Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.5, Oct. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.5.1071

경량 암호 알고리듬 CHAM에 대한 오류 주입 공격

권 홍 필,† 하 재 철‡
호서대학교

Fault Injection Attack on Lightweight Block Cipher CHAM

Hongpil Kwon,† Jaecheol Ha‡
Hoseo University

요 약

최근 가용 자원이 제한된 디바이스에서 사용할 수 있는 구현 성능이 효율적인 경량 블록 암호 알고리듬 CHAM이 제안되었다. CHAM은 키의 상태를 갱신하지 않는 스케줄링 기법을 사용함으로써 키 저장 공간을 획기적으로 감소시켰으며, ARX(Addition, Rotation, and XOR) 연산에 기반하여 설계함으로써 계산 성능을 크게 향상시켰다. 그럼에도 불구하고 본 논문에서는 CHAM은 오류 주입 공격에 의해 라운드 키가 노출될 가능성이 있으며 4개의 라운드 키로부터 마스터 비밀 키를 추출할 수 있음을 보이고자 한다. 제안된 오류 주입 기법을 사용하면 약 24개의 정상-오류 암호문 쌍을 이용하여 CHAM-128/128에 사용된 비밀 키를 찾을 수 있음을 컴퓨터 시뮬레이션을 통해 확인하였다.

ABSTRACT

Recently, a family of lightweight block ciphers CHAM that has effective performance on resource-constrained devices is proposed. The CHAM uses a stateless-on-the-fly key schedule method which can reduce the key storage areas. Furthermore, the core design of CHAM is based on ARX(Addition, Rotation and XOR) operations which can enhance the computational performance. Nevertheless, we point out that the CHAM algorithm may be vulnerable to the fault injection attack which can reveal 4 round keys and derive the secret key from them. As a simulation result, the proposed fault injection attack can extract the secret key of CHAM-128/128 block cipher using about 24 correct-faulty cipher text pairs.

Keywords: Lightweight block cipher, CHAM, Fault injection attack

I. 서 론

최근에는 다양한 형태의 사물들이 네트워크 형태로 연결되어 센서 등에 의해 수집된 정보를 주고받는 사물 인터넷 기술이 각광을 받고 있다. 이러한 네트워크 환경에서 송·수신되는 정보에 대한 기밀성 확보 방법 중 하나가 AES[1]와 같은 블록 암호 기법을 이용하는 것이다. 특히, 사물 인터넷 통신에서는 암호화를 수행하는 디바이스의 자원이 제한된 경우가 많은데 이러한 환경을 고려하여 LEA[2, 3], HIGHT[4] 그리고 SIMON/SPECK[5]과 같은 경량 블록 암호가 제안되기도 하였다. 이러한 경량 암호 알고리듬들은 일정한 수준의 안전도를 유지하면서도 구현의 효율성을 높이기 위해 키 길이, 라운드 수, 처리 데이터 단위를 다양하게 설계되었다.

국내·외적으로 많은 경량 암호 알고리듬들이 제안된 가운데, 최근 ICISC'17에서는 CHAM[6, 7]이라는 초경량 블록 암호 알고리듬이 제안되었다. 이 암호 알고리듬에서는 키 상태를 갱신하지 않으면서 라운드 키를 생성하는 키 스케줄링 기법을 사용함으로써 키 저장 공간을 획기적으로 줄였으며 ARX(Addition,

Received(08. 06. 2018), Accepted(08. 30. 2018)
† 주저자, khp9890@gmail.com
‡ 교신저자, jcha@hoseo.edu(Corresponding author)

Rotation, and XOR)에 기반한 간단한 연산자를 사용하도록 설계함으로써 계산 성능을 향상시켰다.

한편, 사물 인터넷에서 사용하는 디바이스가 저사양, 저성능인 경우가 많아 실제 개발 단계에서 발생할 수 있는 취약점을 고려하지 않을 경우, 구현 공격(implementation attack)[8-12]이라고 하는 물리적 공격에 의해 비밀 키가 노출될 가능성이 있다. 일반적으로 구현 공격은 수동적인 공격과 능동적인 공격으로 구분되는데 수동적인 공격을 부채널 분석(side channel analysis) 공격이라고도 한다. 능동적인 공격은 주로 암호용 디바이스가 동작하는 과정에 의도된 오류를 주입한 후 잘못된 출력을 바탕으로 비밀 키를 분석하는 오류 주입 공격(fault injection attack)이 대표적이다[11-12].

