A Refined Simple Power Analysis Attack on ECC with Countermeasures

Lihui Wang, Qing Li, Zhimin Zhang, Weijun Shan
Central Research Institute
Shanghai Fudan Microelectronics Group Company Limited
Shanghai, 200433, China
E-mail: wanglihui@fmsh.com.cn

David Wei Zhang
School of Microelectronics, Fudan University
Collaborative Innovation Center of IC Design & manufacturing in Yangtze River Delta (2011)
Shanghai, 200433, China
E-mail: dwzhang@fudan.edu.cn

***Abstract*—Elliptic Curve Cryptography (ECC) is becoming widely deployed in embedded cryptographic devices. However, simple power analysis (SPA) attacks may retrieve secret keys by exploiting the power consumption of ECC devices. To resist the SPA attack there appear a number of countermeasures and the most widely used methods are Montgomery ladder and double-and-add-always algorithm. This paper proposes a refined simple power analysis attack to these countermeasures. Experimental results on smart cards demonstrate that this attack method can retrieve secret keys by distinguishing the power consumption of different jump instructions if the implementation of ECC's countermeasure is not carefully designed. The experiments also demonstrate that the proposed countermeasure in this paper can resist this kind of SPA attack.**

Keywords—ECC; cryptography; SPA; scalar multiplication

I. INTRODUCTION

Elliptic Curve Cryptography (ECC) was first proposed by Miller and Koblitz in 1985 [1]. 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.

However, ECC is found to be vulnerable to simple power analysis (SPA) attacks [2] as other public-key cryptosystems. SPA attackers examine the power traces of cryptographic computations and distinguish the power consumption caused by the secret key. The core operation of ECC is binary scalar multiplication, which consists of a point doubling operation if the key bit is '0', or of a point doubling followed by point addition operation if the key bit is '1'.

There are a number of countermeasures for ECC against SPA attacks, and the countermeasures can be classified into two groups. The first group is at the logic level, such as Sense Amplifier Based Logic (SABL) [3], Wave Dynamic Differential Logic (WDDL) [4, 5, 6], Random Switching Logic (RSL) [7, 8], and so on. All of these countermeasures' aim is balancing the power consumed by the cryptography circuit but they all largely increase the computation cost and more expensive to practical application. Another group of countermeasure is at algorithm level, such as Montgomery method [9] of the scalar multiplication and double-and-add-always algorithm [10]. Authors in [11, 12] also proposed an algorithm-level countermeasure -- Side-Channel Atomicity, which divided the whole procedure of point doubling and addition into "atoms". However, Montgomery ladder and double-and-add-always algorithm are most widely used as so far.

However, the above countermeasures can't ensure the absolute security of scalar multiplication when there is more powerful simple power analysis. The main contribution of this paper is the development of a refined and more powerful simple power analysis attack which is even applicable to elliptic-curve scalar point-multiplication algorithms with countermeasures such as Montgomery and double-and-add-always algorithm. This attack shows that if the implementation is not carefully designed, the countermeasures can also be defeated. And we give some advises about countermeasures at last.

The rest of this paper is organized as follows. Section II introduces the background theory regarding the ECC's scalar multiplication algorithm and SPA attack. Section III presents a refined SPA attack of on ECC with countermeasures. Attack results are given in Section IV. Finally, we conclude in Section V.

II. BACKGROUND

The following section will briefly cover the background theory, required for understanding this work. First, we give an overview of ECC algorithm and followed by a brief introduction to side channel analysis.

A. Elliptic curve scalar multiplication

An elliptic curve is a set of points P which are solutions of a bivariate cubic equation over a finite field, such as the prime finite field and the binary finite field [13].

