Algorithms for Protection of Elliptic Curve Point Multiplication Against Side-Channel Attacks

George STEPHANIDES[1], Nicolae CONSTANTINESCU[2], Mirel COŞULSCHI[2]

[1] Dept. of Applied Informatics, Macedonia University,
156 Egnatia str., 540 06 Thessaloniki, Greece
steph@uom.gr

[2] Faculty of Mathematics and Computer Science,
Department of Computer Science,
University of Craiova, ROMANIA
nikyc@central.ucv.ro, mirelc@central.ucv.ro

Abstract. Point multiplications is a sensitive matter when it comes to security of cryptosystems based on elliptic curves. Computing the result of such an operation has considered various methods of representation for points subject to the operation. Also for efficiency reasons, often implementations of such cryptosystems would choose special elliptic curves. This paper aims at achieving same efficiency properties by using a novel point multiplication method which is more general and can be used with recommended elliptic curves in SECG and NIST documents referring to elliptic curves standards.

Keywords: public key cryptography, elliptic curves point multiplications

Math. Subjects Classification 2000: 11G05, 11G07, 11H52.

1 INTRODUCTION

Elliptic curves cryptosystems are an important part of today's cryptography due to the fact that one can achieve the same level of security as for a RSA system with less operations and less storage capacity. This leads to usage of such cryptosystems in smaller and processor challenged devices such as mobile phones and handsets. This is why elliptic curves are an important mathematical and cryptographical study subject.

Side-channel attacks rely on power consumption [10] and timing [9] as indicators for complexity of computations required for point multiplication on implementations of cryptosystems based on elliptic curves. The computational issue here is obtaining a result of the form:

$$R = [e]P$$

where R is the result, $[e]$ is a integer value, always secret, be it ephemeral or long-term used, also e is considered to be the key, while P is a point on the elliptic curve used with the cryptosystem.

The class of elliptic curves used in cryptographic applications require that they have a prime-order sub-group. The order of this sub-group is denoted by P. Points of order p are used only.

Following measures can be employed to randomize computations and make power analysis a harder task for the attacker:

- If the cryptosystem uses projective coordinates for point representation, then the input point can be transformed in a random equivalent representation. (projective coordinates is the most often choice because of efficiency reasons [1],[7].
- The product $[e]P$ can be expressed using a random point Q as

$$R = [e](P - Q) + [e]Q$$

- The product $[e]P$ can be expressed using a random integer n as

$$R = [e + np]P$$

  or

$$R = [e - n]P + [n]P$$

The inserted randomness considered above is meant to reduce the risk of differential attacks which use correlations from multiple computations. However if direct interpretation of readings is possible above measures can become futile. Simple side-channel analysis works with implementations that use a straightforward approach to point multiplication. This is because the bits of the integer e leave quite an easy trace in the power consumption usage, which can be interpreted, along with the fact that P is publicly available this leads to the disclosure of the secret integer e if a double-and-add algorithm is used. Algorithms like m-ary or sliding window method may obscure the bits of e but according to [12] this may still reveal information.

Considering above issues, point representation seems the best solution at hand. Liardet and Smart [12], and Joye and Quiaquater [8] proposed a set of curves and special representations that can be used in such manner that the same formula can be used for both addition and multiplication operations. Their solution however is not cost-effective and induces performance drawbacks.

In [13] proposes a point multiplication algorithm which is improved by Okeya and Sakuray in [15] and then is used in [16] for suitable curves over odd characteristic fields, where the usual group operations are replaced by certain special operations working with triplets of points with y-coordinates omitted. This method makes the retrieval of bits from e much harder.

Details of the above methods are not discussed here, however they all have the disadvantage of inusability with NIST and SECG recommended curves from [14] and [4]. This leads to major drawbacks in what concerns interoperability.

This paper proposes a method by which limitations of specific usage of curves is removed by using a uniform pattern both for addition and multiplication. Because of this feature the method can help interoperability of systems.

The method uses more addition operations than the standard 2^w-ary point multiplication algorithm. Despite this, dummy additions used in other algorithms to achieve a fixed pattern of point doubling and addition [6] are avoided, reasons for this are given in next section.

