Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.2, Apr. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.2.327

랜덤하게 변형된 AES 키 비트열에 대한 키 복구 알고리즘

백 유 진[†‡]
우석대학교

Key Recovery Algorithm for Randomly-Decayed AES Key Bits

Yoo-Jin Baek[†‡]
Woosuk University

요 약

일반적으로 알려진 믿음과는 달리 다양한 컴퓨팅 장치의 메인 메모리로 사용되는 DRAM은 전원이 차단되더라도 저장하고 있던 데이터가 곧바로 사라지지 않고, 대신 어느 정도의 시간 동안 데이터를 유지하게 된다. 특히 DRAM을 냉각시키면 그 데이터 유지 시간이 더 길어진다는 사실 역시 알려져 있다. Cold Boot Attack이란 이러한 DRAM의 데이터 유지 성질을 이용하여, 전원이 차단된 DRAM으로부터 암호 알고리즘의 키와 같은 민감한 정보를 복구해내는 부채널 공격 방법의 일종이다. 본 논문에서는 대칭붕괴모델을 가정한 Cold Boot Attack 방법을 이용하여 전원이 차단된 DRAM으로부터 추출된 AES 키 비트열로부터 원래의 AES 키를 복구하는 알고리즘을 제안한다. 제안된 알고리즘은 추출된 AES 키 비트열의 랜덤성을 테스트하는 방법을 사용하여 후보 키 공간의 크기를 줄이는 방법을 사용한다.

ABSTRACT

Contrary to the common belief, DRAM which is used for the main memory of various computing devices retains its content even though it is powered-off. Especially, the data-retaining time can increase if DRAM is cooled down. The Cold Boot Attack, a kind of side-channel attacks, tries to recover the sensitive information such as the cryptographic key from the powered-off DRAM. This paper proposes a new algorithm which recovers the AES key under the symmetric-decay cold-boot-attack model. In particular, the proposed algorithm uses the strategy of reducing the size of the candidate key space by testing the randomness of the extracted AES key bit stream.

Keywords: Cold Boot Attack, Side-Channel Attack, NIST Randomness Test, AES

I. 서 론

다양한 컴퓨팅 장치의 메인 메모리로 사용되는 DRAM은 일반적인 믿음과 달리 전원이 차단되더라도 그 내용이 곧바로 사라지지 않으며, 특히 DRAM을 냉각하게 되면 그 데이터 유지 시간은 더 늘어난다는 사실이 알려져 있다. Cold Boot Attack이란, 전원이 차단된 후에도 DRAM이 데이터를 유지하는 이러한 성질을 이용하여, 전원이 차단된 DRAM으로부터 암호 알고리즘의 비밀키 등과 같은 민감한 정보를 복구해내는 부채널 공격 방법의 일종이다[1].

Cold Boot Attack이 성공적으로 수행되려면 먼저 전원이 차단된 DRAM으로부터 민감한 정보를 포함하고 있는 비트열을 추출해내야 한다. 이때 추출된 비트열은 일반적으로 원래 저장되어 있던 정보와

Received(01. 06. 2016), Modified(04. 14. 2016), Accepted(04. 14. 2016)
† 주저자, yoojin.baek@gmail.com
‡ 교신저자, yoojin.baek@gmail.com(Corresponding author)

정확히 일치하지 않을 수 있는데, 왜냐하면 어떤 비트들은 전원이 차단된 후 다른 비트로 바뀔 수 있기 때문이다. 그리고 이렇게 추출된 비트가 반대 비트, 즉 비트 0은 비트 1로 그리고 비트 1은 비트 0으로 바뀌는 것을 비트가 붕괴(decayed)되었다고 표현한다.

Cold Boot Attack은 원래 데이터 비트가 0이었지만 전원이 차단된 후 비트 1로 붕괴되는 확률 δ_1과 반대로 원래 비트가 1이었지만 전원 차단후 비트 0으로 붕괴되는 확률 δ_0에 따라 다양한 방식으로 분류될 수 있다. 먼저 비대칭붕괴모델(asymmetric decay model)은 $\delta_0 \neq \delta_1$이라고 가정을 하고 분석을 진행하며, δ_0와 δ_1중 하나는 0이라고 가정하는 완전비대칭붕괴모델과 δ_0, δ_1 모두 0이 아니라고 가정하는 불완전비대칭붕괴모델로 분류된다. 반면에 대칭붕괴모델 (symmetric decay model)은 $\delta_0 = \delta_1 > 0$이라고 가정한다. 본 논문에서는 이 중에서 대칭붕괴모델을 가정하고 이로부터 DRAM에 저장되어 있던 비트열로부터 AES 키를 복구하는 알고리즘을 제안한다.

