Power Analysis Attacks on ECC Randomized Automata

Fan Zhang and Zhijie Jerry Shi
Department of Computer Science and Engineering, University of Connecticut
Email: {fan.zhang, zshi}@engr.uconn.edu

Abstract

Power analysis can exploit the instantaneous power consumptions of Elliptic curve cryptography (ECC) devices and retrieve secret keys. Many countermeasures have been proposed to make ECC implementations secure. One of the approaches is the randomized algorithms proposed by Oswald et al., which combine two scalar point multiplication algorithms and use random variables to decide which algorithm to follow at different stages of the computation. In this paper, we describe a power analysis attack that can break randomized automata proposed by Oswald et al. effectively, even with a small number of power traces.

1. Introduction

The elliptic curve cryptography (ECC) [1][2] has been widely used, especially in resource-constrained environments, because it can provide equivalent security strengths with shorter keys than traditional public-key algorithms. However, ECC implementations are vulnerable to power analysis attacks, which exploit the power consumptions of cryptographic devices to retrieve secret keys.

Some countermeasures have been proposed for ECC. One of the countermeasures is the randomized algorithm proposed by Oswald et al. in [3]. The randomized algorithms seem secure against traditional power analyses. However, in this paper, we describe a method that can break the randomized algorithms with a small number of power traces.

The rest of the paper is organized as follows. Section 2 briefly describes the randomized algorithm proposed by Oswald et al. Section 3 describes our attacks and Section 4 concludes the paper.

2. Randomized Automata for ECC

The major operation in ECC is the scalar point multiplication dP, where d is a scalar and P is a point. The operation can be performed by many algorithms. The randomized algorithms proposed by Oswald et al. combine two scalar point multiplication algorithms [4] and randomly choose one of them to follow during the computation. Oswald et al. have proposed two random algorithms, which can be represented with two automata. Fig. 1 shows the second one, which will be referred to as Randomized Automaton 2 (RA2) [4]. RA2 is more difficult to break than the first randomized algorithm as it has more states. When performing a scalar point multiplication with RA2, we start from State 0 and scan the bits in the scalar d from the least significant bit. For each bit, we follow a transition while performing proper operations, as specified in the automaton. In RA2, some transitions are also decided by random variables e, which are illustrated with dashed lines in Fig. 1. Because of the randomness, RA2 follows different paths and thus generates different power traces even if the input is the same. Although Okeya et al. have proposed a method to attack the first randomized algorithm [5], their method has not been extended to attack RA2. Our prior paper only showed the attacks on the first algorithm as well [6].

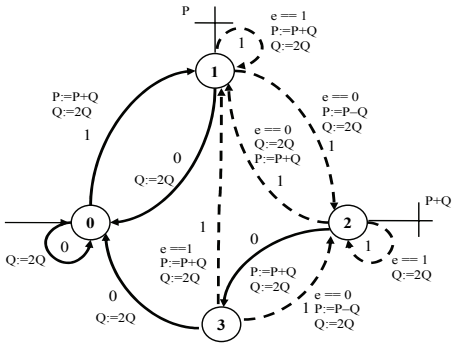

Fig. 1: Randomized Automaton 2 [4]

3. Attacks on Randomized Automata

For convenience, we will denote the doubling operation $2Q$ as D and the addition or subtraction operation ($P+Q$ or $P–Q$) as A. The bits in scalar d are referred to as *input bits*. We assume additions and subtractions are indistinguishable for attackers.

To launch the attack, we first collect a set of power traces by performing the scalar point multiplication N times with the same d. Although all the power traces start from State 0, they visit different states because of the random variables. Looking at all the power traces, we can summarize the state of automaton, after each bit in d is processed, in four cases. 1) The automaton is in State 0 for all power traces. 2) The automaton is in State 1 for all power traces. 3) The automaton can be in either State 1 or 2. And 4) the automation can be in either State 0 or 3. In our attack, we use four *attack states*, AS0, AS1, AS1/2 and AS0/3, to cover the four cases. Since all power traces start with State 0, our attack always starts with AS0. Then, for every attack state, we do the following steps until all the bits in d are recovered.