실제로 오류 주입 공격은 AES 및 RSA와 같은 표준 블록 암호 혹은 상용 공개 키 암호 시스템 등에 쉽게 적용 가능하며 경량 암호 알고리듬인 LEA나 SIMON/SPECK에 대한 공격도 시도된 바 있다. 실제 분석 결과, LEA는 300개의 선택적 오류 주입 암호문을 이용하여 2^{35}의 시간 복잡도로 128비트 마스터 비밀 키 전체를 추출할 수 있음이 증명되었다[13]. SIMON/SPECK에서는 비트 혹은 바이트 단위의 오류 주입을 했을 경우 마지막 라운드 키를 추출할 수 있음이 최근 밝혀졌다[14].

본 논문에서는 경량 블록 암호 알고리듬 CHAM이 오류 주입 공격에 대해 어떠한 취약성이 있는지 그리고 실제 라운드 키가 노출될 가능성은 있는지를 분석하고 그 공격 모델을 제안하고자 한다. CHAM-128/128에 대한 제안된 오류 주입 공격 방법을 이용하여 공격자가 약 24개의 오류 암호문을 획득하면 사용된 라운드 키를 찾을 수 있으며, 추출한 4개의 라운드 키로부터 마스터 비밀 키를 복구할 수 있다. 논문에서는 CHAM에 대한 랜덤 워드에 기반한 오류 주입 공격 모델이 논리적으로 타당함을 컴퓨터 시뮬레이션을 통해 확인하였다.

II. CHAM 경량 암호 알고리즘

CHAM은 2017년 국내 연구진에 의해 제안된 경량 블록 암호 알고리듬으로서 자원이 제한된 디바이스에 효과적으로 사용할 수 있도록 개발되었다[6]. 이 암호 알고리듬은 4개의 가지(branch)를 가지는 Feistel 구조이며 ARX 동작에 기반하고 있다. CHAM에서 사용하는 블록은 64비트와 128비트가 있으며 키 길이는 128비트와 256비트를 사용할 수 있도록 CHAM-64/128, CHAM-128/128 그리고 CHAM 128/256이 제안되었다. CHAM은 사용하는 블록과 비밀 키 길이에 따라 라운드 수와 처리하는 워드의 길이가 다르다.

본 논문에서는 CHAM-128/128을 중심으로 설명할 것인데 이 경우 블록 길이 $n=128$, 키 길이 $k=128$, 라운드 수 $r=80$, 워드의 길이는 $w=32$비트 그리고 $k/w=4$이다.

CHAM-n/k에서 라운드 키는 w비트로 구성되며 총 $2k/w$개의 라운드 키를 생성하여 사용한다. 즉, CHAM-128/128에서는 8개의 라운드 키가 사용된다. 비밀 키는 $w=32$비트로 구성된 4개의 워드로 연접되어 있으며 각 워드로부터 라운드 키를 생성하는 과정을 나타낸 것이 Fig. 1이다. 여기서 ROL_i는 32비트의 워드가 좌측으로 i비트 회전 이동하는 것을 의미한다.

라운드 키를 이용한 암호화 과정을 나타낸 것이 Fig. 2이다. 128비트 평문은 4개의 워드로 나누어 입력되어 각 라운드마다 ARX 연산이 수행되며 레지스터 값을 변경하게 된다. 다음은 라운드 i가 짝수일 때의 암호화 과정을 나타낸 것이다.

$$X_{i+1}[3]=ROL_8((X_i[0]\oplus i)+(ROL_1(X_i[1])\oplus RK[i \bmod 2k/w]))$$
$$X_{i+1}[j]=X_i[j+1] \text{ for } 0\le j\le 2$$

여기서 +는 두 입력을 더한 후 mod 2^w 연산을 수행함을 의미한다.

