Resistance of Randomized Projective Coordinates Against Power Analysis

William Dupuy and Sébastien Kunz-Jacques

DCSSI Crypto Lab,
51, bd de Latour-Maubourg, 75700 PARIS 07 SP
william.dupuy@laposte.net
kunzjacq@yahoo.fr

Abstract. Embedded devices implementing cryptographic services are the result of a trade-off between cost, performance and security. Aside from flaws in the protocols and the algorithms used, one of the most serious threats against secret data stored in such devices is Side Channel Analysis.

Implementing Public Key Cryptography in low-profile devices such as smart cards is particularly challenging given the computational complexity of the operations involved. In the area of elliptic curve cryptography, some choices of curves and coefficient fields are known to speed up computations, like scalar multiplication. From a theoretical standpoint, the use of optimized structures does not seem to weaken the cryptosystems which use them. Therefore several standardization bodies, such as the NIST, recommend such choices of parameters. However, the study of their impact on practical security of implementations may have been underestimated.

In this paper, we present a new chosen-ciphertext Side-Channel Attack on scalar multiplication that applies when optimized parameters, like NIST curves, are used together with some classical anti-SPA and anti-DPA techniques. For a typical exponent size, the attack allows to recover a secret exponent by performing only a few hundred adaptive power measurements.

1 Introduction

The use of elliptic curves for cryptographic purposes was proposed by Miller [10] in 1985 and Koblitz [8] in 1987. Since then, it became an essential part of public key cryptography. In particular, many cryptosystems rely on the intractability of the discrete logarithm problem (DLP) on elliptic curves. The main advantage of this problem is that it is believed to be harder to solve than other number-theoretic problems. As a consequence, for a similar security level, it is possible to use smaller objects than with integer factorization for example. This property is especially attractive for embedded systems, where storage requirements and computation times are critical.

Cryptosystems relying on DLP on elliptic curves use the *scalar multiplication* operation in some large elliptic curve group $(G, +)$

$$P \in G \rightarrow kP \tag{1}$$

where k is a secret data. Because of DLP hardness, it is believed to be infeasible to compute k from the knowledge of one or several pairs (P, kP).

In a situation where no reasonable attack on a cryptographic algorithm is known, Kocher first observed in 1996 [9] that the measurement of the algorithm computation time could still reveal secret information. This paved the way to Side Channel Attacks that take advantage of the measurement of physical signals emitted by a cryptographic device during a computation to gain access to secret data.

Since then, several examples of Side Channel Attacks led to various countermeasures being developed. Concerning scalar multiplication in EC groups, the use of scalar multiplication algorithms with a regular computation flow like *double-and-add always* or *Montgomery Ladder* is an answer to Simple Power Analysis (SPA), while randomized projective coordinates, first proposed by [4], are used to counter Differential Power Analysis (DPA).

In this paper, we present a new side-channel attack against scalar multiplication implementing these countermeasures, when the EC group used is chosen among the NIST [12], ANSI [1] or SEC [13] recommended curves. It is a Goubin-style attack [6] that uses distinguished points whose presence can be detected along the computation by an observation of power traces despite the randomization countermeasure. It leverages the particular shape of the underlying coefficient fields.

The paper is organized as follows. We first briefly review some facts about elliptic curves in section 2. Then section 3 presents some classical Side Channel Attacks and common countermeasures to prevent them. Finally, sections 4 and 5 present the details of our attack.

2 Elliptic Curves

2.1 Elliptic Curve Equation

Let $\mathbb{K}$ be a finite field of characteristic p. Over this field, we set the equation (E)

$$y^2 + a_1xy = x^3 + a_2x^2 + a_4x + a_6$$

The elliptic curve (C) associated to (E) is the set of all points of $\mathbb{K}^2$ satisfying (E), together with a particular point $\mathcal{O}$ called *point at infinity.* $\mathbb{K}$ is the *coefficient field* of the curve.

Up to an affine change of variables, if $p = 2$, we can set $a_1 = 1$ and $a_4 = 0$. The equation can then be rewritten $y^2 + xy = x^3 + a_2x^2 + a_6$. If $p \geq 3$, we can set $a_1 = a_2 = 0$ and then (E) becomes $y^2 = x^3 + a_4x + a_6$.

Together with an addition law, this set forms a commutative group. We do not describe the group law here since it does not play any role in the attack we present.

