Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems

[Published in Ç.K. Koç and C. Paar, Eds., *Cryptographic Hardware and Embedded Systems*, vol. 1717 of *Lecture Notes in Computer Science*, pp. 292–302, Springer-Verlag, 1999.]

Jean-Sébastien Coron[1,2]

[1] École Normale Supérieure
45 rue d'Ulm, 75005 Paris, France
coron@clipper.ens.fr
[2] Gemplus Card International
34 rue Guynemer, 92447 Issy-les-Moulineaux, France
jean-sebastien.coron@gemplus.com

Abstract. Differential Power Analysis, first introduced by Kocher *et al.* in [14], is a powerful technique allowing to recover secret smart card information by monitoring power signals. In [14] a specific DPA attack against smart-cards running the DES algorithm was described. As few as 1000 encryptions were sufficient to recover the secret key. In this paper we generalize DPA attack to elliptic curve (EC) cryptosystems and describe a DPA on EC Diffie-Hellman key exchange and EC El-Gamal type encryption. Those attacks enable to recover the private key stored inside the smart-card. Moreover, we suggest countermeasures that thwart our attack.

Keywords. Elliptic curve, power consumption, differential power analysis.

1 Introduction

The use of elliptic curve in cryptography was first proposed by Miller [17] and Koblitz [12] in 1985. Since that time, a lot of attention has been paid to elliptic curves for cryptographic applications and it has become increasingly common to implement public-key protocols on elliptic curves over large finite field. Elliptic curves (EC) provide a group structure, which can be used to translate existing discrete-logarithm cryptosystems into the context of EC. The discrete logarithm problem in a cyclic group G of order n with generator g refers to the problem of finding x given some element $y = g^x$ of G. The discrete logarithm problem over an EC seems to be much harder than in other groups such as the multiplicative group of a finite field. No subexponential-time algorithm is known for the discrete logarithm problem in the class of *non-supersingular* EC. Consequently, keys can be much smaller in the EC context, typically about 160 bits.

In this paper we consider attacks based on the monitoring of power consumption of smart-card EC implementation. Differential Power Analysis, first described by Kocher *et al.* in [14], is a powerful technique that exploit the leakage of information related to power consumption. The attack was successfully applied to a DES implementation; as few as 1000 encryptions were sufficient to recover the secret key [14]. More recently, the resistance of smart-card implementations of the AES candidates against monitoring power consumption was considered in [1, 3, 5]. The conclusion was that straightforward implementations of AES candidates were highly vulnerable to power analysis. In this paper we show that naive implementations of ECC are also highly vulnerable to power analysis.

The paper is organized as follows. After recalling the principle of EC operations in section 2, we describe in section 3 the principle of our power consumption attack. In section 4, we apply the attack to some common discrete-logarithm based cryptosystems such as Diffie-Hellman key exchange [7] and ElGamal public-key encryption [8]. Finally we suggest three countermeasures that prevent our attack.

2 Elliptic Curve Group Operation

2.1 Definition of an elliptic curve

An elliptic curve is the set of points (x, y) which are solutions of a bivariate cubic equation over a field K (see [16]). An equation of the form:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{1}$$

where $a_i \in K$, defines an elliptic curve over K.

If char $K \neq 2$ and char $K \neq 3$, equation (1) can be transformed to:

$$y^2 = x^3 + ax + b$$

with $a, b \in K$.

In the field $\mathrm{GF}(2^n)$ of characteristic 2, equation (1) can be reduced to the form:

$$y^2 + xy = x^3 + ax^2 + b$$

with $a, b \in K$.

The set of points on an elliptic curve, together with a special point $\mathcal{O}$ called the *point at infinity* can be equipped with an Abelian group structure by the following addition operation:

Addition formula [16] for char $K \neq 2, 3$:

Let $P = (x_1, y_1) \neq \mathcal{O}$ be a point, the inverse of P is $-P = (x_1, -y_1)$. Let $Q = (x_2, y_2) \neq \mathcal{O}$ be a second point with $Q \neq -P$, the sum $P + Q = (x_3, y_3)$

can be calculated as:

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

with

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, \text{ if } P \neq Q, \\ \dfrac{3x_1^2 + a}{2y_1}, \text{ if } P = Q. \end{cases}$$

To subtract the point $P = (x, y)$, one adds the point $-P$.

