Edwards Curves and Extended Jacobi Quartic-Curves for Efficient Support of Elliptic-Curve Cryptosystems in Embedded Systems

Chiara Peretti, Alessio Leoncini, Paolo Gastaldo, Rodolfo Zunino
Dept. of Electrical, Electronic, Telecommunications Engineering and Naval Architecture (DITEN), University of Genoa, Italy

Abstract

The efficient support of cryptographic protocols based on elliptic curves is crucial when embedded processors are adopted as the target hardware platforms. The implementation of Elliptic Curve Cryptography (ECC) offers a variety of design options, mostly covering the specific family of curves and the related coordinate system. At the same time, theory shows that a limited set of solutions can actually lead to efficient computational implementations. This paper analyzes these configurations from an applicative perspective, and mainly addresses two families of curves, namely, Edwards curves and extended Jacobi quartic curves, under a variety of coordinate systems. The experimental results prove that the ECC schemes based on either Edwards curves or extended Jacobi quartic curves can attain remarkable performances in terms of computational efficiency on low-cost, low-resources processors. Edwards curves, in particular, yield best performances when expressed in inverted coordinates.

1. Introduction

The availability of secure services is becoming an essential requirement in today’s media infrastructures at any level. Although modern cryptography offers a variety of mechanisms to ensure confidentiality and integrity, implementations usually may prove demanding from a computational perspective. This is a crucial issue in high-bitrate applications, which might be affected by the encryption/decryption overhead. Likewise, the limitation of computational costs is of paramount importance when supporting cryptographic services in low-cost, low-power mobile devices.

In today’s state of the art, practical cryptosystems support asymmetric cryptography, so that two parties can first set-up a public-key protocol and exchange a shared key, then switch to symmetric cryptography which is are less computationally demanding [1].
In the absence of known cryptanalytic procedures, the effective key length determines the effort to stage "brute force" attacks. That parameter, therefore, measures the level of security provided by the algorithm itself and, in turn, sets the computational complexity of the associate cryptosystem. Technological advances in computing continuously increase the security level (i.e., key length) to ensure an algorithm's robustness, hence one should favors those cryptosystems that best tradeoff computational cost and robustness. This is especially true when addressing embedded implementations.

This paper therefore tackles the efficient implementation of public-key algorithms on embedded systems, and analyzes the performances of asymmetric schemes based on Elliptic Curve Cryptography (ECC). This approach has both a theoretical and a practical rationale.

From a theoretical viewpoint, Elliptic curves have been - in the last twenty years - the object of intense study in the area of public-key cryptography. Both scientific literature and technology standards [2] show that ECC can be a valid alternative to popular schemes such as RSA and Diffie-Hellman, because ECC can deliver the same level and type of security with much shorter keys [2]. The U.S. National Security Agency (NSA) has included a cryptographic scheme based on ECC in the set of recommended algorithms [3].

A practical viewpoint requires taking into account two main features, namely, 1) the family of curves that is chosen and 2) the coordinate system adopted. As to the former, a variety of families of elliptic curves can support an asymmetric protocol and, as far as number theory is concerned, each family complies to a specific algebra [4]. The issue is that those families may differ significantly in both computational cost and security aspects. In addition, every elliptic-curve expression can be remapped into a selection of coordinate systems, which in turn lead to different implementations of the embedded operations and ultimately set the computational complexity of the ECC cryptosystem.
Security and computational issues entangle in the process of point multiplication. Its core operations (point addition and point doubling) have different computational costs, and therefore can be recognized by side-channel attacks [5]. To prevent such vulnerability, practical implementations of point multiplication typically rely on the Montgomery ladder algorithm [6]. This approach, however, notably increases the computational complexity of the overall cryptographic process.
The basic approach described in this paper involves alternative families of elliptic curves, which feature one algorithm to work out both point addition and

point doubling. Additional degrees of freedom stem from the selection of the coordinate system that is adopted to carry out computations. The research presented here analyzes such options, and derives a set of practical criteria for the design and deployment of efficient ECC in embedded systems.

The paper considers two alternative paradigms to standard elliptic curves: Edwards curves [7], and extended Jacobi quartic curves [8]. These options seem quite beneficial for two main reasons. Firstly, Bernstein et al. [9] showed that these families minimize the theoretical number of operations in ECC deployment. Secondly, both types of curves feature a unified group law, featuring a common algorithm for the core algebra. In addition to ensuring immunity against side-channel attacks, one avoids costly Montgomery-ladder implementations in the eventual ECC systems.

