情報保護學會論文誌
第 22 卷 第 5 號, 2012. 10

Euclidean Addition Chain을 사용하는 타원곡선 스칼라 곱셈 연산에 대한 오류 주입 공격*

이 수 정,† 조 성 민, 홍 석 희‡
고려대학교 정보보호연구원

A fault attack on elliptic curve scalar multiplication based on Euclidean Addition Chain*

Soo Jeong Lee,† Sung Min Cho, Seokhie Hong‡
Center for Information Security Technologies, Korea University

요 약

오류 주입 공격은 암호 장치가 동작하는 동안에 오류를 주입하여 얻은 부가적인 정보를 이용하여 비밀키에 대한 정보를 얻는 공격 방법이다. 오류 주입 공격은 소형 암호 장치에 내장된 암호 알고리즘의 키를 찾을 수 있는 가장 강력한 공격 방법으로 오류 주입 공격 및 오류 탐지 방법에 대한 연구가 활발히 진행되고 있다. 2009년 S. Pontarelli 등은 Euclidean Addition Chain (EAC)를 사용하는 타원곡선 스칼라 곱셈 알고리즘에 대한 오류 탐지 방법을 소개하였다. 본 논문에서는 S. Pontarelli 등이 제안한 오류 탐지 방법이 적용된 알고리즘에 대한 새로운 오류 주입 공격 방법을 제안한다. 제안하는 공격 방법은 타원곡선 스칼라 곱셈 알고리즘의 상수 k에 대한 EAC에 비트 플립 오류 (bit flip error)를 주입하여 비밀키에 대한 정보를 얻어낸다.

ABSTRACT

Fault attacks manipulate the computation of an algorithm and get information about the private key from the erroneous result. It is the most powerful attack for the cryptographic device. Currently, the research on error detection methods and fault attacks have been studied actively. S. Pontarelli et al. introduced an error detection method in 2009. It can detect an error that occurs during Elliptic Curve Scalar Multiplication (ECSM). In this paper, we present a new fault attack. Our attack can avoid the error detection method introduced by S. Pontarelli et al. We inject a bit flip error in the Euclidean Addition Chain (EAC) on the private key in ECSM and retrieve the private key.

Keywords: Elliptic Curve Scalar Multiplication, Elliptic Curve Cryptosystem, Fault Attacks, Euclidean Addition Chain

I. 서 론

타원곡선 암호시스템 (ECC, Elliptic Curve Cryptosystem)은 1985년 Koblitz와 Miller에 의해 처음 제안되었다[1,2]. 타원곡선 암호시스템은 기존의 RSA와 Elgamal 공개키 암호시스템에 비하여 짧은 키 길이로 유사한 안전성을 제공한다는 장점을 가지고 있다. 또한 키의 길이가 짧기 때문에 스마트 카드와 같이 자원이 제한된 분야에서 효율적으로 사용될 수 있다. 타원곡선 암호시스템의 안전성과 효율성에 가장 큰 영향을 주는 연산은 타원곡선 스칼라 곱셈

접수일(2012년 9월 10일), 게재확정일(2012년 9월 21일)
* 본 연구는 지식경제부 및 정보통신산업진흥원의 대학IT연구센터육성 지원사업의 연구결과로 수행되었음 (NIPA-2012-H0301-12-3007)
† 주저자, soojeong88@korea.ac.kr
‡ 교신저자, shhong@korea.ac.kr

(Elliptic Curve Scalar Multiplication) 연산이다. 타원곡선 스칼라 곱셈 연산은 주어진 타원곡선 상의 점 P와 임의의 상수 k에 대해서 kP $(=P+...+P,\ k\,\text{times})$를 계산하는 것이다. 타원곡선 스칼라 곱셈 연산의 효율성을 높이기 위한 방법으로 Double and add 알고리즘, Montgomery ladder 알고리즘, Euclidean Addition Chain (EAC)을 사용하는 타원곡선 스칼라 곱셈 알고리즘 등이 제안되었다[3,4]. 이 중 EAC를 사용하는 타원곡선 스칼라 곱셈 알고리즘은 상수 k를 EAC로 표현하여 효율적으로 kP를 계산하는 방법이다. EAC를 사용하는 타원곡선 스칼라 곱셈 알고리즘의 효율성을 높이기 위하여 Co-Z 덧셈 연산 방법, (X, Z)만을 사용하는 덧셈 연산 방법 등이 제안되었다[5-7].

