AES에 대한 오류기반 공격

장화선*, 김광조*

*한국정보통신대학원대학교, 국제정보보호기술연구소

Fault Based Cryptanalysis of AES

Hwasun Chang*, Kwangjo Kim*

*International Research center for Information Security(IRIS)

Information and Communications Univ.(ICU)

요 약

스마트카드와 같은 장치의 부채널 공격이 가능한 것으로 입증되면서 많은 연구가 진행되고 있다. 부채널 공격의 일종인 오류기반 공격은 구현 가능성이 논란의 대상이었지만 카메라 플래쉬를 이용한 광학적 공격이 보고되면서 실현 가능성이 높은 것으로 인식되고 있다. 본 논문에서는 AES에 대해 향상된 오류기반 공격을 제안한다. 제안된 방법을 사용하면 공격에 필요한 암호문 수를 기존 방법보다 많이 줄일 수 있으며 최적의 조건에서는 하나의 암호문만으로 키를 얻을 수 있다. 오류는 일시적으로 생긴 후 없어지는 유형을 사용할 수도 있고 영구적인 것일 수도 있다. 오류에 의한 변경 확률이 1이 아닌 경우와 오류 발생 시점의 오차에 대해서도 살펴본다.

I. 서론

스마트카드의 이용이 증가하면서 이와 같은 장치에 대한 부채널 공격(Side channel attack)이 많이 연구되고 있다. 부채널 공격으로는 암호 알고리즘 계산 시의 소모 전류 변화를 이용하는 SPA(Simple Power Analysis)와 DPA(Differential Power Analysis)[1], 계산 시간의 변화를 이용하는 시간 공격(Timing Attack)[2], 오류를 만들어 이에 따른 결과를 이용하는 오류기반 공격(Fault Attack)[3], 전자기파를 이용하는 공격[4] 등이 소개되었다.

오류기반 공격은 1996년에 Boneh, DeMillo, Lipton에 의해 소개되었다. 소개될 당시에는 RSA 등 산술적 특성을 이용할 수 있는 알고리즘에 적용되었다. 얼마 후 Biham과 Shamir는 DES와 비밀키 알고리즘에 적용할 수 있는 Differential Fault Analysis를 발표하였다[5]. AES가 NIST에 의해 표준 비밀키 알고리즘으로 선정된 후 AES에 적용할 수 있는 몇 가지 오류기반 공격이 제시되었다[6,7,8].

본 논문에서는 최근에 FC03을 통해 소개되었던 BS03의 오류기반 공격을 개선하여 AES에서 오류기반 공격을 효율적으로 구현할 수 있는 방안을 제시한다. 개선된 방법을 사용하면 정상상태의 암호문과 비교할 필요가 없고 오류를 발생시키는 방법에 따라 필요한 암호문 수를 많이 줄일 수 있다. 최적의 상황에서는 하나의 암호문으로 AES 키를 알아낼 수도 있다. 제안된 방법은 현재 많이 사용되고 있는 128비트 키를 가지는 AES에 가장 잘 적용될 수 있다. 발생된 오류는 일시적인 것일 수도 있고 영구적인 것일 수도 있다. 또 한 비트만이 아닌 인접한 비트에 오류가 발생해도 된다. 오히려 많은 비트에 오류가 발생할수록 더 신속하게 키를 알아낼 수 있다.

본론에서는 1절에서 논문과 관련된 부분을 중심으로 AES 알고리즘을 살펴보고, 2절에서는 FC03에 소개된 오류기반 공격을 정리하고 3절에서는 새로 제안하는 공격 방법을 기술한다. 4절에서는 제안된 공격이 오류 발생 확률이 1이 아닌 경우, 5절에서는 발생 타이밍에 오차가 있는 경우 어떻게 적용될 수 있는지 살펴본다. 마지막으로 6절에서는 제안된 공격의 대응방안에 대해 논의한다.

II. 본론

1. AES

본 연구와 관련된 부분을 중심으로 AES를 살펴본다. AES에서는 상태(State)라 불리는 중간 값에 여러 가지 변환을 적용한다. 상태는 바이트의 직사각형 배열로 나타낼 수 있다. 암호 키도 마찬가지 형태로 나타낸다. 배열의 행수는 4이며 열수는 데이터의 경우 N_b, 키의 경우 N_k로 나타낸다. 라운드 수 N_r은 키 길이에 따라 표 1과 같이 결정된다.

