Efficient Elliptic Curve Cryptography with Collapsed Representation

Samuel Antão[*,1], Ricardo Chaves[*,1], Leonel Sousa[*,1]

* *INESC-ID/Instituto Superior Técnico - TU Lisbon*
R. Alves Redol, 9, 1000-029 Lisbon, PORTUGAL

ABSTRACT

This paper introduces a new and efficient method to perform elliptic curve multiplication using a compact representation of the points in the curve. Regarding to the traditional approach, this method reduces bandwidth in half with minimal impact either in the multiplication time or the latency. Different elliptic curve cryptographic processors were implemented in a reconfigurable device, based on sequential and parallel approaches, in order to attest these results. The advantages of this compact representation are particularly relevant in systems where communication power is dominant, such as in sensor networks.

KEYWORDS: Elliptic Curve Cryptography, Coordinate Compressing

1 Introduction

Elliptic Curve (EC) public key cryptosystems have been shown to have higher security per bit than the widely used Rivest-Shamir-Adleman (RSA) cryptosystem. Therefore, the required communication bandwidth and memory requirements are reduced, because smaller keys need to be transmitted and stored in the network nodes in order to establish a secure communication. These characteristics become even more important for applications supported on embedded systems, with severe memory and power consumption constraints. Examples of such devices are smart cards and sensor network nodes for which the power required for transmission is typically around 60% of the total required power [BTn07].

An EC is constructed by a set of two coordinated points $P_i = (x_i, y_i)$ which forms a group, and EC operations, namely point addition, are established over this group. Particularly to binary extension fields (Galois fields) $GF(2^m)$, where operations are efficiently implemented in hardware (namely additions are performed by a bitwise exclusive *OR* operation), this group ($E(GF(2^m), a, b)$), and its elements can be described by a point $(\mathcal{O})$ at infinity and the set of points $P_i(x_i, y_i)$ that obey:

$$y_i^2 + x_i y_i = x_i^3 + ax_i^2 + b,\ a, b \in GF(2^m). \tag{1}$$

[1]E-mail: {sfan,rjfc,las}@sips.inesc-id.pt

Applying point addition recursively, it is possible to exponentiate a point P, which is usually referred as EC multiplication or point multiplication by a scalar s, as $Q = sP$. The security of this cryptosystem arises from the complexity of inverting this operation, known as the EC Discrete Logarithm Problem: it is hard to compute s given P and Q. Since EC multiplication is not only the most important operation in these cryptosystems but also the most computationally demanding, much research has been performed to achieve efficient algorithms and hardware structures for its implementation [CB08]. This paper presents a new method to perform EC multiplication over compact representations of EC points using a single coordinate $\tilde{x}_i$ in $GF(2^m)$. This allows us to efficiently implement EC cryptosystems by transmitting only $\tilde{x}_i$ instead of x_i and y_i, thus reducing the bandwidth by half without penalty in performance or cost. Although this compact representation (referred to as *collapsed representation* for the remainder of this paper) has been already proposed [Ser98], this paper presents for the first time an efficient method to use it without compromising the performance of the systems.

2 Proposed Point Multiplication Method

The EC point multiplication is performed by computing elementary operations over EC points, namely, addition and doubling. These operations are applied while running through the binary representation of the scalar. By using a *double and add* algorithm, a point multiplication with an n bit scalar requires n doubling and $h(s)$ addition operations, with $h(s)$ being the Hamming weight of the scalar.

The main drawback of applying this multiplication algorithm to the traditional representation (x_i, y_i) is that for each elementary operation (EC point doubling or addition), an inversion operation over a field element is required, which is the most computationally demanding operation. A solution for this problem is to use an alternative projective representation for a point, which introduces a third coordinate. In particular, the standard projective representation leads to a multiplication algorithm that allows us to obtain the resulting x coordinate from just the input x coordinate [LD99]. This property is particularly suitable to the collapsed representation which does not use the y coordinate. This standard projective representation maps an affine point (x, y) into a projective point (X, Y, Z) as $x \leftarrow X/Z$ and $y \leftarrow Y/Z$. The conversion from affine to projective representation is straightforward, since $Z = 1$ can be considered, but converting from projective to affine representations requires the inversion of Z. The algorithm in [LD99] is supported by the elementary addition (X_A, Z_A) and doubling (X_D, Z_D) operations for a projective representation, which are, respectively:

$$\begin{cases} X_D = X_1^4 + bZ_1^4 \\ Z_D = X_1^2 Z_1^2 \end{cases} ; \begin{cases} Z_A = (X_1 Z_2 + X_2 Z_1)^2 \\ X_A = xZ_A + X_1 Z_2 X_2 Z_1 \end{cases} \tag{2}$$

