한국산학기술학회논문지
Vol. 11, No. 8 pp. 3045-3052, 2010

오류주입을 이용한 DSA 서명 알고리즘 공격 및 대응책

정철조[1], 오두환[1], 최두식[1], 김환구[1], 하재철[1*]
[1]호서대학교 정보보호학과

Cryptanalysis using Fault Injection and Countermeasures on DSA

Chul-Jo Jung[1], Doo-Hwan Oh[1], Doo-Sik Choi[1], Hwan-Koo Kim[1] and Jae-Cheol Ha[1*]
[1]Dept. of Information Security, Hoseo University

요 약 국제 표준 디지털 서명 알고리즘인 DSA(Digital Signature Algorithm)는 이산 대수 문제에 기반하여 이론적 안전성을 보장하지만 최근 서명 시스템 구동시 오류가 주입되면 디바이스 내부에 있는 비밀키를 노출시킬 수 있는 물리적 공격이 제시되었다. 본 논문에서는 Bao 등이 제시한 DSA의 비밀키 비트에 오류를 주입하는 오류 공격법을 소개하고, 서명에 사용되는 랜덤 수에 오류를 주입하는 새로운 공격 모델을 제안한다. 또한, 오류 확산 기법을 이용하여 두 가지 오류주입 공격을 모두 방어할 수 있는 대응책을 제시하고 컴퓨터 시뮬레이션을 통해 그 안전성과 효율성을 검증한다.

Abstract The international standard signature algorithm DSA has been guaranteed its security based on discrete logarithm problem. Recently, the DSA was known to be vulnerable to some fault analysis attacks in which the secret key stored inside of the device can be extracted by occurring some faults when the device performs signature algorithm. After analyzing an existing fault attack presented by Bao et al., this paper proposed a new fault analysis attack by disturbing the random number. Furthermore, we presented a countermeasure to compute DSA signature that has its immunity in the two types of fault attacks. The security and efficiency of the proposed countermeasure were verified by computer simulations.

Key Words : Digital Signature Algorithm(DSA), Fault Injection Analysis, Cryptographic Device

1. 서론

대부분의 정보보호 디바이스에는 안전하고 신뢰성 있는 서비스 제공을 위해 이산 대수 문제(Discrete Logarithm Problem, DLP)나 소인수 분해 문제(Integer Factorization Problem, IFP)와 같은 수학적 난제에 기반하여 개발된 암호 알고리즘이 구현되어 있다. 하지만 디바이스에 설계된 암호 시스템이 수학적으로 안전하더라도, 구현상의 허점을 이용하거나 물리적 영향에 의해 비밀키가 노출되는 공격에 취약할 수 있다[1,2].

특히, 오류주입 공격이라 불리는 물리적 공격 방법은 1996년 Boneh 등에 의해 처음 소개된 공격으로서 디바이스에 내장된 알고리즘이 구동될 때 오류를 주입하여 비밀키를 찾아내는 위협적인 공격 방법이다[2]. 장치 외부에 물리적 영향을 주어 오류를 주입하는 방법은 레이저, 빛, 전자파 등이 있으며 공격자는 내부의 메모리나 레지스터 등에 의도적으로 오류를 발생시켜 비밀키를 찾아낸다. 오류주입 공격에 관한 연구는 그 동안 AES(Advanced Encryption Standard)와 같은 블럭 암호 알고리즘이나 RSA 공개키 암호 시스템을 대상으로 주로 연구되었다[3,4].

1997년, Bao 등은 이산대수 문제에 기반하고 있는 ElGamal, DSA(Digital Signature Algorithm) 서명에서도 오류주입 공격이 가능함을 보여주었다[5]. 이 공격은 DSA 서명[6]에 사용되는 비밀키에 직접 오류를 주입하는 방식이다. 그 후 Bao 등의 공격에 대응하기 위한 방법들이 제안되었지만 연산량이 많아 비효율적인 특성을 가지고 있다[7,8]. 또한, 2004년에는 Giraud와 Knudsen이

*교신저자 : 하재철(jcha@hoseo.edu)
접수일 10년 07월 02일 수정일 10년 07월 15일 게재확정일 10년 08월 10일

비트 오류 모델을 바이트 오류 모델로 확장하였는데 이 공격에서는 수천 개의 오류 서명이 필요하였다[9].

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 가정하에서 총 27개의 오류 서명으로 비밀키를 추출할 수 있다고 주장하였다[10]. 이와 별개로 최근에는 Schmidt 등에 의해 ECDSA(Elliptic Curve DSA)에 대한 오류주입 공격이 제안되었는데 이 공격에서는 서명시 사용하는 랜덤 수에 대한 오류를 주입하여 랜덤 수의 일부 비트를 찾아내고 이를 이용하여 실제 비밀키를 추출하는 방식이다[11].

