타원곡선 암호시스템에서 Randomized Folding 기법에 대한 MESD 공격 실험

정지은*, 김창균*, 이훈재**, 문상재*

*경북대학교 전자공학과, **동서대학교 인터넷공학부

An Experiment in MESD Attacks on Scalar Multiplication Using a Randomized Folding Scheme for ECC

JiEun Jeong*, ChangKyun Kim*, HoonJae Lee**, SangJae Moon*

*Department of Electronics Engineering Kyungpook National Univ.

**School of Internet Engineering, Dongseo Univ.

요 약

최근 스마트카드에 대한 수요가 증가함에 따라 스마트카드에 대한 보안상의 문제가 대두되고 있다. 스마트카드의 안전성을 위협하는 공격 방법 중 전력공격은 가장 강력한 공격으로 많은 연구가 되고 있다. 본 논문에서는 전력분석 공격의 한 방법인 MESD 공격에 대해 알아보고 이에 대한 대응 방법으로 제안된 RSM 알고리듬에 대해 알아본다. 또한 이 알고리듬의 연산 속도 향상을 위해 folding 기법을 사용한 알고리듬에 대해서도 알아보고 MESD 공격에 안전한지를 실험을 통해 검증한다.

I. 서론

최근 여러 가지 장점으로 인해 스마트카드에 대한 수요가 급증하고 있다. 이로 인해 많은 암호학자들이 스마트카드를 공격하기 위해 오류 공격(fault attack), 시차 공격(timing attack), 그리고 전력분석 공격(power analysis attack)과 같은 물리적 공격 방법들을 제안하고 있다[1-4]. 물리적인 공격에서 스마트카드의 수행 시간, 소비 전력, 오류에 의한 정보 등과 같은 부채널 정보를 이용하는 것을 부채널 공격이라 한다.

부채널 공격을 분류하면 크게 능동적인 공격과 수동적인 공격으로 나눌 수 있다. 능동적인 공격에는 암호시스템에서 연산이 수행될 때 오류를 주입하는 오류 공격이 있다. 1996년에 Bellcore 사에서 처음으로 오류 공격에 대해 발표하였다[1]. 수동적인 공격에는 시차 공격과 전력분석 공격이 있다. 시차 공격이란 부채널로 유출되는 정보 중 암호 연산의 수행시간을 이용하는 것이고, 전력분석 공격이란 암호 연산을 수행할 때 소모되는 전력을 분석하는 것으로 P. Kocher에 의해 1996년과 1998년에 각각 제안되었다[3, 4].

이로 인해, 현재까지 전력분석 공격의 대응방법으로 많은 논문이 발표되었다. 그 중 비밀키를 랜덤하게 선택함으로써 전력분석 공격의 한 방법인 MESD(Multiple-Exponent, Single-Data) 공격을 어렵게 하는 RSM(Randomized Signed-scalar Multiplication) 알고리듬을 들 수 있다[5]. 그리고 논문[6]에서는 이 알고리듬의 연산 속도를 향상시키기 위해 folding 기법을 이용하였다. 본 논문에서는 randomized folding 기법을 이용한 알고리듬이 MESD 공격에 안전한지를 실험을 통해 검증하고자 한다.

II. 타원곡선 암호시스템

1. 타원곡선 암호시스템의 개요

Affine 좌표계에서 유한체 상에서 정의된 타원곡선 E는 Weierstrass 방정식을 만족하는 점(x, y)의 집합이다. Weierstrass 방정식은 다음과 같이 표현된다.

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

타원곡선 E는 위의 등식을 만족하는 점(x, y)과 무한원점 O로 구성되며, 무한원점은 타원곡선 상의 점들에 대한 연산을 수행할 때 덧셈에 대한 항등원으로 작용한다. 이와 같이 타원곡선 상의 점들의 집합과 무한원점은 덧셈에 대한 가환군(commutitive group)을 이룬다.