수학적으로 안전한 것으로 알려진 알고리즘도 구현 단계에서 고려되지 못한 부가적인 정보의 누출이 있다는 것이 알려졌다. 부채널 공격은 이러한 부가적인 정보로부터 비밀키를 알아내는 공격 방법이다[8]. 부채널 공격 중 하나인 오류 주입 공격 (Fault Attacks)은 1996년 Boneh 등이 처음 소개한 공격 방법으로 암호 장치가 동작할 때 오류를 주입하여 비밀키에 대한 정보를 얻는다[9]. 오류 주입 공격은 타원곡선 암호시스템을 포함한 대부분의 암호 알고리즘에 적용 가능한 공격 방법으로 알려져 있다[10-16].

2009년 S. Pontarelli 등은 EAC를 이용한 타원곡선 암호시스템에서 오류 주입 여부를 탐지하기 위한 오류 탐지 방법을 제안하였다[17]. 본 논문에서는 [17]의 오류 탐지 방법으로 오류 주입 여부를 탐지할 수 없는 새로운 오류 주입 공격을 제안한다. 제안하는 방법은 EAC에 비트 플립 오류(bit flip error)를 주입하여 비밀키에 대한 정보를 얻는다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 기술한다. 3장에서 제안하는 오류 주입 공격 방법을 설명한다. 4장에서는 공격 방법에 대한 대응방법을 기술하며 마지막으로 5장에서 결론을 맺는다.

II. 관련 연구

본 장에서는 EAC를 사용하는 타원곡선 스칼라 곱셈 연산 방법을 소개한다. 또한 2009년 S. Pontarelli 등이 제안한 EAC를 사용하는 타원곡선 스칼라 곱셈 알고리즘에 대한 오류 탐지 방법을 설명한다.

2.1 Euclidean Addition Chain(EAC)을 사용하는 타원곡선 스칼라 곱셈 연산

EAC를 사용하는 타원곡선 스칼라 곱셈 연산은 2007년 A. Byrne 등이 처음 소개하였다[4]. 이 방법은 상수 k를 EAC로 표현하여 타원곡선 스칼라 곱셈 연산을 한다. $\forall 4 \le i \le l$에 대하여, $a_1 = 1$, $a_2 = 2$, $a_3 = 3$, …, $a_l = k$ 일 때, 임의의 상수 k에 대하여 EAC $c = (c_4, ..., c_{l-1}, c_l)$를 구성하는 방법은 다음과 같다.

$$\begin{cases} a_i = a_{i-1} + a_{i-2} & if\ c_i = 1 \\ a_i = a_{i-1} + a_j (for\ some\ j < i-2) & if\ c_i = 0 \end{cases}$$

예를 들어, $k = 40$인 경우, k에 대한 EAC는 다음과 같이 표현할 수 있다.

$$\begin{cases} a_1 = 1 \\ a_2 = 2 \\ a_3 = a_2 + a_1 = 3 \\ a_4 = a_3 + a_1 = 4 & \rightarrow c_4 = 0 \\ a_5 = a_4 + a_3 = 7 & \rightarrow c_5 = 1 \\ a_6 = a_5 + a_4 = 11 & \rightarrow c_6 = 1 \\ a_7 = a_6 + a_5 = 18 & \rightarrow c_7 = 1 \\ a_8 = a_7 + a_6 = 29 & \rightarrow c_8 = 1 \\ a_9 = a_8 + a_6 = 40 & \rightarrow c_9 = 0 \end{cases}$$

[표 1]은 $Q = 40P$를 구하는 예이다. (U_1, U_2)를 EAC를 사용하는 타원곡선 스칼라 곱셈 알고리즘의 중간값이라 하자. (U_1, U_2)의 초기값을 $(2P, P)$라 할 때, EAC $c = (0,1,1,1,1,0)$를 사용하여 타원곡선 스칼라 곱셈 $40P$를 계산하는 과정은 [표 1]과 같이 나타낼 수 있다.

