情報保護學會論文誌
第20卷 第5號, 2010. 10

오류 분석 공격에 대응하는 효율적인 DSA 서명 기법

배 기 석,[1†] 백 이 루,[2] 문 상 재[1], 하 재 철,[2‡]
[1]경북대학교, [2]호서대학교

An Efficient DSA Signature Scheme Resistant to the Fault Analysis Attack

KiSeok Bae,[1†] YiRoo Baek,[2] SangJae Moon[1], JaeCheol Ha,[2‡]
[1]Kyungpook National University, [2]Hoseo University,

요 약

오류 분석 공격은 암호용 디바이스가 동작하는 동안 오류를 주입한 후 그 결과를 분석함으로써 사용된 비밀키를 추출하는 물리적 공격 방법이다. 이러한 오류 분석 공격에 의해 표준 서명 알고리듬인 DSA(Digital Signature Algorithm)도 취약한 특성이 있음이 밝혀졌고 이를 방어하기 위한 대응책들도 제시된 바 있다. 본 논문에서는 오류 분석 공격을 방어할 수 있는 새로운 DSA 서명 기법을 제안한다. 제안 기법은 서명 연산시 한 번의 역수 연산만 추가되므로 타 대응 방식에 비해 효율적으로 구현할 수 있다.

ABSTRACT

The fault cryptanalysis is a physical attack in which the key stored inside of the device can be extracted by occurring some faults when the device performs cryptographic algorithm. Since the international signature standard DSA(Digital Signature Algorithm) was known to be vulnerable to some fault analysis attacks, many researchers have been investigating the countermeasure to prevent these attacks. In this paper we propose a new countermeasure to compute DSA signature that has its immunity in the presence of faults. Since additional computational overhead of our proposal is only an inverse operation in signature process, the proposed DSA scheme can be implemented more efficiently compared to previous countermeasures.

Keywords: DSA, Fault Analysis Attack, Bit Flip Error, Error Diffusion.

I. 서 론

최근 스마트카드와 같은 정보보호용 디바이스에 대한 물리적인 보안 문제가 주목을 받고 있다. 특히 정보보호용 암호 알고리듬을 개발할 당시 생각하지 못했던 하드웨어 구현상의 문제가 최근 심각한 보안 이슈로 대두되고 있다. 이 중에서 악의적인 공격자가 암호 알고리듬의 수행 중간에 오류를 주입하여 비밀키를 노출시키는 오류 분석 공격(fault analysis attack) 혹은 오류 주입 공격(fault insertion attack)에 대한 많은 연구가 있었다.

오류 주입 공격은 1997년 Bao 등에 의해 제시된 이후 많은 암호 알고리듬의 안전성을 위협하고 있다[1, 2]. 특히, 중국인의 나머지 정리에 기반한 RSA 암호 시스템에서는 단 한 번의 오류 주입 공격으로 비밀키 전체가 노출될 수 있으며 이에 대한 실험적인 연구 결과도 제시되고 있다[3-5]. 뿐만 아니라 이산대수 문제(discrete logarithm problem)에 기반한 공개키 암호 시스템 중 ElGamal 서명이나 DSA (Digital Signature Algorithm) 서명[6]에 대한 오류 주입 공격도 시도되었다. 특히, 현재 국제 표준으로 사용 중인 DSA에 대한 오류 주입 공격은 Bao 등에 의해 처음 제안되었는데 이때 공격 모델은 비밀

접수일(2010년 3월 15일), 게재확정일(2010년 7월 13일)
† 주저자, gith@ee.knu.ac.kr
‡ 교신저자, jcha@hoseo.edu

키에 대한 한 비트 플립 오류(flip error)를 가정하였다[1]. Bao 등의 공격에서는 한 번의 오류 주입으로 비밀키 한 비트를 찾아낼 수 있다. 또한, 2004년에는 Giraud와 Knudsen이 비트 오류 모델을 바이트 오류 모델로 확장하였는데 이 공격에서는 수천 개의 오류 서명이 필요하였다[7]. 이외에도 서명시 사용하는 랜덤 수 k의 최하위 바이트(Least Significant Byte, LSB)를 강제로 0으로 만드는 공격 모델이 Naccache 등에 의해 제시되었다. 여기에서는 각 k의 LSB를 0으로 리셋시킬 수 있다는 가정 하에서 총 27개의 오류 서명으로 비밀키를 추출할 수 있다고 주장하였다[8].

DSA 서명 알고리듬에 대한 오류 주입 대응책으로 먼저 오류 감지 센서 등을 설계하여 오류 주입 자체를 막는 방법을 고려할 수 있다. 다른 한편으로는 오류 주입 공격에 유연하게 대응할 수 있고 비교적 비용이 적게 드는 소프트웨어적인 방법이 있다. 소프트웨어적인 대응책으로는 오류 주입 여부를 검사하는 방법(error detection)이나 오류가 주입되었을 경우 이 오류를 확산(error diffusion)시켜 공격에 필요한 정보가 누출되는 것을 방지하는 방법 등이 있다. Bao 등은 오류 주입 공격을 방어하기 위해 DSA에 대한 몇 가지 오류 검사 방법[1]을 제시하였으며, 최근 Nikodem은 오류 확산 기법을 이용한 안전한 DSA 서명 방식[9, 10]을 제안한 바 있다.