1.1 이전 연구 결과

전원이 차단된 DRAM으로부터 추출한 키 비트열로부터 암호 알고리즘의 키를 복구하는 많은 방법이 제안되었으며 이 중에서 Halderman 등은 128-비트 키를 사용하는 AES에 대하여 비트 1이 비트 0으로 붕괴될 수 있지만 반대로 비트 0이 비트 1로 붕괴되는 것은 거의 불가능하다는 가정, 즉 $\delta_0 \gg \delta_1 \approx 0$인 불완전비대칭붕괴모델 하에서 AES 키를 복구하는 알고리즘을 제시하였고, 실제로 $\delta_0 = 0.15$인 경우에는 1초 안에, $\delta_0 = 0.3$인 경우에는 몇 분 안에 AES 키를 복구할 수 있음을 보였다[1]. 또한 Tsow는 $\delta_1 = 0, \delta_0 = 0.7$인 완전비대칭붕괴모델을 가정하고 AES 키를 복구하는 알고리즘을 제안하였다[6]. 그리고 Kamal과 Youseff는 완전비대칭붕괴보델에서 AES 키 복구 문제는 Boolean Satisfiability 문제로 변환될 수 있음을 보였으며[7], Albert 등은 키 복구 문제를 노이즈가 있는 연립 다항식 문제를 해결하는 문제로 변환하는 방법을 제안하였다[8]. 이러한 예에서 보듯이 기존의 AES에 대한 Cold Boot Attack은 주로 비대칭붕괴모델을 가정하고 분석을 진행하였으나, 본 논문에서는 이러한 이전 결과와는 다르게 대칭붕괴모델을 가정하고 AES 키를 복구하는 알고리즘을 제시한다.

일반적으로 Cold Boot Attack에서 대칭붕괴모델을 가정하면 비대칭붕괴모델에 비해 몇 가지 어려운 점에 직면하게 되는데, 그 중의 하나는 바로 공격을 진행함에 따라 예측된 키 후보의 유효성을 검증하는 방법이 없다는 것이다. 예를 들어 Heninger와 Shacham은 완전비대칭붕괴모델 하에서 RSA 시스템의 개인키의 일부 비트로부터 개인키 전체를 복구하는 알고리즘을 제안하였고 실제로 RSA 개인키 (p, q, d, d_p, d_q)의 27%가 주어진 경우 개인키 전체를 복구할 수 있음을 보였다[10]. 반면에 Henecka 등은 대칭붕괴모델 하에서 (p, q, d, d_p, d_q)의 비트 중 23.7%가 다른 비트로 붕괴된 경우에 RSA 개인키 전체를 복구하는 알고리즘을 제시하였다[11]. 따라서 이러한 이전 결과가 보이듯이 일반적으로 대칭붕괴모델은 비대칭붕괴모델에 비해 성공 확률과 적용 가능한 비트 에러율에서 많은 불리한 점이 존재하게 된다.

II. 사전 지식

2.1 부채널 공격

일반적으로 암호알고리즘에 대한 공격법은 크게 수학적인 공격과 부채널 공격으로 구분이 된다.

먼저 수학적인 공격법은 암호 알고리즘을 비밀키를 인자로 가지는 수학적 함수로 취급을 하고 수학적인 관점에서 그 안전성을 평가한다. 이러한 공격법의 대표적인 예로는 차분 공격법, 선형 공격법, 대수적인 공격법 등이 있다[9].

반면에 부채널 공격법은 암호 알고리즘을 물리적인 기기에서 동작하는 실제 프로그램으로 간주하며 해당 기기의 암호 연산 과정 중에 발생하는 부채널 정보를 이용하여 기기 내부에 저장된 비밀 정보를 알아내는 공격방법이다. 이러한 부채널 정보의 예로는 연산 시간, 전력 소비, 전자기장 및 의도된 혹은 의도되지 않은 오류 주입으로부터 생성된 잘못된 암호문 등이 있다. 부채널 공격법은 Kocher 등에 의해 스마트카드에 대한 시간분석 공격법이 제안된 이후로 현재까지 여러 가지 다양한 공격법과 그에 대한 대응

기법이 제시되었다[2,3,4]. 본 논문에서 제시하는 Cold Boot Attack 역시 전원이 차단된 DRAM으로부터 추출된 불완전한 암호 키 비트열로부터 전체 키를 복구하는 기술이기 때문에 부채널 공격법의 일종으로 볼 수 있다.

2.2 Cold Boot Attack

Cold Boot Attack은 다양한 컴퓨팅 장치의 메인 메모리에 사용되는 DRAM은 전원이 차단되더라고 어느 정도의 시간 동안은 그 저장 데이터를 유지한다는 성질을 이용한다. 전원이 차단된 후 메모리의 내용이 유지되는 시간은 특히 해당 메모리를 아주 차갑게 냉각시켰을 때 길어질 수 있는데, 예를 들어 메모리를 질소 용액에 의해 급속 냉각을 시키면 그 유지 시간이 수 시간 정도로 길어질 수도 있음이 실험으로 밝혀졌다[1].