To the best of the authors' knowledge, few works in the literature have considered the computational complexity issues related to the support of ECC scheme by using Edwards curves or extended Jacoby quartic curves [9]. When targeting hardware implementations, the literature only considered the adoption of digital custom processors [10]. An important novelty of the research presented here therefore lies in pursuing inexpensive implementations on low-end processors for embedded systems.

The validation of the research involved a massive experimental campaign covering: 1) a variety of embedded processors, 2) varying key length, and 3) a complete ECC cryptosystem supported by different elliptic curve families and different coordinate systems. The resulting standard Diffie-Hellman key-exchange protocol (key length was 256 bits) completed in less than 200 msec on a standard ARM9 processor (clock rate @200 MHz) when adopting an ECC cryptosystem based on Edwards curves. This result is noteworthy from different viewpoints:

- the level of security provided by an ECC cryptosystem with a key size of 256 bit compares with that attained by an RSA cryptosystem having key length of 3072 bit [2].
- state-of-the-art hardware implementations of RSA-based cryptosystems target a key length of 2048 bit [2].

These experiments allow to evaluate to what extent the choice of a coordinate system affects the eventual performance of the overall ECC cryptosystem.

The rest of the paper is organized as follows. Section 2 provides a theoretical background of elliptic curve cryptography. Section 3 discusses the advantages of selecting a suitable coordinate system for the implementation of an ECC cryptosystem and introduces the experimental set up adopted in this research. Section 4 presents the result of the campaign of experiments. Finally, section 5 makes some concluding remarks.

2. Theoretical Background

2.1. Edwards curves

Harold M. Edwards [7] introduced a new normal form for elliptic curves in 2007. Given a prime number p, the Edwards curve over GF(p) is expressed as:

$$ED(p)\colon x^2 + y^2 = c^2(1 + dx^2y^2)\ mod\, p \qquad (\#1)$$

where p is the field ($p > 3$).

Every elliptic curve over a prime field is birationally equivalent to a curve in Edwards form over an extension of the field -and in many cases over the original field- if: 1) c, $d \in GF(p)$, and 2) $cd(1 - dc4\neq 0\ mod\, p$. If p is finite then a sizeable fraction of all elliptic curves over GF(p) can be written as Edwards curves. Many Edwards curves are isomorphic to Edwards curves with d = 1; all Edwards curves are isomorphic to Edwards curves with c = 1. The research presented in this paper always adopts the setting c = 1, thus following the usual approach in state-of-the-art papers on Edwards curves.

Edwards curves and elliptic curves share the same group law, with one exception: point addition in Edwards curves can also be used to compute (P + P); thus, there is no need of a dedicated "point doubling" operation. The complete group laws of an Edward curve are defined only if d is not a square in $GF(p)$, and can be formalized as follows:

Identity: $P + O = O + P = P$, for all $P(x, y) \in ED(p)$; the neutral element O of the group is the rational point with coordinates $(0, c)$, $c \in GF(p)$.

Negation: if $P(x, y) \in ED(p)$ then $(x, y) + (-x, y) = \infty$. The point $(-x, y)$ is denoted by $-P$ and is called the *negative* of P; note that $-P$ is indeed a point in $ED(p)$. Similarly, -O = O.

Unified law: let $P = (x_1, y_1) \in ED(p)$ and $Q = (x_2, y_2) \in ED(p)$ be two rational points; then $P + Q = R(x_3, y_3)$ where:

$$x_3 = \frac{(x_1y_2 + x_2y_1)}{c(1 + dx_1x_2y_1y_2)} \ mod\ p \quad (\#2)$$

$$y_3 = \frac{(y_1y_2 + x_1x_2)}{c(1 + dx_1x_2y_1y_2)} \ mod\ p \quad (\#3)$$

P+Q is on the curve only if the coordinates (x_3, y_3) are defined, i.e., when $dx_1x_2y_1y_2 \notin \{-1,1\}$ [11].

In practice, Edwards curves allow one to implement point doubling by exploiting the point addition algorithm; i.e., $2P = P + P$. This is not true with standard elliptic curves, since point addition $P+Q$ only applies when $P \neq \pm Q$. Such feature determines the fundamental difference between Edwards curves and standard elliptic curves in terms of cryptographic applications.

2.2. Extended Jacobi Quartic curves

An extended Jacobi quartic curve JQ(p) over $GF(p)$ can be expressed as [8]:

$$JQ(p): y^2 = \epsilon x^4 - 2\delta x^2 + 1 \ mod\ p \quad (\#4)$$

where p is the field ($p > 3$), $\epsilon, \delta \in GF(p)$, and $\Delta = 256\epsilon(\delta^2 - \epsilon^2) \neq 0\ mod\ p$.

