Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.35, NO.1, Feb. 2025 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2025.35.1.25

SMAUG-T 곱셈 연산에 대한 단순전력분석 공격 및 대응기법*

유 성 환,[1†] 한 재 승,[2] 한 동 국[3‡]
[1,2,3]국민대학교 (학생, 대학원생, 교수)

Simple Power Analysis and Countermeasure for SMAUG-T Multiplication Operation*

Sung-Hwan Yoo,[1†] Jaeseung Han,[2] Dong-Guk Han[3‡]
[1,2,3]Kookmin University (Undergraduate student, Graduate student, Professor)

요 약

최근 양자 컴퓨터의 발전으로 양자 내성 암호(PQC)에 대한 활발한 연구가 이루어지고 있다. 국내에서는 2022년도부터 국내 PQC 국가공모전(KPQC)을 시작하였고, 8종의 알고리즘이 2라운드 후보로 선정되어 평가 중이다. 본 논문은 KpqC 2라운드 후보 중 격자 기반 PKE/KEM 알고리즘인 SMAUG-T에 대한 2가지 신규 단순전력분석 취약점과 이를 이용한 비밀키 정보 누출 가능성을 제시한다. SMAUG-T 알고리즘은 비밀키 계수의 sparse ternary 특성에 맞춰 다항식 곱셈이 구현되어 있다. 본 논문은 해당 다항식 곱셈 구조가 상수 시간을 만족하지 못하며, 비밀키 계수의 값에 따른 전력 차분이 발생함을 보인다. 그리고 해당 취약점을 통해 단일 전력 파형으로 각 비밀키 다항식의 계수 중 1, 0, -1의 개수를 복구하는 방법을 제안하며, 이를 ARM Cortex- M4 기반 MCU 환경에서 증명하였다. 추가적으로, 이에 대한 대응기법을 제시하고 동일한 환경에서 제안된 단순전력분석 공격을 수행하여 안전성을 실험적으로 입증했으며, Cycle 수와 메모리에서의 오버헤드를 측정하여 성능 평가를 진행했다.

ABSTRACT

Recently, with the development of quantum computers, active research on quantum-resistant cryptography (PQC) has been conducted. In Korea, the domestic PQC National Competition (KPQC) has started in 2022, and eight algorithms have been selected as second-round candidates and are being evaluated. This paper presents two new simple power analysis vulnerabilities and the possibility of secret key information leakage using SMAUG-T, a lattice-based PKE/KEM algorithm. The SMAUG-T algorithm implements polynomial multiplication according to the sparse ternary characteristic of the secret key coefficients. This paper shows that the polynomial multiplication structure does not satisfy constant time and that power differences occur depending on the value of the secret key coefficients. In addition, we propose a method to recover the number of 1, 0, and -1 among the coefficients of each secret key polynomial with a single power waveform through the vulnerability, and showed that the secret key information was recovered in an ARM Cortex-M4-based MCU environment. Additionally, we proposed a countermeasure technique and experimentally verified the safety of the countermeasure technique by performing the proposed simple power analysis attack in the same environment. We also evaluated the performance by measuring the number of cycles and the overhead in memory.

Keywords: Side-Channel Analysis, Korean Post-Quantum Cryptography, Single-Trace Attack, Countermeasure

Received(10. 04. 2024), Modified(12. 18. 2024), Accepted(12. 20. 2024)

* 본 논문은 2024년도 한국정보보호학회 하계학술대회에 발표한 우수논문을 개선 및 확장한 것임

* 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT연구센터사업의 연구결과로 수행되었음(IITP-2024-RS-2022-00164800)

† 주저자, sungwhab4854@kookmin.ac.kr

‡ 교신저자, christa@kookmin.ac.kr(Corresponding author)

