情報保護學會論文誌
第 21 卷 第 3 號, 2011. 6

아핀좌표를 사용하는 페어링 연산의 Miller 알고리듬에 대한 효과적인 오류주입공격*

배 기 석,[1†] 박 제 훈,[2] 손 교 용,[1] 하 재 철,[3] 문 상 재[1‡]
[1]경북대학교, [2]국방기술품질원, [3]호서대학교

Efficient Fault Injection Attack to the Miller Algorithm in the Pairing Computation using Affine Coordinate System*

KiSeok Bae,[1†] JeaHoon Park,[2] Gyoyong Sohn,[1] JaeCheol Ha ,[3] SangJae Moon[1‡]
[1]Kyungpook National University, [2]DTaQ, [3]Hoseo University

요 약

ID 기반 암호시스템의 구현을 위한 Weil, Tate, Ate와 같은 페어링 연산 기법에서는 밀러 알고리듬이 사용된다. 본 연구에서는 밀러 알고리듬에 대한 오류 공격의 하나인 Mrabet의 방법을 분석하여 타원곡선을 표현하는 가장 기본적인 좌표계인 아핀좌표계에서의 효과적인 오류주입공격 방법을 제안하였다. 제안하는 오류주입공격은 밀러 알고리듬의 루프 횟수를 판별하는 분기 구문에 오류를 주입하는 모델이며, 실제 레이저 주입 실험을 수행하여 검증하였다. 이 모델은 기존의 루프 횟수 오류 기법에서 요구하였던 확률적인 분석을 생략할 수 있어 효과적이다.

ABSTRACT

The Miller algorithm is employed in the typical pairing computation such as Weil, Tate and Ate for implementing ID based cryptosystem. By analyzing the Mrabet's attack that is one of fault attacks against the Miller algorithm, this paper presents an efficient fault attack in Affine coordinate system, it is the most basic coordinates for construction of elliptic curve. The proposed attack is the effective model of a count check fault attack, it is verified to work well by practical fault injection experiments and can omit the probabilistic analysis that is required in the previous counter fault model.

Keywords: Pairing computation, Miller algorithm, Fault injection attack

I. 서 론

타원곡선 상의 페어링 연산 기법은 별도의 키 교환 과정 없이 객체간의 암호, 복호가 가능한 ID 기반 암호시스템에 적합하여 크게 주목받고 있다. Boneh와 Franklin에 의해서 2003년 페어링 연산을 기반으로 하는 ID 기반 암호 기법[1]이 처음으로 제안된 이후로 서명 기법[2-4], 3-party 키 공유기법[5], 짧은 서명 기법[6], ID 기반 인증키 공유 기법[7] 등과 같이 다양한 영역에서 응용되고 있다.

페어링 기법을 사용하는 ID 기반 프로토콜에서는 평문을 복호화 하는 과정에서 평문뿐만 아니라 비밀값인 타원곡선 상의 점을 입력으로 사용하기 때문에, 비밀 값의 안전성이 중요한 문제가 된다. 기존의 타원곡선 상의 스칼라 곱셈 알고리듬에 대한 오류 공격 방법들은 비밀 값인 루프 횟수나 멱승 지수가 공격의 대

접수일(2010년 8월 12일), 수정일(2010년 12월 23일), 게재확정일(2011년 3월 18일)
* 이 논문은 2008년 정부(교육과학기술부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (KRF-2008-521-D00449)
† 주저자, gith@ee.knu.ac.kr
‡ 교신저자, sjmoon@ee.knu.ac.kr

상이었다[8]. 그러나 연산 횟수나 스칼라 곱셈 값들은 페어링 연산 기법에서는 공개되어 있을 뿐만 아니라 실제 비밀 값과 아무런 관계가 없어 기존의 공격 방법들을 그대로 적용할 수 없다. 따라서 페어링 연산에 대한 오류 공격들은 주로 오류 연산 결과문과 정상적인 연산 결과문을 비교하여 비밀 값과 관련된 중간 연산 인자에 대한 정보를 얻어내는 데 초점이 맞춰지고 있다. 이때 주입된 오류는 알고리듬 상의 루프 횟수를 변경하거나, 알고리듬이 수행하는 동안 연산된 중간 값을 변형시킬 수 있다고 가정한다[9-11].

페어링 기법에 대한 오류주입공격은 Page와 Vercauteren에 의해서 처음 소개되어졌다[9]. 논문 [9]에서는 Tate 페어링 연산을 효율적으로 수행하기 위해 제안된 Duursma-Lee 알고리듬[12]과 Eta 페어링 연산에서 사용되는 Kwon-BGOS 알고리듬[13,14]에 대해서 루프 횟수를 변형하는 오류 모델을 사용한 오류공격을 제안하고 있다. 다음으로 Whelan과 Scott에 의해서 Weil 페어링과 Eta 페어링에 대한 오류주입공격이 제안되어졌다[10]. 논문 [10]에서는 알고리듬 연산중에 오류를 주입하여 내부의 중간 값을 변형하는 오류 모델을 사용한다. 논문 [9]의 경우 원하는 오류 결과문 쌍을 얻기 위한 수차례의 반복적인 오류 주입 시도가 필요하며, 논문 [10]의 경우 Weil 페어링에 공격을 적용하기 위해서 연산 도중에 중간 값의 y좌표의 부호를 변경해야하는 높은 정밀도를 요구하고 있다. 마지막으로 Mrabet이 제안한 오류주입공격에서는 야코비안 좌표계를 사용하는 밀러 알고리듬상의 루프 횟수를 변형하는 오류를 가정하여 공격을 제안하였다[11]. 또한 밀러 알고리듬을 사용하는 Weil[1], Tate[15], Ate[16] 페어링 등에 모두 적용 가능함을 보였다.

본 논문은 야코비안 좌표계에서 수행된 Mrabet의 공격을 분석하여 타원 곡선을 표현하는 가장 기본적인 아핀 좌표계에서의 공격 방법을 제안하고, 컴퓨터 시뮬레이션을 통해 타당성을 검증하였다. 야코비안 좌표와 아핀 좌표는 사상관계로 동치가 성립하나, Mrabet의 공격에서는 아핀 좌표에서 존재하지 않는 Z좌표에 대한 방정식을 유도하여 비밀 값을 복원하므로 아핀 좌표에서의 새로운 고려가 요구된다. 제안한 공격 방법에서는 모든 좌표계에서 사용되는 x좌표나 y좌표를 사용한 유도과정을 설명하였다. 아핀 좌표계는 야코비안 좌표뿐만 아니라 사영좌표, López - Dahab 좌표계[17] 등의 다양한 좌표계로 사상되므로 이러한 특성을 이용하여 공격 방법의 확장이 가능할 것이다. 페어링 연산 상의 루프 횟수 오류를 이용한 기존의 공격 방법[9,11]들은 변형된 루프 횟수가 연속적인 두 오류 결과문 쌍을 획득하기 위한 오류주입 시도의 확률적인 분석이 필요하였다. 하지만 본 논문에서는 밀러 알고리듬의 루프 횟수를 판별하는 분기 구문에 오류를 주입하여 원하는 루프 시점에서 연산을 단번에 종료시키는 효과적인 방법을 제시하였다. 이 경우 원하는 루프 횟수의 오류 결과문을 바로 획득할 수 있기 때문에 기존의 확률적인 분석을 생략할 수 있으며, 따라서 오류 모델을 보다 간략화하고 정밀화할 수 있다. 이는 실제로 레이저 주입 실험을 통해 분기 구문에 대한 오류주입이 구현 가능함을 확인하였다.

먼저 Ⅱ장에서는 페어링 기법에 관한 소개와 페어링 연산을 위해서 사용되는 밀러 알고리듬에 대해서 알아본다. 다음으로 Ⅲ장에서는 논문[11]에서의 밀러 알고리듬에 대한 공격을 분석하고, 본 논문에서 제안하는 아핀좌표상의 오류주입공격 방법에 대해서 설명한다. 그리고 컴퓨터 시뮬레이션을 통해 제안한 방법의 타당성을 검증한다. Ⅳ장에서는 레이저 오류 주입 실험을 통한 루프 횟수 오류 모델의 실현성을 검토한다. 마지막으로 Ⅴ장에서 결론을 맺는다.