또한 $\forall 4 \le i \le l$에 대하여, (U_1, U_2)는 다음과 같이 표현된다.

[표 1] EAC를 사용하여 $Q = 40P$ 를 계산하는 방법

step	c	U_1	U_2
1		$2P$	P
2	$c_4 = 0$	$3P$	P
3	$c_5 = 1$	$4P$	$3P$
4	$c_6 = 1$	$7P$	$4P$
5	$c_7 = 1$	$11P$	$7P$
6	$c_8 = 1$	$18P$	$11P$
7	$c_9 = 0$	$29P$	$11P$
		$Q = 40P$	

입력 : P, $2P$, 상수 k에 대한 EAC $c=(c_4,...,c_{l-1},c_l)$
출력 : $Q=kP$

1. $(U_1, U_2)\leftarrow(2P,P)$
2. for $i=4$ to l
3. if $c_i=1$ then
4. $(U_1, U_2)\leftarrow(U_1+U_2, U_1)$
5. else
6. $(U_1, U_2)\leftarrow(U_1+U_2, U_2)$
7. end for
8. return $Q\leftarrow U_1+U_2$

〔그림 1〕 Euclidean Addition Chain(EAC)를 사용하는 타원곡선 스칼라 곱셈 알고리즘

$$\begin{cases}(U_1,U_2)=(U_1+U_2,U_1) & if\ c_i=1\\ (U_1,U_2)=(U_1+U_2,U_2) & if\ c_i=0\end{cases}$$

이와 같은 방법으로 구성된 EAC를 사용하는 타원곡선 스칼라 곱셈 알고리즘은 〔그림 1〕과 같다.

2.2 타원곡선 스칼라 곱셈 연산에서 오류 탐지 방법

2009년 S. Pontarelli 등은 EAC를 사용하는 타원곡선 스칼라 곱셈 연산에 대한 오류 탐지 방법을 제안하였다〔17〕. 제안하는 오류를 탐지 방법은 다음과 같다. EAC를 사용하는 타원곡선 스칼라 곱셈 알고리즘의 $i-1$번째 중간값을 $U_1(i-1)$, $U_2(i-1)$라 하자. i번째 중간값 $U_1(i)$, $U_2(i)$는 다음과 같이 표현한다.

$$U_1(i)=U_1(i-1)+U_2(i-1)$$

$$U_2(i)=\begin{cases}U_1(i-1) & if\ c_i=1\\ U_2(i-1) & if\ c_i=0\end{cases} \quad (1)$$

또한, c_i에 대하여 $U_C(i)$를 다음과 같이 정의하자.

$$U_C(i)=\begin{cases}U_2(i-1) & if\ c_i=1\\ U_1(i-1) & if\ c_i=0\end{cases} \quad (2)$$

$U_D(i+1)$를 $U_1(i)$과 $U_2(i)$의 차라고 하면, $U_D(i+1)$는 다음과 같이 계산된다.

$$\begin{aligned}U_D(i+1)&=U_1(i)-U_2(i)\\ &=U_1(i-1)+U_2(i-1)-U_2(i)\end{aligned} \quad (3)$$

수식 (1)에서 $U_2(i)$는 c_i에 따라 $U_1(i-1)$ 또는 $U_2(i-1)$값을 가지기 때문에 $U_D(i+1)$는 $U_2(i-1)$ 또는 $U_1(i-1)$값을 가지게 된다. 즉, c_i의 비트값에 대하여 $U_D(i+1)=U_C(i)$가 성립한다. 이러한 특징 때문에 $U_D(i+1)$와 $U_C(i)$의 값을 비교하여 일치하지 않는 경우 오류가 주입되었음을 탐지할 수 있다. S. Pontarelli 등이 제안한 오류 탐지 방법이 적용된 알고리즘은 〔그림 2〕와 같다. 〔그림 2〕는 U_D와 U_C값을 비교하여 오류 주입 여부를 판단한다. 중간값 U_1 또는 U_2에 오류가 주입되는 경우 U_D값이 올바르지 않기 때문에 U_D와 U_C값을 비교하여 오류를 탐지할 수 있다.

