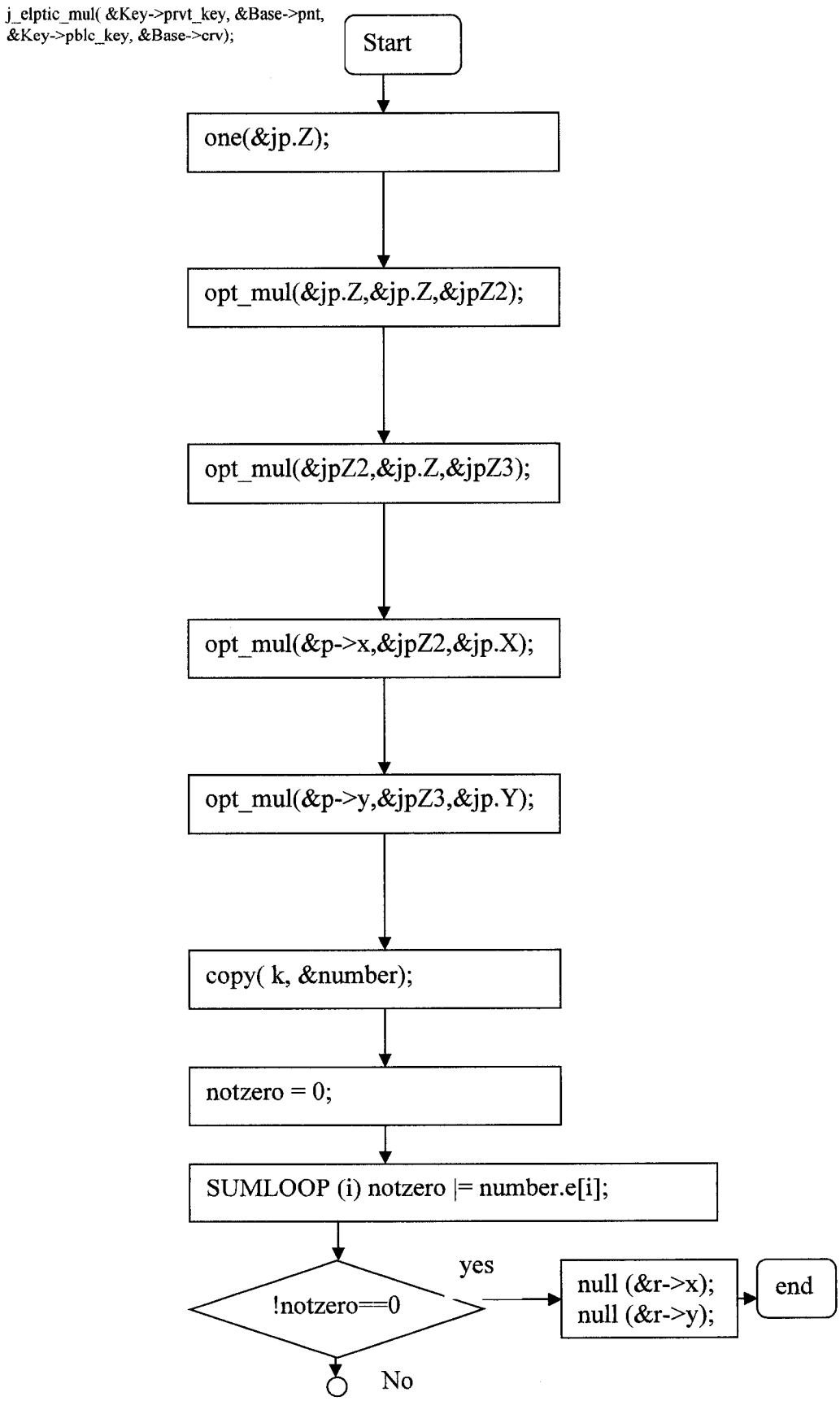


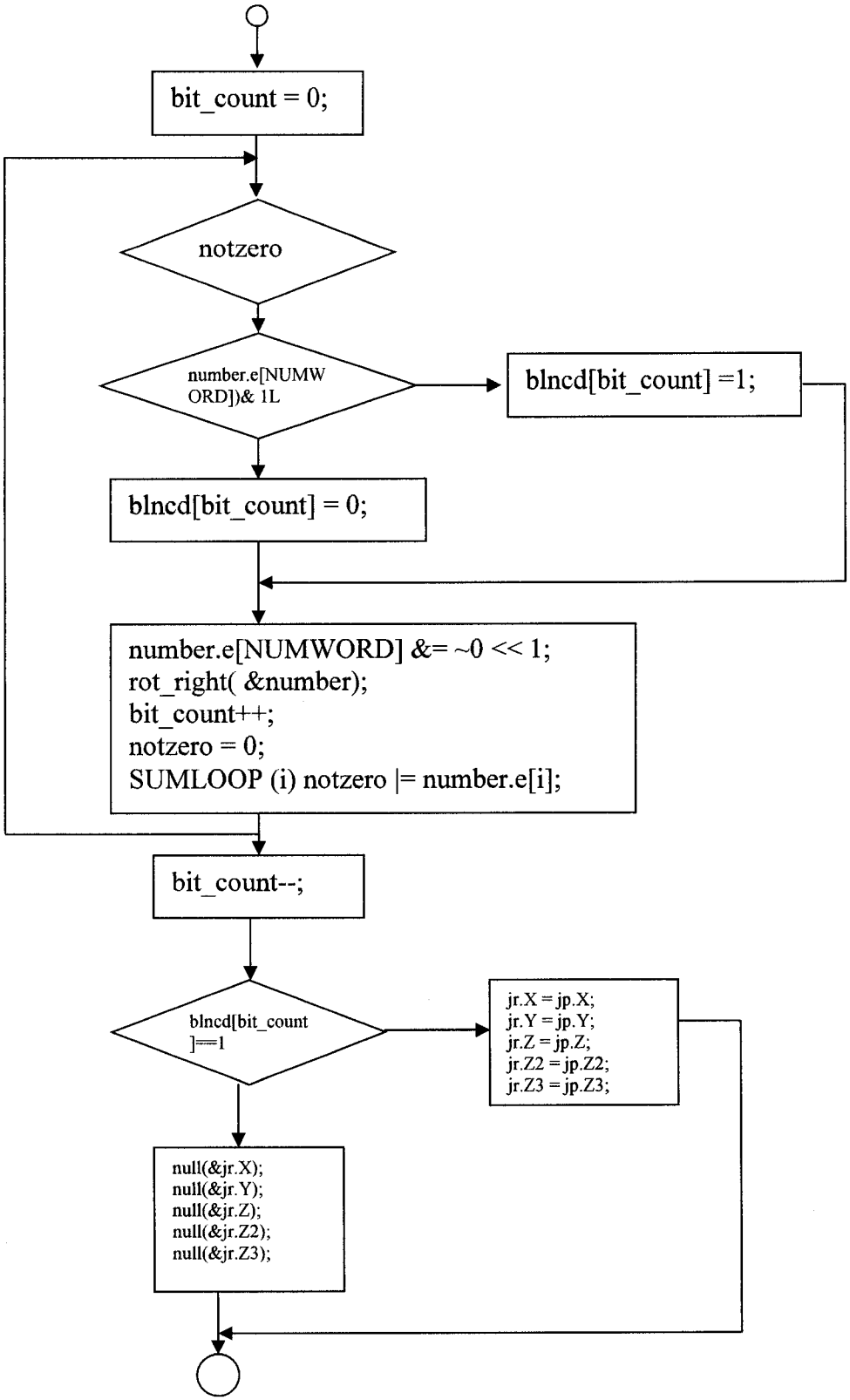
Specification
Function name: p_edbl (p1, p3, curv)

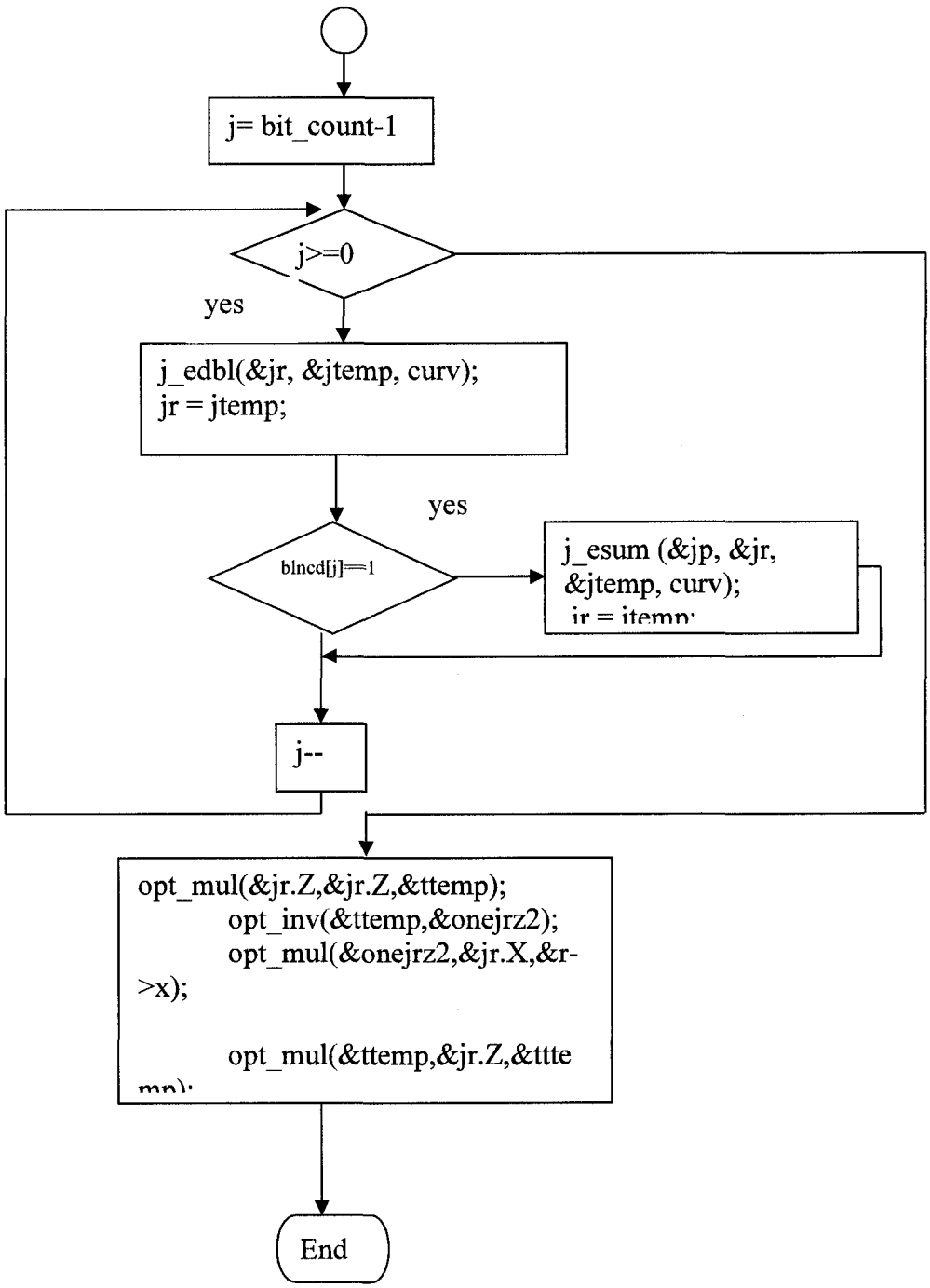
This function is used to calculate point doubling operation using serial-parallel computation. $p_3 = 2p_1$, Projective Coordinate system is used to represent the point. It requires 4 round serial-parallel multiplication operations. This has been designed as a part of this thesis work. It replaces edbl(p1,p3,curv) in elliptic.c [Ros98a].
p_edbl calls parallel_mul four times. parallel_mul is described on page 80.