따라서 본 논문에서는 Schmidt 등이 ECDSA에서 사용한 랜덤 수에 대한 오류주입 모델을 변형하여 DSA에 대한 오류주입 공격을 제안한다. 또한 위에서 언급한 Bao 등의 공격과 제안하는 오류주입 공격을 모두 방어할 수 있는 대응책을 제시하며 이에 대한 안전성을 검증하고 효율성을 분석한다. 제안된 대응책들은 모두 오류확산 기법과 랜덤 수 복구 기법을 사용하여 공격자가 오류 서명 값을 이용하여 비밀키에 관한 어떤 정보도 얻을 수 없도록 설계하였다.

논문의 2장에서는 DSA 서명 방법과 기존의 Bao의 오류주입 공격 기법을 살펴본다. 그리고 3장에서는 DSA에서 사용하는 랜덤 수에 대한 오류주입 공격 방법을 제안한다. 4장에서는 위의 두 가지 오류주입 공격에 대한 대응책을 제안한다. 5장에서는 제안한 대응책을 안전성과 효율성면에서 분석하고 이를 컴퓨터 시뮬레이션을 통해 검증한다.

2. DSA 서명에 대한 오류주입 공격

2.1 DSA 서명 알고리즘

DSA는 1991년 NIST(National Institute of Standard and Technology)가 미국 전자서명 표준에서 사용하기 위하여 발표한 전자서명 알고리즘으로 그 안전성은 이산대수 문제의 어려움에 기반하고 있다[6]. 디지털 서명 알고리즘에 사용되는 시스템 파라미터들은 다음과 같다.

- p : 512비트 이상의 소수
- q : 160비트 소수로서 $p-1$의 약수
- g : Z_p^*의 원소로 위수 q를 가지는 생성자
- $h(\ )$: 해쉬 함수
- x : Z_q^*의 원소로서 사용자의 비밀키
- y : $y=g^x \bmod p$로 계산된 사용자 공개키

메시지 m에 대한 DSA 서명 식은 다음과 같으며 서명자는 서명 쌍 (r,s)를 생성한다.

서명 단계 1) q보다 작은 양의 랜덤 수 k 생성

서명 단계 2) $r=(g^k \bmod p) \bmod q$ 계산

서명 단계 3) $s=k^{-1}(h(m)+rx) \bmod q$ 계산

서명 검증자는 수신한 서명 쌍과 메시지 m을 이용하여 아래 식을 계산하고, 서명문 r과 동일한지를 확인하여 동일하면 정당한 서명으로 인정한다.

검증 단계 1) $r'=(g^{s^{-1}h(m)}y^{s^{-1}r} \bmod p) \bmod q$

검증 단계 2) $r=?\ r'$ 확인

특히, 서명 단계에서는 $g^k \bmod p$를 계산하는 멱승(exponentiation)과정이 필요한데 일반적으로 이진(binary) 알고리즘을 많이 사용한다. 구체적인 Left-to-Right 이진 멱승 알고리즘을 나타낸 것이 그림 1이다. Left-to-Right 이진 멱승 알고리즘은 t비트인 지수 k를 이진수로 표현한 후 그림과 같이 지수의 최상위 비트부터 하위 비트를 차례로 검색하면서 연산을 수행한다. 중간 계산 값을 C라 둔 후 이 값을 반복하여 제곱하면서 해당 비트가 1일 때만 메시지 m를 곱해 주는 방식이다. 여기서 k_i는 k의 i번째 비트를 의미한다.

```
1.  C=1
2.  for i=t-1 downto 0 do {
2.1     C=C·C mod p
2.2     if (k_i=1) C=C·m mod p
    }
3.  Return(C)
```

[그림 1] Left-to-Right 이진 알고리즘

2.2 Bao 등의 비밀키 비트 오류주입 공격

DSA 서명 알고리즘에 대한 최초의 오류주입 공격인 Bao 등[2]이 제안한 공격에서 공격자는 공격 대상 디바이스에 구현된 DSA 알고리즘을 수행시킬 수 있으며, 비밀키 x의 임의의 한 비트에 랜덤한 오류를 주입할 수 있다고 가정하였다. 이러한 가정하에 공격자는 다음과 같이 DSA 알고리즘에 오류주입 공격을 적용하여 비밀키를 찾아내었다.

① 공격자는 목표 디바이스에서 DSA 알고리즘이 동작하는 동안 비밀키 x의 i번째 비트에 오류를 주입한다. 오류가 주입된 비밀키는 $\tilde{x}=x\pm 2^i$라 표현할 수 있다.

② 공격자는 오류가 주입된 비밀키 $\tilde{x}$를 이용하여 다음과 같은 잘못된 오류 서명값을 얻을 수 있다.

- 서명자는 랜덤 수 k를 선택한 후, $r=(g^k \bmod p) \bmod q$를 계산한다.
- $\tilde{s}=k^{-1}(h(m)+\tilde{x}r) \bmod q$를 계산하고 서명쌍 $(r,\tilde{s})$를 출력한다.

③ 공격자는 오류 서명 값 $\tilde{s}$와 메시지 m를 이용하여 다음을 계산한다.