입력 : P, $2P$, 상수 k에 대한 EAC $c=(c_4,...,c_{l-1},c_l)$
출력 : $Q=kP$, Error detection

1. $U_1\leftarrow 2P$, $U_2\leftarrow P$, $U_C\leftarrow P$
2. for $i=4$ to l
3. $U_D\leftarrow U_1-U_2$, $U_1\leftarrow U_1+U_2$
4. if $U_D\neq U_C$ then
5. return $Q=0$, $error=1$
6. end if
7. if $c_i=1$ then
8. $U_2\leftarrow U_1$, $U_C\leftarrow U_2$
9. else
10. $U_2\leftarrow U_2$, $U_C\leftarrow U_1$
11. end if
12. end for
13. $Q\leftarrow U_1+U_2$
{the following steps cheek the last two results of the EAC}
14. $U_D\leftarrow U_1-U_2$, $U_1\leftarrow U_1+U_2$
15. if $U_D\neq U_C$ then
16. return $Q=0$, $error=1$
17. end if
18. $U_2\leftarrow U_2$, $U_C\leftarrow U_1$, $U_1\leftarrow U_1+U_2$, $U_D\leftarrow U_1-U_2$
19. if $U_D\neq U_C$ then
20. return $Q=0$, $error=1$
21. end if
22. return Q, $error=0$

〔그림 2〕 Euclidean Addition Chain(EAC)를 사용하는 타원곡선 스칼라 곱셈 알고리즘 (Error detection)

III. 제안하는 오류 주입 공격

3.1 표기법

본 논문에서는 제안하는 오류 주입 공격 방법을 설명하기 위한 값들을 다음과 같이 표기한다.

- l : Euclidean Addition Chain(EAC)의 길이.
- $c=(c_4,...,c_{l-1},c_l)$: EAC

- $c'=(c_{i+1},...,c_{l-1},c_l)$: 최하위 c_l부터 c_{i+1}까지 EAC.
- $c''=(c_{i+2},...,c_{l-1},c_l)$: 최하위 c_l부터 c_{i+2}까지 EAC.
- c_i : EAC의 i번째 값.
- $\tilde{c}_i$: 비트 플립 오류를 주입한 EAC의 i번째 값.
- $\tilde{c}$: c_i에 비트 플립 오류를 주입한 EAC.
- (U_{1,c_i}, U_{2,c_i}) : c_i에 대한 중간값.
- Q : 올바른 결과값.
- $\tilde{Q}_{c_i}$: c_i에 오류가 주입된 결과값.
- $\tilde{Q}_{c_i,c_{i+1}=0}$: c_{i+1}를 0으로 가정한 뒤, c_i에 오류가 주입된 결과값.

3.2 오류 주입 공격

제안하는 공격 방법은 타원곡선 스칼라 곱셈의 상수 k에 대한 EAC $c=(c_4,...,c_{l-1},c_l)$에 비트 플립 오류(bit flip error)를 주입한다. 제안하는 공격 방법은 c_i에 오류를 주입하여 얻은 결과값과 c_{i+1}까지의 EAC $c'=(c_{i+1},...,c_{l-1},c_l)$를 이용하여 c_{i+1}의 비트값과 c_i에 대한 중간값 (U_{1,c_i}, U_{2,c_i})를 알아낸다. 먼저 다음과 같은 과정으로 c_l과 c_{l-1}에 대한 중간값 $(U_{1,c_{l-1}}, U_{2,c_{l-1}})$를 알 수 있다.

단계 1. 〔그림 1〕의 정상적인 실행으로 올바른 결과값 Q를 얻는다. c_l에 대한 중간값 (U_{1,c_l}, U_{2,c_l})에 대하여 올바른 결과값은 $Q=U_{1,c_l}+U_{2,c_l}$이다.

