Fault Attack on Elliptic Curve with Montgomery Ladder Implementation. FDTC '08. 5th Workshop on Fault Diagnosis and Tolerance in Cryptography, pages 92–98, August 2008. IEEE-CS Press.
Errata 2008-09-10.

Fault Attack on Elliptic Curve with Montgomery Ladder Implementation*

Pierre-Alain Fouque
École normale supérieure - CNRS - INRIA
45 rue d'Ulm, 75230 Paris cedex 05, France
Pierre-Alain.Fouque@ens.fr

Reynald Lercier
DGA/CÉLAR - IRMAR, Université de Rennes
La Roche Marguerite, 35174 Bruz, France
Reynald.Lercier@m4x.org

Denis Réal
DGA/CÉLAR - INSA-IETR, Université de Rennes
La Roche Marguerite, 35174 Bruz, France
Denis.Real@dga.defense.gouv.fr

Frédéric Valette
DGA/CÉLAR
La Roche Marguerite, 35174 Bruz, France
Frederic.Valette@m4x.org

Abstract

In this paper, we present a new fault attack on elliptic curve scalar product algorithms. This attack is tailored to work on the classical Montgomery ladder method when the y-coordinate is not used. No weakness has been reported so far on such implementations, which are very efficient and were promoted by several authors. But taking into account the twist of the elliptic curves, we show how, with few faults (around one or two faults), we can retrieve the full secret exponent even if classical countermeasures are employed to prevent fault attacks. It turns out that this attack has not been anticipated as the security of the elliptic curve parameters in most standards can be strongly reduced. Especially, the attack is meaningful on some NIST or SECG parameters.

Keywords: *EC Cryptosystem, Montgomery Ladder, Fault Attack.*

1. Introduction

Fault attack is a very powerful side-channel technique to break cryptographic schemes. The idea is to inject a fault during the computations of an implementation and to use the faulty outputs to deduce information on the secret key stored in the secure component. Boneh *et al.* first introduced this model in 1997 [?] and show how to recover secret keys of RSA and DLog-based cryptosystems. Since these attacks and other side-channel attacks, many countermeasures have been proposed so far and some implementations are believed to be more secure than others.

Elliptic curve cryptosystems are very important in smart card products since the computational cost is strongly reduced. Indeed such schemes allow to use smaller keys since algorithms to compute discrete logarithms are less efficient in such groups than in the multiplicative group of a finite field. Elliptic curves can be used in signature schemes as in the standard ECDSA or in encryption schemes as in the El Gamal cryptosystem. Consequently, it is useful to have secure implementations of the scalar product algorithm. In 1999, Coron developed three countermeasures to withstand SPA and DPA attacks on elliptic curve scalar binary exponentiations [?]. However, the first two were shown to be inefficient by Fouque and Valette in [?] and the third one by Goubin in [?]. All these attacks are SPA or DPA attacks. However, other implementations of the scalar product algorithm, for instance, Montgomery algorithm when the y-coordinate is not used, are still believed to be secure.

1.1. Previous Work

Fault attacks on elliptic curve cryptosystems appeared since 2000 by Biehl *et al.* at Crypto'00 [?]. The idea is to change the input points or the curve parameters or also the base field so that the computations is performed on a different and weakly secure cryptographic curve. On such curves, the discrete logarithms can be easy to compute using Pohlig-Hellman and Rho-Pollard algorithm for instance. Then, Ciet and Joye in [?] extended this attack by reducing the power of the attacker so that random and unknown errors can be used instead of controlled ones.

Recently at FDTC'06, Blömer *et al.* [?] mounted another side-channel attack by changing the sign of the y-coordinate. They also claimed that the attacks of Biehl *et al.* and of Ciet and Joye can be easily avoided since it is sufficient to verify at the end of the computation whether

*January 10, 2015

the resulting point is on the curve or not. Moreover, they proposed an attack on the Montgomery algorithm when the y-coordinate is used. Furthermore, they reported the following claim of Joye and Yen in [?], that the "Montgomery ladder may be a first-class substitute of the celebrated square-and-multiply algorithm" and put as an open problem to attack Montgomery Ladder algorithm when the y-coordinate is not used.

