情報保護學會論文誌
第 21 卷 第 5 號, 2011. 10

소프트웨어 기반 스트림 암호 Salsa20/12에 대한 상관도 전력분석 공격

박 영 구,[1†] 배 기 석,[1] 문 상 재,[1] 이 훈 재,[2] 하 재 철,[3] 안 만 기[4‡]
[1]경북대학교, [2]동서대학교, [3]호서대학교, [4]국방기술품질원

Correlation Power Analysis Attacks on the Software based Salsa20/12 Stream Cipher

YoungGoo Park,[1†] KiSeok Bae,[1] SangJae Moon,[1] HoonJae Lee,[2]
JaeCheul Ha,[3] MahnKi Ahn[4‡]
[1]Kyungpook National University, [2]Dongseo University, [3]Hoseo University, [4]DTaQ

요 약

최근 유럽연합의 eSTREAM 공모사업에서 소프트웨어 분야에 선정된 Salsa20/12 알고리즘은 제한된 메모리의 8비트 MCU 상에서 AES보다 우수한 성능을 보여주는 스트림 암호이다. 또한 이론적 분석에 따르면 시차분석공격에 대한 취약성은 없으며, 전력분석 공격의 어려움에 대해서는 하위수준(low)로 평가되었으나, 현재까지 실제 전력분석 공격의 연구 결과가 발표된 바 없다. 따라서 본 논문에서는 소프트웨어 기반 Salsa20/12에 대한 상관도 전력분석 공격 방법을 제안하고 실험을 통하여 검증하였다. 실험을 위해서 프로그래밍이 가능한 8비트 RISC 계열의 AVR 마이크로프로세서 (ATmega128L)를 장착한 실험보드에 전력분석 공격의 대응방법이 적용되지 않은 시스템을 구현하고, 해밍무게 모델을 적용한 전력분석 공격을 실시하였다.

ABSTRACT

The Salsa20/12 stream cipher selected for the final eSTREAM portfolio has a better performance than software implementation of AES using an 8-bit microprocessor with restricted memory space. In the theoretical approach, the evaluation of exploitable timing vulnerability was 'none' and the complexity of side-channel analysis was 'low', but there is no literature of the practical result of power analysis attack. Thus we propose the correlation power analysis attack method and prove the feasibility of our proposed method by practical experiments. We used an 8-bit RISC AVR microprocessor (ATmega128L chip) to implement Salsa20/12 stream cipher without any countermeasures, and performed the experiments of power analysis based on Hamming weight model.

Keywords: Salsa20/12 Stream Cipher, Side Channel Cryptanalysis, Correlation Power Analysis

I. 서 론

정보화 기술 및 정보화 기기의 발달로 언제라도 원하는 정보를 실시간으로 이용자 요구에 맞춘 형태로 제공할 수 있는 유비쿼터스 시대가 오고 있으며, 이는 전자금융서비스, 근거리 무선 통신, 물류/유통, 환경 감시, 홈 오토메이션 등 다양한 분야에 적용되고 있다. 따라서 유비쿼터스 핵심기술 중 하나인 무선 센서 네트워크(wireless sensor network)에 대한 관심이 높아지고 있다 [1]. 무선 센서 네트워크는 연산과

접수일(2011년 1월 20일), 수정일(2011년 4월 9일),
게재확정일(2011년 7월 7일)
† 주저자, park09@empas.com
‡ 교신저자, amk93@paran.com

무선 통신능력을 지닌 여러 센서 노드(sensor node, MOTE)들로 구성되는데, 이러한 센서 노드는 여러 형태의 공격(도청, 해킹, 가입자 비밀정보 유출, 서비스 도착상태(마비) 등)에 취약한 것으로 알려져 있다. 따라서 빠른 데이터 처리속도의 장점으로 하는 스트림 암호에 의한 암/복호화가 필요하다[2,3].

최근 활발히 연구되고 있는 전력분석 공격(power analysis attacks) [4]은 암호 알고리즘을 설계할 때 고려되지 못한 부가적인 정보의 누출에 의해 비밀정보를 알아내는 방식으로 스트림 암호 역시 공격의 대상이 될 수 있다. 이미 스트림 암호(RC4, E0, A5, eSTREAM ciphers)에 대한 공격 [5]이 소개되어 졌으며, 국내에서도 최근 스트림 암호에 대한 부채널 분석 공격의 연구가 진행되고 있다 [6].

유럽연합(EU)의 eSTREAM 공모사업의 소프트웨어 분야(profile 1)에서 최종 선정된 알고리즘 중 Salsa20/12 [7]은 순수 암호화 시간만을 놓고 비교할 경우 제한된 메모리를 가지는 8비트 MCU 환경에서 AES에 비해 빠른 성능을 보여주는 것으로 알려져 있다 [8]. 또한, 스트림 암호의 형태이지만 블록 암호 시스템에서 고속처리를 위한 카운터 모드와 유사한 방식으로 동작할 수 있도록 구현되기 때문에, 병렬처리를 필요로 하는 이미지 암호화 영역에서도 활용되고 있다 [9]. 스마트카드, RFID, 센서 노드 등의 무선이동 단말에 구현될 경우 발생할 수 있는 위험성에 대한 이론적 분석 결과, 캐쉬 시차분석공격에 대한 취약점은 없으나, 전력분석 공격에 대한 안전성은 하위수준으로 평가됨에 따라 이에 대한 고려가 필요하다 [10].

본 논문은 128비트의 비밀키와 64비트의 초기벡터를 사용하여 소프트웨어로 구현된 Salsa20/12 알고리즘에 상관도 전력분석 공격(correlation power analysis)을 수행한다. 8비트 RISC 계열의 AVR 마이크로프로세서(ATmega128L)에 대응책이 탑재되지 않은 Salsa20/12 알고리즘을 구현하고 해밍무게 모델을 적용한 실제적인 상관도 전력분석 공격에 취약함을 실험으로 검증한다.