- $w=\tilde{s}^{-1} \bmod q$
- $T=g^{h(m)w}y^{rw} \bmod p$

그리고 아래와 같이 또 다른 검사 값을 계산한다.

- $R_i=(g^{rw})^{2^i} \bmod p,\ (i=0,\cdots,t-1)$

④ 공격자는 다음 수식이 성립하는지를 검사하여 비밀키 중 한 비트를 찾아낸다.

- 만약 $T \cdot R_i \bmod q = r$이면 $x_i=0$
- 만약 $T/R_i \bmod q = r$이면 $x_i=1$.

이 공격에서 공격자는 비밀키 x의 몇 번째 비트에 오류가 주입되었는지 모르기 때문에, ④ 단계의 i값을 바꿔가면서 반복 수행하여 비밀키 x의 한 비트를 알아낼 수 있다. 이러한 공격이 가능한 이유는 한 비트 오류에 의해 잘못된 비밀키 $\tilde{x}=x \pm 2^i$의 영향으로 다음 등식이 성립하기 때문이다.

$$\begin{aligned} r &= (g^{h(m)w}y^{rw}(g^{rw})^{\pm 2^i} \bmod p) \bmod q \\ &= (T \cdot R_i^{\pm 1} \bmod p) \bmod q \end{aligned} \quad (1)$$

2.3 ECDSA에 대한 오류 주입 공격

본 논문에서 제안하고자 하는 공격법은 위에서 설명한 Bao 등의 비밀키 x에 오류를 주입하는 방법과 달리 DSA 서명 연산시마다 랜덤하게 생성하여 사용하는 랜덤 수 k에 오류를 주입하는 공격이다. 제안하는 방식은 2009년 Schmidt 등이 ECDSA에 대한 오류주입 공격에 사용된 방법을 변형한 것이다. Schmidt 등이 사용한 공격 가정은 그림 2와 같은 ECDSA에서 $Q=kP$ 연산시 Left-to-Right 이진 스칼라 곱셈을 수행하게 되는데 이때 $(t-i)$번째 doubling 연산을 건너뛰는 것이다. 이 경우 k의 i번째 이후 비트 값을 $k'=(k_i,..,k_0)_2$로 표현하였으며 수식 (2)를 공격에 이용한다. 이때 Γ는 정상적인 계산 값으로 $\Gamma=kP$이며 Γ_i는 $(t-i)$번째 doubling 연산을 건너뛰어 잘못 계산된 값이다.

$$2\Gamma_i=\Gamma+k'P \quad (2)$$

그런데 Γ와 Γ_i는 ECDSA의 서명 출력 (γ, σ)으로부터 계산할 수 있어 랜덤 수의 일부인 k'를 계산할 수 있다. 단, 여기서 k'가 너무 큰 경우는 공격에 필요한 계산량이 너무 많으므로 작은 수의 k'가 발생되도록 오류를 주입하는 시점을 제어한다. Schmidt 등의 연구 결과에 의하면, 위와 같은 공격으로 얻어진 랜덤 수 k의 부분 정보들을 모아 Lattice 암호 해독법에 적용하면 비밀키를 충분히 계산할 수 있다[12]. 실제로 160비트 크기의 ECDSA 서명 시스템에서 160비트 중 12비트를 아는 50개의 k만 있으면 99%의 확률로 비밀키를 유추할 수 있다고 한다[11,13].

```
1.  Q = O(무한원점)
2.  for i = t-1 downto 0 do {
2.1     Q = 2Q
2.2     if (k_i = 1) Q = Q + P
    }
3.  Return(Q)
```

[그림 2] Left-to-Right 스칼라 이진 곱셈

3. 랜덤 수에 대한 오류주입 공격

본 논문에서는 ECDSA에 사용했던 Schmidt 등의 공격이 DSA에는 적용될 수 없음을 인지하고 이를 변형하여 랜덤 수 k의 부분 정보를 찾아내는 새로운 방법을 제안하고자 한다. 물론 k에 대한 부분 정보를 알고 있어도 Lattice 암호 해독법을 적용하여 비밀키를 계산해야 한다는 점은 기존의 연구 내용과 동일하므로 본 논문에서는 랜덤 수 k의 부분 정보를 찾아내는 공격만 다루고자 한다.

DSA 서명시 오류주입을 통해 랜덤 수의 일부 비트를 찾아내는 공격을 다음과 같이 제안한다. 공격자는 Schmidt 등이 사용한 공격 가정과 유사하게 그림 1의 멱승 연산시 $(t-i)$번째 제곱 연산을 건너뛰어 k가 정상적으로 사용되지 않게 하는 것이다. 즉, k에 대한 멱승 연산시 $g^k \bmod p$의 연산 과정을 잘못하게 함으로써 랜덤 수 k에 오류를 주입하는 결과를 초래하는 공격이다. 여기서 주의할 점은 k에 대한 직접적인 물리적 오류주입은 아니며 제곱 연산을 하는 서브 함수를 건너뛰는 공격임을 주지할 필요가 있다. 그렇지만 결국은 잘못된 k를 사용하는 결과를 가져오므로 "랜덤 수에 대한 오류주입 공격"이라 약칭한다. 여기서도 공격하고자 하는 k의 i번째 이후 비트 값을 $k'=(k_i,..,k_0)_2$로 표기한다.

