A Randomized Efficient Algorithm for DPA Secure Implementation of Elliptic Curve Cryptosystems

S. Agagliate[1], P. Guillot[1], O. Orcière[2]

[1] Canal+ Technologies,
34 place Raoul Dautry, 75516 Paris Cedex 15, France
[2] Thales Communication,
66 rue du Fossé blanc, 92231 Gennevilliers Cedex, France

Abstract

To counteract Differential Power Analysis attack on an Elliptic Curve Cryptosystem, we propose to randomize an automaton realizing an efficient algorithm to compute the "exponentiation" $k.P$ of a point P with exponent k . The advantage of the presented method compared to [OA01] is to transform the exponent binary representation into a ternary one using Horner's rule. This allows to simplify the addition point formulas and to improve the computational time of 2% in comparison with the previous algorithm presented in [MO90].

1 Introduction

Many cryptosystems for signature, encryption or key exchange use "exponentiation" in finite groups. There are many different exponentiation algorithms : "Square and Multiply", "Window method" or "Ternary representation method". The implementation of such algorithm in a smartcard without security considerations would allow Side-Channel Attacks, i.e. someone may deduce a secret from an unintentional leak of information, such as computation timing or power consumption. This paper is focused on software countermeasures against a Differential Power Analysis attack (DPA) on Elliptic Curve Cryptosystems (ECC).

In the case of ECC, the finite group is the group of points of an elliptic curve with additive law. The most common operation used in ECC is the "exponentiation of a point P" : $k \mapsto k \cdot P$ where k is any integer and P a fixed point (base point). ECC arithmetic is characterized by :

- equivalent cost for addition and subtraction unlike arithmetic over F_p where multiplication is faster than division ;
- the use of projective coordinates may be more efficient if one operand is normalized ($z = 1$).

We propose to randomize an efficient exponentiation algorithm for elliptic curve to counteract DPA attack. The advantage of this method compared to [OA01] is to transform the exponent binary representation in a ternary one with the Most Significant Bit ordering (MSB ordering or Horner's rule). Consequently we are able to use simplified point addition formulas with z-coordinate equal to one, which require less elementary operations.

In section 2 we present an overview of different countermeasures found in literature. In section 3, we present an automaton realizing the exponentiation algorithm based on Horner's rule (paragraph 3.1) and the corresponding randomized automaton resilient to DPA attack (paragraph 3.2).

2 Related work

Two different types of methods are suggested in literature against DPA attack on ECC.

The first method consists of blinding variables in the computation of $k.P$: apply a random transformation to base point P or exponent k. The modified exponentiation is computed and the expected result recovered. In [C99] J-S. Coron proposes two ways to blind the base point : add a random point or randomize coordinates. In [JT01] M. Joye and C. Tymen propose to randomize elliptic curve using isomorphism or field isomorphism. For blinding key, different solutions are suggested : an additive mask in [C99], a multiplicative mask in [TB02], a key splitting in [CJ02], an overlapping window method in [IYTT02].

E. Oswald and M. Aigner give a completely different point of view to counteract a DPA attack in [OA01]. The authors propose to randomize the operations sequence itself during exponentiation. Let us assume that one cannot distinguish double operation from add or substract operation with only one power measurement. Because of the randomization, the intermediate values are different for a given exponentiation. This washes out the DPA bias signal. In this article we propose to randomize an automaton realizing an efficient exponentiation algorithm based on Horner's rule.

3 A randomized efficient algorithm as countermeasure against DPA attacks

3.1 An Addition-subtraction chain algorithm based on Horner's rule

To speed up computation in an ECC, one may reduce the number of operations (additions and doublings) during the exponentiation $k \cdot P$ The most natural idea is to use the binary decomposition of k. The ordinary binary algorithm is "Add and Double", beginning computation with the Least Significant Bit (LSB). Another method is to begin with MSB, then we obtain the "Double and Add" algorithm (Horner's rule).