II. Salsa20 알고리즘

소프트웨어 기반 암호인 Salsa20 알고리즘 [7]은 2005년에 Daniel J. Bernstein에 의해 설계 및 구현되어 eSTREAM 공모사업에 제안되었다. 각 수행되는 라운드에 따라 8, 12, 20으로 구분되며, 그 중 12 라운드를 사용하는 Salsa20/12는 2008년 eSTREAM 공모사업의 소프트웨어 분야에 최종 선정된 4개의 알고리즘에 포함되었으며, 256비트를 사용하는 스트림 암호 중에 가장 우수한 데이터 처리속도를 가지고 있다. Salsa20/12는 스트림 암호이나, 블록 암호의 카운터 모드와 유사한 방식으로 동작하기 때문에 병렬처리에 적합한 구조이며, 계층적인 구조의 특성에 의해 하위단계에서 병렬처리가 가능하다면 전체적인 병렬화의 적용이 가능하다. 따라서 고속의 암·복호화 속도를 요구하는 시스템에 효과적인 알고리즘으로 동일한 환경 상에서 소프트웨어 기반의 AES보다 매우 빠른 성능을 가지고 있음이 확인되었다[8,11]. 또한 무선센서 네트워크에 적용될 수 있도록 국내에서 안전성 및 저전력 특성을 확보하는 연구가 활발히 진행되고 있다 [12].

2.1 Salsa20/12 알고리즘의 구조

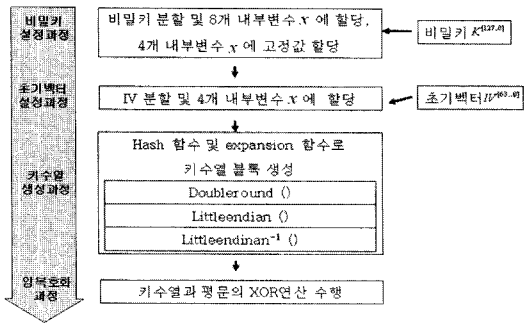
Salsa20/12 알고리즘은 256비트 또는 128비트의 비밀키와 64비트의 초기벡터(IV)를 사용하여 해쉬, 확장, 암·복호화를 차례로 수행한다. 비밀키와 초기벡터에 의해 생성되는 32비트의 워드 단위의 4×4 행렬 내부변수의 크기는 총 64바이트로 Quarterround, Rowround, Columnround, Doubleround, Salsa20 hash, Salsa20 expansion 함수들을 차례로 수행하여 최종적으로 512 비트의 키수열 블록을 생성하게 된다. 64바이트(16개의 32비트 내부변수)의 입력과 출력을 갖는 해쉬 함수는 Salsa20/12의 핵심 기술로 카운터 모드의 역할을 수행하며, 64바이트의 평문과 키, IV를 사용하여 생성한 키수열 블록에 대해 배타적 논리합(XOR) 연산을 수행하여 암호문을 생성한다 [13]. 이를 요약하면 아래와 같다.

(1) 비밀키 설정(key setup)

입력된 256비트 또는 128비트 비밀키의 길이로 연산될 내부변수를 초기화하는 과정이다. Littleendian 연산과정을 통해 사전에 정해진 상수값과 비밀키 값으로 16개 내부변수 중에서 12개를 초기화한다.

(2) 초기벡터 설정(initial vector setup)

입력된 128비트 또는 64비트의 초기벡터로 16개의 내부변수 중 4개를 초기화한다.



〔그림 1〕 Salsa20/12 알고리즘의 연산구조

(3) 키수열 생성

Salsa20 expansion function을 이용하여 512비트의 키수열을 생성한다.

(4) 암·복호화

생성된 키수열과 평문을 XOR 연산하여 8비트씩 암·복호된 비트열을 생성한다.

Salsa20/12는 〔그림 1〕과 같은 구조이며, 해쉬함수와 expansion 함수를 수행하여 64바이트의 키수열 블록를 생성하여 암호문을 출력하게 된다. 사용되는 표기와 함수는 아래와 같다.

- + : 32비트 원소 A와 B의 합(addition)에 mod 232
- $\oplus$: 비트단위의 배타적 논리합 연산 (XOR)
- $A \gg n$: A의 각 비트를 오른쪽으로 n 비트씩 순환이동
- $A \ll n$: A의 각 비트를 왼쪽으로 n 비트씩 순환이동
- $\|$: 연접
- 키수열 생성 과정의 입력 값 : 32비트의 워드로 이루어진 4×4 행렬의 내부변수의 형태는 아래와 같다.

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} \quad (1)$$

비밀키 설정과정에서 할당된 내부변수 x_0, x_5, x_{10}, x_{15}는 고정된 상수값으로 식 (2)와 같이 할당되며, 비밀키가 128비트일 경우는 비밀키의 부분값이 $x_1 = x_{11}$, $x_2 = x_{12}$, $x_3 = x_{13}$ 및 $x_4 = x_{14}$와 같이 할당된다.

$$\begin{aligned} x_0 &= 0x61707865, \\ x_5 &= 0x3120646e, \\ x_{10} &= 0x79622d36, \\ x_{15} &= 0x6b206574. \end{aligned} \quad (2)$$

또한 초기벡터 설정과정에서 입력된 초기벡터는 $x_6 = x_8$, $x_7 = x_9$에 할당되며, 초기벡터가 64비트일 경우 x_8와 x_9는 '0'으로 채워진다.