2. 타원곡선 상에서의 연산

타원곡선 상의 점 $P=(x_0, y_0)$와 $Q=(x_1, y_1)$를 덧셈 연산한 결과로 주어지는 점 $R=(x_2, y_2)$는 다음과 같이 표현된다.

$$R=(x_2, y_2)=(x_0, y_0)+(x_1, y_1)$$

$$(P \neq Q)$$

$$\lambda = \frac{y_1 - y_0}{x_1 - x_0}$$

$$x_2 = \lambda^2 - x_0 - x_1$$

$$y_2 = \lambda(x_0 - x_2) - y_0$$

또한 타원곡선 상의 점 $P=(x_0, y_0)$와 $Q=(x_1, y_1)$를 두배 연산한 결과로 주어지는 점 $R=(x_2, y_2)$는 다음과 같이 표현된다.

$$R=(x_2, y_2)=2(x_0, y_0) \quad (P=Q)$$

$$\lambda = \frac{3x_0^2 + a}{2y_0}$$

$$x_2 = \lambda^2 - 2x_0$$

$$y_2 = \lambda(x_0 - x_2) - y_0$$

III. 타원곡선 암호시스템에서의 Scalar Folding 기법

1. NAF 알고리듬

타원곡선 암호시스템에서 비밀키 0과 1을 수행할 때의 과정은 서로 다르다. 다시 말해서 비밀키 0을 수행할 때에는 두배 연산만 수행하는데 반해 비밀키 1을 수행할 때에는 두배 연산과 덧셈 연산을 수행하게 된다. 그러므로 비밀키에 1의 개수가 적을수록 연산 시간이 적게 소요되므로 연산 속도가 빠르다. 이러한 생각에서 출발한 것이 NAF(Non-Adjacent Form)이다[7]. 결과는 같지만 알고리듬을 수행하는 과정에서 0이 아닌 수가 연속되지 않도록 만드는 것이다.

2. SPA에 대응하는 덧셈-뺄셈 알고리듬

타원곡선 암호시스템의 경우는 구현 시 피연산자의 해밍웨이트에 따라 전력 소모가 다르기 때문에 두배 연산이나 덧셈 연산과 같은 비밀키 연산이 이루어질 때 비밀키의 각 비트에 대해 0과 1 그리고 -1에 따른 구현의 명령을 같게 하여 그림 1과 같이 동일한 연산 시간을 가질 수 있도록 알고리듬을 수정하여 수행한다. 연산량은 증가하지만 SPA 공격에 대응할 수 있다는 장점이 있다.

INPUT : P, $di=(d_{L-1},...,d_0)_2$
OUTPUT : $Q[0]=dP$
1. $Q[0] \leftarrow 0$
2. for i from $L-1$ downto 0 do
    $Q[0] \leftarrow 2Q[0]$
    $Q[1] \leftarrow Q[0] + P$
    $Q[-1] \leftarrow Q[0] - P$
    $Q[0] \leftarrow Q[d_i]$
3. Return $Q[0]$

그림 1: SPA에 대응하는 덧셈-뺄셈 알고리듬.

3. RSM 알고리듬[5]

NAF 레코딩 알고리듬에서 사용했던 보조 carry c_{i+1}을 RSM 알고리듬에서도 사용한다. c_{i+1}는 (i+1)번째 carry를 의미하고 0번째 carry를 뜻하는 c_0는 0으로 둔다. 또한 d_i는 i번째 NAF 값을 의미한다. 그러므로 연속한 $c_{i+1}\ d_i$은 $c_{i+1}\ 2^1 + d_i\ 2^0$을 의미한다. 이러한 이유로 $c_{i+1}\ d_i$=01이라고 표현된 것은 다르게(예를 들어, $c_{i+1}\ d_i$=1-1) 표현될 수도 있고 역도 성립한다.

NAF 레코딩 알고리듬에 랜덤성을 추가하기 위해 n 비트 정수인 랜덤수 $r=(r_{n-1}\ r_{n-2} \cdots r_0)$을 생성한다. RSM 알고리듬에서 d_i값과 보조 carry c_{i+1}의 값은 표 1에 나타난 것과 같이 n 비트 정수인 랜덤수

r에 따라 랜덤하게 생성된다.

표 1: 랜덤하게 부호화된 레코딩 방법.