표 1: 키 길이와 라운드 수의 관계

	키 길이 (N_k)	블록 크기 (N_b)	라운드 수 (N_r)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

AES는 첫 라운드 키 더하기, N_r-1번의 라운드, 최종 라운드로 구성된다. N_r-1번의 라운드 중 한 라운드와 최종 라운드를 의사(Pseudo) C 코드로 나타내면 다음과 같다.

```
Round (State, RoundKey)
{
  ByteSub (State) ;
  ShiftRow (State) ;
  MixColumn (State) ;
  AddRoundKey (State, RoundKey) ;
}

FinalRound (State, RoundKey)
{
  ByteSub (State) ;
  ShiftRow (State) ;
  AddRoundKey (State, RoundKey) ;
}
```

ByteSub 변환은 상태의 각 바이트에 독립적으로 비선형 바이트 치환을 행하는 것이다. ShiftRow 변환은 상태의 각 행을 정해진 수만큼 쉬프트 하는 것이다. MixColumn 변환은 상태의 각 열을 $GF(2^8)$에서의 다항식으로 간주하여 다음으로 주어지는 c(x)를 곱한 후 x^4+1에 대해 모듈러 연산을 한 결과이다.

$$c(x) = '03'x^3 + '01'x^2 + '01'x + '02'$$

AddRoundKey 변환에서는 Exclusive-OR를 사용하여 라운드 키를 상태에 적용한다.

각 라운드 키는 키 스케줄에 의해 암호화키로부터 만들어진다. 키 스케줄은 키 확장과 라운드 키 선택으로 이루어진다. 확장된 키는 4바이트 워드의 선형 배열로 생각할 수 있으며 $W[N_b*(N_r+1)]$로 나타낼 수 있다. N_k가 6이하인 경우의 키 확장은 다음과 같은 의사 C 코드로 나타낼 수 있다.

```
KeyExpansion(byte Key[4*Nk], word W[Nb*Nr+1])
{
  for(i=0; i<Nk; i++)
    W[i] = (Key[4*i], Key[4*i+1], Key[4*i+2], Key[4*i+3]);
  for(i=Nk; i<Nb*(Nr+1); i++)
  {
    temp = W[i-1];
    if(i%Nk == 0)
      temp = SubByte(RotByte(temp)) ^ Rcon[i/Nk];
    W[i] = W[i-Nk]^temp;
  }
}
```

2. BS03의 방법

먼저 $N_k = N_b = 4$인 경우를 생각한다. 이러한 경우에는 전체 암호 키가 첫 AddRoundKey에서 사용된다. 암호 키가 4 x 4 배열에 저장되어 있다고 보고 각 바이트를 k_{ij}라 한다. k_{ij}의 l번째 비트를 k_{ijl} ($0 \le l \le 7$)로 나타낸다. 공격자는 모든 비트가 0인 평문을 암호화한다. 상태의 각 바이트를 a_{ij}라 하면, 첫 AddRoundKey에서는

$$a_{ij} = 0^8 \wedge k_{ij}$$

를 계산한다. 계산 결과는 $a_{ij} = k_{ij}$이다. a_{ij}의 l번째 비트를 a_{ijl} ($0 \le l \le 7$)로 나타내면 다음 변환이 적용되기 전에 공격자는 a_{ijl}을 0으로 한다. 만약 k_{ijl}이 0이었다면 오류 공격을 거친 암호문은 정상 암호문과 같을 것이다. 하지만 만약 k_{ijl}이 1이었다면 오류 공격에 의한 암호문과 정상 암호문은 다른 값을 가질 것이다. 이와 같은 과정을 전체 키 비트에 대해 반복하여 암호 키를 알아낼 수 있다.

$N_k > N_b$인 경우에는, 먼저 앞의 과정을 통해 첫 AddRoundKey에서 사용되는 라운드 키를 계산한다. 이렇게 얻어진 값을 이용하여 첫 라운드에서 AddRoundKey를 적용하기 직전 상태의 비트 값이 모두 0이 되기 위한 평문 값을 계산할 수 있다. 계산된 평문을 이용하여 암호화를 수행하고

앞에서와 유사하게 첫 라운드의 AddRoundKey를 수행한 후 알아내려고 하는 키 비트에 해당하는 값을 0으로 만든다. 키의 해당 비트가 0이었으면 오류 공격에 의한 암호문과 정상 암호문은 같을 것이다. 1이었다면 다를 것이다. 이러한 과정을 반복하여 전체 암호 키를 알아낼 수 있다.