○ Littleendian function

Littleendian 함수는 32비트 내부변수의 순서를 바꾸는 기능으로 입력된 32비트 $b = (b_0, b_1, b_2, b_3)$을 식 (3)과 같이 순서를 변환한다.

$$littleendian(b) = b_0 + 2^8 b_1 + 2^{16} b_2 + 2^{24} b_3 \quad (3)$$

○ Quarterround function

해쉬 연산을 위한 Salsa20 알고리즘의 가장 기본적인 연산함수로 128비트(4개의 32비트 내부변수 $x_i (0 \le i \le 3)$)의 입력으로 128비트의 출력 $y_i (0 \le i \le 3)$을 만들어 낸다. 내부 연산과정은 식 (4)와 같이 입력된 32비트 변수간의 덧셈연산과 순환이동 후 XOR 연산으로 구성된다.

$$\begin{aligned} y_1 &= x_1 \oplus ((x_0 + x_3) \lll 7), \\ y_2 &= x_2 \oplus ((y_1 + x_0) \lll 9), \\ y_3 &= x_3 \oplus ((y_2 + y_1) \lll 13), \\ y_0 &= x_0 \oplus ((y_3 + y_2) \lll 18). \end{aligned} \quad (4)$$

연산순서는 입력값 x_1가 출력값 y_1을 생성하는 순으로 연산되어 최종 y_0까지 연산된다.

○ Rowround function

512비트의 입력(16개의 32비트 내부변수 $x_i (0 \le i \le 15)$)으로 512비트의 출력 $y_i (0 \le i \le 15)$을 만들어 낸다. 내부 연산과정은 Quarterround 함수를 이용하여 식 (5)와 같은 정방행렬을 구성하여, 입력 x_i에 대하여 행렬의 행 구성원소를 갱신한다. 이 때 첫 번째 열은 x_1을 먼저 갱신하고, 두 번째 열은 x_6을, 세 번째 열은 x_{11}을, 네 번째 열은 x_{12}의 순으로 먼저 갱신한다.

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} \quad (5)$$

○ Columnround function

512비트의 입력(16개의 32비트 내부변수, $x_i(0 \le i \le 15)$)으로 512비트의 출력 $y_i(0 \le i \le 15)$을 생성한다. 내부 연산과정도 동일하게 Quarterround 함수를 이용하여 식 (6)과 같은 행렬의 열 구성원소를 갱신한다. 첫 번째 행은 x_4을 먼저 갱신하고 x_8, x_{12} 및 x_0 순으로 계산한다. 두 번째 행은 x_9을, 세 번째 행은 x_{14}을, 네 번째 행은 x_3의 순으로 먼저 갱신한다.

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} \quad (6)$$

○ Doubleround function

512비트의 입력(16개 32비트 내부변수 $x_i(0 \le i \le 15)$)으로 512비트의 출력을 만들어 낸다. Doubleround 함수는 Columnround 함수를 수행한 후 Rowround 함수를 차례로 수행한다.

○ Salsa20 hash function

512비트의 입력(64개의 8비트 배열 $x=(x[0], x[1], ..., x[63])$)으로 512비트의 출력 $y_i(0 \le i \le 15)$을 만들어 낸다. 먼저 64개의 8비트를 4개씩 Littleendian 연산을 통해 16개의 32비트 내부변수 $x_i(0 \le i \le 15)$로 만든다. 그 다음 Doubleround 함수를 10라운드 연산한 결과값 $z_i(0 \le i \le 15)$과 32비트 내부변수 $x_i(0 \le i \le 15)$을 덧셈연산 수행한 후, 역 Littleendian 함수의 입력으로 활용하여 16개의 32비트 출력값을 서로 연접하여 최종 64개의 8비트 출력값을 생성한다. 이러한 연산은 Salsa20 expansion function의 내부동작 과정이며 전력분석 공격의 대상이 되고 있다.

$$Salsa20(x) = x + doubleround^{10}(x) \quad (7)$$

$$x_0 = littleendian(x[0], x[1], x[2], x[3]),$$
$$x_1 = littleendian(x[4], x[5], x[6], x[7]),$$
$$\vdots$$
$$x_{15} = littleendian(x[60], x[61], x[62], x[63])$$
$$z = (z_0, z_1, ..., z_{15}) = doubleround^{10}((x_0, x_1, ..., x_{15}))$$

$$y = (y_0, y_1, ..., y_{15}) = littleendian^{-1}(z_0 + x_0) \,\|\, littleendian^{-1}(z_1 + x_1) \,\|\, ... \,\|\, littleendian^{-1}(z_{15} + x_{15})$$

○ Salsa20 expansion function

키수열 생성과정에 사용되는 Salsa20 expansion함수는 내부변수 x, 비밀키 k, 초기벡터 n, 상수값을 사용하여 4×4 행렬 구조를 구성하고 최종 512비트의 출력값을 생성한다. [그림 1]과 같이 128비트 비밀키 k와 초기벡터 IV를 사용하는 확장함수로서 Doubleround 함수와 Littleendian 함수를 사용한다.

○ Salsa20 encryption function

Salsa20 encryption 함수는 식 (8)과 같이 Salsa20 expansion function의 출력값과 l개의 8비트 평문 m을 XOR 연산하여 l개의 8비트 암호문을 생성한다.

$$Salsa20_k(iv) \oplus (m[0], m[1], ..., m[l-1]) = (c[0], c[1], ..., c[l-1]) \quad (8)$$

III. 제안하는 상관도 전력분석 공격 모델

3.1 Salsa20/12 알고리즘의 공격시점 선정

본 논문에서는 공격대상이 되는 Salsa20/12은

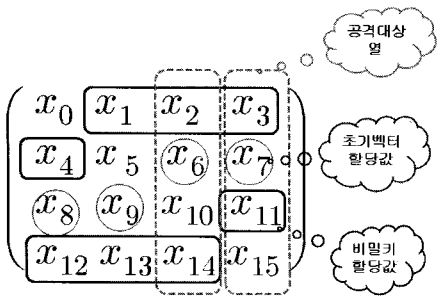
Quarterround 함수
(Salsa20_wordtobyte()의 일부분)