In ECC, the secret key d is mainly involved in the scalar multiplication operations, while scalar multiplication is realized by repeated addition of the same point. If d is a positive integer and P a point on an elliptic curve, the scalar multiplication dP is the result of adding d copies of P [14,15]:

$$dP = \underbrace{P + P + \ldots + P}_{d}$$

There is a common implementation of ECC's scalar multiplication, such as the Binary Method shown in Algorithm 1:

Algorithm 1 (Binary algorithm)

```
Input: $d, P$
Output: $dP$
1. $T_0 = P$
2. for $i = n - 2$ to 0
        $T_0 = 2T_0$
        if $d_i = 1$ then $T_0 = T_0 + P$
3. output $T_0$
```

The above scalar multiplication implementations contain point addition and point doubling. The key d determines the procedure of doubling and addition operation. It consists of a point doubling operation if the key bit is 0, and a point doubling followed by a point addition operation if the key bit is 1.

B. Power analysis

Power analysis attacks use the fact that the instantaneous power consumption of a hardware device is related to the instantaneous computed instructions and the manipulated data. There are two types of power analysis, the simple power analysis (SPA) and the differential power analysis (DPA), which are described in [2, 16]. DPA computes hypotheses of the power consumption for each input and key candidate and compares them to the recorded power consumption of the device. Such a hypothesis can be computed by calculating the Hamming weight (HW) of a processed value. Finding a suited intermediate value, which reveals information about the key, is specified as leakage analysis. In order to compare the calculated hypothesis to the measured power consumption, methods like the Pearson correlation or the difference in means can be used.

In this paper, we only take care of the SPA on scalar multiplication. SPA makes use of such an instruction performed during an scalar multiplication algorithm that depends on the data being processed. Apparently, Algorithm 1 has a branch instruction conditioned by a secret exponent d, and thus it reveals the secret d if power consumptions of point addition and point doubling are distinguishable.

The point doubling and addition can be implemented as the traditional procedure in [14]. When a=p-3 and Z=1, point doubling needs 8 multiplications, but point addition needs 11 multiplications. And the operation procedure of point doubling is fully different from that of point addition. Therefore their power consumption are highly distinguishable and vulnerable to SPA.

C. Countermeasures

In order to be resistant against SPA, any branch instruction of exponentiation algorithm should be eliminated. There are mainly two types of countermeasures at algorithm level: the fixed procedure method and the indistinguishable method. The fixed procedure method deletes any branch instruction conditioned by a secret exponent d like double-and-add-always method, Montgomery-ladder method, and window-based method [17]. The indistinguishable method conceals all branch instructions of exponentiation algorithm by using indistinguishable addition and doubling operations, in which dummy operations are inserted.

The most widely used method to resist SPA attack are double-and-add-always method and Montgomery-ladder method. Double-and-add-always method is described in Algorithm 2. In this method, the operations of point doubling and addition are always executed whatever the key bit is through insertion of false point additions. By measuring the power consumption during the ECC operation, the attackers can't retrieve the secret key.

Algorithm 2 (Double-and-add-always algorithm)

```
Input: $d, P$
Output: $dP$
1. $T_0 = P$
2. for $i = n - 2$ to 0
        $T_0 = 2T_0$
        if $d_i = 1$ then $T_0 = T_0 + P$
        else $T_1 = T_0 + P$
3. output $T_0$
```

III. A Refined Simple Power Analysis

In this section, we present a powerful simple power analysis attack on scalar multiplication with double-and-add-always method. Note that this attack can also be used to defeat the Montgomery-ladder method.

A. Description of the Attack

As is usually the case, the scalar multiplication is realized through the combination of hardware and software. By the definition of ECC, we find that the elementary operations of scalar multiplication are modular multiplication, modular addition and modular subtraction. These elementary operations especially modular multiplication are very cost if they are realized by software. So they are usually realized by hardware such as coprocessor and combined into point doubling and addition even scalar multiplication by implementation of software on a microcontroller. Microcontrollers have an instruction set that typically consist of arithmetic instructions such as addition, logical instructions such as exclusive-or, data transfer instructions such as move and branching instructions such as jump.

