情報保護學會論文誌
第14券 第5號, 2004. 10

랜덤한 덧셈-뺄셈 체인에 대한 부채널 공격*

한 동 국[a)†], 장 남 수[a)‡], 장 상 운[b)], 임 종 인[a)]
고려대학교 정보보호대학원[a)], 국가보안기술연구소[b)]

Side channel attack on the Randomized Addition-Subtraction Chains

Dong-Guk Han[a)†], Nam-Su Chang[a)‡], Sang-Woon Jang[b)], Jongin Lim[a)]
Korea University[a)], National Security Research Institute[b)]

요 약

Okeya-Sakurai는 타원곡선 암호시스템에 대한 부채널 공격의 대응방법으로 소개된 Oswald-Aigner의 랜덤한 덧셈-뺄셈 체인(Randomized Automaton 1, 2) 대응방법 [18]이 SPA 공격에 취약함을 보였다[15,16]. 그러나 본 논문에서는 Okeya-Sakurai의 공격 알고리즘 [15,16]에 두 가지 잠재된 문제가 있음을 보인다. 그리고 두 가지 문제점에 대한 해결책을 제시하고 [15,16,19]와는 다른 새로운 효율적인 공격 알고리즘을 제안한다. 표준에 제안되어 있는 163비트 비밀키를 사용하는 알고리즘에 본 논문의 분석방법을 적용해 구현한 결과, 단순한 형태의 랜덤한 덧셈-뺄셈 체인(Randomized Automaton 1)에서는 20개의 AD수열로 대략 94%의 확률로 공격이 성공하며 30개의 AD수열로는 대략 99%의 확률로 공격이 성공한다. 또한, 복잡한 형태(Randomized Automaton 2)에서는 40개의 AD수열로 94%의 확률로 70개의 AD수열로는 99%로의 확률로 공격이 성공한다.

ABSTRACT

In [15,16], Okeya and Sakurai showed that the randomized addition-subtraction chains countermeasures [18] are vulnerable to SPA attack. In this paper, we show that Okeya and Sakurai's attack algorithm [15,16] has two latent problems which need to be considered. We further propose new powerful concrete attack algorithms which are different from [15,16,19]. From our implementation results for standard 163-bit keys, the success probability for the simple version with 20 AD sequences is about 94% and with 30 AD sequences is about 99%. Also, the success probability for the complex version with 40 AD sequences is about 94% and with 70 AD sequences is about 99%.

Keywords : *Elliptic Curve, SPA, DPA, Randomized Addition-Subtraction Chains*

Ⅰ. 서 론

타원곡선 이론을 이용한 공개키 암호시스템(ECC)은 Koblitz 와 Miller 에 의해 처음으로 제안되었다[8,12]. 타원곡선 암호시스템은 이산대수문제를 기반으로 하는 공개키 암호시스템이며 다른 암호시스템과 비교하여 동일한 안전성에서 상대적으로 작은 키 사이즈를 제공한다. 타원곡선 암호시스템은 메모리와 계산능력이 제한된 스마트카드와 같은 장비에 구현이 적합하다. 또한 최근에 Kocher[10] 등은 암호시스템이 구현 된 스마트카드의 부채널 공격 방법을 소개하였다. 암호시스템의 부채널 공격은 크게 시차 공격, 전력분석 공격, 오류주입 공격이 있으며 이 중에서

접수일 : 2004년 6월 3일 ; 채택일 : 2004년 8월 20일
* 본 연구는 정보통신부 대학 IT연구센터 육성·지원사업의 연구결과로 수행 되었습니다.
† 주저자 : christa@korea.ac.kr
‡ 교신저자 : ns-chang@korea.ac.kr

현재 전력분석 공격은 가장 활발하게 연구되고 있으며 스마트 카드 산업계에서도 공격에 대한 대응방법을 적용하여 제품을 제작하고 있다. 전력분석 공격은 다시 SPA(Simple Power Analysis, 단순전력분석)와 DPA(Differential Power Analysis, 차분전력분석)으로 구분되며, 두 가지 공격은 타원곡선 암호시스템의 실제 구현에 많은 고려사항이 되고 있다. 타원곡선 암호시스템에 대한 전력분석 공격은 Coron이 처음으로 [4]에서 소개하였다. [4]에서 Coron은 SPA에 대한 대응방법으로 double-and-add-always 방법을, 그리고 DPA에 대한 대응방법으로 비밀키의 랜덤화, 기저점의 은닉(Blinding), 그리고 사영좌표계(Projective coordinates) 방법을 소개하였다.

Oswald-Aigner는 SPA와 DPA에 대응하는 두 가지 랜덤한 덧셈-뺄셈 체인[18, Randomized Automaton 1, 2]을 제안하였다. 즉, 알고리즘 실행 중 랜덤비트가 생성되어 타원곡선 스칼라 곱셈의 연산 절차가 랜덤하게 연산되도록 알고리즘을 고안하였다. Okeya-Sakurai는 랜덤한 덧셈-뺄셈 체인이 SPA에 취약함을 보였다[15,16]. 또한 Walter는 마코프 연쇄 모델(Markov chains model)을 이용하여 Oswald와 Aigner의 랜덤한 덧셈-뺄셈 체인의 분석방법을 소개하였다[19]. 그러나 Walter의 방법은 비밀키의 일부를 결정하지 못하는 단점이 있다. 즉, 단지 키의 엔트로피를 감소시킬 뿐 비밀키 전체를 탐색할 수 있는 명시적인 공격 알고리즘을 제시하지 못했다. 또한 구현 측면에서 Okeya-Sakurai의 방법보다 복잡하다.

본 논문에서는 Okeya-Sakurai의 공격 알고리즘의 잠재된 두 가지 문제점을 제시한다. 첫번째는 공격자가 비밀키 d를 알아내기 위해서는 d의 길이를 알아야 한다. 공개키 파라메타로 그룹의 위수가 공개되지만, 비밀키는 그룹의 위수보다 작은 값들 중에서 랜덤하게 선택되어지기 때문에 실제로 정확하게 비밀키의 길이를 안다는 가정은 무리이다. 두 번째 문제점으로 Okeya-Sakurai는 무한원점(point at infinity)의 덧셈연산을 간과했다. 일반적으로 이 경우에는 실제 산술 연산이 일어나지 않는다. 따라서 Okeya-Sakurai의 분석방법을 이용하여 비밀키를 정확히 얻기 위해서는 위의 두 가지 문제점이 해결되도록 공격 알고리즘이 수정되어야 한다. 이에 대한 자세한 사항은 Ⅲ절에서 기술하도록 한다.