Addition formula for char $K = 2$:

Let $P = (x_1, y_1) \neq \mathcal{O}$ be a point, the inverse of P is $-P = (x_1, x_1 + y_1)$. Let $Q = (x_2, y_2) \neq \mathcal{O}$ be a second point with $Q \neq -P$, the sum $P + Q = (x_3, y_3)$ can be calculated as:

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$$
$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1$$
$$\lambda = \frac{y_1 + y_2}{x_1 + x_2}$$

if $P \neq Q$ and:

$$x_3 = \lambda^2 + \lambda + a$$
$$y_3 = x_1^2 + (\lambda + 1)x_3$$
$$\lambda = x_1 + \frac{y_1}{x_1}$$

if $P = Q$.

2.2 Computing a multiple of a point

The operation of adding a point P to itself d times is called *scalar multiplication* by d and denoted dP. Scalar multiplication is the basic operation for EC protocols. Scalar multiplication in the group of points of an elliptic curve is the analogous of exponentiation in the multiplicative group of integers modulo a fixed integer m.

Computing dP can be done with the straightforward *double-and-add* approach based on the binary expansion of $d = (d_{\ell-1}, \ldots, d_0)$ where $d_{\ell-1}$ is the most significant bit of d (the method is the analogous of the *square-and-multiply* algorithm for exponentiation):

```
Algorithm 1 (Double-and-add)
 input P
 Q ← P
 for i from ℓ − 2 to 0 do
     Q ← 2Q
     if d_i = 1 then Q ← Q + P
 output Q
```

Various techniques exist to speed-up scalar multiplication by reducing the number of elementary point operations: see [9] for a good survey. If the point P is known in advance, it may be advantageous to precompute a table of multiples of P [2]. Because elliptic curve subtraction has the same cost as addition, the previous *double-and-add* algorithm can be improved with the *addition-subtraction* algorithm which uses a signed binary expansion of d:

$$d = \sum_{i=0}^{\ell-1} c_i 2^i$$

with $c_i \in \{-1, 0, 1\}$.

The *non-adjacent form* (NAF) of d is a signed binary expansion of d with $c_i c_{i+1} = 0$ for all $i \geq 0$. Each positive integer has a unique NAF. Moreover, the NAF of d has the fewest nonzero coefficients of any signed binary expansion of d [9]. [18] describes an algorithm that generates the NAF of any positive integer.

```
Algorithm 2 (Addition-subtraction method)
  input P
  Q ← P
  for i from ℓ − 2 to 0 do
      Q ← 2Q
      if c_i = 1 then Q ← Q + P
      if c_i = −1 then Q ← Q − P
  output Q
```

The double-and-add method and addition-subtraction method can be generalized to the *m-ary method*, the *window method* and the *signed binary window method* [9, 15].

The problem of finding a method to compute dP with the fewest number of elliptic curve group operations for a given d is equivalent to finding the shortest addition-subtraction chain for d [9]. An *addition chain* [11] for d is a sequence of positive integers:

$$a_0 = 1 \rightarrow a_1 \rightarrow a_2 \rightarrow \ldots \rightarrow a_r = d$$

such that $a_i = a_j + a_k$, for some $k \leq j < i$, for all $i = 1, 2, \ldots, r$.

An addition chain can be extended to an *addition-subtraction chain* [11] with $a_i = \pm a_j \pm a_k$ in place of $a_i = a_j + a_k$. The shortest addition-subtraction chain

for d gives the fewest number of elliptic group operations for computing dP by computing $a_1P, a_2P, \ldots a_rP = dP$.

3 Recovering d in $Q = dP$ From the Power Consumption

In 1998, Kocher described in a technical draft [14] Simple Power Attacks (SPA) and Differential Power Analysis (DPA) on DES. A SPA consists in observing the power consumption of one single execution of a cryptographic algorithm. A DPA is more sophisticated and powerful. It consists in performing a statistical analysis of many executions of the same algorithm with different inputs.

Here we show that monitoring power consumption during the computation of $Q = dP$ knowing P may enable to recover d. First we show that a naive implementation of scalar multiplication may be vulnerable to SPA. However, it is not difficult to make the implementation resistant against SPA. We then describe a DPA attack of an implementation of scalar multiplication.

3.1 Resistance against SPA

Power consumption attacks are based on the observation that the power consumed at a given time during cryptographic process is related to the instruction being executed and the data being manipulated. Power consumption enables to visually identify large features, for example the main loop in algorithm 1. Power consumption analysis may also enable to distinguish between instruction being executed. For example, it might be possible to distinguish between point doubling and point addition in algorithm 1, thereby revealing the bits of the exponent d.