Besides we can choose between several coordinate systems. Generally affine coordinates are not used because it requires an inversion in the base field, which is costly. To avoid the inversion, projective or Jacobian coordinates may be used. The advantage of "Double and Add" is that one of the addition operand is always the base point P. Consequently, we may use a z-coordinate equals to one for the point P. Addition formulas are simplified and the computation of $P + Q$ and consequently of $k \cdot P$ is speeded up.

Furthermore, on the additive group law of an elliptic curve, subtraction has the same computational load as addition. This fact allows us to use a ternary representation of the integer k as follows.

$$k = \sum_{i=0}^{n-1} k_i.2^i, \quad \text{where} \quad k_i \in \{-1, 0, 1\}$$

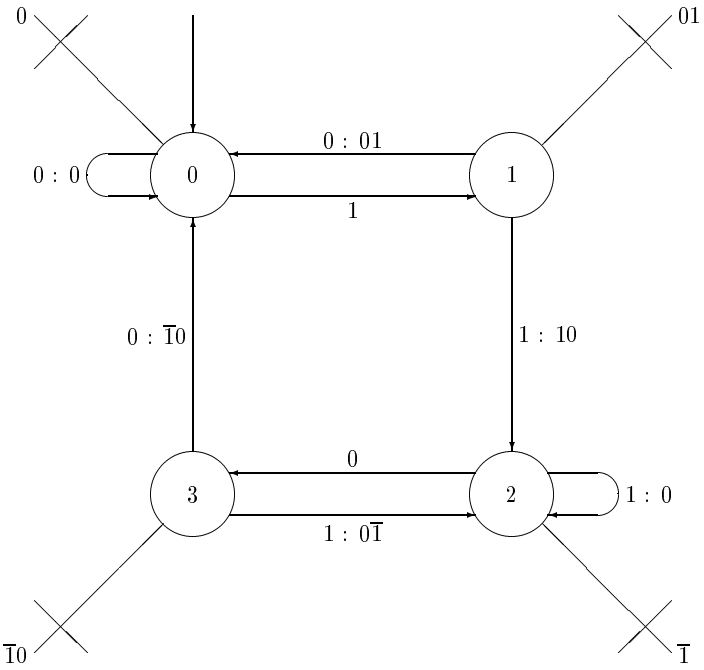

Figure 1: Deterministic MSB automaton

This representation is not unique. A Non-Adjacent Form (NAF) of k is a ternary representation where "1" or "-1" are never adjacent. The NAF representation is unique for a given k and its weight is the lowest among all the possible ternary representation weight. The average number of additions/subtractions for a bit length n is $n/3$ (see [EK90],[Z93]). In [R60] an algorithm is proposed to compute the NAF of k using LSB ordering. In [MO90] an automaton is described to compute $k \cdot P$ starting with the LSB of k, with a minimal weight ternary representation not equal to the NAF.

The automaton proposed here computes $k \cdot P$ through a other minimal weight ternary representation. It reads k from MSB taking advantage of the simplified point addition formulas with z-coordinate equals to one.

The principle of this automaton is to use the two following transformations :

$$1^a \mapsto 10^{a-1}\overline{1}$$

$$1^a 01^b \mapsto 10^a \overline{1} 0^{b-1} \overline{1},$$

with $\overline{1} = -1$, $a \geq 2$ and $b \geq 1$ using $\overline{1}1 = 0\overline{1}$ recursively.

The states of the new automaton are given by the following table :

State	Figure (two last input bits)	Name
0	(00)	no block
1	(01)	no block to block
2	(11)	block
3	(10)	block to no block

Block denotes a sequence of consecutive "1" from the input. The starting point of the automaton is state 0. Each arrow corresponds to a transition from a state to an other, with a conditional transition tagged "a:b", "a" being the input bit and "b" being the output sequence if any. The output ternary representation does not require not to be stored, which may be useful to implement in a low memory device such as a smartcard. We compute the point $Q = k \cdot P$ on the fly applying to output bits the following Horner type rules:

$$\begin{array}{rcl} \overline{1} & \mapsto & Q = 2Q, \quad Q = Q - P \\ 0 & \mapsto & Q = 2Q \\ 1 & \mapsto & Q = 2Q, \quad Q = Q + P \end{array}$$