입력값 : 16개의 32비트 $x=(x_0, x_1, ..., x_{15})$
출력값 : 64개의 8비트 $y=(y_0, y_1, ..., y_{63})$

```
// 식 (2)에 의해 입력된 x를 연산하며 총 10 라운드를 실시
for j=0 to 10 do
x[ 4] = x[ 4]^ROTL32(U32V(x[ 0]+x[12]), 7);
⋮
x[14] = x[14]^ROTL32(U32V(x[10]+x[ 6]), 7);
// 세 번째 열의 연산
x[ 2] = x[ 2]^ROTL32(U32V(x[14]+x[10]), 9);
x[ 6] = x[ 6]^ROTL32(U32V(x[ 2]+x[14]),13);
x[10] = x[10]^ROTL32(U32V(x[ 6]+x[ 2]),18);
x[ 3] = x[ 3]^ROTL32(U32V(x[15]+x[11]), 7);
// 네 번째 열의 연산
x[ 7] = x[ 7]^ROTL32(U32V(x[ 3]+x[15]), 9);
x[11] = x[11]^ROTL32(U32V(x[ 7]+x[ 3]),13);
x[15] = x[15]^ROTL32(U32V(x[11]+x[ 7]),18);
⋮
 end for
⋮
for j=0 to 16 do
  U32TO8_LITTLE(y, x[i]);
end for
```

[그림 2] 입력된 내부변수 x를 연산하는 Quarterround 함수의 가상 코드

128비트 비밀키와 64비트의 초기벡터를 사용한다고 가정한다. 비밀키 설정과정과 초기벡터 설정과정은 키 수열 생성과정에 앞서 수행되며, 비밀키 설정과정은 항상 초기벡터와 무관하게 선행되기 때문에 재동기화 시마다 갱신되는 초기벡터가 아무런 영향을 미치지 못하므로 전력분석 공격의 대상이 될 수 없다. 또한 초기벡터 설정과정도 공격자가 알고 있는 초기벡터를 입력하여도 단순하게 내부변수 x에 초기벡터를 할당하는 과정에 불과하므로 비밀키와 연관되는 중간값이 발생하지 않으므로 공격의 대상이 되지 않는다. 따라서 전력분석 공격은 키수열 생성과정에서 Salsa20 확장함수의 내부함수 중 Doubleround 함수가 10 라운드 수행될 때 첫 번째 라운드의 Columnround 함수 내의 Quarterround 함수의 수행과정에서 실시한다. [그림 2]는 공개된 Quarterround 함수의 가상코드이다. 수식 (4)의 연산과정은 U32V 함수로 두 내부변수를 덧셈(addition)연산하여 ROTL32 함수로 내부변수를 n 비트씩 순환이동시킨 후, 최종적으로 배타적 논리합(XOR)을 수행하는 것으로 구현된다. 총 10 라운드의 반복적인 과정을 통해 내부변수 $x_i(0 \le i \le 15)$는 Littleendian 함수에 의해 최종 8비트씩 64개의 출력값 $y_i(0 \le i \le 63)$으로 변환된다.



[그림 3] Quarterround 함수 내부변수의 할당값 지정 및 공격 모델

가정한 초기벡터는 64비트이므로 내부변수 x_8와 x_9가 '0'으로 설정된다. 공격자는 해밍무게를 추정하기 위해 [그림 3]과 같이 Doubleround 함수의 정방행렬 연산과정에서 세 번째와 네 번째 열의 연산과정을 공격시점으로 결정한다.

3.2 세 번째 열의 내부변수 분석

사전에 비밀키 설정과정과 초기벡터 설정과정에서 내부변수가 할당되어진다. 따라서 공격자는 내부변수 x_6와 x_7에 할당된 해밍무게를 변경할 수 있는 초기벡터값을 변경하여 키수열 생성과정을 분석하는 방법을 사용한다. 공격의 대상은 비밀키가 할당된 내부변수 x_2와 x_{14}가 처음으로 갱신되는 첫 라운드이며, 그 중 내부변수 x_{14}의 연산시점에 대해서 먼저 분석한다. 공격자는 세 번째 열의 내부변수 중에서 초기벡터인 x_6과 상수 값인 x_{10}은 알고 있으므로 다음의 순차적인 공격을 통해 비밀키가 할당된 2개의 32비트 내부변수 x_2와 x_{14}을 알아낸다.

1) 갱신전·후의 32비트 내부변수 x_{14}

공격자는 초기벡터와 연관되는 내부변수 x_6의 해밍무게를 알고 있으며, 재동기화 과정을 통해 이를 변경할 수 있다. 마찬가지로 덧셈 연산을 수행하는 내부변수 x_{10}(상수값)의 해밍무게 또한 알고 있으므로, x_6의 해밍무게를 변화시켜가며 덧셈연산, 순환이동, 배타적 논리합 등의 연산에 따른 소모전력파형의 특성을 파악한다. 공격자는 x_6과 x_{10}의 연산 결과가 공격 대상인 비밀 값이 할당된 내부변수 x_{14}와 XOR 연산되는 시점에서 소모전력파형을 수집하여, x_6과 x_{10}의 연산 결과의 해밍무게 변화에 따른 소모전력파형의 분석을 통해 갱신후의 x_{14}을 알아내었다면 자연스럽게 갱신전의 x_{14}도 알 수 있다. 실험에 적용한 MCU가 8비트이므로 최소 4번에 걸친 공격이 수행되어져야 한다.

2) 갱신전·후의 32비트 내부변수 x_2

공격자가 연산후의 x_{14}을 알아내었다면 내부변수 x_6을 가변시켜 공격자가 원하는 내부변수 x_{14}의 값을 유도해 낼 수 있다. 이러한 방법으로 유도된 내부변수 x_{14}는 [그림 2]와 같이 다음에 이어지는 x_2의 갱신과정에서 사용된다. 이때 x_{14}는 알고 있는 내부변수 x_{10}(상수값)과 덧셈연산 및 순환이동 연산을 차례로 수행한다. 따라서 공격자는 IV가 할당된 내부변수 x_6의 가변에 따라 갱신된 x_{14}의 해밍무게를 알 수 있으므로, 이를 토대로 XOR 연산되는 내부변수 x_2의 해밍무게를 추정할 수 있다. 해밍무게 변화에 따른 소모전력파형의 분석으로 갱신후의 x_2을 알아내었다면 자연스럽게 갱신전의 x_2도 알 수 있다.