2.2 Affine and Projective Representation

A point on a curve of equation (E) is a solution of (E). Therefore the simplest representation of a point on a curve of equation (E) is the corresponding solution of (E) in $\mathbb{K}^2$. This is the *affine* representation.

Nevertheless, other representations can be preferred. We are mainly interested in *projective coordinates*. Given $P = (x, y)$ in affine coordinates, its representation in projective coordinates is $P = (xZ, yZ, Z)$ for any $Z \in \mathbb{K}^*$. If a finite solution of (E) is represented by (α, β, γ), then $\gamma \neq 0$. The point at infinity $\mathcal{O}$ is represented by $(0, \beta, 0)$ for any $\beta \neq 0$.

The projective representation is not unique. In fact, for some finite solution (x, y) of (E) with $x \neq 0$ and $y \neq 0$, any of the three projective coordinates can take an arbitrary value in $\mathbb{K}^*$. This observation is the basis of the randomized projective coordinates countermeasure, which we will describe in section 3.2. Projective representation is also used to increase the efficiency of point addition computations since for example, it allows to compute the group law without having to perform modular inversion in the coefficient field.

2.3 Recommended Coefficient Fields for NIST Elliptic Curves

Curves recommended by standardization bodies such as NIST, ANSI, or SEC are usually defined over $\mathbb{F}_p$, or $\mathbb{F}_2[x]/(P)$ where P a primitive polynomial. We focus on NIST recommended curves from now on. Other standardized curves present similar properties as the ones of the NIST.

Curves Defined on Binary Fields. The coefficient field is here of the form $\mathbb{F}_2[x]/(P)$. The primitive polynomials standardized by the NIST are:

$$\begin{aligned} P_{233}(x) &= x^{233} + x^{74} + 1 \\ P_{283}(x) &= x^{283} + x^{12} + x^7 + x^5 + 1 \\ P_{409}(x) &= x^{409} + x^{87} + 1 \\ P_{571}(x) &= x^{571} + x^{10} + x^5 + x^2 + 1 \end{aligned}$$

We can notice that these polynomials are very sparse. This has to do with hardware efficiency.

Curves Defined on Prime Order Fields. For these curves, the coefficient field is $\mathbb{F}_p$, with p among

$$\begin{aligned} p_{192} &= 2^{192} - 2^{64} - 1 \\ p_{224} &= 2^{224} - 2^{96} + 1 \\ p_{256} &= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\ p_{384} &= 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1 \\ p_{521} &= 2^{521} - 1 \end{aligned}$$

As in the binary case, the sparse form of these primes simplifies and speeds up operations in the coefficient field.

2.4 Scalar Multiplication

As already mentioned, cryptosystems relying on discrete logarithm on elliptic curves make a consistent use of *scalar multiplication*. Given a public point P on the elliptic curve, and a secret scalar k, this operation consists in computing kP.

Let us write $k = \sum_{i=0}^{n-1} k_i 2^i$. The most basic algorithm that computes kP given P and a "black-box" implementation of the group law is the following:

```
[double-and-add from MSB to LSB]
INPUT: P in C
R=0
for i from n-1 to 0
     R <- 2R
     if k_i=1
           R <- R+P
end for;
RETURN R
```

3 Side Channel Attack and Common Countermeasures

3.1 Classes of Attacks

SPA: Simple Power Analysis applies when the sequence of operations performed during some computation depends on a secret value. When the operations used are sufficiently complex, they can be easily detected by physical measures and the sequence of operations performed can be retrieved.

For instance, in a *double-and-add* algorithm, an addition is performed only if the corresponding bit of k is set to 1. Assuming that doubling and adding have noticeably different power consumption signatures, one observation of a power consumption curve can be enough to extract the secret exponent value.

DPA: Differential Power Analysis was introduced by [3] on DES implementations, but it applies to public-key cryptography as well.

For DPA to work, some intermediate value v manipulated by a cryptographic device must depend on known input and output values and on a few secret bits. The power consumption of some operation manipulating v is measured for several input values. To each value k of the secret bits involved corresponds a partition of the input and output messages into subsets leading to the same value for v. A guess for k can be checked as follows: if the value of k is correct, averaging the power consumption inside these subsets should yield noticeably different results among subsets. If it is wrong, results should be roughly identical no matter the subset chosen.

Goubin-Style Attacks. L. Goubin [6] first noticed that some properties of intermediate values may be invariant under randomization. For example, if a coordinate of some projective point representation is zero, it remains equal to zero whatever the randomization applied. If such a remarkable property can be detected, an attack can be built as follows: input values are chosen so that a remarkable value appears during the computation only if some hypothesis about a secret is correct. The measure then allows the attacker to test his hypothesis.