본 논문에서는 Okeya-Sakurai의 방법과 Walter의 방법과는 다른 새로운 강력한 공격 알고리즘을 제안하며 이때 공격자는 비밀키의 길이에 대한 정보를 필요로 하지 않는다. 본 논문의 공격 알고리즘은 Oswald-Aigner이 제안한 두 개의 랜덤한 덧셈-뺄셈 체인 모두에 적용된다. 제안하는 알고리즘을 사용하면 n개의 AD 수열을 사용하여 비밀키을 완벽하게 복원할 수 있다. 비밀키 길이 $|d|$에 대하여, 에러 확률은 최악의 경우

$$\left[1-\left(0.06\left(\frac{3}{4}\right)^n+0.41\left(\frac{1}{2}\right)^n-0.03\left(\frac{1}{4}\right)^n\right)\right]^{0.03|d|}$$

이다. 또한, 본 논문에서 제안하는 분석방법의 특징은 단순한 버전에 대한 분석 방법이 복잡한 버전에 대한 분석방법으로부터 쉽게 전환 가능하다. 즉, 하나의 공격 알고리즘으로 두 개의 랜덤한 덧셈-뺄셈 체인 연산 방식을 분석할 수 있다. 163비트 비밀키에 공격 알고리즘을 적용하여 구현한 결과, 단순한 형태의 알고리즘에 대해서는 20개의 AD수열로 대략 94%의 확률로 공격이 성공하며 30개의 AD수열로는 대략 99%의 확률로 공격이 성공한다. 또한, 복잡한 형태에 대해서는 40개의 AD수열로 94%의 확률로 70개의 AD수열로는 99%로의 확률로 공격이 성공한다. 30개의 AD 수열을 이용했을 때 99%의 성공확률이 의미하는 바는, 만약 공격자가 30개의 AD 수열로 100번의 공격 시도를 한다면 99번은 정확한 비밀키를 획득할 수 있다는 것이다.

그러므로 본 논문에서는 Oswald-Aigner가 제안한 단순 혹은 복잡한 버전의 랜덤한 덧셈-뺄셈 체인을 이용한 대응 방법은 모두 SPA에 취약함을 제시한다. Ⅳ절에서 [18]의 알고리즘 성질과 본 논문에서 제안하는 강력한 공격 알고리즘에 대해서 언급하며, Ⅴ절에서 구현 결과를 제시한다.

Ⅱ. 랜덤한 덧셈-뺄셈 체인 대응방법

Oswald-Aigner는 SPA와 DPA에 대응하는 두 가지 랜덤한 덧셈-뺄셈 체인 방법을 [18]에서 제안하였다. 본 절에서는 Randomized Automaton 1의 복잡한 형태인 Randomized Automaton 2에 대해서만 언급하기로 하며 두 가지 Randomized Automaton에 대한 자세한 설명은 [18]에 기술되어 있다. 주어진 타원곡선의 기저점을 $\boldsymbol{P}$라 하고 비밀키로 쓰이는 스칼라를 d라 할 때, $d\boldsymbol{P}$ 계산하는 랜덤한 덧셈-뺄셈 체인의 연산은 다음과 같다.

Randomized Automaton 2 (복잡한 형태)

(초기값) P ← O, Q ← P, j ← **0**, 그리고 **State=0**, 여기서 O는 무한원점이다.

- State = 0인 경우,
  - $d_i = 1$인 경우: P←P+Q, Q←2Q, j←j+1; **State=1** 으로 이동,
  - $d_i = 0$인 경우: Q←2Q, j←j+1; **State=0** 으로 이동.

- State = 1인 경우,
  - $j = |d|$인 경우: dP의 결과로 P를 반환,
  - $d_i = 1$이고 난수 $e = 1$인 경우: P←P+Q, Q←2Q, j←j+1; **State=1** 으로 이동,
  - $d_i = 1$이고 난수 $e = 0$인 경우: P←P-Q, Q←2Q, j←j+1; **State=11** 으로 이동,
  - $d_i = 0$인 경우: Q←2Q, j←j+1; **State=0** 으로 이동.

- State = 11인 경우,
  - $j = |d|$인 경우: dP의 결과로 P+Q를 반환,
  - $d_i = 1$이고 난수 $e = 1$인 경우: Q←2Q, j←j+1; **State=11** 으로 이동,
  - $d_i = 1$이고 난수 $e = 0$인 경우: Q←2Q, P←P+Q, j←j+1; **State=1** 으로 이동,
  - $d_i = 0$인 경우: P←P+Q, Q←2Q, j←j+1; **State=110** 으로 이동.

- State = 110인 경우,
  - $d_i = 1$이고 난수 $e = 1$인 경우: P←P+Q, Q←2Q, j←j+1; **State=1** 으로 이동,
  - $d_i = 1$이고 난수 $e = 0$인 경우: P←P-Q, Q←2Q , j←j+1; **State=11** 으로 이동,
  - $d_i = 0$인 경우: Q←2Q, j←j+1; **State =0** 으로 이동.

  ($|d|$: d의 길이, $d = (d_{|d|-1}, \cdots, d_1, d_0)_2$)

Ⅲ. [15,16]에서 Okeya-Sakurai에 의해 제안된 공격방법에 대한 고려

Okeya-Sakurai의 공격 알고리즘은 세 가지 변수를 갱신하면서 실행된다. 즉, 비밀키 스칼라의 비트값 d_i의 결정, AD 수열 S_i들의 pivot 시점 k_1의 결정, 알고리즘 **state**의 결정이다. 그러나 그들의 분석 방법은 다음과 같은 두 가지 잠재된 문제가 있다.

- 공격자는 비밀키 d의 비트 길이 $|d|$를 알아야 한다 :

공격 알고리즘에서 $|d|$의 비트 길이가 (a)단계를 제외하고 항상 사용되었다. 따라서 정확한 $|d|$의 값을 알아야 알고리즘으로부터 키의 값을 찾을 수 있다는 의미이다. 하지만, 공격자가 2배 연산과 덧셈 연산을 구별할 수 있는 능력을 가지고 있다면 $|d|$ 값은 쉽게 얻을 수 있다. 왜냐하면 전체 스칼라 곱셈 과정에서 2배 연산은 모든 비트에서 항상 사용되어진다. 따라서 공격자가 자신이 얻은 AD수열에서 2배 연산의 횟수를 센다면 총 횟수가 바로 d의 비트 길가 된다.

- 무한원점의 덧셈 연산을 간과했다 :

스칼라 곱셈 dP 과정에서 Q=2O, Q=P+O와 같은 연산은 실제 산술 연산이 일어나지 않고 이 경우 점의 값을 레지스터에 할당하는 동작만을 하게 된다. 예를 들어 [15,16]의 분석방법의 문제를 살펴보자. $d=(1001)_2$라 하고 2절의 알고리즘을 수행하면 다음과 같다.

i. $d_0 = 1$일 때는, P←O+P, Q←2P, j←j+1; **state=1** 으로 이동.(2배 연산만 실행)
ii. $d_1 = 0$일 때는, Q←2Q, j←j+1; **state=0** 으로 이동.(2배 연산만 실행)
iii. $d_2 = 0$일 때는, Q←2Q, j←j+1; **state=0** 으로 이동.(2배 연산만 실행)
iv. $d_3 = 1$일 때는, P←P+Q, Q←2Q, j←j+1; **state =1** 으로 이동.(덧셈과 2배 연산이 모두 실행)

이 경우 공격자는 S=DDDAD[1]같은 AD수열을 얻게 된다. 따라서 공격자는 주어진 AD수열로부터 Okeya-Sakurai의 공격 알고리즘을 이용하여 비밀키를 $d=(1000)_2$으로 결정할 것이다. 따라서 무한원점의 덧셈 연산을 고려하지 않는다면 잘못된 비밀

1) A : 덧셈, D : 2배 연산

키 d를 유추하게 된다.

