

New Attacks on Randomized ECC Algorithms

Zhijie Jerry Shi and Fan Zhang

Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269, USA

Email: {zshi, fan.zhang}@enr.uconn.edu

Abstract- Elliptic curve cryptography (ECC) has attracted a lot of attention because it can provide similar levels of security with much shorter keys than the arithmetic of multiple-precision integers in finite fields, which has been widely used in many public-key and key-exchange algorithms. Small key sizes are especially important to resource constrained devices as shorter keys require less storage space and consume less power to transmit and compute. However, ECC algorithms are vulnerable to power analysis attacks, which exploit the instantaneous power consumptions of computing devices to retrieve secret data. Many countermeasures have been proposed to make ECC implementations secure against power analysis. One of the approaches is randomized algorithms that generate different power traces even if the input of the algorithm is the same. For example, the randomized scalar point multiplication algorithm proposed by Oswald et al. combines two algorithms and uses random variables to decide which algorithm to follow at different stages of the execution. The randomized algorithm can thwart traditional power analysis attacks. However, in this paper, we propose an effective attack on the randomized algorithm. Our attack does not require a large number of power traces and has a very high success rate.

I. INTRODUCTION

The arithmetic of large integer in finite fields has been adopted in many public-key algorithms, such as RSA and ElGamal, and key-exchange algorithms, such as Diffie-Hellman. In the middle 1980s, Miller and Koblitz proposed the elliptic curve cryptography (ECC) [1][2] as an alternative to large integer arithmetic. ECC has attracted a lot of attention because it can provide equivalent security strengths with shorter keys than traditional large integer operations. Smaller key sizes have many advantages including higher performances, smaller storage space, and lower energy consumptions. These advantages are especially important in resource-constrained environments, such as smart cards, cellular phones, and personal digital assistants (PDAs), where processing power, storage space, communication bandwidth, and power supply are limited [3].

Power analysis is a type of side channel attacks that exploit the power consumptions of cryptographic devices to reveal the secret data used in cryptographic computations. Mobile devices are especially vulnerable to this type of attacks as adversaries can easily gain the physical access to these devices. There are several types of power analyses, such as Simple Power Analysis (SPA) and Differential Power Analysis (DPA). SPA exploits the correlation between the power outputs and the operations. By observing the power trace of cryptographic computations, adversaries can learn what operations are performed and then figure out part or all of the secret data.

SPA can be launched even with only one power trace. Unlike SPA, DPA uses statistical methods to test whether a specific value is generated during cryptographic computations and then deduces the secrets. Typically, DPA needs a large number of power traces. Both of SPA and DPA can be used to attack ECC implementations.

Some countermeasures have been proposed to make ECC systems secure against power analysis. One of the countermeasures is the randomized algorithm proposed by Oswald et al. in [4]. The randomized algorithm seems secure against traditional power analyses. However, we found it is vulnerable to a new type of SPA attacks.

The rest of the paper is organized as follows. Section II describes the basic operations of ECC and Section III describes power analysis attacks and some countermeasures. In Section IV, we discuss our attacks on the randomized algorithm. And Section V concludes the paper.

II. ELLIPTIC CURVE CRYPTOGRAPHY

An elliptic curve is a set of points (x, y) specified by a bivariate cubic equation defined over a field K [5]:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \quad (1)$$

where $a_i, x,$ and $y \in K$. If K is $\text{GF}(2^n)$, equation (1) can be reduced to:

$$y^2 + xy = x^3 + ax^2 + b. \quad (2)$$

where $a, b, x,$ and $y \in \text{GF}(2^n)$.

The set of points on an elliptic curve, together with a special point O called the point at infinity and an addition operation, form an abelian group [6]. For elliptic curves defined over $\text{GF}(2^n)$, the addition operation of points is defined as follows. If a point $P = (x_1, y_1) \neq O$, $-P = (x_1, x_1 + y_1)$. Given another point $Q = (x_2, y_2) \neq O$ and $Q \neq -P$, the sum $(x_3, y_3) = P + Q$ can be computed as:

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a. \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1. \end{aligned} \quad (3)$$

where

$$\lambda = \begin{cases} \frac{y_2 + y_1}{x_2 + x_1}, & \text{if } P \neq Q \\ \frac{y_1}{x_1} + x_1, & \text{if } P = Q. \end{cases}$$

The multiplication of an integer d and a point P is defined as adding d copies of P together.

$$Q = dP = \underbrace{P + \dots + P}_d \text{ times}$$

III. POWER ANALYSIS AND COUNTERMEASURES

In this section, we briefly describe the power analysis of ECC and some countermeasures [7].

A. Power Analysis of ECC

Scalar point multiplication, $Q = dP$, is the basic operation of ECC. It can be performed with the binary algorithm BINALG as shown in Fig. 1a. In the figure, l is the number of bits in d and $d_{l-1} = 1$.