For J-P-ECC,
In onb_integer.c, ECGPK() and onb_DSA_verify() call the function j_elptic_mul(&Key->prvt_key, &Base->pnt, &Key->pblc_key, &Base->crv)

In eliptic.c , j_eliptic_mul(k,p,r,curv) call the functions j_esum(p1,p2,p3,curv) and j_dbl(p1,p3,curv)



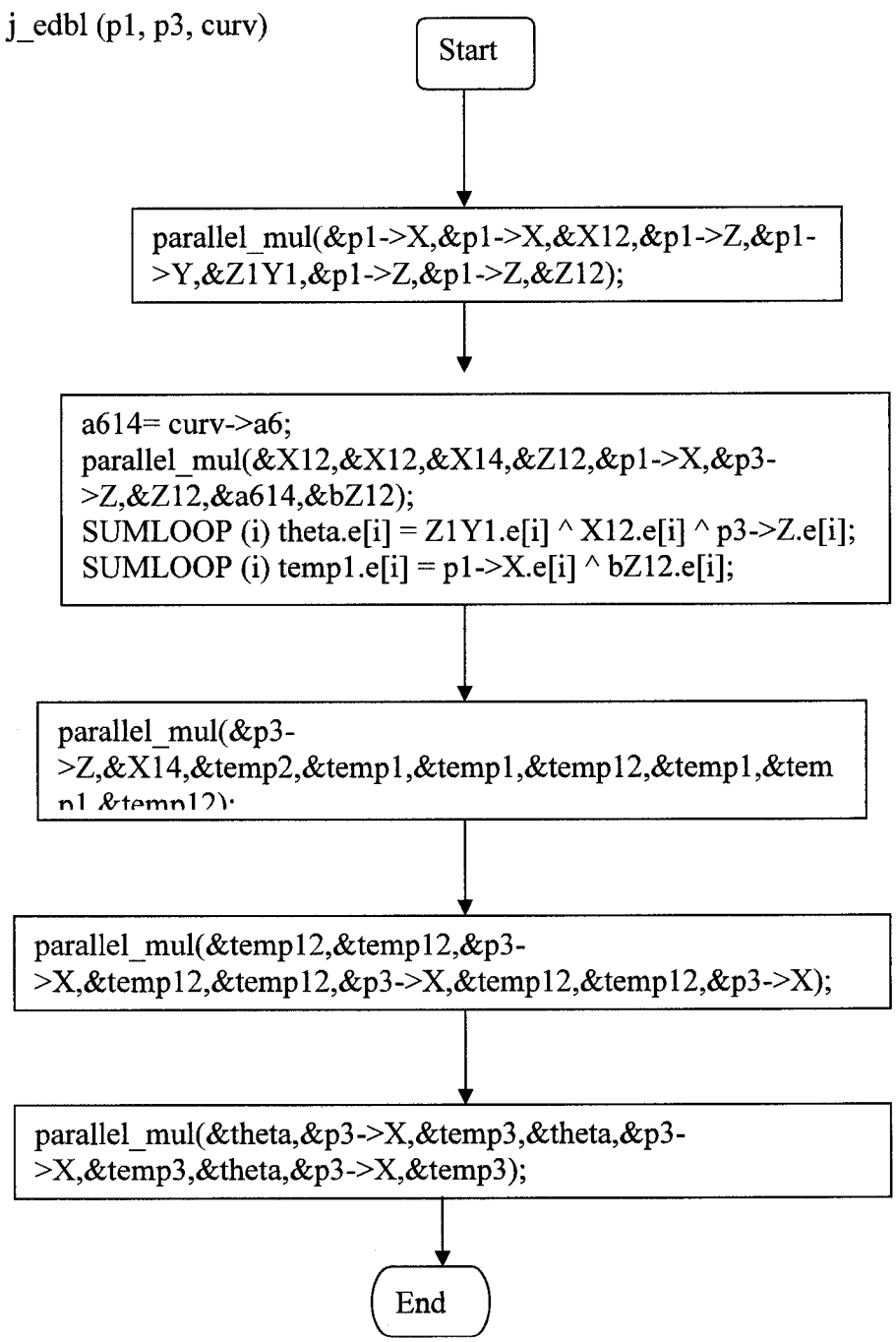




Specification
Function Name: j_eliptic_mul (&Key->prvt_key, &Base->pnt, &Key->pblc_key, &Base->crv)

