情報保護學會論文誌
第17卷 第2號, 2007. 4

sABS 형태의 스칼라 곱셈 연산에 대한 새로운 단순전력 공격*

김 희 석,[1†] 한 동 국,[2‡] 김 성 경,[1] 김 태 현,[1] 박 영 호,[3] 임 종 인[1]
[1]고려대학교 정보경영공학전문대학원, [2]한국전자통신연구원, [3]세종사이버대학교

New Simple Power Analysis on scalar multiplication based on sABS recoding

HeeSeok Kim,[1†] Dong-Guk Han,[2‡] Sung-kyoung Kim,[1] Tae Hyun Kim,[1] Young-Ho Park,[3] Jongin Lim[1]
[1]Graduate School of Information Management and Security, Korea University,
[2]Electronics and Telecommunications Research Institute, [3]Sejong cyber University

요 약

스마트카드와 같이 계산 능력이나 메모리가 제한된 장치에 암호 시스템을 구현할 때, 장치 내에 내장되어 있는 부채널 공격을 고려한 암호학적인 알고리즘은 적은 메모리를 이용하여 효율적으로 수행되어야 한다. 스칼라 곱셈 연산은 타원곡선 암호시스템에서 중요하게 다뤄지는 연산이기 때문에 부채널 공격에 안전하게 구성되어야만 한다. 하지만 부채널 공격에 안전하다고 제시된 여러 대응방법조차도 때로는 고려되지 않은 분석법에 의해 그 취약점이 드러나곤 한다. SPA에 취약하지 않다고 알려진 더미 연산을 추가한 스칼라 곱셈 연산 알고리즘은 Doubling Attack에 의해 그 취약점이 드러났다. 그러나 스칼라 곱셈의 부채널 공격 대응 방법 중 하나인 Hedabou에 의해 제안된 sABS 방법은 Doubling attack이 적용되지 않는다. 본 논문에서는 기존의 Doubling attack을 활용하여 sABS 방법을 분석할 수 있는 새로운 강화된 Doubling attack을 제안하고, 실험적인 결과를 통해 자세한 공격 방법을 소개한다.

ABSTRACT

In cryptographic devices like a smart-card whose computing ability and memory are limited, cryptographic algorithms should be performed efficiently. Scalar multiplication is very important operation in Elliptic Curve Cryptosystems, and so must be constructed in safety against side channel attack(SCA). But several countermeasures proposed against SCA are exposed weaknesses by new un-dreamed analysis. `Double-and-add always scalar multiplication' algorithm adding dummy operation being known to secure against SPA is exposed weakness by Doubling Attack. But Doubling Attack cannot apply to sABS recoding proposed by Hedabou, that is another countermeasure against SPA. Our paper proposes new strengthened Doubling Attacks that can break sABS recoding SPA-countermeasure and a detailed method of our attacks through experimental result.

Keywords : *부채널 공격, sABS 리코딩, 더블링 어택, 스칼라 곱셈*

접수일: 2006년 10월 30일; 채택일: 2007년 2월 5일
1) "본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음" (IITA-2006-(C1090-0603-0025))
† 주저자, 교신저자 heeseokkim@cist.korea.ac.kr
‡ 교신저자, christa@etri.re.kr

Ⅰ. 서 론

지금까지의 많은 암호 시스템 설계자들은 이산대수 문제나 소인수분해 문제와 같은 이론적으로 안전한 알고리즘을 제안해 왔다. 하지만, 수학적으로 안전한 이러한 알고리즘조차도 구현 단계에서 고려되지 못한 부가적인 정보의 누출이 있다는 것이 알려졌고, 이로부터 비밀 키의 값을 알아낼 수 있는 부채널 공격(Side channel attack)이 소개되었다[5]. 알려진 부채널 공격에는 결함 주입 공격(fault insertion attack)[1,11], 시간 공격(timing attack), 전력분석 공격(power analysis attack)[6,7,8], 그리고 전자기 누출 공격(electromagnetic emission attack)등이 있다. 전력분석 공격에는 스마트카드에 물리적 변환을 가하지 않고 단순히 전력신호를 관찰함으로써 사용된 비밀 키의 값을 알아내는 SPA(Simple Power Analysis)방법과 다수의 전력신호를 통계적으로 분석해 비밀 키의 값을 알아내는 DPA(Differential Power Analysis)방법이 있다. SPA가 소개되면서 암호 시스템 설계자들은 여러 가지 대응책을 제시하였고, 그 중, 가장 잘 알려진 것이 더미 연산(dummy operation)을 추가한 알고리즘과 Signed All-Bits Set(sABS)[3] 형태의 스칼라 곱셈 연산 알고리즘이다. 하지만, 더미 연산을 추가한 알고리즘은 Fouque가 소개한 'Doubling Attack'에 의해 SPA에 그 취약점이 드러나게 된다[10].