Ⅳ. 랜덤한 덧셈-뺄셈 체인에 대한 새로운 분석방법

SPA 공격에서 공격자는 하나의 스칼라 곱셈의 전력 소비량을 관찰할 수 있음을 가정한다. 공격자의 목표는 스칼라 곱셈의 전력 소모량 파형을 관찰하여 얻은 정보를 이용하여 키를 찾는 것이다. 스칼라 곱셈은 점의 덧셈, 뺄셈 그리고 2배 연산의 수열로 간주된다. 점의 덧셈 연산은 2배 연산과는 다른 전력 소비 패턴을 가진다. 점의 덧셈과 뺄셈은 거의 같은 연산이므로 공격자가 구별하는 것이 불가능하도록 구성될 수 있다. 본 논문에서는 점의 덧셈과 2배 연산은 구별 가능하고, 점의 덧셈과 뺄셈은 구별 불가능하다고 가정한다. 앞으로는 덧셈과 뺄셈 연산 모두를 덧셈(A)이라 간단히 명한다. Oswald-Aigner의 랜덤한 덧셈-뺄셈 체인에서 스칼라 d의 각 비트값 0과 1에 대응하는 덧셈과 2배 연산 패턴이 고정되지 않아서 SPA 공격으로 비밀키를 유도하는 것이 어려워 보였다.

본 논문에서는 전력 소비량 측정에 의한 비트와 연산의 새로운 필요충분조건 관계를 제안한다. 이 관계에 의하여 본 논문은 어떻게 공격자가 전력 소모량측정을 통하여 비밀키 d를 결정할 수 있는지 보인다. 제안하는 공격 알고리즘은 앞에서 언급한 문제점을 해결하였으며 Oswald-Aigner에 의하여 제안된 두개의 알고리즘에 모두 적용이 가능하다는 장점이 있다.

- **표기** : S는 AD수열이며 S는 덧셈(A)과 2배(D) 연산으로 구성된다. $S_i[k]$는 i번째 수열의 k번째 연산(A 또는 D)을 뜻하며 $S_i[k_i,n]$은 i번째 수열의 k_i번째부터 연속된 n개의 값들을 말한다. $|S|$는 A, D 문자의 수이며 비어있는 $S_i[k]$는 $S_i[k]$=NULL이라 한다.

- **공격자의 목적** : 공격자는 타원 곡선 연산을 관찰할 수 있는 능력이 있으며 주어진 AD 수열을 이용하여 AD 수열과 키와의 관계를 유도하는 것이 그의 목적이다. 이러한 관계에서 생성된 정보를 이용하여 공격자는 비밀키 d를 알아내기 위하여 공격 알고리즘을 적용할 수 있다.

1. 랜덤한 덧셈-뺄셈 체인 대응방법이 가지고 있는 특성들

본 소절에서는 Oswald-Aigner가 제안한 랜덤한 덧셈-뺄셈 체인의 특성을 살펴본다. 또한 표 1,2,3으로부터 키의 비트와 AD수열 값의 관계를 유도한다.

- **가정** : 공격자는 랜덤한 덧셈-뺄셈 체인 대응방법이 적용된 암호장비에 주어진 타원곡선의 기저점 P를 입력하고 AD수열을 얻는다. 공격자는 n번의 반복 수행에서 n개의 AD수열을 얻는다. S_i를 n개의 AD 수열 중에서 i번째 AD 수열이라 가정하자. 본 논문에서는 Okeya-Sakurai의 [15,16]에서의 표기법을 따른다.
- **성질 1**. 만약 모든 i_1에 대해 $S_{i_1}[k_{i_1}]$=A, $S_{i_1}[l]$=D $(0 \le l \le k_{i_1}-1)$이고 어떤 i_2, i_3에 대하여 $S_{i_2}[k_{i_2}+2]$=A, $S_{i_3}[k_{i_3}+2]$=D이면 $d_{j-1}=d_j=1$이며, $j>1$인 경우 $d_0=\cdots=d_{j-2}=0$이다.
- **성질 2**. 만약 모든 i_1에 대해 $S_{i_1}[k_{i_1}]$=A, $S_{i_1}[l]$=D $(0 \le l \le k_{i_1}-1)$이고 모든 i_2, i_3 대하여 $S_{i_2}[k_{i_2}+2]$=A 또는 $S_{i_3}[k_{i_3}+2]$=D이면 $d_{j-1}=0$, $d_j=1$이다. $j=2$인 경우 $d_0=1$이며, 그렇지 않은 경우 단 하나의 $d_l=1\,(0 \le l \le j-2)$이고 나머지비트는 모두 0이다.

이 두 가지 성질은 점의 덧셈에서 하나의 점이 무한원점인 경우에서 발생한다. P에 무한원점이 저장되어 있는 경우 다음 비트가 1인 경우 덧셈 연산은 수행되지 않고 단지 2배 연산만 수행된다. 만약 AD 수열에서 A가 처음으로 나타난다면 현재 비트는 $d_j=1$이며, 정확하게 하나만 $d_l=1\,(0 \le l \le j-1)$이고 나머지는 모두 0이다. 이전에 기술된 내용과 성질 3을 이용하면 성질 1과 2는 쉽게 검증된다. 성질 3은 1.1절에 언급되어 있다.

- **정리 1**. 만약 현재 상태가 **state=0** 또는 **state=110**일 때(Randomized Automa-

ton 2의 경우), 키 비트 $d_i=0$일 필요충분조건은 모든 i_1에 대해 $S_{i_1}[k_{i_1}]=$D이다.

▸ **정리 2.** 만약 현재 상태가 **state=0** 또는 **state=110**일 때(Randomized Automaton 2의 경우) 키의 비트 $d_i=1$일 필요충분조건은 모든 i_1에 대해 $S_{i_1}[k_{i_1}]=$A이다. (단 P$\neq O$)

정리 1과 2는 Randomized Automaton 1과 2에 의하여 자명하다. 성질 1, 2와 정리 1, 2는 Randomized Automaton 1과 2에 동시에 적용된다.

표 1은 이전의 연속된 비트가 11 또는 01이며 무한원점이 아닐 때 d_j, d_{j+1}의 계산 후의 AD수열과 AD 수열이 나타날 확률을 나타낸 것이다. 예를 들어 2행 4열의 1/4는 다음을 의미한다.

$$\Pr[DAAD \mid (d_j, d_{j+1}, d_{j-2}, d_{j-1})=(1,1,1,1)]=1/4.$$

표 2에서는 P$\neq O$이고 연속된 비트가 11 또는 01이며 $d_j=1$일 때 AD수열 및 AD 수열이 나타날 확률을 나타낸다. 표 1과 2의 결과는 Randomized Automaton 1과 2에 적용된다.

1.1. Randomized Automaton 2 알고리즘의 성질

▸ **성질 3.** 만약 어떤 i_1, i_2에 대해 $S_{i_1}[k_{i_1}]=$D와

표 1. 연속된 비트가 11 또는 01일 때의 AD수열($j-1$번째 state는 전부 1이 아니다.)

d_{j-2}, d_{j-1}	d_j, d_{j+1}	AD sequence	Probability
11 or 01	11	DAAD	1/4
		DDA	1/8
		DD	1/8
		ADDA	1/8
		ADD	1/8
		ADAD	1/4
	10	DAD	1/2
		ADAD	1/4
		ADD	1/4
	01	ADAD	1/2
		DAD	1/2
	00	ADD	1/2
		DD	1/2

표 2. 연속된 비트가 11 또는 01이며 $d_i=1$일 때 AD수열

d_{j-2}, d_{j-1}	d_j, d_{j+1}, d_{j+2}	AD sequence	Probability
11 or 01	111	ADDAAD	1/8
		ADDDA	1/16
		ADDD	1/16
		ADADDA	1/16
		ADADD	1/16
		ADADAD	1/8
		DAADDA	1/16
		DAADD	1/16
		DAADAD	1/8
		DDAAD	1/8
		DDDA	1/16
		DDD	1/16
	110	ADDAD	1/4
		ADADAD	1/8
		ADADD	1/8
		DAADAD	1/8
		DAADD	1/8
		DDAD	1/4

$S_{i_2}[k_{i_2}]=$A이면 이전의 두 비트는 $d_{j-2}=d_{j-1}=1$이거나 또는 $j-2$번째 상태가 **state=110**일 때 $d_{j-2}=0\neq, d_{j-1}=1$이다.