(1) **Detect input bits**. We look at *all* power traces and detect one or several input bits a time for the current attack state with *unique sequences*. A unique sequence is a power trace that can be generated by a specific input string in the attack state. Fig. 2 shows the algorithm we used to find unique sequence. The input of the algorithm is a set of 3-tuples (s, k, p), which denotes that p is a power trace that can be generated by input string k from state s. For an attack state, we need to consider all automata states it covers and all possible input strings in those states. The function substr(x, y) on line 4 returns TRUE if y is a substring of x and x starts with y. With the algorithm, we can find one or several unique sequences for every input string in every attack state of RA2. If a unique sequence is observed in attacks, its corresponding input string can be recovered.

```
    Algorithm: FIND-UNIQUE-SEQUENCE
    Input :   R={r1, r2, …, rn} where ri=(si, ki, pi)
    Output:   A set U that contains unique sequences
1   U = {};
2   for (i = 1; i = n; i ++)
3       for (j = 1;j = n; j ++)
4           if (ki != kj and (substr(pi, pj) or substr(pj, pi)))
5               Continue with next i;
6       U = U ∪ ri ;
```

Fig. 2: Algorithm to find unique sequences

(2) **Commit detected bits**. A bit is *committed* if the bit is appended to the recovered input string and its corresponding power outputs are removed from all the power traces. We do not always commit all the bits detected in Step 1. We may withhold one or two detected bits. So in the following attack state, we can consider only the strings that start with the withheld bits. For example, in AS1, if the detected string is 111, we only commit the first two 1's and withhold the last 1. After we get into the next state, say, AS1/2, we consider only the input strings starting with 1.

(3) **Go to next attack state**. We decide the next attack state according to the committed bits. After we enter the next attack state, we repeat the steps and detect more input bits with shortened power traces until all input bits are recovered.

To summarize, we start from attack state AS0 and detect input bits in the current attack state. Then we commit some of the detected bits, i.e., we append them to the recovered input string and remove their corresponding outputs from power traces. We then go to the next attack state and repeat the process with shortened power traces until all input bits are recovered.

Since unique sequences do not always appear, it is possible that our attacks retrieve a wrong key. The more power traces we have, the more likely we retrieve a correct key. With N power traces, the maximum error rate can be calculated as:

$$\frac{1}{4}\times\left(\frac{1}{4}\right)^N+\frac{1}{4}\times\left(\frac{5}{8}\right)^N+\frac{1}{4}\times\left(\frac{7}{8}\right)^N$$

When N = 100, the error rate is about 2^{-21}. If N is 200, the maximum error is about 2^{-40}. So our attack is very effective on RA2.

4. Conclusions and Discussions

In this paper, we presented an attack on randomized ECC algorithms and showed how it can break Randomized Automaton 2 proposed in [4]. Our attack is based on the fact that in every state of the randomized algorithm, an input string generates unique output sequences. Although these unique sequences may not appear in every execution, the attackers can exploit them by collecting many power traces. The attack does not require a large number of power traces to achieve a very high success rate.

5. References

[1] V. S. Miller, "Use of elliptic curves in cryptography," *CRYPTO '85*, pp. 417-426, August 1985.

[2] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203-209, January 1987.

[3] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *CRYPTO'99*, pp. 388-397, August 1999.

[4] E. Oswald and M. Aigner, "Randomized addition-subtraction chains as a countermeasure against power attacks," *Proceedings of CHES 2001*, pp. 39-50, May 2001.

[5] K. Okeya, and K. Sakurai, "On insecurity of the side channel attack countermeasure using addition-subtraction chains under distinguishability between addition and doubling," *Proceedings of ACISP 2002*, pp. 420-435, July 2002.

[6] Z. J. Shi and F. Zhang, "New attacks on randomized ECC algorithms," *Proceedings of EITC 2006*, August 2006.