Cold Boot Attack을 수행하기 위해서는 먼저 작동 중인 컴퓨팅 장치의 전원을 차단한 후 해당 장치의 메모리 모듈을 장치에서 분리한다. 이 후 메모리의 내용은 특정 방법을 이용해서 덤프가 되는데 이런 식으로 덤프된 메모리 내용에는 특히 컴퓨팅 장치에서 민감한 정보를 보호하기 위해 사용되는 암호 알고리즘의 키 정보가 포함될 수 있다. 따라서 이후의 분석 과정을 통해 이렇게 추출된 메모리 정보로부터 암호 알고리즘의 키 정보와 같은 민감 정보를 복구하게 된다.

본 논문에서는 특히 AES 블록 암호알고리즘이 사용된 경우 Cold Boot Attack을 통해 AES 키를 복구하는 방법을 제시한다.

2.3 NIST 랜덤성 테스트

랜덤성 테스트는 어떤 데이터 집합의 랜덤성을 판단하기 위해 사용되는 통계 테스트의 일종이다. 일반적으로 임의의 이진열이 주어진 경우 그 랜덤성을 판단하는 많은 테스트 방법들이 제안되었으며, 특히 주어진 이진열의 통계적인 특성을 분석하여 그 랜덤성을 테스트하는 방법이 많이 사용된다.

이러한 테스트 방법 중 NIST에서 제안한 랜덤성 테스트 기법은 임의의 길이를 가지는 이진열의 랜덤성을 테스트하기 위한 15가지의 방법을 제공하며[5], 이 방법들 중에서 Frequency (Monobit) Test와 Frequency Test within a Block를 제외한 나머지 방법들은 주로 길이가 매우 긴 이진열에 대한 랜덤성을 테스트하기 위한 용도로 사용된다. 그러나 본 논문에서 다루게 되는 AES 키 비트열은 기껏해야 수백 비트 정도의 크기를 가지기 때문에 이러한 NIST 랜덤성 테스트 방법 모두를 적용하는 것은 큰 어려움이 있다. 따라서 본 논문에서 제안하는 방법은 Frequency (Monobit) Test와 Frequency Test within a Block를 주로 사용하여 AES 후보 키 비트열에 대한 랜덤성을 판단하고 이를 이용하여 실제 AES 키를 복구하게 된다.

먼저 Frequency (Monobit) Test는 주어진 이진열의 비트 0과 비트 1의 빈도수를 계산하며, 랜덤한 이진열의 경우 비트 0과 비트 1의 빈도수가 거의 동일할 것이라는 사실을 기반으로 한다. 구체적으로, 먼저 주어진 이진열의 비트 1은 그대로 두고 비트 0은 -1로 변환한 후 이진열의 전체 합 S를 구한다. 이 후 S의 절대값을 이진열의 길이 n으로 나눈 값 $|S|/n$을 구한 후, 함수

$$erfc(z) = \frac{2}{\sqrt{\pi}} \int_{z}^{\infty} e^{-u^2} du$$

에 대한 함수값 $erfc(\frac{|S|}{n})$을 계산한다. 마지막으로 이 함수값이 주어진 p-value보다 작으면 랜덤하지 않다고, 그리고 p-value보다 크면 랜덤하다고 판단하게 된다. 그러나 본 논문에서 테스트하고자 하는 이진열에서는 특정한 $\delta(0 < \delta < 1)$의 확률로 비트 1이 나타나고 $1-\delta$의 확률로 비트 0이 나타나기 때문에 Frequency (Monobit) Test를 그대로 적용할 수는 없다. 따라서 이를 해결하기 위하여 본 논문에서는 비트 1은 $\sqrt{\frac{1-\delta}{\delta}}$로 비트 0은 $-\sqrt{\frac{\delta}{1-\delta}}$로 변환한 후 그 합 S'를 구하고 나서 $erfc(\frac{|S'|}{n})$값을 이용해 주어진 이진열의 랜덤성을 판단하는 방법을 사용한다. 즉 $erfc(\frac{|S'|}{n})$이 주어진 p-value보다 작으면 랜덤하지 않다고, 그리고 p-value보다 크면 랜덤하다고 판단하게 되는 것이다.

Frequency Test within a Block은 주어진 이진열에서 임의의 길이의 연속된 부분이진열을 선택한 후 이 부분이진열에서의 비트 1의 빈도수를 계산

해서 랜덤성을 판단한다. 즉 이 테스트는, 만약 주어진 이진열이 랜덤하고 부분 이진열의 길이가 M이라고 하면 이 부분 이진열에서 비트 1의 개수는 약 $M/2$일 것이라는 예상에 기반한다. 참고로 만약 $M=1$이면 이 테스트는 Frequency (Monobit) Test와 동일하다.

