Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.29, NO.4, Aug. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.4.709

부채널 공격에 대응하는 경량 블록 암호 CHAM 구현을 위한 마스킹 기법 적용 및 분석*

권 홍 필,† 하 재 철‡
호서대학교

Application and Analysis of Masking Method to Implement Secure Lightweight Block Cipher CHAM Against Side-Channel Attack Attacks*

Hongpil Kwon,† Jaecheol Ha‡
Hoseo University

요 약

CHAM은 자원이 제한된 환경에 적합하도록 설계된 경량 블록 암호 알고리즘으로서 안전성과 연산 성능면에서 우수한 특성을 보인다. 그러나 이 알고리즘도 부채널 공격에 대한 취약성을 그대로 내재하고 있기 때문에 마스킹 기법과 같은 대응 기법이 적용되어야 한다. 본 논문에서는 32비트 마이크로프로세서 Cortex-M3 플랫폼에서 부채널 공격에 대응하는 마스킹 기법이 적용된 CHAM 알고리즘을 구현하고 성능을 비교 분석한다. 또한, CHAM 알고리즘이 라운드 수가 많아 연산 효율이 감소되는 점을 고려하여 축소 마스킹 기법을 적용하여 성능을 평가하였다. 축소 라운드 마스킹이 적용된 CHAM-128/128은 구현 결과 마스킹이 없는 경우에 비해 약 4배 정도의 추가 연산이 필요함을 확인하였다.

ABSTRACT

A lightweight block cipher CHAM designed for suitability in resource-constrained environment has reasonable security level and high computational performance. Since this cipher may contain intrinsic weakness on side channel attack, it should adopt a countermeasure such as masking method. In this paper, we implement the masked CHAM cipher on 32-bit microprosessor Cortex-M3 platform to resist against side channel attack and analyze their computational performance. Based on the shortcoming of having many round functions, we apply reduced masking method to the implementation of CHAM cipher. As a result, we show that the CHAM-128/128 algorithm applied reduced masking technique requires additional operations about four times.

Keywords: Physical attack, Symmetric cipher CHAM, ARX cipher, Masking method

I. 서 론

최근 IT(Information Technology) 기술이 발전함에 따라 스마트 폰, 사물 인터넷 장치, 센서 단말 장치와 같은 디바이스의 고속화 및 경량화에 대한 수요가 증가하고 있다. 이러한 초연결 네트워크 환경에서 사용되는 디바이스 장치들은 센싱 데이터와 같은 개인의 민감한 정보를 생성하고, 이 데이터를

Received(03. 11. 2019), Modified(05. 13. 2019), Accepted(05. 14. 2019)
* 이 논문은 2018년도 호서대학교의 재원으로 학술연구비 지원을 받아 수행된 연구임.(2018-0364)

* 본 논문은 2018년 동계학술대회에서 발표한 우수 논문을 개선 및 확장한 것임.

† 주저자, khp9890@gmail.com
‡ 교신저자,jcha@hoseo.edu(Corresponding author)

다른 소형 디바이스나 서버와 교환하게 된다. 이와 같은 단말 디바이스들은 민감한 데이터를 생성 및 교환하기 때문에 전송 패킷에 대한 암호화가 필연적으로 요구된다. 하지만 소형 디바이스는 전력, 메모리 등의 물리적 가용자원이 제한되어 있어 기존에 일반적인 암호 알고리즘을 사용한 경우에는 연산 속도가 느리고, 필요한 데이터 저장 공간이 부족하여 효율성과 가용성이 떨어진다. 따라서 소형 디바이스의 환경에서도 고속 암호가 가능한 경량 암호 알고리즘 설계가 필요하게 되었고 이미 외국에서는 PRESENT[1], CLEFIA[2], SIMON/SPECK[3]와 같은 경량 블록 암호 알고리즘을 개발하여 표준화 작업을 진행하고 있다. 국내에서도 독자적인 암호 기술로 LEA[4, 5], HIGHT[6] 등의 경량 블록 암호 알고리즘을 개발하였으며 2017년 말에 CHAM[7, 8]이라는 초경량 암호 알고리즘을 개발한 바 있다.

하지만 이러한 경량 블록 암호들은 저전력, 저사양의 제한된 구현 환경하에서 개발되기 때문에 물리적인 요소를 통해 얻은 정보를 분석하여 공격하는 부채널 분석 공격(side-channel attack)[9]에 취약한 특성을 보인다. 이러한 부채널 분석 공격은 암호용 디바이스 상에서 암호 라운드 연산이 진행되어지는 암호 중간 값들을 추측하고, 이 추측 값들이 정확한지를 부채널 정보를 이용해서 분석하고 이를 통해 실제 비밀 키 값을 알아내는 기법이다. 따라서 이러한 부채널 공격에 대응하기 위한 기본 원리는 중간 값을 추측할 수 없도록 중간 암호 값에 마스킹(masking)을 적용하는 것이다. 즉, 마스킹 대응 기법은 올바른 암호 중간 값에 랜덤한 마스크 값을 추가 연산함으로써 공격자로 하여금 올바른 값을 추측할 수 없게 하는 것이다. 그러나 실제 중간 값에 대한 마스킹을 적용하면 암호화 연산 시간보다 많은 시간이 소요될 수 밖에 없으며 이는 경량 암호 알고리즘 개발의 효용성을 크게 저하시키는 요소가 될 수 있다

본 논문에서는 최근 국내에서 개발된 초경량 블록 암호 CHAM에 대해 마스킹 기법을 적용하기 위한 설계 기법을 제시한다. 또한, 여러 가지 마스킹 기법들을 CHAM 암호에 적용하여 32비트 마이크로프로세서인 Cortex-M3 보드에 구현하고 CHAM 암호에 적용된 마스킹 기법간의 효율성을 비교 분석한다. 마지막으로 마스킹을 적용한 CHAM 블록 암호 알고리즘의 연산량 증가 원인을 분석하고 전체 연산 속도를 줄일 수 있는 축소 라운드 마스킹 기법을 적용하여 그 효율성을 알아본다.