The complete group laws of an extended Jacobi curve are defined only if ϵ is not a square in $GF(p)$ and can be formalized as follows:

Identity: $P + O = O + P = P$, for all $P(x, y) \in JQ(p)$; the neutral element O of the group is the rational point with coordinates (0,1).

Negation: if $P(x, y) \in JQ(p)$ then $(x, y) + (x,-y) = O$. The point $(-x, y)$ is denoted by $-P$ and is called the *negative* of P; note that $-P$ is indeed a point in $JQ(p)$. Similarly, $-O = O$.

Unified law: let $P = (x_1, y_1) \in JQ(p)$ and $Q = (x_2, y_2) \in JQ(p)$ be two rational points; then $P + Q = R(x_3, y_3)$ where:

$$x_3 = \left(\frac{x_1y_2 + x_2y_1}{1 - \epsilon x_1^2 x_2^2}\right) mod\ p \quad (\#5)$$

$$y_3 = \left(\frac{(y_1y_2 - 2\delta x_1x_2)(1 + \epsilon x_1^2x_2^2) + \\ +2\epsilon x_1x_2(x_1^2 + x_2^2)}{(1 - \epsilon x_1^2x_2^2)^2}\right) mod\ p \quad (\#6)$$

P+Q is on the curve only if the coordinates (x_3, y_3) are defined, i.e., when $\epsilon x_1^2 x_2^2 \neq 1\ mod\ p$.

For the sake of clarity, it should be pointed up that the literature actually provides two different formalizations of the extended Jacobi quartic curves: the one given in equation (#4) and the one expressed as:

$$JQ(p): y^2 = \epsilon x^4 + 2\delta x^2 + 1\ mod\ p \quad (\#7)$$

The two formalizations differ in the sign of the coefficient δ. Indeed, the two equations lead to different formalizations for the unified group law. In this article, extended Jacobi quartic curves are formalized according to equation (#4); hence, the unified law is implemented as reported above.

(#11)

2.3. Elliptic curves for cryptography

The elliptic curve discrete logarithm problem (ECDLP) sets the basis for using elliptic curves in cryptography. To formalize the problem, one first needs to introduce point multiplication (or scalar multiplication) $Q = k \cdot P$, where $Q,P \in E(p)$ and $k \in \mathbb{N}$ ($Q,P \in ED(p)$ when using Edwards curves, or $Q,P \in JQ(p)$ when using extended Jacobi quartic curves). According to ECDLP, given P and Q, it is computationally infeasible to obtain k, if k is large enough [12]. To reconstruct k, one should solve the discrete logarithm of Q with respect to the base P in the Edwards or extended Jacobi quartic curve domain. This problem, in fact, is currently characterized by a full exponential complexity [12]. Public-key cryptographic schemes exploit ECDLP by setting k as the private key and Q as the public key.

Table 1 gives, for increasing values of k, the security level achieved by the corresponding ECC cryptosystem: the first column of the table gives the key size k; the second column gives the effort required to complete a brute-force attack to the cryptographic system, measured in the number of instructions executed in one year at one million instructions per second (MIPS-year); the third column gives the expected lifetime of the cryptosystem; the last column indicates the key size that should be used by a standard

RSA-based cryptosystem to provide the same security level. The reported data confirm that the ECC-based approach can outperform RSA-based schemes, as the former provide the same security level of the latter ones with a shorter key. The rationale behind such behaviour is that the RSA scheme can be attacked with algorithms that have a sub-exponential complexity. Table 1 also proves that a cryptosystem based on elliptic curves attains the same security level of an RSA-based scheme with a lower computational load. This happens because the complexity of a cryptographic scheme correlates to a shorter key. This aspect is crucial in hardware implementations on low-resources embedded devices. Edwards curves and extended Jacobi quartic curves can provide further improvements in terms of computational load with respect to standard elliptic curves.