Ⅱ. 페어링 기법과 밀러 알고리듬

2.1 페어링 기법

위수가 l인 두 아벨군 G_1과 G_2와 동일한 위수를 가지는 곱셈에 관한 아벨군 G_3가 있을 때, 페어링 연산은 겹선형성(bilinearity)과 비축퇴성(non- degenerate)을 만족하는 사상(mapping) 함수의 형태로 정의 된다.

$$e : G_1 \times G_2 \rightarrow G_3 \tag{1}$$

일반적으로 페어링 연산은 q개의 원소를 가지는 유한체 F_q상에 존재하는 타원곡선 E 에서 정의된다. 여기서q는 표수(characteristic) p의 멱승 값이다. l은 $l|E(F_q)$를 만족하는 q와 서로소인 양의 정수라고 한다면, embedding degree k는 $l|q^k-1$을 만족하는 가장 작은 정수이다. 수식 (1)에서 페어링 연산 e는 각각 $G_1 \subset E(F_q)$, $G_2 \subset E(F_{q^k})$를 만족하는 타원곡선 E상의 두 점을 유한체의 곱셈군인 $G_3 \subset F_{q^k}^*$의 한 원소로 사상하는 함수의 역할을 한다. 제안하는 공격 기법의 대상인 아핀좌표 상에서는 Weistrass 타원곡선 방정

식이 유한체 F_{q^k}의 표수(characteristic) q가 2나 3이 아닌 경우, 아래의 수식 (2)와 같이 정의 된다.

$$Y^2 = X^3 + aX + b,\ a,b \in F_{q^k} \quad (2)$$

페어링의 결과 값은 연산에서 사용하는 알고리듬마다 차이가 있으나 입력 값 P를 l번 더한 값인 $[l]P \in G_1$의 유리함수(rational function)에 $Q \in G_2$의 divisor를 인가한 값의 형태를 취한다. 유리함수 $f_{l,P}$의 divisor인 $div(f_{l,P})$는 $l(D)-l(O)$와 동치인 특성을 가지며, $div(f_{l,P}) = l(P)-([l]P)-(l-1)O$를 만족한다. 이때 O는 E 상의 무한점이다. 인가되는 D_Q는 $(Q)-(O)$와 동치인 0차수의 divisor이며, $div(f_{l,P})$와 D_Q는 서로 다른 support를 갖는다. Divisor에 관해서는 단행본 [18]에서 자세히 설명하고 있다. $f_{l,P}$는 누적곱셈의 형태이므로 이를 쉽게 구하기 위해서 밀러 알고리듬[19] 등을 사용한다. 보안 영역에서 사용되고 있는 페어링 연산 기법들은 유사한 형태로 구성되어 있으며, 특히 Weil, Tate, Ate 페어링 기법에서는 밀러 알고리듬이 가장 중요한 역할을 하고 있다.

2.2 밀러 알고리듬[19]

밀러 알고리듬은 타원곡선 상의 곱셈과 덧셈 특성에 따라 입력 점 P에 대해서 더블링과 덧셈 연산을 수행한다. 위수가 l인 점 P를 입력으로 하여 타원곡선 상의 스칼라 곱셈과 동일하게 $[l]P$를 연산하는 동안의 점과 점 P간의 접선에 대한 함수를 누적 곱하여 계산한다. 아핀 좌표를 사용하는 경우에는 밀러 알고리듬의 매 루프 연산마다 중간 값 점 T의 접선(tangent line)과 수직선(vertical line)의 함수들을 도출하여, 두 함수를 나눈 유리 함수 f_P을 계산한다. 도출된 함수 f에 점 Q의 좌표 값을 인가하여 $F_{q^k}^*$상의 한 원소인 밀러 변수(miller variable) $f_P(Q)$를 계산한다. 매 루프마다 연산된 밀러 변수들의 누적 곱 $(f_1(Q)f_2(Q)\cdots f_l(Q))$은 실제 밀러 알고리듬의 출력 값이 되며, 선택한 입력 값 점 P와 연관되기 때문에 $f_{l,P}(Q)$와 같이 표기한다. 실제 알고리듬은 아래 [그림 1]과 같다.

입력 : $P \in G_1$, $Q \in G_2$,
$l = (l_{n-1} \cdots l_0)$: radix 2 presentation
출력 : $F_P(Q) \in G_3$

1. $T \leftarrow P$, $f \leftarrow 1$
2. for $i = n-1$ to 0 do
3. $T \leftarrow [2]T$ //doubling
4. $f \leftarrow f^2 \cdot t_{T,T}(Q)/v_{2T}(Q)$
5. if $l_i = 1$ then
6. $T \leftarrow T + P$ //addition
7. $f \leftarrow f \cdot t_{T,P}(Q)/v_{T+P}(Q)$
8. end if
9. end for
10. return $f \in F_{q^k}^*$

[그림 1] 밀러 알고리듬

III. 밀러 알고리듬에 대한 오류주입공격

3.1 야코비안 좌표계 상의 오류공격[11]

아핀 좌표상의 페어링 연산에 대한 공격 기법들[9,10]과 달리 논문 [11]에서 제안된 공격 기법은 야코비안 좌표계 상에서 동작하는 밀러 알고리듬을 대상으로 하고 있다. 야코비안 좌표계는 (X,Y,Z)를 사용하며, 이는 아핀 좌표$(x = X/Z^2, y = Y/Z^3)$으로 치환가능하다.

논문 [11]에서 소개된 공격기법은 페어링 연산의 결과 값이 유한체상의 기저(basis)를 통해 표현되는 것을 이용한다. 루프 횟수에 대한 오류주입을 통해 획득한 오류 결과 값들의 비율을 계산하여 획득한 밀러 변수 값과 실제 연산 상의 이론적인 수식 표현간의 상관관계를 분석하여 P와 Q의 좌표에 관한 비선형 시스템(non-linear system)을 도출한다. 시스템 내 수식 상의 미지수들의 계수와 차수를 동일하게 맞춰서 해를 구하는 선형 연립 방정식의 풀이 과정과 달리 비선형 시스템의 경우에는 미지수의 종류에 비해 수식의 수가 충분하지 않으며, 미지수들끼리 결합되어 있어 해를 구하는 과정이 좀 더 까다롭다. 따라서 논문 [11]에서는 비선형 시스템의 각 수식들에 대해서 하나의 미지수에 대한 수식으로 정리하여, 대체·치환과정을 거쳐 야코비안 좌표(X,Y,Z)중의 Z 좌표에 대한 고차방정식 형태로 유도하는 과정을 설명하고 있다. 공격자는 미지수가 하나로 치환된 고차방정식을 풀고, 나머지 좌표 값에 대한 후보를 찾아 검증 과정을 통해서 올바른 비밀 값의 좌표를 도출한다. 또한 논문 [11]에서 가정하는 오류주입 모델은 역공학(reverse engineering) 기법을 적용하여 공격 대상 칩의 동작 중 사용되는 밀러 알고리듬과 연관되는 카운터 레지스터의 위치를 파악한 후, 레이저 [20,21]와 같이 정밀

도가 높은 오류주입 기법을 적용하여 카운터의 동작에 장애를 발생하여 루프 횟수를 변형할 수 있다고 설명한다.

일반적으로 야코비안 좌표는 기본적인 좌표계인 아핀 좌표를 사영하여 구현하므로 동치관계에서 벗어나지 않는다. 그러나 논문 [11]의 공격은 아핀 좌표에서 존재하지 않는 Z 좌표에 대한 방정식을 유도하여 비밀 값을 찾아내므로 $x = X/Z^2, y = Y/Z^3$의 관계만으로는 아핀 좌표로의 직접적인 적용이 불가능하다. 또한, 아핀 좌표계는 사영좌표, López - Dahab 좌표계 등과 같은 다양한 좌표계로 표현될 수 있는 가장 기본적인 좌표계이므로, 아핀 좌표계에서의 공격이 잘 정의될 때, 다양한 좌표계에서의 공격으로 활용이 가능하다. 따라서 본 논문에서는 아핀 좌표를 사용하는 밀러 알고리듬에 대해서 논문 [11]처럼 오류 연산을 통해 획득한 값이 유한체 F_q를 구성하는 기저에 의해서 표현된다는 개념을 사용하여 오류주입공격 방법을 새로이 제안한다.

3.2 제안하는 밀러 알고리듬에 대한 오류주입공격

일반적으로 ID 기반 프로토콜에서 사용되는 페어링 기법은 밀러 알고리듬의 두 입력 값인 P와 Q 중에 하나를 공개 값으로, 하나를 비밀 값으로 저장하여 사용한다. 본 논문에서 제안하는 페어링 연산에 대한 오류주입공격은 점 P를 비밀 값으로 한 경우를 먼저 가정한다. 점 P의 좌표를 찾아내기 위해서 본 논문은 밀러 알고리듬의 루프 횟수를 변형하는 오류를 가정한다. 이를 위해서 먼저 단순 전력 분석[22]과 같은 방법으로 공격자가 원하는 적절한 루프 연산 시점을 확인하고, 다음으로 레이저 주입으로 일시적인 오류를 주입하여 밀러 연산을 원하는 루프 연산 횟수에서 종료하도록 유도한다.

페어링 연산에 대한 기존의 루프 횟수 오류 모델은 밀러 알고리듬 상의 루프 횟수를 변형하여 순환 구문의 연산 횟수를 변형하는 오류를 가정하였다. 루프 횟수와 관련된 메모리 레지스터에 대한 오류를 가정하여 루프 횟수가 비정상적으로 증가 또는 감소된다고 가정하였다. 임의의 값으로 변형되기 때문에 총 루프 횟수에 대한 예측이 불가능함에 따라 기존의 공격들은 변형된 루프 횟수가 연속적인 결과문들을 얻기 위한 확률적인 분석이 필요하였고, 이는 루프 횟수를 담당하는 레지스터의 크기에 의존하여 예측되었다[11].

본 연구에서 가정하는 오류 모델은 연산 횟수에 대한 분기 구문(count check)이 수행되는 시점을 대상으로 한다. 분기 구문은 어셈블리어의 구조상 해당 시점의 순환 횟수와 알고리듬이 요구하는 최종 순환 횟수와의 비교 단계(cpi)와 비교 결과에 따라서 순환 구문의 처음이나 순환 구문 밖으로 이동(brcs)하는 단계로 구성된다. 분기 구문 내의 단계들은 어셈블리어 분석을 통해서 확인할 수 있다. 비교 단계나 이동 단계에 오류가 주입된다면 for 구문의 처음으로 회귀하지 않고 해당 루프 시점에서의 탈출이 가능하다. 이러한 오류주입모델은 공격자가 원하는 루프 횟수만큼만 알고리듬을 동작시킨 후 단번에 종료할 수 있는 것으로 기존의 확률적인 분석 없이 연속적인 루프 횟수의 오류 결과문을 얻을 수 있다. 또한 논문 [10]의 데이터 변경 오류 모델의 대상인 부호 비트(significant bit)이나 논문 [11]의 기존 연산 횟수 오류 모델의 대상인 카운터 레지스터 등에 비해 접근이 용이하여 실현 가능성이 높다.