본 논문에서는 오류 주입 공격을 방어하면서도 구현의 효율성이 높은 DSA 서명 방법을 제안하고자 한다. 논문에서 가정한 공격 모델은 Bao 등이 제시한 비밀키에 대한 비트 플립 오류이며 이 공격 모델에 기반하여 DSA 알고리듬의 취약성과 Nikodem의 대응 기법[10]을 집중 분석하였다. 한 비트에 대한 오류 주입은 오류 주입 위치와 시간을 정확히 알고 실행하는 현실적으로 매우 강한 오류 공격 모델이다[11]. 논문[12]와 [13] 등에서는 고정된 RAM이나 EEPROM의 비트를 바꾸는 실험이 있었지만 암호 알고리듬이 장착되어 수행 중인 과정에서는 한 비트 오류 주입은 용이하지 않아 실험적으로 DSA 공격에 성공한 결과는 아직 없다. 제안하는 DSA 서명 방식은 기존의 파라미터나 연산 함수를 그대로 활용하면서 한 번의 역수 계산을 추가하는 계산 오버헤드만 가지고 있어 타 대응 방식에 비해 효율적으로 구현할 수 있다. 그리고 Bao 등의 오류 공격 모델을 일반 DSA 서명 알고리듬에 적용하면 비밀키를 추출할 수 있지만, 제안 방식에는 적용되지 않음을 컴퓨터 시뮬레이션을 통해서 확인하였다.

II. DSA 서명 방식에 대한 오류 분석 공격

2.1 DSA 서명 방식

DSA는 1991년 NIST(National Institute of Standard and Technology)가 미국 전자서명 표준으로 사용하기 위하여 발표한 전자 서명 알고리듬이다[6]. DSA는 ElGamal 서명의[14] 변형으로서 그 안전성은 이산대수 문제의 어려움에 기반하고 있다. DSA 서명 시스템에 사용되는 변수들은 다음과 같다.

- p : 512비트 이상의 소수
- q : 160비트 소수, 단, q는 $p-1$을 나눌 수 있다.
- g : q 위수를 가지는 생성자 $g \in Z_p^*$.
- $h(\ )$: 해쉬 함수, $\{0,1\}^* \rightarrow Z_q$
- a : 사용자의 비밀키 $1 \le a \le q-1$
- y : 사용자의 공개키 $y = g^a \bmod p$

각 사용자는 비밀키 a를 비밀로 보관하며 공개 정보 $<p, q, g, y>$를 저장하고 있다.

2.1.1 서명 알고리듬

입력 메시지 m을 서명하기 위해, 서명자는 랜덤 수 $k(0 < k < q)$를 선택한 후 다음 수식을 이용하여 서명 (r, s)를 계산한다. 이를 나타낸 것이 [그림 1]이다.

2.1.2 검증 알고리듬

m에 대한 서명 값 (r, s)를 받은 수신자는 다음 식을 확인함으로써 서명을 검증한다.

$w = s^{-1} \bmod q$

$v_1 = g^{h(m)w} \bmod p, \quad v_2 = y^{rw} \bmod p$

$(v_1 \cdot v_2 \bmod p) \bmod q = ?\ r$

검증식의 마지막 식이 성립하면 정당한 서명으로 받아들이고 그렇지 않으면 서명을 거부하게 된다.

입력 : 메시지 m, 비밀키 a, 공개키 y, 모듈러스 p, q 생성자 g 출력 : 서명 (r, s)
1. 랜덤 수 $k(0 < k < q)$를 선택한다. 2. $r = (g^k \bmod p) \bmod q$를 계산한다. 3. $s = k^{-1}(h(m) + ar) \bmod q$를 계산한다.

[그림 1] DSA 서명 알고리듬

2.2 오류 주입 공격 분석

Bao 등의 DSA에 대한 오류 주입 공격에서는 서명 알고리듬이 수행되는 동안 비밀키 a에 비트 오류를 주입한다고 가정하였다[1, 10]. 즉, 비밀키 a가 t비트로 구성되어 있으며 공격자는 비밀키가 저장된 레지스터에 오류를 넣어 한 비트 플립 오류(flip error)가 발생한다고 가정하였다. 따라서 비밀키의 i번째 비트 플립이 발생한 것을 수식으로 나타내면 오류 키 $\bar{a}$는 $a \pm 2^i$와 같다[1]. 오류가 주입된 후 DSA의 서명 출력 $(r, \bar{s})$은 다음과 같이 계산된다.

$r = (g^k \bmod p) \bmod q$

$\bar{s} = k^{-1}(h(m) + \bar{a}r) \bmod q$

그러므로 오류 서명 $(r, \bar{s})$를 받은 공격자는 다음과 같이 T를 계산한다.