According to algorithm 2, there is a "if…else…" that is a conditional jump instruction. After being compiled, the execution flow is different when bits of exponent key are not the same. Because each instruction works on a number of bytes and involves different components of the microcontroller, these are physically separate components of the microcontroller and they differ in functionality and implementation. So the power consumption will be different more or less when d_i is 0 or 1. If we can distinguish this tiny difference by some signal processing methods such as pattern match, the private key is broken.

If the Montgomery-ladder method and window-based method also use the conditional jump instruction, it can be broken in the same way.

Note that this attack can also be applied to ECDSA. Despite only the ephemeral key can be retrieved from the scalar multiplication by above attack method, we can recover the sign key easily by follow equation:

$$s = k^{-1}(e + rd)\ mod\ n$$

Where r and s are signature, e is the Hash value of message, n is the order of the elliptic curve, k is ephemeral key, d is private key to be broken.

B. Countermeasures

Because only one power trace of scalar multiplication is need in above attack method and the target is only the exponent key, the common countermeasures like exponent blinding, base point blinding, random projective coordinate [10] is not useful.

As mentioned above, the distinguishing difference of power consumption is brought by conditional jump instruction such as "if…else…" So there should be no conditional jump instruction. But how can we execute different point addition according to different exponent key bit?

For example, if the original execution of the true and false point additions is determined as follows:

Algorithm 3

```
1. if  $d_i = 0$
            ECC_Command(0x1FC8, 0x03010102)
2. else
            ECC_Command(0x1C81, 0x03010203)
```

Where the first parameter means the RAM blocks used which determine the position of input and output, the second parameter means the type of elementary operations. Several different these commands compose true point addition or false point addition.

We can take the d_i as the input parameter, and change algorithm 3 to algorithm 4 as follows:

Algorithm 4

```
        ECC_Command(0x1FC8-$d_i$ *0x0347,
                        0x03010102+$d_i$ *0x00000101)
```

From algorithm 4, there is no conditional jump instruction, and in this method, the same operations of point doubling and addition are always executed whatever the key bit is '0' or '1'. By measuring the power consumption during the ECC operation, the attackers can't retrieve the secret key.

IV. EXPERIMENT RESULT

A. Attack Result Befor Improving Contermeasure

To examine the effect of the proposed SPA attack in section III, we conduct experiments on a software implementation of ECC. Our experimental setup consists of a PC, a power tracer, a smart card and a LeCroy oscilloscope. The smart card has a PAE coprocessor and algorithm library. The coprocessor is used to compute modular multiplication and addition. The algorithm library is used to provide a high level interface to ECC cryptography implemented on the coprocessor and includes countermeasures against SPA and DPA attacks.

Without loss of generality, we set the private key to be "A6" whose binary representation is 10100110. According to algorithm 2, the true and false order of point addition is "FTFFTTF", where 'F' means a false point addition and 'T' means a true point addition. After recording the power consumption of this scalar multiplication we align the first point addition to other six point additions and find some tiny differences in certain region as shown in Fig.1.

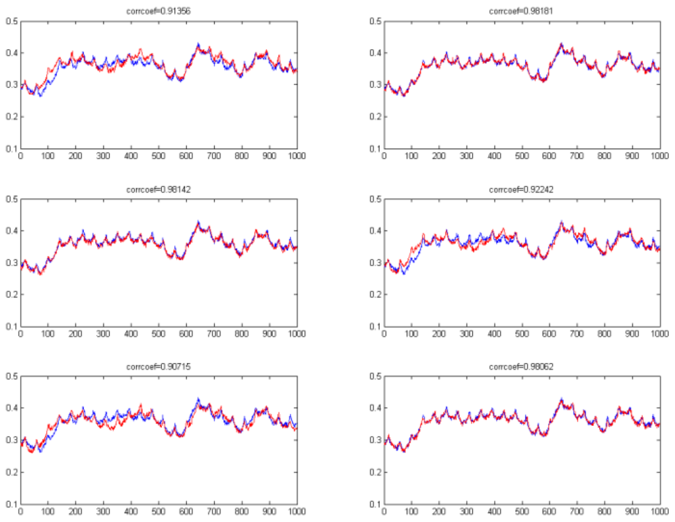

Fig. 1. The differences between first point addition and others befor Improving Contermeasure