제안하는 오류공격 기법은 위에서 설명한 오류주입 모델을 사용하며 다음의 과정으로 동작한다. 먼저 공격자가 의도한 루프 횟수 d번과 $d+1$번을 수행한 오류 연산 결과문 쌍 $(d,d+1)$을 획득한다. 이때 획득한 오류 결과문 쌍은 d의 크기에 상관없이 연속적이다. 획득한 오류 결과문 쌍을 $f_{d,P}(Q)$와 $f_{d+1,P}(Q)$라 표기할 때, 두 결과 값의 비율인 $f_{d+1,P}(Q)/(f_{d,P}(Q))^2$을 계산하여 $d+1$번째 루프에서 연산된 밀러 변수를 추출할 수 있다. 유한체의 원소들은 기저로 표현되므로 이를 추출된 밀러 변수의 수식과 연관하여 점 P에 대한 수식을 유도한다. 마지막으로 수식을 풀어 점 P의 후보군을 찾아낸 후, 후보군에 대한 정상적인 밀러 알고리듬 수행 결과를 통해서 올바른 P의 좌표를 검증하는 과정을 거친다.

3.2.1 연산횟수와 중간값

밀러 알고리듬의 d번째 루프 연산에서 종료될 때 중간 값 T는 $[j]P$와 같다. 이때 스칼라 값 j는 타원곡선의 특징 상 위수인 l로 모듈러 연산을 취하므로 $d < l$을 만족해야 한다. 즉, 오류주입 시점은 밀러 알고리듬의 정상적인 수행 종료 시점의 이전으로 가정한다. 다음으로 $d+1$만큼 수행하고 종료되는 경우에는 l_{d+1}의 비트에 따라서 $[2j]P$ 또는 $[2j+1]P$로 연산된다. d번째와 $d+1$번째의 오류 결과문을 얻을 수 있는 오류주입 시점은 소비되는 전력의 변화를 분석하여 연산횟수를 확인할 수 있다. 이때 $d+1$번째 루프가 수행될 때 소비

되는 전력의 크기를 비교하여 더블링 연산만 수행한 경우인지 더블링과 덧셈 연산까지 수행하는 경우인지 판별 가능하다. 이러한 단순 전력 분석을 통해서 d번째와 $d+1$번째 까지 수행하는 동안의 l_n의 각 비트 값을 추정할 수 있다. $l_{n-1}=0$인 경우에는 더블링 연산만 수행하므로 $j \rightarrow 2j$가 되고, $l_{n-1}=1$인 경우는 덧셈 연산까지 추가적으로 수행하므로 $j \rightarrow 2j+1$이 된다. 따라서 공격자는 밀러 알고리듬의 d번 루프 수행시의 $(l_{n-1},\ldots,l_0)$ 값들을 확인하여 중간 값인 $[j]P$의 스칼라 값인 j를 추정할 수 있다.

3.2.2 유한체의 기저와 밀러 변수

밀러 알고리듬의 매 루프 마다 도출되는 밀러 변수는 유한체의 곱셈군 $F_{q^k}^*$ 상에 존재하므로 유한체를 구성하는 기저에 의해 표현된다. 이 때 embedding degree k에 따라 유한체를 구성하는 기저의 수가 달라진다. 제안하는 공격 기법에서는 연속적인 두 오류 암호문 쌍을 나눈 비율 또한 동일한 기저에 의해 표현된다는 사실을 이용하여, 각 기저에 해당하는 성분들 간의 동치를 통해 비밀 값에 관한 수식을 도출한다. 따라서 표현되는 기저의 수에 상관없이 적용 가능하므로, 본 논문에서는 $k=2$인 경우를 하나의 예로 설명한다. $k=2$일 때, $G_2 \subset F_{q^k}$상의 기저는 $B=\{1,i\}$로 표기하며 이때 $i^2=-1$을 만족한다. 이 때 연산의 효율을 위해서 입력 값 Q는 distortion map을 사용하여 $G_1 \subset F_q$상의 동일한 위수를 가지는 원소 중 P가 아닌 점 하나를 임의로 선택한 후 사상하여 G_2상의 한 점으로 선택한다. G_1상의 한 점의 좌표가 (x,y)일 때, $k=2$ 상에서의 distortion map을 통해서 점 Q의 좌표는 $(x,-yi)$으로 정의되며, 여기서 $x, y \in F_q$이다.

점 P가 (X_P, Y_P), 점 Q가 (x,yi), 그리고 중간 값인 점 T가 (X,Y)로 표기한다면, [그림 1]의 밀러 알고리듬에서 4번 과정과 7번 과정 동안 곱해지는 접선의 함수와 그에 해당하는 밀러 변수들은 아래와 같이 표현된다.

$$\begin{cases} t_{T,T}(Q)=yi-\lambda_j x+(\lambda_j X-Y) \\ v_{2T}(Q)=x-(\lambda_j^2-2X) \\ \qquad with\ \lambda_j=\dfrac{3X^2+a}{2Y}, \\ t_{T,P}(Q)=yi-\lambda_{2j} x+(\lambda_{2j} X-Y) \\ v_{T+P}(Q)=x-(\lambda_{2j}^2-X-X_P) \\ \qquad with\ \lambda_{2j}=\dfrac{Y_P-Y}{X_P-X}. \end{cases} \tag{3}$$

3.2.3 $l_{d+1}=0$인 경우

공격자는 밀러 알고리듬의 연산 도중 오류주입을 통해 출력되는 d번째 밀러 변수 결과 값 $f_{d,P}(Q)$와 $d+1$번째 밀러 변수 결과 값 $f_{d+1,P}(Q)$를 알고 있다고 가정하므로, $d+1$번째 루프 연산시 밀러 알고리듬에서 동작되는 단계들을 기준으로 분석한다. l_{d+1}비트가 0인 경우, 밀러 알고리듬은 [그림 1]의 단계 3, 4에서 수행되는 더블링 연산과 그에 해당하는 밀러 변수를 연산한다. d번째 루프 연산의 결과 값을 $[j]P=(X_j, Y_j)$라 두면, $d+1$번째 루프 연산에서 단계 3의 수행 결과인 점 T는 $[2j]P=(X_{2j}, Y_{2j})$로 나타낼 수 있으며, 아래의 수식으로 표기된다.

$$\begin{cases} X_{2j}=\left(\dfrac{3X_j^2+a}{2Y_j}\right)^2-2X_j \\ Y_{2j}=\left(\dfrac{3X_j^2+a}{2Y_j}\right)(X_j-X_{2j})-Y_j \end{cases} \tag{4}$$

[그림 1]의 단계 4에서는 아래 수식과 같이 이전 루프 연산의 결과인 밀러 변수 $f_{d,P}(Q)$에 단계 3에서 연산된 점 T에 해당하는 함수 값을 도출하여 곱해준다. 여기서 x와 yi는 점 Q의 좌표이다.

$$\begin{aligned} f_{d+1,P}(Q)=(f_{d,P}(Q))^2 \times [\, yi-\left(\frac{3X_j^2+a}{2Y_j}\right)x \\ +\left(\frac{3X_j^2+a}{2Y_j}\right)X_j-Y_j]/(x-X_{2j}) \end{aligned} \tag{5}$$

$f_{d+1,P}(Q)/(f_{d,P}(Q))^2$의 연산을 통해서 $d+1$번째 루프 연산을 통해 계산된 $h_1(Q)=t_{T,T}(Q)/v_{2T}(Q)=R$를 추출할 수 있다. 수식 (5)을 이용하여 $f_{d+1,P}(Q)$와 $f_{d,P}(Q)$에 대한 비율 R을 계산하면 아래의 수식으로 표현된다.

$$\begin{aligned} R&=\frac{f_{d+1,P}(Q)}{(f_{d,P}(Q))^2} \\ &=\frac{1}{(x-X_{2j})}[\left(\frac{3X_j^2+a}{2Y_j}\right)X_j-\left(\frac{3X_j^2+a}{2Y_j}\right)x \\ &\quad -Y_j]+\frac{1}{(x-X_{2j})}yi \end{aligned} \tag{6}$$

$f_{d,P}(Q)$과 $f_{d+1,P}(Q)$ 모두 $F_{q^k}^*$에 포함되기 때문에 R의 연산 결과도 동일하게 $F_{q^k}^*$에 포함된다. 따라서 비율 R은 유한체의 기저 $B=\{1,i\}$에 의해 표기되므로, 오류주입을 통해서 획득한 두 밀러 변수의 비율을 실

제로 구하면 아래의 수식과 같은 형태를 취하게 된다.

$$R = R_0 + R_1 i,\ R_0, R_1 \in F_q \tag{7}$$

수식 (6)과 수식 (7)은 동치이므로, $R = R_0 + R_1 i$에서 R_0와 R_1은 아래와 같이 정리된다.

$$\begin{cases} R_0 = \dfrac{1}{(x - X_{2j})}\left[\left(\dfrac{3X_j^2 + a}{2Y_j}\right)X_j \right. \\ \qquad \left. - \left(\dfrac{3X_j^2 + a}{2Y_j}\right)x - Y_j\right] \\ R_1 = \dfrac{1}{(x - X_{2j})}y \end{cases} \tag{8}$$