본 논문에서는 Doubling Attack이 적용되어질 수 없는 sABS를 이용한 스칼라 곱셈 알고리즘에 대한 새로운 두 가지 분석방법을 제안한다. 첫째는 Recursive Attack으로서 두 개의 입력 값이 연산되어지는 과정에서 비밀 키의 추측한 비트를 전후해서 동일한 두배 연산(point doubling, ECDBL)이 발견되도록 입력 값을 조정해 비밀 키의 최상위 비트부터 순차적으로 키를 찾아내는 공격방법이다. 둘째는 Initializing Attack으로 비밀 키의 추측한 비트에서 항상 일정한 두배 연산이 일어나도록 입력 값을 조정하는 공격 방법이다. 제안하는 분석법은 비밀 키의 일정비트를 추측하고 입력 값을 조정해 키를 찾아내는 방법으로 부채널 공격 방법의 대응법인 비밀 키의 리코딩 방법에 대한 공격 방향을 제시한다. 또한 실험적인 결과를 통해 본 논문에서 제안하는 두 가지 공격 방법이 실제로 적용되어질 수 있음을 보인다.

Ⅱ. 스칼라 곱셈 연산

타원곡선 암호 시스템은 Koblitz와 Miller에 의해 1985년에 제안되었다[4,9]. 타원곡선 암호 시스템은 RSA와 동일한 안전성을 유지하기 위한 암호 시스템의 키 길이가 현저히 줄어들기 때문에 저장 용량과 대역폭에 제한이 있는 스마트카드와 같은 장비가 쓰이는 환경에 적합하다는 장점이 있다.

RSA와 DLP기반의 암호 시스템은 암호화, 복호화에 쓰이는 연산이 지수승 연산(modular exponentiation)이지만, 타원곡선 암호시스템에서는 그와 동일하게 생각되어질 수 있는 스칼라 곱셈 연산(scalar multiplication)을 사용한다. 따라서 타원곡선 암호시스템에서 스칼라 곱셈 연산이 차지하는 비중은 상당히 크다. 스칼라 곱셈 연산은 타원곡선 위의 점 P에 대한 비밀 키 d만큼의 덧셈을 수행하는 연산이다. 다음 알고리즘은 스칼라 곱셈 알고리즘 중 가장 보편적으로 사용되어지는 'Double-and-add 스칼라 곱셈' 알고리즘이다.

알고리즘 1. Double-and-add scalar multiplication
입력 A point P,
$d=(d_{n-1}d_{n-2}\cdots d_1d_0)_2$, $d_{n-1}=1$
출력 $Q=dP$
1. $S=P$.
2. For $i=n-2$ down to 0 do :
 2.1. $S=2S$.
 2.2. If $d_i=1$, $S=S+P$
3. Return S

알고리즘 1은 RSA의 지수승 연산에서 사용되어지는 'square-and-multiply 지수승' 알고리즘과 동일한 방법으로 구성되어 있으며, 비밀 키 d의 최상위 비트부터 최하위 비트(left-to-right)까지 읽어나가며 연산을 수행한다.

Ⅲ. 단순전력 공격 방법과 그 대응법들

전력분석 공격은 스마트카드에 물리적 변환을 가하지 않고 전력 신호를 측정, 분석함으로서 비밀 키의 값을 알아내는 공격 방법이다. 이러한 분석 방법은 SPA와 DPA로 나누어지는데, DPA는 여러 개의 전력 신호를 이용해 통계학적으로 비밀 키를 찾아내는 반면, SPA는 단순히 하나의 전력신호를 관찰함으로써 사용된 비밀 키의 정보를 알아내는 방법이다.

3.1 단순 전력 공격 방법(SPA)

비밀 키 d에 대한 스칼라 곱셈 연산은 d의 최상위 비트부터 스캔하며 d의 비트가 1일 때는 알고리즘 1의 단계 2.1과 2.2를 동시에 수행하고, 0일 때는 단계 2.1만을 수행하게 된다. 즉 비트의 정보에 따라 덧셈 연산(point addition, ECADD)과 두배 연산(point doubling, ECDBL)을 동시에 수행하거나, 두배 연산만을 수행하게 된다. 따라서 스칼라 곱셈 연산의 전력 소모량은 비트의 정보에 따라 달라지기 때문에 비밀 키의 값이 드러나게 된다. 이 때, 이러한 단순 전력 공격 방법을 SPA라 한다.

3.2 대응법

일반적인 스칼라 곱셈 연산을 수행하는 알고리즘 1은 비트 정보에 따른 덧셈 연산의 수행여부가 SPA에 취약하게 되는 원인이 된다. 이러한 취약점을 극복하는 방법으로는 비트 정보에 상관없이 덧셈(뺄셈) 연산을 수행하는 알고리즘들이 제안되었다.

3.2.1 더미연산 알고리즘

SPA의 대응법으로 제안된 다음 알고리즘은 비트 정보에 상관없이 더미 연산(dummy operation)을 수행하는 방법이다[2].

알고리즘 2. Double-and-add-always scalar multiplication

입력 A point P,
$d=(d_{n-1}d_{n-2}\cdots d_1d_0)_2$, $d_{n-1}=1$
출력 $Q=dP$
1. $S[0]=P$.
2. For $i=n-2$ down to 0 do :
 2.1. $S[0]=2S[0]$.
 2.2. $S[1]=S[0]+P$
 2.3. $S[0]=S[d_i]$.
3. Return $S[0]$

알고리즘 2는 d_i의 비트 값에 관계없이 단계 2.2를 수행한다. 단계 2.2는 0인 비트 값에 대해서는 수행되어질 필요가 없는 불필요한 덧셈 연산을 수행하지만 비트 정보에 따른 덧셈 연산의 수행 유무는 SPA에 취약하기 때문에 제안된 방법이다. 하지만 이 알고리즘은 더미 연산으로 인한 효율성의 측면에서도 문제가 있지만, 다음 절에서 소개하는 'Doubling Attack'에 의해 SPA에 안전하지 않음이 밝혀졌다.