성질 3은 $d_j=1$일 때 이전에 연속된 두 비트가 1이거나 이전의 연속된 세 비트가 110일 때 공격자가 AD수열에서 다음 비트를 결정할 수 없음을 함축한다. 첫 번째 경우는 다음과 같다.

$$\Pr[D_j \mid (d_{j-2}, d_{j-1}, d_j)=(1,1,1)] = \Pr[A_j \mid (d_{j-2}, d_{j-1}, d_j)=(1,1,1)]=1/2,$$
$$\Pr[D_j \mid (d_{j-2}, d_{j-1}, d_j)=(1,1,0)] = \Pr[A_j \mid (d_{j-2}, d_{j-1}, d_j)=(1,1,0)]=1/2,$$

이때 AD 수열의 원소 D_j(또는 A_j)는 d_j의 j번째 비트에 대응한다.

두 번째 경우는 다음과 같다.

$$\Pr[D_{j+1} \mid (d_{j-3}, d_{j-2}, d_{j-1}, d_j, d_{j+1})=(1,1,0,1,1)]=1/3$$
$$\Pr[D_{j+1} \mid (d_{j-3}, d_{j-2}, d_{j-1}, d_j, d_{j+1})=(1,1,0,1,1)]=2/3$$
$$\Pr[D_{j+1} \mid (d_{j-3}, d_{j-2}, d_{j-1}, d_j, d_{j+1})=(1,1,0,1,1)]=2/3$$
$$\Pr[D_{j+1} \mid (d_{j-3}, d_{j-2}, d_{j-1}, d_j, d_{j+1})=(1,1,0,1,1)]=1/3$$

그러나 만약 이전의 두 비트가 모두 1이 아니거나 이전의 두 비트가 $d_{j-2}=0$, $d_{j-1}=1$이며 $j-2$번째 상태가 state=110이 아닐 때 다음 비트는 정리 1과 2에 의하여 결정된다. 표 3과 정리 1,2에 의하여 성질 3은 항상 만족한다. 이 경우의 확률은 $(2/3)^n$보다 작다.

▶ **정리 3.** 만약 어떤 i_1, i_2에 대해 $S_{i_1}[k_{i_1}]$=D와 $S_{i_2}[k_{i_2}]$=A이기 위한 필요충분조건은 이전의 두 비트는 $d_{j-2}=d_{j-1}=1$이거나 또는 $j-2$번째 상태가 **state=110**일 때 이전의 두 비트는 $d_{j-2}=0$,$d_{j-1}=1$일 때이다. 필요조건에 대한 오류 확률은 $(2/3)^n$보다 작다.

표 3의 결과는 $P\neq O$이고 이전의 연속된 비트가 110일 때 d_j, d_{j+1}의 계산 후 AD 수열과 AD 수열이 나타날 확률을 나타낸다.

1.2. 제안하는 공격 알고리즘

공격자는 랜덤한 덧셈-뺄셈 체인 대응방법이 적용된 암호화 장비에 타원곡선의 점을 입력하고 AD수열을 얻는다. 공격자는 n번의 반복 수행에서 n개의 AD수열을 얻고, 아래와 같이 비밀키 d를 획득한다.

본 논문에서 제안하는 알고리즘은 두 개의 변수를 갱신하면서 실행된다. 하나는 비밀키 d의 비트 값을 결정하는 것이고, 또 다른 하나는 S값의 위치 k_i를 결정하는 것이다. 제안하는 공격 알고리즘은 Okeya -Sakurai의 것과 달리 state의 변수가 필요하지 않다.

표 3. 연속된 비트가 110일때 AD수열

$d_{j-3}, d_{j-2}, d_{j-1}$	d_j, d_{j+1}	AD sequence	Probability
110	11	ADAD	2/3
		ADDA	1/6
		ADD	1/6
	10	ADD	2/3
		ADAD	1/3
	01	DAD	1
	00	DD	1

1.2.1. Randomized Automaton 2의 공격 알고리즘

Randomized Automaton 2의 공격 알고리즘은 다음 4가지 알고리즘으로 구성된다.

(초기화): $k_i=0$, $j=0$; Key Finding Algorithm을 실행한다. k_i는 AD수열의 위치변수이고 j는 비밀키의 비트와 관련이 있다.

[Key Finding Algorithm I]

- 만약 어떤 i_1에 대하여 $S_{i_1}[k_{i_1}]$=NULL인 경우 : $d_j=1$, 나머지 비트는 0으로 설정되며 알고리즘을 종료한다.
- 만약 모든 i_1에 대하여 $S_{i_1}[k_{i_1}]$=A이고 어떤 i_2에 대하여 $S_{i_2}[k_{i_2}+2]$=NULL인 경우;$d_j=1$이고 단 하나만 $d_l=1(0\leq l\leq j-1)$이며 나머지는 모두 0이다. j번의 조사를 통해 d_l을 결정하고 알고리즘을 종료한다.
- 만약 모든 i_1에 대하여 $S_{i_1}[k_{i_1}]$=D인 경우; $j\leftarrow j+1$, 모든 i에 대해 $k_i\leftarrow k_i+1$, Key Finding Algorithm I 수행한다.
- 만약 모든 i_1에 대하여 $S_{i_1}[k_{i_1}]$=A이고
  - 만약 어떤 i_2와 i_3에 대하여 $S_{i_2}[k_{i_2}+2]$=A이고 $S_{i_3}[k_{i_3}+2]$=D인 경우 : $d_{j-1}=d_j=1$, $j>1$인 경우 $d_0=\cdots=d_{j-2}=0$, $j\leftarrow j+1$, 모든 i에 대해 $k_i\leftarrow k_i+2$, Key Finding Algorithm III 수행한다.
  - 그렇지 않은 경우 : $d_{j-1}=0, d_j=1$. 만약 $j=2$이면 $d_0=1$, 그렇지 않으면 단 하나만 $d_l=1(0\leq l\leq j-2)$이며 나머지는 모두 0이다. $j-1$번의 전수 검사를 통해 d_l을 결정, $j\leftarrow j+1$, 모든 i에 대해 $k_i\leftarrow k_i+2$, Key Finding Algorithm II 수행한다.

[Key Finding Algorithm II]

- 만약 어떤 i_1에 대하여 $S_{i_1}[k_{i_1}]$=NULL인 경우; $d=\sum_{k=0}^{j-1} d_k 2^k$ 이며 알고리즘을 종료한다.
- 만약 모든 i_1에 대하여 $S_{i_1}[k_{i_1}]$=D인 경우; $d_j=0$, $j\leftarrow j+1$; 모든 i에 대해 $k_i\leftarrow k_i+1$, Key Finding Algorithm II 수행한다.

• 그렇지 않은 경우;

– 만약 모든 i_1에 대하여 $S_{i_1}[k_{i_1}]$=A인 경우 : $d_j=1$, $j\leftarrow j+1$, 모든 i에 대해 $k_i\leftarrow k_i+2$, Key Finding Algorithm II 수행한다.