본 논문의 2장에서는 데이터 마스킹 기법에 대한 설명을 하고 산술-부울 상호 마스킹 변환 기법과 산술-부울 상호 마스킹 변환 알고리즘들을 설명한다. 3장에서는 산술-부울 상호 마스킹 변환의 번거로움을 해소하기 위해 개발된 SA(Secure Addition) 마스킹 기법을 설명한다. 4장에서는 경량 블록 암호 CHAM에 마스킹 기법을 어떻게 적용하여 설계할지를 제시한다. 5장에서는 마스킹 기법이 적용된 CHAM 블록 암호(CHAM-128/128)를 32비트 마이크로프로세서인 Cortex-M3 보드에 구현하여 축소 마스킹을 적용하였을 때 마스킹 기법별로 효율성을 분석하고, 마지막으로 결론을 맺는다.

II. 데이터 마스킹 기법

마스킹 기법은 블록 암호 라운드 연산이 진행되는 동안 발생하는 중간 값(x)에 랜덤한 값인 마스크 값(r_x)을 추가 연산함으로서 공격자가 올바른 중간 값을 추출할 수 없게 하는 부채널 공격에 대한 대응 기법이다. 이러한 마스킹 기법은 일반적으로 적용되어지는 암호 라운드 내의 연산 종류에 따라 부울 마스킹과 산술 마스킹으로 나누어 적용된다. 입력 x에 대한 부울 마스킹과 산술 마스킹은 다음과 같이 정의된다.

- 부울 마스킹 : $x' = x \oplus r_x$
- 산술 마스킹 : $A = x - r_x \bmod 2^m$

여기서 x는 마스킹이 적용되기 전의 원래의 중간 암호 값이고, r_x는 랜덤 수로서 마스킹을 위해 사용되는 값이다. 또한, m은 산술 마스킹 연산에 사용되는 입력 데이터의 비트 길이를 나타낸다.

대부분의 블록 암호의 라운드 연산에는 부울 연산과 산술 연산이 혼용되어 사용하기 때문에 마스킹 기법 또한 연산에 따라 상호 변환해 주는 과정이 필요하다. 서로 다른 연산에 대한 마스킹 변환 과정이 필요한 이유는 변환 과정을 거치지 않으면 원래의 암호 값과 마스크 값이 섞여 최종적으로 제대로 된 암호 연산 값을 얻을 수 없기 때문이다. 단, 변환 과정에서 주의할 점은 변환되는 과정에서 원래의 값 x 또는 마스크 값 r이 노출되면 안 된다는 것이다. 변환 알고리즘이 실행되는 가운데 x 또는 r값이 노출되면

마스킹을 적용하지 않았을 때와 동일하게 부채널 공격에 노출될 수 있기 때문이다.

2.1 부울-산술 마스킹 변환 기법

부울-산술 마스킹 변환 기법은 부울 마스킹 된 중간 값을 산술 연산을 위해 산술 마스킹 된 중간 값으로 변환하는 기법이다. 지금까지 부울-산술 마스킹 변환 기법 가운데 Goubin이 제안한 부울-산술 마스킹 변환 알고리즘의 연산량이 매우 적고 효율적이기 때문에 많은 암호 알고리즘에서 사용되어지고 있다[10]. 다음 Fig. 1은 Goubin에 의해 제안된 부울-산술 마스킹 변환 기법으로 부울 마스킹 된 값 $x' = x \oplus r$과 난수 r을 입력받아 연산을 통해 산술 마스킹 된 $A = x - r$값을 출력한다. 본 논문에서는 CHAM 암호에 대한 부울-산술 마스킹 변환 기법으로 Goubin이 제안한 기법을 적용하였다.

Input : $x' = x \oplus r,\ r$
Output : $A = x - r$

1. $\Gamma \leftarrow \gamma$
2. $T \leftarrow x' \oplus \Gamma$
3. $T \leftarrow T - \Gamma$
4. $T \leftarrow T \oplus x'$
5. $\Gamma \leftarrow \Gamma \oplus r$
6. $A \leftarrow x' \oplus \Gamma$
7. $A \leftarrow A - \Gamma$
8. $A \leftarrow A \oplus T$

Fig. 1. Goubin's BtoA conversion method

2.2 산술-부울 마스킹 변환 기법

산술-부울 마스킹 변환 기법은 산술 마스킹 된 중간 값을 부울 연산을 위해 부울 마스킹 된 중간 값으로 변환하는 기법이다. 본 논문에서는 CHAM 암호에 대한 산술-부울 마스킹 변환 기법으로 Goubin 기법, C-T 기법, Debraize 기법을 적용한다.

