Enhancing Simple Power-Analysis Attacks on Elliptic Curve Cryptosystems*

Elisabeth Oswald

Institute for Applied Information Processing and Communications
Graz University of Technology
Inffeldgasse 16a, A-8010 Graz, Austria
Elisabeth.Oswald@iaik.at

Abstract. Recent applications of lattice attacks against elliptic curve cryptosystems have shown that the protection of ephemeral keys in the ECDSA is of greatest importance. This paper shows how to enhance simple power-analysis attacks on elliptic-curve point-multiplication algorithms by using Markov models. We demonstrate the attack on an addition-subtraction algorithm (fixing the sequence of elliptic-curve operations) which is similar to the one described by Morain et al. in [MO90] and apply the method to the general addition-subtraction method described in ANSI X9.62 [ANS99].

1 Introduction

Elliptic curve cryptosystems (ECC) have been introduced in 1985 by Miller and Koblitz and are widely accepted. Since there are no sub-exponential algorithms known for the elliptic-curve discrete-logarithm problem (ECDLP), the keys can be much smaller in elliptic curve cryptography than in other public-key cryptosystems. Consequently, elliptic-curve cryptography offers significant advantages in many practical aspects. Due to their practical advantages, elliptic curve cryptosystems can be expected to be incorporated in many future cryptographic applications and protocols. The most effective cryptanalytic attacks on implementations of elliptic curve cryptosystems nowadays are the power attacks [KJJ99][1]. They use the power consumption of a device performing an elliptic-curve scalar point-multiplication as a side-channel. Both power-analysis variants, the simple power analysis (SPA) and the differential power analysis (DPA), are effective against unprotected implementations of an elliptic-curve scalar point-multiplication. Due to the importance of elliptic curve cryptosystems, many articles related to power analysis and elliptic curve cryptography have recently been published. We want to mention one article especially. This article [RS01] (which is based on [HGS01]) describes how to compute the secret key of the ECDSA, if a few bits of the ephemeral key for several ECDSA

* The work in this paper was partially done while the author visited COSIC, KU Leuven, Belgium, 2002.

[1] EM attacks appear to become increasingly powerful as well, see for example [QS01]

B.S. Kaliski Jr. et al. (Eds.): CHES 2002, LNCS 2523, pp. 82–97, 2003.

signatures are known. Consequently the protection of ephemeral keys is a very important aspect that cannot be neglected. Therefore the implementation of an elliptic-curve scalar point-multiplication algorithm should be resistant against simple power-analysis attacks even if it is used only for signatures.

The main contribution of this paper is the development of a new and more powerful simple power-analysis attack which is even applicable to elliptic-curve scalar point-multiplication algorithms that do not fix the sequence of elliptic-curve operations. This attack shows that certain attempts to counteract simple power-analysis attacks by only obscuring[2] the ephemeral key, fail.

This paper is organized as follows. Section 2 is dedicated to related work. In section 3, the relationship between Markov models and point-multiplication algorithms is established, and the general idea for the enhanced simple power-analysis attack is presented. Finally, in section 4, we apply this method to a addition-subtraction algorithm and to the scalar point-multiplication algorithm defined in ANSI X9.62 ([ANS99] and IEEE P1363a [IEE99]). We also reason about the applicability of this method to the randomized algorithms as presented in [OA01].

2 Related Work

Finding efficient countermeasures to protect implementations of elliptic-curve scalar point-multiplication against power attacks has proven to be a difficult and challenging task. This is due to the fact that different constraints have to be taken into account for an actual implementation. For example, legal issues such as avoiding patents or implementation constraints. Implementations of elliptic curve cryptosystems usually make use of so called EC-accelerator modules that are very often connected via a slow bus to the rest of the IC. Depending on the specific hardware architecture that is used, an algorithm may or may not lead to an efficient (fast, small, or flexible, etc ...) implementation. Another constraint for countermeasures, is due to the fact that NIST[3] published a set of recommended curves [NIS99] which can be used as 'named curves' in certificates and protocols. These curves have one common property. They all have a cofactor of 2. Because of this specific choice of the cofactor, none of these curves has a Montgomery form. Therefore, countermeasures using the Montgomery form cannot be applied to them. On the other hand, these curves have been given OIDs and the set of elliptic-curve parameters can be replaced by these OIDs in certificates. This is certainly a big advantage since certificate sizes can be significantly reduced. Consequently, countermeasures should be applicable to all recommended curves.

[2] Obscuring means in this context, that there is a more complex relationship between the bits of the ephemeral key and the performed elliptic-curve operations. We will discuss this in more detail in section 3.1

[3] There exists a second set of 'named curves' which has been selected by the SECG [Cer00].