Frequency Test within a Block을 수행가기 위해서는 먼저 길이가 n인 이진열을 길이가 M인 $N=\lfloor n/M \rfloor$개의 연속된 부분이진열로 분해한 후 각 부분 이진열의 비트 1의 개수 $\pi_i, i=1,...,N$을 계산한다. 그리고

$$\chi^2 = 4M\sum_{i=1}^{N}(\pi_i - 1/2)^2$$

와

$$p = igamc(N/2, \chi^2/2)$$

을 계산한 후 p값이 주어진 p-value보다 크면 해당 이진열이 랜덤하다고 판단한다. 여기서 $igamc$는 [5]에 주어진 incomplete gamma function를 의미한다. 그러나 본 논문에서 테스트하고자 하는 이진열은 $\delta(0<\delta<1)$의 확률로 비트 1이, $1-\delta$의 확률로 비트 0이 나오기 때문에 Frequency Test within a Block을 그대로 사용하는 대신,

$$\chi'^2 = 4M\sum_{i=1}^{N}(\pi_i - \delta)^2$$

와

$$p' = igamc(N/2, \chi'^2/2)$$

을 계산한다. 그리고 p'값이 주어진 p-value보다 크면 주어진 이진열이 랜덤하다고 판단한다.

III. AES 키 복구 알고리즘

AES(Advanced Encryption Standard)는 SPN(Substitution-Permutation Network) 구조를 가지는 블록 암호알고리즘으로 128-비트 크기의 데이터 블록과 128-, 192- 또는 256-비트 크기의 키를 이용해서 암·복호화 연산을 수행한다. AES의 라운드 함수는 SubBytes, ShiftRows, MixColumns, AddRoundKey라는 4개의 함수로 구성되며 AES는 이러한 라운드 함수를 10, 12 또는 14번 반복 적용하여 암·복호화 연산을 수행한다.

본 논문에서 제안하는 AES 키 복구 Cold Boot Attack은 전원이 차단된 DRAM으로부터 추출된 (에러가 삽입된) AES 키 비트열로부터 실제 AES 키 값을 복구하는 것을 목표로 하기 때문에 먼저 AES 키 생성 알고리즘에 대한 이해가 필요하다. 특히 본 논문에서는 128-비트 크기의 키를 사용하는 AES 알고리즘(이하 AES128)의 키를 복구하는 방법을 제시하기 때문에 이를 위해서는 AES128의 키 생성 알고리즘을 자세히 살펴볼 필요가 있다.

AES128의 키생성 알고리즘은 32-비트 데이터에 대한 워드 연산을 바탕으로, 주어진 128-비트 키를 44개의 워드데이터로 변환하며, 이러한 변환을 좀 더 명확히 설명하기 위해서 본 논문은 다음과 같은 표기법을 사용한다[6]:

- $S_r (0 \le r \le 10)$: 4개의 워드로 구성된 r라운드 부분키
- $S_{r,k} (0 \le r \le 10, 0 \le k \le 3)$: S_r의 k번째 워드
- $S_{r,k,b} (0 \le r \le 10, 0 \le k,b \le 3)$: $S_{r,k}$의 b번째 바이트
- $S_i^w (0 \le i \le 43)$: 생성된 키 스케쥴의 i번째 워드

따라서, 예를 들어 S_r은 $S_{4r}^w, S_{4r+1}^w, S_{4r+2}^w, S_{4r+3}^w$라는 4개의 워드로 구성됨을 알 수 있다.

이러한 표기법을 사용하면 AES128의 키 생성 알고리즘은 다음과 같이 나타낼 수 있는데 여기서 ||는 연접(concatenation) 연산자를, SubWord는 주어진 워드 데이터의 각 바이트에 AES S-box를 적용하는 연산을, RotWord는 주어진 워드 데이터를 1바이트만큼 왼쪽으로 회전하는 함수를, Rcon은 라운드 상수값을 의미한다.

AES128 키 생성 알고리즘

Input: 16개의 바이트 $k[i]$, $i=0,...,15$

Output: 44개의 워드 $S_i^w, i=0,...,43$

1. $i = 0$;
2. while ($i < 4$)
   $S_i^w = k[4*i] \| k[4*i+1] \| k[4*i+2] \| k[4*i+3]$;
   $i = i+1$;
3. $i = 4$;
4. while ($i < 44$)
   $tmp = S_{i-1}^w$;
   if ($i \bmod 4 = 0$)
   $tmp = \mathrm{Sub}(\mathrm{Rot}(tmp)) \oplus \mathrm{Rcon}[i/4-1]$;
   $S_i^w = S_{i-4}^w \oplus tmp$;
   $i = i+1$;

이러한 AES128 키 생성 알고리즘은 다음과 같은 특징이 있음을 알 수 있으며 이러한 성질은 이 후 본 논문에서 제안하는 키 복구 알고리즘에서 매우 중요한 역할을 하게 된다.