① 공격자는 서명 $g^k \bmod p$ 생성 과정에서 $(t-i)$번째 제곱 연산을 건너뛰는 오류를 주입하여 그림 1의 단계 2.1을 건너뛰게 한다. 그 후 공격자는 출력되는 서명 쌍 $(\tilde{r}, \tilde{s})$를 얻게 된다.

② 여기서 오류를 주입하지 않은 원래의 결과를 $\beta = g^k \bmod p$라 하고, 오류를 주입하여 얻은 서명쌍 중 첫번째 것을 $\tilde{r}$라고 할 때 다음 수식을 만족하는 k'이 존재하게 된다.

$$\tilde{r} = (\sqrt{\beta} \bmod p \cdot \sqrt{g^{k'}} \bmod p) \bmod q \quad (3)$$

③ 공격자는 $\tilde{r}$는 알고 있으며 $\sqrt{\beta} \bmod p$값을 계산하여 위 식을 만족하는 k'를 계산할 수 있다. 즉, 결과적으로 오류가 주입된 랜덤키 k의 i번째부터 최하위 비트들을 알아낼 수가 있다. 여기서 $\sqrt{\beta} \bmod p$는 다음과 같은 등식을 이용하여 구하게 된다.

$$\sqrt{\beta} \bmod p = \sqrt{g^{\tilde{s}^{-1}h(m)} \bmod p} \cdot \sqrt{y^{\tilde{s}^{-1}\tilde{r}} \bmod p}$$

이 등식이 성립하는 이유는 아래 수식에 의해 서명 $\tilde{s}$가 생성되었기 때문이다.

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

그런데 위 공격에서 $\sqrt{\beta} \bmod p$를 구하기 위해서는 $\tilde{s}^{-1}h(m) \bmod q$, $\tilde{s}^{-1}\tilde{r} \bmod q$, k의 값이 모두 짝수라는 조건이 필요하다. 그 이유는, $\sqrt{\beta} \bmod p$를 구하기 위해서 $g^{\tilde{s}^{-1}h(m)} \bmod p$ 와 $y^{\tilde{s}^{-1}\tilde{r}} \bmod p$ 연산시 맨 마지막 제곱 연산을 생략함으로써 $\sqrt{g^{\tilde{s}^{-1}h(m)} \bmod p}$ 와 $\sqrt{y^{\tilde{s}^{-1}\tilde{r}} \bmod p}$ 를 쉽게 구할 수 있기 때문이다. 또한 k'는 아래 식을 만족하는 $\bar{k}$을 구한 후 이를 2배하면 $k' = 2\bar{k}$가 되어 k'를 구하게 된다.

$$\tilde{r} = (\sqrt{\beta} \bmod p \cdot g^{\bar{k}} \bmod p) \bmod q$$
$$= (\sqrt{\beta} \bmod p \cdot \sqrt{g^{k'}} \bmod p) \bmod q$$

여기서 Schmidt 등이 사용한 식 (2)와 제안하는 수식 (3)은 비슷해 보이지만 큰 차이를 가지게 된다. 만약 ECDSA에서 사용했던 수식 (2)를 DSA에 적용해 본다면 (4)식과 같은 형태가 될 것이다.

$$(\tilde{r}')^2 = \beta \bmod p \cdot g^{k'} \bmod p \quad (4)$$

그러나 이 경우 β는 구할 수 있지만 서명 $\tilde{r}$을 가지고 $\tilde{r}' \bmod p$나 $(\tilde{r}')^2 \bmod p$를 구할 수 없다. 그 이유는 서명 $\tilde{r}$는 $GF(q)$체 상으로 $\tilde{r} = (g^{k'} \bmod p) \bmod q$와 같이 계산된 결과이기 때문이다. 결국 $\tilde{r}$를 이용하여 $GF(p)$체 상의 $\tilde{r}'$값을 구할 수 없으므로 DSA에서는 (4)식을 이용할 수 없고 식 (3)을 이용하여 랜덤 수의 일부를 계산할 수 있다. 따라서 식 (3)을 이용한 연산을 하려면 $\tilde{s}^{-1}h(m) \bmod q$, $\tilde{s}^{-1}\tilde{r} \bmod q$, k의 값이 모두 짝수라는 제한 조건이 필요하다. 여기에서 $\tilde{s}^{-1}h(m) \bmod q$와 $\tilde{s}^{-1}\tilde{r} \bmod q$의 짝수 여부는 쉽게 구별할 수 있으며, k가 짝수인지의 여부는 식 (3)을 만족하는 k'을 찾는 과정에서 확인할 수 있다. 이와 같은 랜덤 수 k에 대한 부분 정보를 알고 있는 여러 개 서명쌍을 이용하여 Lattice 암호 해독법을 적용하면 Schmidt 등의 공격 방법과 같이 비밀키 x를 추출할 수 있다.

