New Fault Attack on Elliptic Curve Scalar Multiplication

Alexey Chilikov
Passware, Research Department
chilikov@lostpassword.com

Oleg Taraskin
Moscow Engineering Physics Institute (MEPhI)
gost2001@gmail.com

Abstract

In this report we present a new fault attack that applies to some implementations of elliptic curve scalar multiplication (ECSM). We consider the fault model with 'precise control of time', 'loose control of fault location' and 'random number of faulty bits'. We show that in this fault model the secret key can be revealed with polynomial time complexity and linear number of faults. In addition, we discuss different countermeasures to resist this attack.

1 Fault Attacks

Fault attacks are a type of attacks that are based on hardware error analysis. In this case an adversary can actively influence the target device and induce faults during the computation process. After that faulty output can be analyzed and secret information (or some part of it) can be revealed.

Fault attacks were introduced by Boneh at al. in [BDL97]. In further works there were presented many types of physical influence that implement different fault models. We are not going to consider physical aspects of fault attacks. A good analysis of this issue is presented in [Otto04].

Now let's consider a brief classification of mathematical models of fault attacks on the basis of [Otto04].

So, attacks may be classified:

- By control of the fault location. There are three classes: 'no control', 'loose control' (a selected variable can be targeted), and 'complete control' (selected bits can be targeted)

- By control of the timing. There are three classes: 'no control', 'loose control' (a fault is induced in a block of a few operations), and 'precise control' (the exact time can be met)
- By the number of bits affected. There are three classes: 'single faulty bit', 'few faulty bits' (e.g., a byte), and a 'random number of faulty bits' (bounded by the length of the affected variable).
- By duration of effects. There are three classes: 'transient faults', 'permanent faults' and 'destructive faults' (when a physical structure of the chip may be destroyed irreversibly as a result of the fault).

This classification can be used to compare different fault attacks on the same targets.

2 Elliptic Curves

Elliptic curve over field $\mathbb{K}$ is a set of points $(x, y) \in \mathbb{K}^2$ which satisfies the equation

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.1}$$

with augmented infinite point O. This set is an abelian group with operation '+' that is defined by the following rules:

1. for each P, $P + O = O + P = P$
2. if $P = (x, y)$ and $Q = (x, -y)$ then $P + Q = O$
3. otherwise if $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ then $P+Q = R$ and $R = (x_3, y_3)$ is defined as

$$\begin{cases} x_3 &= \lambda^2 + a_1\lambda - (x_1 + x_2) \\ y_3 &= -y_1 - \lambda(x_3 - x_1) + a_1x_3 - a_3 \end{cases} \tag{2.2}$$

where λ is

$$\begin{cases} \dfrac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & \text{if } P = Q \\ \dfrac{y_1 - y_2}{x_1 - x_2} & \text{otherwise} \end{cases} \tag{2.3}$$

Infinity point O is a neutral element of this group. Scalar multiplication of point and integer number can be defined naturally.

There are many computationally hard problems in these groups. For example, Discrete Logarithm Problem (**ECDLP**) and Diffie-Hellman Problem (**ECDHP**) are computationally hard on arbitrary elliptic curve group over large finite field. Their complexity underlies many widespread cryptoalgorithms and cryptosystems. Elliptic curve cryptography that was firstly purposed by Miller ([Mil86]) and Koblitz ([Kob87]) now is an essential component of many software and hardware applications. Many international and national cryptographic standards used algorithms on elliptic curves.

3 Previous Works

Fault attacks on elliptic curves were firstly presented in [BMM00]. Three different attacks were considered. The first of them is based on a controlled fault in the base point before computation starts. In this case, an attacker can force target device to perform computation in another elliptic curve. If this curve is cryptographically weak (e.g., its order is B-smooth for a small B) attacker can solve **ECDLP** for this curve and reveal the secret key. This attack requires choosing of the input point and may be prevented by correctness check of the input point. Another variant of this attack uses a 'single bit fault' in the base point inside the target device.

In [CJ05] there was presented an interesting attack based on the permanent modification of elliptic curve parameters. The main idea is similar that is to force device to perform computation in a weaker curve. The complexity of this attack is subexponential if the induced fault is random. Attack may be prevented if elliptic curve parameters are protected from modification.

In a recent work [FLRV08] there were presented new attacks on Montgomery's Ladder implementation of ECSM. These attacks couldn't be prevented by correctness check of the output point. But only one implementation of ECSM is vulnerable to these attacks. Moreover, these attacks depend on target curve properties and there are some curves for which these attacks are inapplicable.

All these attacks (in contrast to ours) can be applied if only X-coordinate of the output point is available for analysis.

