Computational Cost Analysis of Elliptic Curve Arithmetic

Serap Atay[1], Ahmet Koltuksuz[2], Hüseyin Hışıl[3], Şaban Eren[4]

[1,2,3] *Izmir Institute of Technology, College of Engineering, Dept. of Computer Engineering, Gülbahçe, Urla, 35430 Izmir, Turkey*
{serapatay, ahmetkoltuksuz, huseyinhisil}@iyte.edu.tr

[4]*Ege University, College of Engineering, Dept. of Computer Engineering, Bornova, Izmir, Turkey*
saban.eren@ege.edu.tr

Abstract

Elliptic curves are proposed for the asymmetrical cryptography by Neal Koblitz and Victor Miller in 1986 separately. Elliptic curve cryptography (ECC) is utilized by hardware embedded solutions on mobile equipments and smart cards after 2000. Currently, software implementation of ECC faces the computational speed problem. One of the proposed solutions is to do the arithmetic operations on different Euclidean coordinate systems. This paper concentrates on the research of this technique and delineates the performance results of the implementation of the aforementioned technique on the different cryptographic libraries such as CRYMPIX, GMP and MIRACL.

1. Introduction

The efficiency values of a software application such as running time and space utilization become very critical when the usability of an application is evaluated. This can be even more critical if a cryptographic protocol is used to establish a personal identification entity or to perform an encryption/decryption for a financial transaction with mobile equipment or to utilize a smart card in an unsecured platform such as Internet. The applications should establish the service in an acceptable time with the limited resources. However, the contemporary mobile equipments and smart cards have limits on computational power, main memory and communication bandwidth. In the mean time, the Internet is an open platform and Moor's law [8] is valid for the malicious application programs and persons. Therefore, the cryptographic solution should provide the higher security level with low computational power and shorter key lengths.

Today, the elliptic curve cryptography provides all these requirements with hardware embedded implementations. The application specific circuits (ASICs) or field programmable gate area logic circuits (FPGAs) are preferred in hardware embedded implementations. In ASIC implementation, design phase is expensive, if FPGA is used, the design is cheaper but the production of many products is expensive. The software implementation of elliptic curve cryptography counters these requirements, except the computational speed with the minimal costs. The computational speed problem is due to expensive arithmetic operations on elliptic curves such as calculation of multiplicative inverses and multiplication operations.

This study concentrates on the reduction methodologies of computational cost of arithmetic operations for the elliptic curve cryptography in software platform. The rest of this paper is organized as follows: Section 2 details the different coordinate systems as well as elliptic curve point addition arithmetic on those coordinate systems. The arithmetic of elliptic curves and the computational costs regarding different coordinate systems are also covered in this section.

The performance issue of the arithmetic operations on elliptic curves for different coordinate systems is evaluated in section 3. In this section three different cryptographic libraries, CRYMPIX, GMP and MIRACL are used for the defined operations and, those three libraries are compared and contrasted against one another. Section 4 summarizes the findings and the contributions on the topics given above.

2. Elliptic Curve Point Addition Arithmetic in Different Coordinate Systems

2.1. Efficiency Measurements

The point addition arithmetic is applied on two and three dimensional coordinate systems. The computational cost of each arithmetic operation should be taken into consideration in order to compare the efficiency of algorithms in different coordinate systems. The efficiency is measured as the computational cost, which is in terms of elapsed time. The types of arithmetic operations are listed from high to low computational cost as depicted in Fig. 2.1. The measured units in Fig. 2.1 are as follows:

- Inversion (I) is the multiplicative inverse in modular arithmetic. It has the highest computational cost and one inversion is approximately equals nineteen times of the cost of multiplication cost and denoted as $1I \approx 19M$.
- Multiplication (M) has a lower cost than inversion; therefore all inversions should be converted either to multiplication or to addition.
- Addition (A) and subtraction (S) have the lowest cost, therefore omitted.

Although it is not directly shown in Figure 2.1 there is yet another existing operation named as reduction. The reduction operation is necessary after each arithmetic operation since the results of all arithmetic operations should be guaranteed to be over the selected prime field. Such as if the Montgomery reduction technique is used; then the computational cost of one multiplication and reduction gets to be lower than standard multiplication and reduction operations. [3]

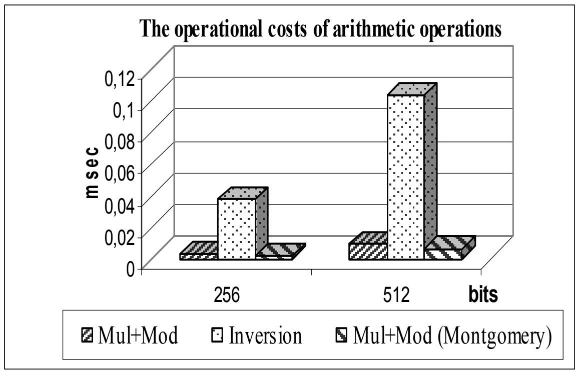

Figure 2.1 The operational costs of arithmetic operations.