단계 2. c_l에 비트 플립 오류를 주입하여 오류가 주입된 결과값 $\tilde{Q}_{c_l}$을 얻는다. c_l에 오류를 주입한 경우, 〔표 2〕와 같이 c_l에 대한 U_1값과 U_2값은 $(U_{1,c_l}, U_{1,c_l}-U_{2,c_l})$이 되며 오류가 주입된 결과값은 $\tilde{Q}_{c_l}=2U_{1,c_l}-U_{2,c_l}$이다. $Q+\tilde{Q}_{c_l}=3U_{1,c_l}$이므로 U_{1,c_l}값을 알 수 있다. 또한 Q와 U_{1,c_l}의 차를 계산하여 U_{2,c_l}값을 알 수 있다.

단계 3. c_{l-1}에 비트 플립 오류를 주입하여 오류가 주입된 결과값 $\tilde{Q}_{c_{l-1}}$을 얻는다. 이 때, c_l를 0이라고 가정하면 〔표 3〕과 같이 c_{l-1}에 오류가 주입된 결과값은 $\tilde{Q}_{c_{l-1},c_l=0}=3U_{1,c_l}-5U_{2,c_l}$이다. $\tilde{Q}_{c_{l-1}}$와 $\tilde{Q}_{c_{l-1},c_l=0}$가 일치하는 경우, $c_l=0$이며 $(U_{1,c_{l-1}}, U_{2,c_{l-1}})=(U_{1,c_l}-U_{2,c_l}, U_{2,c_l})$이다. 일치하지 않는 경우, $c_l=1$이며 $(U_{1,c_{l-1}}, U_{2,c_{l-1}})=(U_{2,c_l}, U_{1,c_l}-U_{2,c_l})$이다.

위와 같이, c_l와 c_{l-1}에 오류를 주입하여 c_l의 비트값과 c_{l-1}에 대한 중간값 $(U_{1,c_{l-1}}, U_{2,c_{l-1}})$를 알아낸다.

〔표 2〕 c_l에 오류를 주입한 경우

c	U_1	U_2
⋮	⋮	⋮
c_l	U_{1,c_l}	U_{2,c_l}
Q	$U_{1,c_l}+U_{2,c_l}$	
↓		
c	U_1	U_2
⋮	⋮	⋮
$\tilde{c}_l$	U_{1,c_l}	$U_{1,c_l}-U_{2,c_l}$
$\tilde{Q}_{c_l}$	$2U_{1,c_l}-U_{2,c_l}$	

〔표 3〕 c_{l-1}에 오류를 주입한 경우($c_l=0$으로 가정)

c	U_1	U_2
⋮	⋮	⋮
c_{l-1}	$U_{1,c_l}-U_{2,c_l}$	U_{2,c_l}
$c_l=0$	U_{1,c_l}	U_{2,c_l}
Q	$U_{1,c_l}+U_{2,c_l}$	
↓		
c	U_1	U_2
⋮	⋮	⋮
$\tilde{c}_{l-1}$	$U_{1,c_l}-U_{2,c_l}$	$U_{1,c_l}-2U_{2,c_l}$
$c_l=0$	$2U_{1,c_l}-3U_{2,c_l}$	$U_{1,c_l}-2U_{2,c_l}$
$\tilde{Q}_{c_{l-1},c_l=0}$	$3U_{1,c_l}-5U_{2,c_l}$	

공격의 효율성을 높이기 위하여 위 방법을 일반화한 c_i에 대한 공격 방법을 설명한다. $c''=(c_{i+2},...,c_{l-1},c_l)$와 c_{i+1}에 대한 중간값 $(U_{1,c_{i+1}}, U_{2,c_{i+1}})$을 알고 있다고 가정한다. 임의의 (U_1, U_2)에 대하여 EAC $c'=(c_{i+1},...,c_{l-1},c_l)$를 사용하는 타원곡선 스칼라 곱셈 알고리즘은 〔그림 3〕과 같다.

입력 : (U_{1,c_i}, U_{2,c_i}), 상수 k에 대한 EAC $c' = (c_{i+1}, ..., c_{l-1}, c_l)$
출력 : $Q = kP$

1. $(U_1, U_2) \leftarrow (U_{1,c_i}, U_{2,c_i})$
2. for $i = i+1$ to l
3. if $c_i = 1$ then
4. $(U_1, U_2) \leftarrow (U_1 + U_2, U_1)$
5. else
6. $(U_1, U_2) \leftarrow (U_1 + U_2, U_2)$
7. end for
8. return $Q \leftarrow U_1 + U_2$

〔그림 3〕 EAC $c' = (c_{i+1}, ..., c_{l-1}, c_l)$를 사용하는 타원곡선 스칼라 곱셈 알고리즘

〔그림 3〕을 이용한 공격 방법은 다음과 같다.