2.2.1 Goubin 산술-부울 마스킹 변환 기법

Goubin이 제안한 산술-부울 마스킹 변환 기법(이하 Goubin 기법)은 산술 마스킹 된 중간 값 $A(=x-r)$와 마스크 값 r을 입력으로 하고 알고리즘 연산을 통해 부울 마스킹 된 값 $x' = x \oplus r$을 출력한다. Goubin 기법의 특징은 입력 비트 길이 m 만큼의 반복 연산이 진행되어 이론상 많은 연산량을 필요로 한다는 것이다. 즉, m비트 연산을 처리하는 경우 $(2m+4)$번의 XOR 연산, $(2m+1)$번의 AND 연산, m번의 Shift 연산으로 총 $(5m+5)$번의 기본 연산이 소요된다. 다음 Fig. 2는 Goubin이 제안한 산술-부울 마스킹 변환 알고리즘을 나타낸 것이다.

Input : $A = x - r,\ r$
Output : $x' = x \oplus r$

1. $\Gamma \leftarrow \gamma$
2. $T \leftarrow 2\Gamma$
3. $x' \leftarrow \Gamma \oplus r$
4. $\Omega \leftarrow \Gamma \wedge x'$
5. $x' \leftarrow T \oplus A$
6. $\Gamma \leftarrow \Gamma \oplus x'$
7. $\Gamma \leftarrow \Gamma \wedge r$
8. $\Omega \leftarrow \Omega \oplus \Gamma$
9. $\Gamma \leftarrow T \wedge A$
10. $\Omega \leftarrow \Omega \oplus \Gamma$
11. for $i = 1$ to $m-1$ do
    $\Gamma \leftarrow T \wedge r$
    $\Gamma \leftarrow \Gamma \oplus \Omega$
    $T \leftarrow T \wedge A$
    $\Gamma \leftarrow \Gamma \oplus T$
    $T \leftarrow 2\Gamma$
    end for
12. $x' \leftarrow x' \oplus T$

Fig. 2. Goubin's AtoB conversion method

2.2.2 C-T 산술-부울 마스킹 변환 기법

Coron과 Tchulkine은 Goubin이 제안한 산술-부울 변환 기법이 필요로 하는 연산량이 많아 비효율적이라는 것을 지적하며 룩업(look-up) 테이블 기반의 새로운 산술-부울 마스킹 변환 기법(이하 C-T 기법)을 제안하였다[11]. 즉, $x' = (A + r_x) \oplus r_x$를 k-비트 단위로 처리할 때 덧셈에 의해 발생하는 캐리($carry$)값을 미리 계산하여 테이블에 저장해 둔 후, 실제 변환 연산이 수행될 때 저장해 놓은 테이블 값을 이용하는 기법이다. 이렇게 룩업 테이블을 이용하면 필요한 임시 메모리 크기가 늘어나지만 연산 속도를 높일 수 있다는 장점이 있다. 다음 Fig. 3은 산술-부울 마스킹 변환 알고리즘을 나타낸 것이다.

Input : $A = x - r$, r
Output : $x' = x \oplus r \oplus (s \| s \| \ldots \| s)$
[Pre-computation – Table T Generation]
1. Generate a random k-bit s, a random 1-bit ρ
2. for $i = 0$ to $2^k - 1$ do
  2.1 $T[\rho \| i] = (i + s) \oplus (\rho \| s)$
  2.2 $T[(\rho \oplus 1) \| i] = (i + s + 1) \oplus (\rho \| s)$
3. Output T, s, ρ

1. $A = A - (s \| s \| \ldots \| s) \bmod 2^{n \cdot k}$
2. $\beta = \rho$
3. for $i = 0$ to $n - 1$ do
  $r = r_h \| r_l$ (k-bit r_l)
  $A = A + r_l \bmod 2^{(n-i) \cdot k}$
  $A = A_h \| A_l$ (k-bit A_l)
  $\beta \| x_i' = T[\beta \| A_l]$
  $x_i' = x_i' \oplus r_l$
  $A = A_h$, $r = r_h$
4. Output $x' = (x_{n-1}' \| x_{n-2}' \| \ldots \| x_1' \| x_0') \oplus (s \| s \| \ldots \| s)$

Fig. 3. C-T's AtoB conversion algorithm

2.2.3 Debraize 산술-부울 마스킹 변환 기법

Debraize는 Coron등이 제안하였던 산술-부울 마스킹 변환 기법의 안전성 및 효율성에 대한 문제를 제기하며 룩업 테이블 기반의 새로운 산술-부울 마스킹 변환 알고리즘(이하 Debraize 기법)을 제안하였다[12]. Debraize 기법은 C-T 기법의 마스킹 변환 단계를 수정하여 연산 효율성을 높였으며 사전 연산 데이터에 대한 메모리를 참조하는 시점에서 발생할 수 있는 부채널 공격을 방지하기 위해 사전 연산 과정에서 1비트의 난수로 메모리 주소를 마스킹 한다는 점이 특징이다. 다음 Fig. 4는 Debraize가 제안한 산술-부울 마스킹 변환 기법 알고리즘을 나타낸 것이다.

III. Secure Addition 마스킹 기법

기존의 마스킹 상호 변환 기법을 이용하려 부울 마스킹 된 두 값을 더하기 위해서는 부울-산술 마스킹 변환, 산술 연산, 산술-부울 마스킹 변환의 3번의 과정을 거쳐야 한다. 하지만 이러한 마스킹 상호 변환 기법의 복잡성과 연산 효율성을 개선하기 위해 SA 기법이 제안되었다. SA 기법은 3번의 마스킹 변환 과정을 거칠 필요 없이 한 번의 과정을 통해 부울 마스킹 된 두 데이터에 대한 산술 덧셈을 수행하되 부울 마스킹 된 결과 데이터를 출력할 수 있다는 큰 장점이 있다.