1.2. Our Results

In this paper, we show that Montgomery method when the y-coordinate is not used can be attacked with only one or two faults during the computation. More generally, we show that a special fault attack that changes the point on the original curve to a point on a cryptographically weak curve can be not detected, while the resulting output, a point of the original strong elliptic curve, can still be used to recover the secret key. The probability of success of the attack is very high as the probability to obtain the right effect is of one half.

The basic idea is to use the twist of the elliptic curve. This curve is associated to the original curve so that a given abscissa corresponds either to a point on the curve or to a point on the twist with probability approximately one half. Consequently, by performing an error at random on some small size register of the abscissa, we can go from a point on the original elliptic curve to a point on its twist and backward. Moreover, Montgomery implementation when the y-coordinate is not used, does not care of whether the computation is performed on the original curve or on its twist. Furthermore, the attack can still be mounted even though classical countermeasures are used(such as exponent masking for example).

Finally, we discover that the elliptic curve parameters recommended in most standards do not take into account such an attack since the order of the twist is often smooth, almost never a prime number.

2. Elliptic Curve Scalar Product and Twist

An elliptic curve E defined over a field k is a (smooth, geometrically irreducible and projective) curve of genus one with a based point $\mathcal{O}$ [?]. For k a finite field $\mathbb{F}_p$ of characteristic $p > 3$, application of the Riemann Roch Theorem yields a Weierstrass projective model for E of the form $Y^2Z = X^3 + AXZ^2 + BZ^3$ where $A, B \in \mathbb{F}_p$, $4A^3 + 27B^2 \neq 0$ and where $\mathcal{O} = (0 : 1 : 0)$. Such a curve admits a group structure with neutral element $\mathcal{O}$ and efficient formulas to add points. Extensive use of the group law is done in cryptography due to the fact that numerous Diffie-Hellman problems are believed to be hard in these groups.

Most of the cryptographic schemes imply the scalar product of a point P by a secret exponent d. The classical way to perform it is the use of a binary exponentiation algorithm (cf. Fig. 1) but it turns out that these algorithms have unpleasant drawbacks in view of side channel attacks.

```
Input: $P \in E, d = (d_0, \ldots, d_{n-1}) \in \{0,1\}^n$
Output: $R = dP$
$R = \mathcal{O}$
for $j = n-1$ downto $0$ do
        $R = 2R$.
        if $d_j = 1$ then $R = R + P$
return $R$
```

Figure 1. Left to right binary algorithm

2.1 Montgomery's Algorithm

In this context, Joye et al [?] emphasize that an alternative (but less general) model due to P. Montgomery [?] could be a valuable countermeasure since no operation depends on the bit of the secret exponent. The so called Montgomery ladder 2 , was extended to all kind of elliptic curves. One attractive advantage of the latter (cf. [?]) is that there exists efficient formulas to obtain the x and z-coordinates of the sum of two points $P = (x_P : . : z_P)$ and $Q = (x_Q : . : z_Q)$ with the help of the extra point $P - Q = (x_{P-Q} : . : z_{P-Q})$. The detail of the computation can be seen in [?]. It needs only 13 multiplications, 4 squarings and 18 additions. It does not not require the use of the y coordinates. The main idea of the algorithm is to compute two values in parallel which have a difference of P. The algorithm only double points or add points with a difference of P.

(cf. Fig 2).

```
Input: $P = (x_P : . : z_P) \in E$,
           $d = (d_0, \ldots, d_{n-1}) \in \{0,1\}^n$
Output: $R = (x_R : . : z_R) = dP$
$R_0 = \mathcal{O}, R_1 = P$
for $j = n-1$ downto $0$ do
        $R_{1-d_j} = R_{1-d_j} + R_{d_j}, R_{d_j} = 2R_{d_j}$
return $R_0$
```

Figure 2. Montgomery Powering Ladder

2.2. The Twist of an Elliptic Curve

One interesting feature of this algorithm for our purposes is that the y-coordinates of the points is not needed. As a consequence, this algorithm is also valid for points $(x :$