This function implements point multiplication of elliptic curve using Jacobian system with serial-parallel computation. The data in the range of order of base point is multiplied by base point to get another point. In this function, Jacobian coordinate system is used to represent a point on the elliptic curve. Binary method (refer to section 3.2.1) is used as

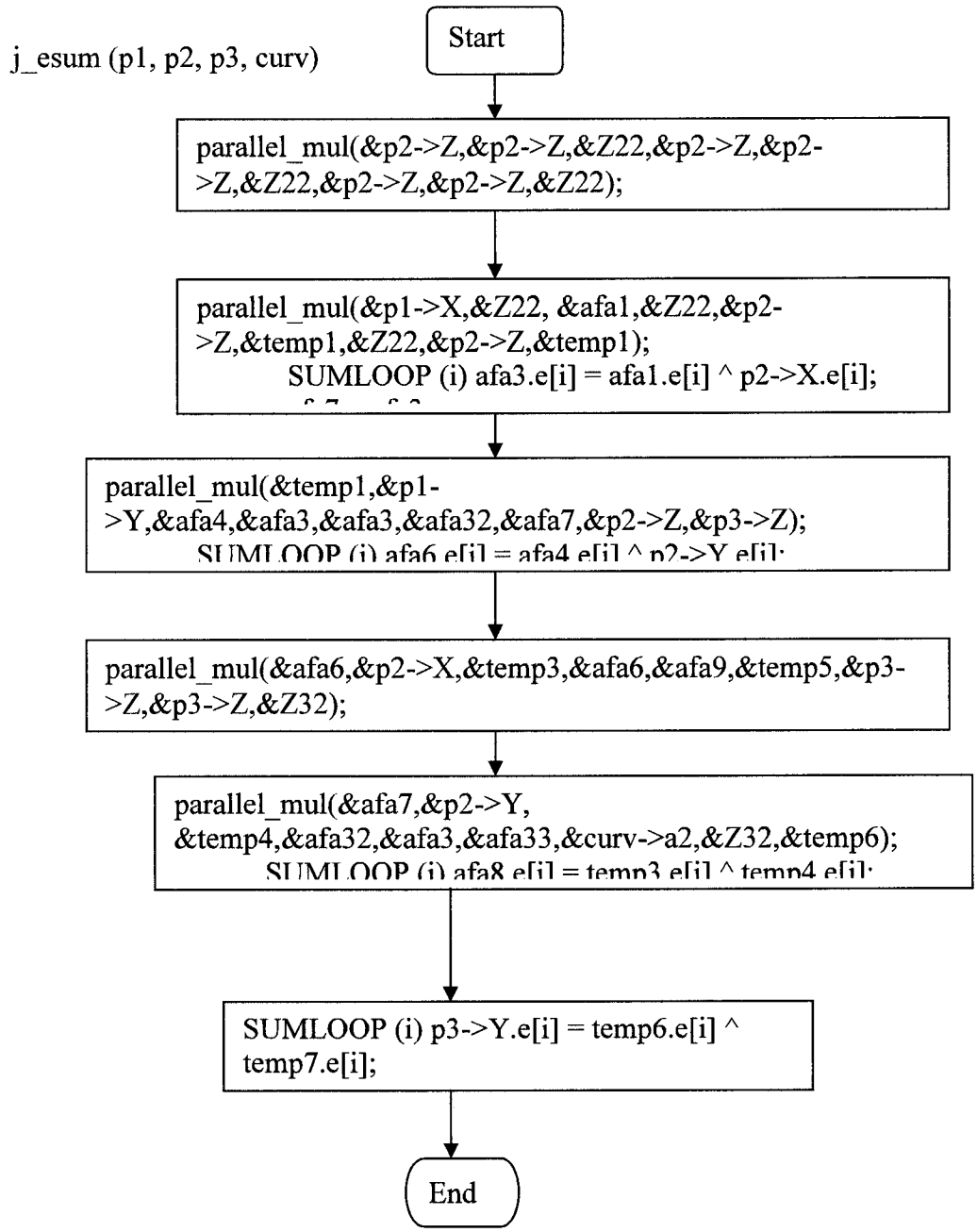
Point Multiplication. This has been designed as a part of the thesis work. This replaces elptic_mul() in elliptic.c.



Specification
Function name: j_edbl (p1, p3, curv)

This function is used to calculate point doubling operation using serial-parallel computation. $p_3 = 2p_1$, Jacobian coordinate system is used to represent the point. It

requires 4 round serial-parallel multiplication operations. This has been designed as a part of this thesis work. It replaces edbl(p1,p3,curv) in elliptic.c [Ros98a].
j_edbl calls parallel_mul six times. parallel_mul is described on page 80.



Specification
Function Name: j_esum (p1, p2, p3, curv)

Point Addition is implemented in this function, $p_3=p_1+p_2$. Point representation uses Jacobian Coordinate system. It requires 7 round serial-parallel multiplication operation. This has been designed as a part of this thesis work. It replaces esum(p1,p2,p3,curv) in elliptic.c [Ros98a].j_esum uses the module parallel_mul seven times.

Appendix C DSA with Message and Hash Message

Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

Key Generation time=0.240000 seconds using affine Coordinate
Signer's secret key : 1988 296d3d8c a8d08ecf 91ff45bf
Signers public key
x : 282b a9e578b1 bcd0319 24bdb6fc
y : 2950 906bb03b 415e5ad8 4bd2c559

Message is:
Slabs of concrete and metal came crashing down from the ceiling onto a seated waiting area at about 7 a.m. Sunday (0500 GMT/1 a.m. ET).