Randomly choosing the integer e avoids failure of the considered method. Failure occurs in cases where addition involves actually a point doubling or the point at infinity. These situations are potentially clearly visible through side-channel analysis.

According to [16] methods of protection presented in the start of this section can be combined with the use of projective coordinates. We also recommend integrating the presented method with randomization techniques above mentioned.

2 SECURITY OF ELLIPTIC CURVES OPERATIONS

Security analysis often uses a model in order to test side-channel information leakage. While it is practically impossible to analyse all possible information leakage, often this model is aimed at specific aspects, configurations or implementations.

Before presenting the method, information leakage is considered at a lower level. Subsection 2.1 discusses special cases of point operations that should be avoided. Subsection 2.2 discusses the importance of using randomized projective coordinates and certain extended point representations. Also this subsection questions the insertion of dummy point additions to achieve uniform behavior.

2.1 POINT OPERATIONS

Side-channel analysis is the operation which involves gathering of information concerning timing of computations and power consumption of such operations. Values that these indicators provide may be interpreted to obtain a certain order of operations involved in a cryptosystem. To conceal this order a careful devised algorithm should use point doubling and addition operations in order to create a uniform pattern which should be independent of the specific multiplier used in considered operations. Of course there are exceptions, and these situations should be treated in a special manner at the time of their occurrence. These are presented in the following statements:

- Point doubling $[2]A$ requires conditional statements for the case that A is the point at infinity or that A is a point of order two. If these cases are avoided, then, expressed in field operations, point doubling runs as a fixed routine.

- Point addition $A + B$ requires conditional statements for the case that one of the points is the point at infinity, or that A coincides with B, or that one point is the inverse of the other. For other cases, it too can be implemented as a fixed routine.

Details of the sequences of field operations used for point doubling and point addition depend on the underlying field (odd characteristic vs. characteristic 2) and the choice of point representations (e.g. either affine coordinates or one of multiple styles of projective coordinates, according to [5]), so implementations may vary widely. The essential observation is that the respective algorithm always behaves the same as long as the above special cases are avoided.

2.2 FIELD OPERATIONS

An important observation is that when an attacker analyses the side-channel information he does not have immediate access to the factors involved in a operation. However it is prudent to say that not all operations look the same, and this is the basic idea for side-channel attacks. Inserting randomization techniques into one's protocol or any other cryptosystem based on elliptic curves is a good idea. This is combined with the usefulness of projective coordinates [6],[16]. Take for example Jacobian projective coordinates, which are triplets of the form (X, Y, Z) with $Z = 0$, they represent affine points $(X/Z^2, Y/Z^3)$; then for any field element $\epsilon \neq 0$,$(\epsilon^2 X, \epsilon^3 Y, \epsilon Z)$ is a representation of the same point on the curve. Randomization makes it difficult for an attacker to guess the values obtained by using a randomly chosen ϵ.

Point doubling or point addition using projective coordinates results in a point represented with a Z-coordinate that is the product of the Z-coordinate(s) of the input point(s) and a short polynomial involving one or more other coordinates of the input points; thus the output point is again in a randomized representation.

Randomization makes it difficult for an attacker to guess or imply that a certain operation involving known points is taking place at a certain point in time. Still the attacker may observe the same operation reoccurring if the same field operation is execute several times throughout the computation. Even if the attacker cannot obtain the factors involved in the operation we still want to mask this as this in some cases may be considered an important information leakage.

Point multiplication, $R = [e]P$, is performed in stages by a great part of existing algorithms, these stages are as follows:

Precomputation stage: First, independently of the specific multiplier e, certain small multiples of P are computed and stored in a table.

Evaluation stage: Second, the product $[e]P$ is evaluated as follows: A variable A is initialized to one of the table values; then, many times, A either is doubled or a table value is added to A, replacing the previous value of A. Finally, A contains the result $[e]P$.

Algorithms that adhere to this structure should use a table to store the value of Z^2 by using a tuple of the form (X, Y, Z, Z^2) and store it in the table, instead of the classical approach of storing (X, Y, Z) and then perform the precomputation phase when needed.Storing the tuple (X, Y, Z, Z^2) is called using *extended point representation*, if the usage of the suggested method of extended point representation presented in [1], and [7] is neglected an attacker may be able to tell which point additions use the same entries in the table.