Let us denote $A \xrightarrow{c} B$ the path from state A to state (or exit) B with the transitional condition c. Having a closer look at the automaton, we notice that there are only four cases where the output sequence has adjacent 1 or $\overline{1}$.

covered path	output bits
$1 \xrightarrow{0:01} 0 \xrightarrow{1} 1 \xrightarrow{1:10} 2$	0110
$3 \xrightarrow{1:0\overline{1}} 2 \xrightarrow{0} 3 \xrightarrow{0:\overline{1}0} 0$	$0\overline{1}\overline{1}0$
$3 \xrightarrow{1:0\overline{1}} 2 \xrightarrow{0} 3 \xrightarrow{\overline{1}0} exit$	$0\overline{1}\overline{1}0$
$3 \xrightarrow{1:0\overline{1}} 2 \xrightarrow{\overline{1}} exit$	$0\overline{1}\overline{1}$

In each case, two adjacent "1" or "-1" are preceeded by at least two zeros. Thus it may be easily transformed into a NAF without changing the weight:

$$00110 \mapsto 010\overline{1}0$$
$$00\overline{1}\overline{1}0 \mapsto 0\overline{1}010$$
$$00\overline{1}\overline{1} \mapsto 0\overline{1}01$$

This proves that the resulting automaton sequence has a minimal weight.

The following table shows the various methods in terms of the average number of operations (multiplications or squares) required to compute the point $Q = k \cdot P$. The integer k is coded with n bits. The modified Jacobian coordinates are used in the addtion formulas (see [CMO98]).

Operations	Add and Double	Double and Add	Morain-Olivos	New Method
nb of op. for $2 \cdot P$	8	8	8	8
nb of op. for $P + Q$	19	14	19	14
nb of op. for $k \cdot P$	$17.5n$	$15n$	$14.3n$	$12.7n$

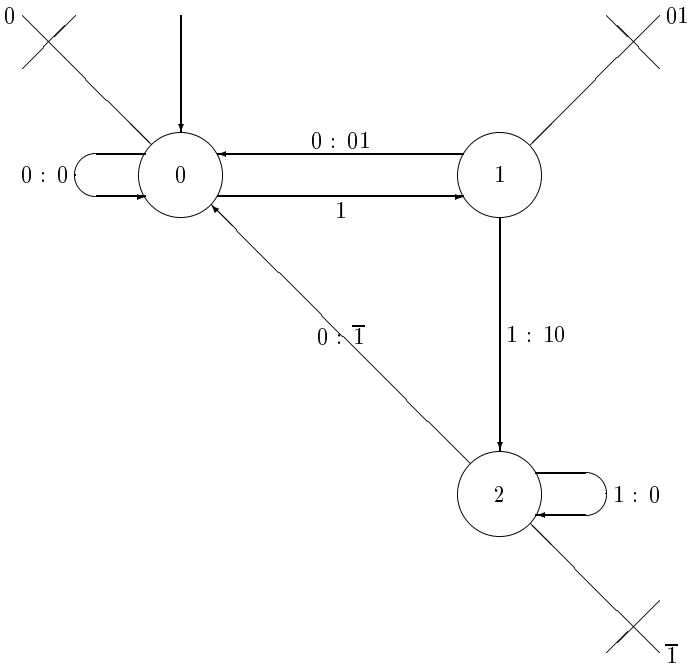

Figure 2: Suboptimal automaton

3.2 Randomization of the automaton

As mentioned before, the automaton shown on figure 1 is efficient to transform the binary exponent representation in a ternary one. To secure it from a DPA attack, we randomize the algorithm itself. The idea is to merge different automata by introducing a random bit in particular automaton states to choose one of them. Thus we create a new automaton with randomized transitions. For fixed operands, the exponentiation will be computed in different ways. So the addition-subtraction sequence is unpredictable.