I. 서 론

최근 4차 산업혁명 시대의 핵심 기술로 양자 컴퓨터와 양자 알고리즘이 지속적으로 발전되고 있다. 그에 따라 RSA와 ECC 암호 알고리즘에 대한 보안 위협도 증가하고 있고 양자 컴퓨터의 상용화 후에도 안전한 암호 통신 체계를 만들고자 전 세계적으로 다양한 노력을 기하고 있다. 미국 국립표준기술연구소(national institute of standards and technology, NIST)에서는 2016년부터 양자 내성 암호(post-quantum cryptograpy, PQC)에 대한 연구를 활발히 진행하고 있다. 2024년 8월 기준 PKE/KEM(Public Key Encryption/Key Encapsulation Mechanism)에서 CRYSTALS-Kyber를 표준으로 선정했고 전자서명은 CRYSTALS-Dilithium, Falcon, SPHINCS+를 선정했다. 2022년에 시작한 국내 KpqC 공모전은 2024년 8월 기준으로 2라운드를 진행하고 있다. 2라운드에서는 PKE/KEM 4종, 전자서명에서 4종이 선정되어 평가되고 있다.

PQC 공모전 이후 현재까지도 다양한 분야에서 PQC 알고리즘에 대한 안전성 분석이 진행되고 있다. 그 중 부채널 분석은 PQC 알고리즘의 직접적인 비밀키 누출을 발생시키는 위협 중 하나이다. 격자(lattice) 기반 PQC에 대한 차분 전력 공격[1], 단일 파형 공격[2, 3], 선택 암호문 부채널 공격[4-6] 등 여러 부채널 분석 결과가 제시됐다. 하지만 대부분의 연구는 NIST PQC 후보에 대한 공격 결과를 제시하고 있으므로 KpqC 알고리즘의 구조 특성에 따른 부채널 분석 연구가 필요하다. 2023년, 이종환 등은 KpqC 1라운드 격자 기반 PKE/KEM 후보 SMAUG에 대한 디캡슐화 동작 시간 차이를 이용한 공격을 제시했다[7]. 이에 따라 KpqC 2라운드 SMAUG-T에서는 sampler의 구현을 수정해 해당 취약점을 제거했다.

본 논문은 KpqC 2라운드 PKE/KEM 알고리즘 중 SMAUG-T 알고리즘[8]에 대한 단순전력분석(simple power analysis, SPA) 공격을 제안한다. 제시한 공격은 다항식 곱셈 연산이 비밀키 값에 의존한 연산 순서를 수행하는 특징을 이용하여 단일 소비전력 파형에 대한 SPA를 통해 비밀키 정보 일부를 복구한다. 추가로 공격에 대한 대응기법을 제시하여 본 논문에서 제시한 SPA 공격에 저항성을 가짐을 보이고, 대응기법 적용 여부에 따른 Clock Cycles과 메모리 오버헤드를 측정하여 성능 비교를 수행한다.

[Contribution]

1. **(새로운 공격 제안)** SMAUG-T의 다항식 곱셈 연산이 상수시간을 만족하지 않으며, 비밀키 계수의 값에 따라 수행 연산이 달라져 소비전력이 구분됨을 보인다. 두 취약점을 통해 각각 1, 0, -1로 이루어진 비밀키 다항식별 계수의 개수를 복구하는 SPA 공격을 제안한다. 그리고 ARM Cortex-M4 기반 STM32F3 환경에서 수집한 단일 전력 파형으로 제안한 SPA 공격을 수행해 공격의 실효성을 보였다.
2. **(대응기법 제안)** 기존의 SMAUG-T 곱셈 연산 구조에 기반하지만 비밀키 계수에 의존되지 않고 상수시간 구현을 만족하는 대응기법을 제안한다. 제안한 대응기법 적용 후 동일환 환경에서 SPA 공격을 수행한 결과 상수시간 연산을 만족하며, 비밀키 계수에 따른 전력 차분이 유출되지 않음을 보였다. 추가로 대응기법 적용에 따른 오버헤드를 보인다.