Table 1. ECC Security level for varying key length

ECC	MIPS Years necessary to attack	Protection lifetime	Integer factorization cryptography (e.g. RSA)
160	10^{12}	Until 2010	1024
224	10^{24}	Until 2030	2048
256	10^{28}	Beyond 2031	3072
384	10^{47}	Beyond 2031	7680
512	10^{66}	Beyond 2031	15360

Point multiplication is the procedure that marks the difference in computational load between conventional ECC schemes and cryptosystems based on Edwards curves or extended Jacobi quartic curves. The “double and add” method is the conventional algorithm that supports the implementation of point multiplication. This research considers the right-to-left binary algorithm; a dual implementation, the left-to-right binary algorithm, can also be adopted [4]. In both cases, a loop involves a point addition (conditionally executed) and a point doubling (always executed). The sequence of ‘0’s and ‘1’s in the private key determines the actual progression of the computation. This is where a faulty implementation yields to side-channels attacks, exploiting differential information about timing, electromagnetic radiation or power consumption to discriminate the two operations. The popular “Montgomery ladder” solution [4] normalizes the multiplication algorithm, but brings about a significant computational overhead. Introducing Edwards curves or extended Jacobi quartic curves overcomes that drawback because point multiplication is carried out according to the simpler double-and-add method. The group law of both Edwards and extended Jacobi quartic curves state that point doubling can be completed by using the regular point addition procedure.

3. Practical cryptosystem architectures

A straightforward application of the unified laws (both Edwards/extended Jacobi Quartic) given in Sect. 2 might result in a drawback, as the expressions all require a modular inversion to compute the coordinates of the result rational point, R. Modular inversion is in fact a computationally demanding process that may severely affect performances. That issue can be tackled by a change in the coordinate system.

The families of elliptic curves can all be represented according to different coordinate systems [4]. In Sect. 2.1 and Sect. 2.2, affine coordinates (x,y) have been adopted. Nevertheless, the literature shows that by mapping affine coordinates to a different coordinate system one can eventually obtain a convenient reformulation of the group laws [1]. In particular, one is interested in those coordinate systems that bypass modular inversions in point multiplication. Clearly, the mapping procedure brings about an additional overhead that should be carefully assessed.

An interesting, although merely theoretical, analysis of the computational efforts required to complete point multiplication in different coordinate systems is provided in [9]. That research showed that, to minimize computational load, one should represent the curve in a *projective* coordinate system; as a result, a rational point P is represented by a triplet (X, Y, Z), which replaces the affine coordinates (x,y). In fact, different projective representations may stem from the general projective coordinates system [4]. In this regard, the literature showed [9] that specific projective representations exist for each family of curves that can lead to the best computational performances for the point operations. Thus, Table 2 reports – for each family – the specific coordinate transformation to be applied and the conventional nomenclature associated to that transformation in the literature. Table 2 includes two available transformations for the Edwards curves, as the analysis published in [9] proved that both the set up can lead to an optimization of the computational costs.

Table 2. Coordinate transformations

Curve	**Transformation**	**Naming Convention**
Extended Jacobi quartic	$x = \frac{X}{Z}$ $y = \frac{Y}{Z^2}$	*projective coordinates*
Edwards	$x = \frac{X}{Z}$ $y = \frac{Y}{Z}$	standard projective coordinates
Edwards	$x = \frac{Z}{X}$ $y = \frac{Z}{Y}$	inverted Edwards coordinates

As compared with the theoretical analysis, the present research evaluates the comparisons listed in Table 2 by estimating the computational performance of the ECC systems that would stem from these configurations. The comparative analysis is carried out within the specific target framework of embedded cryptosystems [7,8]. Sect 3.1 and Sect. 3.2, outline the formalizations of the group laws that stem from the change of coordinate system for Edwards and extended Jacobi quartic curves, respectively. Sect. 3.3 reports on the theoretical assessment of the computational load associated to the unified law with the different configuration options.

3.1. Edwards curves in standard projective and inverted coordinates

A point (x_a, y_a) on a Edwards curve represented in affine coordinates can be transformed in a point (X_s, Y_s, Z_s) that belongs to the same Edwards curve represented in standard projective coordinates. The transformation rule can be formalized as follows:

$$x_a = \frac{X_s}{Z_s} \, mod \, p, \qquad y_a = \frac{Y_s}{Z_s} \, mod \, p, \qquad \text{(\#8)}$$
$$Z_s \neq 0 \; mod \; p$$

Accordingly, the curve equation in standard projective coordinates becomes:

$$(X^2 + Y^2)Z^2 = c^2(Z^4 + dX^2Y^2) \, mod \, p \qquad \text{(\#9)}$$

where parameters c and d should be set by following the rules already reported in Sec. 2.