In order to be resistant against SPA, the instructions performed during a cryptographic algorithm should not depend on the data being processed, *e.g.* there should not be any branch instructions conditioned by the data. It is easy to modify algorithm 1 to achieve this goal:

```
Algorithm 1' (Double-and-add resistant against SPA)
  input P
  Q[0] ← P
  for i from ℓ − 2 to 0 do
      Q[0] ← 2Q[0]
      Q[1] ← Q[0] + P
      Q[0] ← Q[d_i]
  output Q[0]
```

3.2 DPA against double-and-add algorithm

In this section we describe a DPA against an implementation of algorithm 1'. We assume that the algorithm is performed in constant time. Otherwise the

implementation may be subject to timing attack [13] and Simple Power Attacks [14].

DPA on DES [6] algorithm as described in [14] uses correlation between power consumption and specific key-dependent bits which appear at known steps of the encryption computation. For example, a selected bit b at the output of one SBOX of the first round will depend on the known input message and 6 unknown bits of the key. In [14], the correlation between power consumption and b is computed for the 64 possible values of the 6 unknown bits of the key. The correlation is likely to be maximal for the correct guess of the 6 bits of the key. The attack can be repeated for the remaining SBOXes, thus revealing 48 bits of the key. The remaining 8 bits of the key can be recovered by exhaustive search.

A Differential Power Analysis on algorithm 1' in section 3.1 can be performed by noticing that at step j the processed point Q depends only on the first bits $(d_{\ell-1}, \ldots, d_j)$ of d. Now assume that we know how points are represented in memory during computation and select a particular bit (the same for all points) of this representation. When point Q is processed, power consumption will be correlated to this specific bit of Q. No correlation will be observed with a point not computed inside the card. Thus it is possible to successively recover the bits of the exponent by guessing which points are computed by the card.

The second most significant bit $d_{\ell-2}$ of d can be recovered by computing the correlation between power consumption and any specific bit of the binary representation of $4P$. If $d_{\ell-2} = 0$, $4P$ is computed during algorithm 1', and power consumption is thus correlated with any specific bit of $4P$. Otherwise if $d_{\ell-2} = 1$, $4P$ is never computed, and no correlation will be observed with $4P$. This gives $d_{\ell-2}$. The following bits of d can be recursively recovered in the same way.

Assume that algorithm 1' is performed k times with distinct $P_1, P_2, \ldots, P_k$ to compute $Q_1 = dP_1, Q_2 = dP_2, \ldots, Q_k = dP_k$. Let $C_i(t)$ be the power consumption associated with the i-th execution of the algorithm for $1 \leq i \leq k$. Let s_i be any specific bit of the binary representation of $4P_i$ for $1 \leq i \leq k$. The correlation function $g(t)$ between s_i and $C_i(t)$ can be computed as follows:

$$g(t) = < C_i(t) >_{i=1,2\ldots,k|s_i=1} - < C_i(t) >_{i=1,2,\ldots,k|s_i=0} \tag{2}$$

Assume that the points $4P_i$ are processed at time $t = t_1$, power consumption $C_i(t_1)$ will then be correlated with the specific bit s_i of the binary representation of $4P_i$. The average of power consumption for those points $4P_i$ for which $s_i = 1$ will be different from the power consumption for the points $4P_i$ for which $s_i = 0$, and function $g(t)$ will present a "peak" at time $t = t_1$. If the points $4P_i$ are never computed, no "peak" will be observed in function $g(t)$. This is illustrated in figure 1 and 2.[1]

[1] Real power consumption curves were *voluntarily* excluded from this paper to avoid straightforward product identification.

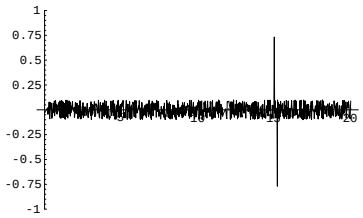

Fig. 1. Simulated correlation function $g(t)$ between the points $4P_i$ and power consumption $C_i(t)$ when $d_{\ell-2} = 0$. A peak is observed corresponding to the computation of $4P_i$ inside the card.

Fig. 2. Simulated correlation function $g(t)$ between the points $4P_i$ and power consumption $C_i(t)$ when $d_{\ell-2} = 1$. No peak is observed since the points $4P_i$ are never computed by the card.

3.3 Extending the attack to any scalar multiplication algorithm

In this section we show how to extend the previous attack to any scalar multiplication algorithm executed in constant time with a constant addition-subtraction chain, *i.e.* for any point P the algorithm computes the sequence of point:

$$a_0P = P \rightarrow a_1P \rightarrow a_2P \rightarrow \ldots \rightarrow a_rP = dP$$

such that $a_i = \pm a_j \pm a_k$, for some $k \leq j < i$, for all $i = 1, 2, \ldots, r$.