3. 향상된 공격 방법

BS03에서 제안한 방법에서는 키 값 중 두 비트 이상을 동시에 알아낼 수는 없다. 오류는 정확히 한 비트에만 만들어져야 한다. 또한 오류 공격에 의한 암호문과 정상 암호문을 비교해야 하기 때문에 정상 암호문과 오류에 의한 암호문을 얻어야 한다. 만들어진 오류는 다음 비트에 대한 공격 전까지 복구되어야 하는 문제도 있다.

향상된 오류 공격으로 이와 같은 문제를 없앨 수 있다. $N_k = N_b = 4$인 경우를 생각하면 다음과 같다. 최종 라운드에서 라운드 키가 4 x 4 배열에 저장되어 있다고 보고 각 바이트를 rk_{ij}라 한다. rk_{ij}의 l번째 비트를 rk_{ijl} $(0 \le l \le 7)$로 나타낸다. 향상된 공격에서는 평문을 임의로 선택할 수 있는 장점도 있다. 임의의 평문을 암호화하다가 마지막 라운드의 AddRoundKey를 수행하기 직전에 a_{ijl}을 0으로 한다. 암호문 b_{ijl}은 다음과 같이 된다.

$$b_{ijl} = a_{ijl} \hat{} rk_{ijl} = 0 \hat{} rk_{ijl} = rk_{ijl}$$

이로부터 마지막 라운드 키의 해당 비트를 알 수 있다. 이러한 과정을 모든 상태 비트에 대해 반복하면 최종 라운드 키를 알 수 있다. 최종 라운드 키로부터 암호화키를 계산할 수 있음은 알려져 있다[7].

오류를 2비트 이상에서 만들 수 있는 경우에는 더 신속하게 키를 찾아낼 수 있다. 발생된 오류에 해당하는 최종 라운드 키를 알 수 있기 때문이다. 모든 비트에 오류를 만들 수 있는 경우에는 한 번에 키를 알아낼 수도 있다.

$N_k > N_b$인 경우에는 최종 라운드 키와 바로 앞의 라운드 키를 알아내야 암호화키를 알아낼 수 있다. 이러한 경우에는 먼저 앞에서와 같은 방법으로 최종 라운드 키를 알아낸다. 단, 이러한 경우에는 최종 라운드 키를 알아내기 위해 만든 오류가 일시적인 것이어야 한다. 다음으로 직전 라운드의 AddRoundKey 전에 알고자 하는 라운드 키의 비트에 해당하는 a_{ijl}을 0으로 한다. 이 연산 이후에 해당 비트는 라운드 키의 해당 비트 값 rk_{ijl}의 값을 가지게 된다. rk_{ijl} 값은 오류에 의한 암호문과 전에 구했던 최종 라운드 키를 이용하여 역으로 계산하면 알아낼 수 있다.

4. 확률적 오류 모델

오류에 의해 특정 비트 값이 0으로 바뀔 확률이 1이 아닌 경우에 대해서는 다음과 같이 할 수 있다.

암호화 중간 값이 원래 1의 값을 갖는 경우에 대해 1이 0으로 바뀔 확률을 p_1이라 부르고 0.5보다 큰 값을 갖는다고 가정한다. 중간 값의 원래 값이 0일 때 1로 바뀔 확률을 p_0라하고 0.5보다 작은 값을 갖는다고 가정한다. 또 오류에 의해 중간 값이 0일 확률을 p_{r0}라하고 원래 중간 값이 0일 확률을 p_{i0}, 원래 중간 값이 1일 확률을 p_{i1}이라 하자. 임의의 평문을 암호화하는 경우에는 $p_{i0} = p_{i1} = 0.5$로 할 수 있을 것이다. p_{r0}는 다음과 같이 계산할 수 있다.

$$\begin{aligned} p_{r0} &= Pr[\text{원래 값이 0이고 오류에 의한 값이 0}] \\ &\quad + Pr[\text{원래 값이 1이고 오류에 의한 값이 0}] \\ &= p_{i0} * (1 - p_0) + p_{i1} * p_1 \\ &= 0.5 * (1 - p_0) + 0.5 * p_1 \\ &= 0.5 * (1 - p_0 + p_1) \; > 0.5 \end{aligned}$$

계산 결과로부터 평문을 임의로 선택하는 경우 p_0와 p_1이 조건을 만족하면 오류에 의한 중간 값은 0.5이상의 확률로 0이 됨을 알 수 있다.