[Organization]

본 논문의 구성은 아래와 같다. 2장에서 SMAUG-T 알고리즘과 구현 특징을 소개한다. 3장에서는 2장에서 소개한 특징에 의한 SMAUG-T 곱셈 연산의 SPA 취약점을 설명하며, 해당 취약점을 통한 SPA 공격을 제안한다. 또한, ARM Cortex-M4 기반 STM32F3 MCU에서의 제안한 공격 결과를 제시한다. 4장에서는 제시한 취약점에 대응할 수 있는 대응기법을 제시하며, 동일한 실험환경에서 대응기법 적용 시 제안한 공격에 저항성을 가짐을 보인다. 5장에서는 4장에서 소개한 대응기법 적용 여부에 따른 모드별 Clock Cycle과 메모리 사용에 대한 오버헤드를 보인다. 6장에서는 본 논문의 결과를 요약하고 향후 연구에 대해 논의하며 결론을 맺는다.

II. 배경지식

2장에서는 SMAUG-T의 특징과 sparse ternary 개념, 복호화 연산 과정에서 발생되는 다항식 곱셈 구조, 및 비밀키 저장 방식에 대해 설명한다.

2.1 SMAUG-T 알고리즘

SMAUG-T는 격자 기반 KEM 알고리즘으로

KpqC 1라운드 후보 중 SMAUG와 TiGER를 결합한 알고리즘이다. 격자 중 모듈(module) 격자에서 정의되는 MLWE(module learning with error)와 MLWR(module learning with rounding) 문제를 사용하며 키 생성은 MLWE, 암·복호화는 MLWR를 기반한다. Fig.1.은 키 생성 알고리즘의 의사코드이다.

Output : pk =$(seed_A, b)$, sk = s
1. seed ← $\{0,1\}^{256}$
2. $(seed_A, seed_{sk}, seed_e)$ ← XOF$(seed)$
3. A ← expandA$(seed_A)$
4. s ← $HWT_{h_s}(seed_{sk})$
5. e ← $dGaussian_{\sigma}(seed_e)$
6. b = $-A^T \cdot s + e$
7. **return** pk = $(seed_A, b)$, sk = s

Fig. 1. Key generation pseudocode

line 6은 MLWE 구조를 생성하는 단계로 행렬 A와 비밀키 s, Error 값 e를 통해 공개키를 생성한다. Fig.2.는 복호화 알고리즘의 의사코드로 비밀키와 암호문을 통해 메시지를 반환한다.

Input : sk = s, c = (c_1, c_2)
Output : message = μ'
1. $\mu' = \lfloor t/p \cdot \langle c_1, s \rangle + t/p' \cdot c_2 \rceil$
2. **return** μ'

Fig. 2. Decryption pseudocode

line 1은 MLWR 연산 과정으로 t, p, p'을 활용해 비밀키 s와 암호문 c_1의 다항식 곱셈을 수행한다. SMAUG-T는 저사향 환경에서 동작하는 TiMer와 보안 강도에 따라 SMAUG-T128, SMAUG-T192, SMAUG-T256으로 구분되어 있다. Table 1.은 각 알고리즘 별 설정되는 파라미터를 설명한다.

Table 1. Parameter

sign	TiMer	SMAUG -T128	SMAUG -T192	SMAUG -T256
k	2	2	3	5
q	1024	1024	2048	2048
p'	8	32	256	64

2.2 Sparse ternary

SMAUG-T는 1라운드 SMAUG에서 사용하는 sparse ternary 개념을 유지한다. sparse ternary는 다항식 계수를 1, 0, -1로 구성하며 1과 -1의 개수가 주어진 해밍웨이트 만큼 할당하는 개념이다. Table 2.는 모드마다 부여되는 비밀키 정보로 k는 모듈 개수, h_s는 sparse 해밍웨이트 값이다.