First, we define an automaton which is to be merged with the automaton of figure 1. Let us remember that the latter uses the two following transformations :

$$1^a \mapsto 10^{a-1}\overline{1}$$

$$1^a 01^b \mapsto 10^a\overline{1}0^{b-1}\overline{1}$$

for $a \geq 2$ and $b \geq 1$ using $\overline{1}1 = 0\overline{1}$ recursively.

We consider now the automaton of figure 2 which only uses the first transformation with MSB. Thus the exit sequence $\overline{1}1$ is allowed. The previous three states automaton carries out this transformation. We remain on state 0 (no block) as long as there is a sequence of zeros. When a one appears we go to state 1 (no block to block). If there is an other one then we have a block : we go from state 1 to state 2 (block) and write "10" followed by "0" until the number one appears. At the end of the block we go back to state 0.

Let us merge the automata of figures 1 and 2 by introducing a random bit e from state 2 when the incoming exponent bit is 0. If $e = 0$ then we cover automaton 1 thus just go to state 3. If $e = 1$ then we cover automaton 2 i.e. go to state 0 with outcoming "$\overline{1}$".

This randomization changes the transition of the automaton. To increase the security of this automaton against DPA attack, we use two additional randomizations. Now it deals with synonyms in the ternary expansion.

Note that $01 = 1\overline{1}$. We put a random bit e in state 1 when there is a zero incoming. If $e = 0$ then we do not change anything, else we go to state 0 with outcoming $1\overline{1}$ instead of 01.

Similarly, we introduce a random bit e in state 3 when there is a one incoming, and randomly change the outcoming $0\overline{1}$ by $\overline{1}1$ on the transition to state 2. Therefore the final randomized automaton is given in figure 3.

The pseudo code corresponding to automaton 3 is the following:

```
Final_Randomized_Automaton(P, k=(k_m,..., k_1, k_0) with the MSB k_m==1) {
       state=0;  Q=O;
       for (i=m;i>=0;i--) {
              switch(state) {
                     case 0: if (k_i==1) state=1;
                             else Q:=2Q;
                     case 1: if (k_i==1) {state=2; Q:=2Q; Q:=Q+P; Q:=2Q;}
                             else {
                                     e=rand(); state=0;
                                     if (e==0) {Q:=2Q; Q:=2Q; Q:=Q+P;}
                                     else {Q:=2Q; Q:=Q+P; Q:=2Q; Q:=Q-P;}
                             }
                     case 2: if (k_i==1) Q:=2Q;
                             else {
                                     e=rand();
                                     if (e==0) state=3;
                                     else {state=0; Q:=2Q; Q:=Q-P;}
                             }
                     case 3: if (k_i==1) {
                                     e=rand(); state 2;
                                     if (e==0) {Q:=2Q; Q:=2Q; Q:=Q-P;}
                                     else {Q:=2Q; Q:=Q-P; Q:=2Q; Q:=Q+P;}
                             }
                             else {state=0; Q:=2Q; Q:=Q-P; Q:=2Q; }
              }
       }
       switch(state) {
                case 0: Q:=2Q;
                case 1: Q:=2Q; Q:=2Q; Q:=Q+P;
                case 2: Q:=2Q; Q:=Q-P;
                case 3: Q:=2Q; Q:=Q-P; Q:=2Q;
       }
       return Q;
}
```