In BINALG, point doubling $2Q$ is performed in every iteration while point addition $Q+P$ is performed only for 1 bits. Since point doubling and point addition are performed with different formulae (see Section II) and thus have different power outputs, BINALG is vulnerable to SPA. An adversary can monitor the power consumption of a device performing BINALG and identify the point addition and doubling operations. A doubling followed by an addition indicates a 1 bit in d and two consecutive doublings indicate a 0 bit. The adversary can easily retrieve the value of d , which is secret in many algorithms.

To make an algorithm SPA resistant, one can remove the correlations between the execution path and the data being processed. In BINALG, the execution of point additions depends on the value of d_i . The additions are performed only when $d_i = 1$. In order to thwart SPA attacks, Coron [5] proposed BINALG', which is shown in Fig. 1b. BINALG' is a variant of BINALG. It always performs the point addition. The correct value is selected by d_i and carried over to the following iterations. According to [5], BINALG' is secure against SPAs. However, Coron discovered that it is vulnerable to DPA [5]. Oswald and Aigner further pointed out that any similar algorithms are also vulnerable to DPA attacks [4].

B. Randomized Algorithm

Oswald and Aigner proposed a randomized algorithm to thwart both SPA and DPA. They combined a variant of BINALG and an algorithm based on addition-subtraction chains [8]. The combined algorithm can be illustrated by the automaton in Fig. 2, which will be referred to as Randomized Automaton 1. When performing a scalar point multiplication dP , we start from State 0 and scan the bits in d from the least significant bit. For every bit, we either advance to a new state or stay in the same state, as specified in Fig. 2, and perform proper operations during transitions.

Since Randomized Automaton 1 is the combination of two automata, a random variable is used in States 1 and 11 to decide which of the automata to follow. When the automaton enters State 1 or 11, a random variable e is drawn. If the next bit is 1, only the transition with the correct random value will be followed. Suppose the automaton is in State 1 and the next input bit is 1, for example, the automaton will stay in State 1 if $e = 1$. Otherwise it will go to State 11. The transitions dependent on random variables are illustrated with dashed lines in Fig. 2.

Because of the random variables, the scalar point multiplication algorithm follows a different path and thus

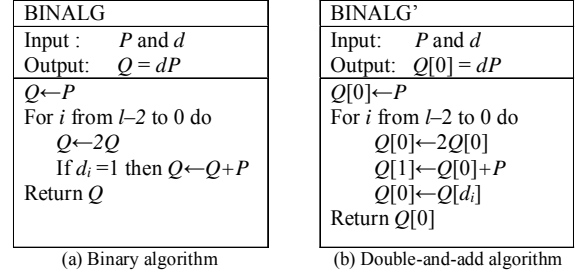


Fig. 1: Scalar point multiplication algorithms

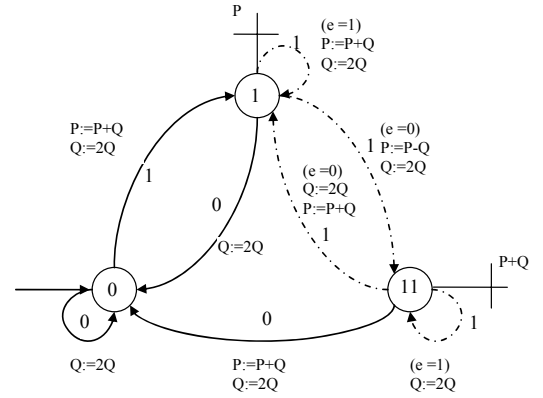


Fig. 2: Randomized Automaton 1 [4]

performs different operations even if the scalar d remains the same. Therefore it is difficult to launch traditional SPA or DPA to Randomized Automaton 1.

IV. ATTACKS ON RANDOMIZED ALGORITHMS

A. Attacks on Randomized Automaton 1

Randomized Automaton 1 seems secure against power analyses. Each of its executions generates a different power trace even if the scalar d remains the same. It is difficult to find out d by examining only one trace. However, we can collect many power traces. Each of the traces will tell us a different bit of information about the same d . With enough number of power traces, we can figure out all the bits in d .

Let us denote the doubling operation $2Q$ in Randomized Automaton 1 as D and the addition or subtraction operation ($P + Q$ or $P - Q$) as A . Here, we assume we cannot distinguish additions from subtractions. Also, we will refer to the bits in d as input bits as they are the input string to the automaton during the computation.

When launching the attack, we first collect a set of power traces by performing the scalar point multiplication N times. We assume that we can tell what operations, additions or doublings, were performed from the power traces. The point in the N multiplications can be the same as the random variable will result in different power traces.