The attack we present follows this framework.

3.2 Countermeasures

Many countermeasures have been developed to make the attacks presented in section 3.1 impractical. Most widely used ones are presented here.

Regularization of the Instructions Flow. For an algorithm to be protected against SPA, its instruction flow must not depend on secret values. *Double-and-add always*, or *Montgomery ladder* [11] are examples of such algorithms:

```
[Double-and-add always from MSB to LSB]
INPUT: P in C
R[0]=0
for i from n-1 to 0
      R[0]<- 2R[0]
      R[1]<- R[0]+P
      R[0]<- R[k_i]
end for;
RETURN R[0]

[Montgomery ladder]
INPUT: P in C
R[0]=0;R[1]=P
for i from n-1 to 0
      R[1-k_i]<- R[k_i]+R[1-k_i]
      R[k_i]<- 2R[k_i]
end for;
RETURN R[0]
```

Randomization of Data Representation. is targeted at DPA. If the representation of temporary values is randomized, an intermediate value does not depend only on inputs and key bits, but also on some random data out of control of the attacker. Consequently, aggregating measures is no longer possible.

In the case of scalar multiplication, expressing a point in randomized projective (or Jacobian) form, as suggested by Coron [4], is a common instantiation of this countermeasure.

Randomization of the computation flow In order to prevent Goubin-style attacks, randomness can be introduced in the sequence of operations performed. Here are two examples of techniques applied to the scalar multiplication $P \to kP$:

- **Point blinding** The hardware computes $k(P+R)$ and kR, for some random point R, separately [4] or together using a trick due to Shamir [7].
- **Random exponent** If q is the order of the underlying group, then $qP = 0$. Therefore if $(k+rq)P$ is computed instead of kP for some random value r, the final result is unchanged, but the binary representation of the secret key is scrambled by the addition of rq all along the computation.

4 The Attack: Theory

4.1 Assumptions on the Target Device

We aim at retrieving the n-bit secret scalar k stored in a cryptographic device performing scalar multiplication $P \to kP$ for any point P of our choice, on an elliptic curve whose coefficient field is defined by a sparse polynomial for the binary field case or a "sparse" prime for the prime field case (see 2.3).

An element e in the coefficient field can always be written $e = \sum_{i=0}^{n-1} e_i u^i$ with $u = 2$ if $\mathbb{K} = \mathbb{F}_p$, and $u = x$ if $\mathbb{K} = \mathbb{F}_{2^n} = \mathbb{F}_2[x]$. Since we will observe Hamming weights during the attack, we assume that our target crypto device represents e in the standard way by the binary string $\{e_i\}$.

The secret scalar, on the other hand, is an object of $\mathbb{Z}/q\mathbb{Z}$ where q is the number of elements of the chosen elliptic curve group. We will write

$$k = \sum_{i=0}^{n-1} k_i 2^i$$

The attack we propose applies to implementations having the following properties:

- Points are represented with randomized projective coordinates.
- No randomization of the computation flow is performed.

We focus on *double-and-add always* from the MSB to the LSB or on the Montgomery ladder. However, the particular choice of the scalar multiplication algorithm used is irrelevant, and we target more generally algorithms that perform one computation step per exponent bit. We suppose that in step j the point $K_j P$ is manipulated, with

$$K_j = \sum_{i=n-1-j}^{n-1} k_i 2^{i-(n-1-j)}$$

On the measurement side, we assume we have access to the Hamming weights of the values manipulated, up to some noise.

4.2 Overview of the Attack

Suppose that some special point P_0 can be distinguished from a random point, for example by power analysis. Since we assumed in section 4.1 that on input P and during step j, the multiplication algorithm manipulates

$$K_j.P = \left(\sum_{i=n-1-j}^{n-1} k_i 2^{i-(n-1-j)} \right) P$$

asking for the computation of $k.(1/K_j P_0)$ makes P_0 appear at the j-th step of computation. Because $K_j = 2K_{j-1} + k_{n-1-j}$, assuming K_{j-1} is known, the value of the next unknown bit k_{n-1-j} can be recovered as follows:

Assume that $k_{n-1-j} = 0$ and that consequently $K_j = 2K_{j-1}$. Observe the computation of $k\,(1/K_j)P_0$. If P_0 is detected at step j, the hypothesis on bit k_{n-1-j} was correct. Otherwise, $k_{n-1-j} = 1$ and $K_j = 2K_{j-1} + 1$.