In the new coordinate system, the neutral element is $(0, c, 1)$, and the inverse of (X, Y, Z) is $(-X, Y, Z)$. Given two points $P_1\ (X_1, Y_1, Z_1)$ and $P_2\ (X_2, Y_2, Z_2)$ belonging to the Edwards curve (#9), one has that:

the unified law $P_3(X_3, Y_3, Z_3) = P_1(X_1, Y_1, Z_1) + P_2(X_2, Y_2, Z_2)$ is expressed by the following equations [11]:

$$X_3 = AF\big((X_1 + Y_1)(X_2 + Y_2) - C - D\big) mod \, p; \qquad \text{(\#10)}$$
$$Y_3 = AG(D - C) \; mod \, p;$$
$$Z_3 = cFG \;\; mod \; p$$

where:

$$A = Z_1 Z_2 \;\; mod \, p; \;\; B = A^2 \;\; mod \, p;$$
$$C = X_1 X_2 \;\; mod \, p; \;\; D = Y_1 Y_2 \;\; mod \, p;$$
$$E = dCD \;\; mod \, p; F = B - E \;\; mod \, p;$$
$$G = B + E \;\; mod \, p;$$

The above formalism shows that the unified law does not involve any modular inversion when the curve is represented in projective coordinates. The coordinate transformation involves the affine, bi-dimensional coordinates system and a projective, three-dimensional coordinates system. The computational overhead brought about by the mapping process can be fully ascribed to the conversion from standard projective coordinates to affine coordinates; i.e., the conversion from the three dimensional space to the bi-dimensional space. In this case, such procedure involves one modular inversion. Thus one might instead consider inverted Edwards coordinates [13]. Within that framework, a point (x_a, y_a) on an Edwards curve (in affine coordinates) is mapped to a point (X_i, Y_i, Z_i) that belongs to the same Edwards curve but is represented in the new coordinate system. The mapping rule is formalized as follows:

$$x_a = \frac{Z_i}{X_i} \, mod \, p, \qquad y_a = \frac{Z_i}{Y_i} \, mod \, p, \qquad \text{(\#11)}$$
$$Z_i \neq 0 \;\; mod \; p$$

Accordingly, the curve equation in inverted Edwards (#22) coordinates becomes [13]:

$$(X^2 + Y^2)Z^2 = c^2(X^2Y^2 + dZ^4) \, mod \, p \qquad \text{(\#12)}$$

where $XYZ \neq 0$; parameters c and d should be set by following the rules already reported in Sect.2.

(#23)

TABLE 3. COMPUTATIONAL COSTS FOR THE UNIFIED LAW UNDER DIFFERENT CONFIGURATIONS

Curve	Coordinates	Addition / Unified Law
Edwards	affine	$2I + 6M + 4H + 1\bar{c} + 1\bar{d}$
	standard projective	$10M + 1S$
	inverted Edwards	$9M + 1S$
extended Jacobi quartic	affine	$2I + 5M + 5S + 7H + 2\bar{\epsilon} + 1\bar{\delta}$
	projective	$8M + 3S$

When Edwards curves are represented according to the inverted Edwards coordinates system, the neutral element and the inverse of (X, Y, Z) do not belong to the curve (#9). The constraint $XYZ \neq 0$ stems from such condition [13]. Indeed, given two points $P_1\,(X_1, Y_1, Z_1)$ and $P_2\,(X_2, Y_2, Z_2)$ belonging to the Edwards curve (#9), one has that:

the unified law $P_3(X_3, Y_3, Z_3) = P_1(X_1, Y_1, Z_1) + P_2(X_2, Y_2, Z_2)$ is expressed by the following equations [13]:

$$X_3 = (E + B)H \ \ mod\, p; \qquad (\#13)$$
$$Y_3 = (E - B)I \ \ mod\, p;$$
$$Z_3 = AHI \ \ mod\, p.$$

where

$A = Z_1 Z_2 \ \ mod\, p;\ \ B = dA^2 \ \ mod\, p;$
$C = X_1 X_2 \ \ mod\, p;\ \ D = Y_1 Y_2 \ \ mod\, p;$
$\ \ = CD \ \ mod\, p;\ \ H = C - D \ \ mod\, p;$
I$= (X_1 + Y_1)(X_2 + Y_2) - C - D \ \ mod\, p;$

The important fact is that moving to the inverted Edwards coordinates allows one to avoid modular inversions. Again, a computational overhead stems from the need to map a three-dimensional projective system back to the two-dimensional affine system. In the case of inverted Edwards coordinates, this procedure involves two modular inversion and two modular multiplications.

3.2. Extended Jacobi quartic curves in projective coordinates

A point (x_a, y_a) belonging to an extended Jacobi quartic curve represented in affine coordinates can be transformed in the point (X_s, Y_s, Z_s) belonging to the same curve represented in projective coordinates. The mapping rule can be formalized as follows:

$$x_a = \frac{X_s}{Z_s} \ mod\, p, \qquad y_a = \frac{Y_s}{Z_s^2} \ mod\, p, \qquad (\#14)$$
$$Z_s \neq 0 \ \ mod\, p$$

The curve equation in projective coordinates then becomes [14]:

$$Y^2 = \epsilon X^4 - 2\delta X^2 Z^2 + Z^4 \ \ mod\, p \qquad (\#15)$$

where parameters δ and ϵ should be set by following the rules already reported in section 2.