2.2. Using Different Coordinate Systems

The NIST curves for all coordinate systems were employed in this study [9]. Point is defined by only two dimensions in the Affine coordinate system, it has the simplest form and requires the lowest bandwidth, and therefore it is preferred for communication between any two parties [4]. However, modular inversion is required for point addition and doubling rendering the Affine coordinate system highly inefficient. Other coordinate systems require at least one extra value to represent a point but do not require the use of modular inversion.

Figure 2.2 represents the computational cost of NIST curves while summarizing the addition and doubling operations and their respective computational costs in terms of time spent in different coordinate systems. The studied systems are Affine, Projective, Jacobian, Modified Jacobian and of Chudnovsky [1] [2] [3] [4] [6].

The Jacobian coordinate has faster doubling but slower addition than Projective coordinates. The Modified Jacobian and Chudnovsky Jacobian have the same arithmetical operations with that of Jacobian. In other words, the Modified Jacobian and Chudnovsky Jacobian are the variants of the Jacobian coordinate system [4].

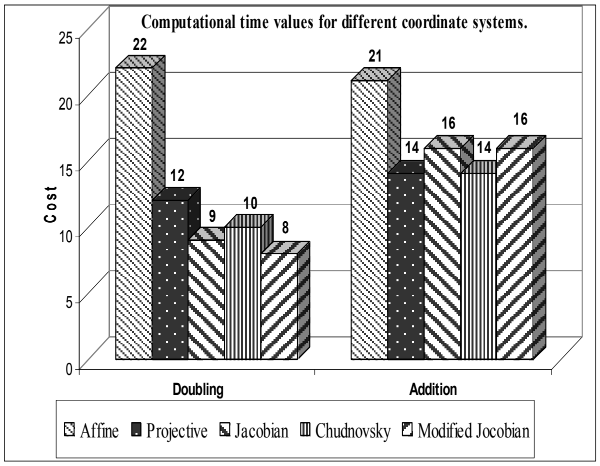

Figure 2.2 Computational time values for different coordinate systems.

Cohen proposed the mixed coordinates, where the inputs and outputs to point additions and doublings may be in different coordinates [2]. This can be very efficient when scalar multiplication is implemented with the base point stored in Affine coordinates. The costs of point conversion are represented in Table 2.1.

The conversion between Affine and Projective coordinates is inefficient due to a required inversion operation. Nevertheless; the conversion among the three Jacobian variants (Jacobian, Modified Jacobian and Chudnovsky Jacobian) is efficient and thus preferred for the scalar multiplication. The scalar multiplication can be started with point doubling in Modified Jacobian and continue with point addition in Chudnovsky Jacobian with the lowest cost.

Table 2.1 The computational costs of point conversion between different coordinate systems

From/To	Affine	Projective	Jacobian	Chud-novsky	Modified
Affine					
Projective	2M+I		2M+I	2M+I	2M+I
Jacobian	4M+I	4M+I		2M	3M
Chudnovsky	4M+I	4M+I			3M
Modified	4M+I	4M+I		2M	

[4].

3. Elliptic Curve Arithmetic on Different Software Libraries and Coordinate Systems

The point doubling and point addition is implemented by using NIST curves with 192, 224, 256, 384 and 521 bit elliptic curves and their domain parameters [9]. A Pentium 4 microprocessor computer is used in this study; it has 512MB RAM, 256KB cache, and Windows XP operating system, plus an ANSI C compiler.

The elliptic curve point addition and doubling algorithms are implemented on CRYMPIX, MIRACL and GMP multiprecision cryptographic software libraries. Each arithmetic operation is tested 10000 times in each coordinate system by different libraries and the average running times were collected for all.

3.1. Implementation with CRYMPIX Software Library

The "CRYMPIX" is a multiprecision cryptographic software development library [5]. It is in version 0.1.2.1 now and is developed by a research group in IS^3 Laboratory of Izmir Institute of Technology [10]. CRYMPIX triggers base case methods for ECC sized operands. The function call overhead is decreased by the help of C macros. The memory management overhead is negligible with built-in kernel functions. All codes are compiled in "optimization -2". Figures 3.1 and 3.2 depict the point addition and point doubling operations performed in CRYMPIX with different coordinate systems.