Another problem with algorithms that follow the above structure is that they need to insert dummy additions in their structure in order to achieve a fixed pattern of doubling and additions. These additions involve additions of a table value to the variable A, and then discarding the result. The main issue with dummy additions is that the value of A is never changed, this becomes a major problem when working with Jacobian projective coordinates. This is because each point operation requires squaring the Z-coordinate of A, if two consecutive point operations are performed the same value of Z will be used leading to the same squaring being performed. This is why dummy operations should be avoided.

It is possible, but inefficient, to randomize the point representation after each point operation. (If this is done, dummy additions are no longer a problem.) For point multiplication algorithms of the above form, it is more practical to use randomization just twice or once: If the precomputed table of multiples of P is stored in projective coordinates, then the representation of P should be randomized before the table is computed. Also at the beginning of the second stage, after the initial value has been assigned to A, the representation of A should be randomized. If the table of multiples of P is stored in affine coordinates (to speed up the evaluation stage by using mixed addition of affine and projective points [5]), then the first randomization obviously is not necessary.

3 MULTIPLIER RECORDING PROVIDING RESISTANCE AGAINST SIDE-CHANNEL ATTACKS

We show how to perform point multiplication $[e]P$ in a way such that doubling and additions occur in a fixed pattern in order to provide resistance against side-channel attacks. Subsection 3.1 describes an algorithm for recoding the multiplier e into a special signed-digit encoding. In section 3.2, we discuss the multiplication algorithm implied by this encoding. Section 3.3 shows that, unless e is ill-chosen, this point multiplication algorithm indeed limits information leakage as intended.

3.1 RECORDING ALGORITHM

Let the positive integer e be given in 2^w-ary digits where $w \geq 2$ is a small integer: That is,

$$e = \sum_{i=0}^{k'} b'_i \cdot 2^{wi}$$

with $b_i \in \{0, 1, ..., 2^w - 1\}$. We demand that k. be chosen minimal, i.e. $b'_{k'} \neq 0$. For $i > k$, we define that $b'_i = 0$. We will show how to convert this into a representation

$$e = \sum_{i=0}^{k} b_i \cdot 2^{wi}$$

such that $b_i \in \{-2^w, 1, 2, ..., 2^w - 1\}$. This means that we disallow digit value 0 and instead introduce the new value -2^w. Intuitively, the recoding algorithm replaces 0 digits by -2^w and increments the next more significant digit to adjust the value. It is easy to see that bk must be positive (otherwise e would be negative) and that the representation of e needs to grow by at most one digit, i.e. $k = k'$ or $k = k' + 1$. We express the recoding algorithm recursively, using auxiliary values c_i and t_i such that $0 \leq c_i \leq 2$ and $0 \leq t_i \leq 2^w + 1$: Let

$$c_0 = 0,$$

and for $i = 0, ..., k' + 1$, let

$$t_i = b'_i + c_i$$

and

$$(c_{i+1}, b_i) = \begin{cases} (1, -2^w) \; if \; t_i = 0 \\ (0, t_i) \quad if \; 0 < t_i < 2^w \\ (2, -2^w) \; if \; t_i = 2^w \\ (1, 1) \quad if \; t_i = 2^w + 1. \end{cases}$$

Note that we always have $c_{i+1} \cdot 2^w + b_i = t_i$.

If $b_{k'+1} = -2^w$, then $e = \sum_{i=0}^{k+1} b_i \cdot 2^{wi}$ in this case, we set $k = k' + 1$. Otherwise, we have $e = \sum_{i=0}^{k'} b_i \cdot 2^{wi}$ and $b'_k \neq -2^w$, and we set $k = k'$. Observe that if $b_k = 1$ (and $k > 0$), then $b_{k-1} \neq -2^w$. As a measure against side-channel attacks on the recoding algorithm itself, assignments can be implemented by table lookups instead of using conditional statements. Storing the converted representation of e requires almost no additional memory compared with the original representation: Digits with value -2^w can be encoded as a pattern of bits all set to zero. If w is understood, then this new representation can be stored as an ordinary integer. (Leading 0 digits can simply be ignored because bk is never -2^w.) This integer is at most two bits longer than e; the maximum possible length growth occurs if $k = k' + 1$ and $b_k = 2$. It may be desirable to ensure that the encoded form has a predetermined maximum index k. For our point multiplication application, if kw is large enough compared with the binary length of p, this is easy to do: If necessary, adjust e by adding p or a multiple of p. If a random multiplier is required and a uniform distribution is not essential, then random bits can be used directly to generate the recoded form of e. In this case, too, it should be ensured that $b_{k-1} \neq -2^w$ if $b_k = 1$.