In the new coordinate system the neutral element is (0,1,1), and the inverse of (X, Y, Z) is $(-X, Y, Z)$. Moreover, given two points $P_1\,(X_1, Y_1, Z_1)$ and $P_2\,(X_2, Y_2, Z_2)$ that belong to the extended Jacobi quartic curve (#15), one has that:

the unified law $P_3(X_3, Y_3, Z_3) = P_1(X_1, Y_1, Z_1) + P_2(X_2, Y_2, Z_2)$ is expressed by the following equations [14]:

$$X_3 = X_1 Z_1 Y_2 + Y_1 X_2 Z_2 \ \ mod\, p; \qquad (\#16)(\#28)$$
$$Y_3 = [B + G][D - \delta F] + \epsilon F(X_1^2 Z_2^2 + X_2^2 Z_1^2) \ \ mod\, p;$$
$$Z_3 = B - G \ \ mod\, p.$$

where

$A = Z_1 Z_2 \ \ mod\, p; B = A^2 \ \ mod\, p;$
$C = X_1 X_2 \ \ mod\, p;\ \ D = Y_1 Y_2 \ \ mod\, p;$
$\ \ = C^2 \ \ mod\, p;\ \ F = 2AC \ \ mod\, p;$
$G = \epsilon E \ \ mod\, p;$

As a result, moving from affine coordinates to the projective coordinates allows one to avoid modular inversions. The coordinate transformation process again involves the affine, bi-dimensional coordinates system and a projective, three-dimensional coordinates system. The transformation from projective coordinates back to affine coordinates requires one to complete two modular inversions and three modular multiplications.

3.3. Theoretical analysis of the computational load

Table 3 provides the computational load of the unified law for the different configurations of the ECC system analyzed in this research. Thus, the first column of the table gives the curve family; the second column indicates the coordinate system adopted; finally, the third report the computational costs of the unified law operation when the ECC system is based on the specific set up {curve, coordinate system}. The following notations have been used in the formalization of the computational costs:

- *I*: cost of a modular inversion;
- *M*: cost of a modular multiplication;
- *S*: cost of a modular squaring;
- *H*: cost a modular addition/subtraction.

It is worth noting that in the literature one finds that $I \approx 100 \cdot M$, while $M \approx S$.

When the unified law involves rational points belonging to an Edward curve, the following quantities should also be defined:

- $\bar{c}$: cost of a modular multiplication by c (Edwards curves);
- $\bar{d}$: cost of a modular multiplication by d (Edwards curves).

When the operation involves rational points belonging to an extended Jacobi quartic curve, the following quantities should be defined:

- $\bar{\epsilon}$: cost of a modular multiplication by ϵ (extended Jacobi quartic curves);
- $\bar{\delta}$: cost of a modular multiplication by δ (extended Jacobi quartic curves).

The values of the computational costs, $\bar{c}$, $\bar{d}$, $\bar{\epsilon}$, and $\bar{\delta}$ actually depend on the number of bits required to represent the parameters c, d, ϵ and δ, respectively, and are set accordingly.

An analysis of the computational costs reported in Table 3 leads to a first, important conclusion: the unified law always proves less demanding when carried out in a projective coordinates system, irrespectively of the specific family of curves adopted. The crucial issue is the modular inversion, which cannot be avoided when using affine coordinates. Table 3 shows that the computational effort required to complete the operation in projective coordinates does not exhibit large variations among the various families. However, one should take into account that any point multiplication procedure eventually iterates several times this formula; as a result, a small saving in the computational load on the single operation may lead to significant improvements on the complete ECC system.

3.4. Overall cryptosystem

The previous Sections showed that the design of an ECC cryptosystem to be hosted by a programmable microprocessor requires addressing a few aspects: the family of curves to be adopted, and the coordinate system to be used. Each one of these design options is relevant to the computational efficiency of the eventual cryptosystem.

The proposed cryptosystem supports the point multiplication that is the core operation of the standard elliptic curve Diffie–Hellman (ECDH) key agreement protocol [1]. The general scheme of the ECC cryptosystem is the following:

- coordinate transformation (from affine to projective coordinates);
- point multiplication;
- coordinate transformation (from projective to affine coordinates).