여기서 공격자는 오류주입을 통해 획득한 결과 값을 가지고 비율을 계산하므로 R_0와 R_1에 해당하는 값을 알고 있다. 또한 공개키인 $Q=(x, yi)$의 값과 수식 (4)의 X_{2j}의 수식 표현을 알고 있으며, $[j]P$가 타원곡선 상의 한 점으로 $Y_j^2 = X_j^3 + aX_j + b$의 수식을 만족함을 알고 있다. 따라서 수식 (8)에서 R_1에 대한 이론적인 형태를 수식 (4)의 X_{2j} 값과 타원곡선 방정식, Q 값을 사용하여 점 $[j]P$의 X 좌표에 대한 수식으로 정리할 수 있으며, 이는 아래 수식 (9)와 같이 표현할 수 있다.

$$\begin{aligned} & R_1 X_j^4 - 4(R_1 x_0 - y_0)X_j^3 - 2aR_1 X_j^2 \\ & \quad - 8bR_1 X_j - 4a(R_1 x_0 - y_0)X_j \\ & \quad + a^2 R_1 - 4b(R_1 x_0 - y_0) = 0 \end{aligned} \tag{9}$$

공격자는 R_1, x_0, y_0, a, b 등의 계수에 대해서 알고 있으므로 수식 (9)는 $[j]P$의 X 좌표인 X_j에 관한 4차 방정식으로 표현된다. 정수론 상의 수학 연산을 돕는 공개 라이브러리인 NTL[23]상의 CanZass 함수를 사용하여 수식 (9)를 소인수분해하고, 최종적으로 X_j에 관한 해들을 구할 수 있다. CanZass 함수는 유한체 상의 다항식을 소인수 분해하는 Cantor - Zassenhaus 알고리듬을 구현한 함수이다[24]. 수식 (10)의 R_0에 관해서도 X_j에 관한 방정식을 도출할 수 있지만, R_1을 통한 수식 (9)의 경우가 보다 단순하며 풀이에 유리하다. 수식 (9)에 대한 해들을 구한 후, 타원곡선 방정식에 의해 하나의 X_j에 해당하는 2개의 Y_j 값을 도출할 수 있다. 즉 $[j]P$의 후보 값들을 최대 $8 = 4 \times 2$개로 유추할 수 있다. 앞서 3.2.1의 내용처럼 공격자는 연산횟수 $[j]$의 값을 찾아낼 수 있기 때문에 $j^{-1} \bmod l$을 계산한 후, $[j]P$의 각 후보 값들을 j^{-1}만큼 스칼라 곱셈 연산하게 되면 점 P에 해당하는 후보 값들을 찾아낸다.

$$[j^{-1}][j]P = [j^{-1} \cdot j]P = P \tag{10}$$

수식 (10)을 통해서 찾아낸 점 P의 후보군들은 원래의 위수인 l만큼 스칼라 곱셈 연산 시 타원곡선의 특성에 따라서 무한점 O가 된다. 또한 공격자는 오류주입이 없을 때의 정상적인 밀러 연산 결과 값을 가지고 있으므로 점 P의 후보군을 입력으로 하여 밀러 알고리듬을 수행한 결과 값과 비교하여 후보 값이 올바른 점 P인지를 판별할 수 있다. 따라서 공격자는 아래 [그림 2]의 조건들을 만족하는 후보군을 선택하여 비밀 값인 점 P의 좌표의 올바른 값을 찾아낼 수 있다.

3.2.4 $l_{d+1} = 1$인 경우

3.2.3의 경우와 달리 l_{d+1}의 비트가 1인 경우에는 더블링 연산에 추가적으로 점 P에 대한 덧셈 연산이 포함되어 있다. 따라서 덧셈 과정에 의한 밀러 변수의 변화를 고려해야 한다. 앞서 3.2.3에서 도출하였던 밀러 변수 값을 $h_1(Q)$, 덧셈 과정의 결과 값인 점 T와 점 P에 대한 밀러 변수를 $h_2(Q)$라 두었을 때, [그림 1]의 단계 7을 통해서 연산되는 $d+1$번째 밀러 변수는 아래 수식과 같다.

$$f_{d+1,P}(Q) = (f_{d,P}(Q))^2 \times h_1(Q) \times h_2(Q) \tag{11}$$

d번째 루프의 결과 값을 $[j]P = (X_j, Y_j)$라 할 때, 더블링의 결과 값은 $[2j]P = (X_{2j}, Y_{2j})$이다. 더해지는 비밀키 값인 점 $P = (X_P, Y_P)$이고 덧셈 연산의 결과 값은 $[2j+1]P = (X_{2j+1}, Y_{2j+1})$이라 할 때, 덧셈 연산으로 곱해지는 밀러 변수는 아래의 수식으로 나타낼 수 있다.

$$\begin{aligned} h_2(Q) = & \left[y_0 i - \left(\frac{Y_P - Y_{2j}}{X_P - X_{2j}}\right)x_0 \right. \\ & \left. + \left(\frac{Y_P - Y_{2j}}{X_P - X_{2j}}\right)X_{2j} - Y_{2j}\right] / (x_0 - X_{2j+1}) \end{aligned} \tag{12}$$

3.2.3에서 기술한 공격방법을 그대로 적용하여 d번째 밀러 변수 결과 값과 $d+1$번째 밀러 변수 값의 비

1. 비밀 값 점 P는 위수가 l이므로, $[l]P$의 스칼라 곱셈 결과는 무한점이다.
   $[l]Candidate(P) = O$
2. 점 P의 후보 값 좌표를 입력으로 한 밀러 알고리듬의 결과 값은 밀러 알고리듬의 정상 수행 결과 값과 같다.
   $f_{Candidate(P)}(Q) = f_P(Q)$

[그림 2] 후보 값에 대한 검증 조건

율 $R=f_{d+1,P}(Q)/(f_{d,P}(Q))^2$을 구하면, 다음과 같다.

$$R=h_1(Q)h_2(Q) \tag{13}$$

비율 R의 이론적인 형태는 수식 (6)과 수식 (12)를 곱한 수식으로 나타나며, X_j와 Y_j, X_P와 Y_P에 관한 수식으로 표현된다. 앞서 기술한 바와 같이 비율 R은 $F_{q^k}^*$상의 한 원소이므로 $R_0+R_1 l$의 형태가 된다. 여기에 타원곡선 방정식에 의한 $Y_j^2=X_j^3+aX_j+b$와 $Y_P^2=X_P^3+aX_P+b$ 두 수식을 사용하여 R_0와 R_1에 대해 정리하면, 각각 X_j와 X_P에 관한 서로 다른 두 식으로 유도할 수 있다.

$$\begin{cases} R_0 = W_1(X_j, X_P) \\ R_1 = W_2(X_j, X_P) \end{cases} \tag{14}$$

수식 (14)에서 공격자는 두 미지수 X_j, X_P에 대해서 두 식을 찾아내었기 때문에, 연립 방정식의 풀이 방법을 적용하면 X_j와 X_P에 관한 후보들을 찾아낼 수 있다. 앞서 3.2.3에서 소개한 방법과 달리 $l_{d+1}=1$의 경우에서는 점 P의 좌표 X_P에 대한 후보군을 바로 찾아낼 수 있기 때문에 $[j^{-1}][j]P$의 스칼라 곱셈 연산 과정이 불필요하다. X_P의 후보들을 찾아낸 후에는 [그림 2]의 조건을 사용하여 $P=(X_P, Y_P)$의 후보들의 위수가 l인지, 그리고 후보 값들을 입력으로 하여 밀러 알고리듬이 정상 수행된 결과가 동일한지를 확인하여 올바른 P의 좌표들을 확인한다.

3.2.5 공격 방법의 활용

3.2.3과 3.2.4에 기술된 공격 방법은 입력 값 점 P가 비밀 값인 경우에 한정하였으나, 점 Q가 비밀 값인 경우에는 비율 R을 통해 도출된 방정식이 다차항이 아닌 일차항의 방정식의 형태로 나타나기 때문에 더 쉽게 좌표를 복원할 수 있다. 페어링 연산을 위해서 밀러 알고리듬을 사용하는 경우, 밀러 알고리듬의 두 입력 값 중 어느 점을 비밀 값인지에 상관없이 제안하는 공격기법에 취약성을 보이게 된다.

연속적인 오류 결과문의 비율을 통해 오류 주입 위치의 루프에서 연산된 유리 함수를 도출하는 형식은 밀러 알고리듬이 아니라도 루프 형태의 연산을 수행하는 다른 페어링 연산의 알고리듬에도 적용할 수 있다. 대부분의 페어링 연산에서 사용되는 알고리듬들은 루프 연산 마다 도출되는 함수 값을 이전 루프 연산에서의 결과에 누적 곱해지는 형태이므로 루프 횟수가 하나 차이 나는 두 오류 결과문의 차이를 통해 함수 값을 추출할 수 있을 것이다. 이는 Duursma-Lee 알고리듬이나 Eta 페어링의 BKLS 알고리듬 등에도 적용됨이 이미 알려져 있다[9].

또한, 아핀 좌표의 x,y좌표를 사용한 제안한 공격 기법의 수식들은 $x=X/Z, y=Y/Z^2$의 관계를 만족하는 사영좌표나 $x=X/Z, y=Y/Z^2$의 관계를 만족하는 López - Dahab의 좌표계의 경우에도 관계식에 맞춰 활용 가능할 것이다. 또한 4개의 좌표 값을 사용하는 Chudnovsky 좌표계[25]의 경우에도 $x=X/Z^2$, $y=Y/Z^3$의 관계를 만족하므로 적용 가능할 것으로 판단된다.