3.2 POINT MULTIPLICATION ALGORITHM

Remember that we want to compute $[e]P$ where point P should have order p, p being a large prime that divides the order of the elliptic curve $E(\mathbb{F}_q)$. For our point multiplication algorithm to work as intended, $ord(P)$ may be any positive multiple of p.

If point P can be chosen by the attacker, we must be aware that it might have incorrect order. In case that $\#E(\mathbb{F}_q) = p$, then (besides testing that P is indeed a point on the curve) we just have to verify that P is not the point at infinity, $\mathcal{O}$. Otherwise, we need an additional sanity check. Let h be the cofactor of the elliptic curve, i.e. the integer $\#E(\mathbb{F}_q)/p$. Curves used for cryptography are usually selected such that h is small, e.g. $h \leq 4$ as required by [3]. Thus $[h]P$ can be computed quickly; if it is not $\mathcal{O}$, then we know that p divides $ord(P)$. If P has incorrect order, computing $[e]P$ must be rejected. Otherwise, given a representation

$$e = \sum_{i=0}^{k} b_i \cdot 2_{wi}$$

of e as determined by the recoding algorithm of 3.1, $[e]P$ can be computed as follows:

Assume that points $P, [2]P, [3]P, ..., [2w-1]P, [-2^w]P$ have been stored in a table. Let $e_j = \sum_{i=j}^{k} b_i \cdot 2^{w(i-j)}$ for $j = 0, ..., k$. The goal is to compute $[e_0]P = [e]P$. We can determine $[e_k]P = [b_k]P$ simply by table lookup. Then, to compute $[e_j]P$ for j going down to 0, we can use that $e_j = 2^w \cdot e_j + 1 + b_j$ and thus $[e_j]P = [2^w]([e_{j+1}]P) + [b_j]P$, where $[b_j]P$ again is available by table lookup. This is essentially the 2^w-ary algorithm for computing $[e]P$, except that we use an unusual set of digits. The procedure is given in algorithm 3.2. We show a high-level description of the algorithm; as explained in subsection 2.2, implementations should use randomized projective coordinates, and values computed in the precomputation stage should be stored in extended point representation. The algorithm requires exactly $2^{w-1} + kw$ point doubling and $2^{w-1} - 1 + k$ point additions. (Inverting $[2w]P$ to obtain $[-2^w]P$ is essentially free in elliptic curves.)

3.3 UNIFORMITY OF THE POINT MULTIPLICATION ALGORITHM

It is apparent that algorithm 3.2 uses doubling and additions in a regular pattern. As discussed in Section 2, to show that the algorithm has uniform behavior and thus is suitable for limiting information leakage during the computation of $[e]P$, we also have to show that the following special cases of point doubling and point addition can be avoided:

- $[2]\mathcal{O}$
- $[2]A$ where $ord(A) = 2$
- $A + O$ or $O + B$
- $A + A$
- $A + (-A)$

We have required that $ord(P)$ be a positive multiple of p. Without loss of generality, here we may assume that $ord(P) = p$. (Otherwise, multiply P by the cofactor $h = \#E(\mathbb{F}_q)/p$ to obtain a point of order p. In the algorithm performed for P, the above special cases can occur only when one of them would occur

in the algorithm performed for $[h]P$; in particular, $ord(A) = 2$ would imply $[h]A = \mathcal{O}$, so points of order 2 are not an issue if we can show that the point at infinity can be avoided.) Then, as 2^w can be expected to be much smaller than p, all precomputed points $P_{b_j} = [b_j]P$ in algorithm 3.2 will be of order p. Thus, initially we have $A \neq \mathcal{O}$; and as long as we avoid additions of the form $A + (-A)$, it will stay like this. So what remains to be checked is whether in the evaluation stage of algorithm described in 3.2; we can avoid that $A = [b_j]P$ or $A = [-b_j]P$ before an addition takes place. With e_j defined as in section 3.2, the point addition step in the second loop of this algorithm performs the point addition $[2^w \Delta e_{j+1}]P + [b_j]P$. Thus we are save if