3.2.2 Signed All-Bits Set(sABS) 리코딩

비밀 키를 랜덤화하여 스칼라 곱셈 연산을 수행하는 방법이 SPA의 또 다른 대응법으로 제시되었다.

Hedabou에 의해 제안된 다음의 sABS 리코딩 방법은 비밀 키의 비트 값이 0인 부분을 없애고 1과 -1로만 표현되게 구성하는 방법이다.

알고리즘 3. sABS recoding

입력 An odd scalar $d=(d_{n-1},\cdots,d_1,d_0)_2$
출력 $d'=(d'_{n-1},\cdots d'_0)$, $d'_i\in\{-1,1\}$
1. For i from 1 to $n-1$ do
 1.1 $k[0]\leftarrow d_i, k[1]\leftarrow d_{i-1}, k[2]\leftarrow -d_{i-1}$.
 1.2 $d'_i\leftarrow k[1-d_i], d'_{i-1}\leftarrow k[2-d_i]$.
2. Return d'

알고리즘 3에서의 리코딩 기법은 0001을 $1\bar{1}\bar{1}\bar{1}$의 0이 없는 형태로 변환할 수 있다는 사실을 이용하여 제안된 기법이다.(이 때, $\bar{1}=-1$로 나타낸다.) 다음 알고리즘은 이 변환된 sABS 리코딩 값을 이용해 스칼라 곱셈 연산을 수행한다.

알고리즘 4. Scalar multiplication with sABS recoding

입력 A point P,
$d=(d_{n-1}d_{n-2}\cdots d_1d_0)_2$, $d_{n-1}=1$
출력 $Q=dP$
1. If(d is even) then $t=d+1$
2. Recoding :
 $t'=(t'_{n-1}t'_{n-2}\cdots t'_1t'_0)_2$, $t'_i\in\{-1,1\}$
 ← record t with sABS recoding
3. $S=P$
4. For $i=n-2$ downto 0
 4.1 $S=2S$
 4.2 If($t'_i=1$), then $S=S+P$
 else then $S=S-P$
5. If(d is even) then $S=S-P$
6. Return S

Ⅳ. Doubling Attack

SPA에 안전하다고 알려진 더미 연산을 추가한 알고리즘 2의 'Double-and-add always 스칼라 곱셈' 방법은 공격자가 얻은 정보의 통계치를 이용한 차분 전력 공격(DPA)에 취약하다고 알려져 있다. 하지만 DPA는 공격자에게 많은 양의 정보와 시간을 요구한다.

Fouque에 의해 소개된 Doubling Attack은 공격자가 point A와 B에 대해 2A의 연산과 2B의 연산에서 동일한 전력량이 발견되면, A=B로 판단할 수 있는 능력을 가졌을 때 알고리즘 2의 double-and-add always scalar multiplication이 DPA가 아닌 SPA로 공격될 수 있는 방법이다.

Doubling Attack은 아래 표처럼 입력(input)값 P에 대한 dP의 계산과 입력 값 $2P$에 대한 $d(2P)$값을 계산할 때, 그 전력량을 비교해 비트 값이 0인 비트를 전후해서 같은 두배 연산이 일어나 같은 전력량이 사용되는 것을 확인하여 비밀 키의 전체 값을 알아내는 공격 방법이다.

input	1	0	1	0	0	1	0	0	1
P	0	2P	**4P**	10P	**20P**	**40P**	82P	**164P**	**328P**
	P	3P	5P	11P	21P	41P	83P	165P	329P
2P	0	**4P**	8P	**20P**	**40P**	80P	**164P**	**328P**	656P
	2P	6P	10P	22P	42P	82P	166P	330P	658P

Doubling Attack은 이전에 언급한 것과 같이 스마트 카드가 두 입력 값 P와 2P에 대한 스칼라 곱셈 연산을 수행할 때, 일정 비트에서 같은 두배 연산을 수행한다면, 공격자가 그 값을 알지는 못하더라도 같은 point를 두배 연산 했다는 정보는 얻을 수 있다는 전제 조건이 있을 때 가능한 공격 방법이다.

4.1 sABS의 Doubling Attack에 대한 안전성

Doubling Attack은 d의 일정 비트가 0일 때 드러나는 취약점을 이용한 공격 방법이다. 그러므로 이러한 공격 방법은 0인 비트가 없는 sABS 리코딩에서는 적용이 불가능하다. 알고리즘 4에서 공격자가 Doubling Attack을 시도하여, 비밀 키 d(or $d+1$)의 리코딩한 값의 상위부터 l번째 비트인 t'_{n-l}에 있는 값을 예상하려고 한다면, 입력 값 P에 대해 리코딩 값의 상위 l비트까지 계산한 값과 입력 값 2P에 대해 상위 $l-1$번째 비트까지 계산한 결과가 다음 식처럼 같아야 한다.

$$\left(\sum_{i=0}^{l-1} t'_{n-l+i}2^i\right)P=\left(\sum_{i=0}^{l-2} t'_{n-l+i+1}2^i\right)2P \quad (1)$$

$$\Rightarrow \quad \left(\sum_{j=0}^{l-1} t'_{n-l+j}2^j\right)P=\left(\sum_{j=1}^{l-1} t'_{n-l+j}2^j\right)P \quad (1)$$

따라서 식 (1)이 만족하기 위해선 $t'_{n-l}=0$ 이 되어야 함을 쉽게 알 수 있다. 하지만 sABS 리코딩 값은 1과 -1로 구성되기 때문에, Doubling Attack에 취약하지 않다.