– 그렇지 않은 경우 : Key Finding Algorithm III 수행한다.

[Key Finding Algorithm III]

• 만약 어떤 i_1에 대하여 $S_{i_1}[k_{i_1}]$=NULL인 경우 : $d=\sum_{k=0}^{j-1} d_k 2^k$이며 알고리즘을 종료한다.

• 만약 어떤 i_2에 대하여 $S_{i_2}[k_{i_2}+2]$=NULL인 경우 : $d_j=1$이고 $d=\sum_{k=0}^{j} d_k 2^k$이며 알고리즘을 종료한다.

• 만약 모든 i_1, i_2에 대해 $S_{i_1}[k_{i_1},3]$=DAA이고 $S_{i_2}[k_{i_2}+5]$=NULL인 경우 : $d_j=d_{j+1}=1$이고 $d=\sum_{k=0}^{j} d_k 2^k$이며 알고리즘을 종료한다.

• 만약 어떤 i_1에 대해 $S_{i_1}[k_{i_1},3]$=DAA인 경우;

– 만약 어떤 i_2에 대해 $S_{i_2}[k_{i_2},4]$=DDAD 또는 $S_{i_2}[k_{i_2},5]$=ADDAD인 경우; $d_j=d_{j+1}=1$, $d_{j+2}=0$, $j\leftarrow j+3$, 모든 i에 대해 $S_{i_1}[k_{i_1},4]$=DDAD이면 $k_i\leftarrow k_i+4$, 모든 i에 대해 $S_i[k_i,5]$=DAADD, ADDAD, 또는 ADADD이면 $k_i\leftarrow k_i+5$, 다른 경우이면 $k_i\leftarrow k_i+6$, Key Finding Algorithm IV 수행한다.

– 그렇지 않은 경우 : $d_j=d_{j+1}=1(d_{j+2}=1)$, $j\leftarrow j+2$: 만약 모든 i에 대해 $S_{i_1}[k_{i_1},3]$=DDD이면 $k_i\leftarrow k_i+2$, 만약 모든 i에 대해 $S_{i_1}[k_{i_1},3]$=DDA 또는 $S_{i_1}[k_{i_1},4]$=ADDD이면 $k_i\leftarrow k_i+3$ 다른 경우이면 $k_i\leftarrow k_i+4$ Key Finding Algorithm III 수행한다.

• 그렇지 않은 경우 :

– 만약 어떤 i_1에 대해 $S_{i_1}[k_{i_1},3]$=DAD인 경우;

◦ 만약 어떤 i_2에 대해 $S_{i_2}[k_{i_2},3]$=ADD인 경우; $d_j=1$, $d_{j+1}=0$, $j\leftarrow j+2$; 모든 i에 대해 $S_i[k_i,4]$=ADAD이면 $k_i\leftarrow k_i+4$, 그렇지 않으면 $k_i\leftarrow k_i+3$, 부분 Key Finding Algorithm IV 수행한다.

◦ 그렇지 않은 경우;$d_j=0$, $d_{j+1}=1$, $j\leftarrow j+2$; 모든 i에 대해 $S_{i_1}[k_{i_1},3]$=DAD이면 $k_i\leftarrow k_i+3$, 그렇지 않으면 $k_i\leftarrow k_i+4$; Key Finding Algorithm III 수행한다.

– 아닌 경우 $d_j=0$, $d_{j+1}=0$ $j\leftarrow j+2$; 모든 i에 대해 $S_i[k_i,2]$=DD이면 $k_i\leftarrow k_i+2$, 그렇지 않으면 $k_i\leftarrow k_i+3$; Key Finding Algorithm II 수행한다.

[Key Finding Algorithm IV]

• 만약 어떤 i_1에 대하여 $S_{i_1}[k_{i_1}]$=A인 경우 ; $d_j=1$, $j\leftarrow j+1$; 모든 i에 대해 $k_i\leftarrow k_i+2$, Key Finding Algorithm III 수행한다.

• 그렇지 않은 경우 ; $d_j=0$, $j\leftarrow j+1$; 모든 i에 대해 $k_i\leftarrow k_i+1$, Key Finding Algorithm II 수행한다.

1.2.2. Randomized Automaton 2의 공격 알고리즘의 증명

[Validity of the Key Finding Algorithm I]

• Key Finding Algorithm에서 공격 알고리즘이 종료되는 경우는 두 가지 경우가 있다. 첫 번째 경우는 $d_{|d|-1}=1$이고 $d_0=\cdots=d_{|d|-2}=0$ 인 경우이며 이 확률은 $(1/2)^{|d|}$이다. 두 번째 경우는 $d_{|d|-1}=1$이고, $d_l=1\,(0\leq l\leq |d|-2)$이고 나머지 비트가 모두 0인 경우이며 이 확률은 $(|d|-1)\cdot(\frac{1}{2})^{|d|-1}$이다. 전자의 경우 d는 오류 없이 결정되지만 후자의 경우 $|d|-1$번의 전수조사를 통해 d_l을 찾아야 한다.

• 만약 모든 i_1에 대하여 $S_{i_1}[k_{i_1}]$=A이고, 어떤 i_2, i_3에 대하여 $S_{i_2}[k_{i_2}+2]$=D이고 $S_{i_3}[k_{i_3}+2]$=D이면; $d_{j-1}=d_j=1$이며, 또한 $j>1$이면 $0\leq l\leq k_{i_1}-1$인 모든 l에 대해 $S_{i_1}[l]$=D인 사실과 성질 1에 의해 $d_0=\cdots=d_{j-2}=0$이다. $j+1$번째 단계에서 AD 수열 S_i의 위치는 |AD|=2이므로 k_i+2이 된다. 이 경우 오류확률은 정리 3에 의해 $(2/3)^n$이다.

◦ 모든 i_1, i_2, i_3에 대하여 $S_{i_1}[k_{i_1}]$=A, $S_{i_2}[k_{i_2}+2]$=D, $S_{i_3}[k_{i_3}+2]$=D이면 $d_{j-1}=0, d_j=1$이다.

만약 $j=2$이면 $d_0=1$이다. $j>2$인 경우에는 $0\le \ell \le k_{i_1}-1$인 모든 l에 대해 $S_{i_1}[l]=$D인 사실과 성질 2에 의해서 $d_l=1\,(0\le \ell \le |d|-2)$이고 나머지 비트는 모두 0 이다.

[Validity of the Key Finding Algorithm II]

- 이전 두 비트가 $d_{j-2}=d_{j-1}=0$ 또는 $d_{j-2}=0$, $d_{j-1}=1$일 때(모든 AD 수열에 대하여, $j-1$번째 상태가 1인 경우), Key Finding Algorithm II이 호출된다.
- 만약 모든 i_1에 대하여 $S_{i_1}[k_{i_1}]=$D 인 경우 ; 정리 1에 의하여 $d_j=0$이다. |D|=1이므로 $j+1$번째 단계에서 AD 수열 S_i의 위치는 k_i+1이다.
- 그렇지 않은 경우 ;
  - 모든 i_1에 대하여 $S_{i_1}[k_{i_1}]=$D 인 경우 ; 정리 2로부터 $d_j=1$이다. |AD|=2이므로 $j+1$번째 단계에서 AD 수열 S_i의 위치는 k_i+2이다.
  - 그렇지 않은 경우 ; 이전 두 비트는 모두 1 이다.