3.3 네 번째 열의 내부변수 분석

공격 방법은 세 번째 열에 적용한 방법과 동일하나

고려해야 할 내부변수가 3가지이다. 네 번째 열의 내부변수 중 먼저 연산이 되는 x_3는 비밀키의 일부이다. x_3의 갱신은 세 번째 열과 달리 고정된 상수인 내부변수 x_{15}와 비밀키의 일부인 x_{11}에 의해서 수행되므로, 알 수 없는 값 x_3, x_{11}과 알고 있는 x_{15}만으로는 분석이 용이하지 않다. 따라서 공격자는 네 번째 열에서 초기벡터에 의한 해밍무게를 변경할 수 있는 내부변수 x_7의 XOR 연산시점을 먼저 분석하여 갱신 후의 x_3을 확인한 후, 나머지 내부변수를 차례로 알아낸다. 다음은 [그림 2]의 순차적인 연산코드에서 비밀키로 할당된 2개의 32비트 내부변수 x_3와 x_{11}을 알아내는 공격단계이다.

1) 갱신후 32비트 내부변수 x_3

가변이 가능한 IV의 일부인 내부변수 x_7의 갱신 연산을 공격 시점으로 하여 공개된 상수 값 x_{15}와 연산되는 갱신 후의 내부 변수 x_3을 알아낸다. 수집된 소모전력파형을 이용하여 x_7의 해밍무게 변화에 따른 전력분석 공격을 통해 고정된 x_{15}와 연산되는 갱신 후의 x_3을 추출할 수 있으며, 갱신된 x_7의 값도 알아낼 수 있다. 그러나 갱신전 내부변수 x_3의 값은 내부변수 x_{11}을 알아내야 얻을 수 있다.

2) 갱신전·후의 32비트 내부변수 x_{11}

공격자는 앞 단계에서 알아낸 갱신된 내부변수 x_3, 갱신된 x_7을 사용한 덧셈과 순환이동 연산에 대해서 해밍무게를 추측하여 내부변수 x_{11}을 알아낸다. 알고 있는 x_3과 가변되는 x_7의 해밍무게를 이용하여 x_{11}과 XOR 연산되는 시점에 대한 소모전력파형을 수집하고, 해밍무게의 변화에 대한 전력분석 공격을 통해 갱신전·후의 내부변수 x_{11}를 알아낼 수 있다. 갱신되지 않은 내부변수 x_{11}의 값은 다음 단계의 공격 대상인 내부변수 x_3의 갱신 연산에서 입력으로 사용된다.

3) 갱신전 32비트 내부변수 x_3

상수인 내부변수 x_{15}, 갱신 후의 내부변수 x_3, 그리고 갱신되지 않은 내부변수 x_{11}을 활용하여 갱신되지 않은 32비트 내부변수 x_3을 자연스럽게 알아낼 수 있다.

IV. 상관도 전력분석 공격 모델의 실험 및 분석

4.1 실험 환경

실험을 위해서 8비트 RISC 계열의 AVR 마이크로프로세서(ATmega128L)에 공개된 Salsa20/12 알고리즘 코드를 AVR Studio의 컴파일러를 통해 탑재하였다. 칩에서 소모되는 전력을 측정하기 위해서 디지털 오실로스코프를 사용하며, PC에서는 제어프로그램을 통해 전력소모파형을 수집한다. 또한 칩에 인가되는 초기벡터의 해밍무게를 구분하거나 수집된 전력소모파형을 분석하기 위해서 Matlab Toolbox 프로그램을 사용하며, 센서 노드가 동작할 때 반복적인 초기벡터의 입력을 위해서 칩과 연동되는 신호처리 제어프로그램을 구현하여 사용한다. [그림 4]는 전력분석 공격을 위한 구성과 실제 장비 설치도이다.

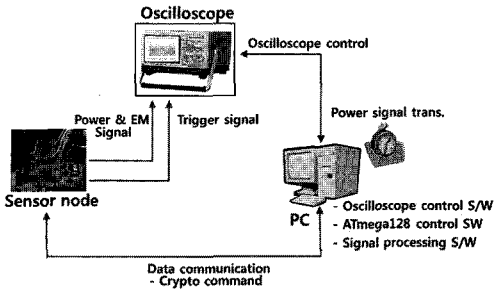
4.2 상관도 전력분석 공격 수행 결과

Salsa20/12 알고리즘이 동작할 때 키수열 생성과정에서 상관도 전력분석 공격을 수행한다. 키수열 생성과정의 상관도 전력분석 공격은 해밍무게 모델의 식 (9)와 같은 추정모델을 사용하여 공격을 수행한다. 공격 대상인 8비트 칩의 특성에 따라 공격자는 최하위부터 8비트씩 추정한다.

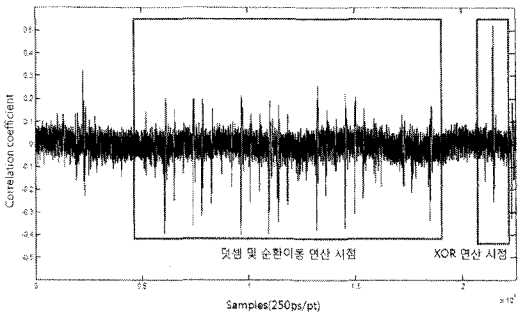
추정모델 :

$$HW((x_{14} \oplus ((x_{10} + Variable\ \ x_6) \lll 7))|_{8LSB}) \qquad (9)$$

공격 시점을 정확하게 찾아내기 위해서 먼저 전체 전력소모파형에서 비밀키 설정과정 및 초기벡터 설정과정과 키수열 생성과정의 동작하는 시점을 구분하는 것이 요구된다. 따라서 초기벡터를 변경시켜가며 소모전력파형을 수집하여 평균한 후 차분하면 매번 동일한