$y\ :z) \in E$ with $y \in \mathbb{F}_{p^2}$, instead of simply $y \in \mathbb{F}_p$. This remark yields thus some interest on the subset of points with x and z-coordinates defined in $\mathbb{F}_p$ on an elliptic curve with model in $\mathbb{F}_p$, but defined over $\mathbb{F}_{p^2}$. Let us denote S, this set of points, that is

$$\begin{aligned} S &= \{(0\ :1\ :0)\} \\ &\cup \{(x\ :y\ :1) \in E(\mathbb{F}_{p^2}) \text{ with } x \in \mathbb{F}_p,\ y \in \mathbb{F}_{p^2}\}. \end{aligned}$$

Obviously, $S = \{\mathcal{O}\} \cup S^0 \cup S^1 \cup S^2$ with $S^0 = \{(x\ :0\ :1) \in E(\mathbb{F}_{p^2})$ with $x \in \mathbb{F}_p\}$ (points of order 2), $S^1 = \{(x\ :y\ :1) \in E(\mathbb{F}_{p^2})$ with $x \in \mathbb{F}_p, y \in \mathbb{F}_p^*\}$ and $S^2 = \{(x\ :y\ :1) \in E(\mathbb{F}_{p^2})$ with $x \in \mathbb{F}_p, y \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p\}$. Let α be the number of points of order two, that is the number of roots of $x^3 + ax + b$, then S contains $2p + 1 - \alpha$ distinct points. Let us furthermore assume that $E(\mathbb{F}_p)$ (that is, $\mathcal{O} \cup S^0 \cup S^1$) has got $p + 1 - c$ points, the cardinality of S^2 is thus equal to $p+c-\alpha$, and therefore the cardinality of $\mathcal{O} \cup S^0 \cup S^2$ is equal to $p + 1 + c$. The attentive reader will notice immediately that this number is exactly equal to the cardinality of the quadratic twist of E, $\tilde{E}$. In truth, it is not very surprising since the equation for $\tilde{E}$ is equal to $(\varepsilon)y^2z = x^3 + axz^2 + bz^3$, where ε is a quadratic non-residue in $\mathbb{F}_p$, it is then clear that if x is not the abscissa of a point of E, it is a point on $\tilde{E}$. Points in $\mathcal{O} \cup S^0 \cup S^2$ can therefore be easily mapped to points on the twist. Finally, we can thus deduce that if we have on input of the Montgomery's ladder algorithm an exponent d and coordinates x and z in $\mathbb{F}_p$ which are not coordinates of points of E, the result of the algorithm is equal to d times $(x\ :\ .\ :z)$ on the twist of E (since this algorithm makes no use of the coefficient ε for the twist).

In a constructive approach, this observation suggests an elegant way to implement cryptographic protocols which makes a simultaneous use of an elliptic curve and its twist, especially to get with uniform distribution elements in $\mathbb{F}_p$ from points on elliptic curves. Such a scheme was for instance proposed by Kaliski in 1991 for getting a random permutation from a random function [**?**]. More recently, Boyd *et al.* applies this idea to the field of password-authenticated exchange [**?**] to avoid partition attack, Möller to the field of public key encryption [**?**] and Chevassut *et al.* to the field of randomness extraction for the Internet Key Exchange protocol [**?**]. Of course, real implementations of these protocols must resist to side-channel attacks and in this context, one especially must take care how the switch between a curve and its quadratic twist is implemented. With the Montgomery's ladder powering algorithm, there is no need of the switch and thus this difficulty can be easily erased.

According to [**?**], it is the more efficient implementation secure against simple power analysis. Various implementations of the Montgomery ladder are available but the most interesting is when y-coordinate is not used. Moreover, according to [**?**], this implementation seems very efficient as well to defeat fault attacks thanks to a single verification that the point is on the curve (a verification that $x^3 + ax + b$ is a square is enough). We will see in the following that this might not be the case.

2.3. Generic Algorithms for Discrete Logarithms

In [**?**], Pollard describes a heuristic method, called the ρ-method, to compute discrete logarithms in a generic cyclic group with constant memory and in probabilistic time approximately equal to $3\sqrt{2^n\pi/2}$ group operations, where n is the size of the group cardinal. In particular, this algorithm can be applied on a prime order group. Then, using the Pohlig-Hellman algorithm [**?**], it is possible to recover the entire discrete logarithm in time at most $O(2^{n/2})$, even though the order is not prime.