2.1 Previous Results

The basic principles of how to apply power analysis attacks on elliptic curve cryptosystems have been discussed in [Cor99]. To counteract both simple power-analysis attacks and (first order) differential power-analysis attacks there are basically two things that have to be done. Firstly, one has to randomize the expressions (i.e. the coordinates) of calculated points. This can be done by using randomized projective coordinates (DPA countermeasure). Secondly, one has to conceal the ephemeral key. It would be optimal if there would be no statistical relationship between the sequence of elliptic-curve operations and the bits of the ephemeral key (SPA countermeasure). Countermeasures applicable to arbitrary curves fixing the sequence of elliptic-curve operations have been presented by Coron [Cor99], Möller [Möl01] and Izu et al. [IT02]. Countermeasures applicable to arbitrary curves not fixing the sequence of elliptic-curve operations have been presented by Oswald et al. [OA01] and Brier et al. [BJ02]. Countermeasures applicable to special curves fixing the sequence of elliptic-curve operations have been presented by Hasan [Has00] and by Okeya et al. [OS00]. Countermeasures applicable to special curves not fixing the sequence of elliptic-curve operations have been presented by Liardet et al. [LS01] and Joye et al. [JQ01].

In none of the papers it was ever tried to extend the obvious simple power-analysis attack to more general point-multiplication algorithms, i.e. algorithms that also use elliptic-curve point-subtraction or do not fix the sequence of elliptic-curve operations.

3 An Attack Based on a Markov Model for the Elliptic-Curve Scalar Point-Multiplication Algorithm

In a simple power-analysis attack, the adversary is assumed to be able to monitor the power consumption of one scalar point-multiplication, $Q = kP$, where Q and P are points on an elliptic curve E, and $k \in \mathbb{Z}$ is a scalar. The attacker's goal is to learn the key using the information obtained from carefully observing the power trace of a complete scalar point-multiplication. Such a scalar point-multiplication consists of a sequence of point-addition, point-subtraction and point-doubling operations. Each elliptic-curve operation itself consists of a sequence of elementary field-operations. The sequence of elementary field-operations in an elliptic-curve point-addition operation differs from the sequence of elementary field-operations in elliptic-curve point-doubling operation. Every elementary field-operation has its unique power-consumption trace. Hence, the sequence of elementary field-operations that form the point-addition operation has a different power-consumption pattern than the sequence of elementary field-operations that form the point-doubling operation. Because point addition and point subtraction only differ slightly, they can be implemented in such a way that they are indistinguishable for an attacker. This is why we will not distinguish between these two operations in the subsequent sections.

3.1 Elliptic-Curve Scalar Point-Multiplication Algorithms

The simplest way of performing a scalar point-multiplication is the binary algorithm (see table 1 for the bottom-up version).

Table 1. Bottom-up version of the binary algorithm

binalg(P,M,k)
$Q = M$
if $k_0 = 1$ then $P = M$ else $P = 0$
for $i = 1$ to $n - 1$
$\quad Q = Q * Q$
$\quad$ if $(k_i == 1)$ then
$\quad\quad P = P * Q$
return P

For validity and explanation see [Knu98]. In table 1 the operator $*$ denotes the general elliptic-curve point-addition operation. The expression $P * Q$ denotes the point-addition operation (short A) which adds two distinct points P and Q on the elliptic curve, while $P * P$ denotes the point-doubling operation (short D) which adds P to itself.

What makes this algorithm so vulnerable to SPA is the strong relation between the multiplier bits (i.e. the k_i) and the performed operation (i.e. the point addition) in the conditional branch (see table 1). If and only if the i-th bit of k is set, a point-addition operation is performed. Another way of saying this is that the conditional probability that k_i is non-zero equals 1 under the assumption that an elliptic-curve point-addition operation has been observed. An attacker can simply look for two different patterns in the power trace of the scalar point-multiplication algorithm. One pattern corresponds to the point-addition operation and the other pattern corresponds to the point-doubling operation. Since only the point-addition operation can be induced from a non-zero bit, the attacker learns where the non-zero bits in the binary representation of k are. With this information the attacker has to test at most two values (this is because the attacker does not know which of the two observed patterns corresponds to the point addition-operation and which corresponds to the point-doubling operations) to determine the ephemeral key.