$\bar{w} = \bar{s}^{-1} \bmod q$

$\bar{v_1} = g^{h(m)\bar{w}} \bmod p, \quad \bar{v_2} = y^{r\bar{w}} \bmod p$

$T = \bar{v_1} \cdot \bar{v_2} \bmod p$

이때 T를 다시 정리하면 아래와 같다.

$T = g^{(h(m)\bar{w} \bmod q)} y^{(r\bar{w} \bmod q)}$

$= g^{(\bar{w}(h(m)+ar) \bmod q)} \bmod p$

여기서 공격자는 $i = 0, 1, 2, \ldots, t-1$에 대해서 $R_i \equiv g^{(\bar{w}\, r\, 2^i \bmod q)} \bmod p$ 를 구하고 다음의 두 식을 계산한다.

$TR_i = g^{(\bar{w}(h(m)+r(a+2^i) \bmod q))} \bmod p$

$T/R_i = g^{(\bar{w}(h(m)+r(a-2^i) \bmod q))} \bmod p$

그리고 공격자는 다음 두 식을 확인해 봄으로써 오류가 발생했던 위치의 비밀키 한 비트를 찾을 수 있다.

$TR_i = r \bmod p \bmod q$ 이면, $a_i = 0$,

$T/R_i = r \bmod p \bmod q$ 이면, $a_i = 1$.

따라서 공격자는 서로 다른 i에 대해 반복적으로 $TR_i \bmod p \bmod q$와 $T/R_i \bmod p \bmod q$가 r과 같은지 비교함으로써 공격자는 비밀키 a의 i번째 비트를 찾을 수 있다. 만약 비밀키 비트가 0에서 1로 바뀌었다면 T에 R_i를 곱한 후 mod q연산을 하면 r과 같아질 것이고 비밀키 비트가 1에서 0로 바뀐 경우라면 T에 R_i를 나눈 후 mod q연산을 하면 r과 같아지게 된다.

2.3 오류 주입 공격 대응책 분석

2.3.1 비교 검사 방법

DSA 서명이 정확히 생성되었는지를 확인하는 방법으로 가장 확실한 방법은 서명을 출력하기 전에 검증 과정을 통해 서명이 정확한지 확인하는 방법이다. 그러나 이 방법은 서명 검증을 위해 약 2번의 멱승 연산이 더 필요하게 된다. 또한, 서명 연산 자체를 두 번 생성하여 서로 같은 출력이 생성되는지 비교함으로써 오류 주입 여부를 확인할 수도 있다. 그러나 이 방법은 일반 서명 연산의 두 배의 시간이 필요하게 되며, 특히 비밀키 a에 영구적인 오류(오류 $\bar{a}$가 서명 연산 동안 계속 지속되는 경우도 포함)나 두 번의 연산시 a에 동일한 오류가 발생하는 경우에는 오류 공격에 취약하다.

Bao 등은 위에서 설명한 오류 주입 공격을 제시하면서 보다 간단한 대응책을 제안하였는데, 그 방법은 미리 비밀키의 역수 a^{-1}를 저장하여 두고 이를 이용하여 서명 s가 정확히 계산되었는지 비교 검사하는 방법이다[1]. 즉, 서명 s는 $s = k^{-1}(h(m) + ar) \bmod q$와 같이 생성되므로 $a^{-1}(sk - h(m)) \bmod q$를 계산하여 이 값이 r과 동일한지 비교하여 같으면 서명을 출력하고 그렇지 않으면 서명 장치를 리셋하게 된다. 이 방법은 비밀키의 역수를 미리 계산하여 저장할 뿐만 아니라 비교 연산을 통해 오류 주입 여부를 판단하게 된다. 그러나 비교 연산은 또 다른 오류 주입으로 그 과정을 건너뛸 수 있다는 취약점이 있다. 그리고 실제 실험을 통해 비교 연산 과정을 건너뛰는 오류 주입 공격이 성공한 사례도 있었다[5].

2.3.2 오류 확산 방법

최근 Nikodem은 오류 주입 공격을 방어할 수 있는 DSA 서명 알고리듬을 제안하였다. 그의 대응책에서는 비밀키 a나 서명 과정 중에서 오류가 주입되면 그 오류가 서명문 전체에 확산되는 성질을 이용하였다[10]. [그림 2]는 Nikodem이 제안한 오류 주입 공격에 강인한 "Nikodem의 DSA 알고리듬-1"을 나타내고 있다.

[그림 2]에서는 오류 주입 여부를 확인하기 위해 단계 4에서 V를 계산한다. 만약 단계 3에서 오류가 주입되지 않았으면 $V = 0$이 된다. 즉, 이 방식에서는 단계 3에서 비밀키 a를 사용하는데 이 비밀키가 정상적으로 사용되었는지 여부를 단계 4를 통해서 검증하게 된다. 단계 4에서의 검증은 공개 키 y를 이용하여 위에서 사용한 비밀키가 정상적인 값인지 검사하게 된다.