4. DSA에 대한 오류주입 공격 대응책

본 장에서는 위에서 설명한 Bao의 공격과 랜덤 키 오류주입 공격을 동시에 방어하는 대응책을 제안하고자 한다.

먼저 Bao의 공격은 비밀키 x에 직접 오류를 주입하는 공격이므로 x가 올바르게 사용되도록 하는 것이 중요하며 오류가 주입되었을 경우에는 그 오류가 서명문 $\tilde{s}$에 전체적으로 확산되어 공격에 필요한 식 (1)이 성립하지 않도록 하는 것이 중요하다. 따라서 논문에서는 오류 확산 기법을 이용하여 공격자가 오류 서명문을 가지고도 비밀키를 계산할 수 없도록 설계하였다. 여기에서는 비밀키 x에 대한 역원 $x^{-1} \bmod q$를 사전에 저장하여 오류 검증 및 확산에 이용하게 된다. Bao의 오류주입 공격을 방어하기 위한 서명 알고리즘을 기술한 것이 그림 3이다.

단계 1) q보다 작은 양의 랜덤 수 k 생성
단계 2) $r = (g^k \bmod p) \bmod q$을 계산
단계 3) $s_t = k_r^{-1}(h(m) + rx) \bmod q$를 계산
단계 4) $T = (s_t k_r - h(m))x^{-1} \bmod q \oplus r$를 계산
단계 5) $s = (s_t \oplus T) \bmod q$를 계산

[그림 3] Bao 등의 공격에 대한 대응 알고리즘

그림 3의 대응책에서 보면 단계 1), 2), 3)은 기존의 서명 r과 s를 생성하는 것과 비슷하다. 그러나 단계 3에서 랜덤 수 k를 사용하지 않고 복구된 k_r을 사용한다. 이 k_r은 다음에 설명할 랜덤 수 오류주입 공격을 방어하기 위해 생성한 것으로 정상적으로 복구되면 k와 동일한 값이 된다. 또한, 단계 4는 비밀키 x가 정상적으로 사용되었는

지를 확인하는 연산으로서 정상적인 경우 T값은 0이 되어 서명 s와 임시 서명 s_t는 동일한 값이 된다. 즉, 단계 4에서 r과 s의 관계를 사전 계산된 x^{-1}를 이용하여, 비밀키 x에 오류가 주입되었는지, 아닌지를 한 번 더 검증하게 된다. 만약 x^{-1}을 사전계산하지 않는다면, 오류가 전파된 비밀키 $\tilde{x}$에 대한 역원 $\tilde{x}^{-1}$이 검증에 사용되기 때문에 올바른 검증을 할 수 없게 된다. 따라서 x에 대한 역원 x^{-1}을 사전에 계산하여 저장한 후 이를 검증에 활용해야 한다.

결국, 단계 3에서 비밀키에 한 비트 오류가 발생하면 T는 의미없는 랜덤한 수가 되어 최종적인 오류 서명 $\tilde{s}$는 식 (1)의 등식을 만족하지 않게 된다. 따라서 이러한 오류 확산 기법을 이용하면 Bao 등의 비밀키에 대한 한 비트 오류주입 공격을 막을 수 있다. 즉, 각 단계별로 오류가 확산되는 과정은 아래와 같다.

단계 3) $\tilde{s}_t = k_r^{-1}(h(m) + r\tilde{x}) \bmod q$를 계산

단계 4) $\tilde{T} = (\tilde{s}_t k_r - h(m))x^{-1} \bmod q \oplus r$를 계산

단계 5) $\tilde{s} = (\tilde{s}_t \oplus \tilde{T}) \bmod q$를 계산

두 번째로 랜덤 수에 대한 오류주입 공격을 방어할 수 있는 대응책을 제안한다. 여기서 유의할 것은 서명 과정에서 고정된 비밀키 x는 서명 연산에 1번 사용되는 반면, 랜덤 수 k는 매 서명마다 생성되는 임시 비밀 값으로서 단계 2에서는 멱승의 지수로, 단계 3에서는 그 역원을 구하여 사용된다는 점이다. 따라서 제안하는 대응책에서는 단계 2에서 k에 대한 멱승을 시행하면서 동시에 그 역과정을 통해 k_r값을 복원하는 기법을 사용한다. 그리고 그 복원된 k_r을 다시 서명의 단계 3과 4에서 활용하자는 것이다. 이를 위해 단계 2에서 수행하는 멱승 알고리즘을 제시하면 그림 4와 같다. 여기서 "$C = C \cdot C \bmod p$ with $k_r = 2k_r$"의 의미는 모듈라 제곱 연산을 수행하는 서브 함수내에 k_r을 두배하는 연산을 포함하고 있다는 의미이다.

```
1.   C = 1, k_r = 0
2.   for i = t-1 downto 0 do {
2.1      {C = C · C mod p with k_r = 2k_r}
         if (k_i = 1) {
2.2      C = C · m mod p with k_r = k_r + 1}
     }
3.   Return(C, k_r)
```

[그림 4] 개선된 Left-to-Right 이진 알고리즘

그림 4는 그림 1과 C를 계산하는 과정은 동일하다. 그리고 추가적으로 멱승에 사용되는 랜덤 수 k를 순차적으로 복원하게 된다. 이 과정을 정상적으로 마치게 되어 복구된 k_r은 멱승에 사용된 k와 같은 값이 되는데 이 값을 그림 3의 서명 단계 3과 4에 사용함으로써 오류주입 공격을 방어하게 된다.