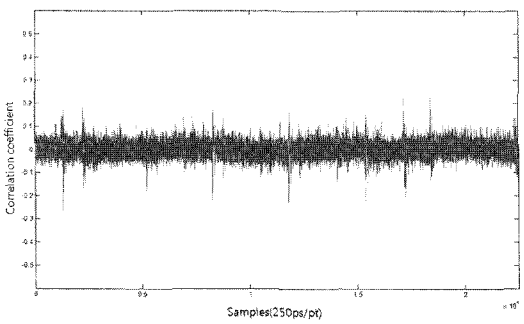
[그림 4] 전력분석 공격을 위한 장비간 제어신호 구성



〔그림 5〕 내부변수 x_{14}의 하위 8비트의 상관도 전력분석 공격결과(올바른 추정의 경우)

연산을 수행하는 비밀키 생성과정은 0에 가까운 결과를 보이게 되며, 마찬가지로 일부의 초기벡터만을 변경하는 과정을 통해서 초기벡터 설정과정 연산시점도 확인할 수 있었다. 키수열 생성과정에서의 정확한 시점을 파악하기 위해서는 공격 대상 칩에 탑재되는 어셈블리어를 기반으로 동작클럭 파형을 측정하여 분석하였다. 어셈블리어들은 일정한 동작클럭에 맞춰 동작하므로 공격 대상인 Quarterround 함수 이전에 사용되어야할 클럭 수을 확인한 후, 소모전력파형, 동작클럭을 분석하여 각 내부변수의 갱신 시점을 찾아낼 수 있었다.

본 논문에서 수행된 해밍무게 모델을 이용한 상관도 전력분석 공격에서는 내부변수가 32비트씩 차례로 연산되나 공격 대상 칩의 특성에 따라 공격자는 8비트씩 추측하여 총 4회를 수행한다. 먼저 세 번째 열의 내부변수 x_{14}의 연산과정에 대한 공격 결과, 1,000개의 소모전력파형을 사용하였을 때 〔그림 5〕와 같은 공격자가 원하지 않는 지점에서의 고스트 피크를 얻을 수 있었다. 동작클럭 분석 결과 고스트 피크 성분들은 고정상수값 x_{10}(0x79622d36)과 내부변수가 덧셈연산 및 순환이동 연산과정에서 나타났다. 정확한 분석을 위해서 공격자가 원하는 XOR 연산시점에서의 피크 성분을 분석한 결과 32비트 중 추정한 하위 8비트의 값이 올바른 경우에는 〔그림 5〕와 같이 XOR 연산 시점에서 큰 피크 성분이 검출된 반면, 〔그림 6〕과 같이 올바르지 않은 추정의 경우에는 특별한 피크 성분이 검출되지 않아 올바른 경우와의 차이를 확인할 수 있었다. 〔그림 5와 6〕상에서 파형을 표현하기 위한 샘플율은 250ps/pt이므로 총 10,000개의 샘플 포인트를 사용하고 있으며, 하나의 포인트는 x축의 한 점이 된다.

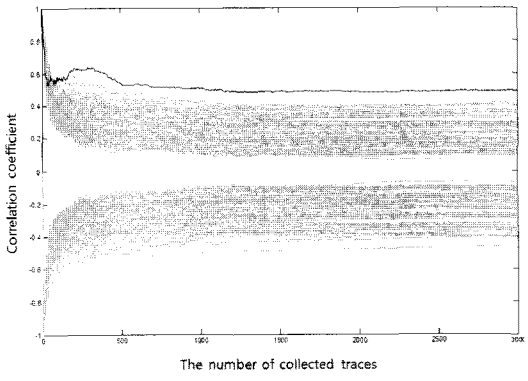


〔그림 6〕 내부변수 x_{14}의 하위 8비트의 상관도 전력분석 공격결과(잘못된 추정의 경우)

〔그림 7〕은 내부변수 x_{14}의 상관도 전력분석 공격 시 올바른 경우를 찾을 수 있는 전력소모파형 수를 보여주고 있다. 하위 8비트를 올바르게 추측한 경우가 검정색이며, 엷은 회색은 그 외 255가지 경우의 상관도이다. 최대 상관계수의 측정은 〔그림 5〕의 XOR 연산시점을 기준으로 하였으며, 수집한 파형의 수가 커질수록 올바른 경우와 나머지 경우와의 차이가 뚜렷하게 나타남을 확인할 수 있었다. 〔그림 7〕에서는 최대 상관 계수의 결과가 대칭적으로 나타나는데, 이는 XOR 연산의 특성상 해밍무게에 대한 보수의 경우 음의(negative) 상관계수 값이 나타나기 때문이다.
동일한 방법을 반복적으로 수행하게 되면 세 번째 및 네 번째 열의 원하는 내부변수 x_2, x_{14}, x_3, x_{11}를 모두 알아낼 수 있다. 그러므로 공격자는 8비트씩 총 16회의 상관도 전력분석 공격을 통하여 4개의 내부변수로 비밀키(128비트)를 알아낼 수 있다.

4.3 상관도 전력분석 공격의 실험결과 분석

eSTREAM 공모사업의 소프트웨어 분야에 선정된



〔그림 7〕 추측 가능한 모든 내부변수의 상관계수(수집한 소모전력파형 수)

Salsa20/12 알고리즘의 연산소요시간은 약 6.6ms 이며, 8비트 마이크로프로세서의 소모전력은 해밍무게 모델과 연관성을 가지고 있음을 확인하였다. 전력분석 공격 측면에서 알고리즘이 동작할 때 안전성은 공격에 필요한 전력소모파형의 수에 의해 결정되며, 전력소모파형의 수는 최대 상관계수와 관련이 있다. S. Mangard [14]가 제시한 공식에 따라 신뢰도 $\alpha=0.9999$에서 $Z_{0.9999}=3.7190$일 때, [그림 5]의 하위 8비트에 대한 공격 결과를 기준으로 상관계수($\rho_{\max}$)가 0.48인 경우 약 110개 정도의 전력소모파형수로도 충분히 상관도 전력분석 공격이 가능함을 확인하였다.