In the sequel, we use the following facts:

- The group order of the original elliptic curve and of its twist is roughly of the same size;
- With probability approximatively one half a random abscissa corresponds to a point on the original curve or to a point on its twist;
- Montgomery algorithm when the y-coordinate is not used works without difference either on the original curve or on its twist;
- Even though the order of the group on the elliptic curve is a prime, there is no reason that the order of its twist is also a prime, and consequently generic algorithms to compute discrete logarithms in time the square-root of the largest factor can always be used.

3. Fault Attack

First, we present the fault attack on a non-secure implementation. In this case, the public point P can be chosen by the adversary and we will see that with a well chosen point, the result of the computation $d.P$ is enough to recover the secret exponent d. Note that this first attack may not apply in real applications but it helps to understand the more sophisticated one. Finally, if a countermeasure is implemented during the computation, we will see that there are two ways of recovering the value d with very few faults.

3.1. Basic attack without Countermeasure

If the computation is not protected against fault attacks (it is still possible as Montgomery ladder was said to be secure against fault attack), the attacker can choose a point P

which is not on the curve but on its twist. This can be easily done as half of the possible values of x correspond to a point on the curve and the other values of x give points on the twist. We have seen in section 2 that a Montgomery exponentiation works either on the curve or on its twist. This can be an advantage in some case but usually, no property is required on the twist as this appears in section 4. So with usual constraints, the twist of an elliptic curve is weak, *i.e.* the number of points on the twist is smooth. So the attacker can choose a point P which is on the twist. In this case, the computation gives the value $d.P$. With this value, the attacker is able with classical algorithms, such as ρ-method and Pohlig-Hellman, to compute the value $d \bmod \mathrm{ord}(\tilde{E})$ with a time complexity which is given by the square root of the largest factor of the twist order. If we assume that the number of points on the twist is random, then according to [**?**] fact 3.7, the size of the largest factor is smaller than $\mathrm{ord}(\tilde{E})^{2/3}$ with probability at least $1/2$ and the complexity of the attack is about $\mathrm{ord}(\tilde{E})^{1/3}$, which is feasible for typical sizes of elliptic curves used in crypto-applications. Once the value is computed, there is only one or two possibilities for d as the order of the curve and the order of its twist are roughly of the same size. So with only one message, an attacker is able to retrieve easily the secret scalar d.

3.2. The Classical Countermeasure

To avoid the previous attack, countermeasures can be implemented. According to [**?**], a verification that the point belongs to the curve is sufficient and may prevent fault attacks. So in the following, we assume that the scalar product checks if the point $d.P$ is on the curve. The verification can be done at the beginning or at the end of the computation or at both but this changes nothing to our attack. So we can decide that the protected algorithm is described in Fig. 3.

```
Input: a point $P$ with abscissa $x$, a scalar $d$
Output: $d.P$
Compute $d.P$
if $d.P$ is on the curve, i.e. $x^3 + ax + b$ is a square, then
    return $d.P$
else return Error
```

Figure 3. Secure implementation

Note that verifying whether a point is on a curve is equivalent to check whether x^3+ax+b is a square. Respectively, if $x^3 + ax + b$ is not a square than the point is on the twist. We see in the sequel the impact of a fault during each step of the algorithm.