라운드 i가 홀수일 때는 짝수일 때와 비교하여 좌측 순환하는 비트 수만 다르게 암호화를 수행한다.

$$X_{i+1}[3]=ROL_1((X_i[0]\oplus i)+(ROL_8(X_i[1])\oplus RK[i \bmod 2k/w]))$$

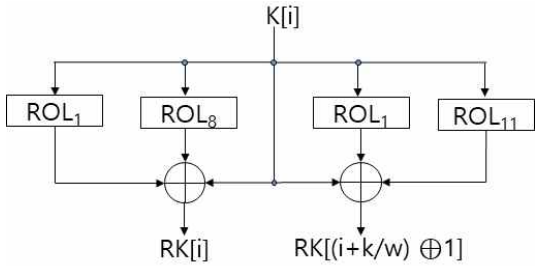

Fig. 1. Key schedule functions

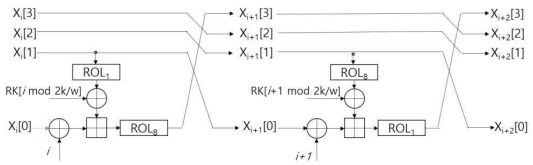

Fig. 2. Two round functions beginning with the even i-th round

$$X_{i+1}[j] = X_i[j+1] \text{ for } 0 \le j \le 2$$

여기서 $RK[i \bmod 2k/w]$는 각 라운드 키를 나타낸다. 본 논문에서 설명하는 CHAM-128/128을 기준으로 할 때 8개의 라운드 키는 80라운드를 수행하는 동안 10번 반복해서 사용된다.

III. CHAM에 대한 오류 주입 공격 모델

3.1 오류 주입 공격

본 절에서는 CHAM에 대한 오류 주입 공격 모델을 제안하고자 한다. 오류 주입 공격에서는 오류가 주입된 출력을 분석하여 비밀 키를 바로 분석하는 경우도 있지만 중간 결과 값을 추출한 후 최종적으로 라운드 키나 비밀 키를 추출하기도 한다.

본 논문에서 제안하는 오류 주입 공격은 정상 암호문과 오류 암호문을 차분하는 방식을 이용하여 중간 결과 값을 추출한 후 라운드 키를 계산하는 차분 오류 분석(Differential Fault Analysis, DFA) 기법이다. 제안하는 오류 주입 공격 모델에서 가장 중요한 부분은 차분 오류를 구하기 위해 오류를 주입하는 위치와 오류 단위이다. CHAM에 대한 오류 공격 모델을 나타낸 것이 Fig. 3인데 여기서 오류 위치는 암호 알고리듬의 마지막 라운드이며 오류 단위는 랜덤한 워드 오류(single random word fault)가 주입되었음을 가정한다.

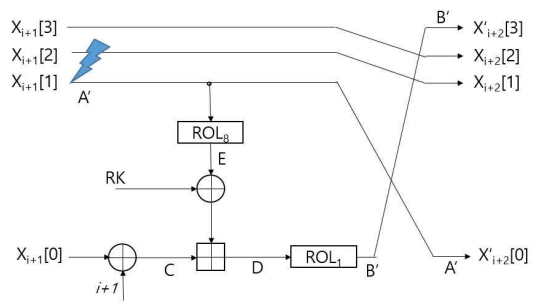

Fig. 3. Fault Injection attack model on CHAM

먼저 특정한 평문을 입력으로 하는 정상 암호문을 저장한다. 이때 설명의 편의를 위해 최종 출력 값 $X_{i+2}[0]$를 간단히 A라 표기하였고 $X_{i+2}[3]$를 B라 나타내었다. 또한, 동일한 평문을 입력으로 하고 마지막 라운드에서 오류가 주입된 후 그 출력을 저장한다. 이때 오류는 그림에서 보는 바와 같이 이전 라운드의 레지스터 $X_{i+1}[1]$값에 주입한다. 이 오류 값은 다음 라운드의 $X'_{i+2}[0]$ 값, 즉 A'값이 된다. 이 오류가 주입되면 $X'_{i+2}[3]$ 값을 출력하는데 이를 B'이라 하자. 즉, 오류 값 A'이 주입되면 B' 값을 저장하게 되는데 이를 정리하면 다음과 같다.

$$\begin{aligned} B &= ROL_1(ROL_8(A) \oplus RK + C) \\ D &= ROR_1(B) = ROL_8(A) \oplus RK + C \\ &= E \oplus RK + C \\ D - C &= E \oplus RK \end{aligned} \quad (1)$$

여기서 $D = ROR_1(B)$이고, $E = ROL_8(A)$이며 ROR_i는 32비트의 워드를 우측으로 i비트 회전 이동한 것을 의미한다. 또한, C값은 $X_{i+1}[0] \oplus (i+1)$를 간단히 표시한 것이다.