4 Fault Model

Any fault attack depends on the specific implementation of target algorithm and may be inapplicable for other implementation. Therefore we choose the certain implementation of elliptic curve scalar multiplication to demonstrate our attack in detail. Further we briefly discuss the applicability of similar attacks for other popular implementations.

Let the target device implement the 'square-and-add' algorithm of elliptic curve scalar multiplication. The secret key d stored in the protected area of the device. d is a positive integer and n is the number of bits of d (n is a known value). Elliptic curve parameters are public and can't be modified by adversary (e.g., they are also stored in the protected area). Let d_i be i-th bit of d.

The following pseudo-code demonstrates a target algorithm:

```
Constants: d - positive integer (secret key)
Input: P - basic point
Output: Q = dP

Q = O

for( i = n-1; i >= 0; i-- )
```

```
 Q = Q + Q       // duplicate the intermediate point
 if( d[i] == 1 )
  Q = Q + P      // add the base point to the intermediate point

return Q
```

Suppose that adversary can induce a random fault in the X-coordinate of the intermediate point Q (but not the base point P). Also suppose that the fault time may be chosen precisely.

5 Attack Scenario

A simple attack scenario is following: induce a random fault before the point duplication for each round of algorithm, collect faulty outputs and analyze them to reveal the secret key.

More sophisticated and effective scenario requires analysis of the output between the faults in i-th and $i+1$-th round to reveal one bit of the secret key. If this stage of the analysis fails, random fault in this round is repeated (to receive more appropriate output). Otherwise, go to the next round. For this scenario, faults must be induced from the last step of the algorithm to the first (and the secret key bits will be revealed in the same order).

Now let's consider how it works for the last step of the algorithm. Let $Q_i = (X_i, Y_i)$ be the value of the intermediate point without any faults. The attacker knows the value of Q_n as a result of non-faulty algorithm. When he modifies X_{n-1} before the last duplication, the device performs the last round with a faulty value $Q'_{n-1} = (X'_{n-1}, Y_{n-1})$. The faulty output point is $Q'_n = Q'_{n-1} + Q'_{n-1}$ if $d_{n-1} = 0$ and $Q'_n = (Q'_{n-1} + Q'_{n-1}) + P$ otherwise. Here '+' means a result of a formal application of the formulas (2.2), (2.3) to Q and P.

Usually, Q'_{n-1} doesn't lie in the original curve. Note that (2.2), (2.3) doesn't use a_6. Consequently, the result of $Q'_{n-1} + Q'_{n-1}$ is a result of the duplication of Q'_{n-1} in the curve with changed a_6 and the same other parameters. There is only one such curve that contains Q'_{n-1}. Hereinafter, we are going to call these curves as *shifted curves*. Note that any point lies in some shifted curve and this curve is unique.

As for operation $Q'_{n-1} + P$, note that the corresponding formulas don't use a_2, a_4, a_6 (exclude the case $Q'_{n-1} = P$). Consequently, the result of it is a of addition of Q'_{n-1} and P in the curve that contains both points and with the same parameters a_1, a_3 and changed a_2, a_4, a_6.

Faulty output Q'_n lies in the same shifted curve. The equation $2R' + d_{n-1}P = Q'_n$ has at least one solution R' for the real value of d_{n-1}. Moreover, the Y-coordinate of this solution equals to the Y-coordinate of the real Q_{n-1}.

For the false value this equation may be unsolvable. But even though there are solutions for this equation it is very unlikely that the corresponding Y coincides with the Y of the real Q_{n-1}.

So, the attacker can use the following way to reveal d_{n-1}:

1. Receive Q_n and Q'_n
2. Find the sets of solutions for equations $2R + \delta P = Q_n$ ($\mathfrak{R}_\delta$) и $2R' + \delta P = Q'_n$ ($\mathfrak{R}'_\delta$) for both $\delta \in \{0, 1\}$
3. Compute the intersections $Y_\delta = Y(\mathfrak{R}'_\delta) \cap Y(\mathfrak{R}'_\delta)$ of sets of Y-coordinates for these solutions
4. If for δ the set Y_δ is empty then real value of $d_{n-1} \neq \delta$
5. Otherwise the value δ is acceptable

Obviously, $\delta = d_{n-1}$ is acceptable always. For $\delta \neq d_{n-1}$ the probability of an accidental collision of Y-coordinates is no greater than $O(1/N)$ where N is the order of the original field $\mathbb{K}$.

If, however, both values of δ are acceptable the adversary can induce a repeated fault at the same round. Usually, the number of these repeated faults is small.