입력 : 메시지 m, 비밀키 a, 공개키 y, 모듈러스 p, q 생성자 g 출력 : 서명 (r, s)
1. 랜덤 수 $k(0<k<q)$를 선택한다. 2. $r=(g^k \bmod p) \bmod q$를 계산한다. 3. $v=k+ar \bmod q$를 계산한다. 4. $V=((y^{-r}g^v \bmod p) \bmod q)-r \bmod q$ 를 계산한다. 5. $k'=(k \oplus V)^{-1} \bmod q$를 계산한다. 6. $s=k'(h(m)+v)-1 \bmod q$를 계산한다.

〔그림 2〕 오류 주입 공격에 대응하는 Nikodem의 DSA 알고리듬-1

Nikodem은 〔그림 2〕의 기법뿐만 아니라 이를 좀 더 단순화 시킨 대응책도 제안하였다. 이를 나타낸 것이 〔그림 3〕의 "Nikodem의 DSA 알고리듬-2"인데 대응책의 원리는 위에서 분석한 것과 유사하다. 〔그림 2〕의 기법과 차이가 나는 것은 단계 3에서 v를 계산할 때 k를 제외하여 $v=ar \bmod q$와 같이 계산한 것과 단계 4에서 V를 $((y^{-r}g^v \bmod p) \bmod q)-1 \bmod q$와 같이 계산하는 것만 다르다.

그러나 위의 두 가지 오류 확산 기법은 기존의 서명 방법에 비해 약 3배 정도의 연산이 필요하다. 따라서 〔그림 2〕의 DSA 알고리듬에서 연산량을 줄이기 위해 단계 4의 V값을 아래와 같이 계산할 수도 있다. 이와 같은 방식을 "Nikodem의 DSA 알고리듬-3"이라 하자.

$$V=((g^{-ar+v} \bmod p) \bmod q)-r \bmod q.$$

그러나 이 방식은 비밀키 a에 영구적인 오류나 단계 3과 4의 a에 동일한 오류가 발생하는 경우에는 오류 공격에 취약한 특성을 보인다.

또한, 미리 저장된 a의 역수 a^{-1}를 이용하여 〔그림 2〕의 V값을 아래와 같이 계산할 수도 있다. 이 방식을 "Nikodem의 DSA 알고리듬-4"라고 하자. 그러나 이 방법 역시 역수를 미리 저장하는 저장 공간이 필요하며 한 번의 멱승 연산이 추가적으로 필요함을 확인할 수 있다.

입력 : 메시지 m, 비밀키 a, 공개키 y, 모듈러스 p, q 생성자 g 출력 : 서명 (r, s)
1. 랜덤 수 $k(0<k<q)$를 선택한다. 2. $r=(g^k \bmod p) \bmod q$를 계산한다. 3. $v=ar \bmod q$를 계산한다. 4. $V=((y^{-r}g^v \bmod p) \bmod q)-1 \bmod q$ 를 계산한다. 5. $k'=(k \oplus V)^{-1} \bmod q$를 계산한다. 6. $s=k'(h(m)+v) \bmod q$를 계산한다.

〔그림 3〕 Nikodem의 DSA 알고리듬-2

$$V=((g^{a^{-1}(v-k)-r} \bmod p) \bmod q)-1 \bmod q.$$

III. 오류 분석 공격에 대응하는 DSA 서명 제안

본 절에서는 오류 분석 공격에 안전하고 메모리 저장 공간과 연산량이 적은 DSA 서명 방식을 제안한다. 오류 공격 모델은 앞장에서 기술한 공격 모델로 가정한다. 제안하는 방식은 오류를 검출하여 서로 비교하는 방법이 아닌 오류가 확산되는 기법을 이용하여 설계한다.

3.1 제안하는 DSA 서명 기법

제안하는 오류 주입 공격에 대응하는 새로운 DSA 서명 알고리듬을 도시한 것이 〔그림 4〕이다. 이 방식에서 가장 큰 특징 중 하나는 r을 생성할 때 생성자 g를 사용하는 것이 아니라 자신의 공개키 y를 사용하여 r을 계산한다는 점이다. 따라서 서명 r은 $(y^k \bmod p) \bmod q=(g^{ak} \bmod p) \bmod q$가 되므로 〔그림 1〕의 기존 방식과 비교해 보면 랜덤 수 k에 비밀키 a를 곱한 후 g에 대한 멱승 연산으로 r을 구한 것과 같다.

따라서 다른 서명 쌍 s를 구하는 수식은 아래와 같게 된다. 즉, 〔그림 1〕에서 k^{-1}를 곱하던 것을 $(ak)^{-1}$를 곱하여 전개한 것과 같게 된다.

$$\begin{aligned} s &= (ak)^{-1}(h(m)+ar) \bmod q \\ &= ((ak)^{-1}h(m)+k^{-1}r) \bmod q \end{aligned}$$

제안 방식에서는 s를 생성할 때 비밀키 a를 사용하지 않고 a의 역수를 사용하도록 설계하였다. 이렇게 함으로써 비밀키 a에 한 비트 오류가 발생하더라고 비선형 함수인 역수 연산을 통해 오류가 확산되도록 하였다.