오류가 주입된 후 얻은 출력으로부터 다음과 같은 결과를 얻을 수 있다.

$$\begin{aligned} B' &= ROL_1(ROL_8(A') \oplus RK + C) \\ D' &= ROR_1(B') = ROL_8(A') \oplus RK + C \\ &= E' \oplus RK + C \\ D' - C &= E' \oplus RK \end{aligned} \quad (2)$$

이 경우에도 마찬가지로 $D' = ROR_1(B')$이며 $E' = ROL_8(A')$이다.

따라서 정상 암호문과 오류 암호문을 얻은 공격자는 (1)식과 (2)식을 차분하면 다음과 같은 공격 모델을 얻을 수 있다.

$$K = E \oplus E' = (D - C) \oplus (D' - C) \quad (3)$$

결국, 공격자가 E, E', D 그리고 D'값을 안다면, 이 값들을 통해 중간 암호문 C를 추출할 수 있다.

만약, 공격자가 C를 알게 된다면 다음과 같이 사

용된 마지막 라운드 키 RK를 계산할 수 있다. 즉, 중간 암호문 C를 찾는 것과 라운드 키를 찾는 것은 같은 의미가 된다.

$$RK = E \oplus (D - C)$$
$$= ROL_8(A) \oplus (ROR_1(B) - C)$$

따라서 공격자는 식 (3)을 이용하여 중간 암호문 C를 추출해야 하는데 이 문제는 아래와 같은 등식이 성립하는 가운데 K, D 그리고 D'을 알고 있는 상태에서 C값을 계산하는 것과 같다.

본 논문에서는 식 (3)를 통해 가능한 중간 암호문 C를 추출할 수 있음을 예를 통해 증명하고자 한다. 예를 들어 $D = 0011$, $D' = 1000$ 그리고 $K = 0101$이었다고 가정하고 가능한 C값을 최하위 비트(Least Significant Bit, LSB)부터 순차적으로 추측해 보자. 이를 단계적으로 설명한 것이 Fig. 4이다.

- 1단계 : 먼저 식 (3)에서 C의 최하위 비트인 C_0가 0이거나 1이거나 관계없이 K_0가 1이 될 수 있으므로 가능한 C_0의 후보는 2개가 된다.
- 2단계 : 그 후 가능한 두 후보 중에서 다음 상위 비트인 C_1이 0인지 1인지에 따라 가능한 K_1K_0를 구하고 이 값이 이미 알고 있는 01인지 확인한 후 중간 암호 값 C_1C_0를 선택한다. 예제에서 보면 C_1C_0값이 00이나 10인 경우에는 K_1K_0이 11이 도출되므로 중간 값 후보에서 제외된다.
- 3단계 : 위에서 구한 C_1C_0값이 01이나 11인 경우에는 C_2가 0일 때나 1일 때 모두 식 (3)을 만족하므로 후보 암호문이 4개가 된다.
- 4단계 : 4가지의 $C_2C_1C_0$ 후보 값이 있고 C_3가 0일 때나 1일 때와 같이 4비트의 암호화 과정에서는 중간 암호문의 후보가 4개로 줄어들게 된다.

이와 같은 과정을 반복함으로써 식 (3)을 만족하는 중간 암호문 C값을 최하위 비트부터 순차적으로 탐색하게 되며 위 예제에서는 16개의 후보(0000~1111) 중에서 4개의 C만(0001, 1001, 0011, 1011) 등식을 만족하는 값임을 알 수 있다.

그러나 32비트 정도의 정상 암호문 A로부터 유추된 E과 오류 암호문 A'로부터 유추된 E'에 대한 중간 암호문 C는 수백~수천 개의 후보 값을 가지게 된다. 결국, 한 쌍의 정상 암호문과 오류 암호문을 가지고 있는 공격자는 라운드 키 후보를 2^{32}개에서 수백~수천 개까지 줄일 수 있다. 그러므로 여러 개의 정상-오류 암호문 쌍을 획득하여 식 (3)을 만족하는 중간 암호문의 후보를 더 줄일 수 있으며 최종적으로 마지막 라운드 키 $RK[7]$ 값을 계산할 수 있다.