Table 2. Sparse Hamming Weight

sign	TiMer	SMAUG -T128	SMAUG -T192	SMAUG -T256
k	2	2	3	5
h_s	100	140	198	175

SMAUG-T128의 비밀키는 2개의 256차 다항식으로 구성되어 있고, 512개의 계수 중 1과 -1의 개수는 140개이다. 두 다항식에 대한 sparse 값은 공개값이지만, 다항식 별 할당되는 sparse 값은 랜덤으로 할당되는 비밀값이다. SMAUG-T팀은 sparse ternary 구현을 위해 HWT (hamming weight sampler)를 직접 제안했으며, 해당 Sampler는 Dilithium[9]의 SampleInBall과 BIKE[10]의 CWW(constant weight word)를 조합하여 만들었다. 또한 sparse ternary 특징을 통해 다항식 저장 시, 1과 -1인 항의 차수만을 저장하여 키 크기를 줄이는 방식을 제안했다. 키 저장 시 계수가 1인 항의 차수를 배열에 먼저 저장한 후 -1인 항의 차수를 뒤이어 저장한다. Table 3.은 비밀키 저장 구조체 변수, Fig.3.은 비밀키 저장 의사코드다. sx는 차수 값이 저장되는 변수로 Fig.3.에서 res로 사용된다. 차수 저장 시 1에 대한 차수는 오름차순, -1에 대한 차수는 내림차순으로 저장되도록 구현되어 있다. neg_start는 Fig.3.의 반환 값으로 -1 에 대한 차수가 처음 저장되는 인덱스 값이다. Fig.4.는 변수 sx에 대한 하나의 예시이다.

Table 3. Variation of sparse ternary polynomial storage

Variation	Role
sx	Array to store the degrees
neg_start	index of first sotred -1 degree
cnt	length of array sx (Total number of 1 and -1)

```
Input : res, res_length, op, op_length
Output : neg_start
1.  index = 0
2.  arr = [0, res_lengh-1]
3.  for i from 0 to op_length - 1 do
4.     index = (op[i] & 0x80) >> 7
5.     b = (-(uint64_t)op[i]) >> 63
6.     t = (-b) & (res[arr[index]] ^ i)
7.     res[arr[index]] ^= t
8.     arr[index] += (int8_t)op[i]
9.  return arr[0]
```

Fig. 3. Sparse ternary polynomial storage pseudo code

$$x^2 - x^3 - x^5 + x^6 + x^7 + \cdots + x^{42} - x^{43} + x^{45}$$

coefficient	1	...	1	-1	...	-1
degree	2	...	45	43	...	3
sort method	Ascending			Descending		

Fig. 4. Example of Variation sx

2.3 비밀키 다항식 곱셈 연산

SMAUG-T는 sparse ternary 특징을 활용하여 기존 격자 기반 암호 알고리즘에서 사용하는 NTT 곱셈 연산 대신 빠르고 효율적인 곱셈 연산을 제안했다. Fig.5.는 새롭게 제안한 다항식 곱셈 연산의 의사코드이다. 해당 알고리즘은 2.2절에서 제안된 다항식 저장 방식으로 구현되었기 입력 값 b는 sparse ternary 다항식 구조이며, neg_start는 -1에 대한 차수가 처음 저장되는 인덱스이다. 결과 값은 a와 b를 곱한 결과 값에 reduce를 수행한 256차 다항식이 반환된다. 2.2절의 특성으로 곱셈 시 neg_start 전까지는 덧셈 연산을, neg_start 이후부터는 뺄셈 연산을 통해 곱셈 연산을 대체한다.

```
Input : a, b, neg_start
Output : c
1.  c = 0
2.  for i from 0 to neg_start - 1 do
3.     degree = b[i]
4.     for j from 0 to n - 1 do
5.        c[degree + j] = c[degree + j] + a[j]
6.  for i from neg_start to len(b) - 1 do
7.     degree = b[i]
8.     for j from 0 to n - 1 do
9.        c[degree + j] = c[degree + j] - a[j]
10. for j from 0 to n - 1 do
11.    c[j] = c[j] - c[n + j]
12. return c
```

Fig. 5. poly_mult_add function pseudocode

III. 다항식 곱셈 SPA 공격

3장에서는 ARM Cortex M4 환경에서 단일파형 SPA 분석을 통해 랜덤하게 부여되는 비밀키 정보를 복구하는 공격을 제안한다.