만약 〔그림 4〕와 같은 서명 알고리듬을 사용하더라도 검증 방법은 일반 DSA 에서 사용했던 것과 동일함을 알 수 있다. 즉, m에 대한 서명 값 (r, s)를 받은 수신자는 다음과 같이 계산하여 최종 검사 과정이 통과하는지 확인함으로써 서명을 검증한다.

입력 : 메시지 m, 비밀키 a, 공개키 y, 모듈러스 p, q 생성자 g 출력 : 서명 (r, s)
1. 랜덤 수 $k(0<k<q)$를 선택한다. 2. $r=(y^k \bmod p) \bmod q$를 계산한다. 3. $s=(ak)^{-1}h(m)+k^{-1}r \bmod q$를 계산한다.

〔그림 4〕 제안하는 오류 주입 공격에 강인한 효율적인 DSA 서명 알고리듬

$w = s^{-1} \bmod q$

$v_1 = g^{h(m)w} \bmod p$, $v_2 = y^{rw} \bmod p$

$(v_1 \cdot v_2 \bmod p) \bmod q = ?\ r$

(증명) DSA 검증 알고리듬

서명식 : $s = (ak)^{-1}h(m) + k^{-1}r \bmod q$.

따라서 $k = s^{-1}(a^{-1}h(m) + r) \bmod q$ 이다.

검증식 : $(v_1 \cdot v_2 \bmod p) \bmod q$

$= (g^{h(m)w}\ y^{rw} \bmod p) \bmod q$

$= (g^{h(m)w}\ g^{arw} \bmod p) \bmod q$

$= (g^{w(h(m)+ar)} \bmod p) \bmod q$

$= (g^{aw(a^{-1}h(m)+r)} \bmod p) \bmod q$

$= (y^{s^{-1}(a^{-1}h(m)+r)} \bmod p) \bmod q$

$= (y^k \bmod p) \bmod q$

$= r$

3.2 공격에 대한 안전성

제안 방식이 위에서 언급한 공격 모델에 안전한지 여부를 검증하기 위해 서명 알고리듬이 수행되는 동안 공격자는 비밀키가 저장된 레지스터에 오류를 넣어 한 비트 플립 오류가 발생한다고 가정하자. 서명 s가 생성되는 동안 a의 i번째 비트가 보수(complement)로 바뀐다고 가정하면 오류가 주입된 후 DSA의 서명 출력 $(r, \bar{s})$은 다음과 같이 계산될 것이다.

$r = (y^k \bmod p) \bmod q$를 계산한다.

$\bar{s} = (\bar{a}k)^{-1}h(m) + k^{-1}r \bmod q$를 계산한다.

따라서 공격자는 Bao 등의 공격과 동일한 방법으로 다음과 같이 T를 계산한다.

$\bar{w} = \bar{s}^{-1} \bmod q$

$\bar{v_1} = g^{h(m)\bar{w}} \bmod p$, $\bar{v_2} = y^{r\bar{w}} \bmod p$

$T = \bar{v_1} \cdot \bar{v_2} \bmod p$

이때 T를 다시 정리하면 아래와 같다.

$T = g^{(h(m)\bar{w} \bmod q)} y^{(r\bar{w} \bmod q)}$

$= g^{(\bar{w}(h(m)+ar) \bmod q)}$

$= y^{\bar{w}(a^{-1}h(m)+r) \bmod q} \bmod p$

따라서 공격자가 TR_i나 T/R_i를 계산했을 때 $y^k \bmod p$가 되는 R_i는 다음과 같이 표현된다.

$R_i = y^{\bar{w}((a+2^i)^{-1}h(m)+r)} / y^{\bar{w}(a^{-1}h(m)+r)} \bmod p$

$= y^{\bar{w}h(m)((a+2^i)^{-1}-a^{-1})} \bmod p$

공격자는 비밀키 비트 추출을 위해 R_i값을 계산할 수 있어야 하지만 이러한 R_i값은 비밀 키 a를 알지 못하면 구할 수 없는 값이다. 간단히 예를 들어 비밀키 a의 최하위 비트에 오류가 주입되었다고 가정하면 $i = 0$이고 R_0값은 아래와 같다. 하지만 이 값은 a에 대한 정보 없이는 구할 수 없다.

$R_0 = y^{\bar{w}h(m)((a+1)^{-1}-a^{-1})} \bmod p$

기존 DSA 서명 알고리듬에서는 서명 s를 만들 때 비밀키 a를 그대로 사용하는 형태이므로 T와 R_i를 아래와 같이 쉽게 구한 후 $r = (g^k \bmod p) \bmod q$와 비교할 수 있었다.

$T = g^{(h(m)\bar{w} \bmod q)} y^{(r\bar{w} \bmod q)}$

$= g^{\bar{w}(h(m)+ar) \bmod q} \bmod p$,

$R_i \equiv g^{\bar{w}\, r\,(a+2^i) - \bar{w}\, r a} = g^{\bar{w}\, r\, 2^i \bmod q} \bmod p$.