From Fig.1 we can see that the first point addition is almost the same with the third, forth, seventh point addition and different with others more or less. This judge is also derived from correlation coefficient of two point additions. For example, when the types of two executions are the same, the correlation coefficient is higher than 0.98, otherwise is lower than 0.93. So we can select a proper threshold such as 0.96 to determine whether the two executions are the same.

Based on the previous analysis we can infer that the true and false order of point addition may be "FTFFTTF" or "TFTTFFT", after a simple verify using input and output of scalar multiplication, the former can be confirmed and we can recover the secret key is "A6".

B. Attack Result After Improving Contermeasure

After improving the countermeasure as described in section III, we repeat the above attack, and also set the private key to be "A6". After recording the power consumption of this scalar multiplication we align the first point addition to other six point additions and try to find some tiny differences in the same region as shown in Fig.2.

From Fig.2 it can be seen that the first point addition is almost the same with others. The correlation coefficients are all higher than 0.97 and the second, third, sixth correlation coefficient are not always higher than other three. We can't distinguish between the false point addition and the true point addition by observing the power consumption or comparing correlation coefficients. So the refined attack method can't be applied to this countermeasure.

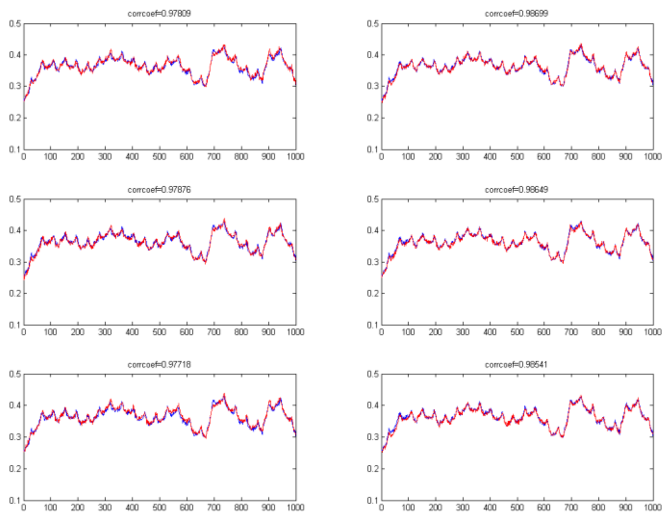

Fig. 2. The differences between first point addition and others after Improving Contermeasure

After that we make the above method into program and attack on a standard scalar multiplication using double-and-add-always algorithm. The experimental results show that it is easy to recover the whole secret key less than a second.

V. CONCLUSION

We presented a refined simple power analysis attack on elliptic curve scalar multiplication such as double-and-add-always algorithm. The method basically uses the information about the conditional jump instruction of elliptic curve operations, and retrieves the secret key by measuring the power consumption of different kind of point addition during scalar multiplication. The method can be considered as an enhancement because it works even in cases where the standard simple power attack fails because of the countermeasures such as the double-and-add-always algorithm. The approach is a general method in the sense that it can be used to attack arbitrary elliptic curve scalar multiplication algorithms that using conditional jump instructions. We demonstrated that it can be effectively applied on the example of a modification of a scalar multiplication algorithm where the standard simple power-analysis attack fails. We also analyzed the reason of leakage and pointed out that this can be improved by discarding the conditional jump instructions. Based on the analysis of a concrete implementation code we proposed a method that only take the d_i as the input parameter in order to get same operations of point doubling and addition whatever the key bit is ‘0’ or ‘1’. 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 whose countermeasure implementation is not carefully designed. Our analysis also indicates that the concrete implementation of countermeasure is very important despite it appears to be safe. The work done in this paper is just a try to breaking the ECC’s private key using a simple power analysis, more novel method and further researches are expected in the future.