Ⅴ. 제안하는 공격들

본 논문에서는 타원곡선의 스칼라 곱셈 연산에서 sABS 형태의 스칼라로 변환했을 때 공격 가능한 새로운 공격 방법으로 Recursive Attack과 Initializing Attack을 제시한다. 이 두 개의 새로운 공격 방법은 Doubling Attack과 같이 타원곡선 상의 포인트 A와 B에 대해 2A의 연산과 2B의 연산에서 동일한 전력량이 발견되면, A=B로 판단할 수 있는 능력이 공격자에게 있을 때, 가능한 공격 방법이다.

5.1 Recursive Attack

제안하는 공격 방법 중, Recursive Attack의 기본 아이디어는 알아내고자 하는 비밀 키 d의 일정 비트 값을 예측해 해당 비트를 전후해서 두 입력 값이 동일한 두 배 연산을 수행해 같은 전력이 발견되도록 두 입력 값을 조정하는 것이다. 이러한 방법으로 비밀 키 d의 모든 비트 정보를 순차적으로 알아낼 수 있다.

5.1.1 입력 값의 조정

알고리즘 4에서 비밀 키 d를 리코딩한 값 $t'=(t'_{n-1}t'_{n-2}\cdots t'_1t'_0)_2$, $t'_i\in\{-1,1\}$ 의 상위 비트 $t'_{n-1}t'_{n-2}\cdots t'_{u+2}t'_{u+1}$의 정보를 알고 있다고 가정했을 때, t'_u의 값을 알아내기 위해 입력 값을 조정해 보자. 두 입력 값을 xP, yP라고 했을 때, 알고리즘 4의 단계 4.1에서 입력 값 xP의 t'_u에서 t'_{u-1}단계로 수행될 때의 두배 연산과 입력 값 yP의 t'_{u+1}에서 t'_u단계로 수행될 때의 두배 연산을 동일하게 만들기 위해 x와 y값을 조정한다. 만약 $t'_u=1$ 이라고 예상했다면, 알고리즘 4

의 단계 4에서 입력 값이 xP일 때 $i=u$일 때까지 반복문을 수행한 S의 값은 $S=(\sum_{i=u+1}^{n-1} t'_i 2^{i-u}+1)xP$이고, 입력 값이 yP일 때 $i=u+1$일 때까지 반복문을 수행한 S의 값은 $S=(\sum_{i=u+1}^{n-1} t'_i 2^{i-u-1})yP$이다. 이 때, 동일한 두배 연산이 일어나기 위해,

$$(\sum_{i=u+1}^{n-1} t'_i 2^{i-u}+1)xP=(\sum_{i=u+1}^{n-1} t'_i 2^{i-u-1})yP$$ 라 하고,

이 식을 만족하는 두 입력 값 xP와 yP를 선택하면 다음 식과 같다.

$$\begin{aligned} xP&=(\sum_{i=u+1}^{n-1} t'_i 2^{i-u-1})P \\ yP&=(\sum_{i=u+1}^{n-1} t'_i 2^{i-u}+1)P \end{aligned} \quad (2)$$

$t'_u=-1$ 이라고 예상했다면 동일한 이유로 두 입력 값을 다음의 식처럼 선택하면 된다.

$$\begin{aligned} xP&=(\sum_{i=u+1}^{n-1} t'_i 2^{i-u-1})P \\ yP&=(\sum_{i=u+1}^{n-1} t'_i 2^{i-u}-1)P \end{aligned} \quad (3)$$

식 (2)과 식 (3)의 $\sum_{i=u+1}^{n-1} t'_i 2^{i-u-1}$부분은 알고리즘 4의 t'에서 우리가 찾아내려는 비트 값 t'_u보다 상위 비트 값이므로 이 값을 k라고 한다면, $t'_u=1$이라고 예상했을 때, 두 입력 값은 kP, $(2k+1)P$이고 $t'_u=-1$이라고 예상했을 때, 두 입력 값은 kP, $(2k-1)P$로 선택되어질 수 있다. 이러한 방법으로 비밀 키 d에 대한 정보를 최상위 비트부터 순차적으로 찾아낼 수 있다.

5.1.2 Recursive Attack의 예

본 소절에서는 Recursive Attack의 이해를 돕기 위해 간단한 예를 들어본다. 알고리즘 4의 입력 값 d를 $(101010011)_2$라고 한다면 단계 2에서 $t'=11\bar{1}1\bar{1}1\bar{1}11$이 된다. t'의 상위 네 개의 비트를 알고 있다고 가정하고, 다섯 번째 비트를 아래 표처럼 추측한다.(상위 네 개의 비트 값 : $11\bar{1}1=(11)_{10}$) 우선 공격자는 1로서 다섯 번째 비트를 추측하고 $11P$, $(2*11+1)P$를 입력 값으로 선택하고 다섯 번째 비트를 전후해 같은 두배 연산이 일어나는지 다음과 같이 확인한다.

input	1	1	$\bar{1}$	1	$\bar{1}$	1	$\bar{1}$	$\bar{1}$	1
11P	11P	22P	66P	110P	242P	**462P**	...	...	...
		33P	55P	121P	**231P**	473P	...	...	...
23P	23P	46P	138P	230P	506P	...	...	...	...
		69P	115P	253P	483P	...	...	...	...