The final step of this algorithm is the computation of a function that converts the projective in affine coordinates, and is the only step where the input y coordinate is required, while computing the resulting y coordinate (y_R) [RHSPK07]:

$$y_R = y + (x + X_1/Z1)\,(xZ_1Z_2)^{-1}\left[(X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1Z_2)\right]. \tag{3}$$

The main goal of this paper is to positively answer the question: is it possible to embed the collapsed representation in this algorithm, in order to perform point multiplication in this representation without computing the y coordinate? Starting from the resulting x coordinate

$x_R = X_1/Z_1$, (3) can be rewritten as:

$$\frac{y_R}{x_R} = (X_1 + xZ_1)(X_2 + xZ_2)\left(\frac{1}{x_R Z_1 Z_2} + \frac{1}{x Z_1 Z_2}\right) + \frac{x^2}{x_R} + \frac{y}{x_R} + \frac{x^2}{x} + \frac{y}{x} + \frac{y}{x_R}. \tag{4}$$

Applying the trace operator $T(.)$ to (4):

$$T\left(\frac{y_R}{x_R}\right) = T\left(\frac{y}{x}\right) + T\left(\frac{(xZ_1 + X_1)\left(x(X_1Z_2 + X_2Z_1) + X_1X_2\right)}{xX_1Z_1Z_2}\right),\ T(\alpha) = \sum_{k=0}^{m-1} \alpha^{2^k}. \tag{5}$$

The trace operator is linear, outputs only one bit, and can be very efficiently computed with an exclusive *OR* operation [Ser98]. From the result in (5), one can conclude that it is possible to obtain the resulting $T(y_R/x_R)$ by adding $T(y/x)$ and a term that only depends on x. This means that one can compute the multiplication algorithm using the collapsed representation without computing the y coordinate at all.

3 Experimental Results

In order to properly evaluate the performance of the new point multiplication method with coordinate collapsing in a real application, cryptographic processors were implemented and tested on a FPGA device (Xilinx XC4VSX35). Table 1(a) presents the field operations schedule to compute the final step of the point multiplication algorithm without collapsing (3) and for the collapsed representation (5). Addition and squaring operations over the field relative costs can be disregarded for complexity evaluation. The collapsed representation requires only 1 extra multiplication compared to the traditional solution. Two of these processors implement a sequential approach of the point multiplication algorithm based on a microcoded solution [AaCS09], where instruction and data BRAMs are used together with an arithmetic circuit in order to perform one field operation at once. These two processors differ only in the instruction BRAM content, which corresponds to a program with and without coordinate collapsing, using the schedules presented in Table 1(a) for the final projective to affine conversion. The inversion required in the algorithm is computed with the Itoh-Tsujii method which is supported on several field squaring and multiplication operations [CB08].

Two other processors implement a parallel approach for the point multiplication. These processors are based on the work proposed in [AaCS08] but without the point addition logic: they possess three dedicated field multipliers and a divider, allowing to parallelize the EC multiplication algorithm with three field multiplications at once and also to parallelize field inversion and the multiplication in the projective to affine conversion. A parallelized version of the schedule in Table 1(a) was also used to obtain parallel versions with and without

Table 1: Scheduling of the conversion from projective to affine coordinates with and without collapsing.

(a) With and without collapsing schedulling.