3.1 SPA 공격 방법

SMAUG-T의 비밀키는 sparse ternary 특성으로 인해 다항식 별 1과 -1의 개수가 랜덤하게 생성되어 비밀로 유지된다. 하지만, Fig.5.의 line 6에서 곱셈 연산량이 비밀키 배열 크기 만큼만 수행되기에 상수 시간 연산을 만족하지 않는다. 즉, 다항식 별 랜덤하게 할당되는 sparse 값에 따라 연산 시간이 일정하지 않다. 또한, line 5, 9를 통해 다항식 곱셈이 계수에 따라 덧셈과 뺄셈으로 구분되고 연산 사이의 소비전력 차이가 존재하기에 파형에서 연산 과정이 식별된다. 따라서 곱셈 단일 파형에 SPA 공격을 수행하여 랜덤하게 형성되는 다항식 별 해밍웨이트 정보를 식별할 수 있다.

3.2 실험 결과

본 논문의 실험 환경은 Fig.6.과 같이 CW308-UFO STM32F303 MCU 보드를 사용하여 알고리즘을 동작시켰고, ChipWhisperer-Lite 보드를 사용하여 전력 파형을 수집했다. 공격 대상 코드는 SMAUG-T팀의 Github[11] 3.0.2 버전 중 SMAUG

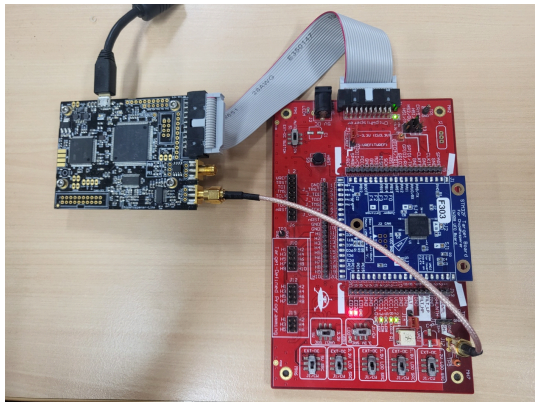


Fig. 6. Experimental environment

-T128 모드를 사용했다.

선택된 암호문과 키 한 쌍에서 비밀키 다항식 중 하나를 선택해 곱셈 연산 구간을 수집한 결과 142,164 포인트가 수집되었다. Fig.7.은 수집한 파형 전체를 나타낸다.

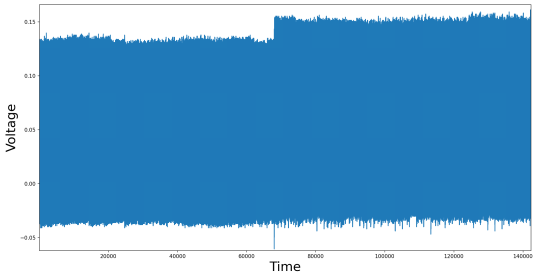


Fig. 7. Multiplication operation Power consumption trace for one polynomial

해당 파형 중 소비전력 차이가 식별되는 구간이 존재하며 확인 결과 덧셈은 67,987포인트, 뺄셈은 74,177포인트로 식별된다. Fig.8.은 덧셈과 뺄셈 함수가 1회씩 동작하는 동안 수집한 전력 파형이다.