The attack consists in successively guessing the a_i starting from $a_0 = 1$ to $a_r = d$. At step $i \geq 1$, one constructs the set A_i of all possible $a'_i = \pm a_j \pm a_k$ for all $0 \leq k \leq j < i$, and for each $a'_i \in A_i$ computes the correlation function $g(t)$ between the point a'_iP and power consumption. If a peak can be observed in $g(t)$, this will indicate that the point a'_iP has been computed by the device and thus $a_i = a'_i$. This enables to recover $d = a_r$ in $\mathcal{O}(r^2)$ time.

4 Attacks on Elliptic Curve Public Key Protocols

In this section we apply the attack to elliptic curve public key protocols such as El-Gamal encryption and Diffie-Helman key exchange. The attack can not apply to the ECDSA signatures, since in this case scalar multiplication is performed with a random exponent instead of a fixed exponent.

4.1 Elliptic Curve Encryption Scheme

This scheme is analogous to El-Gamal encryption [8].

System parameters:
An elliptic curve $\mathcal{E}$ over $GF(p)$ or $GF(2^n)$.
The order of $\mathcal{E}$ denoted $\#\mathcal{E}$ must be divisible by a large prime q.
$G \in \mathcal{E}$ of order q.

Key generation:
Secret key: $d \in_R [1, q-1]$.
Public key: $Q = dP$.

Encryption of a message m:
Pick $k \in_R [1, q-1]$.
Compute the points $kP = (x_1, y_1)$ and $kQ = (x_2, y_2)$, and $c = x_2 + m$.
The ciphertext is (x_1, y_1, c).

Decryption:
Compute $(x'_2, y'_2) = d(x_1, y_1)$ and $m = c - x'_2$.

The attack described before enables to recover d when the device decrypts the ciphertext (x_1, y_1, c) for various points (x_1, y_1).

4.2 Elliptic Curve Diffie-Hellman key exchange

The EC Diffie-Hellman protocol derives a common secret value z from one party's private key and another party's public key. The protocol is referenced as ECSVDP-DH (Elliptic Curve Secret Value Derivation Primitive, Diffie-Hellman version) in [10]. If the two parties correctly execute this primitive, they will produce the same output.

System parameter:
An elliptic curve $\mathcal{E}$ over $GF(p)$ or $GF(2^n)$.
The order of $\mathcal{E}$ denoted $\#\mathcal{E}$ must be divisible by a large prime q.
Alice's own private key s.
Bob's public key W.

Derivation of the shared secret value z:
Compute the point $P = sW$.
If $P = \mathcal{O}$ output "error" and stop.
The shared secret value is $z = x_p$, the x-coordinate of P.

The attack described in the previous section recovers Alice's secret key when she computes the point $P = sW$ for Bob's public key W.

5 Countermeasures Against DPA

In this section we describe three countermeasures that prevent from the attack described in section 3. Recall that the attack enables to recover d when $Q_i = dP_i$ are computed inside the card for various P_i for $1 \leq i \leq k$. These three countermeasures are based on introducing random numbers during the computation of $Q = dP$. We underline that other attacks might of course not be thwarted by our countermeasures.

5.1 First countermeasure: randomization of the private exponent

Let $\#\mathcal{E}$ be the number of points of the curve. The computation of $Q = dP$ is done by the following algorithm:

1. Select a random number k of size n bits. In practice, one can take $n = 20$ bits.
2. Compute $d' = d + k.\#\mathcal{E}$.
3. Compute the point $Q = d'P$. We have $Q = dP$ since $\#\mathcal{E}P = \mathcal{O}$.

This countermeasure makes the previous attack infeasible since the exponent d' in $Q = d'P$ changes at each new execution of the algorithm.

5.2 Second countermeasure: blinding the point P

The method is analogous to Chaum's blind signature scheme for RSA [4]. The point P to be multiplied is "blinded" by adding a secret random point R for which we know $S = dR$. Scalar multiplication is done by computing the point $d(R + P)$ and subtracting $S = dR$ to get $Q = dP$. The points R and $S = dR$ can be initially stored inside the card and refreshed at each new execution by computing $R \leftarrow (-1)^b 2R$ and $S \leftarrow (-1)^b 2S$, where b is a random bit generated at each new execution. This makes the previous attack infeasible since the point $P' = P + R$ to be multiplied by d is not known to the attacker.