The above applies for $j = 0$ as well with $K_{-1} = 0$.

For each bit, several computations might be performed to improve the reliability of the guess of k_{n-j}. Then, by iterating this algorithm, the whole secret k can be extracted.

4.3 Using Hamming Weights to Build a Distinguishable Point

We choose a point of the form

$$P_0 = (u^\lambda, y)$$

in affine coordinates, with λ as small as possible. Its representation in projective form is $P_0 : (X = u^\lambda Z, Y = yZ, Z)$ for some random $Z \in \mathbb{K}^*$.

For each value of λ we can expect that there is a point with abscissa u^λ with probability $1/2$: in $\mathbb{F}_p$, this is the case if and only if $2^{3\lambda} + a2^\lambda + b$ is a square, while in $\mathbb{F}_{2^n}$, it depends on whether the polynomial $p(y) = y^2 + x^\lambda y + x^{3\lambda} + ax^{2\lambda} + b$ has roots. For all NIST curves, λ can be chosen ≤ 5.

Detecting the Distinguishable Point. Because of the form of common coefficient fields such as NIST fields, we show in sections 4.4 and 4.5 that for a random Z, $X = u^\lambda Z$ is close to Z rotated by λ bits on the left ($Z <<< \lambda$), therefore

$$U = \mathsf{Ham}(X) - \mathsf{Ham}(Z)$$

is small. At the opposite, for a random point where coordinates are uncorrelated, U has mean 0 and variance $\mathsf{V}(U) = 2(n/4) = n/2$. Therefore,

We measure U to discriminate P_0 from a random point.

As usual, increasing the number of experiments decreases the error probability; several scalar multiplications lead to as many observations of U as necessary. Statistical tests can then be performed as described in section 5.1 to make a decision according to the observations.

Now, let us estimate the Hamming distance between $u^{\lambda}Z$ and $(Z <<< \lambda)$ on both fields types.

4.4 Binary Fields

Let $P(x) = 1+x^n+\sum_{i=1}^{I} x^{m_i}$ with $1 \leq m_i < m_{i+1} < n$ be a primitive polynomial over $\mathbb{Z}_2[X]$ of degree n. Let $e = n-\deg(P(x)-x^n) = m_I$. We assume that $e > \lambda$; this is true for NIST curves which satisfy $e > n/2$. More generally, multiplication optimization in $\mathbb{F}_{2^n} = \mathbb{F}_2[x]/(P)$ commands to choose e large.

Let $Z \in \mathbb{F}_{2^n}$. Remember that elements of $\mathbb{F}_{2^n}$ are represented in the usual polynomial base. For $\lambda < e$, set $Z = Z_1 + x^{n-\lambda}Z_2$ with $\deg(Z_1) < n - \lambda$ and $\deg(Z_2) < \lambda$: $(Z <<< \lambda) = x^{\lambda}Z_1 + Z_2$.

$x^{\lambda}Z$ and $(Z <<< \lambda)$ mod P are related by:

$$\begin{aligned} x^{\lambda}Z &\equiv x^{\lambda}Z_1 \oplus x^n Z_2 \\ &\equiv x^{\lambda}Z_1 \oplus (x^n - P)Z_2 \\ &\equiv (Z <<< \lambda) \oplus Z_2 \oplus (x^n - P)Z_2 \\ &\equiv (Z <<< \lambda) \oplus \sum_{i=1}^{I} x^{m_i}\, Z_2 \end{aligned}$$

Since $\lambda < e$, the above result is the reduced expression of the difference mod P. Each term $x^{m_i}Z_2$ is a λ-bit pattern that can affect at most a λ-bit window of the difference. Therefore at most $I\lambda$ bits differ from Z and $x^{\lambda}Q$.

Under the assumption that the λ-bit windows do not overlap, the exact computation of the probability law of $\mathsf{Ham}(x^{\lambda}Q) - \mathsf{Ham}(Q)$ can be carried out; this is useful to improve the attack (see section 5.1, Neyman-Pearson). The computation is performed in appendix A.1. The non-overlapping assumption is satisfied for the NIST curves P_{233} and P_{409}.

4.5 Prime Fields

We work here in $\mathbb{F}_p$ with p is prime. This case is more complex than the binary case because of the carry propagations that occur while adding values mod p.