The following Section presents experiments to compare the performance of the different implementations of the ECC cryptosystem that can be obtained by setting the two features: type of curve and coordinate system.

4. Experimental results

In order to test all the possible approaches that can be made to produce an elliptic curve point multiplication, different embedded platforms were considered:

- ARM9 running at 210 MHz;
- ARM11 running at 700 MHz;
- ARM Cortex A8 running at 1.0 GHz;

Experimental results show the associate timings required to complete the computing process.

The experimental session involved six different Edwards curves and as many extended Jacobi quartic curves. In both cases, three different key sizes have been used: 256 bit, 384 bit, and 512 bit. All the curves are cryptographically secure, as they belong to the curve database provided in [15].

Table 4. Edwards curve cryptosystem on ARM9@ 210Mhz (Values are in msec).

ID	bits	Affine multiplication	Standard projective multiplic.	Inverted multiplic.
E-1	256	242.1	153.6	145.9
E-2	256	247.5	158.9	149.8
E-3	384	521.3	352.1	335.7
E-4	384	519.3	325.4	309.5
E-5	512	955.1	582.4	544.5
E-6	512	946.4	604.4	563.6

Table 5. Edwards curve cryptosystem on ARM11@ 700Mhz (Values are in msec).

ID	bits	Affine multiplication	Standard projective multiplic.	Inverted multiplic.
E-1	256	51.86	31.84	31.48
E-2	256	54.57	34.94	34.46
E-3	384	115.21	70.71	68.97
E-4	384	115.76	67.18	65.02
E-5	512	212.02	129.32	120.22
E-6	512	208.39	129.86	120.83

Table 6. Edwards curve cryptosystem on ARM CortexA8@ 1Ghz (Values are in msec).

ID	bits	Affine multiplication	Standard projective multiplic.	Inverted multiplic.
E-1	256	24.63	17.96	17.42
E-2	256	24.62	18.51	18.11
E-3	384	55.52	37.73	35.77
E-4	384	55.74	36.82	35.58
E-5	512	107.01	68.15	64.60
E-6	512	106.37	73.43	65.83

Table 7. Extended Jacobi quartic curve cryptosystem on ARM9@ 210Mhz (Values are in msec).

ID	bits	Affine multiplic.	Standard projective multiplication
JQ-1	256	333.7	197.3
JQ-2	256	358.7	212.1
JQ-3	384	790.9	503.4
JQ-4	384	779.0	492.9
JQ-5	512	1298.3	831.1
JQ-6	512	1313.0	818.3

Table 8. Extended Jacobi quartic curve cryptosystem on ARM11@ 700Mhz (Values are in msec).

ID	bits	Affine multiplic.	Standard projective multiplication
JQ-1	256	77.08	47.14
JQ-2	256	81.62	49.43
JQ-3	384	168.12	106.90
JQ-4	384	167.90	106.66
JQ-5	512	286.73	185.16
JQ-6	512	298.31	190.34

Table 9. Extended Jacobi quartic curve cryptosystem on ARM CortexA8@ 1Ghz (Values are in msec).

ID	bits	Affine multiplic.	Standard projective multiplication
JQ-1	256	45.19	23.64
JQ-2	256	37.56	24.74
JQ-3	384	80.49	53.28
JQ-4	384	78.14	51.47
JQ-5	512	148.05	100.04
JQ-6	512	145.80	97.68

Table 10. Edwards curves coordinate conversions on ARM11@ 700Mhz (Values are in msec).

ID	bits	Standard projective to affine coord's	Inverted to affine coordinates
E-1	256	0.04	0.08
E-2	256	0.05	0.09
E-3	384	0.07	0.12
E-4	384	0.06	0.13
E-5	512	0.10	0.18
E-6	512	0.09	0.18

Table 11. Extended Jacobi quartic curve coordinate conversions on ARM11@ 700Mhz. (Values are in msec).

ID	bits	Projective to affine coordinates
JQ-1	256	0.09
JQ-2	256	0.09
JQ-3	384	0.14
JQ-4	384	0.14
JQ-5	512	0.18
JQ-6	512	0.19

Tables 4-9 report on the results obtained by evaluating point multiplication with the binary/right-to-left algorithm. Each table refers to a specific embedded architecture. For each curve, each Table gives: the curve identifier, including the bit-length of curve parameters; the time spent to accomplish a point multiplication on the Edwards curve, and the same quantity for the extended Jacobi quartic curves, for different coordinate systems. All timings are expressed in msec.