[Validity of the Key Finding Algorithm III]

- 이전 두 비트가 $d_{j-2}=d_{j-1}=1$ 또는 $d_{j-2}=0$, $d_{j-1}=1$일 때 (모든 AD 수열에 대하여, $j-1$번째 상태가 1이 아닌 경우), Key Finding Algorithm III이 호출된다.
- Key Finding Algorithm III이 종료되는 경우는 세 가지가 있다. 첫 번째 경우는 자명하며, 두 번째는 $j=|d|-1$ 일 때이며, 세 번째는 $j=|d|-2$, $d_j=1$인 경우이다.
- 만약 $j=|d|-1$인 경우 ; 어떤 i_1에 대하여 $(1\le i_1\le n)$ $S_{i_1}[k_{i_1}+2]=$NULL이다. 따라서 최상위 비트 $d_j=1$이며 알고리즘은 종료된다.
- 만약 $j=|d|-2$, $d_{|d|-2}=1$인 경우 ; 최상위 비트는 $d_j=d_{j+1}=1$ 이다. 따라서 $S_{i_1}[k_{i_1},3]=$DAA인 i_1이 존재하고, 모든 i_2에 대하여 $S_{i_2}[k_{i_2}+5]=$NULL이다. 이 경우 오류확률은 표 1에 의하여 $(3/4)^n$이다.
- 이전의 연속된 두 비트가 11 또는 01 ($j-1$번째 상태가 모두가 1이 아닌 경우) 이고 d_j=1인 경우 ;
  - $S_{i_1}[k_{i_1},3]=$DAA인 i_1과 $S_{i_2}[k_{i_2},4]=$DDAD 또는 $S_{i_2}[k_{i_2},5]=$ADDAD 인 i_2가 존재하는 경우 ; 표 2로부터 $d_j=d_{j+1}=1$, $d_{j+2}=0$ 이다. 이 경우 오류확률은 주어진 n개의 AD 수열에 대하여 $(3/4)^n+(1/2)^n-(1/4)^n$ 이 된다.
  - $S_{i_1}[k_{i_1},3]=$DAA 인 i_1이 존재하고 모든 i_2, i_3에 대하여 $S_{i_2}[k_{i_2},4]\ne$DDAD, $S_{i_2}[k_{i_2},5]\ne$ADDAD 인 경우 ; 표 2로부터 $d_j=d_{j+1}=1$, $(d_{j+2}=1)$ 이며 k_i값 또한 결정된다. 이 경우 오류확률은 주어진 n개의 AD 수열에 대하여 $(3/4)^n$ 이 된다.
  - 모든 i_1에 대하여 $S_{i_1}[k_{i_1},3]\ne$DAA 이고 $S_{i_2}[k_{i_2},3]=$DAD 과 $S_{i_3}[k_{i_3},3]=$ADD 을 만족하는 i_2와 i_3가 존재하는 경우 ; 표 1로부터 $d_j=1$, $d_{j+1}=0$ 이며 k_i 값 또한 결정된다. 이 경우 오류확률은 n 개의 AD 수열에 대하여 $(3/4)^n+(1/2)^n-(1/4)^n$이 된다.
  - 모든 i_1에 대하여 $S_{i_1}[k_{i_1},3]\ne$DAA 이고 $S_{i_2}[k_{i_2},3]=$DAD 인 i_2가 존재하며 모든 i_3에 대하여 $S_{i_3}[k_{i_3},3]\ne$ADD 인 경우; 표 1로부터 $d_j=0$, $d_{j+1}=1$ 이며 k_i 값 또한 결정된다. 이 경우 오류확률은 n개의 AD 수열에 대하여 $(1/2)^n$이 된다.
  - 모든 i_1, i_2에 대하여 $S_{i_1}[k_{i_1},3]\ne$DAA이고 $S_{i_2}[k_{i_2},3]\ne$DAD 인 경우 ; 표 1로부터 $d_j=d_{j+1}=0$이며 k_i 값 또한 결정된다. 이 경우 오류확률 없이 키 비트가 결정된다.

[Validity of the Key Finding Algorithm IV]

- 이전 세 비트가 $d_{j-3}=d_{j-2}=1$, $d_{j-1}=0$일 때, Key Finding Algorithm IV이 호출된다.
- $S_{i_1}[k_{i_1}]=$A을 만족하는 i_1이 존재하는 경우 ; 표 3으로부터 $d_j=1$이며 k_i 값 또한 결정된다. 이 경우 오류확률 없이 키 비트가 결정된다.
- $S_{i_1}[k_{i_1}]=$D 을 만족하는 i_1이 존재하는 경우 ; 표 3 으로부터 $d_j=0$이며 k_i 값 또한 결정된다. 이 경우 오류확률 없이 키 비트가 결정된다.

1.2.3. 공격 알고리즘의 오류확률

공격 알고리즘이 종료될 경우를 제외하고 Key Finding Algorithm Ⅲ만이 키 비트 결정시 오류가 발생한다. Key Finding Algorithm Ⅲ에서 각 경우에 대한 오류확률은 다음과 같이 계산된다.

$$\begin{cases} (3/4)^n+(1/2)^n-(1/4)^n & \text{if } (d_j,d_{j+1},d_{j+2})=(1,1,0),\\ (3/4)^n & \text{if } (d_j,d_{j+1},d_{j+2})=(1,1,1),\\ (3/4)^n+(1/2)^n-(1/4)^n & \text{if } (d_j,d_{j+1})=(1,0),\\ (1/2)^n & \text{if } (d_j,d_{j+1})=(0,1),\\ 0 & \text{if } (d_j,d_{j+1})=(0,0). \end{cases}$$

예를 들어 $(d_j,d_{j+1},d_{j+2})=(1,1,0)$인 경우 오류확률은 다음과 같이 계산된다.

$$(3/4)^n=\Pr[\forall i_1, S_{i_1}[k_{i_1},3]\neq DAA],$$

$$(1/2)^n=\Pr[\forall i_2, S_{i_2}[k_{i_2},4]\neq DDAD \;\&\; S_{i_2}[k_{i_2},5]\neq ADDAD],$$

$$(1/4)^n=\Pr[\forall i, S_i[k_i,3]\neq DAA \;\&\; S_i[k_i,4]\neq DDAD \;\&\; S_i[k_i,5]\neq ADDAD].$$

따라서 Key Finding Algorithm Ⅲ에서 총 오류확률 P 는 다음과 같다.

$$\begin{aligned} P :=\; & p_1[(\tfrac{3}{4})^n+(\tfrac{1}{2})^n-(\tfrac{1}{4})^n]+p_2(\tfrac{3}{4})^n \\ & +p_3[(\tfrac{3}{4})^n+(\tfrac{1}{2})^n-(\tfrac{1}{4})^n]+p_4(\tfrac{1}{2})^n \\ =\; & (p_1+p_2+p_3)(\tfrac{3}{4})^n+(p_1+p_3+p_4)(\tfrac{1}{2})^n \\ & -(p_1+p_3)(\tfrac{1}{4})^n \end{aligned} \tag{1}$$

여기서 p_i $(1\leq i\leq 5)$ 들은 Key Finding Algorithm Ⅲ이 각 상황에 경유할 확률을 말한다. L을 공격 알고리즘 실행 중 Key Finding Algorithm Ⅲ 이 경유되는 평균횟수라고 할 때, 제안하는 공격 알고리즘의 성공확률은 $(1-P)^L$이 된다. n 이 커짐에 따라 $(1-P)^L\to 1$임에 주목하자.

