Combining Point Operations for Efficient Elliptic Curve Cryptography Scalar Multiplication

Cristian-Liviu Leca[1], and Cristian-Iulian Rîncu[1]
[1]*Military Technical Academy, Bucharest, Romania*

***Abstract*—Elliptic curve cryptosystems have gained increase attention and have become an intense area of research, mainly because of their shorter key length when compared to other public key cryptosystems such as RSA. Shorter key length brings advantages such as reduced computation effort, power consumption and storage requirements, making it possible to increase the available security for portable devices, smartcards and other power strained devices. ECC manages to cover all the significant cryptographic operations such as key exchange and agreement or digital signature with greater efficiency than previous systems. These operations rely heavily on point multiplication which is also the most time-consuming operation. This paper evaluates point operations (doubling, tripling, quadrupling, and addition) and proposes an algorithm for combining the operations in order to achieve faster scalar multiplication when compared to the standard algorithm for scalar multiplication of double and add.**

***Keywords*-cryptography; elliptic curve cryptography; scalar multiplication; point operations.**

I. INTRODUCTION

Elliptic curve cryptography (ECC) has gained popularity and acknowledgment since its introduction in 1985, mainly due to the increased difficulty of the mathematical problem that is the basis for elliptic curve cryptosystems [1]. Today, elliptic curve cryptography has become a commercial alternative to classic public key cryptosystem such as RSA and DSA, because for similar key sizes elliptic curve cryptosystem offer a greater deal of security. This makes EC cryptosystems with small key sizes (under 400 bits) suitable for use with constrained devices where lengthy key sizes (1024 bits) are not easily implemented because of limitations regarding computing power, storage and bandwidth [2]. Although ECC offers increased security compared to previous public-key cryptosystems, its main disadvantage is slower computation speed available with current algorithms. For example, an EC cryptosystem with a key size of 256 bits offers the same level of security as an RSA cryptosystem with a key size of 3072 bits [1].

Table I
KEY LENGTH COMPARISON FOR PUBLIC-KEY AND SYMMETRIC KEY CRYPTOGRAPHY

Symmetric-key	ECC	RSA/DLP
64 bit	128 bit	700 bit
80 bit	160 bit	1024 bit
128 bit	256 bit	3072 bit
256 bit	512 bit	15360 bit

The strength of EC cryptosystems security is based on the elliptic curve logarithm, represented by the difficulty of solving the discrete logarithm problem (DLP) in a group defined by the points of an elliptic curve over a finite field [2].

The main steps of EC cryptosystems (key agreement, signature generation, signing and verification) are all based upon scalar multiplication, which is also the most time consuming and resource demanding operation. Scalar multiplication makes use of point addition repeatedly over a finite field (GF), as represented in (1), where k is an integer belonging to a finite field $F(p)$ and P is a point on the elliptic curve defined over the finite field $F(p)$.

$$kP = \sum_{i=1}^{k} P \tag{1}$$

The inverse operation, which means finding the scalar k when the points P and Q are known, represents the Elliptic Curve Discrete Logarithm Problem (ECDLP). The increased security of EC cryptosystems assumes that the ECDLP has no published sub exponential time solving algorithm for a properly selected curve and group [3]. This means that an attacker trying to obtain the scalar using the points, will not be able to obtain it in a shorter time than the one assumed by the designers of the cryptosystems.

The increased computation time of EC cryptosystems is also reflected in the encryption process. Improving the efficiency of scalar multiplication is the main way to greatly increase the overall performance of an ECC cryptosystem. The techniques proposed [1,2,4] are mainly based on the different methods for factoring the scalar k. The common method for achieving this is the *double and add* algorithm [5], which is based on the binary representation of k, and repeated point doublings for even scalars and additions when the factoring of k is odd. The algorithm is represented in Fig. 1.

```
1: Q = O
2: for i from n – 1 to 0 do
3:         Q = 2Q (double)
4:         if k_i = 1 then Q = Q + P (add)
5: return Q
```

Figure 1. Double and add algorithm for scalar multiplication

Point doubling and addition involve operations over prime or binary fields. Addition, subtraction and multiplication of integers are negligible operations when compared to field addition, multiplication, squaring or inversion [6]. The addition of a point P to Q requires solving three equations which contain one inversion, one squaring and two field multiplications of which the most time-consuming is the inversion. Newly proposed methods aim at replacing the double or add operation

978-1-4799-2385-4/14/$31.00 ©2014 IEEE

with more efficient operations that avoid a large number of inversions for factoring the scalar k [6].