같은 두배 연산이 일어나지 않았기 때문에 공격자는 $(2*11-1)P$를 입력 값으로 선택해 다섯 번째 비트를 전후해서 $11P$와 같은 두배 연산이 일어남을 다음 표와 같이 확인하고 다섯 번째 비트를 $\bar{1}$로 알아낸다.

input	1	1	$\bar{1}$	1	$\bar{1}$	1	$\bar{1}$	$\bar{1}$	1
11P	11P	22P	66P	110P	242P	**462P**	...	...	...
		33P	55P	121P	**231P**	473P	...	...	...
21P	21P	42P	126P	210P	**462P**	...	...	...	...
		63P	105P	**231P**	441P	...	...	...	...

5번째 비트가 $\bar{1}$인 것을 알았기 때문에, 상위 다섯 개의 비트 값은 21이 된다. (5번째 비트를 계산할 때 이용했던 값을 다시 이용할 수 있다.) 따라서 추가적으로 입력 값을 $(2*21+1)P$로 넣어서 그 전에 이용했던 입력 값 $21P$와 여섯 번째 비트를 전후해 같은 두배 연산이 일어났는지 확인해 그 다음 비트인 여섯 번째 비트를 다음과 같이 추측한다.

input	1	1	$\bar{1}$	1	$\bar{1}$	**1**	$\bar{1}$	$\bar{1}$	1
21P	21P	42P	126P	210P	462P	882P	**1806P**	...	...
		63P	105P	231P	441P	**903P**	1785P	...	...
43P	43P	86P	258P	430P	946P	**1806P**	...	...	...
		129P	215P	473P	**903P**	1849P	...	...	...

여섯 번째 비트를 전후해서 같은 두배 연산이 일어났기 때문에 여섯 번째 비트는 1로 찾아낼 수 있다. 이러한 방법으로 타원곡선의 스칼라 곱셈 연산의 비밀 키 d에 관한 정보를 최상위부터 순차적으로 찾아낼 수 있다.

5.2 Initializing Attack

sABS 리코딩 값 형태의 스칼라에 대한 Recursive Attack은 비밀 키 d를 최상위비트부터 순차적으로 알아내는 공격 방법으로 최저 한 번, 최대 두 번의 입력 값을 넣어 비밀 키의 각 비트 값을 알아낼 수 있다. sABS

리코딩 값 형태의 스칼라에 대한 두 번째로 제안하는 공격 방법인 Initializing Attack은 Recursive Attack이 두 개의 입력 값을 이용하는 반면, 하나의 입력 값을 이용하여 비트 값을 알아낼 수 있는 공격 방법이다. 이 공격 방법은 알고리즘 4의 단계 4.1에서 입력 값 P에 대해 P에서 $2P$로의 두배 연산에 대한 정보를 공격자가 쉽게 얻을 수 있다는 취약점을 이용한다.

5.2.1 입력 값의 조정

Initializing Attack은 Recursive Attack과 비슷한 방법으로 알아내려는 비밀 키 d의 상위 일정 비트를 알고 있을 때, 그 다음 비트를 순차적으로 알아내는 공격이다. 이 공격 방법에 대한 기본 아이디어는 공격자가 선택한 입력 값 xP에 대해 알아내려는 비트까지의 계산 값이 P가 되게 하고, 그다음 비트의 계산을 수행할 때 P에서 $2P$로의 두 배 연산이 나타나게 하는 것이다. 공격자가 알고리즘 4에서 sABS 리코딩 값 t'의 상위비트 $t'_{n-1}t'_{n-2}\cdots t'_{u+2}t'_{u+1}$를 알고 있다고 가정했을 때, t'_u를 1로서 예상했다면, 단계 4에서 입력 값이 xP일 때 $i=u$일 때까지 반복문을 수행한 S의 값은 $S=(\sum_{i=u+1}^{n-1} t'_i 2^{i-u}+1)xP$이고, 이 값이 P와 동일한 값을 가지게 x를 조정한다. $k=\sum_{i=u+1}^{n-1} t'_i 2^{i-u-1}$이라고 한다면 이 때 x값은 식 (4)와 같이 선택하면 된다.

$$x=(2k+1)^{-1} \bmod \#E \tag{4}$$

$\#E$ 는 타원곡선 암호시스템에서 타원 곡선의 order를 뜻한다.