We then start from State 0 and figure out one or several input bits a time by looking at the operations performed in *all* traces. From the input bits we detected, we determine the corresponding power outputs and the new state for each trace. Note that the automaton may end up in different states even on the same input. After we determine the new states, we remove from the power traces the outputs generated by the detected input bits. Then we continue to detect more input bits until all input bits are known.

The cases in State 0 are trivial because the power outputs reflect the input bits. If we see *A* is the first operation in the power traces, the corresponding input bit must be 1. If we see *D*, then the input bit is 0. After detecting the input bit, we also know the next state. And the next state of all traces is the same.

The cases in States 1 and 11 are more complicated because for the same input bit, the next state is not the same for all power traces because it is decided by random variables. Moreover, the transitions to different states have the same power outputs. For example, when the automaton is in State 1 and the next input bit is 1, the power output is always *AD* no matter the new state is State 1 or 11. So looking at the power output of the next input bit does not tell us the next state. Without knowing the next state, we cannot detect more input bits.

The problem can be solved by looking at three bits a time. Table I lists all possible power outputs of three input bits when the automaton is in State 1 or 11. The first column is the current state. The second column is the three input bits and the third column is all possible power outputs. The fourth column lists the input bits we can detect. The last column lists the new state after the detected bits in the fourth column are processed.

We first look at State 1. If the first bit of the 3-bit string is 0, the automaton will go to State 0 and perform a double, which is different from the operations performed on any other inputs. So we first check if *D* is the next operation in the power traces. If it is the case, the next input bit must be 0 and the automaton will go to State 0, for all traces.

We now look at the input strings starting with 1. There are four such strings, and all of them are listed in Table I. Because the automaton takes random transitions, we observe different power traces even for the same input string. For example, for string 111, we may observe six different power traces. Meanwhile, a power trace can be generated by different input strings. For instance, *ADADD* can be generated by 111, 110, or 100. So we cannot determine the input string by looking at only one power trace.

We noticed that some power outputs are unique to specific inputs. If such a power output sequence cannot be converted into another valid sequence by adding or removing operations at the end, we call it a unique sequence. For example, *ADDAAD* is a unique sequence of 111 and it is the only unique sequence of 111. Note that although *ADDDA* is listed under 111 only, it is not a unique sequence. When the trailing *A* is removed, it becomes *ADDD*, a valid output of 111 and 100.

To identify the correct input among the strings that start with 1, we check if any traces start with a unique sequence. If a unique sequence is found, the corresponding string must be

the correct input string. For example, if a power trace starts with *ADDAAD*, we can conclude that the next three input bits must be 111. Note that the unique sequences do not appear in every power trace. However, once they appear, it is certain that the corresponding string is the correct input. Since transitions are decided by random variables, we should be able to observe all possible outputs, including the unique sequences, if we have enough power traces. If none of unique sequences appears in a large number of power traces, we can exclude the strings that have unique sequences and then identify the input string among other strings. For example, in State 1, if *ADDAAD* does not appear in a large number of traces, we can exclude 111 and look at 110, 101, and 100 only.

With some strings being excluded, new unique sequences can be identified for the rest of the strings and then be used to check whether the corresponding strings are the correct input. If the new unique sequences are not found in any power traces, the corresponding strings can be excluded as well. We repeat the process until the correct input string is identified. For example, in State 1, after 111 is excluded, 100 has a unique sequence *ADDD*. We can check whether 100 is the correct input by searching for any power traces that start with *ADDD*. If no such power trace is found, we exclude 100 as well.

TABLE I: AD SEQUENCES OF THREE INPUT BITS IN STATES 1 AND 11 (RANDOMIZED AUTOMATON 1)

State	Input Strings	Power Traces	Bits Detected	Next State
1	0xx	D	0	0
	111	ADDD	1	1 or 11 (the next two input bits 11 are known)
		ADADD		
		ADDDA		
		ADADAD		
		ADDAAD		
		ADADDA		
	110	ADADD	110	0
		ADDAD		
		ADADAD		
	101	ADDAD	101	1
	100	ADDD	100	0
ADADD				
11	0xx	AD	0	0
	111	DDD	1	1 or 11 (the next two input bits 11 are known)
		DDDA		
		DDAAD		
		DAADD		
		DAADDA		
		DAADAD		
	110	DDAD	110	0
		DAADD		
		DAADAD		
	101	DADAD	101	1
	100	DADD	100	0

After the correct input string is identified, we remove from all power traces the outputs that correspond to the detected input string. We then continue to detect more bits starting from the new state. Input strings 110, 101, and 100 can be handled this way because all power traces end up in the same state. The string 111, however, is different because the automaton could be in State 1 or 11 after 111 is accepted. To handle this case, we only output one bit when we detect 111. The first 1 bit may put the automaton in State 1 or 11. Although we do not know which state a specific trace is in, we know that the next two input bits are 11. So the next three bits can only be 110 or 111. We search the unique sequences of 110 or 111 in both states, States 1 and 11, and determine which of 110 and 111 is the real input. If 111 is, we output the first 1 and repeat the process. If 110 is, we output all three bits and all traces will end up in State 0.