$N_k = 4$인 경우를 생각하면, 최종 라운드 키 비트 값은 다음과 같이 얻을 수 있을 것이다. 임의의 평문을 발생시켜 암호화를 하면서 AddRoundKey 이전에 오류를 발생시킨다. 이로부터 해당 최종 라운드 키 비트 값 후보를 얻는다. 이와 같은 과정을 여러 번 반복한다. 여러 개의 라운드 키 비트 값 후보 중 0과 1의 빈도를 비교하여 더 많이 얻어진 값을 선택한다.

5. 타이밍에서의 오차

오류가 정확히 AddRoundKey 이전에 발생하지 않는 경우에 대해서는 다음과 같이 할 수 있다.

최종 라운드에서의 AddRoundKey 이후에 오류가 발생하면 암호문의 해당 비트는 라운드 키에 관계없이 0이 될 것이다. 오류에 의해 해당 비트가 0으로 될 확률이 1이라면 오류가 있는 암호화를 여러 번 시행하여 0이 아닌 값이 나오는 경우가 있는 지를 살펴보고 있다면 이 값을 올바른 값

으로 사용하여 이러한 타이밍 오류 문제를 해결한다.

ShiftRow 이전에 오류가 발생했다면 오류의 위치가 바뀌게 된다. 오류의 위치가 맞는 것만 사용하여 ShiftRow 이전에 발생한 오류를 버릴 수 있다.

6. 대응 방안

본 논문에서 제안한 오류 공격은 SA03에서 제안된 카메라 플래시에 의한 오류 공격을 가정하고 있다. 제안된 공격에서는 원하는 값을 비트 단위로 조작할 수 있었다. SA03에서는 이러한 공격의 가능성을 발표하면서 대응 방안으로 Self-timed dual-rail logic을 제안하였고 제안한 공격의 대응 방안 중 하나가 될 수 있을 것이다.

III. 결론

본 논문에서는 FC03에 발표된 오류기반 공격을 개선한 오류기반 공격을 제안하였다. 두 가지 방법을 비교하면 표 2와 같다.

표 2: AES-128의 경우 기존 방법과의 비교

	기존 방법	제안한 방법
정상 암호문과의 비교	필요	필요 없음
최적 조건에서의 필요 암호문 수	129	1
오류 범위	한 비트	한 비트 이상도 됨
평문	모든 비트 0	모든 값
오류 유형	일시적 오류	일시적 및 영구적 오류

참고문헌

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis", Advances in Cryptology - Crypto '99, LNCS 1666, pp. 388-397, 1999

[2] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", Advances in Cryptology - Crypto '96, LNCS 1109, pp. 104-113, 1996

[3] D. Boneh, R. DeMillo, and R. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults", Advances in Cryptology - Eurocrypt '97, LNCS 1233, pp. 37-51, 1997

[4] J. Quisquater and D. Samyde, "ElectroMagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards", E-smart 2001, LNCS 2140, pp. 200-210, 2001

[5] E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems", Advances in Cryptology - Crypto '97, LNCS 1294, pp. 513-525, 1997

[6] J. Blömer and J. Seifert, "Fault based cryptanalysis of the Advanced Encryption Standard (AES)", Financial Cryptography 03, 2003

[7] P. Dusart, G. Letourneux, and O. Vivolo, "Differential Fault Analysis on A.E.S", http://eprint.iacr.org/2003/010.pdf

[8] C. Giraud, "DFA on AES", http://eprint.iacr.org/2003/008.ps

[9] S. Skorobogatov and R. Anderson, "Optical Fault Induction Attacks", CHES 2002, LNCS 2523, pp. 2-12, 2003

[10] K. Itoh, M. Takenaka, and N. Torii, "DPA Countermeasure Based on the Masking Method", ICICS 2001, LNCS 2288, pp. 440-456, 2002

[11] NIST, "Federal Information Processing Standards Publication 197 - Specification for the Advanced Encryption Standard (AES)", http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001