(1) 해밍무게 모델과 공격 효율성

내부변수를 찾아내기 위해서 사용한 해밍무게 모델은 8비트를 기준으로 $2^8=256$ 가지의 가정을 토대로 분석된다. 전력분석 공격의 측면에서 공격의 대상이 1비트여도 공격의 실현이 가능하지만, 비트의 크기가 커짐에 따라 공격에 필요한 파형 수가 감소하는 등의 효율이 높아진다. 본 논문에서 설정한 실험 환경에서는 8비트 MCU를 사용하고 있기 때문에, 최대 8비트의 해밍무게 모델을 사용하게 된다. 따라서 32비트 내부변수를 찾기 위해서는 총 4회의 걸친 상관도 전력분석 공격이 수행되어야 한다. 앞서 수행한 하위 8비트 외에도 나머지 24비트에 대해서도 각각 공격을 수행하였으며, 그 결과는 [표 1]과 같다. 추가적인 실험의 결과로 32비트 내부변수 전체를 4차례에 걸친 8비트 단위의 전력분석 공격으로 찾아낼 수 있음을 확인하였다. 상관도 공격 결과들은 S. Mangard의 공식에 따라 평균 약 130여개의 전력소모파형으로 올바른 가정을 구분할 수 있어 충분히 효율적이며, 신호처리 과정이 추가된다면 보다 작은 수의 파형으로도 충분히 가능할 것이다. 또한 16비트나 32비트를 처리하는 MCU를 적용할 경우에는 추정할 경우의 수가 2^{16} ~ 2^{32}개로 상당히 증가하는 반면, 1~ 2회의 상관도 전력분석 공격을 수행으로 32비트의 내부변수를 알아내는 이점도 있다.

(2) 수행해야할 공격횟수

스트림 암호인 Salsa20/12의 경우 세 번째 열의 x_{14}와 x_2를 알아내기 위해 8비트의 해밍무게를 기준으로 8번의 공격이 필요하며, 네 번째 열의 갱신 전, 후의 x_3, x_{11}들을 알아내기 위해서 8번의 공격이 추가로 필요하다. 또한, 스트림 암호의 특성상 비밀키와 초기벡터와 연관된 새로운 내부변수들이 갱신됨에 따라 이에 대한 각 내부 변수들의 해밍무게 모델을 새로이 추정해야한다. 각 열의 내부 변수를 찾아내는 과정에서도 연관성이 존재하여 하나의 공격이 수행되지 않는다면 비밀키를 알아낼 수 없는 구조를 이루고 있다.

(3) 적용 가능한 대응책

소프트웨어 기반의 Salsa20/12 스트림 암호에서 고려할 수 있는 전력분석 공격의 방어책은 하드웨어적인 방어책을 배제한 소프트웨어 상의 방어책을 고려해야한다. 널리 알려진 소프트웨어 기반의 방어대책으로는 마스킹(masking)에 의한 중간 값의 랜덤화(randomization)와 더미 코드 삽입 또는 임의의 지연시간 삽입을 통한 공격 시점의 숨김화(hiding)가 사용되고 있다[15]. 그러나 스트림 암호 영역에서는 재동기화의 특성상 연산 시간이 일정해야하며, 마스킹을 적용할 때 발생하는 연산의 비효율성 때문에 대응기법에 대한 고려가 부족한 현실이다. 최근 소개된 스트림 암호 RC4에 적용된 연산 순서의 랜덤화의 경우 [16]에도 이전 연산에서 갱신된 내부변수를 사용해야하는 Salsa의 특성상 적용이 쉽지 않다. 마스킹을 적용하는 경우에도 공격의 대상이 되는 XOR 연산뿐만 아니라 공격 분석의 과정에 포함되는 덧셈 연산에 대해서도 고려해야 하기 때문에 부울 마스킹뿐만 아니라 산술 마스킹, 각 마스킹 간의 전환 등의 높은 비용이 요구될 것으로 고려된다.

V. 결 론

본 논문에서는 제한된 메모리를 가지는 8비트 MCU

[표 1] 상관도 공격 결과 비교(8비트 단위)

8비트 / 최대상관계수	target[31,…,24]	target[23,…,16]	target[15,…,8]	target[7,…,0]	비고
$\rho_{\max}$	0.43	0.47	0.41	0.48	파형 개수 N=1000

측면에서 AES를 대치하기에 가장 적합한 스트림 암호인 Salsa20/12 알고리즘의 상관도 전력분석 공격을 제안하고 실험을 통하여 검증하였다. Salsa20/12 알고리즘도 고성능 8비트 RISC 계열의 AVR 마이크로프로세서를 장착한 실험보드에 적용시 해밍무게 모델을 적용한 상관도 전력분석 공격에 취약함을 알 수 있었다. 이는 이론적인 분석과 함께 알고리즘의 특성을 분석한 공격으로 볼 수 있다. 따라서 전력분석 공격을 eSTREAM 공모사업의 소프트웨어 분야에 선정된 다른 알고리즘에도 적용되기 위해서는 각 알고리즘의 특성에 대한 분석이 선행되어야 하며, 향후 다양한 대응방법의 적용과 함께 고차 전력분석 공격 가능성도 함께 활발히 연구되어져야 할 것이다.

참고문헌

[1] F. Zhao, and L. Guibas, Wireless Sensor Networks: An Information Processing Approach, Elsevier/Morgan-Kaufmann, Oct. 2003.

[2] Y. Chen, and W.-S. Ku, "Self-Encryption Scheme for Data Security in Mobile Devices," Proceeding of Consumer Communications & Networking Conference (CCNC), pp. 1-5, Jan. 2009.

[3] R. Tahir, M. Y. Javed, A. Ahmad and R. Iqbal, "SCUR : Secure Communications in Wireless Sensor Networks using Rabbit," Proceedings of World Congress on Engineering 2008 - WCE 2008, vol 1, pp. 518-522, Jul. 2008.

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

[5] K.Wu, H.Li, B.Peng, F.Yu, "Correlation Power Analysis Attack against Synchronous Stream Ciphers", Proceedings of The 9th International Conference for Young Computer Scientists, pp.2067-2072, Nov. 2008.