Let e be the the greatest integer such that $2^n - 1 - p < 2^{n-e}$. For all NIST curves, $e \geq 32$. Distinguished points for curves on prime fields satisfy $\lambda \leq 3$: thus we always have $e - \lambda \geq 29$.

Let $Z \in \mathbb{F}_p$, $Z = Z_1 + 2^{n-\lambda}Z_2$, with $Z_1 < 2^{n-\lambda}$ and $Z_2 < 2^{\lambda}$.

$(Z <<< \lambda) = 2^{\lambda}Z_1 + Z_2$ and

$$\begin{aligned} 2^\lambda Z &\equiv 2^\lambda Z_1 + 2^n Z_2 \,[p] \\ &\equiv 2^\lambda Z_1 + (2^n - p) Z_2 \,[p] \\ &\equiv (Z <<< \lambda) + \Delta \,[p] \quad \text{with } \Delta = (2^n - 1 - p) Z_2 \end{aligned}$$

Since $Z_2 < 2^\lambda$ and $2^n - 1 - p \leq 2^{n-e}$, $\Delta < 2^{n-(e-\lambda)}$. Since $p \geq 2^n - 2^{n-e}$, with probability around $1 - 2^{e-\lambda} \geq 1 - 2^{-29}$, $(Z <<< \lambda) + \Delta$, viewed as an integer, is reduced mod p (i.e. it lies in the interval $[0, p-1]$). We can thus forget reduction mod p and study the effect of adding Δ to $(Z <<< \lambda)$ in $\mathbb{Z}$.

Sparse primes like NIST primes satisfy relations of the form $2^n - 1 - p = \sum_{i=1}^{I} \varepsilon_i \, 2^{m_i}$, with I small and $\varepsilon_i = \pm 1$ (see section 2.3; in the NIST case, $I \leq 3$). Therefore $\Delta = \sum_{i=1}^{I} \varepsilon_i \, 2^{m_i} \, Z_2$. Δ is composed of I λ-bit blocks; we now assume as in the binary case that these blocks do not overlap, and this hypothesis is fulfilled for all NIST curves mod p.

On average, carries beyond λ-bit blocks of multiples of Z_2 ("block carries") do not change $U = \mathsf{Ham}(Z <<< \lambda) - \mathsf{Ham}(2^\lambda Z)$, and have a small influence on $\mathsf{V}(U)$ as shown in appendix A.2. Since inside each block the Hamming weight is not changed on average, $\mathsf{E}(U) = 0$ as in the binary case. Excluding the block carries, at most $I\lambda$ bits differ between $2^\lambda Z$ and Z.

5 The Attack: Practice

5.1 Statistical Tests

During the course of the attack, we target some specific bit k_{n-j} manipulated during step $j+1$. We compute m times $k.(1/(2K_j)P_0)$ and collect m measures U_i, $1 \leq i \leq m$, of U. We must then choose a guess for k_{n-j} depending on $S = (U_1, \ldots, U_m)$. Let $\mathcal{D}_h$ be the law of U if $k_{n-j} = h$, $\mathsf{P}_{\mathcal{D}_0}(U = k) = p_{k,0}$ and $\mathsf{P}_{\mathcal{D}_1}(U = k) = p_{k,1}$.

Neyman-Pearson Test. It is well known from the Neyman-Pearson lemma that the test that has the smallest error probability if both hypothesis on k_{n-j} are equally likely, consists in computing the probability of the sample S observed according to both hypotheses, and to select the hypothesis $k_{n-j} = h$ for which the probability of the sample is the highest; this is the hypothesis that explains best the observed value. Knowing the $p_{k,h}$, one can compute the probability of S under hypothesis h through

$$\mathsf{P}_{\mathcal{D}_h}(S) = \mathsf{P}_h = p_{U_1,h} p_{U_2,h} \cdots p_{U_m,h} \tag{2}$$

Test Based on a Variance Estimator. While the Neyman-Pearson test on S is optimal, it requires the exact knowledge of $\mathcal{D}_0$ and $\mathcal{D}_1$. A slightly less efficient, but simpler test consists in estimating the variance of S. If $k_{n-j} = 0$, $\mathsf{V}(U) = V_0 = (I\lambda)/2$ (binary case) or $(I(\lambda+1))/2$ (prime case), whereas if $k_{n-j} = 1$, $\mathsf{V}(U) = V_1 = n/2$.