Acknowledgment

This work is supported by "China’s 12th Five-Year Plan" Microelectronics Advance Research Projects Fund (No. 51308010609) and Shanghai Science and Technology Talents Funds (No. 14R21421100).

References

[1] D. Hankerson, A. Menezes, and S. Vanstone, “Guide to Elliptic Curve Cryptography,” Springer-Verlag New York, 2004.

[2] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” Proceedings of CRYPTO’99, Springer-Verlag, Berlin, pp. 388–397, 1999.

[3] K. Tiri, M. Akmal, and I. Verbauwhede, “A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards,” ESSCIRC 2002 pp. 403-406, 2002.

[4] K. Tiri and I. Verbauwhede, “A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation,” Proceedings of the Design, Automation and Test in Europe Conference, Paris, France, 2004: 246-251.

[5] K. Tiri and I. Verbauwhede, “Secure Logic Synthesis,” International Conference on Field Programmable Logic and Applications (FPL 2004), Lecture Notes in Computer Science, vol. 3203, pp. 1052-1056, August 2004.

[6] I. Verbauwhede, K. Tiri, D. Hwang, A. Hodjat, and P. Schaumont, “Circuits and Design Techniques for Secure ICs Resistant to Side-Channel Attacks,” International Conference on IC Design & Technology (ICICDT 2006), pp. 1-4, May 2006..

[7] D. Suzuki, M. Saeki, and T. Ichikawa, “Random Switching Logic: A Countermeasure against DPA based on Transition Probability,” Cryptology ePrint Archive (http://eprint.iacr.org/), Report2004/346, 2004

[8] T. Popp and S. Mangard, “Masked Dual-Rail Pre-charge Logic: DPA Resistance without the Routing Constraints,” Proc. of the Workshop on Cryptographic Hardware and Embedded Systems (CHES 2005), LNCS 3659, pp. 172-186, August 2005.

[9] P. L. Montgomery, “Speeding the Pollard and Elliptic Curve Methods for Factorizations,” Mathematics of Computation, vol. 48, pp. 243-264, 1987.

[10] J.S. Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems, Cryptographic Hardware and Embedded Systems (CHES’ 99), LNCS1717, (1999), 292-302

[11] B. Chevallier-Mames, M. Ciet, and M. Joye, “Lowcost solutions for preventing simple side-channel analysis: Side-channel atomicity,” IEEE Transactions on Computers, 53(6):760–768, 2004.

[12] T. Chen, H.Y. Li, et al. Countermeasure of ECC against Side-Channel Attacks: Balanced Point Addition and Point Doubling Operation Procedure. Information Processing, 2009. APCIP 2009.

[13] A. J. Menezes, “Elliptic Curve Public Key Cryptosystems,” Kluwer Academic Publishers, Norwell, 1993.

[14] IEEE Std-1363-2000: Standard Specifications for Public Key Cryptography, January 2000.

[15] X. F. Tang, “The VLSI Implementation of ElliPtie Curve Cryptography IP,” Master Thesis, Circuits and Systems of Department of Electric Engineering of Zhejiang University, China, February 2004

[16] C. Kocher, ” Timing attacks on Implementations of Diffie-Hellman, RSA, DSS, and other system ” , CRYPTO ’ 96, Lecture Notes in Computer Science, 1109(1996), Springer-Verlag, 104－113

[17] K. Koyama and Y. Tsuruoka, “Speeding up elliptic cryptosystems by using a signed binary window method ” , Advances in Cryptology-Proceedings of Crypto’92, Lecture Notes in Computer Science, 740 (1993), Springer-Verlag, 345－357.