Step	C	K	Candi-date
1	$C_0 = 0$	$K_0 = 1$	O
	$C_0 = 1$	$K_0 = 1$	O
2	$C_1C_0 = 00$	$K_1K_0 = 11$	X
	$C_1C_0 = 10$	$K_1K_0 = 11$	X
	$C_1C_0 = 01$	$K_1K_0 = 01$	O
	$C_1C_0 = 11$	$K_1K_0 = 01$	O
3	$C_2C_1C_0 = 001$	$K_2K_1K_0 = 101$	O
	$C_2C_1C_0 = 101$	$K_2K_1K_0 = 101$	O
	$C_2C_1C_0 = 011$	$K_2K_1K_0 = 101$	O
	$C_2C_1C_0 = 111$	$K_2K_1K_0 = 101$	O
4	$C_3C_2C_1C_0 = 0001$	$K_3K_2K_1K_0 = 0101$	O
	$C_3C_2C_1C_0 = 1001$	$K_3K_2K_1K_0 = 0101$	O
	$C_3C_2C_1C_0 = 0101$	$K_3K_2K_1K_0 = 1101$	X
	$C_3C_2C_1C_0 = 1101$	$K_3K_2K_1K_0 = 1101$	X
	$C_3C_2C_1C_0 = 0011$	$K_3K_2K_1K_0 = 0101$	O
	$C_3C_2C_1C_0 = 1011$	$K_3K_2K_1K_0 = 0101$	O
	$C_3C_2C_1C_0 = 0111$	$K_3K_2K_1K_0 = 1101$	X
	$C_3C_2C_1C_0 = 1111$	$K_3K_2K_1K_0 = 1101$	X

Fig. 4. Candidates of intermediate cipher value

마지막 라운드 키를 추출하면 이전 라운드의 출력값도 계산할 수 있으므로 동일한 원리를 사용하여 이전 라운드 키 값을 추출할 수 있다. 즉, 이전 라운드의 레지스터 $X_i[1]$값에 오류를 여러 번 주입하여 그 결과 값을 얻으면 중간 암호문 후보군을 줄일 수 있으며 결국 라운드 키 $RK[6]$을 계산할 수 있다. 최종적으로는 위 과정을 반복함으로써 마지막 라운드 키 $RK[7]$부터 역순으로 모든 라운드 키들을 추출할 수 있다.

3.2 라운드 키를 이용한 통한 비밀 키 계산

본 절에서는 라운드 키를 알고 있는 공격자가 마스터 비밀 키를 추출하는 과정을 살펴보고자 한다. 라운드 키 생성 과정을 수식으로 정리하면 아래와 같은데 하나의 비밀 키 워드로부터 두 개의 라운드 키를 계산한다.

$$RK[i] = K[i] \oplus ROL_1(K[i]) \oplus ROL_8(K[i])$$
$$RK[(i+k/w) \oplus 1] = K[i] \oplus ROL_1(K[i]) \oplus ROL_{11}(K[i])$$

따라서 하나의 라운드 키를 알 수 있으면 2^{32}번의 전탐색 공격을 통해 비밀 키 워드 $K[i]$를 구할 수도 있다.

본 논문에서는 전탐색 기법을 사용하지 않고 라운드 키로부터 마스터 비밀 키를 구하는 방법을 제안하고자 한다. 상기한 Fig. 1에서 보는 바와 같이 i번째 라운드에서 오른쪽 라운드 키($RK[4] \sim RK[7]$)는 각 비밀 키 워드를 왼쪽으로 1번 그리고 11번 회전 이동 후 원래의 비밀 키와 XOR 연산을 수행한 것이다.

$$RK = K \oplus ROL_1(K) \oplus ROL_{11}(K)$$

이를 다시 비밀 키 K의 각 비트로 표현하면 다음과 같이 정리할 수 있다.

$$K_i = RK_i \oplus K_{(i-1+w) \bmod w} \oplus K_{(i-11+w) \bmod w} \quad (0 \le i < w)$$

Fig. 5는 하나의 라운드 키로부터 마스터 비밀 키를 계산하는 과정을 그림으로 도시한 것이다. 이 키 복구 과정에서는 먼저 $K_0 \sim K_{10}$ 값을 가정한 후 $K_{11} \sim K_{31}$을 계산한다. 그 후 $K_{11} \sim K_{31}$을 이용하여 $K'_0 \sim K'_{10}$을 유추한다. 그리고 처음에 가정한 $K_0 \sim K_{10}$과 $K'_0 \sim K'_{10}$이 같은 값인지 확인한 후 같으면 $K_0 \sim K_{31}$을 비밀 키로 확정한다. 만약 같지 않다면 새로운 $K_0 \sim K_{10}$값을 가정하고 위 과정을 반복함으로써 비밀 키를 검증하게 된다. 따라서 $K_0 \sim K_{10}$값을 가정하는 2^{11}의 복잡도로 하나의 비밀 키 워드를 찾을 수 있다.