The usage of more sophisticated versions of the binary algorithm, like the window-method, signed representations like the non-adjacent form (short NAF), etc. ... (see [Gor98] for an excellent survey), can obscure the private multiplier to a certain extent. The main goal of these sophisticated algorithms is to speed up the scalar point-multiplication. This is usually accomplished by recoding[4]

[4] Well known methods referring to the same basic principle are for example Booth recoding, or Canonical recoding or NAF.

the multiplier k. The recoded form k' leads to fewer operations that have to be performed in the scalar point-multiplication. The arithmetic of elliptic curves makes it possible to use signed representations, i.e. use digits $-1, 0, 1$, because point subtraction is almost the same operation as point addition. The assumption we made in the beginning of this section, namely that we cannot distinguish between point addition and point subtraction, introduces an additional difficulty in the task of the attacker. Observing a point-addition operation gives the attacker less information, since a point addition corresponds to both -1 and 1 in the digit-expansion of k. Having more difficult relations between certain occurrences of operations in the power trace and specific bits (or maybe combination of bits) is what is meant by "obscuring" in this context. In general it can be said that whenever we don't have an "if and only if" relationship between bits and operations, we are not able to mount a simple power-attack in the way we described it for the standard binary algorithm. However, we show in this paper how an ordinary simple power-analysis attack can be enhanced in order to mount an efficient simple power-analysis attack on certain types of these algorithms.

3.2 The Attacker's Task

The attacker has the ability to observe a sequence of elliptic curve operations, thus, the attacker's aim is to calculate and exploit the probabilities of certain sequences of bits given an observed sequence of elliptic curve operations.

Using the information of such conditional probabilities, the key-space that has to be searched to find the correct ephemeral key, can be significantly reduced. This is because certain combinations of patterns in the power trace and certain combination of digits are less likely than the others (or even not possible at all). The attacker's task can be stated in a more formal way.

Let X be a random variable that denotes a sequence of elliptic-curve operations and $|X|$ the length of X (i.e. the number of elliptic-curve operations in this sequence). For example, X="DDD" (i.e. the realization of the random variable X consists of three consecutive elliptic-curve point-double operations) thus $|X| = 3$, or X="DAD" (i.e. the realization of the random variable X consists of an elliptic-curve point-double operation, an elliptic-curve point-addition operation and an elliptic-curve point-double operation) thus $|X| = 3$.

Let Y be a random variable that denotes a sequence of digits in the digit representation of k and $|Y|$ the length of Y (i.e. the number of digits). For example Y = "000" (i.e. the realization of the random variable Y consists of three consecutive zeros) thus $|Y| = 3$, or Y = "01" (i.e. the realization of the random variable Y consists of a zero and an one digit) thus $|Y| = 2$.

Then the attackers goal is to calculate and exploit the conditional probability

$$P(Y = y|X = x) = \frac{P(Y = y \cap X = x)}{P(X = x)} \tag{1}$$

for many different realizations x of X and y and Y. Equation 1 is the mathematical definition for the conditional probability.

It is an important observation that the calculation of the right hand side of (1) requires the knowledge of the probability to be in a specific state of the point-multiplication algorithm (the terminology used here will be explained in the next section). This is because in order to calculate the probabilities $P(X = x)$, one has to calculate the sum of the probabilities of all possible sequences of digits that lead to the pattern x. Since such a sequence can basically start from any state of the algorithm, the probabilities are dependent on the probability of the starting-state. These probabilities can be calculated by using Markov models which we are going to introduce in the following section.

3.3 Markov Models

The general assumption for the rest of this paper is that the multiplier bits k_i, are independently drawn and identically distributed. We can see a point-multiplication algorithm as a Markov process (see for example [GS92]). A Markov process in general can be used to analyze random, but dependent events. In a Markov process, the next state (or event) is only dependent on the present state but is independent of the way in which the present state arose from the states before. This is often referred to as "memoryless" process. The transitions between the states are determined by a random variable and occur with certain probabilities, that are either known or have to be estimated. A common way to work with Markov processes is to visualize them in so called transition graphs. For example, figure 1 shows the transition graph for the binary algorithm as presented in table 1.

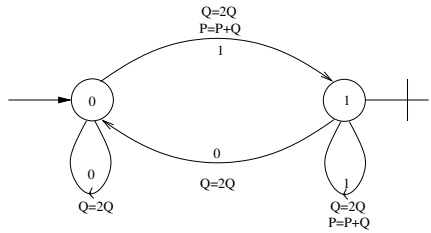

Fig. 1. Transition-graph of the Binary Algorithm

According to the description given for table 1, P and Q denote elliptic-curve points. Circles represent states (0 and 1 in this case) and the arrows between the circles represent transition between the states. Output paths are marked by an additional bar. In the binary algorithm, transitions are triggered by the bits k_i. Consequently, in figure 1 the values next to the arrows, correspond to the possible values of k_i (i.e. 0 and 1). The expressions $Q = 2Q$ (point double) and $P = P + Q$ (point addition) indicate what operation is triggered by the value of k_i.