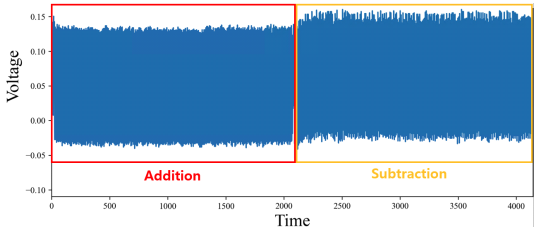


Fig. 8. Power comsunption trace when performing addition and subtraction once

해당 파형의 수집된 포인트는 4,144포인트로 SPA 결과 덧셈과 뺄셈 연산당 약 2,060포인트가 수집됨을 확인했다. 따라서 전체 142,164포인트를 2,060포인트로 나눈 값이 덧셈과 뺄셈 연산의 총 횟수이자 랜덤하게 생성되는 해밍웨이트 값이다. 실제 실험에서 선택한 비밀키의 해밍웨이트는 69이며, 비밀키 정보도 SPA 공격 결과와 일치함을 확인했다.

SPA 공격에 있어 명확한 수치를 기반으로 공격하고자 파형의 극대, 극솟값으로 소비전력 차이의 평균을 계산한 덧셈 연산에서는 약 0.162, 뺄셈 연산에서는 약 0.094 정도의 소비전력 차이가 발생함을 확인했다. 두 연산 사이의 소비전력 차이 원인은 Fig.9.와 같이 뺄셈 파형 사이의 미세한 소비전력

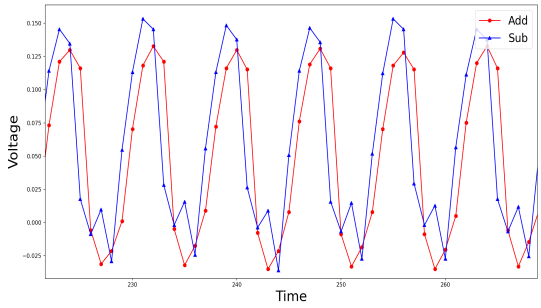


Fig. 9. Difference in power consumption between addition and subtraction

파형이 수집되기 때문이다. 수치화된 소비전력 차이를 기준으로 파형을 구분한 결과 Fig.10.과 같이 비밀키 정보가 노출된다.

Trace Num	Difference	Operation	Counts
1	0.16283428	Addition	33
...	...	...	
33	0.16100412	Addition	
34	0.09609608	Subtraction	36
...	...	...	
69	0.09339899	Subtraction	

Fig. 10. Power consumption SPA analysis results

IV. 다항식 곱셈 SPA 대응기법

4장에서는 다항식 곱셈 알고리즘의 비 상수 연산 시간을 상수 시간으로 수정하여 부채널 공격에 대응하는 대응기법을 제안한다.

4.1 대응기법 구조

Fig.5.에서 소개한 다항식 곱셈 알고리즘은 neg_start 전까지 1에 대한 덧셈, neg_start부터 배열 길이까지 -1에 대한 뺄셈 연산을 수행한다. 따라서 랜덤하게 생성되는 sparse값으로 연산 시간이 상수 연산 시간을 만족하지 못한다. 이를 해결하기 위해 연산 시 Dummy 연산을 추가하여 곱셈 연산이 상수 시간을 만족하도록 수정하였다. Fig.5.의 배열 c는 256차 곱셈 연산 값을 저장하기에 512 byte로 할당된다. Dummy 연산의 경우 결과 값에 영향을 주면 안되기에 256byte를 추가로 할당하여 결과 값을 저장한다. Dummy 연산을 포함한 연산별 고정 횟수는 sparse 해밍웨이트 값에 영향을 받기에 모드에 따라 변동된다. SMAUG-T128의 경우

sparse 해밍웨이트는 140이기에 연산별 고정 횟수는 각 140회씩 수행된다. Table 4.는 모드별 덧셈과 뺄셈의 고정 연산 값을 나타낸다.