Remark 1. Randomized Automaton 2에 대한 제안하는 공격 알고리즘은 Randomized Auto maton 1에도 쉽게 적용 가능하다. 즉, Key Find ing Algorithm Ⅳ와 Key Finding Algorithm Ⅳ으로 진입하는 path를 제거하고 Key Finding Algorithm Ⅲ에서 생성되는 몇몇 path를 변경하면 Randomized Automaton 2에 대한 공격 알고리즘은 Randomized Automaton 1에 대한 공격으로 변환된다.(그림 1 참조) 따라서 제안하는 분석방식은 하나의 공격 알고리즘을 이용하여 Random ized Automaton 1,2를 동시에 분석할 수 있는 장점이 있다.

1.2.4. 공격 알고리즘 예제

본 소절에서는 Randomized Automaton 2에 대한 공격 알고리즘을 적용한 예를 들어 본다. 공격자가 다음 5개의 AD 수열을 획득하였다고 가정하자.

S_1=DDADADDADADADDAADADDADDAD,
S_2=DDADADDADADDDDADDADDAD,
S_3=DDADADDADADDDAADDDADDAD,
S_4=DDADADADADADADADADADDADDAD,
S_5=DDADADADADDAADADADDDADDAD.

공격자는 다음의 단계를 거쳐 비밀키 d을 찾게 된다.

STEP 1. (초기화) $k_i=0$, $j=0$으로 설정 : Key Finding Algorithm Ⅰ으로 간다.

STEP 2. 모든 $1\leq i\leq 5$에 대하여 $S_i[0]$=D이기 때문에, $j=1$, $k_i=1$ $(1\leq i\leq 5)$으로 설정 : Key Finding Algorithm Ⅰ으로 간다.

STEP 3. 모든 $1\leq i\leq 5$에 대하여 $S_i[1]$=D이기 때문에, $j=2$, $k_i=2(1\leq i\leq 5)$으로 설정: Key Finding Algorithm Ⅰ으로 간다.

STEP 4. 모든 $1\leq i\leq 5$에 대하여 $S_i[2]$=A, $S_i[4]$=A이기 때문에, $d_0=1$, $d_1=0$, $d_2=1$, $j=3$, $k_i=4$ $(1\leq i\leq 5)$으로 설정 : Key Find ing Algorithm Ⅱ으로 간다.

STEP 5. 모든 $1\leq i\leq 5$에 대하여 $S_i[4]$=A이기 때문에, $d_3=1$, $j=4$, $k_i=6$ $(1\leq i\leq 5)$으로 설정 : Key Finding Algorithm Ⅱ으로 간다.

STEP 6. $S_1[6]$=D, $S_4[6]$=A이기 때문에, Key

Finding Algorithm Ⅲ으로 간다.

STEP 7. 모든 $1 \le i \le 5$에 대하여 $S_i[6,3] \neq$DAA, $\neq$ADD이고, $S_1[6,3]$=DAD이기 때문에, $d_4=0$, $d_5=1$, $j=6$, $(k_1, k_2, k_3, k_4, k_5)=(9,9,9,10,10)$으로 설정 ; Key Finding Algorithm Ⅲ으로 간다.

STEP 8. 모든 $1 \le i \le 5$에 대하여 $S_i[k_i,4] \neq$DDAD, $S_5[10,3]$=DAA이기 때문에, $d_6=d_7=1$ ($d_8=1$), $j=8$, $(k_1, k_2, k_3, k_4, k_5)=(13,12,12,14,14)$으로 설정 ; Key Finding Algorithm Ⅲ으로 간다.

STEP 9. $S_1[13,3]$=DAA, $S_2[12,4]$=DDAD이기 때문에, $d_8=d_9=1$, $d_{10}=0$, $j=11$, $(k1,k2,k3,k4,k5)=(19,16,17,20,19)$으로 설정 ; Key Finding Algorithm Ⅳ으로 간다.

STEP 10. $S_1[19]$=D이기 때문에, $d_{11}=0$, $j=12$, $(k_1, k_2, k_3, k_4, k_5)=(20,17,18,21,20)$으로 설정 ; Key Finding Algorithm Ⅱ으로 간다.

STEP 11. 모든 $1 \le i \le 5$에 대하여 $S_i[k_i]$=A이기 때문에, $d_{12}=1$, $j=13$, $(k_1, k_2, k_3, k_4, k_5)=(22,19,20,23,22)$으로 설정 ; Key Finding Algorithm Ⅱ으로 간다.

STEP 12. 모든 $1 \le i \le 5$에 대하여 $S_i[k_i]$=D이기 때문에, $d_{13}=0$, $j=14$, $(k_1, k_2, k_3, k_4, k_5)=(23,20,21,24,23)$으로 설정 ; Key Finding Algorithm Ⅱ으로 간다.

STEP 13. 모든 $1 \le i \le 5$에 대하여 $S_i[k_i]$=A이기 때문에, $d_{14}=1$, $j=16$, $(k_1, k_2, k_3, k_4, k_5)=(25,22,23,26,25)$으로 설정 ; Key Finding Algorithm Ⅱ으로 간다.

STEP 14. $S_1[25]$=NULL이기 때문에, d=(10 1001111101101)2으로 설정한다.

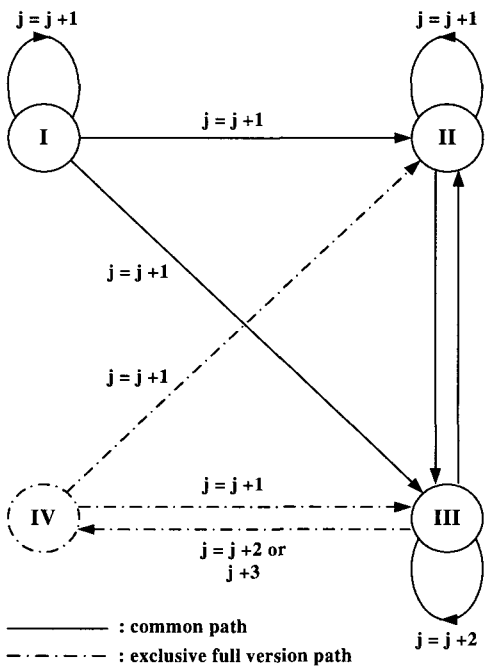


그림 1. 공격 Automaton

Ⅴ. 구현결과와 대응방법

1. 구현결과

본 논문의 공격 알고리즘을 Randomized Automaton 1과 2에 적용하였다. 표준에서 제안된 163 비트에 적용한 구현 결과는 그림 1과 같다. 그림 1로부터 Automaton 1은 AD수열 20개와 30개로 각각 대략 94%와 99%의 확률로 공격이 성공한다. 그리고 복잡한 오토마타 2는 AD수열 40개와 70개로 각각은 대략 94%와 99%의 확률로 공격이 성공한다. 예를 들어 AD수열 30개로의 99%의 성공 확률은 공격자가 같은 비밀키에 대하여 AD수열 30개로 100번을 반복하였을 때 대략 99번 비밀키를 검출할 수 있음을 뜻한다. 식 (1)에서 확률 p_i 값은 시뮬레이션 결과 $p_1=0.02$, $p_2=0.04$, $p_3=0.01$, $p_4=0.38$, $p_5=0.53$으로 나타났으며, 163 비트 비밀키에 대하여 $L=50$ 이라는 결과를 얻었다. 시뮬레이션에 의해 얻은 값들로부터 20개의 AD 수열에 대하여 공격 알고리즘 성공 확률은 0.98이 된다.