After m experiments, $\mathsf{V}(U)$ is estimated by $\overline{V} = \frac{1}{m}\sum_{i=1}^{m} U_i^2$. The probability P_h that $\overline{V}$ takes some specific value under $\mathcal{D}_h$ is then computed by approximating both laws $\mathcal{D}_0$ and $\mathcal{D}_1$ by normal laws[1]: the law of $\overline{V}$ under $\mathcal{D}_h$ is approximated by V_h/m times a χ_2 with m degrees of freedom. The Neyman-Pearson decision rule is then used *on* $\overline{V}$: $k_{n-j} = 0$ is decided if and only if $\mathsf{P}_0 > \mathsf{P}_1$.

Necessary Number of Experiments. The error probability of the Neyman-Pearson decision rule on some function f of the observation S *for one experiment* depends on the statistical distance between $f(\mathcal{D}_0)$ and $f(\mathcal{D}_1)$

$$\sum_k |\mathsf{P}_{\mathcal{D}_0}(f(S) = k) - \mathsf{P}_{\mathcal{D}_1}(f(S) = k)|$$

and similarly, on several experiments, the distance between $f(\mathcal{D}_0) \times \ldots f(\mathcal{D}_0)$ and $f(\mathcal{D}_1) \times \ldots \times f(\mathcal{D}_1)$ could be computed. However, this is not practical. Some approximations exist, like the Kullback-Leibler distance, or the Square Euclidean Imbalance (see [2] or [5]). Very roughly, they state that for a constant error rate the number of experiments depends on the distributions like $\left(\sum_k [\mathsf{P}_{\mathcal{D}_0}(f(S) = k) - \mathsf{P}_{\mathcal{D}_1}(f(S) = k)]^2\right)^{-1}$.

Practically, we prefer adaptive strategies that estimate on the fly the error probability.

Adaptive Strategies. If m measures are performed, resulting in some observation S of probability P_h under $\mathcal{D}_h$, the probability that hypothesis h actually holds is

$$\mathsf{P}(h = 0|S) = \frac{\mathsf{P}_0}{\mathsf{P}_0 + \mathsf{P}_1} \quad \text{and} \quad \mathsf{P}(h = 1|S) = \frac{\mathsf{P}_1}{\mathsf{P}_0 + \mathsf{P}_1}$$

During a series of m experiments, *m being a fixed value*, the probability ratio $\mathsf{P}(h = 0|S)/\mathsf{P}(h = 1|S) = \mathsf{P}_0/\mathsf{P}_1$ indicates the confidence in the decision made. In the experiments we perform, some threshold $\delta > 1$ is set. We perform more experiments as long as $1/\delta < \mathsf{P}_0/\mathsf{P}_1 < \delta$. If $\mathsf{P}_0 > \delta\mathsf{P}_1$ we decide $h = 0$, and if $\mathsf{P}_1 > \delta\mathsf{P}_0$ we decide $h = 1$. Since the number of experiments is computed adaptively, experiments are no longer independent and for example (2) is not strictly true anymore. However we assume that the confidence estimation $\mathsf{P}_0/\mathsf{P}_1$ is still meaningful.

Recovering the Whole Key. Even if the error probability for each bit guess is small, since we are dealing with large secret values (at least 192 bits), the probability that at least one error occurs during the attack is high. Additionally, after one error at step j, since next experiments rely on the value of K_j, subsequent tests will fail to detect P_0 and with high probability, the next guessed bits will be equal to 1.

Of course, one way to overcome this problem is to have a very low error probability per bit. However, more subtle approaches can be devised: for example, if

[1] this is justified by the central limit theorem for $\mathcal{D}_1$; for $\mathcal{D}_0$, this can be considered as an heuristic hypothesis.

a long run of ones is guessed, one can attempt to restart from the computation step where the run begins.

5.2 Experimental Results

We simulated a Montgomery Ladder using randomized projective coordinates on the various NIST curves. We used the most basic variance estimator, with no backtracking in case of long runs of ones. We looked for the number of measurements required to guess the whole secret scalar with a success probability of 90%. No noise was added to the measurements, unlike in a real setting.

The number of measurements that had to be performed in order to reach a confidence level of 90% does not grow linearly in the size of the scalar. In fact, it depends on $I\lambda$; this is to be expected because of the expression of $\mathsf{V}(U)$ under the hypothesis $k_{n-1-j} = 0$.

Current results are summarized in table 1 below.