3.3. Fault attack with chosen points.

If the verification process is implemented in order to resist to the previous attack, the attacker can try to modify the result of the computation before the verification. In this case, the abscissa x of the result $d.P$ can be changed by the attacker to a points Q with abscissa $x \oplus \epsilon$, denoted $d.P \oplus \epsilon$, which belongs to the curve with a probability $1/2$. So with a very high probability, the attacker obtains such a result, his main problem is that the value ϵ is unknown. According to [**?**], the attacker can assume that the fault only modifies a register of small size s, 8 or 16 bits, with respect to n, the bitsize of the order. The attacker can guess the value of ϵ and for each guess, try to find the associated d value. Unfortunately, this attack makes his computation effort larger with a factor $2^s \cdot n/s$ that makes the attack unfeasible for large registers. In fact, this basic attack can be improved easily to have the same complexity as in the previous section. For this purpose, the attacker just needs to collect two faulty results associated to the same message with different values for ϵ. With those results, he can see which registers have been modified. For example, the two results look like $x \oplus \epsilon$ and $x \oplus \epsilon'$, that is written in basis 2^s: $x_1, x_2, \ldots, x_i \oplus \epsilon, \ldots, x_{n/s}$ and $x_1, x_2, \ldots, x_j \oplus \epsilon', \ldots, x_{n/s}$. Consequently, he has only two possibilities for the result, $x_1, x_2, \ldots, x_i, \ldots, x_j, \ldots, x_{n/s}$ or $x_1, x_2, \ldots, x_i \oplus \epsilon, \ldots, x_j \oplus \epsilon', \ldots, x_{n/s}$, and he can retrieve the value of d easily.

So with only two faulty results (a faulty result happens with probability one half), the attacker is able to retrieve the secret key by solving one discrete logarithm on the twist of the initial curve.

3.4. Fault attack with a fixed point

In the previous attacks, the adversary is able to choose the value of the point P. In some specific applications, this may not be the case. For example, the point can be stored in the card as described in [**?**]. In that case, the attack still works but it needs more faults to retrieve the secret exponent. Precisely, the attacker injects a fault at the beginning of the computation (he modifies the point P) and at the end of the computation to bypass the final verification. In this way, he can collect equations of the form $d \cdot (P \oplus \epsilon_i) = Q_i \oplus {\epsilon'}_i$ for $i = 1, 2, \ldots, f$. In a first approximation, we can assume that the attacker is lucky, so the effect of the fault on P always gives a point on the twist.

Once the faulty results collected, the attacker must find the values for ϵ_i and ϵ'_i. Of course he can try all the possible values and solve the discrete logarithm on the twist and check if the result is true on the curve but this not very efficient. So the main idea is to use multiple results to eliminate the wrong values for ϵ_i and ϵ'_i. Once a unique possibility remains for each value, the attacker is in the same condition than in the previous attack and he can solve the discrete logarithm with one point on the twist.

In order to find the solution, the attacker works in a small subgroup of the twist which order is t. For each possible value of $d \bmod t$, the attacker verifies whether the pair $(\epsilon_i, \epsilon'_i)$ is a solution of the previous equation. For the right value of $d \bmod t$, a solution exists (linked with the effect of the fault). For a wrong value, the probability to find a solution is roughly $|C|.|D|/t$ where C and D respectively denote the set of all possible values for ϵ and ϵ'. To find only one solution for $d \bmod t$, the method should be applied a sufficient number of time, depending on the value t. After f faults, the number of possible solutions is $t.(|C|.|D|/t)^f$. So if the attacker chooses f and t such that $t.(|C|.|D|/t)^f < 1$, he is able to find a unique solution $d \bmod t$ with the associated values $(\epsilon_i, \epsilon'_i)$ for all the faulty results collected. With all of this information, he can solve the DLog problem $d \cdot P' = Q'$ in the twist of the curve. The complexity of the first step of the attack is $f.t.|C|.|D| = f \cdot t \cdot 2^{2s} \cdot n^2/s^2$ with $(2^{2fs} \cdot n^{2f})/(s^{2f} \cdot t^{f-1}) < 1$. The attack can be optimized if t is smooth e.g. $t = \prod t_k$. In this case, the computation can be done in all the small subgroups and the complexity is then

$$f \cdot \sum_k t_k \cdot 2^{2s} \cdot (n^2/s^2).$$

Numerical examples for these complexities are given in the next section.

If the attacker is not lucky, he has to collect more faulty results in order to have f equations with $P \oplus \epsilon_i$ on the twist (roughly $2f$ faulty results which need $4f$ experiments to be obtained). If the attacker is not able to distinguish interesting faulty results from others, he can run the previous algorithms on all the experiments and keep only the value $d \bmod t$ which is valid for the highest number of experiments. The remaining candidates can be eliminated by enlarging the value t.