Input : A, r_x
Output : x'
[Pre-computation]
1. Generate random number k-bit s, 1-bit ρ
2. for $A = 0$ to $2^k - 1$ do
2.1 $T[\rho \| A] = (A + s) \oplus (\rho \| s)$
2.2 $T[(\rho \oplus 1) \| A] = (A + s + 1) \oplus (\rho \| s)$

1. $A = A - (s \| s \| \ldots \| s) \bmod 2^{n \cdot k}$
2. $\beta = \rho$
3. for $i = 0$ to $n - 1$ do
3.1 $A = A_h \| A_l$, $r_x = r_{xh} \| r_{xl}$ (k-bit A_l, r_{xl})
3.2 $A = A + r_{xl} \bmod 2^{(n-i) \cdot k}$
3.3 $\beta \| x_i' = T[\beta \| A_l]$
3.4 $x_i' = x_i' \oplus r_{xl}$
3.5 $A = A_h$, $r_x = r_{xh}$
4. $x' = (x_0' \| x_1' \| \ldots \| x_{n-1}') \oplus (s \| s \| \ldots \| s)$

Fig. 4. Debraize's AtoB conversion algorithm

3.1 KRJ SA 마스킹 기법

Karroumi 등은 부울 마스킹 된 두 중간 값을 입력받아 새로운 값으로 부울 마스킹 된 덧셈 결과를 출력하는 SA 마스킹 기법(이하 KRJ 기법)을 처음으로 제안하였다[13]. KRJ 기법은 AND, XOR, Shift 연산과 같은 기본적인 논리 연산만으로 효율적인 덧셈 마스킹을 수행할 수 있다.

만약 입력 비트 길이를 k라고 한다면, Goubin

Input : x', y', r_x, r_y
Output : z'

1. $C = \gamma$	10. $B = \Omega \ll 1$
2. $T = x' \wedge y'$	11. $C = C \ll 1$
3. $\Omega = C \oplus T$	12. $z' = x' \oplus y'$
4. $T = x' \wedge r_y$	13. $r_z = r_x \oplus r_y$
5. $\Omega = \Omega \oplus T$	14. $T = C \wedge z'$
6. $T = y' \wedge r_x$	15. $\Omega = \Omega \oplus T$
7. $\Omega = \Omega \oplus T$	16. $T = C \wedge r_z$
8. $T = r_x \wedge r_y$	17. $\Omega = \Omega \oplus T$
9. $\Omega = \Omega \oplus T$	
18. for $i = 2$ to $k - 1$ do	
18.1 $T = B \wedge z'$	18.4 $B = B \oplus T$
18.2 $B = B \wedge r_z$	18.5 $B = B \ll 1$
18.3 $B = B \oplus \Omega$	
19. $z' = z' \oplus B$	20. $z' = z' \oplus C$

Fig. 5. KRJ SA masking method

기법의 경우 총 $(5k+21)$의 기본 연산이 필요하지만, KRJ 기법은 $(5k+8)$의 기본 연산이 필요하게 되어 보다 효율적으로 산술 덧셈 연산을 수행할 수 있다. 다음 Fig. 5는 KRJ SA 알고리즘을 나타낸 것이다.

3.2 V-G SA 마스킹 기법

Vadnala와 Großschdl는 KRJ 기법의 연산량을 개선하기 위해 사전 테이블을 이용한 SA 마스킹 기법(이하 V-G 기법)을 제안하였다[14]. VG 기법의 기본 개념은 k-비트 단위의 $(x\oplus s_1)+(y\oplus s_2)$ 값을 계산할 때 발생하는 캐리를 미리 계산하여 실제 마스킹 연산 단계에서 이용하는 것이다. 다음 Fig. 6은 V-G SA 마스킹 기법을 나타낸 것이다.

3.3 CGTV SA 마스킹 기법

Coron 등은 기본적인 덧셈 알고리즘을 사용하는 KRJ 기법과 다르게 Kogge-Stone 올림 수 예견 덧셈기에 기반한 마스킹 기법을 제안하였다(이하 CGTV 기법)[15]. CGTV 기법은 $k=2^n$의 환경에서 $(28n+4)$번의 기본 논리 연산을 가지게 되는데 이는 Goubin 기법이나 KRJ 기법이 $O(k)$의 연산 복잡도를 갖는 것에 비하면 보다 효율적인 SA 기법이다. 다음 Fig. 7은 CGTV 기법을 나타낸 것인데, 그림에서 SecXor( ), SecAnd( ), SecShift( ) 함수들은 각각 마스킹 기반의 안전한 XOR, AND, Shift 연산을 의미한다.

Input : x', y', r_x, r_y
Output : z'

[Pre-computation]
1. Generate random number k-bit s_1, s_2, 1-bit ρ
2. for $A=0$ to 2^k-1 do
2.1 for $B=0$ to 2^k-1 do
2.1.1 $C=(A\oplus s_1)+(B\oplus s_2)$
2.1.2 $T[\rho\|A\|B]=C\oplus(\rho\|s_1)$
2.1.3 $T[(\rho+1)\|A\|B]=(C+1)\oplus(\rho\|s_1)$

1. $\beta=\rho$
2. $x'=x'\oplus(s_1\|s_1\|...\|s_1)\oplus r_x$
3. $y'=y'\oplus(s_2\|s_2\|...\|s_2)\oplus r_y$
2. for $i=0$ to $n-1$ do
2.1 $\beta\|z_i'=T[\beta\|x_i'\|y_i']$
3. $z'=(z_1'\|z_2'\|...\|z_{n-1}')\oplus r_x\oplus r_y$
4. $z'=z'\oplus(s_1\|s_1\|...\|s_1)$