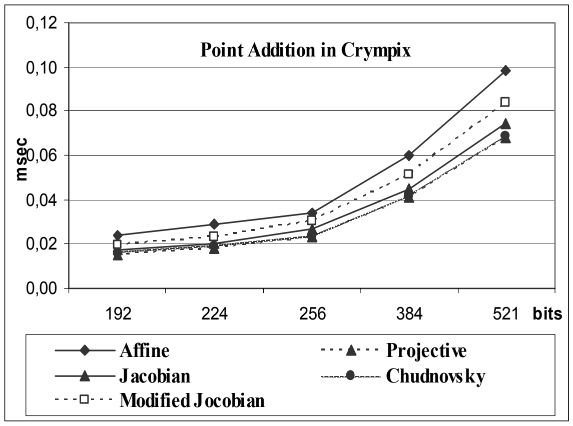

Figure 3.1 Point addition in different coordinate systems in CRYMPIX.

From an algorithmic analysis point of view, which was presented in Figure 2.2, the Chudnovsky Jacobian and Projective coordinates have the same computational cost for addition and, that is evident in Figure 3.1.

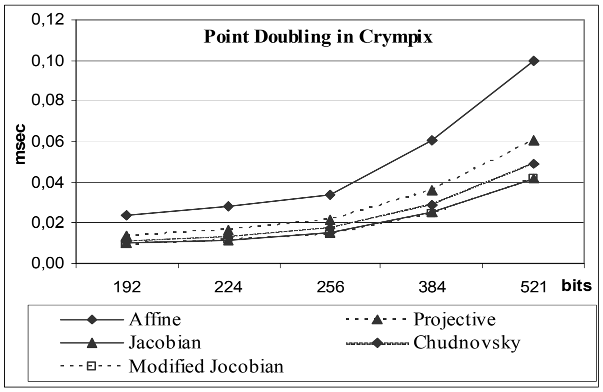

Figure 3.2 Point doubling in different coordinate systems in CRYMPIX.

Again, all the measurements represented in Figure 3.2 are clearly in accordance with the measurements depicted in Figure 2.2.

3.2. Implementation with GMP Software Library

GMP is a multiprecision software development library [11]. The GMP is now in version 4.1.4, assembly modules are excluded for the test operations; and compilation is done in "optimization -2". Figures

3.3 and 3.4 reflect the same operations performed in GMP library.

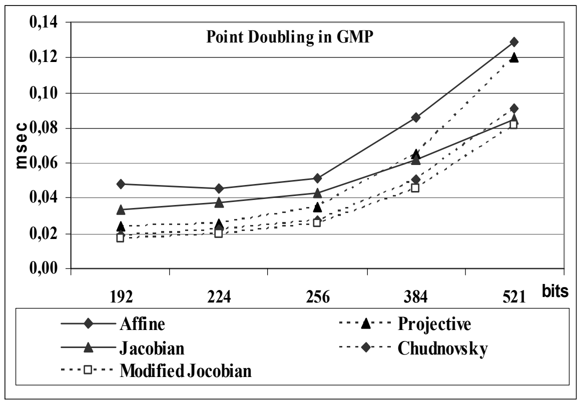

Figure 3.3 Point doubling in different coordinate systems in GMP.

Specifically the point doubling has the most efficient values in Modified Jacobian Coordinates with GMP.

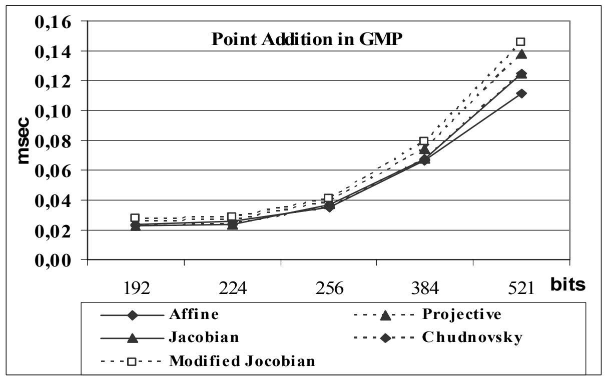

Figure 3.4 Point addition in different coordinate systems in GMP.

GMP uses the "Exact division scheme" for inversion. This inversion method is very efficient. The result of this effectiveness is clearly seen in Figure 3.4. Specifically, 521 bits elliptic arithmetic is more efficient in Affine coordinate system. The utilization of efficient inversion and standard reduction techniques causes this unexpected result with GMP.