단계 1. c_i에 비트 플립 오류를 주입하여 $\tilde{Q}_{c_i}$을 얻는다.

단계 2. c_{i+1}를 0으로 가정하면, c_i에 대한 중간값은 $(U_{1,c_i}, U_{2,c_i}) = (U_{1,c_{i+1}} - U_{2,c_{i+1}}, U_{2,c_{i+1}})$이다. c_i에 비트 플립 오류가 주입되었다고 가정하면, $\tilde{c}_i$에 대한 중간값은 $(U_{1,c_i}, U_{2,c_i}) = (U_{1,c_{i+1}} - U_{2,c_{i+1}}, U_{1,c_{i+1}} - 2U_{2,c_{i+1}})$이다. $c' = (c_{i+1}, ..., c_{l-1}, c_l)$과 $(U_{1,c_i}, U_{2,c_i}) = (U_{1,c_{i+1}} - U_{2,c_{i+1}}, U_{1,c_{i+1}} - 2U_{2,c_{i+1}})$를 이용하여 〔그림 3〕을 수행한 뒤, 오류가 주입된 결과값 $\tilde{Q}_{c_i, c_{i+1}=0}$을 얻는다.

단계 3. $\tilde{Q}_{c_i}$와 $\tilde{Q}_{c_i, c_{i+1}=0}$가 일치하는 경우, $c_{i+1} = 0$이며 $(U_{1,c_i}, U_{2,c_i}) = (U_{1,c_{i+1}} - U_{2,c_{i+1}}, U_{2,c_{i+1}})$이다. 일치하지 않는 경우, $c_{i+1} = 1$이며 $(U_{1,c_i}, U_{2,c_i}) = (U_{2,c_{i+1}}, U_{1,c_{i+1}} - U_{2,c_{i+1}})$이다.

단계 4. 단계 1 ~ 단계 3을 반복한다.

즉, EAC인 $c = (c_4, ..., c_{l-1}, c_l)$의 최하위 c_l부터 최상위 c_4까지 순차적으로 오류를 주입하여 모든 $c = (c_4, ..., c_{l-1}, c_l)$를 알아낼 수 있다. 또한 알아낸 $c = (c_4, ..., c_{l-1}, c_l)$를 이용하여 상수 k를 복원할 수 있다. 제안하는 공격 방법은 $c = (c_4, ..., c_{l-1}, c_l)$에 비트 플립 오류를 주입한다. 그 결과, EAC를 사용하는 타원곡선 스칼라 곱셈 알고리즘은 $\tilde{c} = (c_4, ..., \tilde{c}_i, ..., c_l)$에 대한 타원곡선 스칼라 곱셈 연산을 하게 된다. $c_i = 1$이라고 가정하자. 이 때, 수식 (1)의 $U_2(i)$는 $U_1(i-1)$이고, 수식 (2)의 $U_C(i)$는 $U_2(i-1)$이며 수식 (3)의 $U_D(i+1)$는 $U_2(i-1)$이다. c_i에 비트 플립 오류가 주입되어 $\tilde{c}_i$이 되면, $U_2(i)$값은 $U_2(i-1)$, $U_C(i)$는 $U_1(i-1)$, $U_D(i+1)$은 $U_1(i-1)$로 변경된다. 즉, 오류가 주입된 경우에도 $U_D(i+1)$와 $U_C(i)$는 같은 값을 가지기 때문에 S. Pontarelli가 제시한 오류 탐지 방법으로 오류를 탐지할 수 없다. 마찬가지로 $c_i = 0$인 경우에도 오류 탐지가 불가능하다.