In this paper we propose and evaluate an efficient algorithm for combining simple operations such as point tripling (*3P*), quadrupling (*4P*), double and add (*2P+Q*), in order to obtain a significantly less time-consuming method for scalar multiplication, which aims at reducing the number of inversions required for the operation. The paper is organized as follows: the second section provides an insight into background elliptic curve mathematics; the third and fourth sections deal with point operations by analyzing the cost and their efficiency; the fifth section presents the proposed algorithm and the results obtained. Finally, the sixth section brings forth the conclusions that we have drawn.

II. ECC MATHEMATICS

Elliptic curves can be defined over prime fields or over binary fields. If q is a prime number or prime power, then F_q will denote the field that contains exactly q elements. When the greatest common divisor of q and b is 1, the elliptic curve over the field F_q is expressed by the following equation:

$$E : y^2 = x^3 + ax + b \tag{2}$$

with a,b in F_q and $4a^3 + 27b^2 \neq 0$. While the general curve equation is defined as:

$$E_2 : y^2 + xy = x^3 + ax^2 + b \tag{3}$$

The group used for EC cryptosystems is the group of points on a curve over the field F_q. In affine coordinates the points are represented by the pair (x,y) where x and y belong to the field and satisfy the curve equation. Also, the point at infinity O belongs to the group and serves as the identity for the group law.

The addition of any two points can be attained using the following calculus algorithm [3], considering the curve was defined using (2):

Let $P(x_1,y_1)$ and $Q(x_2,y_2)$ be two different points on a curve.
If either point is O then the result is the other point.
If $P = Q$, use the double algorithm.
If $x_1 = x_2$ *and* $y_1 \neq y_2$, $P + Q = O$.
If $P \neq Q$ then $P + Q = R(x_3,y_3)$, where
$\lambda = (y_1 + y_2)/(x_1 + x_2)$
$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$
$y_3 = \lambda(x_1 + x_3) + x_3 + y_1$

Figure 2. $P+Q$ Algorithm

While the doubling of a point [3] is attained by the use of the following algorithm:

Let $P(x_1,y_1)$ be a point on the curve.
If $x_1 = 0$, then the result of $2P$ is O.
If $x_1 \neq 0$, $2(x_1,y_1) = R(x_3,y_3)$, where
$x_3 = \lambda^2 + \lambda + a$
$y_3 = x_1^2 + (\lambda + 1)x_3$
$\lambda = x_1 + y_1/x_1$

Figure 3. $2P$ Algorithm

By expressing the cost of the operations by the number of multiplications (M), additions (A) and inversions (I) required, the result will be: 1I + 3M + 6A for the add operation and 1I + 7M + 4A for the double operation, resulting in similar efficiency, because the number of inversions is equal.

III. POINT OPERATIONS

Using the cost of the previous operations we can study the efficiency of other proposed methods for point tripling, quadrupling, double and add (2P+Q).