입력				출력		
k_{i+1}	k_i	c_i	r_i	c_{i+1}	d_i	비고
0	0	0	0	0	0	NAF
0	0	0	1	0	0	NAF
0	0	1	0	0	1	NAF
0	0	1	1	1	-1	AF
0	1	0	0	0	1	NAF
0	1	0	1	1	-1	AF
0	1	1	0	1	0	NAF
0	1	1	1	1	0	NAF
1	0	0	0	0	0	NAF
1	0	0	1	0	0	NAF
1	0	1	0	1	-1	NAF
1	0	1	1	0	1	AF
1	1	0	0	1	-1	NAF
1	1	0	1	0	1	AF
1	1	1	0	1	0	NAF
1	1	1	1	1	0	NAF

4. 고속화를 위한 Folding 기법[6]

이제 RSM 알고리듬의 연산 속도를 증가시키기 위하여 제안된 folding 기법에 대해 알아보고자 한다.

비밀키 k를 n비트라고 했을 경우 표 1에 의해 생성되는 재부호화 된 비밀키 d의 비트수는 n 또는 $n+1$이 된다. 랜덤수 r에 따라 달라지는 비밀키 d를 다음과 같이 표현할 수 있다.

$$d = d_n 2^n + d_{n-1} 2^{n-1} + \cdots$$

$$\cdots + d_1 2^1 + d_0 2^0$$

여기서 $d_i \in \{-1, 0, 1\}$이다. RSM 알고리듬을 통해 재부호화 된 비밀키 d의 값은 다음과 같은 고속 스칼라 곱셈 방법에 의해 한번 더 재부호화 된다.

$$d = 2^h(e_{h-1} 2^{h-1} + e_{h-2} 2^{h-2} + \cdots$$

$$+ e_0 2^0) + (f_{h-1} 2^{h-1} + f_{h-2} 2^{h-2} +$$

$$\cdots + f_0 2^0)$$

$$= \sum_{i=0}^{h-1} 2^i (e_i 2^h + f_i)$$

여기서 $e_i, f_i \in \{-1, 0, 1\}$이고,

$h = \lceil (n+1)/2 \rceil$ 이다.

비밀키 d를 상위와 하위로 분할할 다음, 다음과 같이 재부호화 한다.

$g_i' = e_i 3^1 + f_i 3^0$, $g_i' \in \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$

$g_i = | g_i' |$, $g_i \in \{0, 1, 2, 3, 4\}$

INPUT : P, $d_i = (d_{L-1}, \ldots, d_0)_2$
OUTPUT : $Q[0] = dP$
==== Precomputation Phase($P[3] = 2^h P$) ====
1. $P[3] = P$
2. for $i = h-1$ to 0 by -1 do {
3. $P[3] = 2P[3]$ }
========== Evaluation Phase ==========
1. $P[0] = P$, $P[1] = P$, $P[2] = P[3] - P[1]$, $P[4] = P[3] + P[1]$
2. for $i = h-1$ to 0 by -1 do {
$Q[0] = 2Q[0]$
$R[0] = R[1] = P[g_i]$
$R[-1] = -P[g_i]$
$Q[1] = Q[-1] = Q[0] + R[s_i]$
$Q[0] = Q[s_i]$}
3. Return $Q[0]$

그림 2: Folding 기법을 이용한 스칼라 곱셈.

또한, g_i'=0이면 s_i=0으로, g_i'>0이면 s_i=1로, g_i'<0이면 s_i=-1로 둔다. 이런 방법으로 연산에 필요한 배열의 위치를 나타낼 수 있다.

비밀키를 부호화하여 g_i와 s_i가 구해지면 이를 통해 $Q = dP$를 계산할 수 있다. 알고리듬은 그림 2와 같다. 먼저 고정된 한 점 P와 비밀키 k를 이용하여 $2^h P$를 미리 계산해서 $P[3]$에 저장하고 연산에 필요한 정보를 각 배열에 저장한다. 단계

3에서 g_i와 s_i에 따라 스칼라 곱셈을 실시하는데 총 h번의 loop를 반복하게 된다.

IV. Randomized Folding 기법에 대한 MESD 공격 실험

1. NAF를 이용한 덧셈-뺄셈 알고리듬에 대한 MESD 공격 실험

NAF를 이용한 덧셈-뺄셈 알고리듬에서는 첫 번째 비트와 두 번째 비트는 10이므로 세 번째 비트부터 공격하면 된다. 또한 비밀키 비트가 1 또는 -1이라는 것을 알게 되면 다음 비트는 0이라는 것을 알 수 있다.