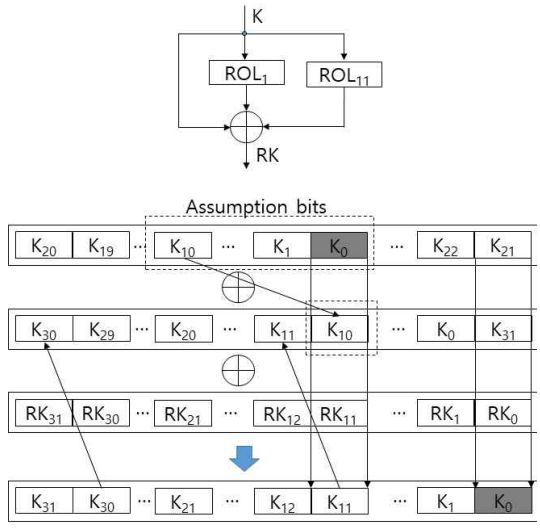

Fig. 5. Secret key recovery from round key

또한, i번째 라운드에서 왼쪽 라운드 키($RK[0] \sim RK[3]$)는 각 비밀 키 워드를 왼쪽으로 1번 그리고 8번 회전 이동 후 원 비밀 키와 XOR 연산을 수행한 것이다. 따라서 제안 방법을 동일한 원리로 적용하면 하나의 라운드 키를 획득한 공격자는 2^8의 복잡도로 하나의 비밀 키 워드를 추출할 수 있다.

결론적으로 CHAM-128/128에서 오류 주입 공격에 의해 $RK[4] \sim RK[7]$까지 4개의 라운드 키를 추출한 공격자는 전탐색 공격을 하지 않더라도 4×2^{11}번의 비밀 키 복구 연산을 통해 128비트 마스터 비밀 키를 모두 추출할 수 있다.

IV. 오류 주입 공격 시뮬레이션

CHAM에 대한 오류 주입 공격에서는 오류 주입을 통해 얻은 정상 암호문과 오류 암호문의 차분을 통해 마지막 라운드의 중간 암호문을 구하고 마지막 라운드 키를 계산한다. 그 후 반복 과정을 거쳐 역순으로 4개의 라운드 키를 계산한 후 이를 기반으로 마스터 비밀 키를 계산할 수 있다.

오류 주입 공격을 타당성을 검증하기 위해 먼저 CHAM을 C언어로 구현하고 정상 암호문과 오류 주입을 가정한 오류 암호문을 수집하여 분석함으로써 비밀 키를 찾아낼 수 있음을 시뮬레이션하였다. 처음 시뮬레이션에서는 몇 개의 정상-오류 암호문 쌍을 가지고 하나의 라운드 키를 찾을 수 있는지 실험하였다. Table 1은 오류 주입 공격에 필요한 정상-오류

Table 1. Number of round key candidates according to correct-faulty cipher text pairs

Correct-Faulty Pairs	2^1	2^2	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
1	0	0	0	0	0	0	0	1	3	7	5	12	8	64
2	1	1	2	3	14	16	21	18	10	10	3	1	0	0
3	3	12	18	25	23	12	5	1	1	0	0	0	0	0
4	20	31	25	16	7	1	0	0	0	0	0	0	0	0
5	45	38	14	3	0	0	0	0	0	0	0	0	0	0
6	74	21	4	1	0	0	0	0	0	0	0	0	0	0

암호문 쌍 수와 가능한 라운드 키의 개수를 나타낸 것이다.

이 시뮬레이션은 총 100번 수행하였는데 하나의 정상-오류 암호문 쌍을 이용할 경우 2^9개 이상의 중간 암호문 후보(라운드 키 후보와 같은 의미로 볼 수 있음)를 얻을 수 있었으며, 어떤 정상-오류 암호문 쌍을 이용할 경우에는 가능한 중간 암호문 C의 개수가 2^{15}개 이상 검출되기도 하였다. 예를 들면, 표에서 1개의 정상-오류문을 가지는 경우, 키 후보가 2^9개를 가지는 경우도 있으며 많게는 2^{15}개를 가지는 경우가 있었다. 총 100번의 시뮬레이션 데이터 중에서는 2^9개의 암호문 후보를 가지는 경우는 1번, 2^{10}개의 후보를 가지는 경우는 3번, 그리고 2^{15}개의 후보를 가지는 경우가 64번이 있었다.