그러나 제안 방식에서는 서명 s를 만들 때 비밀키 a를 그대로 사용하지 않고 a^{-1}를 사용하는 구조이므로 동일한 공격에 대해 아래와 같이 T값이 계산된다. 따라서 역수 연산의 비선형성(nonlinearity) 때문에 비밀키에 대한 정보없이 $r = (y^k \bmod p) \bmod q$와 비교할 R_i값을 구할 수 없다. 따라서 위에서 사용한 오류 주입 공격 방법을 적용할 수 없게 된다.

$T = g^{(h(m)\bar{w} \bmod q)} y^{(r\bar{w} \bmod q)}$

$= g^{(\bar{w}(h(m)+ar) \bmod q)}$

$= y^{(\bar{w}(a^{-1}h(m)+r) \bmod q} \bmod p$

$R_i = y^{\bar{w}((a+2^i)^{-1}h(m)+r)} / y^{\bar{w}(a^{-1}h(m)+r)} \bmod p$

$= y^{(\bar{w}h(m)((a+2^i)^{-1}-a^{-1})) \bmod q} \bmod p$

제안한 DSA 서명 알고리듬에서는 서명 r값을 생성할 때 비밀키 a가 포함된 공개 키 y를 밑수로 사용함으로써 최종 검증식이 $r = (y^k \bmod p) \bmod q$가 되도록 하였다. 따라서 이때 사용된 k값은 $k = s^{-1}(a^{-1}h(m)+r) \bmod q$와 같으므로 비밀키 a와 비선형적 관계를 가지도록 구성하였다. 따라서 서명 s를 구하는 동안 레지스터에 저장된 비밀키 a에 한 비트 오류가 발생하더라도 a에 대한 역수를 구하는 과정에서 오류가 확산되도록 설계하였다. 서명 s를 생성할 때 비밀키에 한 비트 오류 e가 주입되면 오류 서명 $\bar{s}$는 아래와 같이 계산된다. 여기서 공격자는 E를 알고 있어도 k나 a에 관한 정보를 얻을 수 없다.

$\bar{s} = ((a+e)k)^{-1}h(m) + k^{-1}r \bmod q$

$= (ak+ke)^{-1}h(m) + k^{-1}r \bmod q$

$= ((ak)^{-1} + E)h(m) + k^{-1}r \bmod q$

$= s + Eh(m) \bmod q$

한편, 제안하는 대응책은 비밀키에 대한 오류 주입을 가정한 것이므로 Naccache 등이 제시한 공격 모델[8]에는 적용되지 않는다. 그 이유는 Naccache 등의 공격 방법은 랜덤 수 k의 LSB 바이트를 0으로 고정시키는 것이므로 s를 $s = (ak)^{-1}h(m) + k^{-1}r \bmod q$와 같이 생성하더라도 공격에 필요한 1차식 $sk = ah(m) + r \bmod q$

이 성립하게 되어 비밀키 a를 찾아낼 수 있다. 그러나 Naccache 등은 논문에서 자신들의 공격에 대한 대응책을 제시하였는데, 제안 알고리듬에 그 대응책을 적용하면 두 오류 공격을 모두 방어할 수 있다.

IV. 효율성 분석 및 시뮬레이션

제안하는 DSA 서명 알고리듬과 기존의 알고리듬과의 효율성을 비교한 것이 〔표 1〕이다. 효율성은 서명 생성시 필요한 멱승 수와 추가적인 파라미터를 저장하는데 필요한 공간으로 나누어 비교하였다. 효율성 비교에서 160비트에 대한 역수, 모듈라 덧셈, 모듈라 곱셈 등은 멱승에 비해 계산량이 작기 때문에 필요한 멱승 수만 비교하였다[15, 16]. 비교 검사를 이용하는 방법 중 검증 후 서명을 출력하는 방식이나 두 번 서명 연산을 거쳐 비교하는 방법은 연산량이 많거나 영구적 오류에도 취약한 특성이 있다. 또한 비밀키의 역수를 저장하고 이를 이용하여 서명 결과를 검증하는 방식은 연산 효율면에서는 매우 유리하나 비교 연산을 건너뛰는 2차 오류 주입 공격에 취약할 수 있다.

이에 반해 Nikodem에 의해 제안된 4가지 오류 확산 방법은 비교 구문을 건너뛰는 공격에는 강하지만 오류를 확산시키기 위해 멱승 연산이 추가적으로 필요하다. 또한 영구적 오류 혹은 동일 오류 2회 발생시 공격에 취약할 수도 있으며 비밀키 역수 값을 미리 저장해야 하는 경우도 있다.

제안 방식은 비밀키에 대한 $(ak)^{-1} \bmod q$를 계산함으로써 비밀키의 역수를 저장하지 않으면서도 오류 주입 공격에 대응할 수 있다. 즉, s를 생성할 때 비밀키 a에 오류를 넣으면 $(ak)^{-1} \bmod q$과정을 거쳐 s전체에 오류를 확산하는 효과를 가지게 된다. 이러한 오류 확산 효과를 유도하기 위해 서명 r을 생성할 때는 생성자 g대신 서명자의 공개키 y를 사용하도록 설계한 것이다. DSA를 구현하는 환경에 따라 다르지만 만약 서명자의 공개키 y가 제공되지 않는다면 비밀키를 이용하여 사전에 공개키를 생성하여 저장한 후 사용할 수도 있다.