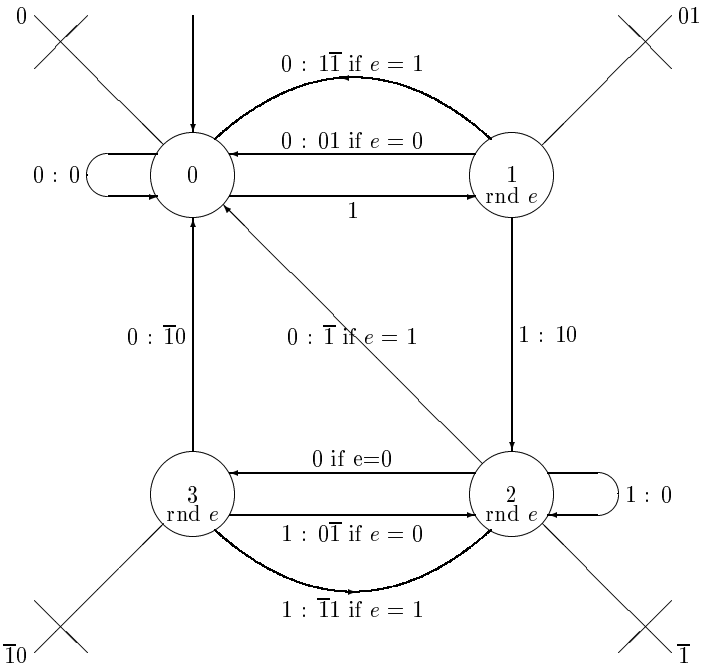

Figure 3: Final randimized automaton

To compute the mean weight of the ternary representation, we use the following method :

A transition between two states is represented by a polynomial $p_i X^i$ with i the outcoming number of "1" or "-1" and p_i the transition probability. We assume a uniform distribution for the current exponent bit and the random bit e. For instance, the polynomial used for the transition $1 \longrightarrow 2$ is given by $1/2X$. The one for transition $1 \longrightarrow 0$ is $1/4X + 1/4X^2$. Let A be the transition matrix with coefficients $a_{i,j}$ the polynomials corresponding to the transition $i \longrightarrow j$. Let E be the input vector and S the output vector. If the exponent has n bits, then $P_n =^t EA^nS$ represents the weight probability enumerator polynomial. The weight mean value is given by $P'_n(1)$.

The following table shows the results with uniformly distributed randomized transitions :

size of incoming exponent (n)	160	200	250	300
mean weigth of outcoming exponent(w)	68.8776	86.0204	107.449	128.878
ratio n/w	2.32296	2.32503	2.32669	2.32778
additional operations from automaton 1	10.7%	10.7%	10.7%	10.6%
additional operations from [MO90]	-2.1%	-2.2%	-2.2%	-2.2%

We assume that the cost of an addition (A) equals the cost of a substraction (S). The doubling cost (D) is slightly less than A. In state 1 and 3 the cost of automaton 3 is $2D + A$ if $k_i = 0, 1$. The cost of new automaton is $2D + A$ except when $k_i = 0, e = 1$ in which case it

is $2D + 2A$. This shows that this automaton is not suitable to prevent SPA attack if we are able to distinguish doubling from adding or substracting.

4 Conclusion

To compute the exponentiation $k.P$ of point P with exponent k, we usually use additions and doublings. We want to lower the number of additions. One way to do that is to introduce a new operation : substraction. So we have to define the sequence of additions, subtractions and doublings needed to compute the exponentiation, the so-called "addition-substraction chain".

This is equivalent to compute a ternary representation of exponent k which represents k with $-1, 0, 1$ digits in base 2. This representation is computed from the binary representation of integer k which represents integer k with $0, 1$ digits in base 2. Contrary to the binary expansion, the ternary expansion is not unique except in the non adjacent form case (NAF) : two "1" or "-1" are not allowed to be adjacent in this form. In addition, the MSB ordering (or Horner's rule) is chosen in order to use addition with a fixed point : the most significant binary bits of integer k are used first.

Using the binary bits of integer k using MSB ordering, we are able to compute on the fly its corresponding ternary bits : starting with a temporary point $Q = 0$, computing a "0" corresponds to doubling the current point Q, computing a "1" corresponds to adding the base point P to Q and computing a "-1" corresponds to substracting the base point P to Q. This is the case only because Horner's rule is chosen. At the end the point $k \cdot P$ is computed in the variable Q. This process defines the "addition-substraction chain" on the fly.