5.3 Third countermeasure: randomized projective coordinates

Projective coordinates [16] can be used to avoid the costly field inversion for point addition and doubling. The projective coordinates (X, Y, Z) of a point $P = (x, y)$ are given by:

$$x = \frac{X}{Z} \qquad y = \frac{Y}{Z}$$

Another system of projective coordinates may be found in [10]. The projective coordinates of a point are not unique because:

$$(X, Y, Z) = (\lambda X, \lambda Y, \lambda Z) \tag{3}$$

for every $\lambda \neq 0$ in the finite field.

The third countermeasure consists in randomizing the projective coordinate representation of a point $P = (X, Y, Z)$. Before each new execution of the scalar multiplication algorithm for computing $Q = dP$, the projective coordinates of P are randomized according to equation (3) with a random λ. The randomization can also occur after each point addition and doubling.

This makes the attack described above infeasible since it is not possible for the attacker to predict any specific bit of the binary representation of P in projective coordinates.

6 Conclusion

We have shown that unless protected, implementations of elliptic curve cryptosystems such as El-Gamal type encryption or Diffie-Hellman key exchange are vulnerable to Differential Power Analysis. We have introduced three countermeasures that address specifically these attacks. Those countermeasures are easy to implement and do not impact efficiency in a significant way. However, we do not pretend that those countermeasures thwart from all kinds of power attacks, since it may be possible to exploit the information leakage through power consumption in a different way.

Acknowledgments I thank David Naccache and Jean-Marc Robert for their careful reading and valuable suggestions, and the anonymous referees for their helpful comments.

References

1. E. Biham, A. Shamir. Power analysis of the key scheduling of the AES candidates, *Proceedings of the second AES Candidate Conference*, March 1999, pp. 115-121.
2. E. Brickell, D. Gordon, K. McCurley, D. Wilson. Fast Exponentiation with Precomputation (Extended Abstract), *Advances in Cryptology - Eurocrypt '92*, LNCS 658 (1993), Springer-Verlag, pp. 200-207.
3. S. Chari, C. Jutla, J.R. Rao, P. Rohatgi. A cautionary note regarding evaluation of AES candidates on smart-cards, *Proceedings of the second AES Candidate Conference*, March 1999, pp. 133-147.
4. D. Chaum. Security without identification: transaction systems to make Big Brother obsolete, *Communications of the ACM*, vol. 28, n. 10, Oct 1985, pp. 1030-1044.
5. J. Daemen, V. Rijmen. Resistance against implementation attacks A comparative study of the AES proposals, *Proceedings of the second AES Candidate Conference*, March 1999, pp. 122-132.
6. FIPS 46, Data encryption standard, Federal Information Processing Standards Publication 46, U.S. Department of Commerce/National Bureau of Standards, National Technical Information Service, Springfield, Virginia, 1977.
7. W. Diffie and M. Hellman. New directions in cryptography, *IEEE Trans. Info. Theory*, IT-22, 1976, pp 644-654.
8. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Info. Theory*, IT-31, 1985, pp 469-472.
9. D.M. Gordon. A Survey of Fast Exponentiation Methods, *Journal of Algorithms* 27, 129-146 (1998).
10. IEEE P1363/D7. Standard Specifications for Public Key Cryptography. September 11, 1998.
11. D.E. Knuth, *Seminumerical Algorithms*, The Art of Computer Programming, 2 Addison Wesley, 1969.
12. N. Koblitz. Elliptic Curve Cryptosystems, *Mathematics of Computation*, vol. 48, 1987, pp. 203-209.
13. Paul Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems, *Advances in Cryptology, Proceedings of Crypto' 96, LNCS 1109*, N. Koblitz, Ed., Springer-Verlag, 1996, pp. 104-113.
14. Paul Kocher, Joshua Jaffe, and Benjamin Jun, Introduction to Differential Power Analysis and Related Attacks, `http://www.cryptography.com/dpa/technical`, 1998.
15. K. Koyama, Y. Tsuruoka, Speeding up elliptic cryptosystems by using a signed binary window method, *Advances in Cryptology - Proceedings of Crypto '92*, LNCS 740, pp. 345-357, Springer-Verlag, Berlin/New-York, 1993.
16. A. J. Menezes, "Elliptic Curve Public Key Cryptosystems", Kluwer Academic Publishers, 1993.
17. V.S. Miller. Use of Elliptic Curves in Cryptography, *Proceedings of Crypto 85*, LNCS 218, Springer, 1986, pp. 417-426.
18. F. Morain, J. Olivos. Speeding up the computation of an elliptic curve using addition-subtraction chains, *Inform. Theory Appl.* 24 (1990), 531-543.