실제 스마트카드의 비밀키가 1001010-101이라고 했을 때, 세 번째 비트인 0을 공격한다고 하자. 세 번째 비트를 모르는 상황에서 공격자는 0, 1 그리고 -1로 추측할 수 있다. 만약 0으로 추측을 했다면 올바르게 추측한 것이므로 피크(peak)가 발생하지 않을 것이다. 공격자는 실험 결과 피크가 발생하지 않았다면 세 번째 비트를 공격하는데 성공했다고 생각할 수 있다. 그리고 세 가지 경우 중 두 번을 실험했을 때, 두 가지 경우에서 모두 피크가 발생하면 실제 비밀키는 추측하지 않았던 비트라고 생각할 수 있다. 세 번째 비트를 추측할 때 연산되는 값들을 살펴보면 표 2와 같다. 표 2에서는 비밀키 중 상위 다섯 비트만을 고려하였다.

공격자는 두 번째 비트가 0임을 알고 있으므로 세 번째 비트부터 추측한다. 실제로 세 번째 비트에서 연산되는 값은 $4P$가 될 것이다. 그러나 세 번째 비트를 1로 추측한 경우에는 $5P$, -1로 추측한 경우에는 $3P$가 될 것이다. 또한 실제로 네 번째 비트에서 연산되는 값은 $9P$가 될 것이다. 그러나 세 번째 비트를 1로 추측한 경우에 네 번째 비트에서 연산되는 값은 $10P$, $11P$ 혹은 $9P$가 될 것이다. 그리고 세 번째 비트를 -1로 추측했다면 네 번째 비트에서 연산되는 값은 $6P$이거나 $7P$ 혹은 $5P$가 될 것이다. 그러므로 실제 비밀키인 0이 수행될 때 소모되는 전력과 1이나 -1로 추측했을 때 소모되는 전력을 차분하면 세 번째 비트 위치부터 피크가 발생하는 것을 관찰할 수 있다.

그림 3은 스마트카드의 비밀키인 0의 평균 전력 파형과 비밀키를 1과 -1로 추측한 평균 전력 파형을 차분한 것이다. 측정 회수는 각각 200회이고 표본화율(sampling/sec)은 초당 10^6개이다. 두 가지 경우 모두 추측이 틀린 경우이므로 피크가 발생하는 것을 확인할 수 있다.

표 2: 세 번째 비트에서 연산될 데이터.

실제 비밀키	1	0	0	1	0	…
두배 연산		$2P$	$4P$	$8P$	$18P$	…
덧셈 연산				$9P$		…
추측한 비밀키	1	0	1	X	X	…
두배 연산		$2P$	$4P$	$10P$	·	…
덧셈 연산			$5P$	$11P$	·	…
뺄셈 연산				$9P$	·	…
추측한 비밀키	1	0	-1	X	X	…
두배 연산		$2P$	$4P$	$6P$	·	…
덧셈 연산				$7P$	·	…
뺄셈 연산			$3P$	$5P$	·	…

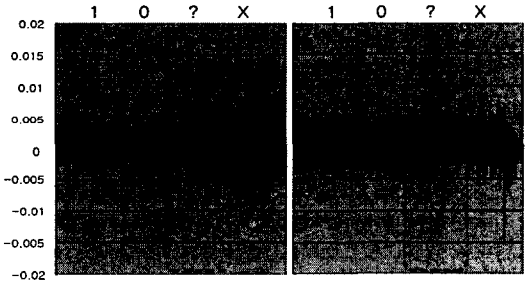

그림 3: 두 가지 추측에 대한 차분 전력 파형.

2. Randomized Folding 기법을 적용한 알고리듬에 대한 MESD 공격 실험

기본적인 설정은 NAF를 이용한 덧셈-뺄셈 알고리듬에 대한 MESD 공격 실험을 할 때와 같게 한다. 스마트카드의 비밀키가 k=1011011일 때 랜덤수 r에 따라 나타날 수 있는 재부호화 된 비밀키 d는 여러 가지가 있을 수 있다. 이렇듯 비밀키가 랜덤하게 변화하기 때문에 MESD 공격으로 비밀키를 찾기가 쉽지 않다[8, 9].