Part of the raised terminal structure then collapsed onto airport service vehicles underneath.

The collapse left a hole 50 meters (yards) by 30 meters in the long, tunnel-like building.

"It's like a scene after an earthquake," one firefighter said.

Officials said there was nothing to indicate a terrorist attack.

Hundreds of rescue workers rushed to the scene, and temporary hospitals were set up on the tarmac and inside the terminal.

Interior Minister Dominique de Villepin, inspecting the site, said there were five confirmed dead and "perhaps six." Officials earlier said six people were killed.

Hubert de Mesnil, director general of Paris airports, said all the dead were likely passengers, The Associated Press reported.

De Mesnil said there was "absolutely nothing" in the past to indicate a structural problem.

"It's the structure that gave way, the structure itself," h

message_digest is:
1017204939-172124312545655915 2-1119359275800469993

Signature take time=0.240000 seconds using affine Coordinate
first component of signature : c0c2 421b5284 57235894 76f9862a
second component of signature : 8a4f c6e37485 ca1ad3a8 536fcacd
Message is:
Slabs of concrete and metal came crashing down from the ceiling onto a seated waiting area at about 7 a.m. Sunday (0500 GMT/1 a.m. ET).

Part of the raised terminal structure then collapsed onto airport service vehicles underneath.

The collapse left a hole 50 meters (yards) by 30 meters in the long, tunnel-like building.

"It's like a scene after an earthquake," one firefighter said.

Officials said there was nothing to indicate a terrorist attack.

Hundreds of rescue workers rushed to the scene, and temporary hospitals were set up on the tarmac and inside the terminal.

Interior Minister Dominique de Villepin, inspecting the site, said there were five confirmed dead and "perhaps six." Officials earlier said six people were killed.

Hubert de Mesnil, director general of Paris airports, said all the dead were likely passengers, The Associated Press reported.

De Mesnil said there was "absolutely nothing" in the past to indicate a structural problem.

"It's the structure that gave way, the structure itself," h

message_digest is:
1017204939-172124312545655915 2-1119359275800469993

Verify Signature time=0.490000 seconds using affine Coordinate
Message Verifies

Appendix D Sample output of Serial-Parallel Computation

The following output shows one bit operation within binary methods when executing Key Generation

Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

first multiplier and its parent using processes : 21335 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

second multiplier and its parent using processes : 21336 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point

x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

Third multiplier and its parent using processes : 21337 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

Third multiplier and its parent using processes : 21340 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

first multiplier and its parent using processes : 21338 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

second multiplier and its parent using processes : 21339 21334

, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

first multiplier and its parent using processes : 21341 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

second multiplier and its parent using processes : 21342 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

Third multiplier and its parent using processes : 21343 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

first multiplier and its parent using processes : 21344 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

second multiplier and its parent using processes : 21345 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

Third multiplier and its parent using processes : 21346 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

first multiplier and its parent using processes : 21347 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

second multiplier and its parent using processes : 21348 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

Third multiplier and its parent using processes : 21349 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

first multiplier and its parent using processes : 21350 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

second multiplier and its parent using processes : 21351 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

Third multiplier and its parent using processes : 21352 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

first multiplier and its parent using processes : 21353 21334
, Current Process 21334 Koblitz 113
form = 1

a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

second multiplier and its parent using processes : 21354 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

Third multiplier and its parent using processes : 21355 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

second multiplier and its parent using processes : 21357 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point

x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

first multiplier and its parent using processes : 21356 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

Third multiplier and its parent using processes : 21358 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

first multiplier and its parent using processes : 21359 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point

x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

second multiplier and its parent using processes : 21360 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

Third multiplier and its parent using processes : 21361 21334
, Current Process 21334 Koblitz 113
form = 1
a2 : 1ffff ffffffff ffffffff ffffffff
a6 : 1ffff ffffffff ffffffff ffffffff

random point
x : 7d17 b7423658 5f8dae64 d624d542
y : 100a4 823ec133 6f1d883c a95f43ac

Base point
x : 1f43c 6942b1a4 9aaaac4a b572fdbf
y : 10145 c084d629 96208f8e 44d9f291

Process=21334 Key Generation time=0.090000 seconds using Chudnovsky Coordinate
Signer's secret key : 1988 296d3d8c a8d08ecf 91ff45bf
Signers public key
x : 16c2 c3ed78d 8232c3a3 a510d008
y : 3673 989ddd2b 3ee2febe 9a4f0549

Appendix E Experimental Data

Table 5.2.2.1 Result of 44 times running time using P-ECC

Number	Key Generation	Signature	Verification
1	0.33	0.34	0.69
2	0.34	0.34	0.7
3	0.34	0.35	0.7
4	0.34	0.35	0.69
5	0.33	0.35	0.67
6	0.35	0.35	0.71
7	0.35	0.34	0.7
8	0.34	0.34	0.7
9	0.34	0.36	0.69
10	0.33	0.35	0.69
11	0.34	0.35	0.69
12	0.35	0.35	0.69
13	0.34	0.35	0.7
14	0.34	0.34	0.7
15	0.35	0.35	0.7
16	0.34	0.34	0.69
17	0.34	0.35	0.67
18	0.35	0.34	0.69
19	0.33	0.35	0.68
20	0.34	0.35	0.69
21	0.34	0.33	0.69
22	0.34	0.36	0.69
23	0.34	0.35	0.7
24	0.34	0.33	0.7
25	0.35	0.34	0.7
26	0.34	0.36	0.69
27	0.34	0.36	0.71
28	0.36	0.35	0.68
29	0.35	0.35	0.69
30	0.32	0.34	0.69
31	0.35	0.36	0.69
32	0.34	0.34	0.65
33	0.33	0.34	0.69
34	0.34	0.35	0.72
35	0.34	0.34	0.72
36	0.34	0.34	0.7
37	0.32	0.35	0.68
38	0.35	0.35	0.7
39	0.35	0.34	0.72
40	0.35	0.35	0.7
41	0.35	0.34	0.7