표준문서 FIPS 186-2, SECG, WTLS, ISO/IEC 15946-4에 나와있는 타원 곡선의 order는 q, $2q$, $4q$, $6q$ (q: prime)의 형태이므로 $2k+1$은 2, q와 서로소이지만 $\gcd(2k+1, 6q)=3$일 수 있기 때문에, 타원곡선의 order가 $6q$인 경우 $(2k+1)^{-1}$이 존재하지 않는 경우가 존재한다. 하지만 이 경우, 즉 $2k+1=3t$인 경우에 $2k-1=3t-2$의 형태로 3의 배수가 아니므로 $\gcd(2k-1, 6q)=1$이 된다. 따라서 $\gcd(2k+1, \#E)\neq 1$일 경우, t'_u를 1이 아닌 -1로 예상한다. 그러면, 단계 4에서 입력 값이 xP일 때, $i=u$일 때까지 반복문을 수행한 S의 값은 $S=(\sum_{i=u+1}^{n-1} t'_i 2^{i-u}-1)xP$이고, 이 값이 P와 동일한 값을 가지게 x를 조정한다. $k=\sum_{i=u+1}^{n-1} t'_i 2^{i-u-1}$이라고 한다면 이 때 x값은 식 (5)와 같이 선택하면 된다.

$$x=(2k-1)^{-1} \bmod \#E \tag{5}$$

5.2.2 Initializing Attack의 예

본 소절에서는 Initializing Attack의 이해를 돕기 위해서 간단한 예를 들어본다. 우선 알고리즘 4에서 사용한 elliptic curve의 order가 73이라고 가정하자. 비밀 키 d를 예를 들어 $(101010011)_2$라고 한다면 단계2에서 리코딩 값은 $t'=11\bar{1}1\bar{1}1\bar{1}\bar{1}1$이 된다. 공격자가 t'의 상위 네 개의 비트를 알고 있다고 가정하고, 다섯 번째 비트를 1로서 예측하는 경우를 다음 표처럼 고려한다.(상위 네 개의 비트 값 : $11\bar{1}1=(11)_{10}$, $23^{-1} \bmod 73=54$)

input	1	1	$\bar{1}$	1	$\bar{1}$	1	$\bar{1}$	$\bar{1}$	1
54P	54P	35P	67P	26P	14P	**66P**	...	...	...
		70P	13P	7P	**33P**	47P	...	...	...

다섯 번째 비트에서 P에서 $2P$로 두배 연산이 발견되지 않았기 때문에 $\bar{1}$로서 다섯 번째 비트 값을 알아낼 수 있고, 상위 다섯 개의 비트 값은 21이 된다. 여섯 번째 비트를 순차적으로 알아내기 위해 1로서 그 값을 예상했다면, 입력 값 xP에서 x를 $(2*21+1)^{-1} \bmod \#E=17$로 선택할 수 있다.

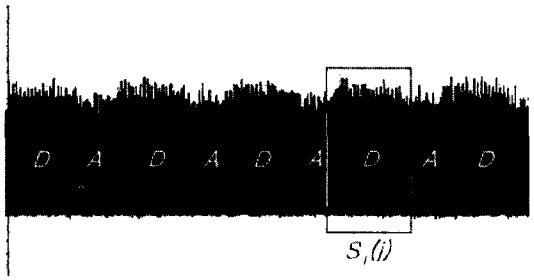
input	1	1	$\bar{1}$	1	$\bar{1}$	**1**	$\bar{1}$	$\bar{1}$	1
17P	17P	34P	29P	24P	9P	57P	**2P**	...	...
		51P	12P	41P	65P	**P**	58P	...	...

P에서 $2P$로의 두배 연산이 여섯 번째 비트에서 발견되었으므로 여섯 번째 비트는 1인 것을 알아낼 수 있다. 이러한 방법을 d의 최상위 비트부터 순차적으로 시행하면 타원곡선의 스칼라 곱셈 연산의 비밀 키의 값 d를 찾아낼 수 있다.

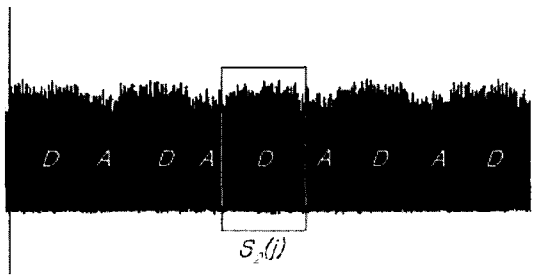
VI. 실험 결과 및 적용

본 장에서는 제안하는 공격방법의 실질적인 적용을 위해 Doubling Attack과 본 논문에서 제안하는 두 가지

공격방법의 '타원곡선 상의 포인트 A와 B에 대해 2A의 연산과 2B의 연산에서 동일한 전력량이 발견되면, A=B로 판단할 수 있는 능력을 공격자는 가지고 있다' 는 가정이 실제로 가능함을 실험 결과와 함께 제시한다.



(a) 입력값 A에 대한 파형 곡선



(b) 입력값 B에 대한 파형곡선

〔그림 1〕 입력값 A, B에 대한 파형곡선

위의 그림처럼 스칼라 곱셈 연산에서 덧셈 연산과 두 배 연산의 전력량을 구별할 수 있다면, 비밀키의 상위 세비트를 알고 있는 공격자는 Recursive Attack을 이용, 순차적으로 상위 네 번째 비트를 알기 위해 두 입력 값을 조정해서 넣어 첫 번째 입력 값의 네 번째 두배 연산이 일어나는 부분과 두 번째 입력 값의 세 번째 두배 연산이 일어나는 부분을 근사적으로 뽑아내 파형을 비교한다. 근사적으로 뽑아낸 후 두배 연산이 시작되는 부분을 맞추는 작업은 '정렬(align)' 작업을 통해 가능하다. 이 정렬된 두 개의 두배 연산 파형에서 같은 두배 연산이 일어났는지의 판단은 다음과 같은 실험을 통해 판단 할 수 있다.