3.3 제안하는 공격 방법의 검증

3.3.1 시뮬레이션 구성

제안하는 오류주입공격 기법의 유용성을 판별하기 위해서, 논문 [26]에서 소개된 수학 소프트웨어 Mathematica상에서 구동하는 밀러 알고리듬에 대한 함수 소스를 적용하여 C++ 환경에서 구현하였다. 오류결과 값의 분석과 연산을 위해서 라이브러리인 NTL을 사용하였다. 구현한 밀러 알고리듬의 정상 동작 여부를 판별하기 위해서 논문 [10]의 부록 B에 기술된 512비트 크기의 예제 파라미터들을 활용하여 유한체를 구현하였다.

타원곡선이 $Y^2=X^3+aX+b$로 정의 될 때, 타원곡선이 위치한 유한체 F_p의 파라미터와 타원곡선의 계수는 아래와 같다.

p: C8DF90FD89D7A9827ED3A42409D36AED1
3BA334D9E716169FC674B6CB2F98E407A5
0E9E0FBDC15512D1F74A679C08D3AC42F
292C8A95DF30758293235E04739B_{16}

a: C8DF90FD89D7A9827ED3A42409D36AED1
3BA334D9E716169FC674B6CB2F98E407A5
0E9E0FBDC15512D1F74A679C08D3AC42F
292C8A95DF30758293235E047398$_{16}$

b: 491224746BC3C29BAB0157FE84580BB3E5
3912605012B88BCCEFE240FB679E4605EA
BBBBA9A642BEA71A3840107915913F998
E71662C2501A2E35122C569BF68$_{16}$

밀러 연산을 위해서 입력되는 비밀 값인 점 P의 좌표와 그 위수 l, 그리고 공개 값인 점 Q의 좌표는 아래와 같다.

l : 800000000000000000000000000000000000
20001_{16}

P = (C27938C41E38949CA9177135C99374D27
659D9489955F65F312EEEF10375C78CAE
2348E0E8E7B32D5021E181CABC42EFE
30D44038950F23E7F49980D2654C09B8$_{16}$,
59F5C3910EB567CDB4B16BB2A0B2A8
DFEA3ED92A64A45247890576568B779A
846ECBC9FEA150CD492223836EE4796
B23751D547B4DC06CD48E230DCAE436
32DA$_{16}$)

Q = (16EEB9A9E3ABE93CF0B0A0B256ED07
BAF6CC6D7192B566E14AFB96C769C88
0CDE2A2AA5003639E8AEF95EC157160
B210FE1F23433E1003F94D08988E29E37
F10$_{16}$,
iA747D64825676DB1176A4A2CF79EAD
209525FBFFE008749CEB0458C8433501
979524635D3436CE5D232FF1CF122F01
B659E36CA0EFB1AED8318EEAB32FD8
423C$_{16}$)

정상적인 밀러 알고리듬 수행시 밀러 변수의 값 $f_P(Q)$는 다음과 같이 출력된다.

$f_P(Q)$ = A3FD8803E30ED1613B62B6685CAC
5756ADF38A765DACDFAFBFBA9E
080B78D4095D971310D1AAE10D456
45D7802A05527636B05F4D61A444B5
A7A598A8CB9FA75$_{16}$ +
iB7B6FB5D9B26EC1CBD12E46FE9
BA63FCA75A7CAC5C11b7D90E030
F1FB3DD08CD155211B24E3161E12
FEC1CBCBE80BADFE48E8C451B
811CFA96A31356E1A1C04E$_{16}$

3.3.2 시뮬레이션 구성

오류주입을 가정하여 임의의 루프에서 밀러 알고리듬의 동작을 정지하고, 해당 루프에서의 밀러 연산 결과를 확인하였다. $d=8$인 경우와 $d+1=9$인 경우에서의 밀러 연산 결과 $f_{[40_{16}]P}(Q)$와 $f_{[80_{16}]P}(Q)$는 아래와 같다.

$f_{[40_{16}]P}(Q)$ = 84B8EA53CE8D51BC073ABFFB3
9C48FFFB3ABD582B09C8C965D
1E3EB6FC6D454D5465A09D37658
ET22958132830596860F0DB96B173
6EEDFFC290DB0172D2ED3F$_{16}$ +
i1358D9D363A8C6C057E464A3D3
79C05B8D2DA13BBCB07BB4869
53497AF1DADDCC1F8F4F17ED
D68A2E16F88A2CB5DE2A6DA9
AC1B311E60653C9A804B4BFD8
AEF3$_{16}$

$f_{[80_{16}]P}(Q)$ = C7DFAA44F1811B27DE10F69BA1
3B2511E7C19A70EEECB1B9E5A2
E00DE6763D03553D6BFB75A38A
190BAE0CCB7FAFCA07A30A661
434752608447AF1101CEE3C02$_{16}$ +
i718E9C019AAD9B46EC553087ED
53E435AECB4720144DAF81347F
C5DAC272C526CC810A2FAB82A
8DB8AE636C2D8D8873C6497238
CEA5EFC7819E2A8CA6227DB2E$_{16}$

비율 $R=f_{d+1,P}(Q)/(f_{d,P}(Q))^2$을 구하면 아래의 값과 같으며, $R_0+R_1 i$의 형태로 도출됨을 확인하였다.

R = 4F1F6D078FD374621D9B9686BB7A960727
E5EA808DD98E8560851C057F13D4F3C67D
D65D7420993760FFC52780AF262FC61CBF
A9E58817D26752EA13A20D9AE$_{16}$ +
i267A95711D09829670E09673C8FABB3DB
7D457D24184EC59E539AAD8F583F2BE46
67D9B53A6BEACAB3B89BDBC7459A400
0C172C335E17F574D4343A72692BAF0$_{16}$

R_1 값을 수식 (9)에 인가하여, X_j에 관해 방정식을 풀게 되면 X_j의 후보 값들을 찾을 수 있다.

$$X_j = \begin{cases} B8F6A31066FBD7268ADC8A15B9D737 \\ B7BD5344D839E34207AB3F10C525CE \\ F34089DFFD964544FB9BA44EB939E8 \\ 4E8E01AF9DE0E7D2DDBB70010BDC \\ B033DAF29B_{16} \\ 41F66EEA471EEA4E94A6B96704DBFDA \\ D26B70974EEEEB1C34290BC71ABED2 \\ 05830F1080D68792336A35672065B4E \\ CDCEA8AD9E2538D1E3E4DE3525C2 \\ BCC0940F72D_{16} \end{cases}$$

수식 (9)에 해당되는 4개의 근 중에서 실제로 타원곡선 상에 존재하는 X_j의 후보 값은 2개이며, 타원곡선 방정식에 따라서 각 X_j의 후보 값에 해당하는 Y_j의 값은 2개씩 존재하므로, 총 4개의 $[j]P$의 좌표 값 후보군이 도출된다. 후보군들을 검증하기 위해서 일단 아래와 같은 모듈러스 l상의 j의 역원 값을 구한 후, 각 후보군에 해당하는 P의 좌표를 구한다.

$j^{-1} \bmod l$ = 7ED0000000000000000000000
00000000001F801$_{16}$

다음으로 P의 후보군에 대해서 스칼라 곱셈 $[l]P$를 수행한 결과가 타원곡선 상의 무한점 O_∞에 해당하는지를 검증한다. $[l]P$가 무한점이 되는 P의 후보군들을 입력으로 하여 위수 l의 정상적인 밀러 알고리듬을 수행한 후, 그 결과 값이 앞서 정상적인 밀러 알고리듬의 결과 값과 동일한지를 확인하여 P의 올바른 좌표

값을 찾아낼 수 있다. $[l]P$의 값이 무한점인지의 여부와 각 후보 값에 대한 밀러 알고리듬 수행 결과를 판별한 시뮬레이션 결과는 부록 1에 기재하였다.

512비트의 유한체를 사용하는 밀러 알고리듬에 대한 공격 시뮬레이션 결과를 통해 실제로 오류결과 값들의 비율 R이 기저에 의해서 표현됨을 확인하였다. 또한, 제안하는 공격기법의 수식 (9)를 이용한다면, 방정식의 해를 구하고 검증하는 과정을 거쳐 원래의 비밀 값 P의 좌표를 복원할 수 있음을 확인하였다.

IV. 오류주입 모델 구현

4.1 오류주입 모델의 실험적 검증

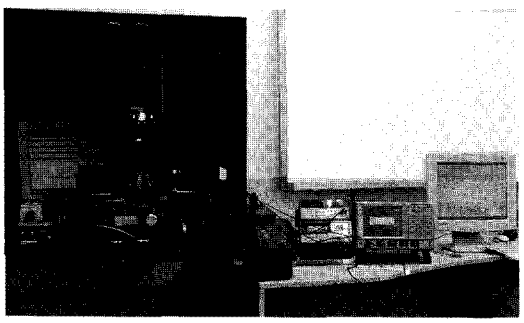
루프 구문에 대한 연산 횟수 변경 오류는 기존의 RSA나 ECC 등의 다양한 알고리듬에서 제안되었다. 그러나 실제 오류 주입을 통한 실현 가능성은 확인되어진 바가 없다. 따라서 우리는 제안한 공격의 실현성을 검증하기 위해서 오류주입 실험을 수행하였다. 이를 위해서 먼저 다음과 같은 레이저 오류주입 환경을 구축하였다.