The empirical evidence obtained on the various hardware platforms highlights some interesting outcomes. First, the use of projective coordinate systems allows one to obtain consistent improvements on the point multiplication procedure. Secondly, the Edwards curves in both coordinates seem to outperform in general the extended Jacobi quartic curves, in terms of computing speed. But the best performance was obtained by the Edwards curves in inverted coordinates. Finally, from the tables 10 and 11, one can note that the computational time required by coordinate conversions is in general very low with respect to the projective multiplication phase regardless of the type of curves.

The extensive experimental session aims to demonstrate the flexibility of presented framework, allowing different hardware platform to complete cryptographic primitives over both Edwards and extended Jacobi quartic curves and different coordinate systems.

5. Conclusions

Elliptic curve cryptography provides an appealing alternative to conventional public-key algorithms, as the former cryptosystem can obtain a comparatively higher security level per key-bit. Since the computational complexity of a cryptographic machine correlates with the size of the key, low-resource, low-power embedded devices mostly benefit from that property. However, the actual implementation of elliptic-based cryptosystems requires one to deal with a range of design options, which in turn may affect the eventual computational complexity of the system.

The paper showed that Edwards curves can effectively support cryptosystems in low-resource, low-power embedded devices. In principle, both Edwards curves and extended Jacobi quartic curves provide interesting features that allow one to optimize the number of operations to be completed for implementing ECC. Indeed, the experimental session proved that the former configuration can attain remarkable results in terms of computational efficiency. In this regard, it is worth noting that a cryptosystem based on Edwards curves attains satisfactory performances even in devices running at a clock frequency of 200 MHz.

6. Acknowledgments

The authors would like to thank S. Decherchi for his research and his teaching that made possible this work.

7. References

[1] Menezes AJ, van Oorschot PC, Vanstone SA: *Handbook of Applied Cryptography*. CRC Press 1996.

[2] E. Barker, W. Barker, W. Burr, W. Polk, M. Smid, NIST special publication 800-57, "Recommendation for Key Management – Part 1: General (Revision 3)". July 2012. Available "http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf".

[3] Fact Sheet NSA Suite B Cryptography, National Security Agency, http://www.nsa.gov/ia/ programs/suiteb_cryptography/, retrieved November 17, 2012.

[4] D. Hankerson, A. Menezes, S. Vanstone, "Guide to Elliptic Curve Cryptography", Springer, 2004.

[5] Quisquater, J.J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards. e-smart 2001, LNCS 2140 (2001) 200–210.

[6] P. Montgomery, "Modular Multiplication Without Trial Division", Mathematics of Computation, vol. 44, pp. 519–521, 1985.

[7] H.M. Edwards, "A normal form for elliptic curve", Bulletin of the American Mathematical Society - 2007.09.04.

[8] H. Hisil, K. K. Wong, G. Carter, E. Dawson. "Jacobi Quartic curves Revisited", ACISP 2009, LCNS Vol. 5594, pp 452-468, Springer-Verlag, 2009.

[9] D.J.Bernstein, P.Birkner, T.Lange and C.Peters. "Optimizing double-base elliptic curve single-scalar multiplication". INDOCRYPT'07 Proceedings of the cryptology 8th international conference on Progress in cryptology, pp 167-182, Springer-Verlag Berlin, 2007.

[10] Guerric Meurice de Dormale, Jean-Jacques Quisquater. "High-speed hardware Implementations of Elliptic Curve Cryptography: A survey". Journal of Systems Architecture 53 (2007), pp. 72–84, 2006.

[11] D.J. Bernstein, T. Lange, "Faster addition and doubling on elliptic curves", University of Chicago, Chicago, IL 60607-7045,USA, Technische Universititeit Eindoven, P.O.,Box 513, 5600 MB Eindhoven, Netherlands - 2007.09.06.

[12] S. A. Vanstone, “Next generation security for wireless: elliptic curve cryptography”, Computers and Security, Vol 22, No 5, Aug. 2003.

[13] D.J.Bernstein and T.Lange. “Inverted Edwards coordinates”. AAECC'07 Proceedings of the 17th international conference on Applied algebra, algebraic algorithms and error-correcting codes, pp 20-27, Springer-Verlag Berlin, 2007.

[14] O. Billet, M. Joye. ”The Jacobi Model of an Elliptic Curve and Side-Channel Analysis” Proceedings of the 15th international conference on Applied algebra, algebraic algorithms and error-correcting codes AAECC'03, pp 34-42 Springer-Verlag Berlin, 2003.

[15] H. Ivey-Law, R. Rolland. “Constructing a database of cryptographically strong elliptic curves”. Proc. of 5th Conf. on Network Architectures and Information Systems Security, SAR-SSI 2010.