42	0.34	0.33	0.69
43	0.34	0.34	0.68
44	0.34	0.35	0.69

Table 5.2.2.2 Result of 44 times using J-ECC

Number	Key Generation	Signature	Verification
1	0.31	0.31	0.61
2	0.3	0.32	0.61
3	0.3	0.31	0.61
4	0.31	0.31	0.6
5	0.3	0.3	0.61
6	0.3	0.31	0.61
7	0.29	0.31	0.62
8	0.3	0.31	0.62
9	0.3	0.3	0.62
10	0.3	0.31	0.62
11	0.31	0.3	0.62
12	0.3	0.31	0.62
13	0.31	0.32	0.6
14	0.3	0.3	0.61
15	0.3	0.29	0.61
16	0.3	0.31	0.6
17	0.3	0.3	0.6
18	0.3	0.31	0.6
19	0.3	0.32	0.61
20	0.3	0.31	0.61
21	0.3	0.32	0.61
22	0.31	0.3	0.61
23	0.31	0.31	0.6
24	0.31	0.3	0.61
25	0.3	0.3	0.62
26	0.3	0.31	0.6
27	0.31	0.3	0.6
28	0.3	0.31	0.62
29	0.32	0.32	0.62
30	0.3	0.31	0.65
31	0.3	0.3	0.66
32	0.31	0.31	0.63
33	0.3	0.31	0.61
34	0.3	0.31	0.6
35	0.3	0.32	0.61
36	0.3	0.31	0.59
37	0.3	0.32	0.6

38	0.33	0.31	0.62
39	0.31	0.31	0.61
40	0.3	0.3	0.63
41	0.31	0.32	0.61
42	0.3	0.3	0.61
43	0.3	0.31	0.63
44	0.3	0.3	0.59

Table 5.2.2.3 Result with 44 times using CC-ECC

Number	Key Generation	Signature	Verification
1	0.3	0.32	0.62
2	0.3	0.3	0.59
3	0.3	0.33	0.64
4	0.31	0.31	0.6
5	0.3	0.3	0.63
6	0.3	0.32	0.61
7	0.3	0.31	0.61
8	0.29	0.3	0.6
9	0.31	0.32	0.61
10	0.31	0.31	0.62
11	0.31	0.31	0.61
12	0.31	0.3	0.61
13	0.3	0.3	0.61
14	0.3	0.29	0.62
15	0.32	0.3	0.6
16	0.31	0.3	0.62
17	0.31	0.31	0.62
18	0.31	0.3	0.62
19	0.3	0.31	0.62
20	0.31	0.3	0.62
21	0.3	0.31	0.62
22	0.31	0.31	0.62
23	0.32	0.33	0.6
24	0.3	0.31	0.62
25	0.31	0.31	0.6
26	0.3	0.31	0.64
27	0.3	0.32	0.61
28	0.3	0.3	0.61
29	0.3	0.29	0.6
30	0.29	0.3	0.61
31	0.3	0.3	0.61
32	0.3	0.3	0.61
33	0.31	0.31	0.6

34	0.3	0.3	0.61
35	0.31	0.31	0.62
36	0.31	0.31	0.6
37	0.31	0.32	0.62
38	0.3	0.3	0.61
39	0.3	0.32	0.63
40	0.31	0.32	0.62
41	0.3	0.31	0.6
42	0.3	0.31	0.62
43	0.3	0.32	0.62
44	0.3	0.31	0.61

Table 5.2.2.4 Result of running 44 times in ECDSA using P-P-ECC

Number	Key Generation (Seconds)	Signature (Seconds)	Verification (Seconds)
1	9.9	10.18	19.02
2	9.48	9.6	18.7
3	9.95	9.52	18.65
4	9.76	9.26	18.94
5	9.76	9.56	18.47
6	9.54	9.29	18.44
7	9.58	9.28	18.88
8	9.21	9.4	18.6
9	9.77	9.66	19.67
10	9.65	9.59	18.79
11	9.33	9.45	18.78
12	9.5	9.37	18.22
13	9.52	9.14	19.07
14	9.76	9.78	20.43
15	10.09	10.03	20.09
16	10.25	9.97	20.03
17	10.34	10.72	20.54
18	9.88	9.89	19.54
19	10.58	9.74	19.27
20	9.89	10.11	20.18
21	10.03	10.04	20.11
22	10.01	10.01	20.21
23	10.06	10.44	19.82
24	10.51	10.36	20.16
25	10.68	9.89	20.09
26	9.78	9.68	18.79
27	9.86	9.49	19.51
28	10.2	11.04	20.05
29	11.1	9.94	20.97