This graph or Markov process, respectively, has two important properties. The first property is that the graph is *irreducible* in the sense that all states can be reached from all other states with a finite number of steps. The second important property is that the graph is *aperiodic*, in the sense that all states are aperiodic. A state is aperiodic, if the period (that is the greatest common divisor of the set of times a chain has a positive probability of returning to the same state) is equal to 1. These two properties are conditions for the main theorem of Markov theory. This theorem states basically that for Markov processes having the properties of being aperiodic and irreducible a *steady state* always exists[5]. The row-vector $\pi = (\pi_1, \ldots, \pi_n)$ representing the steady state has the following two properties:

$$\sum_{i=1}^{n} \pi_i = 1, \tag{2}$$

$$\pi T = \pi. \tag{3}$$

The variable T in the second equation is a matrix (subsequently referred to as *transition matrix*) containing the transition probabilities. (3) is the formal way of saying that the distribution has become steady. (2) makes clear that we are actually dealing with a probability distribution for the states (since all the individual probabilities sum up to 1). The entries of the row-vector π are simply the probabilities of the states the algorithm can be in. What is important for the attack presented in this paper is that π depends solely on the transition matrix T and can be obtained by calculating the eigenvectors (with the associated eigenvalues) of the transition matrix (this follows directly from (3)). The transition matrix itself can be obtained in a straightforward way. The transition probabilities associated with the transition variables can be written in the transition matrix

$$T = \begin{pmatrix} P(s_{i+1} = 0 | s_i = 0) & P(s_{i+1} = 0 | s_i = 1) \\ P(s_{i+1} = 1 | s_i = 0) & P(s_{i+1} = 1 | s_i = 1) \end{pmatrix} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}.$$

T contains the probabilities to get from one state to another state. The random variable s_i denotes the state the algorithm is in. For example, the matrix entry in the first row and the second column is the probability to get from state 1 to state 0. Calculating $\pi = (1/2, 1/2)$ leads to the probabilities for being in state 0 and state 1, respectively.

Additionally, we can deduce the number of elliptic-curve operations that need to be executed. We know that every transition requires the calculation of a point-doubling operation, but only a transition leading from state 0 to state 1 or a transition leading from state 1 to state 1 requires the computation of a point-addition operation. Putting this all together and assuming that n denotes

[5] There are several more expressions for the term "steady state". Amongst others, the most common terms seem to be "stationary distribution" and "invariant distribution".

the bit-length of k, we get n point-doubling operations and $n/2$ point-addition operations. So, $3n/2$ elliptic-curve operations have to be performed.

In this section, we have established a relationship between elliptic-curve scalar point-multiplication algorithms and Markov processes. The most important observation was that the main theorem for an important class of Markov processes, namely the irreducible and aperiodic Markov processes, also gives a solution to the problem of calculating the conditional probability defined in (1). Furthermore we have shown how to calculate the number of elliptic-curve operations that need to be performed in a point-multiplication algorithm, with the aid of Markov models.

4 Results

In order to demonstrate that the observations presented in the previous section lead to an effective attack, we apply the technique on well known point-multiplication algorithms. The first algorithm we attack was introduced in the article of Morain et al. [MO90] and belongs to the class of addition-subtraction algorithms. We attack the modified version as given in [OA01]. The second algorithm we discuss is the *addition-subtraction method* or *NAF-method* which is used in important standards such as the ANSI X9.62 and the Annex A of IEEE 1363.

4.1 Analysis of a Double-Add and Subtract Algorithm

We demonstrate how this method can be used to construct a known-ciphertext attack on the example of a modification ([OA01]) of the first algorithm given by Morain et al. [MO90].