IV. 제안하는 공격에 대한 대응방법

타원곡선 스칼라 곱셈 연산에 대한 오류 주입 공격을 막기 위하여 다양한 오류 탐지 방법들이 제시되고 있다. 일반적으로 결과값이 타원곡선상의 점인지 확인하는 방법을 가장 많이 사용한다〔9〕. 이외에도 타원곡선 스칼라 곱셈 연산을 반복 수행하여 두 결과값을 비교하거나, 타원곡선 위의 점 P 또는 상수 k를 랜덤하게 하는 방법 등이 있다〔16〕.

제안하는 공격에 대한 대응방법으로 상수 j를 이용하여 타원곡선 스칼라 곱셈 알고리즘의 상수 k를 랜덤하게 하는 방법을 사용할 수 있다. 이 방법은 〔18〕에서 처음 소개된 방법으로 $k'' = k + j(\#E)$를 만족하는 랜덤한 k''에 대하여 타원곡선 스칼라 곱셈 연산을 수행하는 것이다. 타원곡선 위의 점 P의 위수(order) n은 $nP = O$를 만족하는 가장 작은 양의 정수이며, n은 타원곡선의 위수 $\#E$를 나눈다. 따라서 $(\#E)P = O$이고 $k''P = (k + j(\#E))P \equiv kP$를 만족한다. 제안하는 공격 방법은 EAC를 사용하는 타원곡선 스칼라 곱셈 알고리즘에 반복적으로 비트 플립 오류를 주입한다. 또한 $c'' = (c_{i+2}, ..., c_{l-1}, c_l)$과 c_{i+1}에 대한 중간값 $(U_{1,c_{i+1}}, U_{2,c_{i+1}})$을 알고 있다는 가정하에 c_{i+1}를 찾는다. 그러나 오류를 주입할 때마다 매번 다른 k''에 대한 EAC를 사용하는 경우, c'' 또한 매번 변경되어 제안하는 공격 방법을 적용할 수 없다. 따라서 제안하는 오류 주입 공격에 안전하다.

V. 결론

본 논문에서는 EAC를 이용한 타원곡선 스칼라 곱셈 연산에 대한 새로운 오류 주입 공격을 제안하였다. 제안하는 방법은 c_i에 오류가 주입된 결과값 $\tilde{Q}_{c_i}$와 c_{i+1}의 값을 0으로 가정한 뒤 얻은 결과값 $\tilde{Q}_{c_i, c_{i+1}=0}$을 비교하여 c_{i+1}를 알아낸다. 이러한 비트 플립 오류를

이용하여 EAC를 사용하는 타원곡선 스칼라 곱셈 알고리즘의 EAC $c=(c_4,...,c_{l-1},c_l)$를 모두 알아낸다. 또한 알아낸 EAC $c=(c_4,...,c_{l-1},c_l)$를 이용하여 타원곡선 스칼라 곱셈 알고리즘의 상수 k를 알 수 있다. 제안하는 공격 방법은 기존에 제안된 S. Pontarelli의 오류 탐지 방법으로는 오류 주입 여부를 판단할 수 없으며, 제안하는 오류 주입 공격에 안전하게 하기 위해 상수 k를 랜덤하게 하는 방법을 사용할 수 있다.

참고문헌

[1] N. Koblitz. "Elliptic curve cryptosystems," Mathematics of Computation, Vol. 48, no.177, pp. 203 - 209, Jan. 1987.

[2] V.S. Miller, "Use of elliptic curves in cryptography," CRYPTO'85, LNCS 218, pp. 417-426, 1986.

[3] N. Meloni, "New point addition formulae for ECC Applications," WAIFI'07, LNCS 4547, pp. 189-201. Jun. 2007.

[4] A. Byrne, F. Crowe, W.P. Marnane, N. Meloni, A. Tisserand, and E. Popovici, "SPA resistant elliptic curve cryptosystem using addition chains," International Journal of High Performance Systems Architecture, Vol. 1, no.2, pp. 133 - 142, Oct. 2007.

[5] R.R. Goundar, M. Joye and A. Miyaji, "Co-Z addition formulæ and binary ladders on elliptic curves," In: Mangard, S., Standaert, F-X. (eds.) CHES'10. LNCS 6225, pp. 65 - 79. Aug. 2010.