논문에서는 일반 DSA 서명은 Bao 등의 오류 주입 공격에 의해 비밀키가 노출될 수 있음을 보이고, 제안하는 대응 방식에는 이 공격이 적용되지 않음을 컴퓨터 시뮬레이션을 통해 검증하였다. 검증은 PC에서 DSA 알고리듬을 구현한 후 소프트웨어적인 방법으로 오류 주입을 가정하고 시뮬레이션 하였다. 구현한 DSA시스템은 소수 p가 512비트이고 q는 160비트를 가정하였다. 해쉬 함수는 SHA-1 알고리듬을 이용하였으며 money.txt라는 특정 파일을 서명하는 것으로 구성하였다. 시뮬레이션에 사용된 파라미터는 아래와 같으며 16진수로 표기하였다.

소수 p=8df2a494 492276aa 3d25759b
b06869cb eac0d83a fb8d0cf7
cbb8324f d7882e5 d0762fc5
b7210eaf c2e9adac 32ab7aac
49693dfb f83724c2 ec0736ee
31c80291

소수 q=c773218c 737ec8ee 993b4f2d
ed30f48e dace915f

생성자 g=626d0278 39ea0a13 413163a5
5b4cb500 299d5522 956cefcb
3bff10f3 99ce2c2e 71cb9de
fa24babf 58e5b795 21925c9c
c42e9f6f 464b088c c572af53
e6d78802

〔표 1〕 오류 주입 대응기법의 효율성 비교

대응 기법		서명생성시 멱승 수	추가적인 변수 저장	비고
일반 DSA 서명		1	-	오류 주입 공격 취약
비교 검사 방법	검증 후 서명 출력	3	-	비교 구문 skip(2차 오류)공격에 취약
	2회 서명 연산 비교	2	-	
	비밀키의 역수	1	a^{-1}	
오류 확산 기법	Nikodem 방식 -1	3	-	
	Nikodem 방식 -2	3	-	
	Nikodem 방식 -3	2	-	영구적 혹은 동일 오류 2회 발생시 공격 취약
	Nikodem 방식 -4	2	a^{-1}	비밀키 역수값 저장
	제안 방식	1	-	역수 연산이 1회 추가$(ak)^{-1}$

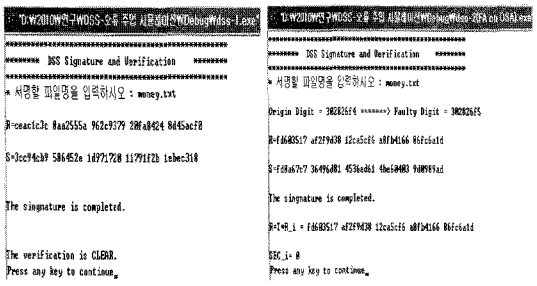
비밀키 a=634df67a 7c22d358 bca5f3fa
3e9d3f6e 302826f4
공개키 y=810e5859 3317694 e8f3172b
f79fa216 cf2d4b54 b17f4872
c5ac0a5d dc428726 ce555d40
754b55b3 269ebf6f 672be197
ae90869e 49628098 5519330e
84a86a75
해쉬값 $h(m)$=6ae4e6a 6d3e8489 7490be99
5d41b47c 2bfcae06

먼저 Bao의 오류 주입 공격이 동작함을 보인 화면이 [그림 5]이다. 그림의 (a)는 DSA 서명과 검증이 정상적으로 이루어지고 있음을 보이는 화면이며 (b)는 비밀키 a에 한 비트 오류가 발생했을 경우 공격자가 비밀키 비트를 추출하는 과정을 보인 것이다. 즉, 비밀키의 최하위 비트 a_0가 0에서 1로 바뀐 것을 가정하면 공격자는 정확히 최하위 비트를 찾아낼 수 있었다.

제안하는 서명 방식이 정상적으로 동작하며 오류 주입 공격에 강인함을 보인 화면이 [그림 6]이다. 그림의 (a)는 DSA 서명과 검증이 정상적으로 이루어지고 있음을 보이는 화면이며 (b)는 위의 공격 모델과 동일하게 비밀키 a에 한 비트 오류가 발생했을 경우에도 공격자가 비밀키 비트를 추출할 수 없음을 보인 것이다. 즉, 비밀키의 최하위 비트가 0에서 1로 바뀐 것을 가정하더라도 공격자는 최하위 비트를 알아낼 수 없다.