Fig. 6. V-G SA masking method

Input : x', y', r_x, r_y
Output : z'

1. Generate random number k-bit s_1, s_2
2. $P'=SecXor(x',y',r_y)$
3. $G'=SecAnd(x',y',r_x,r_y,u)$
4. $G'=G'\oplus r_x$ 5. $G'=G'\oplus u$
6. for $i=1$ to $n-1$ do
6.1 $H=SecShift(G',r_x,s_1,2^{i-1})$
6.2 $U=SecAnd(P',H,r_x,s_1,s_2)$
6.3 $G'=SecXor(G',U,s_2)$
6.4 $H=SecShift(P',r_x,s_1,2^{i-1})$
6.5 $P'=SecAnd(P',H,r_x,s_1,s_2)$
6.6 $P'=P'\oplus r_x$ 6.7 $P'=P'\oplus s_2$
7. $H=SecShift(G',r_x,s_1,2^{n-1})$
8. $U=SecAnd(P',H,r_x,s_1,s_2)$
9. $G'=SecXor(G',U,s_2)$
10. $z'=SecXor(y',x',r_x)$
11. $z'=z'\oplus 2G$ 12. $z'=z'\oplus 2r_x$
13. $z'=z'\oplus r_x$

Fig. 7. CGTV SA masking method

IV. CHAM 블록 알고리즘 및 마스킹 기법 적용

국내에서 개발된 CHAM 블록 암호 알고리즘은 입력된 평문을 4개의 블록으로 나누어 처리하는 Feistel 구조의 초경량 블록 암호이다. CHAM 암호는 키 스케줄 과정에서 키 상태를 업데이트하지 않는 특징이 있고, 특히, 라운드 키의 수를 라운드 수보다 적게 하여 라운드 키가 재사용되도록 설계함으로서 라운드 키를 저장하는데 필요한 메모리 크기를 크게 줄였다.

본 장에서는 경량 블록 암호 CHAM을 설명하고,

CHAM 암호에 마스킹을 적용하기 위해 꼭 필요한 과정인 산술-부울 상호 마스킹 변환 과정과 SA 마스킹 과정을 어떻게 적용할 것인지 제안한다.

4.1 경량 블록 암호 알고리즘 CHAM

CHAM 블록 암호의 라운드 함수는 ARX(Addition, Rotation, XOR) 연산을 기본으로 이루어져 있으며, 입력 크기는 64~128비트, 키 길이는 128~256비트를 지원한다. 다음 Fig. 8은 CHAM 암호의 두 라운드 연산을 나타낸 것이다. Fig. 8과 같이 두 번의 좌측 순환 이동(ROL, Rotation of Left) 연산의 비트 수는 라운드 i가 짝수일 때와 홀수일 때 각각 (1, 8), (8, 1)로서 다소 차이가 있다. 마지막으로 CHAM 암호의 입력 파라미터는 Table. 1과 같은데 라운드 연산이 간단한 것에 비해 라운드 수가 80~96으로 다른 경량 암호 알고리즘에 비해 많다는 특징을 가지고 있다. 그러나 CHAM 블록 암호 알고리즘도 대부분의 블록 암호 알고리즘과 마찬가지로 비밀 키가 사용된 이후의 암호 중간 값들을 추측하고 부채널 정보를 이용하여 이 값을 찾아낼 수 있다는 원리가 적용 가능하여 부채널 공격에 취약한 특성을 그대로 내재하고 있다고 볼 수 있다.

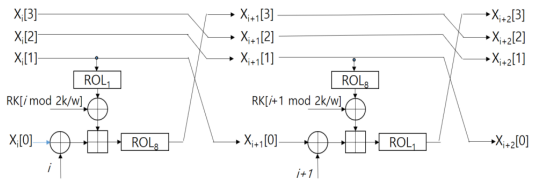

Fig. 8. Two round functions beginning with the even i-th round

Table 1. CHAM cipher parameters

Cipher	n	k	r	w	k/w
CHAM-64/128	64	128	80	16	8
CHAM-128/128	128	128	80	32	4
CHAM-128/256	128	256	96	32	8

4.2 산술-부울 상호 마스킹 변환 기법 적용

CHAM 라운드 함수에는 부울 연산인 XOR 연산과 좌측 순환 이동 연산이 있으며 산술 연산인 덧셈 연산이 존재한다. 따라서 부울 연산과 산술 연산 모두 존재하기 때문에 CHAM 라운드 함수에 마스킹을 적용할 때는 산술-부울 상호 마스킹 변환 과정을 사용하거나 SA 마스킹 기법을 시용하여야 한다.

먼저, 산술-부울 상호 마스킹 변환 기법을 적용할 때에는 덧셈 연산 전에 부울 마스킹되어 있는 두 중간 값을 산술 마스킹 값으로 변환한 후 덧셈 연산을 수행한다. 그 다음 덧셈 연산을 수행한 산술 마스킹 된 중간 값은 다시 부울 마스킹 값으로 변환하는 과정을 거친다. 이와 같이 산술-부울 상호 마스킹 변환 기법을 라운드 함수에 적용한 것이 Fig. 9이다. 여기서 BtoA, AtoB로 표기되어 있는 것은 각각 부울-산술 마스킹 변환과 산술-부울 마스킹 변환을 의미한다.