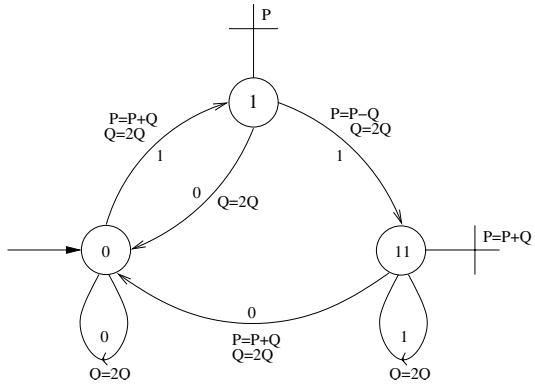

Fig. 2. Transition-graph of Double-Add and Subtract as proposed in [OA01]

The idea of this point-multiplication algorithm is basically to replace a block of at least two consecutive 1's in the binary representation of the multiplier k, by a block of 0's and a $-1 : 1^a \mapsto 1\,0^{a-1}-1$. The original proposal is modified in such a way that for every transition a point-doubling operation or a point-doubling and a point-addition operation has to be performed. Mounting a standard simple power-attack is not possible since there is no "if and only if" relation between the bits and the elliptic curve operations. For both, zero and non-zero bits, elliptic-curve point-addition and elliptic-curve point-doubling operations can occur.

The transition graph of the finite state machine in figure 2 (the notation used here is the same as used before in figure 1) visualizes this algorithm. From this graph one can easily deduce the transition matrix

$$T = \begin{pmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 0 & 0 \\ 0 & 0.5 & 0.5 \end{pmatrix}$$

and therefrom the steady-state vector which is $(1/2, 1/4, 1/4)$. With this information the conditional probabilities for many realizations of X and Y can be calculated. With a modest computational effort, one can do this up to $|Y| = 16$. The results we obtained show how poor the secret key is obscured. We found out that for a chosen X and for fixed and small $|Y|$, only **three** bit-patterns are possible, or there is no bit-pattern at all. We derived this result from our computer program, which checked all conditional probabilities up to $|Y| = 12$. For example, table 2 shows some of the results for small values of $|Y|$.

Table 2. Non-zero conditional probabilities. In this table we use an abbreviated notation, i.e. we write $p(000|DDD)$ instead of $p(Y = 000|X = DDD)$. We use the LSB first representation.

$p(000\|DDD) = 1/2$	$p(01\|DAD) = 1/2$	$p(11\|ADAD) = 1/2$
$p(100\|DDD) = 1/4$	$p(10\|DAD) = 1/4$	$p(10\|ADAD) = 1/4$
$p(111\|DDD) = 1/4$	$p(11\|DAD) = 1/4$	$p(01\|ADAD) = 1/4$
$p(001\|DDAD) = 1/2$	$p(000\|ADDD) = 1/4$	$p(110\|ADADAD) = 1/2$
$p(101\|DDAD) = 1/4$	$p(100\|ADDD) = 1/2$	$p(101\|ADADAD) = 1/4$
$p(110\|DDAD) = 1/4$	$p(111\|ADDD) = 1/4$	$p(011\|ADADAD) = 1/4$

The probabilities in table 2 can be derived with the help of the transition matrix T and the Markov model for the point-multiplication algorithm. We illustrate how to derive them on a simple example.

Example 1. Assume we want to calculate the probability that two consecutive elliptic-curve point-doubling operations occur (under the assumption that we only look at two transitions, i.e. $|Y| = 2$) when performing the scalar point-multiplication algorithm depicted in figure 2. Table 3 lists all possible transitions between two states of algorithm 2 in the leftmost column. The column in

the middle lists the to the sequence of transitions corresponding elliptic-curve operations. The rightmost column indicates the occurrence of two consecutive point-doubling operations.

Table 3. The calculation of the probability $p(X = DD)$ and $p(Y = 00|X = DD)$ can be done with this table. The leftmost part of the table shows the transitions, the column in the middle shows the corresponding elliptic-curve operations and the rightmost column indicates the occurrences of $X = DD$.

$0 \to 0 \to 1$	DAD	
$0 \to 0 \to 0$	DD	$\Leftarrow$
$0 \to 1 \to 0$	ADD	$\Leftarrow$
$0 \to 1 \to 11$	ADAD	
$1 \to 0 \to 0$	DD	$\Leftarrow$
$1 \to 0 \to 1$	DAD	
$1 \to 11 \to 11$	ADD	$\Leftarrow$
$1 \to 11 \to 0$	ADAD	
$11 \to 11 \to 11$	DD	$\Leftarrow$
$11 \to 11 \to 0$	DAD	
$11 \to 0 \to 0$	ADD	$\Leftarrow$
$11 \to 0 \to 1$	ADAD	

For each row of table 3 the corresponding probability can be calculated. This can be done by using the steady state vector and the probabilities which we already derived for the transition matrix. For example, the first row defines a sequence of transitions that starts in state 0, then stays in state zero and ends up in state 1. We know that the probability to be in state 0 is 1/2 and the probability to go from state 0 to state 0 or state 1 is 1/2. Thus, the probability that row one occurs is $(\frac{1}{2})^3$. To calculate $p(X = DD)$ one has to calculate the sum of the probabilities of the six rows which are marked with an arrow. To calculate the conditional probability $p(Y = 00|X = DD)$ one has to calculate the numerator of equation 1. This numerator can be calculated by summing up the probabilities of those three rows of the six marked rows that assume two consecutive zero bits (these are row two, row five and row eleven).