3.3. Implementation with MIRACL Software Library

MIRACL v.4.8 includes the related functions for point doubling and addition in Affine and Jacobian coordinate systems. Therefore, the test results could be collected only for these coordinates, and these functions include "is_the_point_on_the_curve" control operation, therefore calculations could be done for point addition 10000 times after first doubling operation. MIRACL uses Karatsuba and Comba multiplication algorithms and utilizes Montgomery reduction method [1] [3]. Figure 3.5 depicts the point addition operation in MIRACL library.

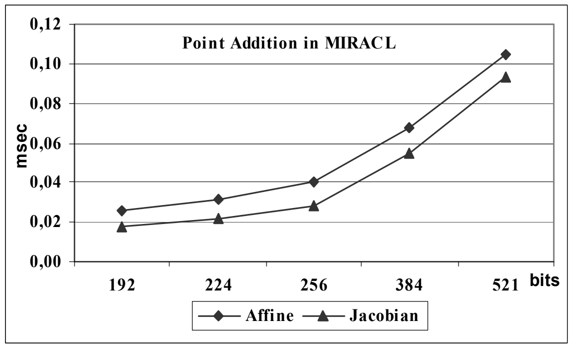

Figure 3.5 Performance values for point addition in MIRACL.

As shown in Figure 3.5, the computational cost of elliptic curve arithmetic is cheaper in Jacobian coordinate system and it is harmonized with Figure 2.2 as expected.

4. Conclusions & Contributions

The speed problem of ECC implementation comes from the highly computational cost of elliptic curve arithmetic. The inversion has the highest cost and if "three dimensional" coordinate systems are used instead of Affine coordinate system; the inversion is eliminated. When the Projective coordinate system and Jacobian coordinate systems are evaluated, the speed can be increased approximately 30%. Moreover, the computational cost of point conversion from one coordinate system to the other is another important point and the computational costs of point conversion between different coordinate systems are summarized in Table 2.1.

Taking into account the Table 4.1 it could be seen that, the Modified Jacobian has the minimal cost for doubling and Chudnovsky Jacobian has the minimal cost for addition and the cost for addition is the same for Projective and Chudnovsky Jacobian. However, the conversion cost from Modified Jacobian to Projective is 4M + I. Therefore, the execution should be done completely on Jacobian coordinate system. For example, the sequence of using coordinate systems can be preferred as Modified Jacobian for doubling and Chudnovsky Jacobian for point addition.

			Conversion		
	Doubling	**Addition**	**Modified**	**Chud-novsky**	**Pro Jective**
Projective	12M	14M	2M+I	2M+I	
Chudnovsky	10M	14M	3M		4M+I
Modified Jocobian	8M	16M		2M	4M+I

Table 4.1 Comparison of the computational costs of the Projective and Jacobian coordinate systems for scalar multiplication.

10. References

[1]. Blake I., Serousii G. & Smart N., *Elliptic Curves in Cryptography*, Cambridge University Press, 1999.

[2]. Cohen H., Miyaji A., Ono T., “Efficient elliptic curve exponentiation using mixed coordinates”, *In Advances in Cryptology – Asiacrypt’98 (Beijing), valume 1514,* Lecture notes in Computer Sciences, Springer Verlag Berlin, 1998, pp. 51-65.

[3]. Hankerson D., Menezes A., Vanstone S., *Guide to Elliptic Curve Cryptography*, Springer Verlag, 2003.

[4]. Hitchcook Y., Dawson E., Clark A., Montague P., “Implementing an Efficient Elliptic Curve Cryptosystem over GF(p) on a Smart Card”, *The 10th Biennial Computational Techniques and Applications Conference,* 16-18 July 2001, University Of Queensland,Brisbane, Australia, Received 1 June 2001, revised 24 October 2002.

[5]. Koltuksuz, A., Hişil, H., “Crympix: Cryptographic Multiprecision Library”, *LNCS 3733, ISBN: 3-540-29414-7, vol. 3733/2005*, Springer-Verlag, pp. 884-893.

[6]. Silverman J. H., *The Arithmetic of Elliptic Curves*, Springer Verlag, New York, 1986.

[7]. Washington L. C., *Elliptic Curves Number Theory and Cryptography*, Chapman Publications, 2003.

[8]. http://www.intel.com/research/silicon/mooreslaw.html

[9]. http://csrc.nist.gov/fips

[10]. http://is3.iyte.edu.tr/

[11]. http://www.swox.com/gmp/