오류주입 실험을 위해 센서 네트워크 시스템에서 많이 사용되는 상용 8비트 마이크로프로세서인 ATmega128 칩에 오류주입을 시도하여 그 결과를 분석하였다. 또한, 오류주입 시점을 결정하고 오류 주입에 의한 공격 대상 칩의 동작 변화를 소비전력을 통해 관측/분석하기 위해서 LeCroy사의 LT374M 모델를 사용하였다. 마지막으로 공격 대상 칩의 내부 회로에 직접 레이저 오류를 주입할 수 있는 EzLaze 3 모델을 사용하였다.

ATmega128 칩은 물리적 공격에 대한 대응책이 탑재되지 않은 칩으로 국외의 오류관련 연구 사례에서 자주 사용되고 있다. 본 논문에서는 연산 횟수 변경을 위한 정확한 오류주입 시점을 찾기 위해서 어셈블리어에 대한 분석을 수행한다. 즉, 동작 클럭에 맞춰 수행되는 어셈블리어의 순서를 파악한 후, 나노초(ns)단위로 시간 조정이 가능한 레이저주입장치를 사용하여 원하는 오류 반응을 일으킬 수 있는 어셈블리어의 수행 시점에 오류를 주입한다.

실제 레이저 주입 시점을 찾기 위해서 [그림 4]와 같이 간단한 for 순환 연산 구문를 구현하고, 구현된 어셈블리어를 분석하였다. for 순환 구문의 구조는 트리거를 위한 1번과 5번 과정을 제외한 3단계의 과정으로 구성되는데, 최초의 초기화 과정 이후에는 for 순환 구문의 최종적인 반복횟수를 확인하기 위한 연산을 수행하는 3번 과정과 255와의 차이(subi)를 통해 루프 횟수를 증가시키는 4-1번, 루프 횟수와 최종 연산 횟수와 비교(cpi)하여 3번 과정으로 되돌아가거나 5번 과정으로 순환 구문을 빠져나가는 것을 선택(brcs)하는 4-2번 과정을 반복적으로 수행하게 된다. 알고리듬이 수행한 후의 출력 값은 3번 과정을 통해 확인된 for 순환 구문의 반복 횟수이다. 이는 [그림 5]의 전력 소비 파형과 칩의 동작 클럭 신호의 비교 분석을 통해 확인할 수 있다.

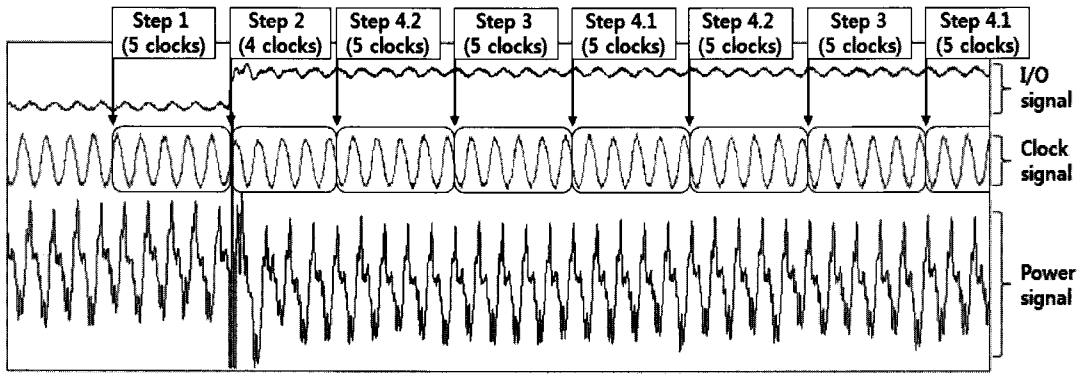
만약 주입된 레이저 오류가 하나의 어셈블리어 명



[그림 3] 오류주입 환경

Step 1	I/O = 5V	
	354: e2 e3 ldi r30, 0x32	// 1 clock
	356: f0 e0 ldi r31, 0x00	// 1 clock
	358: 8f ef ldi r24, 0xFF	// 1 clock
	35a: 80 83 st Z, r24	// 2 clock
Step 2	for (i=0;i<100;i++)	
	35c: 18 8a std Y+25, r1	// 2 clock
	35e: 06 c0 rjmp .+12 ; 0x36c	// 2 clock
Step 3	a = a+1;	
	360: 8f 85 ldd r24, Y+27	// 2 clock
	362: 8f 5f subi r24, 0xFF	// 1 clock
	364: 8f 87 std Y+27, r24	// 2 clock
Step 4.1	for (i=0;i<100;i++)	
	366: 88 89 ldd r24, Y+25	// 2 clock
	368: 8f 5f subi r24, 0xFF	// 1 clock
	36a: 88 8b std Y+16,r24	// 2 clock
Step 4.2	36c: 88 89 ldd r24, Y+25	// 2 clock
	36e: 84 36 cpi r24, 0x64	// 1 clock
	370: b8 f3 brcs .-18; 0x360	// 1/2 clock
Step 5	I/O = 0V	
	372: e2 e3 ldi r30, 0x32	// 1 clock
	374: f0 e0 ldi r31, 0x00	// 1 clock
	376: 10 82 st Z, r1	// 2 clock

[그림 4] for 명령어를 사용한 테스트 프로그램 (어셈블리 코드)



〔그림 5〕 클럭 신호와 전력 파형 신호의 비교를 통한 단계 분석

령 구문을 변형시킬 수 있다고 가정하면, 가능한 오류들은 아래와 같다.

- 루프 횟수인 i값의 잘못된 호출(ldd) 또는 i 값 증가 부분의 생략 또는 오류(subi)에 의한 i의 변형
- 100(0x64)와 비교(cpi)나 분기이동(brcs) 단계의 오류 또는 생략을 통한 순환구문의 탈출

4-1 단계를 수행할 때, 'ldd'에 의한 호출 구문이나 'subi'에 의해 루프 횟수가 변경되는 구문에서 호출 구문의 생략, 'subi'에 의해 차분될 값의 변형 또는 'subi' 변경 구문의 생략 등을 통해서 루프 횟수가 직접적으로 변형될 수 있다. 4-2 단계의 분기에서는 최종 연산 횟수인 '100'의 변형(ldd)이나 비교 구문(cpi), 비교 결과에 의한 회귀 구문(brcs)의 생략을 통해서 유도될 수 있다. 특히 비교 구문이나, 회귀 구문의 오류 발생시 해당 루프 연산에서 빠져나올 것으로 예측하였다.

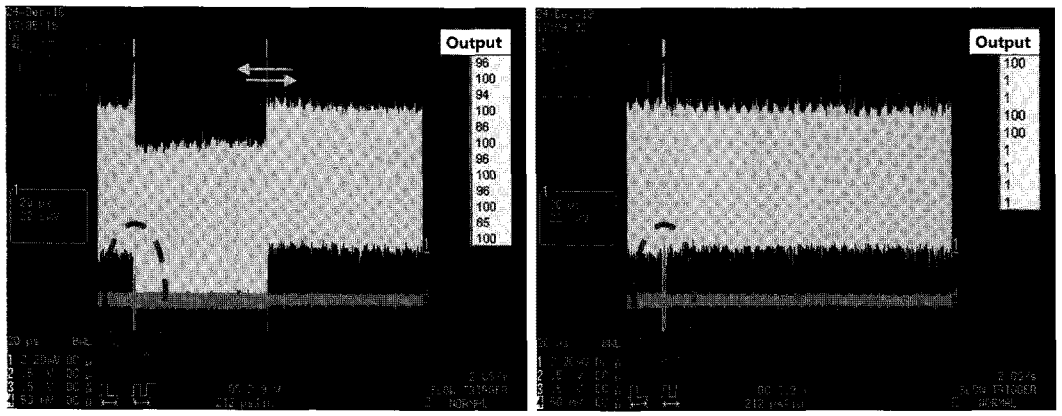
예측한 오류의 형태를 확인하기 위해서 for 순환 연산 구문을 ATmega128 칩에 탑재하여 실제로 레이저 오류를 주입하였다.

〔그림 6〕의 (a)는 그림 4의 4.1번 단계에서의 오류주입 결과이며, (b)는 4-2번 단계의 오류주입 결과이다. 본 실험은 〔그림 5〕의 분석에 따라 각 단계에서의 오류주입을 용이하게 하기위해서 첫 번째 for 구문이 수행된 이후를 오류주입 시점으로 하였으나, 동작 클럭 신호 분석에 따라 다양한 루프 횟수에서 주입 가능 하다. 각 그림의 상단 신호는 연산 전체를 구분하기 위한 I/O 트리거 신호이며, 두 번째 신호는 for 구문의 소비전력 파형, 그리고 가장 하단의 신호는 레이저 주입 인가 신호로 피크 성분이 나타난 시점에서 레이저가 주입된다. (a)의 경우에는 루프 횟수 i가 랜덤하게 바뀜에 따라서 최종적인 수행 횟수가 100이 아닌 다양한 값으로 나타나며, 그에 따라 전체 연산의 수행 시간이 달라짐을 확인하였다. (b)의 경우에는 잘못된 'cpi' 수행이나 'brcs'의 오동작으로 인해 첫 번째 루프 연산 이후 탈출하여 연산을 종료함을 확인하였다.