만약 3장에서 제시한 공격처럼 $(t-i)$번째 제곱 연산을 건너뛰게 되더라도 k_r에 관한 두 배 연산도 같이 건너뛰게 함으로써 멱승 연산에 사용된 k와 복구된 k_r는 항상 같게 된다. 그러므로 단계 2와 단계 3에서 사용된 랜덤 수는 같으므로 항상 정상 서명 (r, s)를 만들게 된다. 이러한 정상 서명은 공격자에게 오류 분석에 필요한 정보를 전혀 노출하지 않게 된다. 즉, 3장에서 제안한 랜덤 수 오류주입 공격은 [그림 3]에서 단계 2에서 오류를 주입하므로 단계 3과 단계 4에 사용된 랜덤 수가 서로 다를 경우에만 공격이 성공할 수 있는데 제안 알고리즘에서는 단계 2에서 k에 오류가 주입되더라도 그 복구된 k_r도 같은 오류가 주입되는 오류 확산 효과를 가져와 정상 서명 (r, s)를 만들게 되고 결국 랜덤 수에 대한 부분정보를 계산할 수 없게 된다.

5. 분석 및 시뮬레이션

5.1 안전성 및 효율성 분석

본 논문에서 제안한 대응책은 3장에서 설명한 두 종류의 오류주입 공격들을 각각 방어하기 위한 것이지만 하나의 서명 연산 과정에 통합하여 사용하여야 한다. 먼저 비밀키에 대한 오류주입 공격은 비밀키에 대한 직접적인 변형을 의미하므로 다른 시스템 파라미터를 이용하여 오류주입시 그 결과가 서명 전체로 확산되게 설계한 것이 특징이다. 그렇게 함으로써 공격 식 (1)이 성립하지 않도록 랜덤한 오류를 발생시키도록 하였다.

그림 3에서 오류가 주입된 경우, T를 연산하는 단계에서 오류 $\tilde{x}$와 x^{-1}이 서로 소거되지 않기 때문에 $(s_t k - h(m))x^{-1} \bmod q$는 r과 같지 않게 되며, 결과적으로 $T \neq 0$이 되며, 최종 서명은 비밀키를 추측하기 어려운 랜덤 값을 가지게 된다.

랜덤 수에 대한 오류주입 공격은 기본적으로 제곱 연산과정을 건너뛰어 잘못된 k를 사용하게 하는 결과를 가져와 오류 서명문을 이용하여 공격 식 (3)을 만족하는 랜덤 수 일부를 찾는 방법이다. 그러나 제안 방식은 멱승시

잘못된 k를 사용하게 되더라도 k_r을 복구하는 과정도 같이 건너뛰게 하는 대응책이다. 그러면 오류가 주입되어도 멱승에 사용된 랜덤 수 $\tilde{k}$와 복구된 $\tilde{k}_r$는 항상 같도록 만들어 오류 서명이 발생하지 않도록 하는 것이 특징이다. 멱승시 오류를 주입하였더라도 서명이 정상 서명이 되면 비밀키 공격에 필요한 정보를 얻을 수 없게 된다.

다음으로 제안하는 대응책을 사용함으로써 추가되는 연산량을 비교해 본다. 그림 4에서 k_r을 복구하는 연산은 모듈라 곱셈이나 제곱 연산에 비해 극히 미미하므로 연산량 분석에서 제외한다. 그림 1과 같이 알고리즘을 적용하면 추가되는 연산량은 단계 3과 4의 과정만 추가된다. 그러므로 $GF(q)$체 상에서 모듈라 곱셈 두 번과 한 번의 모듈라 덧셈 그리고 두 번의 XOR 연산이 추가된다. 그러나 이 연산량은 단계 2에서 멱승 연산에 비해 매우 작은 연산량이 된다. 표 1은 원래 DSA 알고리즘에 필요한 연산량을 기준으로 기존의 Nikodem의 대응책[8] 등과 비교한 것이다.