A. Double and add *2P+Q*

The double and add method presented as in [6] is attained by a supplementary addition with the point P inside of the addition operation as in Fig 4. (Note: this operation should not be confused with the scalar multiplication algorithm presented in Fig. 1)

Let $P(x_1,y_1)$ and $Q(x_2,y_2)$ be two different points on a curve.
If $x_1 = x_2$ and $y_1 \neq y_2$, $2P + Q = P$.
$X = (x_2 - x_1)^2$; $Y = (y_2 - y_1)^2$
$d = X(2x_1 + x_2) - Y$
If $d = 0$, $2P + Q = O$.
$D = (x_2 - x_1)$; $I = D^{-1}$
$\lambda_1 = dI(y_2 - y_1)$
$\lambda_2 = 2y_1X(x_2 - x_1)I - \lambda_1$
$2P + Q = R(x_3,y_3)$ where:
$x_3 = (\lambda_2 - \lambda_1)(\lambda_2 + \lambda_1) + x_2$
$y_3 = (x_1 - x_3)\lambda_2 - y_1$

Figure 4. $2P+Q$ Algorithm

The resulting cost will be of 1I +13M + 11A that is significantly lower than the use of two additions with a cost of 2I + 6M + 12A.

B. Point tripling *3P*

Point tripling [6] is similar to the double and add method previously presented and is attained as presented in Fig. 5.

Let $P(x_1,y_1)$ be a point on the curve.
If $y_1 = 0$, then the result of $3P$ is P.
$X = (2y_1)^2$; $Z = 3x_1^2 + a_4$; $Y = Z^2$;
$d = X(3x_1) - Y$;
If $d = 0$, $3P = O$.
$D = d(2y_1)$; $I = D^{-1}$
$\lambda_1 = dIZ$
$\lambda_2 = X^2I - \lambda_1$
$R(x_3,y_3)$ where:
$x_3 = (\lambda_2 - \lambda_1)(\lambda_2 + \lambda_1) + x_1$
$y_3 = (x_1 - x_2)\lambda_2 - y_1$

Figure 5. $3P$ Algorithm

The total cost of the operation for point tripling is also similar to that of double and add operation: 1I + 16M + 8A, which makes for close efficiency.

C. Point quadrupling *4P*

Point quadrupling [7] is attainable with a cost of 1I+26M+13A, using the following algorithm.

Let $P(x_1,y_1)$ be a point on the curve.
If $y_1 = 0$, then the result of $4P$ is P.
$A = x_1^2$; $B = 3A + a$; $C = 2y_1^2$; $D = C^2$
$E = (x_1 + B)^2 - A - D$; $d = B(3E - B^2) - 2D$
If $d = 0$, $4P = O$.
$F = 2y_1d$; $I = F^{-1}$; $\lambda_1 = dIB$
$x_3 = \lambda_1^2 - 2x_1$; $y_3 = \lambda_1(x_1 - x_3) - y_1$
$H = 3x_3^2 + a$; $\lambda_2 = 2DIH$
$4P = R(x_4,y_4)$ where:
$x_4 = \lambda_2^2 - 2x_3$; $y_4 = \lambda_2(x_3 - x_4) - y_3$

Figure 6. *4P* Algorithm

IV. POINT OPERATION EFFICIENCY

In order to determine the overall efficiency of the previously presented point operations we proposed the following equations which represent the cost of the point operation related to the value of the scalar.

For the $2P+Q$ operation, the equation is:

$$\frac{aI + bM + cA}{x + 1} \tag{4}$$

While for xP operations, the equation is:

$$\frac{aI + bM + cA}{x} \tag{5}$$

Using the equations the following costs of each operation have resulted:

Table II
COST OF OPERATIONS

Operation	Equivalent scalar	Cost	Equivalent cost
2P	2	1I + 7M + 4A	0.5I + 3.5M + 2A
P + Q	2	1I + 3M + 6A	0.5I + 1.5M + 3A
2P + Q	3	1I + 13M + 11A	0.33I + 4.33M + 3.66A
3P	3	1I + 16M + 8A	0.33I + 5.33M + 2.66A
4P	4	1I + 26M + 13A	0.25I + 4.5M + 3.25A

As the scalar increases the overall efficiency of the operations also increases, proving the need for a scalar multiplication algorithm that will make use of the previously presented operations.

V. SCALAR MULTIPLICATION ALGORITHM

A. Proposed Algorithm

Considering the fact that the operations of double and that of add will result in a high cost for scalar multiplication due to the increased number of inversions, we propose the following algorithm that will make efficient use of the previous point operations in order to achieve faster computation times.