이외에도 〔그림 4〕의 4.1번 단계에 오류를 주입하는 경우 루프 횟수가 100이상의 값으로 바뀌게 되면 4.2번 단계의 경우처럼 단번에 for 순환 연산을 탈출하는 경우가 발생하였다. 이는 〔그림 7〕에서 동작 클럭 신호와 오류 주입에 의해서 변형된 소비전력 파형의 위치를 통해서 확인할 수 있다. 상단의 두 신호들은 I/O 트리거 신호이며, 세 번째 신호는 동작 클럭 신호, 네 번째 신호는 소비전력 파형이다. 레이저 오류가 주입되는 시점에서 소비전력 파형의 변형이 발생하므로 이를 통해 오류가 주입된 위치를 동작 클럭 신호와 비교할 수 있다. 〔그림 7〕의 (a)의 경우 4.1번 단계의 1~3번째 클럭에 걸쳐 소비전력 파형의 변형이 발생하며, (b)의 경우 4.2번 단계의 3~4번째 클럭에 걸쳐 변형이 발생하였다. 그러나 (a)의 결과는 랜덤한 경우의 하나이기 때문에 변형되는 i값에 대한 확률적인 분석이 추가로 필요하므로 4.2번 단계에 비해서 비효율적이다. 따라서 4.2번 단계의 비교 단계에 레이저 오류가 주입되면, 원하는 시점에서 단번에 루프 연산이 종료되므로, 본 논문에서 가정하는 연산 횟수의 변경 모델은 쉽게 가능한 것으로 판단된다.

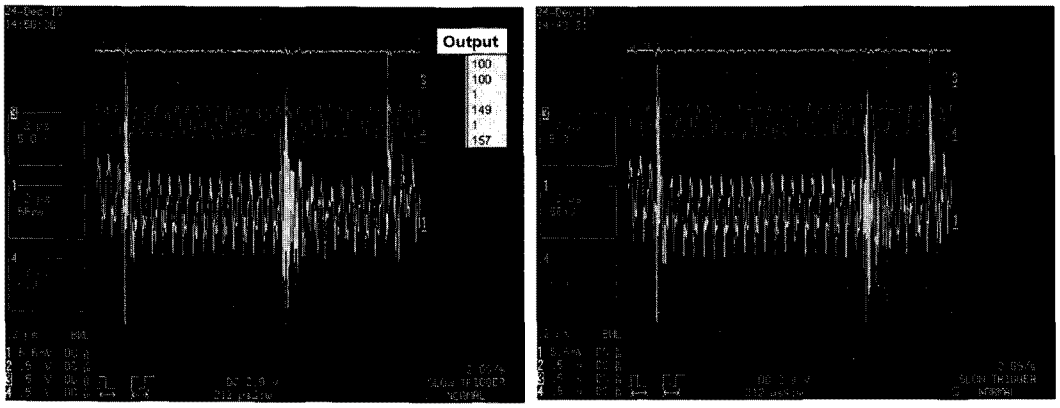
4.2 실험 고찰

실험 결과를 통해 루프 횟수 오류에 대한 모델이 실제로 구현이 가능함을 확인하였다. 추가로 어셈블리어



(a) (b)

〔그림 6〕 순환 구문에 대한 오류주입 결과



(a) (b)

〔그림 7〕 루프 연산 종료시 오류주입 위치 비교

분석에 의한 오류주입 대상의 정교화를 통해 기존에 제안되었던 루프 횟수의 변경에 국한되지 않고 원하는 루프 시점에서 연산을 단번에 종료하는 것이 가능함을 보였다. 즉, 원하는 시점에서 루프 연산을 종료할 수 있기 때문에 기존 공격들에서 가정한 것과 같이 루프 횟수가 연속적인 두 오류 결과문을 얻기 위한 오류주입 시도의 불필요한 반복을 줄일 수 있으며, 오류주입 시도에 대한 확률적인 분석이 필요하였던 기존의 가정을 간략화 할 수 있다. 이러한 실험적 결과는 제안하는 페어링에 대한 공격 방법 외에도 루프 횟수가 비밀값으로 사용되는 기존 RSA, ECC 등에 대한 오류공격에서도 사용될 수 있다. 예를 들어 RSA와 ECC에 대해서 루프 횟수가 연속적인 결과문을 통해 해당 루프 횟수에 해당하는 비밀 키의 비트를 알아내는 등의 공격은 이미 이론적으로 제안되어 있으나, 본 연구에서 실험으로 검증된 루프 횟수 오류 모델을 적용한다면 실현성을 갖출 수 있게 된다.

일반적인 마이크로프로세서에서 페어링 연산은 효율성을 높이기 위해서 보조적인 연산장치를 필요로 하고 있다. 따라서 큰 비트의 유한체를 사용하는 페어링 연산을 구현하려면, 범용 마이크로프로세서에 비해 연산 능력이 높은 마이크로프로세서가 필요하다. 실험적으로 레이저 오류주입이 가능함이 확인된 ATmega128 칩 환경에서는 512비트의 유한체를 사용하는 밀러 알고리듬의 수행이 불가능하기 때문에 유한체의 구성과 밀러 연산에 대한 효율적인 구현이 필요함을 확인하였다.

V. 결 론

본 논문에서는 페어링 연산 기법에서 가장 기본적으로 사용되는 아핀좌표상에서 구현된 밀러 알고리듬

에 대한 오류주입공격을 제안하였다. 오류주입으로 얻어진 오류 결과문들을 분석하여 비밀 값인 점 P의 좌표를 복원할 수 있는 수식을 유도하고 컴퓨터 시뮬레이션을 통해 검증하였다. 제안하는 공격은 밀러 알고리듬의 연산 횟수를 판별하는 분기 구문에 오류를 주입하여 원하는 루프 시점에서 연산을 단번에 종료시키는 방법을 사용한다. ATmega128 칩을 사용하여 레이저 오류주입을 실제로 수행하여 제안한 오류주입 모델이 구현 가능함을 실험적으로 검증하였다. 따라서 밀러알고리듬이 실제 칩에 구현되었을 때, 실험을 통해 검증된 오류주입 기법을 사용한다면 제안한 공격의 구현이 가능할 것이다. 또한, 실험을 통해 검증된 루프 횟수 오류 모델은 루프 형태로 구현된 페어링 연산의 다른 알고리듬이나 RSA, ECC 등의 암호 알고리듬에 대한 기존의 오류주입공격에도 적용될 수 있다.

참고문헌

[1] D. Boneh and M. Franklin, "Identity based encryption from the Weil Pairing," Advanced in Cryptology, Crypto 2001, LNCS 2139, pp.213-229, 2001.

[2] J.C. Cha and J.H. Cheon, "An Indentity-Based Signature from Gap Diffie-Hellman Groups," Proc. of PKC 2003, LNCS 2567, pp. 18-30, 2003.

[3] F. Hess, "Exponent group signature schemes and efficient identity based signature schems based on pairing," Proc. of SAC 2002, LNCS 2595, pp. 310-324, 2002.

[4] K.G. Paterson, "ID-based signature from pairings on elliptic curves," Electronics Letters, vol.38, no.18, pp. 1025-1026, Aug. 2002.

[5] A. Joux, "A One Round Protocol for Tripartite Diffie-Hellman," Proc. of Algorithmic Number Theory, LNCS 1838, pp. 385-393, 2000.

[6] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," Journal of Cryptology, vol.17, no.4, pp. 297-319, Sep. 2004.

[7] N.P. Smart, "An identity based authentication key agreement protocol based on pairing," Electronics Letters, vol.38, no.13, pp. 630-632, June 2002.

[8] C. Kim, J. Ha, and S. Moon, "A Blinding-Based Scalar Multiplication Algorithm Secure against Power Analysis Attacks," 정보보호학회논문지 17(3), pp. 117-121, 2007. 6.

[9] D. Page and F. Vercauteren, "A Fault Attacks on Pairing based Cryptography," IEEE Transactions on Computers, vol. 55, no.9, pp. 1075-1080, Sep. 2006.

[10] C. Whelan and M. Scott, "The Importance of the Final exponentiation in Pairings when considering Fault Attacks," Proc. of Pairing 2007, LNCS 4575, pp. 225-246, 2007.

[11] N.E. Mrabet, "What about Vulnerability to a Fault Attack of the Miller's Algorithm During an Identity Based Protocol?," Advances in Information Security and Assurance - ISA'09, LNCS 5576, pp. 122-134, June 2009.

[12] I.M. Duursma and H.S. Lee, "Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$," Advanced in Cryptology - Asiacrypt 2003, LNCS 2894, pp. 111-123, 2003.

[13] P. Barreto, S. Galbraith, C. O'hEigeartaigh, and M. Scott. "Efficient Pairing Computation on Supersingular Abelian Varieties," IACR ePrint 2004-375. Sep. 2005.

[14] S. Kwon. "Efficient Tate Pairing Computation for Supersingular Elliptic Curves over Binary Fields," IACR ePrint 2004-303. Nov. 2004.

[15] P. Barreto, H. Kim, B. Lynn, and M. Scott. "Efficient Algorithms for Pairing Based Cryptosystems," Advanced in Cryptology-CRYPTO 2002, LNCS 2442, pp. 354-368, 2002.

[16] F. Hess, N.P. Smart, and F. Vercauteren. "The Eta Pairing Revisited," IEEE Trans-

actions on Information Theory, vol.52, no.10, pp. 4595-4602, Oct. 2006.

[17] J. Lopez and R. Dahab, "Improved Algorithms for Elliptic Curve Arithmetic in $GF(2^n)$," Proc. of SAC'98, LNCS 1556, pp. 201-212, 1998.

[18] J. Siverman, The Arithmetic of Elliptic Curves, Springer-Verlag, 1986.

[19] V. Miller. "The Weil Pairing, and its Efficient Calculation," Journal of Cryptology, vol.17, no.4, pp. 235-261, Sep. 2004.

[20] R. Anderson and S. Skoroboatov, "Optical fault induction attacks," CHES 2002, LNCS 2523, pp. 31-48, 2003.