또한, 표에서 보는 바와 같이 2개의 정상-오류 암호문 쌍을 이용하여 병렬로 후보 값을 탐색한 결과, 2^8개의 중간 암호문 후보를 얻는 경우가 가장 많았다. 동일한 방법으로 4개의 정상-암호문 쌍을 이용하면 중간 암호문의 범위를 약 4개까지 축소시킬 수 있었다.

최종적으로 50%이상의 확률로 2개의 중간 암호문(라운드 키)을 얻기 위해서는 6개의 정상-오류 암호문 쌍이 필요하였다. 마지막 라운드 키 후보는 최소 2개까지 줄일 수 있는데 그 이유는 암호화 연산 시 각 라운드에서 mod 2^w에 대한 모듈러 덧셈 시 캐리(carry)가 발생해서 무시한 경우와 캐리가 발생하지 않는 경우 즉, 2가지 형태의 연산이 존재하기 때문이다.

Fig. 6은 100번의 시뮬레이션을 했을 경우 1개의 정상-오류 암호문 쌍과 6개의 정상-오류 암호문

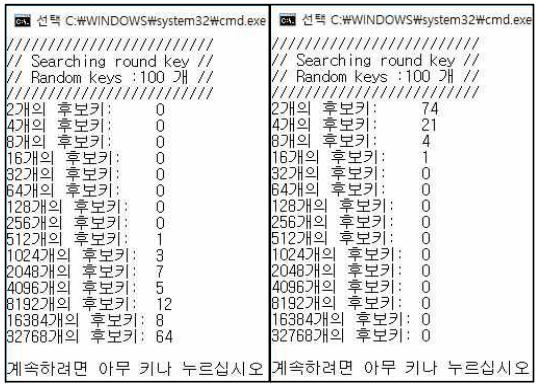

Fig. 6 The number of round key candidates using one or six correct-faulty cipher text pair(s)

쌍을 이용하여 추출되는 라운드 키의 수를 나타낸 시뮬레이션 화면이다. 그림에서 보는 바와 같이 6개의 정상-오류 암호문 쌍을 이용하는 경우, 두 개의 라운드 키를 가지는 경우가 74번, 4개의 라운드 키를 가지는 경우가 21번, 8개가 4번, 그리고 16개의 라운드 키를 추출하는 경우도 1번이 발생함을 볼 수 있다.

시뮬레이션에서 보는 바와 같이 각 라운드마다 6개의 오류 암호문을 이용하면 2가지 종류의 라운드 키가 발생하게 됨으로써 4개의 라운드 키를 찾기 위해서는 모두 24(6개씩 4라운드)개의 정상-오류 암호문 쌍이 필요하며 2^4개의 라운드 키 후보를 얻을 수 있게 된다. 결론적으로 CHAM-128/128에서는 하나의 정상 암호문과 약 24개의 오류 암호문을 이용하여 2^4의 복잡도로 4개의 라운드 키를 찾을 수 있으며 이 라운드 키로부터 비밀 키를 완전히 복구할 수 있다.

한편, 중간 암호문 후보 값을 한 라운드에 4개까지 허용한다면 정상-오류 암호문쌍이 4개 정도만 있어도 50% 이상은 4개의 후보 암호문을 찾을 수 있다. 이 경우 16(4개씩 4라운드)개의 정상-오류 암호문 쌍을 이용하여 4^4번의 라운드 키 확인 연산을 통해 4개의 라운드 키를 찾을 수 있다.

제안한 공격 기법을 CHAM-128/256에 동일한 원리로 적용할 수 있는데, 약 48(6개씩 8라운드)개의 정상-오류 암호문 쌍과 2^8의 복잡도로 8개의 라운드 키를 찾을 수 있다. 또한, 워드의 길이가 16비트인 CHAM-64/128을 공격할 경우에는 하나의 라운드 키를 추출하는데 5개를 이용하면 50%이상의

확률로 라운드 키를 2개까지 추출할 수 있었다. 따라서 약 40(5개씩 8라운드)개의 오류 암호문과 2^8의 복잡도로 8개의 라운드 키를 찾을 수 있다.