Note that in practical cases the set $\mathfrak{R}_\delta$ has the only element (for each δ) because the real value of Q_{n-1} must lie in the original cyclic group with a known large prime order r. This value may be calculated effectively from Q_n as $Q_{n-1} = ((r+1)/2)(Q_n - \delta P)$.

When some bits are known, the similar analysis may be applied to reveal the next bit. All manipulations with Q_n are invertible if d_i known for $i > k$. As a result, the adversary reveals Q_{k+1} unambiguously. Further he can compute possible candidates for Q_k for both variants of d_k.

However, for Q'_i the situation is different. The problem is to determine the point Q'_i by a known value of $Q'_i + Q'_i$. This point can be determined as a solution of the corresponding algebraic equation. But if there are more than one solution, all of them must be analyzed. As a result, the number of branches of algorithm may increase dramatically.

The exact number of solutions depends on the structure of 2-subgroup of the shifted curve. Accordingly, the full complexity of the key revealing depends on the number of false branches that occurred in the analysis and, consequently, on the structure of 2-subgroups of intermediate shifted curves. According to the well-known result from the theory of elliptic curves ([Sil86]) there are three different cases: this group is trivial, cyclic or the direct sum of two cyclic groups. If the group is trivial the equation has only one equation for each $R = Q'_i + Q'_i$. In other words, it means that half of the point may be calculated unambiguously. If 2-subgroup is cyclic there are two cases: the equation may have two different solutions or nothing. The probabilities of these cases are equal to 1/2. Finally, if 2-subgroup is the direct sum of two cyclic groups there are two cases: the equation has four different solutions or nothing. The probability of the first case is 1/4.

Now we can estimate the order of growth of the number of branches and therefore, full complexity of the algorithm.

The structure of 2-subgroup depends on the number of roots of some polynomial of degree 3 in the field $\mathbb{K}$. This dependency is very simple: trivial subgroup

corresponds to no roots of polynomial in $\mathbb{K}$, cyclic subgroup corresponds to one root, and the direct sum of two cyclic groups corresponds to three different roots. It is an easy combinatorial task to calculate the number of polynomials of these types over known field $\mathbb{K}$.

In fact, for large field $\mathbb{K}$ roughly one third of polynomials are irreducible, half of them have the unique root and one sixth have three different roots. Hence, three types of 2-subgroups have the same distribution.

If the same distribution is correct for shifted curves we can estimate an average number of branches which occurred in the analysis process. Any branch may be true (that contains real values of Q'_i induced by analyzed fault) or false. True branch is unique but any number of false branches is possible.

The rough approximation uses the simple (but not precise) model with independent results of analysis for each steps and branches. In this case each false branch on some step may produce 0, 1, 2 or 4 branches in the next steps. Corresponding probabilities are $p_1 \approx 1/3$ for 1 new branch, $p_2 \approx 1/2 \cdot 1/2$ for 2 branches, $p_4 \approx 1/6 \cdot 1/4$ for 4 branches and $p_0 \approx 1/2 \cdot 1/2 + 1/6 \cdot 3/4 = 3/8$ for no produced branches. All produced branches are false.

For true branches the situation is slightly different. The probability distribution is the same. But in this case the equation certainly has one solution at least (which corresponds with true branch). It means that for cyclic groups there are exactly two solutions and for direct sum there are four. Hence, probabilities are $q_1 \approx 1/3$ for 1 new branch, $q_2 \approx 1/2$ for 2 (1 true and 1 false) and $q_4 \approx 1/6$ for 4 (true and 3 false).

Let $n(i)$ be the average number of false branches in the i-th step. Then

$$n(i+1) \approx n(i)(p_1 + 2p_2 + 4p_4) + (q_2 + 3q_4) = n(i) + 1 \tag{5.1}$$

The results of computer simulation with different curves, basic points and keys conform to this estimation.

However, we must note that basic model with independent rounds isn't quite adequate because it ignores dependencies between rounds. Really, if the revealed key bit for some round equals to 1 the equations for this and next round apply to different shifted curves and their independency is a reasonable hypothesis. But if the key bit is 0 these equations apply to the same curve and their dependency is guaranteed. The solvability of both equations depends on the structure of 2-subgroup of this curve (and on concrete point that is chosen as solution in current round). Similar dependency occurs between more than two consecutive rounds if all the corresponding key bits are zero.

Apparently, it doesn't have an important influence on the number of branches because probability distribution for false branches is very similar. But in practice this fact can lead to significant growth of complexity if the analyzed fault is unlucky and point from true branch in some step lies in bad curve (with large 2-subgroup). That is not so important for false branches because usually many produced sub-branches are unsolvable in the following steps and at that probablity of significant growth decreases in proportion to this growth. But for true branch the solvability is guaranteed and the maximal growth is limited only

by order of 2-subgroup and number of consecutive zero bits in the key. Fortunately, the probability of such bad curves is small and the simplest way to avoid this problem is to repeat a fault when number of false branches is significantly greater than the expected average value.