1) 만약 $S_{0,0,0}, S_{1,0,0}$가 주어지면 이로부터 $S_{0,3,1}$을 계산할 수 있다. 마찬가지로 $S_{1,0,0}, S_{2,0,0}$가 주어지면 $S_{1,3,1}$을, $S_{2,0,0}, S_{3,0,0}$가 주어지면 $S_{2,3,1}$을 계산할 수 있다.

2) 만약 $S_{0,3,1}, S_{1,3,1}$이 주어지면 이로부터 $S_{1,2,1}$을 계산할 수 있다. 마찬가지로 $S_{1,3,1}, S_{2,3,1}$이 주어지면 이로부터 $S_{2,2,1}$을 계산할 수 있다.

3) 만약 $S_{1,2,1}, S_{2,2,1}$이 주어지면 이로부터 $S_{2,1,1}$을 계산할 수 있다.

4) 따라서 만약 $S_{0,0,0}, S_{1,0,0}, S_{2,0,0}, S_{3,0,0}$가 주어지면 이로부터 $S_{0,3,1}, S_{1,3,1}, S_{2,3,1}, S_{1,2,1}, S_{2,2,1}, S_{2,1,1}$을 계산할 수 있다.

5) 마찬가지로 만약 $S_{0,0,1}, S_{1,0,1}, S_{2,0,1}, S_{3,0,1}$가 주어지면 이로부터 $S_{0,3,2}, S_{1,3,2}, S_{2,3,2}, S_{1,2,2}, S_{2,2,2}, S_{2,1,2}$를, 만약 $S_{0,0,2}, S_{1,0,2}, S_{2,0,2}, S_{3,0,2}$가 주어지면 이로부터 $S_{0,3,3}, S_{1,3,3}, S_{2,3,3}, S_{1,2,3}, S_{2,2,3}, S_{2,1,3}$을, 만약 $S_{0,0,3}, S_{1,0,3}, S_{2,0,3}, S_{3,0,3}$가 주어지면 이로부터 $S_{0,3,0}, S_{1,3,0}, S_{2,3,0}, S_{1,2,0}, S_{2,2,0}, S_{2,1,0}$을 계산할 수 있다.

이해를 돕기 위하여 위의 특징 중 1)에서 4)까지에서 제시된 바이트의 계산 순서는 Fig. 1에 좀 더 자세히 묘사된다.

이제 위의 특성을 이용하여 AES 키를 복구하는 방법의 원리는 다음과 같다. 먼저 만약 전원이 차단된 DRAM으로부터 $\tilde{S}_{0,0,0}, \tilde{S}_{1,0,0}, \tilde{S}_{2,0,0}, \tilde{S}_{3,0,0}$ 및 $\tilde{S}_{0,3,1}, \tilde{S}_{1,3,1}, \tilde{S}_{2,3,1}, \tilde{S}_{1,2,1}, \tilde{S}_{2,2,1}, \tilde{S}_{2,1,1}$를 추출하였고 또한 이 데이터는 원 데이터인 $S_{0,0,0}, S_{1,0,0}, S_{2,0,0}, S_{3,0,0}$ 및 $S_{0,3,1}, S_{1,3,1}, S_{2,3,1}, S_{1,2,1}, S_{2,2,1}, S_{2,1,1}$의 각 비트가 δ의 확률로 0이 1로, 또는 1이 0으로 랜덤하게 변형이 되었다고 가정을 하자. 그러면 원데이터와 에러가 추가된 상태로 추출된 데이터에 XOR 연산을 적용하게 되면 전체 80-비트 데이터 중에서 비트 1은 δ의 확률로, 비트 0은 $1-\delta$의 확률로 나타나게 된다. 따라서 이 80-비트의 XOR된 데이터에 앞에서 제시한 랜덤성 테스트를 적용하여 그 데이터의 랜덤성을 테스트하게 되면, 만약 가정된 원데이터가 원래 키와 동일하면 XOR된 데이터가 랜덤성 테스트를 통과하게 될 것으로 예상이 된다. 즉 본 논문에서는 다음의 방법을 적용하여 추출된 AES 키 비트열로부터 원 AES 키 비트열을 복구하게 된다.

Algorithm 1. (AES128 키의 일부를 복구)

1) 전원이 차단된 DRAM으로부터 에러가 삽입된 AES 키 비트열 중 $\tilde{S}_{0,0,0}, \tilde{S}_{1,0,0}, \tilde{S}_{2,0,0}, \tilde{S}_{3,0,0}$ 및 $\tilde{S}_{0,3,1}, \tilde{S}_{1,3,1}, \tilde{S}_{2,3,1}, \tilde{S}_{1,2,1}, \tilde{S}_{2,2,1}, \tilde{S}_{2,1,1}$를 추출한다.