2. 대응 방법

SPA를 방어하는 방법은 크게 double-and-add-always 방법과 덧셈과 두 배 연산의 통합된 공식을 사용하는 방법 두 가지가 있다. 예를 들어 Hesse 또는 Jacobi 형태의 타원곡선에서 덧셈과 2배 연산을 같은 공식으로 구성이 가능하다[6,11]. double-and-add-always 방법의 예는 Coron의 방법[4], Montgomery 형태 타원곡선에서의 덧셈 체인 방법[12], 임의의 타원곡선에서의 Montgomery 방식 덧셈 체인 방법이 있다.

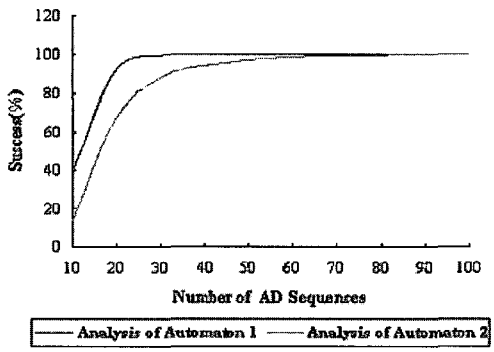


그림 2. Randomized Automaton 1과 2에 대한 공격 알고리즘의 성공 확률 비교

DPA 공격을 방어하기 위해서, Coron은 비밀키 마스킹, 기저점 블라인딩 방식, 사영좌표를 이용한 기저점의 랜덤한 표현 방식과 같이 타원곡선 파라미터가 랜덤화 시켰다[4]. 또한 Joye-Tymen의 랜덤한 타원곡선 동형사상이나 랜덤한 유한체 동형사상을 이용하면, 스칼라 곱셈의 연산을 랜덤한 대상에서 계산되어 DPA 공격을 방어할 수 있게 된다[7].

참 고 문 헌

[1] 안만기, 하재철, 이훈재, 문상재, "타원곡선 암호시스템에서 랜덤 m-ary 방법을 사용한 전력분석 공격의 대응방법," 정보보호학회논문지, 13권 3호, 35-43, 2003.

[2] 임채훈, "부가채널 공격에 안전한 효율적인 타원곡선 상수배 알고리즘," 정보보호학회논문지, 12권 4호, pp. 99-83, 2002.

[3] 하재철 , 곽동진, 문상재, "Folding 기법을 이용한 전력분석 공격에 대응하는 고속 스칼라 곱셈," 정보보호학회논문지, 13권 3호, pp. 57-64, 2003.

[4] J. S. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Crytosystems," In Workshop on Cryptograp-hic Hardare and Embedded Systems (CHES 1999), LNCS 1717, pp. 292-302, 1999.

[5] L. Goubin, "A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems," Public Key Cryptography (PKC 2003), LNCS 2567, pp. 199-211, 2003.

[6] M. Joye and J. Quisquater, "Hessian elliptic curves and side-channel attacks," In Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001), LNCS 2162, pp. 402- 410, 2001.

[7] M. Joye and C. Tymen, "Protections against differential analysis for elliptic curve cryptography," In Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001), LNCS 2162, 377-390, 2001.

[8] N. Koblitz, "Elliptic curve cryptosystems," In Mathematics of Computation, volume 48, pp. 203-209, 1987.

[9] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," In Advances in Cryptology-CRYPTO 1996, LNCS 1109, pp. 104-113, 1996.

[10] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," In Advances in Cryptology-CRYPTO 1999, LNCS 1666, pp. 388-397, 1999.

[11] P. Liardet and N. Smart, "Preventing SPA/DPA in ECC systems using the Jacobi form," In Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001), LNCS 2162, pp. 391-401, 2001.

[12] V.S. Miller, "Use of elliptic curves in cryptography," In Advances in Cryptology-CRYPTO 1985, LNCS 218, pp.

417-426, 1986.

[13] F. Morain and J. Olivos, "Speeding up the computation on an elliptic curve using addition-subtraction chains," Inform Theory Appl., vol 24, pp. 531-543, 1990.

[14] K. Okeya, K.Sakurai, "Power analysis breaks elliptic curve cryptosystems even secure against the timing attack," Indocrypt 2000, LNCS 1977, pp. 178- 190, 2000.

[15] K. Okeya, K. Sakurai, "On Insecurity of the Side Channel Attack Countermeasure Using Addition-Subtraction Chains under Distinguishability between Addition and Doubling," Information Security and Privacy (ACISP 2002), LNCS 2384, pp. 420-435, 2002.

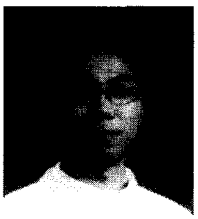
[16] K. Okeya, K. Sakurai, "A Multiple Power Analysis Breaks the Advanced Version of the Randomized Addition-Subtraction Chains Countermeasure against Side Channel Attacks," to appear in the proceedings of 2003 IEEE Information Theory Workshop.

[17] K. Okeya, H. Kurumatani and K. Sakurai, "Elliptic curves with the Montgomery form and their cryptographic applications," Public Key Cryptography (PKC 2000), LNCS 1751, pp. 446-465, 2000.

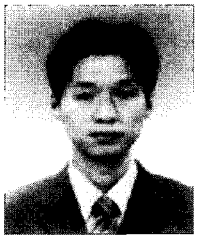
[18] E. Oswald, M. Aigner, "Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks," In Workshop on Crypto graphic Hardware and Embedded Systems (CHES 2001), LNCS 2162, pp. 39-50, 2001.

[19] C.D. Walter, "Security Constraints on the Oswald-Aigner Exponentiation Algorithm," Cryptology ePrint Archive, Report 2003/013, 2003. http://eprint.iacr.org/.

〈著者紹介〉



한 동 국 (Dong-Guk Han) 정회원
1999년 2월 : 고려대학교 수학과 학사
2002년 2월 : 고려대학교 수학과 석사
2002년 3월~현재 : 고려대학교 정보보호대학원 박사 과정
2004년 4월~현재 : 일본 큐슈대학에 교환 연구원
〈관심분야〉 정수론, 공개키 암호, CMVP, 부채널 공격.



장 남 수 (Nam-Su Chang) 학생회원
2002년 2월 : 서울시립대학교 수학과 학사
2002년 3월~현재 : 고려대학교 정보보호대학원 석사 과정
〈관심분야〉 공개키 암호 알고리즘, 무선LAN 기술, 암호칩 설계 기술

장 상 운 (Sang-Woon Jang) 정회원
2002년 2월 : 고려대학교 수학과 이학사
2004년 2월 : 고려대학교 정보보호대학원 석사
2004년 2월~현재 : 국가보안기술연구소 연구원
〈관심분야〉 공개키 암호 알고리즘, 부채널 공격, 암호 분석

임 종 인 (Jong-in Lim) 정회원
1980년 2월 : 고려대학교 수학과 학사
1982년 2월 : 고려대학교 수학과 석사
1986년 2월 : 고려대학교 수학과 박사
1999년 2월~현재 : 고려대학교 자연과학대학 정교수, 한국통신정보보호학회 편집위원장, 고려대학교 정보보호대학원 원장, 고려대학교 정보보호기술연구센터 센터장
〈관심분야〉 블록 암호 및 스트림 암호의 분석 및 설계, 암호 프로토콜, 공개키 암호 분석, 정보보호정책