본 논문에서는 랜덤수 r이 1100101일 때 실제 실험을 통해 이를 검증하고자 한다. 공격자는 비밀키의 비트수와 연산에 사용되는 알고리듬을 알고 있다고 가정한다. 또한 첫 번째 비트 1은 공격

자가 알고 있는 값이다. 비밀키의 비트수 n의 값이 7이므로 h는 4가 된다. 비밀키를 반으로 접으면(folding) 나올 수 있는 가능한 쌍은 (0, 0), (0, 1), (0, -1), (1, 0), (1, 1), (1, -1), (-1, 0), (-1, 1), (-1, -1)과 같이 9개의 쌍이 있을 수 있다. 이 경우 정확한 첫 번째 쌍은 (1, -1)이다. (1, -1)이라고 추측한 경우를 제외하면 실제 비밀키와 다른 연산을 수행할 것이므로 피크가 발생해야 하고, (1, -1)이라고 추측한 경우에는 피크가 발생하지 않아야 공격이 성공한 것이 된다. 본 논문에서는 첫 번째 쌍을 (1, -1)로 올바르게 추측한 경우를 실험하였다. 그림 4는 올바르게 추측한 경우의 차분 파형을 보여준다. 그림 4에서 보는 바와 같이 올바르게 추측했지만 전 구간에서 걸쳐 피크가 발생했다. 이러한 실험 결과로 공격자가 실제 비밀키의 첫 번째 쌍이 (1, -1)이라는 것을 알아내기는 힘들다.

이 실험을 통해 비밀키를 랜덤하게 변화시킨 후 folding 기법을 적용한 알고리듬은 MESD 공격에 대한 대응방법으로 적절함을 알 수 있다.

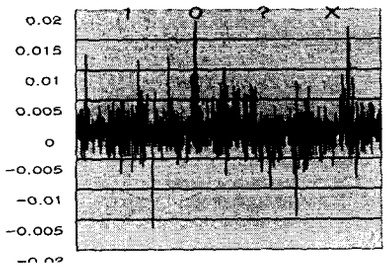

그림 4: Randomized scalar folding 알고리듬에 대한 차분 파형.

V. 결론

본 논문에서는 스마트카드의 MESD 공격에 대응하는 방법 중 하나인 RSM 알고리듬과 이 알고리듬의 연산 속도 향상을 위해 folding 기법을 사용한 알고리듬에 대해 살펴보았다. 그리고 실제 MESD 공격 실험을 통하여 NAF를 사용한 타원곡선 암호 알고리듬과 randomized folding 기법을 사용한 알고리듬을 공격하였다. 실험을 통하여 MESD 공격에 대한 randomized folding 기법을 사용한 알고리듬의 안전성을 확인할 수 있었다.

참고문헌

[1] Bellcore Press Release, "New threat model breaks crypto codes," Sept. 1996.

[2] D. Boneh, R.A. DeMillo, and R.J. Lipton, "On the importance of checking cryptographic protocols for faults," In Advances in Cryptology - EUROCRYPT '97, LNCS 1233, PP. 37-51, Springer-Verlag, 1997.

[3] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in Proceedings of Advances in Cryptology-CRYPTO '96, pp. 104-113, Springer-Verlag, 1996.

[4] P. Kocher, J. Jaffe, and B. Jun, "Introduction to Differential Power Analysis and Related Attacks," http://www.cryptography.com/dpa/technical, 1998.

[5] J. C. Ha and S. J. Moon, "Randomized signed-scalar multiplication of ECC to resist power attacks," in Proceedings of Workshop on Cryptographic Hardware and Embedded Systems- CHES 2002, pp. 553-565, Springer-Verlag, 2002.

[6] 하재철, 곽동진, 문상재, "Folding 기법을 이용한 전력분석 공격에 대응하는 고속 스칼라 곱셈," 한국정보보호학회지 (KIISC - Korea Institute of Information Security & Cryptology), vol. 13, no. 3, 2003.

[7] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, "Handbook of applied crkyptography," CRC Press, 1997.

[8] J. S. Coron, "Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems," in Workshop on Cryptographic Hardware and Embedded Systems, pp. 292-302, Springer-Verlag, 1999.

[9] M.A. Hansan, "Power analysis attacks and algorithm approaches to their countermeasure for Koblitz curve crypto-system," in Proceedings of Workshop on Cryptographic Hardware and Embedded Systems, Springer-Verlag, 2000.