The Attack. The concrete attack works as follows.

1. **Precomputation phase:** Find the Markov model, i.e. the transition matrix and the steady state vector, for the given point-multiplication algorithm. Calculate the conditional probabilities for all combinations of X and Y up to a suitable $|Y|$.
2. *Data collection phase:* Deduce from the power trace of an elliptic-curve scalar point-multiplication operation the sequence of point additions and point doublings. This stage is in fact the same as in an ordinary simple power-analysis attack.

3. **Data analysis phase:** Split this sequence into a number of sub-sequences, whereby each sub-sequence must be chosen in a way that it can be produced by a well defined sequence of digits. The number of digits in the sequence have to sum up to the total number of digits of k to ensure a valid partitioning in sub-sequences. *Remark:* After this step one knows for each sub-sequence the number of digits that produced this subsequence. Of course in general, there are several valid possibilities for a such a partitioning, but it is only important to choose and fix one.
4. **Key testing phase:** Check all combinations of bit-patterns that have a non-zero probability to occur, with the known pair of plain- and ciphertext. This will lead finally to the secret key. *Remark:* One should check the combinations of bit-patterns with the highest probabilities first.

We now illustrate the attack on a toy example.

Example 2. In this example, we use the scalar point-multiplication algorithm depicted in figure 2 which we discussed on beforehand. The scalar k which serves as ephemeral key in this example is 560623. The first row shows the sequence of observed point-doubling and point-addition operations. In the second row the same sequence is split into sub-sequences that contain patterns for which we already calculated the conditional probabilities (see table 2). In the third, fourth and fifth rows the possible bit-patterns are listed according to their conditional probabilities. From all possible bit-pattern combinations the attacker can determine the correct one with the aid of the known plaintext-ciphertext pair.

Table 4. Example : $k = 11110111101100010001$, LSB first representation

ADADDDADADADDDADADADADDDADDDDAD							
ADAD	DDAD	ADAD	DDAD	ADAD	ADDD	ADDD	DAD
11	001	**11**	001	**11**	100	**100**	**01**
10	101	10	101	10	**000**	000	10
01	**110**	01	**110**	01	111	111	11

Effectiveness. We denote the ephemeral secret key with k, it's bit-length by n and the length of the sub-sequences by l. Then, in the worst case where we only take the non-zero conditional probabilities into account but not their actual values, we have to test $3^{3n/2l}$ keys on average (this is because $3n/2$ elliptic-curve operations are calculated on average in this point-multiplication algorithm). If we take into account, that one of the three non-zero conditional probabilities is always $1/2$, we deduce that we only have to test $2^{3n/2l}$ keys on average. In the case of $n = 163$ and $l = 16$ (the size of n is chosen according to the smallest recommend curve by NIST) we can deduce that we only have to test $2^{15.28}$ keys on average.

4.2 Application to the NAF-Method

This algorithm calculates the NAF of the multiplier k. The NAF of k has the property that no two consecutive digits in the expansion of k are non-zero. As a consequence, the average number of non-zero digits is reduced and fewer elliptic-curve operations have to be performed.

Table 5. The NAF algorithm

NAF-mult(P,M,k)
$Q = P$
Set $h = 3k$. Let h_l denote the most significant bit of h.
for $i = l - 1$ down to 1
$Q = Q * Q$
if $(h_i == 1)$ and $(k_i = 0)$ then $Q = Q + P$.
if $(h_i == 0)$ and $(k_i = 1)$ then $Q = Q - P$.
return Q

Table 5 gives a brief description of the NAF algorithm. The finite-state machine describing this algorithm is depicted in figure 3.

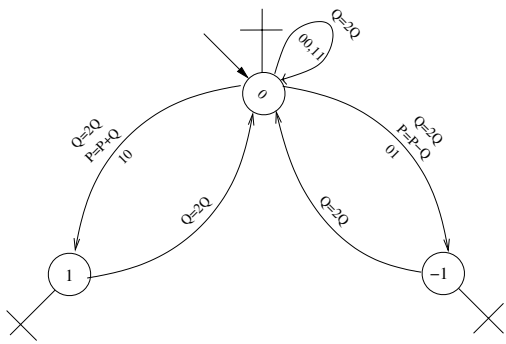

Fig. 3. Transition graph of the NAF algorithm. The label "00","01", "10" and "11" correspond to the values of h_i and k_i in table 5.