2) $S_{0,0,0}, S_{1,0,0}, S_{2,0,0}, S_{3,0,0,0} \in \{0, ..., 255\}$의 모든 경우에 대하여 다음을 수행한다.

   i) AES128 키 생성 알고리즘을 이용하여 $S_{0,0,0}, S_{1,0,0}, S_{2,0,0}, S_{3,0,0,0}$로 부터 $S_{0,3,1}, S_{1,3,1}, S_{2,3,1}, S_{1,2,1}, S_{2,2,1}, S_{2,1,1}$을 계산한다.

   ii) $D_{0,0,0} = \tilde{S}_{0,0,0} \oplus S_{0,0,0}$,

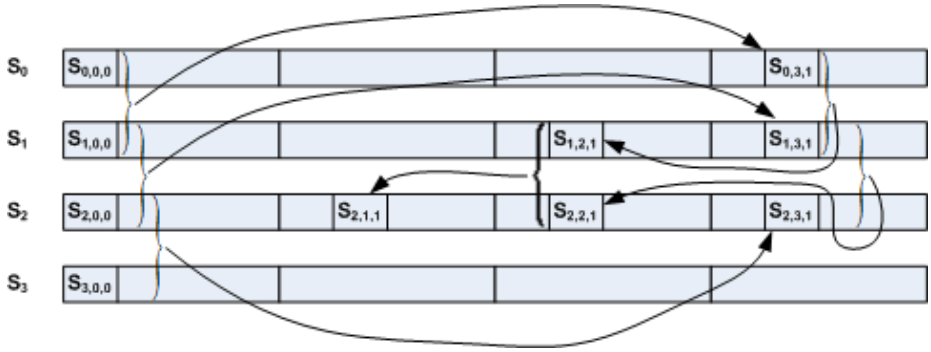

Fig. 1. Computation Flow of AES SubKey Bytes

$D_{1,0,0} = \tilde{S}_{1,0,0} \oplus S_{1,0,0}$,
$D_{2,0,0} = \tilde{S}_{2,0,0} \oplus S_{2,0,0}$,
$D_{3,0,0} = \tilde{S}_{3,0,0} \oplus S_{3,0,0}$,
$D_{0,3,1} = \tilde{S}_{0,3,1} \oplus S_{0,3,1}$,
$D_{1,3,1} = \tilde{S}_{1,3,1} \oplus S_{1,3,1}$,
$D_{2,3,1} = \tilde{S}_{2,3,1} \oplus S_{2,3,1}$,
$D_{1,2,1} = \tilde{S}_{1,2,1} \oplus S_{1,2,1}$,
$D_{2,2,1} = \tilde{S}_{2,2,1} \oplus S_{2,2,1}$,
$D_{2,1,1} = \tilde{S}_{2,1,1} \oplus S_{2,1,1}$을 계산한다.

iii) $D_{0,0,0}, D_{1,0,0}, D_{2,0,0}, D_{3,0,0,0}$ 및 $D_{0,3,1}, D_{1,3,1}, D_{2,3,1}, D_{1,2,1}, D_{2,2,1}, D_{2,1,1}$에 대하여 NIST의 (변형된) Frequency (Monobit) Test 및 Frequency Test within a Block을 이용하여 랜덤성 테스트를 수행한다. 만약 테스트 후 랜덤하다는 결과가 나오면 $S_{0,0,0}, S_{1,0,0}, S_{2,0,0}, S_{3,0,0,0}$를 후보키로 판정을 하고 그렇지 않으면 $S_{0,0,0}, S_{1,0,0}, S_{2,0,0}, S_{3,0,0,0}$는 키 후보에서 제외한다.

위의 방법을 적용하게 되면 다음 2가지의 오탐율이 해당 방법의 실제 응용 가능성을 결정하게 된다. 먼저 false positive 오탐율은 실제 키가 아니면서 랜덤성 테스트를 통과하게 될 확률을 의미하고, false negative 오탐율은 실제 키지만 랜덤성 테스트를 통과하지 못하는 확률을 의미한다.

Table 1은 위에서 제시된 방법을 서로 다른 10,000개의 키 값에 적용했을 경우의 테스트 결과를 보여주며 테스트는 다음과 같은 환경에서 진행되었다:

- 프로세서: Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz 3.40 GHz
- 메모리: 4GB
- OS: Windows7 Professional
- 컴파일러: Microsoft Visual Studio Professional 2012

Table 1에서 δ는 AES 비트열의 에러율을, p-value는 NIST의 (변형된) Frequency (Monobit) Test 및 Frequency Test within a Block에 대한 판단 기준이 되는 p-value를, '# of failures'는 실제 키임에도 불구하고 상기 테스트를 통과하지 못하는 횟수를, # of candidates는 상기 테스트를 통과한 키 후보의 개수를 의미한다. 따라서 표에서 false positive 오탐율은

$$\frac{\#\text{of candidates}}{2^{32}}$$

로, false negative 오탐율은

$$\frac{\#\text{of failures}}{10000}$$

로 계산 가능하다.