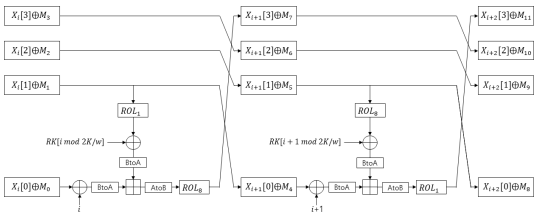

Fig. 9. Round function with AtoB and BtoA masking conversion method

4.3 SA 마스킹 기법 적용

SA 마스킹 기법은 부울 마스킹 된 두 값을 입력으로 받아 산술 덧셈 연산을 수행한 후 새로운 값으로 부울 마스킹된 결과를 출력하는 기법이다. 따라서 기존 라운드 함수에서 산술 덧셈 연산을 하는 위치에 SA 마스킹 기법을 적용하면 된다. CHAM 암호 알고리즘의 각 라운드에 SA 마스킹 기법을 적용한 것이 Fig. 10이다.

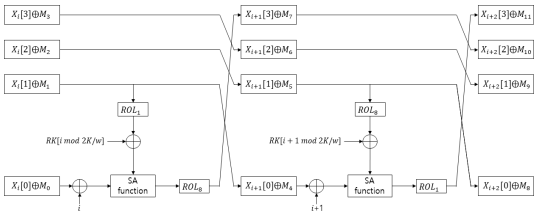

Fig. 10. Round function with SA masking method

V. 마스킹을 적용한 CHAM 구현 및 성능

본 논문에서는 2장 및 3장에서 설명한 총 6가지 마스킹 기법을 CHAM-128/128 암호 알고리즘에 적용하였으며, 실제 32비트 마이크로프로세서인 Cortex-M3 플랫폼에 구현하여 효율성을 분석하였다. Cortex-M3 보드 구현에는 IAREW-ARM에서 제공하는 컴파일러를 사용하였다. 컴파일 시 속도 및 메모리 최적화 옵션은 별도로 지정하지 않았다. 또한 각 마스킹 기법을 CHAM에 적용하여 각각의 성능을 비교 분석하기 위해 컴파일러 디버깅 모드에서 제공하는 클럭 사이클 수를 측정하였다. 다음 Table 2는 상기한 6가지 마스킹 기법을 CHAM-128/128에 적용하여 구현한 후 측정한 수행 속도를 나타낸 것이다.

구현한 CHAM에서는 라운드 키를 미리 생성하여 저장한 후 순차적으로 사용하였으며 사전 계산은 룩업 테이블을 구성하는데 필요한 시간을 의미한다. C-T기법과 Debraize 기법에서는 $k=8$로 설정하여 각각 $2^k=256$ 개와 $2^{k+1}=512$ 개의 룩업 테이블을 사용하였으며 V-G 기법에서는 $k=4$로 설정하여 $2^{2k+1}=512$ 개의 룩업 테이블을 사용하였다.

CHAM-128/128 블록 암호에 대해 마스킹을 적용하지 않았을 경우, 암호 연산에 필요한 연산량은 1,408번의 사이클로 SPECK이나 LEA와 같은 다른 경량 블록 암호 알고리즘들과 비교하였을 때 상당히 좋은 성능을 보인다. 하지만 마스킹 기법을 적용하였을 때에는 15배에서 25배로 상당히 많은 추가 연산을 필요로 한다.

다음 Fig. 11은 CHAM-128/128에 대해 마스킹 기법 적용 시 필요한 사이클 수를 그림으로 비교하여 도시한 것이다. 그림에서 보는 바와 같이 마스킹으로 인한 부가 연산은 암호 구현의 상당한 부담이 되고 있으며, C-T 기법을 적용했을 때 상대적으로 빠른 속도를 보이는 것으로 분석되었다.

Table 2. Performance comparison of masking methods applied on CHAM

Masking method	Pre-computation cycles	Encryption cycles	Ratio
No-Masking	-	1,408	1.00
Goubin	-	26,087	18.52
C-T	2,360	21,127	15.00
Debraize	9,994	21,287	15.11
KRJ	-	26,784	19.02
CGTV	-	35,624	25.30
V-G	7,089	28,144	19.99

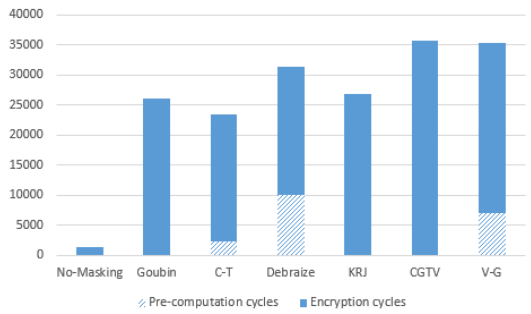

Fig. 11. Computational cycles for encryption on masked CHAM-128/128

다음 Table 3은 Cortex-M3 보드 플랫폼 상에서 Goubin과 KRJ 마스킹 기법을 사용하여 다른 경량 블록 암호들을 구현한 후 측정한 연산량이다. 표에서 보는 바와 같이 다른 경량 블록 암호 라운드에 마스킹 기법을 적용하였을 때 적게는 4배에서 많게는 10배까지 추가 연산이 늘어나는 것을 볼 수 있다. 그에 비해 마스킹된 CHAM 암호 라운드의 연산량 증가 비율이 상당히 높다는 것을 알 수 있다.