We define an automaton which compute the ternary bits on the fly with

- four states $(0, 1, 2, 3)$
- eight transitions $(0 \to 0, 0 \leftrightarrow 1, 1 \to 1, 1 \to 2, 2 \leftrightarrow 3, 3 \to 0)$
- seven writing rules (writing nothing,"0","-1","01","10","0-1","-10")

This automaton computes a minimal weight ternary expansion different from NAF with MSB ordering on the fly. The automaton can be viewed as a set of transitions along with writing rules : the current state and current bit determine the transition to be used. This transition gives the new state along with the writing rules to output a ternary expansion of integer k with Horner's rule on the fly.

To counteract DPA attacks for an ECC, we propose a variant of this automaton with one randomized transition and two randomized ternary expansion writing rules. The ternary expansion computed is no longer of minimal weight. On the contrary, it randomly varies each time it is computed for the same integer k. The new automaton chooses at random a transition sequence among the two equivalent transition sequence : "$2 \to 0$" or "$2 \to 3 \to 0$" and two randomized writing rules : replacing at random "01" by "1-1" and "0-1" by "-11" in the ternary expansion of integer k.

The randomized algorithm increases the number of operations of approximately 10.7% compared to the original one. Furthermore, since we read the exponent from the MSB, we can use simplified point addition formulas with z-coordinate equals to one. With modified Jacobian coordinates we have about 11.6% less operations than if we had used LSB to compute an exponentiation. Therefore our randomized algorithm uses 2.2% less operations than

the non-randomized automaton beginning with the LSB described by Morain and Olivos in [MO90].

References

[CJ02] C. Clavier, M. Joye: "Universal Exponentiation Algorithm - A First Step Towards Provable SPA-Resistance"; Cryptographic Hardware and Embedded Systems – CHES 2001, LNCS 2162, pp. 300-308, 2001

[CMO98] H. Cohen, A. Miyaji, T. Ono: "Efficient elliptic curve exponentiation using mixed coordinates" ; Asiacrypt 98, LNCS 1514, pp. 51-65, 1998

[C99] J-S Coron: "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems" ; Cryptographic Hardware and Embedded Systems - CHES'99, LNCS 1717, pp. 292-302, 1999

[EK90] O. Egecioglu, C.K. Koc: "Fast modular exponentiation" ; In E.Arikan, editor, Communication, Control and Signal Processing: Proceedings of 1990 Bilkent International Conference on New Trends in Communication, Control, and Signal Processing, pp.188-194, Bilkent Univ. Ankara, Turkey, 1990

[IYTT02] K Ito, J. Yajima, M. Takenaka, N. Torii: "DPA countermeasures by Improving the Window Method" ; Cryptographic Hardware and Embedded Systems - CHES 2002, LNCS, 2002

[JT01] M. Joye, C. Tymen: "Protection against Differential Analysis for Elliptic Curve Cryptography-An Algebraic Approach" ; Cryptographic Hardware and Embedded Systems - CHES 2001, LNCS 2162, pp. 377-390, 2001

[MO90] F. Morain, J. Olivos: "Speeding up the computations on an elliptic curve using addition-substraction chains" ; published in RAIRO Inform. Théor. Appl., 1990, 24, 6, p. 531-543.

[OA01] E. Oswald, M. Aigner: "Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks" ; Cryptographic Hardware and Embedded Systems - CHES 2001, LNCS 2162, pp. 39-50, 2001

[R60] G.W. Reitwieesner: "Binary Arithmetic" ; Advances in Computer, 1:231-308,1960

[TB02] E.Trichina, A.Bellezza: "Implementation of ECC with Built-in Counter Measures against Side Channel Attacks" ; Cryptographic Hardware and Embedded Systems – CHES 2002, LNCS, 2002

[Z93] C.N. Zhang: "An improved binary algorithm for RSA" ; Computer Math. Applic., vol. 25, no. 6, pp. 15-24, 1993