Table 1. Experiments Required for a 90% Confidence Level

Curve	Total number of experiments	Experiments per bit	λ	$I\lambda$
p_{192}	1117	6	2	2
p_{224}	2347	10	6	6
p_{256}	2729	11	4	12
p_{384}	2519	7	1	3
p_{521}	1305	3	n.a.	0
B_{233}	482	2	1	1
B_{283}	1854	7	5	15
B_{409}	789	2	1	1
B_{571}	2219	4	5	15

6 Conclusion

In this paper, we presented a new chosen-ciphertext Side-Channel Attack on elliptic curve scalar multiplication. It does not apply to any elliptic curve, but rather to curves whose coefficient fields are chosen to enable efficient implementations on resource-constrained hardware; unfortunately, this kind of hardware is precisely the target of choice for Side-Channel Attacks.

The attack is able to defeat some widely used countermeasures like anti-SPA scalar multiplication algorithms and projective coordinate randomization. It is stopped by more complex defenses like point blinding and scalar randomization; these countermeasures do not however come for free in hardware. The attack might also be prevented if the cryptanalyst cannot have full control over the scalar multiplication input.

Practically, basic simulations show that the attack is able to recover a secret scalar with a success rate of 90% on any NIST curve using no more than 11 power measurements per bit guessed, using a very simple statistical test. This lead us to think that it is a practical threat that should be taken into account by implementors.

References

1. ANSI X9.62-1998. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), 1998.
2. T. Baigneres, P. Junod, and S. Vaudenay. How Far Can We Go Beyond Linear Cryptanalysis? In *Advances in Cryptology – ASIACRYPT' 04*, volume 3329 of *LNCS*, pages 432–450. Springer-Verlag, 2004.
3. C.Kocher, J.Jaffe, and B.Jun. Differential Power Analysis. In *Advances in Cryptology – CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer-Verlag, 1999.
4. J.-S. Coron. Resistance Against Differential Analysis for Elliptic Curve Cryptography. In *Advances in Cryptology – CHES'99*, volume 1717 of *LNCS*, pages 292–302. Springer-Verlag, 1999.
5. W. Feller. *An Introduction To Probability Theory and Its Applications.* Wiley Series In Probability And Mathematical Statistics. John Wiley & Sons, 1968.
6. L. Goubin. A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems. In *Advances in Cryptology – PKC'03*, volume 2567 of *LNCS*, pages 199–210. Springer-Verlag, 2003.
7. K. Itoh, T. Izu, and M. Takenaka. Efficient Countermeasures Against Power Analysis for Elliptic Curve Cryptosystems. In *CARDIS*, pages 99–114, 2004.
8. N. Koblitz. Elliptic Curve Cryptosystems. In *Mathematics of Computation*, volume 48, pages 203–209. Springer-Verlag, 1987.
9. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer-Verlag, 1996.
10. V. Miller. Use of Elliptic Curve in Cryptography. In *Advances in Cryptology – CRYPTO' 85*, volume 218 of *LNCS*, pages 417–426. Springer-Verlag, 1985.
11. P. Montgomery. *Speeding the Pollard and Elliptic Curves Methods of Factorization*, volume 44. Math. Comp, 1985.
12. NIST. Recommended Elliptic Curves for Federal Government Use, 2000.
13. Standards for Efficient Cryptography Group/ Certicom Research. SEC 2: Recommended Elliptic Curve Cryptography Domain Parameters, Version 1.0, 2000. `http://www.secg.org`.

A Computation of the Probability Law of U

A.1 Binary Case

In that section, we compute the exact probability law of the Hamming weight difference between $x^\lambda Z$ and $Z <<< \lambda$, under the assumption that the λ-bit windows do not overlap.

We use the same notations as in 4.4: $\mathbb{F}_{2^n} = \mathbb{F}_2[X]/(P)$ and Z is a random uniform value in F_{2^n}. For some $\lambda < n - k_I$,

$$U = \mathsf{Ham}(Z) - \mathsf{Ham}(x^\lambda Z)$$

is the random variable whose law we want to compute. We saw in section 4.4 that if $Z = Z_1 + x^{n-\lambda} Z_2$ with $\deg(Z_1) < n - \lambda$ and $\deg(Z_2) < \lambda$, then

$$x^\lambda Z \equiv (Z <<< \lambda) \oplus \sum_{i=1}^{I} x^{k_i}.Z_2$$

Set $(Z <<< \lambda) = \sum_{i=0}^{n-1} z_i x^i$. Then $Z_2 = \sum_{j=0}^{\lambda-1} z_j\, x^j$. Let U_j be the contribution of the j-th bit of Z_2, z_j, to U. Under the non-overlapping condition, $U = U_0 + \ldots + U_{\lambda-1}$ and