As long as the scalars factorization is achieved by one of the integers: 4, 3, 2 in the presented order, the algorithm will make use of the corresponding operation. If an odd factor not divided by 3 is found then the $2P+Q$ operation is used.

```
Input: scalar k and point P.
Output: R = kP.
Variables: tk, i, tp, c, l, array[]
tk = k
While tk <> 1
        If tk mod 2 = 1 and tk mod 3 <> 0 then
                tk = (tk – 1)/2 , array[i=i+1]=1
        While tk mod 4 = 0
                tk = tk/4, array[i=i+1]=4
        while tk mod 3 = 0
                tk = tk/3, array[i=i+1]=3
        while tk mod 2 =0
                tk = tk/2, array[i=i+1]=2
tk = k, tp = P, l = i-1
for i=l, i > 0, i=i-1
        c=array[i]
        switch c
                case 1 then tp = 2tp + P
                case 2 then tp = 2tp
                case 3 then tp = 3tp
                case 4 then tp = 4tp
R=tp
```

Figure 7. Proposed Algorithm

By using the proposed algorithm the resulting number of inversions required for scalar multiplication will be significantly lower as exemplified in the following table:

Table III
NUMBER OF INVERSIONS REQUIRED FOR SCALAR MULTIPLICATION

Scalar	1026	1753	3442	9216	10519	49290
No. of Inversions for double and add algorithm	11	16	15	14	19	20
No. of Inversions for proposed algorithm	7	7	11	7	9	11

When trying to reduce computation time and increase the overall performance for ECC operations, it is also important to find an adequate solution for implementing the operations [8]. For that we organized the field, elliptic curve and scalar operations on different levels required for the functionality of an ECC based encryption algorithm.

In order to prove the effectiveness of the algorithm we have implemented the proposed algorithm presented in Fig. 7 together with the double and add algorithm presented in Fig. 1, on a PIC32MX795F512L MCU with 80MHz clock speed, 512Kb ROM and 128Kb RAM, using the C programming language.

Aiming at obtaining increased performance for the point operations and finding the adequate solutions for implementing the mathematic functions behind, the following model was considered. On level 1 we included field operations (multiplication, addition, inverse) modulo the order of the field $F(p)$. On level 2 we placed the point addition, doubling, tripling and quadrupling all of which are realized inside the cyclic group determined by the elliptic curve. Scalar multiplication was placed on the third level because it makes use of lower level operations together with the scalar k, which belongs to the field $F(p)$. On level 4 we considered protocols based upon ECC, such as ECDSA [9] (Elliptic Curve Digital Signature Algorithm), ECDH (Elliptic Curve Diffie Hellman).

B. *Results*

The first evaluation was based on the elliptic curve determined by the parameters: *p=15559*, *a=1*, *b=1*, resulting the following scalar multiplication durations for different scalars and the same point that was chosen to be 4:

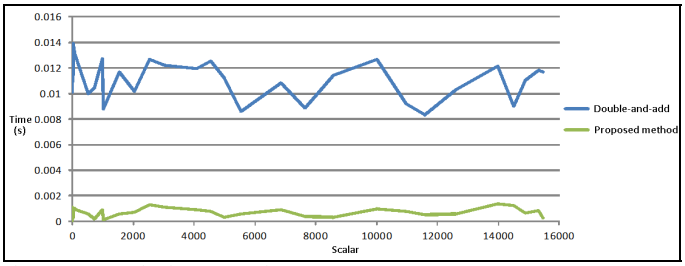

Figure 8. Results for same point

The proposed algorithm manages to reduce the computation requirements by eight to ten times. An a priori advantage in obtaining these results was the fact that the chosen point was of small dimension, meaning that looking up the point will not significantly influence the operation.