$$2^w \neq e_{j+1} \pm b_j (\text{mod } p).$$

Since $\mid e_j \mid \leq 2^{(1+k-j)w}$ and $\mid b_j \mid \leq 2^w$, for many indices j we have $\mid 2^w \cdot e_{j+1} \mid + \mid b_j \mid < p$, meaning that reduction modulo p does not matter and it suffices to check whether $2^w \cdot e_{j+1} \leq \pm b_j$. The largest index to consider is $j = k - 1$: Can we be sure that $2^w \cdot e_k \leq \pm b_{k-1}$? Indeed we can, as $e_k = b_k$ and if $b_k = 1$, then $b_{k-1} \neq -2^w$ (see section 3.1). It follows that $e_j \cdot 2^{(k-j)w}$, which shows that the incongruence is satisfied for indices $j < k - 1$ too as long as we do not have to consider reduction modulo p. For indices j so small such that this modular reduction is relevant, we argue that if e is sufficiently random, then the probability of picking an e such that the above incongruence is not satisfied can be neglected: It is comparable to the probability that e mod p can be guessed in a modest number of attempts, which can be presumed to be practically impossible.

4 EFFICIENCY ANALYSIS

Efficiency of presented algorithm and similar methods is measured in the number of field operations needed by the algorithm to compute each addition or point doubling. For example the Joye-Quisquater algorithm needs a number of 12 multiplications. This result is the expected number of operations for the computation of the product $[e]P$ per bit of e. The Liardet-Smart method requires 16 field operations each addition or doubling [11]. Because of this we will take into consideration only the Joye-Quisquater algorithm in our efficiency comparisons.

The most efficient algorithm in use today is the point multiplication algorithm based on signed windows [12], if $2^w - 1$ points are precomputed then the number of field multiplications is $(1 + \frac{1}{w+3}) \cdot 12$, and this estimation does not include precomputation effort. For multipliers whose length l is between 120 and 336 bits, the expected number of operations required by the signed window method is minimized when $2^3 - 1 = 7$ points are precomputed; the expected number of field operations then is about

$$(7 + \frac{7}{6} \cdot l) \cdot 12 = 84 + 14 \cdot l$$

which includes the precomputation effort.

For elliptic curve cryptography over fields of odd characteristic, curves are usually chosen such that a doubling can be done in 8 field multiplications and an addition can be done in 16 field multiplications using Jacobian projective coordinates [7]. (Further speed-ups are possible by using mixed coordinates for faster point addition; see [5] and [2].) With our method of section 3, using an extended point representation for precomputed points as explained in section 2.2, one additional field multiplication is required for each precomputed point, but only 15 field multiplications are needed for each evaluation stage point addition. Thus, if we use $w = 4$, we need $8 + 14 \cdot 15 = 11.75$ field multiplications for each bit of e (not including the effort to precompute 15 points). This is less than 12, so for sufficiently long e this method will be faster than the Joye-Quisquater method.

Choosing $w = 4$ minimizes the number of field multiplications for scalars of length l between 84 and 277 bits; it is about

$$192 + 11.75\ l,$$

including $2^3 \cdot 8 + (2^3 - 1) \cdot 16 + 2^4 = 192$ multiplications for precomputation using extended point representation (but neglecting the small additional cost for randomising projective coordinates). Compared with the Joye-Quisquaterapproach, this is nearly 10% faster even for multipliers as short as 120 bits. The method of [16] for curves having a Montgomery form is more efficient than any of the other methods: It needs only 11 field multiplications per bit [15], and as it is directly based on the binary point multiplication algorithm, it does not involve any precomputation.