이렇게 CHAM 암호에 마스킹 알고리즘들을 적용시켰을 때 추가로 늘어나는 연산량이 많은 이유는 다른 경량 블록 암호들에 비해 많은 라운드 수를 갖는 CHAM 암호의 특징 때문이다. CHAM 암호는 버전에 따라 80 또는 96 라운드 수를 갖는데 이는 24~32의 라운드 수를 갖는 LEA 암호와 비교하였을 때 CHAM 암호의 라운드 수가 3~4배 정도 많음을 알 수 있다. 이렇게 라운드 수가 많을 경우 각 마스킹 알고리즘이 마스킹 값 계산을 위해 필요한 연산이 라운드 수만큼 가중되기 때문에 마스킹 기법을

Table 3. Performance comparison of masking methods applied on some lightweight block ciphers

		Non	Goubin	KRJ
SPECK	Cycles	14,791	66,467	58,979
	Ratio	1.00	4.49	3.98
LEA	Cycles	2,392	24,352	24,712
	Ratio	1.00	10.18	10.33
CHAM	Cycles	1,408	26,087	26,784
	Ratio	1.00	18.52	19.02

적용하는 것이 연산 속도면에서 매우 불리하다.

실제로 마스킹이 적용된 CHAM 암호에서 연산량이 가중되는 원인을 분석하기 위해 구현한 코드를 세부적으로 분석하여 나타낸 것이 Fig. 12이다. 그림에서 보는 바와 같이 원래의 CHAM 암호 라운드 연산량이 18사이클인 반면, Goubin 기법을 적용하였을 때에는 마스킹 알고리즘 연산까지 추가되어 325사이클이 필요하다. 즉, 원래의 CHAM 암호 라운드 연산량과 비교하였을 때 마스킹 적용 시 연산량이 급증하는 것을 볼 수 있다. 또한, 80번 또는 96번의 라운드 수만큼 마스킹 연산이 다른 경량 암호들에 비해 비교적 많이 반복되어 연산량이 크게 가중되는 것으로 분석된다.

상기한 바와 같이 CHAM 블록 암호에 마스킹 기법을 전체 라운드에 적용한 결과 추가되는 연산량이 과중하여 경량 디바이스 환경에 활용하기 부적합한 면이 있다. 따라서 본 논문에서는 CHAM 암호의 보안 강도를 그대로 유지하면서 연산 속도를 개선하기 위해 일부 라운드에만 마스킹을 적용하는 축소 마스킹을 적용하였다. 일반적으로 축소 마스킹 기법은 암호 라운드의 처음 부분과 마지막 부분의 라운드에

▪ Non-Masked CHAM round code

```
b0=rol32((b0^rc++)) + rol32(b1, 1) ^ rk[0]), 8);        | 18 cycles
```

▪ Masked CHAM round code

```
temp_x = B2A(b0 ^ (rc++), m0);                          | 16 cycles
temp_m = rol32(m1, 1);
temp_y = B2A(rol32(b1, 1) ^ rk[0], temp_m);             | 325 cycles
temp_m = (m0 + temp_m)&LAND;
b0 = rol32 (A2B_G(temp_x + temp_y, temp_m), 8);
m0 = rol32(temp_m, 8);
```

```
#define k_length  32
uint32_t A2B_G(uint32_t input, uint32_t mask)
{
        uint32_t output;
        uint32_t gamma, c_gamma, omega, T;
        int i;

        gamma = 0x6677aabb;
        c_gamma = gamma;
        T = (c_gamma << 1) &LAND;
        output = c_gamma ^ mask;
        omega = c_gamma & output;
        output = T ^ input;
        c_gamma = c_gamma ^ output;
        c_gamma = c_gamma & mask;
        omega = omega ^ c_gamma;
        c_gamma = T & input;
        omega = omega ^ c_gamma;
        for (i = 1; i < k_length; i++) {                | 252 cycles
                c_gamma = T & mask;
                c_gamma = c_gamma ^ omega;
                T = T & input;
                c_gamma = c_gamma ^ T;
                T = (c_gamma << 1) & LAND;
        }
        return(output ^ T);
}
```

Fig. 12. Comparison of cycles for one round computation on non-masked and masked (Goubin method) CHAM

마스킹을 적용하고 있다.

CHAM 암호 알고리즘은 버전에 따라 8개 또는 16개의 라운드 키를 생성하여 이를 반복 사용하는 구조로 이루어져 있다. 따라서 본 논문에서는 이러한 CHAM 암호 라운드의 특성을 이용하여 생성된 라운드 키 전체가 처음 한번 사용되는 구간과 마지막 한번 사용되는 구간에만 마스킹을 적용하는 축소 마스킹 적용 방법을 제안한다. 즉, 제안하는 축소 마스킹 방법에서는 CHAM 암호의 버전에 따라 16라운드 또는 32라운드에만 마스킹을 적용하게 된다. 다음 Table 4는 본 논문에서 적용한 CHAM 암호에 대한 축소 마스킹 사양을 나타낸 것인데 전체 암호화 과정에서 라운드 키가 처음 사용되는 구간과 마지막 사용되는 구간에만 마스킹을 적용하게 된다. Table 5는 CHAM-128/128에 축소 마스킹을 적용하여 수행한 경우의 연산 속도를 나타낸 것이다.