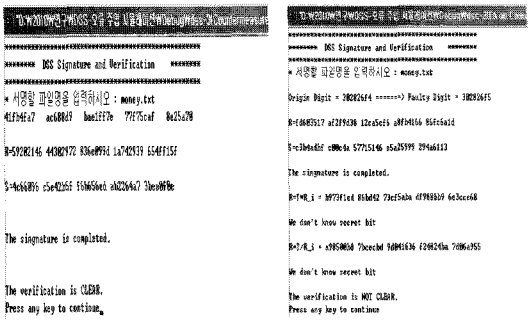
V. 결 론

본 본문에서는 DSA 서명 시스템에 대한 오류 주입 공격과 그 대응책을 분석하였다. 또한 오류 확산 기법을 이용한 효율적인 DSA 서명 방식을 제안하였다. 제안하는 DSA 서명 방식에서는 서명자의 공개키와



(a) (b)

[그림 5] DSA 서명 및 오류 주입 공격 시뮬레이션 화면



(a) (b)

[그림 6] 제안하는 DSA 서명 및 오류 주입 공격 대응 시뮬레이션 화면

비밀키를 동시에 이용하도록 하였으며, 공격자에 의해 비밀키에 비트 오류가 발생해도 오류가 확산되어 키 추출을 위한 정보를 얻을 수 없도록 설계하였다. 또한, 기존 DSA에서 사용되는 파라미터나 함수를 그대로 이용하면서 단지 1회 정도의 역수 연산만 추가하여 구현할 수 있다. 따라서 제안 방식은 메모리 공간이 제한적이거나 연산 성능이 낮은 암호 디바이스에서 오류 주입 공격에 대응하는 DSA를 구현하는데 적용할 수 있다.

참 고 문 헌

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

[2] E. Biham, A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," CRYPTO-1997, LNCS vol. 1294, pp. 513-525, 1997.

[3] D. Boneh, R. A. DeMillo and R. J. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults," EUROCRYPT-1997, LNCS vol. 1233, pp. 37-51, 1997.

[4] C. Aumuller, P. Bier, W. Fischer, P. Hofreiter, and J. P. Seifert, "Fault Attacks on RSA with CRT: Concrete

Results and Practical Countermeasures, CHES-2002, LNCS 2523, pp. 206-275, 2003.

[5] C. H. Kim, J. J. Quisquater, "Fault Attacks for CRT Based RSA: New Attacks, New Results, and New Countermeasures," WISTP-2007, LNCS vol. 4462, pp. 215-228, 2007.

[6] "National Institute of Standards and Technology," FIPS PUB 186-2: Digital Signature Standard, 2000.

[7] C. Giraud and E. Knudsen, "Fault Attacks on Signature Schemes," ACISP-2004, LNCS vol. 3108, pp. 478-491, 2004.

[8] D. Naccache, P. Nguyen, M. Tunstall and C. Whelan, "Experimenting with Faults, Lattices and the DSA," PKC-2005, LNCS vol. 3386, pp. 16-28, 2005.

[9] M. Nikodem, "Error Prevention, Detection and Diffusion Algorithms for Cryptographic Hardware," 2nd International Conference on Dependability of Computer Systems - DepCoS-RELCOMEX'07, pp. 127-134, June, 2007.

[10] M. Nikodem, "DSA Signature Scheme Immune to the Fault Cryptanalysis," CARDIS-2008, LNCS vol. 5198, pp. 61-73, 2008.

[11] J. BlÖmer, M. Otto and J. P. Seifert, "A new RSA+CRT algorithm secure against Bellcore attacks," In 10th ACM conference on Computer and Communication Security, pp. 311-320, Oct. 2003.

[12] S. P. Skorobogatov, R. J. Anderson, "Optical Fault Induction Attacks," CHES-2002, LNCS vol. 2523, pp. 31-48, 2003.

[13] J. J. Quisquater, D. Samyde, "Eddy current for magnetic analysis with active sensor," In the proceedings of E-Smart 2002, pp 185 - 194, Sept. 2002.

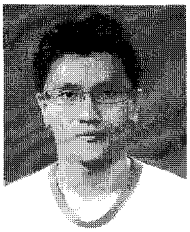
[14] T. ElGamal, "A Public-Key Cryptosystems and Signature Scheme Based on Discrete Logarithms," IEEE Trans, Information Theory, vol. IT-31, no. 4, pp. 469-472, July, 1985.

[15] A. Menezes. P. Oorschot, and S. Vanstone, "Handbook of Applied Cryptography," pp. 66-72, CRC Press, 1997.

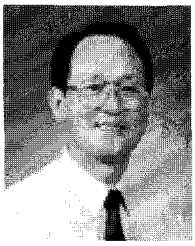
[16] C. H. Lim and P. J. Lee, "A Study on the Proposed Korean Digital Signature Algorithm," ASIACRYPT'98, LNCS vol. 1514, pp. 175-186, 2000.

〈著者紹介〉

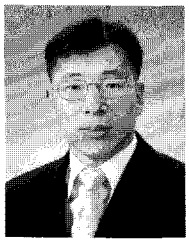
배 기 석 (KiSeok Bae) 학생회원
2006년 8월: 경북대학교 전자·전기공학부 졸업
2008년 8월: 경북대학교 전자공학과 석사
2009년 3월~현재: 경북대학교 전자공학과 박사과정
〈관심분야〉 정보보호, 네트워크 보안, 스마트카드 보안



백 이 루 (YiRoo Baek) 학생회원
2008년 8월: 호서대학교 정보보호학과 학사
2008년 9월~현재: 호서대학교 정보보호학과 석사과정
〈관심분야〉 네트워크 보안, 프로토콜, 스마트 카드 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



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