[표 1] 연산량 및 저장 공간 비교

구 분	멱승	역수	곱셈	덧셈	XOR	저장 메모리
원 DSA서명	1	1	2	1	-	-
검증후 출력	3	2	5	-	-	-
Nikodem[8]	3	1	3	3	1	-
제안 방식	1	1	4	2	2	1

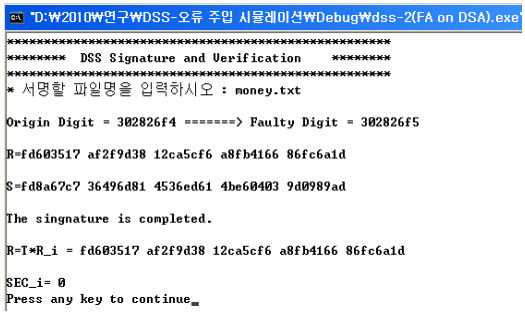
표 1에서 분석된 바와 같이 서명을 출력하기 전에 서명 검증을 실시하여 그 결과가 맞으면 서명쌍을 출력하는 방식은 서명 검증에 필요한 연산이 많이 추가되지만 오류주입 공격에 대응할 수 있는 안전한 방법이 된다. 반면 Nikodem의 대응책도 서명 검증과정을 거치게 되므로 멱승이 두 번 정도 더 필요하게 되어 약 3배정도의 연산량이 추가된다.

그러나 제안하는 대응책은 전체적으로 원래 DSA 서명 알고리즘보다 1~2% 정도의 추가 연산만 필요하다. 다만 비밀키의 역원 x^{-1}을 저장하는 160비트 크기의의 저장 메모리가 한 개 요구된다.

5.2 시뮬레이션

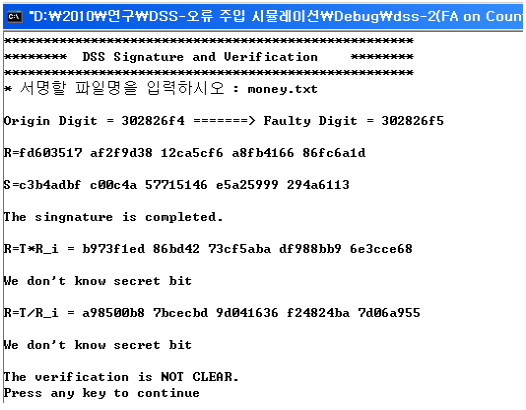
본 논문에서는 오류주입 공격의 타당성 검증을 위해 컴퓨터 시뮬레이션을 실시하였다. 물론 오류주입을 암호용 디바이스에 하지 않고 컴퓨터상에서 데이터를 변경하는 방식으로 오류주입을 가정하였다. 소프트웨어로 구현된 DSA에 대한 비밀키 오류주입 모델과 랜덤 수 오류주입 모델을 시뮬레이션 한 결과 그 공격 방법이 정확함을 확인할 수 있었다. 또한 대응책을 적용한 후 똑같은 오류주입 공격 모델을 시뮬레이션 한 결과 이 경우에는 비밀키를 추출할 수 없음을 검증하였다.

그림 5는 Bao의 오류주입 공격이 동작함을 보인 화면이다. 그림은 비밀키 x에 한 비트 오류가 발생했을 경우 공격자가 비밀키 한 비트를 추출하는 과정을 보인 것이다. 즉, 비밀키의 최하위 비트 x_0가 0에서 1로 바뀐 것을 가정하면 공격자는 정확히 최하위 비트를 찾아낼 수 있었다.



[그림 5] 비밀키 오류주입 공격 시뮬레이션

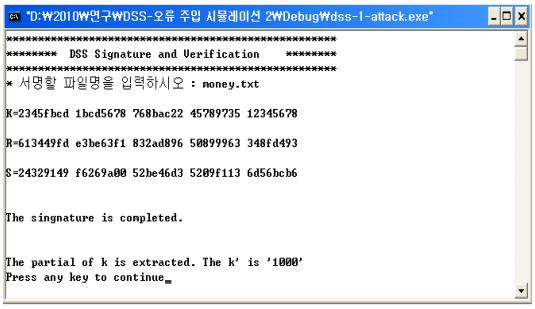
오류주입 공격에 대응하는 서명 방식이 정상적으로 동작하면 오류주입 공격에 강인함을 보인 화면이 그림 6이다. 그림은 위의 공격 모델과 동일하게 비밀키 x에 한 비트 오류가 발생했을 경우에도 공격자가 비밀키 비트를 추출할 수 없음을 보인 것이다.



[그림 6] 비밀키 공격에 대한 대응책 적용

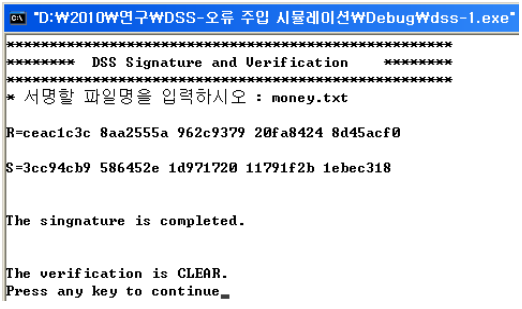
다음 그림 7은 제안하는 랜덤 수에 대한 오류주입 공

격을 시뮬레이션한 화면이다. 화면 하단에 보는 바와 같이 멱승 연산시 오류를 주입하여 랜덤 수 k의 하위 4비트($k' = 1000_2$)를 찾아낼 수 있음을 알 수 있다.