Like in the previous section, we can derive the transition matrix from this transition graph quite easily. The NAF property (i.e. that there are no consecutive non-zero digits allowed) is clearly visible in both the transition graph and the transition matrix. The steady-state vector for the transition matrix

$$T = \begin{pmatrix} 0.50\ 1\ 1 \\ 0.25\ 0\ 0 \\ 0.25\ 0\ 0 \end{pmatrix}$$

is $(4/6, 1/6, 1/6)$.

We derived once again a well known result about the distribution of zero and non-zero digits in the NAF-representation, which is, that about 2/3 of all NAF-digits are zeros. We also know immediately from figure 3 that we have an almost perfect and known statistical relationship between the NAF-digits and the elliptic-curve operations. If and only if a non-zero NAF-digit occurs, an elliptic-curve point-addition or an elliptic-curve point-subtraction, has to be performed.

This means, that we not need to calculate conditional probabilities to mount an attack. An attack can be mounted in the very straightforward way by just searching the power trace for occurrences of point-addition patterns. Each such pattern can be produced by either a point-addition or a point-subtraction operation. Since we know that about 1/3 of the NAF-digits are non-zero on average, we know that we would have to test about $2^{n/3}$ keys on average (n denotes the number of digits of the ephemeral key k).

4.3 A General Comment on the Key Testing Phase

The speed with which we can test all the possible ephemeral keys is also certainly an important issue for the whole attack. The scalar point-multiplications which we have to perform in the key testing phase can be done much faster than scalar point-multiplications with random points. This is because the point which is multiplied with a scalar is fixed, and due to elliptic curve protocols, almost never changed. This means that one can use window-methods combined with the NAF-method, for example, to speedup the key testing phase. We already know that because of the NAF property, not all of the bit-patterns (and digit-patterns resp.) are possible. In fact, when using a window-method for the point-multiplication in the key testing phase, one can use rather large windows, since not all, but only a very few points have to be precomputed. For example, for a window with length ten only 144 points need to be calculated. An analysis of window NAF-methods can be found for example in [BHLM01]. This means that in an EC-accelerator module which is designed especially for the key testing phase, one could possibly use such methods.

We now give some rough estimates for testing the number of possible keys if the NAF-method was used on a 163-bit curve. Our analysis in section 4.2 showed that we would have to test about $2^{54,3}$ possible ephemeral keys to determine the correct ephemeral key. Based on the performance data given in [OP00] it turns out that a single device needs appr. 2^{17} years to test all $2^{54,3}$ possible ephemeral keys. Modifying the architecture of [OP00] in such a way that it heavily makes use of precomputed points and thus achieves a further speed-up should be possible. Considering all this, we are concerned about the long-term security margin of the NAF-method in the case of a 163−bit curve.

4.4 A Note on the Application to Randomized Algorithms

In [OA01] the usage of randomized addition-subtraction chains was proposed as a countermeasure against simple and differential power-analysis attacks. We analyzed these randomized algorithms as well. It is clear that the attack as it is described in this paper can directly be applied on the randomized algorithms as well. One can derive the transition matrix for the randomized algorithms in exactly the the same way as we demonstrated it in this paper. The steady state can then be derived from the transition matrix and therefrom the conditional probabilities can be calculated. Due to the limitations of the number of pages for this paper we cannot present a full analysis, but in our investigations it turned out that the Markov method on the randomized algorithms is not better than on the NAF-method. Thus, we consider the randomized algorithms as more secure than the NAF-method with respect to the attack presented in this paper.

5 Conclusion

We presented an enhanced simple power-analysis attack on elliptic-curve point-multiplication algorithms. The method basically uses the information about the conditional probabilities of observed sequences of elliptic-curve operations, and bit-patterns that form the secret key. The method can be considered as an enhancement because it works even in cases where the standard simple-power attack fails. The approach is a general method in the sense that it can be used to attack arbitrary elliptic-curve point-multiplication algorithms that do not even necessarily fix the sequence of instructions. We demonstrated that it can be effectively applied on the example of a modification of a point-multiplication algorithm originally proposed by Morain and Olivos where the standard simple power-analysis attack fails. We also analyzed the security of the NAF-method which is often used in standards. We pointed out that this method applies to randomized algorithms as well. Based on the analysis we conjecture that methods that only use three digits for encoding the ephemeral key are susceptible to this attack. To summarize the results, the method is new and more efficient than the standard simple power-analysis attack and poses a serious threat against certain algorithms that only try to obscure the ephemeral key. Our analysis also indicates that the long-term security margin of the NAF-method in the case of a 163-bit curve is not too large.