[6] 배기석, 안만기, 박제훈, 이훈재, 문상재, "Rabbit 알고리즘에 대한 전력분석 공격," 한국정보보호학회 추계학술대회, pp. 12-24, 2010년 10월.

[7] D. Bernstein, "Salsa20," Available at http://www.ecrypt.eu.org/stream/salsa20p3.html

[8] S. Babbage, C. D. Canni`ere, A. Canteaut, C. Cid, H. Gilbert, T. Johansson, M. Parke, B. Preneel, V. Rijmen, and M. Robshaw, "The eSTREAM portfolio," eSTREAM, ECRYPT Stream Cipher Project, http://www.ecrypt.eu.org/stream/portfolio.pdf, Apr. 2008

[9] A. Jolfaei, and A. Mirghadri, "Survey : Image Encryption Using Salsa20," International Journal of Computer Science Issues, Vol. 7, Issue 5, pp. 213-220, Sep. 2010.

[10] B. Gierlichs, L. Batina, C. Clavier, T. Eisenbarth, A. Gouget, Helena H, T. Kasper, K .Lemkerust, S. Mangard, A. Moradi and E. Oswald, "Susceptible of eSTREAM Candidates towards Side Channel Analysis," Proceedings of SASC 2008-Candidate of the Art of Stream Ciphers, pp. 123-150, Feb. 2008.

[11] 윤민, 한태윤, 이문규, 오희국, "GPU를 이용한 Salsa20 스트림 암호의 병렬화," 한국차세대컴퓨터학회 논문지, Vol.5, No.1, pp. 53-62, 2009년 3월.

[12] 윤민, 나형준, 이문규, 박근수, "안전한 센서 네트워크를 위한 스트림 암호의 성능 비교 분석," 한국정보보호학회, 제18권, 제5호, pp. 4-16, 2008년 10월

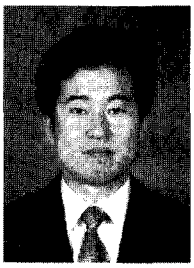
[13] Daniel J. Bernstein, Salsa20 specication, http://cr.yp.to/snuffle/spec.pdf

[14] S. Mangard, "Hardware Countermeasures against DPA-A Statistical Analysis of Their Effectiveness," Topics in Cryptology, CT-RSA 2004, LNCS 2964, pp. 222-235, 2004.

[15] M. Rivain, E. Prou, and J. Doget. "Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers," Cryptographic Hardware and Embedded Systems, CHES 2009, LNCS 5747, pp. 171-188, 2009.

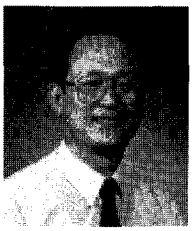
[16] Ilya Mironov, "(Not So) Random Shuffles of RC4," Advances in Cryptology, CRYPTO 2002, LNCS 2442, pp. 304-319, 2002.

〈著者紹介〉



박 영 구 (YoungGoo Park) 정회원
1986년 8월: 경북대학교 전자공학과 졸업(공학사)
1989년 2월: 경북대학교 대학원 전자공학과(공학석사)
1998년 8월: 경북대학교 대학원 전자공학과(박사수료)
1989년 1월~1997년 2월: 국방과학연구소 선임연구원
1997년 3월~2008년 2월: 창신대학 모바일통신과 조교수
2010년 1월~현재: 에스테크 연구개발 이사
〈관심분야〉 정보보호, 정보통신, 부채널 공격 등

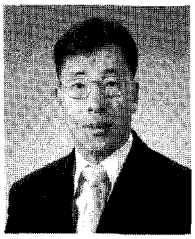
배 기 석 (KiSeok Bae) 학생회원
2006년 8월: 경북대학교 전자·전기공학부 졸업
2008년 8월: 경북대학교 전자공학과 석사
2009년 3월~현재: 경북대학교 전자전기컴퓨터공학부 박사과정
〈관심분야〉 정보보호, 네트워크 보안, 스마트카드 보안



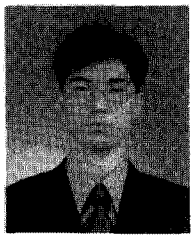
문 상 재 (SangJae Moon) 종신회원
1972년 2월: 서울대학교 공업교육(전자전공)과 학사
1974년 2월: 서울대학교 전자공학과 석사
1984년 6월: 미국 UCLA 전기공학과 박사
1984년 7월~1985년 6월: UCLA Postdoctor 근무
1984년 7월~1985년 6월: 미국 OMNET 컨설턴트
1997년 9월~1998년 8월: 경북대학교 전자전기공학부 학부장
1974년 12월~현재: 경북대학교 전자전기컴퓨터공학부 교수
2000년 8월~현재: 경북대학교 이동네트워크 정보보호기술 연구센터장
2002년 2월~현재: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

이 훈 재 (HoonJae Lee) 정회원
1985년 2월: 경북대학교 전자공학과 졸업(공학사)
1987년 2월: 경북대학교 전자공학과 졸업(정보통신공학, 공학석사)
1998년 2월: 경북대학교 전자공학과 졸업(정보통신공학, 공박사)
1987년 2월~1998년 1월: 국방과학연구소 선임연구원
1998년 2월~2002년 1월: 경운대학교 컴퓨터전자정보공학부 조교수
2002년 2월~현재 : 동서대학교 컴퓨터정보공학부 조교수
〈관심분야〉 암호이론, 네트워크보안, 디지털 통신

〈著者紹介〉



하 재 철 (JaeCheul 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



안 만 기 (MahnKi Ahn) 정회원
2000년 2월: 경북대학교 전자전기공학부 졸업(학사)
2000년 1월~2001년 1월: 삼성전자 프린터사업부 C-LBP 연구원
2003년 2월: 경북대학교 전자공학과 석사(정보통신공학)
2011년 2월: 경북대학교 정보보호학과 박사
2003년 4월~현재: 국방기술품질원 선임연구원
〈관심분야〉 정보보호, RFID/USN 보안, 부채널분석 공격