[그림 7] 랜덤 수 오류주입 공격 시뮬레이션

그러나 논문에서 제안하는 대응 알고리즘을 적용하고 랜덤 수에 오류를 주입하는 공격을 시도해 보면 그림 8과 같이 정상 서명이 생성되어 공격에 필요한 정보를 얻을 수 없음을 볼 수 있다.



[그림 8] 랜덤 수 공격에 대한 대응책 적용

6. 결론

지금까지 기술한 바와 같이 정보보호용 디바이스에 장착되어 사용되는 국제 서명 알고리즘 DSA는 물리적 영향으로 인한 오류주입 공격에 취약하다. 논문에서는 Bao 등이 제안한 비밀키 자체에 대한 한 비트 오류주입 공격의 위험성을 분석하였으며 랜덤 수에 대한 새로운 오류주입 방법을 제안하였다.

그리고 DSA서명 알고리즘에 대한 위의 공격을 모두 방어할 수 있는 대응 방안을 제안하였다. 제안된 대응책들은 오류확산 기법을 사용하여, 공격자로 하여금 서명 정보를 이용하여 공격에 이용할 수 없도록 설계하였다. 또한, 제안하는 서명 알고리즘은 간단한 계산 기법만 사용함으로써 기존의 서명 알고리즘과 비교하여 추가되는 연산량이 거의 없이 오류주입 공격을 방어할 수 있다.

참고문헌

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

[2] 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

[3] C. H. Kim and J. -J. Quisquater, "New Differential Fault Analysis on AES Key Schedule: Two Faults are enough", CARDIS-2008, LNCS 5189, pp. 48-60, 2008.

[4] S. Yen, S. Kim, S. Lim, and S. Moon, "RSA speedup with Chinese Remainder Theorem Immune Against Hardware Fault Cryptanalysis," IEEE Transaction on Computer, Special issue on CHES, vol. 52, no. 4, pp. 461-472, 2003.

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

[6] National institute of standards and technology. Digital Signature Standard, NIST FIPS PUB 186-2, 2000.

[7] M. Nikodem, "Error Prevention, Detection and Diffusion Algorithms for Cryptographic Hardware", International Conference on Dependability of Computer System (DepCos-RELCOMEX'07), pp. 127-134, IEEE-CS, 2007.

[8] M. Nikodem, "DSA Signature Scheme Immune to the Fault Cryptanalysis", CARDIS-2008, LNCS, vol. 5189. pp. 61-73, 2008

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

[10] 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.

[11] J. Schmidt, M. Medwed, "A Fault Attack on ECDSA", Fault Diagnosis and Tolerance in Cryptography,

FDTC-2007, pp. 93-99, 2009.
[12] N. Howgrave-Graham and N. P. Smart. "Lattice Attacks on Digital Signature Schemes", Designs, Codes and Cryptography, vol. 23, no. 3, pp. 283-290, 2001.
[13] T. Römer and J. P. Serfert, " Information Leakage Attack against Smart Card Implementation of the Elliptic Curve Digital Signature Algorithm," International Conference on Research in Smart Cards, E-smart-2001, LNCS vol. 2140, pp. 211-219, 2001.

정 철 조(Chul-Jo Jung) **[정회원]**

- 2009년 2월 : 호서대학교 정보보호학과 (공학사)
- 2009년 3월 ~ 현재 : 호서대학교 대학원 정보보호학과(석사과정)

<관심분야>
암호학, 보안성 평가, 인증

오 두 환(Doo-Hwan Oh) **[정회원]**

- 2010년 2월 : 호서대학교 정보보호학과 (공학사)
- 2010년 3월 ~ 현재 : 호서대학교 대학원 정보보호학과(석사과정)

<관심분야>
스마트 카드 보안, 네트워크 보안

최 두 식(Doo-Sik Choi) **[정회원]**

- 2010년 2월 : 호서대학교 정보보호학과 (공학사)
- 2010년 3월 ~ 현재 : 호서대학교 대학원 정보보호학과(석사과정)

<관심분야>
스마트 카드 보안, 네트워크 보안, 부채널 공격

김 환 구(Hwan-Koo Kim) **[정회원]**

- 1987년 2월 : 경북대학교 수학과 (이학사)
- 1991년 2월 : 경북대학교 수학과 (이학석사)
- 1998년 5월 : U. of Tennessee-Knoxville 수학과(이학박사)
- 2002년 3월 ~ 현재 : 호서대학교 정보보호학과 부교수

<관심분야>
평가 및 인증, 암호학

하 재 철(Jae-Cheol Ha) **[종신회원]**

- 1989년 2월 : 경북대학교 전자공학과 (공학사)
- 1993년 8월 : 경북대학교 전자공학과 (공학석사)
- 1998년 2월 : 경북대학교 전자공학과 (공학박사)
- 1998년 3월 ~ 2007년 2월 : 나사렛대학교 정보통신학과 부교수
- 2007년 3월 ~ 현재 : 호서대학교 정보보호학과 부교수

<관심분야>
정보보호, 네트워크 보안, 부채널 공격