5 CONCLUSION

We have presented a method for elliptic curve point multiplication that can be shown to provide security against side-channel attacks. The algorithm uses a special signed-digit encoding to ensure that point doublings and point additions occur in a uniform pattern. No dummy additions are required; implementing the method using randomised projective coordinates and storing precomputed points in extended point representation limits information leakage to a minimum. While the method of Okeya and Sakurai [16] for curves with Montgomery form is more efficient, the approach of the current paper is much more general: Unlike various previous proposals, it is applicable to the recommended curves of [14] and [4].

References

[1] **I. F. Blake, G. Seroussi and N. P. Smart**, Elliptic Curves in Cryptography, vol. 265 of London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.

[2] **M. Brown, D. Hankerson, J. Lopez and A. Menezes** , Software implementation of the NIST elliptic curves over prime fields. In Progress in Cryptology – CT-RSA 2001 (2001), D. Naccache, Ed., vol. 2020 of Lecture Notes in Computer Science, pp. 250–265.

[3] Certicom Research. Standards for efficient cryptography – SEC 1: Elliptic curve cryptography. Version 1.0, 2000. Available from http://www.secg.org/.

[4] Certicom Research. Standards for efficient cryptography – SEC 2: Recommended elliptic curve cryptography domain parameters. Version 1.0, 2000. Available from http://www.secg.org/.

[5] **H. Cohen, T. Ono and A. Miyaji**, Efficient elliptic curve exponentiation using mixed coordinates. In *Advances in Cryptology – ASIACRYPT '98* (1998), K. Ohta and D. Pei, Eds., vol. 1514 of Lecture Notes in Computer Science, pp. 51–65.

[6] **J.-S. Coron, Resistance against differential power analysis for elliptic curve cryptosystems. In** *Cryptographic Hardware and Embedded Systems – CHES '99* (1999), C. K. Ko¸c and C. Paar, Eds., vol. 1717 of Lecture Notes in Computer Science, pp. 292–302.

[7] Institute of Electrical and Electronics Engineers (IEEE). IEEE standard specifications for public-key cryptography. IEEE Std 1363-2000, 2000.

[8] **M. Joye, J.-J. Quisquater**, Hessian elliptic curves and side-channel attacks. In *Cryptographic Hardware and Embedded Systems – CHES 2001* [Pre-]Proceedings (2001), C. K. Ko¸c, D. Naccache, and C. Paar, Eds., pp. 412–420.

[9] **P. C. Kocher**, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology – CRYPTO '96* (1996), N. Koblitz, Ed., vol. 1109 of Lecture Notes in Computer Science, pp. 104–113.

[10] **P. C. Kocher, J. Jaffe, B. Jun**, Differential power analysis. In *Advances in Cryptology – CRYPTO '99* (1999), M. Wiener, Ed., vol. 1666 of Lecture Notes in Computer Science, pp. 388–397.

[11] **P.-Y. Liardet, N. P. Smart**, Preventing SPA/DPA in ECC systems using the Jacobi form. In *Cryptographic Hardware and Embedded Systems – CHES 2001* [Pre-]Proceedings (2001), C. K. Ko¸c, D. Naccache, and C. Paar, Eds., pp. 401–411.

[12] **A. Miyaji, T. Ono, H. Cohen**, Efficient elliptic curve exponentiation. In *International Conference on Information and Communications Security – ICICS '97* (1997), Y. Han, T. Okamoto, and S. Qing, Eds., vol. 1334 of Lecture Notes in Computer Science, pp. 282–290.

[13] **P. L. Montgomery, Speeding the Pollard and elliptic curve methods of factorization.** *Mathematics of Computation* **48** (1987), 243–264.

[14] National Institute of Standards and Technology (NIST). Digital Signature Standard (DSS). FIPS PUB 186-2, 2000.

[15] **K. Okeya, H. Kurumatani, K. Sakurai**, Elliptic curves with the Montgomery-form and their cryptographic applications. In *Public Key Cryptography – PKC 2000* (2000), H. Imai and Y. Zheng, Eds., vol. 1751 of Lecture Notes in Computer Science, pp. 238–257.

[16] **K. Okeya, K. Sakurai**, Power analysis breaks elliptic curve cryptosystems even secure against the timing attack. In *Progress in Cryptology – INDOCRYPT 2000* (2000), B. K. Roy and E. Okamoto, Eds., vol. 1977 of Lecture Notes in Computer Science, pp. 178–190.