V. 결 론

사물 인터넷과 같은 통신 환경에서 저사양의 디바이스에 기밀성을 제공하기 위해 경량 암호 알고리듬들이 제안되고 있다. CHAM은 최근에 제안된 ARX 기반의 초경량 암호 알고리듬으로서 사용 메모리가 적으며 구현 성능도 우수하여 이 암호 알고리듬의 구현과 활용에 대한 관심이 증대되고 있다. 그러나 CHAM 암호 알고리듬도 부채널 공격이나 오류 주입 공격과 같은 구현 공격에 취약한 특성이 있는지 알고리듬 개발 단계에서 점검해 보아야 한다.

본 논문에서는 경량 암호 알고리듬 CHAM에 대한 오류 주입 공격에 대한 취약성과 공격 모델을 찾을 수 있었다. CHAM은 6개 정도의 정상-오류 암호문 쌍을 가지면 충분히 라운드 키를 찾아낼 수 있으며, 약 2^4번의 검증 연산을 통해 4개의 라운드 키를 모두 찾을 수 있었다. 또한, 라운드 키를 알고 있을 경우 간단한 회전 이동 분석 공격을 통해 한 워드의 비밀 키를 찾을 수 있고 4개의 라운드 키로 마스터 비밀 키 전체를 찾을 수 있음을 컴퓨터 시뮬레이션을 통해 확인하였다.

References

[1] National Institute of Standards and Technology, "Advanced Encryption Standards," NIST FIPS PUB 197, 2001.

[2] D. Hong, J. Lee, D. Kim, D. Kwon, K. Ryu, and D. Lee, "LEA, A 128-bit block cipher for fast encryption on common processors," WISA'13, LNCS 8267, pp. 3-27, 2014.

[3] TTA, "128-bit lightweight block cipher LEA," TTAK.KO-12.0223, Dec, 2013.

[4] D. Hong, J. Sung, S. Hong, J. Lim, S.Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee, "HIGHT: A new block cipher suitable for low-resource device," CHES'06, LNCS 4249, pp. 46-59, 2006.

[5] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK lightweight block ciphers," In Design Automation Conference(DAC'15), pp. 1-6, 2015.

[6] B. Koo, D. Roh, H. Kim, Y. Jung, D. Lee, and D. Kwon, "CHAM: A Family of lightweight block ciphers for resource-constrained devices," ICISC'17, LNCS 10779, pp. 3-25, 2017.

[7] H. Seo, "Memory-efficient imple- mentation of ultra-lightweight block cipher algorithm CHAM on low-end 8-bit AVR processors," Journal of The Korea Institute of Information Security & Cryptology(JKIISC), Vol. 28, No. 3, pp. 545-550, 2018.

[8] P. Kocher, "Timing Attacks on Implementation of Diffie-Hellman, RSA, DSS, and Other Systems," CRYPTO'96, LNCS 1109, pp. 104-113, 1996.

[9] J. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," CHES'99, LNCS 1717, pp. 292-302, 1999.

[10] T. Messerges, E. Dabbis, and R. Sloan, "Power analysis attacks of modular exponentiation in smartcard," CHES'99, LNCS 1717, pp. 144-157, 1999.

[11] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," CRYPTO'97. LNCS 1294, pp. 513 - 525, 1997.

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

[13] M. Park, J. Kim, "Differential Fault Analysis of the Block Cipher LEA", Journal of The Korea Institute of Information Security & Cryptology (JKIISC), Vol. 24 No. 6, pp. 1117-1126, 2014.

[14] H. Tupsamudre, S. Bisht, and D. Mukhopadhyay, "Differential Fault Analysis on the Families of SIMON and SPECK Ciphers," In Workshop on Fault Diagnosis and Tolerance in Cryptography(FDTC'14), pp. 40-48, 2014.

〈저자소개〉

권 홍 필(Hongpil Kwon) 학생회원
2018년 2월: 호서대학교 정보보호학전공 학사
2018년 3월: 호서대학교 정보보호학과 석사 과정
〈관심분야〉 암호학, 부채널 공격

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
2007년 3월~현재: 호서대학교 컴퓨터정보공학부 교수
2013년 1월~현재: 한국정보보호학회 상임부회장
2009년 1월~현재: 한국산학기술학회 이사
〈관심분야〉 암호학, 네트워크 보안, 부채널 공격,