Table 1이 보여주듯이 본 논문에서 제시하는 방법은 δ 및 p-value에 따라 AES 키 복구의 성공 가능성이 다양하게 변함을 알 수 있으며, 특히 p-value는 본 논문에서 제시하는 방법의 성공확률에 많은 영향을 주고 있다. 즉 p-value가 높아지면 실제 키임에도 불구하고 상기 테스트를 통과하지 못할 확률, 즉 false negative 오탐율은 높아지지만, false positive 오탐율은 낮아짐을 알 수 있다. 따

Table 1. Result of Recovering the Partial AES Key Bits

δ	p-value	# of failures	# of candidates
0.1	0.01	0	484
	0.02	0	491
	0.05	0	209
	0.1	0	78
	0.2	0	25
	0.3	2	9
0.15	0.01	0	60,300
	0.02	0	24,678
	0.05	0	9,331
	0.1	0	2,586
	0.2	6	820
	0.3	10	199
0.2	0.01	0	2,685,556
	0.02	0	1,098,030
	0.05	0	399,403
	0.1	7	69,975
	0.2	10	20,407
	0.3	17	5,719

라서 공격자의 공격 환경에 따라 p-value 값을 조정함으로써 공격 시간 및 공격 성공 확률을 trade-off할 수 있게 된다.

이러한 키 복구 방법을 확장하면 다음과 같은 AES128 알고리즘의 키 전체를 복구하는 알고리즘을 쉽게 유도할 수 있다.

Algorithm 2. (AES128 키 전체를 복구)

1) Algorithm 1을 이용하여 $\tilde{S}_{0,0,0}, \tilde{S}_{1,0,0}, \tilde{S}_{2,0,0}, \tilde{S}_{3,0,0}$ 및 $\tilde{S}_{0,3,1}, \tilde{S}_{1,3,1}, \tilde{S}_{2,3,1}, \tilde{S}_{1,2,1}, \tilde{S}_{2,2,1}, \tilde{S}_{2,1,1}$로 부터 후보 $S_{0,0,0}, S_{1,0,0}, S_{2,0,0}, S_{3,0,0,0}$ 및 $S_{0,3,1}, S_{1,3,1}, S_{2,3,1}, S_{1,2,1}, S_{2,2,1}, S_{2,1,1}$를 계산한다.

2) Algorithm 1을 이용하여 $\tilde{S}_{0,0,1}, \tilde{S}_{1,0,1}, \tilde{S}_{2,0,1}, \tilde{S}_{3,0,1}$ 및 $\tilde{S}_{0,3,2}, \tilde{S}_{1,3,2}, \tilde{S}_{2,3,2}, \tilde{S}_{1,2,2}, \tilde{S}_{2,2,2}, \tilde{S}_{2,1,2}$로 부 터 후보 $S_{0,0,1}, S_{1,0,1}, S_{2,0,1}, S_{3,0,0,1}$ 및 $S_{0,3,2}, S_{1,3,2}, S_{2,3,2}, S_{1,2,2}, S_{2,2,2}, S_{2,1,2}$를 계산한다.

3) Algorithm 1을 이용하여 $\tilde{S}_{0,0,2}, \tilde{S}_{1,0,2}, \tilde{S}_{2,0,2}, \tilde{S}_{3,0,2}$ 및 $\tilde{S}_{0,3,3}, \tilde{S}_{1,3,3}, \tilde{S}_{2,3,3}, \tilde{S}_{1,2,3}, \tilde{S}_{2,2,3}, \tilde{S}_{2,1,3}$로 부 터 후보 $S_{0,0,2}, S_{1,0,2}, S_{2,0,2}, S_{3,0,0,2}$ 및 $S_{0,3,3}, S_{1,3,3}, S_{2,3,3}, S_{1,2,3}, S_{2,2,3}, S_{2,1,3}$를 계산한다.

4) Algorithm 1을 이용하여 $\tilde{S}_{0,0,3}, \tilde{S}_{1,0,3}, \tilde{S}_{2,0,3}, \tilde{S}_{3,0,3}$ 및 $\tilde{S}_{0,3,0}, \tilde{S}_{1,3,0}, \tilde{S}_{2,3,0}, \tilde{S}_{1,2,0}, \tilde{S}_{2,2,0}, \tilde{S}_{2,1,0}$로 부 터 후보 $S_{0,0,3}, S_{1,0,3}, S_{2,0,3}, S_{3,0,0,3}$ 및 $S_{0,3,0}, S_{1,3,0}, S_{2,3,0}, S_{1,2,0}, S_{2,2,0}, S_{2,1,0}$를 계산한다.