30	10.13	10.1	20.7
31	9.67	9.49	19.11
32	9.67	10.36	19.56
33	10.25	10.43	19.8
34	10.29	10.32	20.85
35	9.91	10.02	19.89
36	10.03	10.45	20.14
37	10.07	10.6	20.01
38	10.19	10.59	20.56
39	10.36	10.11	20.79
40	9.87	10.5	20.83
41	10.42	10.15	20.55
42	10.16	10.51	20.84
43	10.53	10.23	20.59
44	10.04	10.31	20.49

Table 5.2.2.5 Result of running time in ECDSA using J-P-ECC

Number	Key Generation (Seconds)	Signature (Seconds)	Verification (Seconds)
1	10.54	10.52	20.53
2	10.31	10.35	20.46
3	10.68	10.64	21.2
4	10.62	10.46	20.72
5	10.5	10.64	20.53
6	10.73	10.59	20.9
7	10.58	10.66	20.91
8	10.96	10.7	20.89
9	10.69	10.64	20.84
10	10.74	10.77	21.32
11	10.87	10.42	20.98
12	10.52	10.69	20.91
13	10.73	10.48	20.68
14	10.64	10.84	20.92
15	10.4	10.59	20.73
16	10.73	10.69	20.84
17	10.89	10.62	20.89
18	10.56	10.65	20.96
19	10.67	10.69	20.87
20	11.1	10.91	21.14
21	10.72	10.63	20.87
22	10.63	10.62	20.83
23	10.76	11.14	20.99
24	10.5	11.03	21.25
25	10.75	10.6	20.76

26	10.67	10.71	20.77
27	10.41	10.49	20.4
28	10.67	10.76	20.96
29	10.64	10.62	20.55
30	10.69	10.48	20.38
31	10.49	10.32	20.2
32	10.67	10.47	20.72
33	10.65	10.46	21.5
34	10.4	10.41	20.79
35	10.85	10.54	20.43
36	10.25	10.37	20.27
37	10.54	10.45	20.65
38	10.56	10.28	20.27
39	10.66	10.51	20.73
40	10.73	11.04	20.79
41	10.45	10.32	20.46
42	10.5	10.62	20.71
43	10.45	10.86	21.76
44	10.58	10.59	20.79
45	10.79	10.7	20.53
46	10.57	10.22	20.37
47	10.62	10.21	20.52
48	10.81	10.54	20.51
49	10.5	10.67	20.91
50	10.61	10.22	20.79
51	10.57	10.24	20.31
52	10.18	10.17	20.42

Table 5.2.2.6 Result of running time in ECDSA using CC-P-ECC

Number	Key Generation (seconds)	Signature (Seconds)	Verification (seconds)
1	8.77	8.67	16.81
2	8.72	8.38	16.84
3	8.79	8.69	17.36
4	8.85	9.14	17.49
5	9	8.8	17.77
6	8.78	8.89	17.84
7	8.91	9	16.87
8	8.88	8.72	17.61
9	8.95	8.34	17.26
10	8.63	8.64	17.2
11	8.52	8.58	17.43
12	8.36	8.29	17.03
13	8.68	8.69	17.16

14	8.73	8.86	17.18
15	8.77	8.53	16.86
16	8.44	8.75	16.8
17	8.58	8.44	16.67
18	8.51	8.67	17.25
19	8.79	8.61	16.91
20	8.94	8.77	17.3
21	8.9	8.83	17.25
22	8.94	8.8	17.28
23	8.71	8.75	17.01
24	8.6	8.49	16.87
25	8.53	8.53	16.94
26	8.74	8.65	16.69
27	8.68	8.58	17.13
28	8.71	8.63	17.22
29	8.73	8.57	17.01
30	9.04	8.72	17.46
31	8.89	8.51	17.49
32	8.63	8.62	16.89
33	9.04	8.73	17.2
34	8.88	8.92	17.8
35	8.95	8.64	17.46
36	8.66	8.6	17.29
37	8.62	8.57	16.94
38	8.57	8.48	16.89
39	8.44	8.61	17.04
40	8.59	8.56	16.73
41	8.73	8.51	17.41
42	8.71	8.48	17.34
43	8.46	8.57	17.03
44	8.54	8.86	16.82
45	8.72	8.58	17.19
46	8.65	8.5	16.71
47	8.77	9.17	17.27
48	8.78	8.72	17.59
49	8.85	8.8	17.54
50	8.52	8.74	17.41
51	8.87	8.71	17.56
52	8.84	8.84	17.22

References

[AM03]Adnan Abdul-Aziz Gutub and Mohammad K.Ibrahim, "*Power-Time Flexible Architecture For GF(2^k) Elliptic Curve Cryptosystem Computation*" , Proceedings of the 13th ACM Great Lakes Symposium on VLSI, 2003

[AM03a]Adnan Abdul-Aziz Gutub and Mohammad K.Ibrahim, "HIGH RADIX PARALLEL ARCHITECTURE FOR GF(P) ELLIPTIC CURVE PROCESSOR" IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003.

[AMR+02]Essame, AI-Daoud, Ramlan Mahmod, Mohammad Russhadan, Adem Killcman, "*A New Addition Formula for Elliptic Curves over GF(2^n)*", IEEE TRANSACTIONS ON COMPUTERS, VOL.51, NO.8, AUGUST 2002

[AMS95]Alfred Menezes, Minghua,Qu, Scott, Vanstone , "Memorandum IEEE P1363, Part 6: Elliptic Curve System" Aug 22, 1995.
http://citeseer.nj.nec.com/menezes95elliptic.html

[AMV93]G.B.Agnew,R.C.Mullin and S.A.Vanstone. "*An implementation of elliptic curve cryptosystem over* $F_{2^{155}}$ ". IEEE Transactions on Selected Areas in Communications, 11:804-813,1993

[BSS99]I. Blake, G. Seroussi, N. Smart, "Elliptic Curves in Cryptography", pp:60-72. Cambridge, U.K: Cambridge Univ. Press, 1999.