Acknowledgments. We'd like to thank Mathijs Coster for his comments on Markov chains. Additionally, we'd like to thank the anonymous reviewers for their helpful comments and Vincent Rijmen for proofreading the final version of this paper.

References

[ANS99] ANSI. ANSI X9.62 Public Key Cryptography for the Financal Services Industry:The Elliptic Curve Digital Signatur Algorithm (ECDSA), 1999.

[BHLM01] Michael Brown, Darrel Hankerson, Julio Lopez, and Alfred Menezes. Software Implementation of the NIST Elliptic Curves Over Prime Fields. In *Progress in Cryptology – CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 250–265. Springer, 2001.

[BJ02] E. Brier and M. Joye. Weierstrass Elliptic Curves and Side-Channel Attacks. In *Public Key Cryptography*, volume 2274 of *Lecture Notes in Computer Science*, page 335 ff. Springer, 2002.

[Cer00] Certicom Research. Standards For Efficient Cryptography – SECG 2: Recommended Elliptic Curve Cryptography Domain Parameters. Version 1.0, 2000. Available from http://www.secg.org/.

[Cor99] J.-S. Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In *Workshop on Cryptographic Hardware and Embedded Systems – CHES 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999.

[Gor98] D. M. Gordon. A survey of fast exponentiation methods. *Journal of Algorithms*, (27): pp. 129–146, 1998.

[GS92] Geoffrey Grimmett and David Stirzaker. *Probability and Random Processes*. Oxford University Press, 2nd edition, 1992. ISBN: 0198536658.

[Has00] M. A. Hasan. Power Analysis Attacks and Algorithmic Approaches to Their Countermeasures for Koblitz Cryptosystems. In *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 93–108. Springer, 2000.

[HGS01] N. Howgrave-Graham and N. Smart. Lattice attacks on digital signature schemes. *Designs, Codes and Cryptography*, (23):283–290, August 2001.

[IEE99] IEEE. Standard Specifications for Public Key Cryptography, Annex A, D13, 1999.

[IT02] T. Izu and T. Takagi. Fast Parallel Elliptic Curve Multiplications Resistant to Side Channel Attacks. In *to appear in International Workshop on the Practice and Theory of Public Key Cryptography (PKC2002)*, Lecture Notes in Computer Science (LNCS). Springer, 2002.

[JQ01] M. Joye and J.-J. Quisquater. Hessian Elliptic Curves and Side-Channel Attacks. In *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 402–410, 2001.

[KJJ99] P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Advances in Cryptology-CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Knu98] D. E. Knuth. *The Art of Computer Programming. Seminumerical Algorithms*, volume 2. Addison-Wesley, 3rd edition, 1998.

[LS01] P.-Y. Liardet and N.P. Smart. Preventing SPA/DPA in ECC Systems Using the Jacobi Form. In *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 391–401, 2001.

[MO90] F. Morain and J. Olivos. Speeding up the computation on an elliptic curve using addition-subtraction chains. *Inform. Theory Appl.*, (24):531–543, 1990.

[Möl01] B. Möller. Securing Elliptic Curve Point Multiplication against Side-Channel Attacks. In *Information Security – 4th International Conference, ISC 2001*, volume 2200 of *Lecture Notes in Computer Science (LNCS)*, page 324 ff. Springer, 2001.

[NIS99] NIST. Recommended Elliptic Curves For Federal Government Use, 1999.

[OA01] E. Oswald and M. Aigner. Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks. In *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science (LNCS)*, pages 39–50. Springer, 2001.

[OP00] G. Orlando and Ch. Paar. A high performance reconfigurable elliptic curve processor for gf(2^m). In *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 41–56. Springer, 2000.

[OS00] K. Okeya and K. Sakurai. Power Analysis Breaks Elliptic Curve Cryptosystems even Secure against the Timing Attack. In *Progress in Cryptology – INDOCRYPT 2000*, volume 1977 of *Lecture Notes in Computer Science (LNCS)*, pages 178–190. Springer, 2000.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *Smart Card Programming and Security, E-smart 2001*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[RS01] T. Römer and J.-P. Seifert. Information leakage attacks against Smart Card implementations of the Elliptic Curve Digital Signature Algorithm. In *Smart Card Programming and Security (E-Smart 2001)*, volume 2104 of *Lecture Notes in Computer Science (LNCS)*, page 211 ff. Springer, 2001.