6 Countermeasures

The simplest way to prevent this attack is the correctness check of the output point. If the calculated output point was modified as a result of a fault, it couldn't lie in the original curve (or, more precisely, the possibility of this event is negligible). In this case, the device could return an error instead of a faulty point. As a result of it an adversary will not receive any information for analysis. This way is very simple, cheap and could be implemented in any device.

Another effective countermeasure is to send to the device output only X-coordinate of the calculated point. In this case, an adversary will have not enough information for successful analysis (but several bits of secret key still could be revealed). But some protocols require both coordinates of a point for further operations and this countermeasure couldn't be applied in this case.

There are many other implementations of the elliptic curve scalar multiplication. Possibly, some of them could be resistant to this type of attacks. However, it's an open issue how to adapt this attack to alternative implementations easily. So, the effectiveness of this countermeasure must be investigated especially in the definite case.

7 Conclusions

We presented a new fault attack on the implementation of the elliptic curve scalar multiplication. Our attack has a small complexity (linear number of faults and polynomial time is required). This attack is applicable when only random number of bits of register may be changed by fault. In several cases this attack may be more applicable or more effective than the previous attacks. The results of the computer simulation demonstrate that the secret key can be revealed with this attack for feasible time (up to a few days) for widespread elliptic curves (for example, NIST-recommended, [NIST99]).

References

[BDL97] *D. Boneh, R. A. DeMillo, and R. J. Lipton.*
On the Importance of Checking Cryptographic Protocols for Faults (extended abstract).
W. Funny, editor, Advances in Cryptology (EUROCRYPT 1997), volume 1233 of Lecture Notes in Computer Science, pages 37-51. Springer-Verlag, 1997.

[BMM00] *I. Biehl, B. Meyer, and V. Müller.*
Differential Fault Attacks on Ellitic Curve Cryptosystems.
M. Bellare, editor, Advances in Cryptology – CRYPTO'2000, volume 1880 of Lecture Notes in Computer Science, pages 131-146. Springer-Verlag, 2000.

[BOS06] *J. Blömer, M. Otto, and J-P. Seifert.*
Sign Change Fault Attacks on Elliptic Curve Cryptosystems.
L. Breveglieri, I. Koren, D. Naccache, and J-P. Seifert, editors, Fault Diagnosis and Tolerance in Cryptography, volume 4236 of Lecture Notes in Computer Science, pages 36-52. Springer-Verlag, 2006.

[BS97] *Eli Biham and Adi Shamir.*
Differential Fault Analysis of Secret Key Cryptosystems.
Advances in Cryptology – CRYPTO'97 (B. S. Kaliski, Jr., ed.), Lecture Notes in Computer Science, volume 1294, pages 513-525. Springer-Verlag, 1997.

[CJ05] *M. Ciet and M. Joye.*
Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults.
Designs, Codes and Cryptography, (36(1)):33-43, 2005.

[FLRV08] *P-A. Fouque, R. Lercier, D. Réal, and F. Valette.*
Fault Attack on Elliptic Curve Montgomery Ladder Implementation.
L. Breveglieri, S. Gueron, I. Koren, D. Naccache, and J-P. Seifert, editors, Fault Diagnosis and Tolerance in Cryptography (FDTC 2008), pages 92-98. IEEE Computer Society, 2008.

[Kob87] *N. Koblitz.*
Elliptic Curve Cryptosystems.
Mathematics of Computation, pages 203-209. 1987, 48.

[Kob92] *N. Koblitz.*
CM-curves with good cryptographic properties.
Advances in Cryptology – CRYPTO '91 (J. Feigenbaum, ed.), Lecture Notes in Computer Science, volume 576, pages 279-287. Springer-Verlag, 1992.

[Mil86] *V. S. Miller.*
Use of Elliptic Curves in Cryptography.
Advances in Cryptology – CRYPTO'85, Lecture Notes in Computer Science, volume 218, pages 417-426. Springer-Verlag, 1986.

[NIST99] *NIST.*
Recommended elliptic curves for federal government use, July 1999.
csrc.nist.gov/encryption.

[Otto04] *M. Otto.*
Fault Attack and Countermeasures.
Dissertation, Facultät für Elektrotechnik, Informatik und Mathematik, Institut für Informatik, Universität Paderborn, 2004.

[Sil86] *J. Silverman.*
The Arithmetics of Elliptic Curves.
Heidelberg etc.: Springer, 1986.