5) 위에서 계산된 키 후보를 이용하여 2라운드 부분키 S_2를 복구한다.

6) S_2로부터 AES 후보 키를 계산하고 이 후보 키의 유효성을 검사한다.

Algorithm 2의 6번 단계의 키 유효성 검사는 예를 들어, 실제 AES키로 암호화된 평문, 암호문 쌍이 있다는 가정 하에 복구된 키를 이용하여 해당 평문을 암호화한 후 그 결과 암호문이 주어진 암호문과 일치하는지를 검사함으로써 가능하다.

Table 2는 이러한 과정을 통해 실제 AES 키를 복구하는 과정에 대한 실험 결과를 보여주고 있으며,

Table 2. Result of Recovering the whole AES Key Bits

δ	p-value	success prob.	# of candidates
0.1	0.01	100	$5.5*10^{10}$
	0.02	100	$5.8*10^{10}$
	0.05	100	$1.9*10^{9}$
	0.1	100	$3.7*10^{7}$
	0.2	100	$3.9*10^{5}$
	0.3	99.9	$6.6*10^{3}$
0.15	0.01	100	$1.3*10^{19}$
	0.02	100	$3.7*10^{17}$
	0.05	100	$7.6*10^{15}$
	0.1	100	$4.5*10^{13}$
	0.2	99.8	$4.5*10^{11}$
	0.3	99.6	$1.6*10^{9}$
0.2	0.01	100	$5.2*10^{25}$
	0.02	100	$1.5*10^{24}$
	0.05	100	$2.5*10^{22}$
	0.1	99.7	$2.4*10^{19}$
	0.2	99.6	$1.7*10^{17}$
	0.3	99.3	$1.1*10^{15}$

표에서 success prob.는 성공 확률을, # of candidates는 Algorithm 2의 1) ~ 4)단계를 모두 통과한 후보 키의 개수를 나타낸다.

IV. 결 론

Cold Boot Attack이란, 전원이 차단된 DRAM으로부터 암호 알고리즘의 키 비트열과 같은 민감한 정보를 복구해내는 일종의 부채널 공격 방법으로, 본 논문에서는 이 중에서 대칭붕괴모델을 가정하고 이로부터 원래의 키를 복구하거나 또는 원래 키의 후보가 될 수 있는 키후보 공간의 크기를 줄일 수 있는 알고리즘을 제시하였다. 제시된 방법은 추출된 AES 키 비트열의 랜덤성을 테스트하는 방법을 사용하여 후보 키 공간의 크기를 줄이는 방법을 사용하였다.

References

[1] J.A. Halderman, S.D. Schoen, N. Heninger, W. Clarkson, W. Paul, J.A. Calandrino, A.J. Feldman, J. Appelbaum, E.W. Felten, "Lest we remember: cold boot attacks on encryption keys", USENIX Security Symposium, pp. 45-60, 2008.

[2] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems", CRYPTO '96, LNCS 1109, pp. 104-113, 1996.

[3] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis", CRYPTO '99, LNCS 1666, pp. 388-397, 1999.

[4] T. Meserges, "Securing the AES finalists against power analysis attacks", FSE 2000, LNCS 1978, pp. 150-165, 2000.

[5] NIST, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", NIST Special Publication 800-22 Revision 1a, 2010.

[6] A. Tsow, "An Improved Recovery Algorithm for Decayed AES Key Schedule Images", SAC 2009, LNCS 5867, pp. 215-230, 2009.

[7] A.A. Kamal and A.M. Youssef, "Applications of SAT Solvers to AES key Recovery from Decayed Key Schedule Images", SECURWARE 2010, 2010.

[8] M. Albrecht and C. Cid, "Cold Boot Key Recovery by Solving Polynomial Systems with Noise", ACNS 2011, LNCS 6715, pp. 57-72, 2011.

[9] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996.

[10] N. Heninger and H. Shacham, "Reconstructing rsa private keys from random key bits", CRYPTO '09, LNCS 5677, pp. 1−17, 2009.

[11] W. Henecka, A. May and A. Meurer, "Correcting Errors in RSA Private Keys", CRYPTO '10, LNCS 6223, pp. 351−−369, 2010.

〈저자소개〉

백 유 진 (Yoo-Jin Baek) 종신회원
1997년 2월: 서울대학교 수학과 졸업
1999년 2월: 서울대학교 수학과 이학석사
2003년 2월: 서울대학교 수리과학부 이학박사
2003년 3월~2003년 6월: KAIST 박사후 연구원
2003년 7월~2013년 3월: 삼성전자 책임연구원
2013년 3월~현재: 우석대학교 정보보안학과 조교수
〈관심분야〉 부채널 공격, 정보 보안