4. Concrete Example with Standardized Parameters of Elliptic Curves

In this section, we compute for the different curve parameters, proposed by the NIST in [?] and by the consortium SECG [?], the security of the curve according to our attack. We evaluate the security as half of the size of the larger factor of the group order of the twist. The order of the twist can be deduced from the order of the original elliptic curve thanks to the following equation: $\mathrm{ord}(\tilde{E}) = p + 1 - \mathrm{ord}(E)$ where p is the number of field elements in the finite field where the curve is defined.

Values `secp`	P1363 IPSEC	X9.62 X9.63	NIST	Strength	Security
`112r1`	c/c/r			56	**27**
`112r2`	c/c/c			56	**31**
`128r1`	c/c/c			64	**37**
`128r2`	c/c/c			64	**59**
`160k1`	c/r/c	c/r		80	**59**
`160r1`	c/c/r	c/c		80	**59**
`160r2`	c/c/c	c/r		80	62
`192k1`	c/c/r	c/r		96	69
`192r1`	c/c/c	r/r	r	96	**48**
`224k1`	c/c/c	c/r		112	**56**
`224r1`	c/c/c	c/r	r	112	**59**
`256k1`	c/c/c	c/r		128	**50**
`256r1`	c/c/c	r/r	r	128	121
`384r1`	c/c/c	c/r	r	192	193*
`521r1`	c/c/c	c/r	r	256	231

We put in bold font, all the securities below 2^{60} since such a computation can be performed. Only one parameter leads to a prime number of the order of the twist where we put a '*'. The mention 'r' denotes parameters explicitly recommended in the standard, while the mention 'c' denotes parameters in conformance with the standard. The column "Strength" refers to the standard [?].

With the curve `secp256k1`, the order of the twist is

$$3 \times 197 \times 1559 \times 96769 \times 146849 \times 2587814237219 \times 375925338294461779 \times 1010091789365275595885630233 59.$$

So on implementations without protections, the attacker can compute the discrete logarithm in the twist with a cost of 2^{50} and retrieve the secret scalar for $n = 256$. The details of all the attacks described previously with this curve, implemented first on 8 bits registers where the fault produces all possible values on 8 bits, secondly implemented on 16 bits registers where the fault produces all possible values on 16 bits and finally implemented on 32 bits registers where the fault moves a register to zero:

Register	$\lvert C\rvert$ or $\lvert D\rvert$	f	t	preprocess
8-bit	2^{13}	4	$3 \cdot 197 \cdot 1559 \cdot 96769 > 2^{35} = (2^{26})^{4/3}$	2^{47}
16-bit	2^{20}	6	$3 \cdot 197 \cdot 1559 \cdot 96769 \cdot 146849 > (2^{40})^{6/5}$	2^{61}
32-bit	2^{3}	2	$3 \cdot 197 > 2^{6}$	2^{15}

In this example we can see that in the two cases, with less than 6 faulty results (which can be obtained with less than

24 experiments) and an overall complexity smaller than 2^{50} (except the middle one), the attacker can retrieve the secret scalar.

5. Conclusion

In this paper, we have presented a very powerful fault attack on the Montgomery ladder implementation when the y-coordinates is not used. The attack needs only few faults in a very classical and realistic fault model. The major problem of the implementation we studied is that the probability to belong to the curve for a random point is very high (due to the use of the abscissa only). In that case, the verification that the point is on the curve is not efficient. A basic countermeasure can be to repeat the verification a sufficient number of time during the computation to lower the probability of success of the attacker. For example, if $n = 160$ the verification of the intermediate point can be done every five steps of the Montgomery ladder, that gives a probability of success for the attacker of $2^{-\frac{160}{5}} = 2^{-32}$. It is not clear whether this countermeasure can be defeated but it is a challenge to find a more efficient countermeasure to avoid this attack. It is still possible to choose elliptic curves such that the order of the twist is prime [**?**] but despite it is less efficient and harder to realize, faults on the parameters of the curve as described in [**?**] might be still valid.

5.1. References

Errata 2008-09-10. It is fair to add that it was convential wisdom in the cryptographic community, prior to this work, to beware of elliptic curves with weak twists. For instance, we refer to D.J. Bernstein's article, *Curve25519: new Diffie-Hellman speed records*, published in the proceedings of PKC'06.