우선 실험 환경은 PIC microcontroller에서 이루어졌다. 실험 환경의 제약으로 인해 GF(2^8)의 작은 타원곡선 상에서 affine coordinate로 sABS 형태의 스칼라 곱셈 연산을 수행하도록 구현한 모듈을 PIC-chip에 올렸다.

비교할 두 개의 포인트 P_1, P_2의 두 배 연산에 대한 파형을 $S_1(j)$, $S_2(j)$ $(1 \le j \le n_D)$라고 한다면 다음의 함수 값을 계산할 수 있다.

$$Diff(S_1, S_2) = \frac{1}{n_D}\sum_{j=1}^{n_D}(S_1(j) - S_2(j))^2$$

$Diff(S_1, S_2)$함수는 암호 시스템이 같은 연산을 수행할 때와 다른 연산을 수행할 때를 구별하기 위한 함수로서 다른 연산일 때 그 값이 크게 나타난다. 이 함수에서 n_D의 값은 두배 연산이 일어나는 전체 구간을 나타내는 값일 필요는 없다. 단지 같은 두배 연산이 일어나는지 공격자가 판단할 수 있으면 되므로 affine coordinate에서 두 좌표 모두 적어도 한 번은 연산 과정에 포함되는 적당히 큰 값이면 되고, 다른 포인트의 두배 연산에 대한 $Diff(S_1, S_2)$의 함수 값을 더 크게 하기 위해선 n_D값으로 보다 큰 값을 선택하면 된다.

실제 실험에서 n_D값을 65000으로 정하고 같은 포인트에 대한 두배 연산과 다른 포인트에 대한 두배 연산의 파형을 모아 $Diff(S_1, S_2)$을 계산해 보는 과정을 반복하여 수행했을 때, 그 값은 〔표 1〕처럼 상당한 차이를 보인다.

〔표 1〕 $Diff(S_1, S_2)$에 의한 파형 구별

	$Diff(S_1, S_2)$		
$P_1 = P_2$	21.982..	19.243..	25.338...
$P_1 \neq P_2$	71.338..	118.29...	86.030..

실제 스칼라 곱셈 연산은 GF(2^8)보다 더 큰 유한체에서 이루어지기 때문에 이 $Diff(S_1, S_2)$ 값은 본 실험에서 얻었던 값보다 훨씬 큰 차이를 낼 수 있으므로 더 확연히 구분할 수 있을 것으로 보인다.

Initializing attack의 경우도 비슷한 방법으로 공격이 진행되어질 수 있다. 즉, 입력 포인트 P에 대한 파형 S_1을 얻은 후, 이 전력 파형에서 처음의 두배 연산에 대한 파형만을 취하고, 비밀키의 한 비트를 알아내기 위한 조정한 입력 포인트 xP에 대한 비교하고자 하는 두배 연산에 대한 파형 S_2를 이용, 정렬 후에 $Diff(S_1, S_2)$의 값을 통해 비밀 키의 값을 알아낼 수 있다.

Ⅶ. 비교

본 논문에서 제안한 두 가지 공격 방법을 sABS 형태의 스칼라 곱셈 연산에 적용했을 때 공격자가 시행해야 되는 계산 량을 비교해 보도록 한다. 우선 Recursive

Attack은 하나의 비트를 알아내는데 평균적으로 1.5번의 스칼라 곱셈 연산을 수행해야한다. 즉 160 bit의 비밀 키 전체 값을 알아내기 위해서 $1.5\times(160-1)$번의 스칼라 곱셈 연산을 수행해야 된다. 반면, 두 번째 제안한 Initializing Attack은 하나의 비트를 알아내기 위해선 한 번의 스칼라 곱셈 연산을 수행하지만, 하나의 비트 정보를 알아낼 때마다 정수의 역원을 구하는 계산이 공격자에 의해 수행되어야만 한다. n비트 스칼라 곱셈 연산에 대한 공격의 평균적인 계산 량은 아래 표와 같다.

〔표 2〕 두 공격에 요구되어지는 연산 량

	Recursive	Initializing
스마트카드의 스칼라곱셈	$\frac{3(n-1)}{2}$	$n-1$
공격자의 정수 inversion연산	0	$n-1$

Ⅷ. 결론

본 논문에서는 Fouque가 제안한 Doubling Attack이 적용 불가능한 sABS 형태의 스칼라 곱셈 연산에 대한 새로운 두 가지 공격 방법으로 Recursive Attack과 Initializing Attack을 제시하고 실험적인 결과를 통해 같은 연산에 대한 파형과 다른 연산에 대한 파형을 구별함으로서 제시한 공격들이 실제로도 적용될 수 있는 구체적인 분석 방법을 소개하였다. 제안한 두 가지 공격 방법은 비밀 키의 일정 비트를 예측해 입력 값을 조절하여 공격자가 예측한 두배 연산이 발견되면 옳은 키로 판단하는 방법으로 부채널 공격 방법의 대응법인 스칼라 리코딩에 대한 공격방향을 제시한다.