We may use similar methods to detect the input bits when in State 11, but it is not necessary in the attacks on Randomized Automaton 1. When we handle the 111 cases in State 1, we already cover the cases in State 11.

B. Success Rate

Assume X is a random variable that indicates the current state of Randomized Automaton 1, Y is a random variable that indicates the real input bits, and Z is the event that the next detected bit is correct. We use $P[Z|(X=x \cap Y=y)]$ to denote the success rate in State x when the real input string is y . x can be 0, 1, or 11 and y can be 000, 001, 010, ..., or 111. Table II lists $P[Z|(X=x \cap Y=y)]$ for all three states on 3-bit inputs. In State 0, we can always detect the input bit correctly. So we have $P[Z|(X=0 \cap Y=y)]=1$ for all y 's. No matter in which state, if the input bit is 0, we can always detect it successfully. Therefore, $P[Z|(X=x \cap Y=0xx)]=1$. In State 1 and 11, if y starts with 1 and has a unique sequence, $P[Z|(X=x \cap Y=y)]$, where $x = 1$ or 11, is calculated by dividing the number of paths that generate a unique sequence by the total number of possible paths of y . If in State x , y is the last string after all others are excluded, $P[Z|(X=x \cap Y=y)] = 1$ because y is never misidentified as other strings. For example, $P[Z|(X=1 \cap Y=101)]=1$ because in State 1, 101 is only identified after 111, 110, and 100 are excluded.

From Table II, we can see that State 1 has the lowest success rate. In State 1, we may make mistakes if the input string is 100, 110, or 111 because the unique sequences may not appear. With a single power trace, the probability of wrong detections is 1/2, 1/2, and 3/4 for input strings 100, 110, and 111, respectively. As the number of power traces increases, the probability decreases exponentially. With N power traces, the overall error rate in State 1 can be calculated as:

$$\frac{1}{8} * \left(\frac{1}{2}\right)^N + \frac{1}{8} * \left(\frac{1}{2}\right)^N + \frac{1}{8} * \left(\frac{3}{4}\right)^N = \frac{1}{2^{N+2}} + \frac{3^N}{2^{2N+3}}$$

When N is 100, the error rate is about 2^{-44} . When N is 200, it is about 2^{-86} . So our attacks do not require a large number of power traces to achieve a very high success rate.

TABLE II: SUCCESS RATE WITH ONE POWER TRACE

Y \ X	0	1	11
0xx	1	1	1
100	1	1/2	1
101	1	1	1
110	1	1/2	1
111	1	1/4	1/4

C. Related Work

Okeya and Takagi also proposed a method to attack Randomized Automaton 1 [9]. Their method has three weaknesses [10]. First, they only focused on Randomized Automaton 1. Second, they assumed that the attacker knows the bit length of the secret value. Third, they overlook the case that one of the points in addition is the point at infinity.

Compared with Okeya and Takagi's algorithms, our attacks do not need to know the bit length of the secret key and are not limited to Randomized Automaton 1.

V. CONCLUSIONS

The randomized scalar point multiplication algorithm was proposed to thwart power analysis attacks. In this paper, we proposed an effective attack on the randomized algorithm. Our attack does not require a large number of power traces to achieve a very high success rate. With 200 power traces, the error rate of a single bit is smaller than 2^{-86} .

REFERENCES

- [1] V. S. Miller, "Use of elliptic curves in cryptography," *Advances in Cryptology, 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] S. Vanstone, "ECC holds key to next-gen cryptography," Available: <http://www.us.design-reuse.com/articles/article7409.html>, [Mar 2004].
- [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] J. S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," *Proceedings of CHES 1999*, pp. 292-302, August 1999.
- [6] A. J. Menezes, "Elliptic curve public key cryptosystems," Kluwer Academic Publishers, 1993.
- [7] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *Proceedings of CRYPTO'99*, pp. 388-397, August 1999.
- [8] F. Morain, and J. Olivos, "Speeding up the computations on an elliptic curve using addition-subtraction chains," *Theoretical Informatics and Applications*, vol. 24, pp. 531-543, 1990.
- [9] 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 7th Australasian Conference on Information Security and Privacy, ACISP 2002*, pp. 420-435, July 2002.
- [10] D. Han, N. S. Chang, S. W. Jung, Y. H. Park, C. H. Kim, and H. Ryu, "Cryptanalysis of the full version randomized addition-subtraction chains", *Proceedings of 8th Australasian Conference on Information Security and Privacy, ACISP 2003*, pp. 67 - 78, July 2003.