Table 4와 같이 CHAM-128/128 암호 라운드에 대한 축소 마스킹을 적용한 경우 전체 라운드의 20%에 해당하는 부분에만 마스킹을 적용하게 된다. 이렇게 축소 마스킹을 적용한 CHAM-128/128을 Cortex-M3 보드 상에서 구현한 결과 Table 5와 같이 4~5배 정도의 추가 연산량만 필요하게 된다. 이 결과는 Table 2의 전체 라운드에 마스킹을 적용한 결과 값과 비교해 볼 때 효율적으로 암호 중간 값에 대한 추측을 방지하여 부채널 공격에 대응할 수

Table 4. Reduced masking specification for CHAM

Version	r	Number of round key	Number of masked round	Ratio
64/128	80	16	32	40%
128/128	80	8	16	20%
128/256	96	16	32	33%

Table 5. Performance comparison of reduced masking methods applied on CHAM

Masking method	Pre-computation cycles	Encryption cycles	Ratio
No-Masking	-	1,408	1.00
Goubin	-	6,431	4.56
C-T	2,360	5,439	3.86
Debraize	9,994	5,471	3.88
KRJ	-	6,566	4.66
CGTV	-	15,465	10.98
V-G	7,089	6,838	4.85

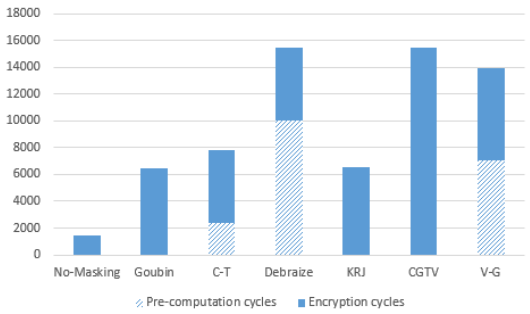

Fig. 13. Reduced Masked CHAM-128/128

있음을 확인할 수 있다.

블록 암호 알고리듬은 대부분 암호 운영 모드를 사용하게 되므로 전체적인 속도는 사전 계산량을 제외하고 수행속도를 비교해 볼 필요가 있다. 사전 계산 연산량을 제외했을 때, 전체 라운드 마스킹은 C-T 기법의 경우 마스킹을 하지 않은 경우(1,408 사이클) 보다 약 15배(21,127 사이클) 정도의 많은 연산량이 필요했지만 축소 마스킹 기법을 적용했을 때에는 3.86배(5,439 사이클)의 연산량이 소요되어 여러 마스킹 기법 중 가장 효율적인 기법으로 CHAM 암호에 적용될 수 있음을 확인하였다. 단, C-T 기법은 사전 계산된 캐리 값을 저장하는 메모리 공간 256~512 바이트 정도가 필요하였다.

VI. 결 론

본 논문에서는 초경량 블록 암호 CHAM에 대한 부채널 분석 공격의 대응 방법 중 하나인 마스킹 기법을 적용하기 위한 설계 방법을 제시하였으며, 여러 마스킹 기법들을 CHAM 암호에 적용하여 효율성을 비교·분석하였다. 그러나 32비트 마이크로프로세서 Cortex-M3 보드에 6가지 마스킹 기법들이 적용된 CHAM-128/128을 구동시킨 결과 15배에서 25배로 사실상 마스킹 기법을 CHAM 암호 전체 라운드에 적용하는 것이 가용성면에서 부적합하다는 것을 확인할 수 있었다.

따라서 마스킹 기법을 적용함에 있어 CHAM이 갖는 설계적 약점을 보완하기 위해 본 논문에서는 축소 마스킹을 적용하였다. 특히, 8개 또는 16개의 라운드 키를 생성하여 라운드 연산이 진행되는 동안 반복하여 사용하는 CHAM 암호의 특징을 이용하여 라운드 키 전체가 암호화 단계의 처음과 마지막에 한 번씩 사용되는 부분에 대한 축소 마스킹을 적용하였다. CHAM-128/128의 경우 축소 마스킹을 적용하였을 때가 4~5배 정도의 추가 연산량만 필요하게 되어 전체 라운드 마스킹을 했을 때보다 효율적으로 부채널 공격에 대응할 수 있음을 확인하였다.

References

[1] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," CHES'07, LNCS 4727, pp. 450-466, 2007.

[2] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata, "The 128-bit Block cipher CLEFIA(Extended Abstract)," FSE'07, LNCS 4593, pp. 181-195, 2007.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] H. Seo, "Memory-efficient implementation of ultra-lightweight block cipher algorithm CHAM on low-end 8-bit AVR processors," Journal of The Korea Institute of Information Security & Cryptology(JKIISC), 28(3), pp. 545-550, Jun. 2018.
[9] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," CRYPTO'99, LNCS 1666, pp. 388-397, 1999.
[10] L. Goubin, "A sound method for switching between Boolean and arithmetic masking," CHES'01, LNCS 2162, pp. 3-15, 2001.
[11] J. Coron and A. Tchulkine, "A new algorithm for switching from arithmetic to Boolean Masking," CHES'03, LNCS 2779, pp. 89-97, 2003.
[12] Blandine Debraize, "Efficient and Provably Secure Methods for Switching from Arithmetic to Boolean Masking," CHES'12, LNCS 7428, pp. 107-121, 2012.
[13] M. Karroumi, B. Richard, and M. Joye, "Addition with blinded operands," Constructive Side- Channel Analysis and Secure Design, COSADE'14, LNCS 8622, pp. 41-55, 2014.
[14] P. Vadnala and J. Groβschadl, "Faster mask conversion with lookup tables," Constructive Side-Channel Analysis and Secure Design, COSADE'15, LNCS 9064, pp. 207-221, 2015.
[15] J. Coron, J. Groβschadl, M. Tibouchi, and P. Vadnala, "Conversion from arithmetic to Boolean masking with Logarithmic complexity," Fast Software Encryption, FSE'15, LNCS 9054, pp. 130-149, 2015.

〈저자소개〉

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

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