Then we evaluated the algorithms using the previous curve and different scalars together with randomly chosen points, resulting the following:

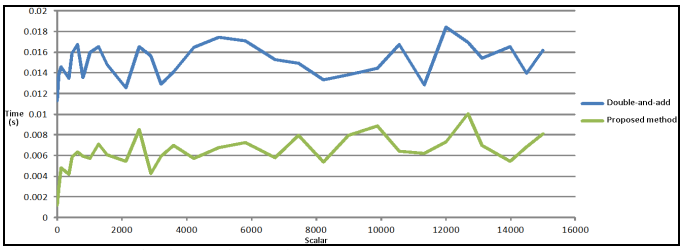

Figure 9. Results for random point

The performance of the proposed algorithm manages to maintain a ratio between two and three for the decrease in computation time, compared to the double and add algorithm.

Finally we tested two implementations of the ECDSA [9], on fields of increasing order, one using the proposed algorithm and the other using the double and add algorithm with the following results:

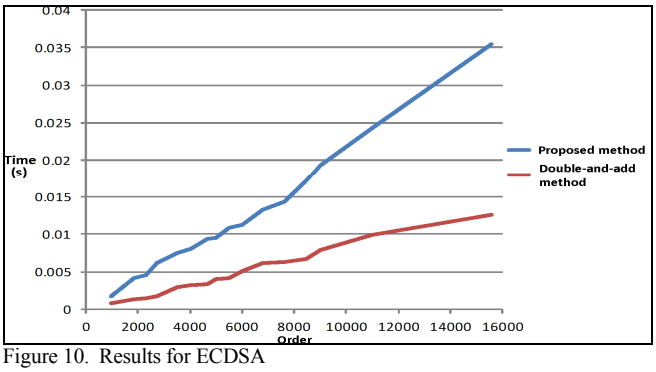

Figure 10. Results for ECDSA

The proposed algorithm manages to complete the signing operation two to three times faster than the double-and-add algorithm on fields of different order, obtaining similar performance when implemented inside the ECDSA protocol as was obtained during stand-alone testing.

VI. CONCLUSIONS

Implementing ECC cryptosystems can offer proper solutions for secure applications based on limited resources, but as we presented, some of the operations implied can lead to highly complex cryptosystems which can seriously reduce the performance of real-time applications that are sensitive to delays. This fact implies the necessity of finding new solutions for improving essential ECC operations such as scalar multiplication. The proposed algorithm manages to improve the overall performance of scalar multiplication and reduce the complexity of the operation by lowering the number of inversions involved compared to the double and add algorithm. This is achieved by efficiently combining the previously presented point operations. The proposed algorithm was also implemented on a PIC32 MCU in order to prove that it leads to better results on a constrained device when compared to the classic algorithm.

REFERENCES

[1] J. Lopez, and R. Dahab, An Overview of Elliptic Curve Cryptography, Institute of Computing, State University of Campinas, 2000.

[2] M. Brown, D. Hankerson, J. Lopez, and A. Menezes, Software Implementation of the NIST Elliptic Curves over Prime Fields, Certicom Research, 2001.

[3] P. Stavroulakis, and M. Stamp, Handbook of Information and Communication Security, Springer, 2010.

[4] E. Wenger, and M. Hutter, "Exploring the Design Space of Prime Field vs. Binary Field ECC-Hardware Implementations", Information Security Technology for Applications, Springer, 2012.

[5] C. Cockrum, Implementation of an Elliptic Curve Cryptosystem on an 8-bit Microcontroller, Available: http:// www.cockrum.net.

[6] M. Ciet, M. Joye, K. Lauter, and P. L. Montgomery, "Trading Inversions for Multiplications in Elliptic Curve Cryptography", Designs, Codes and Cryptography, vol. 39, no.6, 2006.

[7] L. Duc-Phong, "Fast Quadrupling of a Point on Elliptic Curves Cryptography", SoICT, 2012.

[8] F. Rodriguez-Hernandez, N. A. Saqib, A. Diaz-Perez, and C. Kaya Koc, Cryptographic Algoritms on Reconfigurable Hardware, Springer, 2006.

[9] Standards for Efficient Cryptography: Elliptic Curve Cryptography (SEC 1), Certicom Research, 2009.