[6] J. López and R. Dahab, "Fast multiplication on elliptic curves over GF(2m) without precomputation," CHES'99, LNCS 1717, pp. 316-327, Aug. 1999.

[7] T. Izu and T. Takagi, "A fast parallel elliptic curve multiplication resistant against side channel attacks," PKC'02, LNCS 2274, pp. 280-296, Feb. 2002.

[8] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other Systems," Advances in Cryptology, Proc. CRYPTO'96, pp. 104-113, Aug. 1996.

[9] D. Boneh, R.A. DeMillo and R.J. Lipton, "On the importance of eliminating errors in cryptographic computations," Journal of Cryptology, 2001; Extended abstract in Proc. EUROCRYPT'97, Vol. 14, no. 2, pp. 110-119, May. 1997.

[10] F. Bao, R.H. Deng, Y. Han, A. Jeng, A.D. Narasimbalu and T. Ngair, "Breaking Public Key Cryptosystems on Tamper Resistant Devices in the Presence of Transient Faults," Security Protocols Workshop 1997, LNCS 1361, pp. 115-124, Apl. 1997.

[11] J. Schmidt and C. Herbst, "A Practical Fault attack on Square and Multiply," FDTC'08, IEEE Computer Society, pp. 53-58, Aug. 2008.

[12] P. Fouque and R. Lercier, "Fault Attack on Elliptic Curve with Montgomery Ladder Implementation," FDTC'08, IEEE Computer Society, pp. 92-98, Aug. 2008.

[13] I. Biehl, B. Meyer, and V. Müller. "Differential fault attacks on elliptic curve cryptosystems," CRYPTO'00, LNCS 1880, pp. 131-146, Aug. 2000.

[14] M. Ciet, M. Joye, "Elliptic curve cryptosystems in the presence of permanent and transient faults," Cryptology ePrint Archive, Vol. 36, no. 1, pp. 33-43, Jul. 2005.

[15] J. Blömer, M. Otto, and J. Seifert, "Sign change fault attacks on elliptic curve cryptosystems," FDTC'06, LNCS 4236, pp 36-52, Oct. 2006.

[16] A. Dominguez-Oviedo and M. Anwar Hasan, "Error Detection and Fault Tolerance in ECSM Using Input Randomization," IEEE Trans. Dependable and Secure Computing Vol. 6, no.3, pp. 175-187, Jul. 2009.

[17] S. Pontarelli, G.C. Cardarilli, M. Re, and A. Salsano. "Error detection in addition

chain based ecc point multiplication," IEEE International On-Line Testing Symposium, pp. 192 - 194, Jun. 2009.
〔18〕 J. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems," Proc. CHES'99, LNCS 1717, pp. 292-302, Aug. 1999.

〈著者紹介〉

이 수 정 (Soo Jeong Lee) 학생회원
2011년 2월: 성신여자대학교 수학과 학사 졸업
2011년 2월~현재: 고려대학교 정보보호대학원 정보보호학과 석사과정
〈관심분야〉 부채널 공격, 공개키 암호 알고리즘

조 성 민 (Sung Min Cho) 학생회원
2008년 2월: 광운대학교 수학과 학사 졸업
2011년 8월: 고려대학교 정보경영공학전문대학원 석사 졸업
2011년 8월~현재: 고려대학교 정보보호대학원 정보보호학과 박사과정
〈관심분야〉 부채널 공격, 공개키 암호 알고리즘, 암호구현

홍 석 희 (Seokhie Hong) 종신회원
1995년 2월: 고려대학교 수학과 학사 졸업
1997년 2월: 고려대학교 수학과 석사 졸업
2001년 8월: 고려대학교 수학과 박사 졸업
1999년 8월~2004년 2월: (주)시큐리티 테크놀로지스 선임연구원
2003년 8월~2004년 2월: 고려대학교 정보보호기술연구센터 선임연구원
2004년 4월~2005년 2월: K.U.Leuven. ESAT/SCD-COSIC 박사후연구원
2005년 3월~현재: 고려대학교 정보보호대학원 부교수
〈관심분야〉 대칭키·공개키 암호 분석 및 설계, 컴퓨터 포렌식