[21] 박제훈, 문상재, 하재철, "CRT-RSA 암호시스템에 대한 광학적 오류 주입 공격의 실험적 연구," 정보보호학회논문지 19(3), pp. 51-59, 2009. 6.

[22] P. Kocher, J. Jaffe, and B. Jun. "Differential Power Analysis," CRYPTO 1999, LNCS 1666, pp. 388-397, 1999.

[23] NTL, A Library for doing Number Theory, http://www.shoup.net/ntl/

[24] D.G. Cantor and H. Zassenhaus. "A New Algorithm for Factoring Polynomials Over Finite Fields". Mathematics of Computation, vol.36, pp.587-592, 1981.

[25] D.V. Chudnovsky and G.V. Chudnovsky. "Sequences of numbers generated by addition in formal groups and new primality and factorization tests". Advances in Applied Mathematics, vol.7, no.4, pp. 385-434, Dec. 1986.

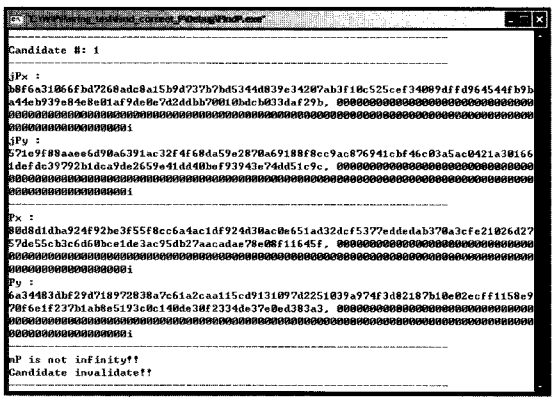
[26] M. Maas, "Pairing-Based Cryptography", Master Thesis, Technische Universiteit Eindhoven, 2004.

부 록 1 : 점 $[j]P$의 후보군에 대한 검증 시뮬레이션 결과

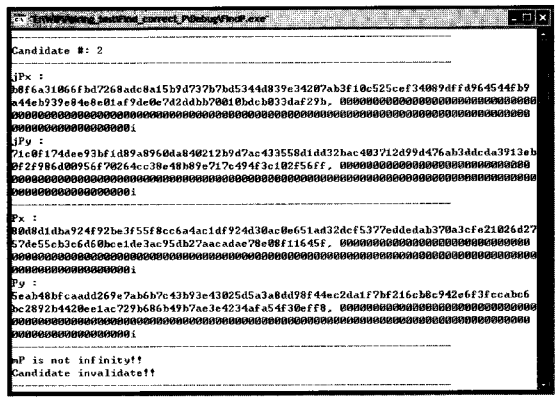
시뮬레이션을 위해서 구현한 검증 알고리듬은 먼저 $[j]P$의 후보군을 보여주고, $[j^{-1}][j]P$의 연산을 수행하여 후보에 대한 점 P의 좌표 값을 확인한 후, 스칼라 곱셈 $[l]P$의 값이 무한점이 되는지 검증한다. 마지막으로 무한점이 되는 P의 후보군에 한해서 정상적인 밀러 알고리듬을 수행하는 과정을 수행한다. [그림 8]은 각 후보군에 대해서 검증 알고리듬을 수행한 결과를 보여주고 있다.

[그림 8]의 (가)와 (나)의 후보 값들은 연산 결과 무한점이 되지 않음을 확인하였다. 그러나 (다)와 (라)에서 보여지는 후보 3과 후보 4의 값들의 경우 $[l]P$ 연산 결과가 무한점이 되었다. 후보 1, 2를 제외한 나머지 두 후보 값 3, 4 중에서 올바른 값을 찾기 위해서는 정상적인 밀러 알고리듬을 수행한 결과를 비교하여 정상 수행시의 결과 값과 동일한지 판별한다. (라)에서 나타난 바와 같이 후보 4의 밀러 알고리듬 결과 값이 아래의 정상 수행시의 결과 값과 같으므로 후보 4의 좌표가 점 P의 올바른 좌표 값임을 확인할 수 있다.

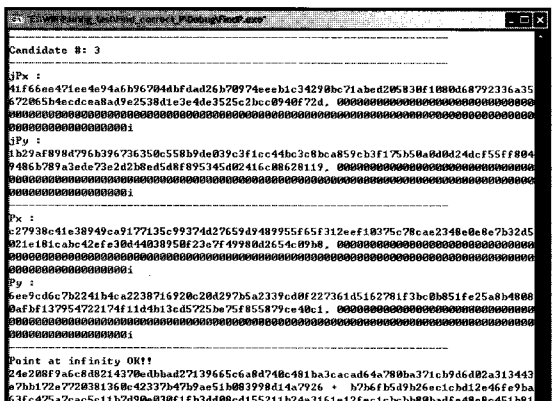
$$f_P(Q) = A3FD8803E30ED1613B62B6685CAC\\ 5756ADF38A765DACDFAFBFBA9\\ E080B78D4095D971310D1AAE10D4\\ 5645D7802A05527636B05F4D61A44\\ 4B5A7A598A8CB9FA75_{16}\\ + iB7B6FB5D9B26EC1CBD12EA6FE\\ 9BA63FC175A7CAC5C11b7D90E0\\ 30F1FB3DD08CD155211B24E3161\\ E12FEC1CBCBB80BADFE48E8C4\\ 51B811CFA96A31356E1A1C04E_{16}$$



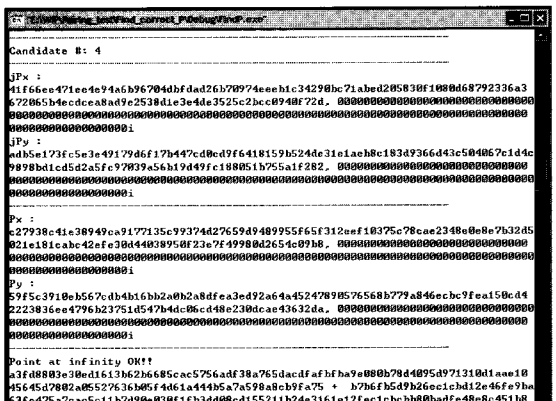
(가) 후보 1에 대한 검증



(나) 후보 2에 대한 검증



(다) 후보 3에 대한 검증

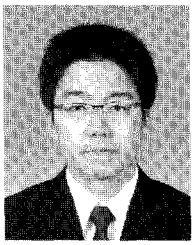


(라) 후보 4에 대한 검증

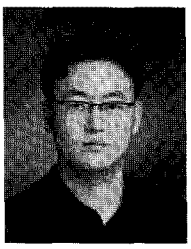
[그림 8] 검증 시뮬레이션 결과

〈著者紹介〉

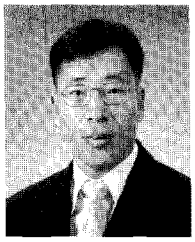
배 기 석 (KiSeok Bae) 학생회원
2006년 8월: 경북대학교 전자·전기공학부 졸업
2008년 8월: 경북대학교 전자공학과 석사
2009년 3월~현재: 경북대학교 전자전기컴퓨터공학부 박사과정
〈관심분야〉 정보보호, 네트워크 보안, 스마트카드 보안



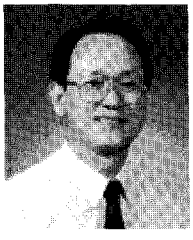
박 제 훈 (JeaHoon Park) 학생회원
2004년 2월: 경북대학교 전자·전기공학부 졸업
2006년 2월: 경북대학교 전자공학과 석사
2011년 2월: 경북대학교 전자공학과 박사
2011년 3월~현재: 국방기술품질원 선임연구원
〈관심분야〉 정보보호, 네트워크 보안, 스마트카드 보안



손 교 용 (Gyoyong Sohn) 정회원
2000년 2월: 영남대학교 수학과 졸업
2002년 2월: 경북대학교 산업응용수학과 이학석사
2008년 8월: 경북대학교 산업응용수학과 이학박사
2008년 9월~2010년 2월: 충북대학교 전자정보대학 박사후 연구원
2010년 3월~현재 : 경북대학교 전자전기컴퓨터공학부 박사후 연구원
〈관심분야〉 (초)타원곡선 암호 이론, 페어링 기반 암호 이론, 정보보호



하 재 철 (JaeCheol Ha) 종신회원
1989년 2월: 경북대학교 전자공학과 학사
1993년 8월: 경북대학교 전자공학과 석사
1998년 2월: 경북대학교 전자공학과 박사
1998년 3월~2007년 2월: 나사렛대학교 정보통신학과 부교수
2006년 7월~2006년 12월: QUT in Australia 연구 교수
2007년 3월~현재: 호서대학교 정보호호학과 부교수
2002년 3월~현재: 한국정보보호학회 이사
2009년 1월~현재: 한국산학기술학회 이사
〈관심분야〉 정보보호, 네트워크 보안, 스마트카드 보안



문 상 재 (SangJae Moon) 종신회원
1972년 2월: 서울대학교 공업교육(전자전공)과 학사
1974년 2월: 서울대학교 전자공학과 석사
1984년 6월: 미국 UCLA 전기공학과 박사
1984년 7월~1985년 6월: UCLA Postdoctor 근무
1984년 7월~1985년 6월: 미국 OMNET 컨설턴트
1997년 9월~1998년 8월: 경북대학교 전자전기공학부 학부장
1974년 12월~현재: 경북대학교 전자전기컴퓨터공학부 교수
2000년 8월~현재: 경북대학교 이동네트워크 정보보호기술 연구센터장
2002년 2월~현재: 한국정보보호학회 명예회장
〈관심분야〉 정보보호, 디지털 통신, 이동 네트워크