Without Collapsing		With Collapsing	
$T_1 = xZ_2$	$T_2 = Z_1Z_2$	$T_1 = X_1Z_2$	$T_1 = T_1 + T_3$
$T_3 = T_1Z_1$	$T_3 = x^2$	$T_2 = xZ1$	$T_3 = X_1X_2$
$T_4 = T_3^{-1}$	$T_3 = T_3 + y$	$T_3 = T_1T_2$	$T_7 = xT_1$
$T_3 = X_1T_1$	$T_6 = T_2.T_3$	$T_4 = T_3^{-1}$	$T_1 = T_7 + T_3$
$x_R = T_5 = T_3T_4$	$T_1 = T_1 + T_6$	$x_R = T_5 = xT_1$	$T_7 = T_1T_2$
$T_2 = T_1 + X_2$	$T_2 = T_1T_4$	$T_6 = X_1T_5$	$T\left(\frac{y_R}{x_R}\right) = T(T_7T_4)$
$T_3 = xZ_1$	$T_1 = x + T_5$	$T_5 = T_6T_4$	
$T_3 = T_3 + X_1$	$T_3 = T_1T_2$	$T_2 = T_2 + X_1$	
$T_1 = T_2T_3$	$y_R = T_3 + y$	$T_3 = X_2Z_1$	

(b) Xilinx Virtex 4 Implementation Results.

	Parallel	Sequential
Without Collapsing	9,021 Slices 107.8 MHz 14,388 clk. cycles	1,095 Slices 148.7 MHz 201,618 clk. cycles
With Collapsing	8,331 Slices 103.5 MHz 14,388 clk. cycles	1,095 Slices 148.7 MHz 201,750 clk. cycles

coordinate collapsing. Table 1(b) summarizes the obtained experimental results. By executing the sequential version of the projective to affine conversion scheduling, the collapsing version requires one more multiplication, but 3 less additions and 1 less squaring. In total, a penalty of 132 additional clock cycles is observed for point multiplication regarding the no collapsing schedule. However, this penalty corresponds only to a performance degradation of 0.065%, which can be neglected when comparing with the whole point multiplication time. In the parallel implementations, it can observed that the no collapsing solution requires more area, obtaining similar operating frequencies. This is due to the y coordinate management, which requires more space to store it and more demanding routing. The Frequency/Slices metric of the no collapsing approach is 11.94 against 12.42 of the collapsing solution, meaning that the latter is more balanced in terms of area/time. The required clock cycles for both the solutions are the same, since the extra multiplication can be parallelized with the final inversion.

The obtained results suggest that the efficient collapsed representation has a minimal impact in the computational cost of the EC point multiplication, but allows us to reduce the communication bandwidth by half while significantly reducing the power consumption of embedded systems using EC cryptography.

4 Conclusions

The communication bandwidth and memory capacity requirements of an EC cryptosystem can be even further reduced by using collapsed representations of the elliptic curve points. This paper proposes a method to efficiently perform point multiplication, the most computationally intensive operation, embedding this collapsed representation, accomplishing minimal penalty in either performance or cost. This analysis is carefully supported with FPGA experimental results on a Xilinx Virtex 4 device. This allows EC cryptographic systems to store and communicate half the data typically required to represent an elliptic curve point, which is a significant advantage for systems such as smart cards and sensor networks nodes, where communication power is very limited and storage resources are scarce.

References

[AaCS08] S. Antão, R. Chaves, and L. Sousa. Efficient Ellitpic Curve Processor over GF(2^m). *International Conference on Field-Programmable Technology 2008, Proceedings*, December 2008.

[AaCS09] S. Antão, R. Chaves, and L. Sousa. Compact and Flexible Microcoded Elliptic Curve Processor for Reconfigurable Devices). *7th IEEE Symposium on Field-Programmable Custom Computing Machines 2009, Proceedings*, March 2009.

[BTn07] BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks. *BTnode Harware Reference*, 2007. http://www.btnode.ethz.ch/Documentation/ /BTnodeRev3HardwareReference.

[CB08] W.N. Chelton and M. Benaissa. Fast Elliptic Curve Cryptography on FPGA. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(2):198–205, February 2008.

[LD99] J. Lopez and R. Dahab. Fast Multiplication on Elliptic Curves over GF(2^m) without Precomputation. *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES 99, Proceedings*, 1717:316–327, August 1999.

[RHSPK07] F. Rodriguez-Henriquez, N. Saqib, A. Perez, and Ç. Koç. *Cryptographic Algorithms on Reconfigurable Hardware*. Springer, New York, NY, 2007.

[Ser98] G. Seroussi. Compact Representation of Elliptic Curve Points over F_{2^n}. *HP Laboratories Technical Report*, September 1998.