[CC86] D.V.Chudnovsky and G.V. Chudnovsky "*Sequences of numbers generated by addition in formal groups and new primality and factorization tests*" Advanced in Applied math., 7 (1986), 385-434

[Certi04]http://www.certicom.com/index.php?action=company,press_archive&view=283

[CMO97]H.Cohen, A.Miyali and T.Ono, "*Efficient elliptic curve exponentiation*", Advanced in cryptography-Proceddings of ICICS'97, Lecture Notes in Computer Science, 1334(1997), Springer-Verlag, 282-290

[CMO98]Henri Cohen,Atsuko Miyaji, Takatoshi Ono, *"Efficient elliptic curve exponentiation using mixed coordinates"*

[FIPS186-2] "DIGITAL SIGNATURE STANDARD(DSSS)", FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION 2000 January 27

[HR83]M.Hellman and J.Reyneri. Fast computation of discrete logarithms in *GF(q)*. In advances in Crytpology: Proceedings of Crypto'82. Plenum Press, 1983

[IEEE1363]*IEEE Standard 1363, IEEE Standard Specifications For Public-Key Cryprography,1999*

[IT88]T.Itoh and S.Tsujii. "*A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases*". Info. And Comput., 78(3):171-177,1988

[Ko87]Neal Koblitz. "*Elliptic curve cryptosystem*". Mathematics of Computation/American Mathematical Society, 48(177): 203-209, 1987

[Kob87a]N.Koblitz. "*A Course in Number Theory and Cryptography*", Springer-Verlag, 1987

[knu81]A.W.Kunth."The Art of Computer Programming, 2-Semi-numerial Algorithms".Addison-wesley, 2nd edition, 1981.

[LH94]K.-Y.Law and L.C.K.Hui. "Efficiency of SS(l) square-and-multiply exponentiation algorithms".Electronics Letters, 30, 2115-2116, 1994.

[LN94]R.Lidl and H.Niederreiter. "*Intorduction to Finite Fields and Their Applications*". Cambridge University Press, 1993

[Men93]A.J.Menezes. "*Elliptic Curve Public Key Cryptosystems*", Kluwer Academic Publisher, 1993

[Mi86]Victor S.Miller. "*Use of Elliptic Curves in Cryptography* " Advance in Cryptology – CRYPTO'85, LNCS 218, pp.417-426, 1986

[MOV97]A.J.Menezes,P.C van Oorschot, and S.A.Vanstone, *Handbook of Applied Cryptography*. Boca Raton,FL:CRC Press,1997

[MOVW89]R.C.Mullin, I.M.Onyszchuk, S.A.Vanstone, and R.M.Wilson."*Optimal normal bases in GF(p^n)*". Discrete Applied Mathematics, 22:149-161,1988/89

[Odl84]A.Odlyzko. "*Discrete logarithms in finite fields and their cryptographic significance*". In advances in cryptology Eurocrypt'84, pages 224-314. Springer-Verlag, 1984

[OW99]P. van Oorschot and M. Wiener, "Parallel collision search with cryptanalytic applications", *Journal of Cryptology* 12 (1999), 1-28.

[PH78] S. Pohlig and M. Hellman, "*An improved algorithm for computing logarithms over GF(p) and its cryptographic significance*", IEEE Transactions on Information Theory 24 (1978), 106-110.

[PID92]J.SternP.Ivey, S.Walker and S.Davidson. "*Anultra-high speed public key encryption processor*". In Proceedings of the IEEE Custom Integrated Circiuts Conference,pages 19.6.1-19.6.4, 1992

[Pol75]J.M.Pollard. "*A MonteCrlo method for factorization*".BIT,15(3):331-334,1975

[Pollard78] J. Pollard, "Monte Carlo methods for index computation (mod p)", *Mathematics of Computation* 32 (1978), 918-924.

[Rei87]H.Reisel. "*Prime Numbers and Computer Methods for Factorization*", 2nd Edtion. Birkhauser,1987

[Ros98a]M.Rosing. "*Implementing Elliptic curve Cryptography*". Manning, 1998.

[RSA78]R.Rivest, A.Shamir and L.Adlemann. "*A method for obtaining digital signatures and public-key cryptosystems*. Communication of the ACM, 21:120-126, 1978

[SOOS95]R.Schroeppl, H.Orman, S.O'Mally and O.Spatscheck. "*Fast key exchange with elliptic curve systems*". Communication of the ACM, 21:120-126, 1978.

[Wi99]William Stallings, *CRYPTOGRAPHY AND NETWORK SECURITY: Principle and Practice, SECOND EDITION.*

VITA AUCTORIS

NAME:	Xiaoguang Wang
PLACE OF BIRTH	P.R. China
YEAR OF BIRTH	1969
EDUCATION	The 57th High School, Tianjin, P.R. China 1984 - 1987
	China University of GeoSciences, WuHan, HuBei 1987 – 1991 B.Sc.
	University of Windsor, Windsor, Ontario 2002-2004 M. Sc.