참고문헌

〔1〕 Bellcore Press Release, "New threat model breaks crypto codes," Sep. 1996 or D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults", Advances in Cryptology-EUROCRYPT '97, LNCS 1233, pp. 37-51, Springer-Verlag, 1997.

〔2〕 J. S. Coron, "Resistance against differential power analysis for Elliptic Curve Cryptosystems," Proc. of Workshop on Cryptographic Hardware and Embedded Systems, pp. 292-302, Springer-Verlag, 1999.

〔3〕 M.Hedabou, P.Pinel, and L. Bebeteau, "Countermeasures for Preventing Comb Method Against SCA Attacks," Information Security Practise and Experience Conference, ISPEC05, LNCS 3439, pp. 85-96, Springer-Verlag, 2005

〔4〕 N. Koblitz, "Elliptic curve cryptosystems," Math. of Computation, vol. 48, pp. 203-209, 1987.

〔5〕 P. Kocher, J. Jaffe, and B. Jun, "Timing Attacks on Implementations of Diffie- Hellman, RSA, DSS, and Others Systems." CRYPTO'96, LNCS 1109, pp. 104-113, Springer- Verlag, 1996.

〔6〕 P. Kocher, J. Jaffe, and B. Jun, "Differential power analysi," Advances in Cryptology-CRYPTO'99, pp. 388-397, Springer-Verlag, 1999.

〔7〕 P. Kocher, J. Jaffe, and B. Jun, "Introduction to differential power analysis and related attacks," http://www.cryptography.com/dpa/technical, 1998.

〔8〕 T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Power analysis attacks on modular exponentiation in Smart cards," Proc. of Workshop on Cryptographic Hardware and Embedded Systems, pp. 144-157, Springer-Verlag, 1999.

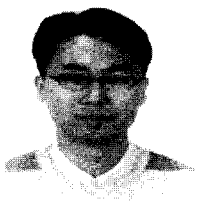
〔9〕 V. Miller, "Uses of elliptic curves in crypto in cryptography," Proc. of Advances in Cryptology-CRYPTO' 85, pp. 417-426, Springer-Verlag, 1985.

〔10〕 Pierre-Alain Fouque and Frederic Valette, "The Doubling Attack - Why Upwards Is Better than Downwards," CHES 2003, LNCS 2779, pp. 269 - .280, 2003, Springer-Verlag Berlin Heidelberg 2003

〔11〕 S. M. Yen, S. J. Kim, S. G. Lim, and S. J. Moon, "A countermeasure against one physical cryptanalysis May Benefit Another Attack", Proc. of the ICISC 2001, Korea. Dec. 2001

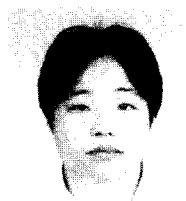
〈著者紹介〉

김 희 석 (HeeSeok Kim) 학생회원
2006년 2월: 연세대학교 수학과 졸업(학사)
2006년 3월~현재: 고려대학교 정보경영공학전문대학원 석사과정
<관심분야> 부채널 공격, 공개키 암호시스템 안전성 분석 및 고속구현, 타원곡선



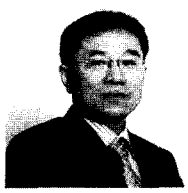
한 동 국 (Dong-Guk Han) 일반회원
1999년: 고려대학교 수학과 졸업(학사)
2002년: 고려대학교 수학과 석사 (이학석사)
2005년: 고려대학교 정보보호대학원 박사 (공학박사)
2004년 4월~2005년 4월: 일본 Kyushu Univ., 방문연구원
2005년 4월~2006년 4월: 일본 Future Univ.-Hakodate, Post.Doc.
2006년 6월~현재: 한국전자통신연구원 정보보호연구단 선임연구원
<관심분야> 공개키 암호시스템 안전성 분석 및 고속 구현, 부채널 분석, RFID/USN 정보보호 기술

김 성 경 (Sung-Kyoung Kim) 학생회원
2005년 2월: 동의대학교 수학과 학사
2005년 3월 ~ 현재: 고려대학교 정보경영공학전문대학원 석사과정
<관심분야> 부채널 공경, 공개키 암호, 암호칩 설계 기술



김 태 현 (Tae Hyun KIM) 정회원
2002년 2월: 서울 시립대학교 수학과 이학사
2004년 8월: 고려대학교 정보보호 대학원 공학석사
2005년 2월~현재: 고려대학교 정보경영공학전문대학원 박사과정
<관심분야> 부채널 공격, 공개키 암호 알고리즘, 암호칩 설계 기술

박 영 호 (Young-Ho Park) 정회원
1990년: 고려대학교 수학과 이학사
1993년: 고려대학교 수학과 이학석사
1997년: 고려대학교 수학과 이학박사
2006년 2월~현재: 세종 사이버 대학교 정교수
<관심분야> 정수론, 공개키 암호, 암호 프로토콜, 부채널 공격



임 종 인 (Jongin Lim) 정회원
1980년 2월: 고려대학교 수학과 학사
1982년 2월: 고려대학교 수학과 석사
1986년 2월: 고려대학교 수학과 박사
1999년 2월~현재: 고려대학교 정보보호대학원 원장, CIST 센터장
<관심분야> 암호이론, 정보보호정책