$$\begin{aligned} U_j &= \mathsf{Ham}(Z <<< \lambda) - \mathsf{Ham}\left((Z <<< \lambda) \oplus z_j \sum_{i=1}^{I} x^{k_i+j}\right) \\ &= z_j \sum_{i=1}^{I} (2 z_{k_i+j} - 1) = z_j \left(2 \sum_{i=1}^{I} z_{k_i+j} - 2I\right) \end{aligned}$$

and for each i, j, z_j and z_{k_i+j} are independent because $k_i \neq 0$. If W is a binomial random variable $\mathcal{B}(I, 1/2)$,

$$\begin{aligned} \mathsf{P}(U_j = k) &= \frac{1}{2}\,\mathsf{P}(2W - I = k) \quad \text{if } k \neq 0 \\ \mathsf{P}(U_j = 0) &= \frac{1}{2} + \frac{1}{2}\,\mathsf{P}(2W = I) \end{aligned}$$

In particular, $\mathsf{E}(U_j) = 0$, and $\mathsf{V}(U_j) = I/2$. Now $U_0, \ldots, U_{\lambda-1}$ depend on different bits of Z and are therefore independent: the law of U is simply the law of the sum of λ independent "copies" of U_0, for example. In order to implement a Neyman-Pearson test on outcomes of U, its law can therefore be derived by computing the λ-th convolution power of the law of U_0. In order to perform variance tests, we only need $\mathsf{E}(U) = 0$, and $\mathsf{V}(U) = I\lambda/2$.

A.2 Large Prime Case

In the prime field case, we want to approximate the law of $U = \mathsf{Ham}(Z) - \mathsf{Ham}(2^\lambda Z)$, where $0 \leq Z < p$ is random and the Hamming weight is computed on reduced representations mod p.

In section 4.5, we proved that the law of U is very close to the law of

$$U' = \mathsf{Ham}(Z') - \mathsf{Ham}(Z' + \Delta)$$

with Z' a random value in $[0, 2^n - 1]$,

$$\Delta = (2^n - p - 1)(Z' \mod 2^\lambda) = \sum_{i=1}^{I} \varepsilon_i\, 2^{m_i}\, (Z' \mod 2^\lambda)$$

and $\varepsilon_i = \pm 1$. Set $Z' \mod 2^\lambda = Z'_2$. λ copies of Z'_2 are added or subtracted at I different λ-bit windows in Z. For the prime numbers we consider, $m_{i+1} - m_i \gg \lambda$ and we will therefore assume that these windows do not overlap, and even more, that no carry can propagate from one window to the other. We will handle separately bit differences occurring inside these windows and bit differences outside them, caused by carries overflowing the windows. The first category of bit differences will be enumerated by a random value U'_i, and the second one by U'_o: $U' = U'_o + U'_i$. We will assume that U'_i and U'_o are independent.

The contribution c_i of each λ-bit window to U'_i is a random binomial value satisfying $c_i/2 - 1 \sim \mathcal{B}(\lambda, 1/2)$, and these contributions are independent because they involve independent bits of Z' (and although they both involve Z'_2). Therefore $U'_i/2 - 1 \sim \mathcal{B}(I\lambda, 1/2)$.

Let us focus on the contribution c_o of a term $2^{m_i} Z'_2$ to U'_o, corresponding to a case $\varepsilon_i = 1$. With probability $1/2$, no carry occurs and $c_o = 0$. If a carry occurs, c_o contributes to U'_o in the following way:

Contribution	1	0	-1	$-i$
Probability	$1/4$	$1/8$	$1/16$	$2^{-(i+2)}$

For example, in the second case of the above table, two bits 01 in Z' are changed by the carry into 10.

In fact, $c_0 = b(1 - Z)$ where b is a Bernoulli variable that is equal to one if and only if a carry occurs, $\mathsf{P}(Z = i) = 2^{-i+1}$ for $i \geq 0$, and b and Z are independent. With the help of this expression, one can check that $\mathsf{E}(c_o) = 0$ and $\mathsf{V}(c_o) = 1/2$. We would have obtained the same result for $\varepsilon_i = -1$, although c_0 would be changed into $-c_0$.

Finally, in the simplified model corresponding to the assumptions we made, $\mathsf{E}(U) = 0$ and $\mathsf{V}(U) = (I(\lambda + 1))/2$. Also note that the modeling above can be used to compute the probability law of U.