Table 4. Fixed number of operations

Mode	TiMer	SMAUG-T128	SMAUG-T192	SMAUG-T256
h_s	100	140	198	172
Fixed number	100	140	200	176

SMAUG-T192와 SMAUG-T256은 상수 시간 연산을 위해 수정될 비밀키 구조 특성으로 고정 횟수가 각각 200, 178로 설정된다. 비밀키 구조 수정에 관련한 내용은 4.2.1항에서 자세히 설명한다.

```
Input : a, b, neg_start
Output : c
1.  c = 0
2.  for i from 0 to HS_O - 1 do
3.      sign_bit = (b[HS_O*2+i/8]>>(7-i%8))&1
4.      degree = b[i] + 512 * (sign_bit ⊕ 1)
5.      for j from 0 to n - 1 do
6.          c[degree + j] = c[degree + j] + a[j]
7.  for i from HS_O to HS_O * 2 - 1 do
8.      sign_bit = (b[HS_O*2+i/8]>>(7-i%8))&1
9.      degree = b[i] + 512 * (sign_bit ⊕ 1)
10.     for j from 0 to n - 1 do
11.         c[degree + j] = c[degree + j] - a[j]
12. for j from 0 to n - 1 do
13.     c[j] = c[j] - c[n + j]
14. return c
```

Fig. 11. poly_mult_add function pseudocode with countermeasure

Fig.11.은 Dummy 연산을 기반으로 상수 시간을 만족시킨 알고리즘이다. Fig.11.의 입력 값 a, b는 다항식 값이며, 수정된 비밀키는 b로 입력된다. HS_O 값은 Table 4.에서 소개한 Fixed number 값을 의미하며, line 2, 7의 반복문은 각 HS_O번 수행하여 상수 시간을 만족시켰다. 또한 별도로 저장할 Dummy 연산을 위해 offset 전달 시 비밀키에 저장되어 있는 구분자 bit값을 시프트 연산을 사용하여 Dummy는 1을, 1과 -1일 경우 0을 반환한다. 이후 offset 전달 시 bit 반환 값에 512를 곱한 값을 더하여 설정한다. 그 결과, Dummy는 512, 그 외는 차수 값이 반환되며 이는 line 3-4, 8-9에 구현하였다. 구분자 bit에 대한 자세한 설명은 4.2.1항에서 설명한다.

4.2 대응기법 구현

4.2절에서는 4.1절의 대응기법을 구현하기 위한 비밀키 구조 수정과 차분 축소를 위해 수정될 비밀키 생성 과정을 소개한다.

4.2.1 비밀키 저장 구조

비밀키 구조 수정은 다항식 곱셈의 상수 시간을 만족하기 위해 필요한 과정이다. 기존 비밀키 구조는 0부터 neg_start-1까지 1에 대한 차수, neg_start부터 len(b)-1까지 -1에 대한 차수 값이 저장된다. 4.1절에서 상수 시간을 위해 Dummy 연산을 추가하였기에 비밀키 저장 시 1과 -1의 차수 뿐만 아니라 Dummy 만큼의 0 값이 할당되어야 한다. SMAUG-T128 기준 덧셈과 뺄셈이 각 140회씩 수행되어야 하기에 배열 크기 280byte이다. Dummy를 위한 0 값은 1과 -1 차수 사이에 저장하여 연산 사이에 Dummy 연산을 수행한다. Fig.12.는 Fig.4.의 SMAUG-T128 비밀키의 수정 구조이다.

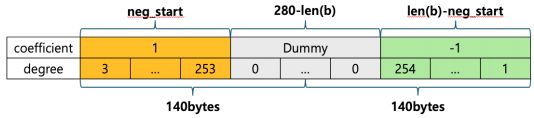


Fig. 12. Modified array structure based on SMAUG-T128

또한, Dummy 연산은 offset을 512로 할당하여 별도로 저장해야 한다. 따라서 Dummy와 1, -1을 구분하는 별도의 데이터 구조가 필요하고 해당 데이터는 다항식 배열 뒤에 추가로 할당한다. 차수 값마다 1bit를 부여하여 Dummy의 경우 0, 1 혹은 -1일 경우 1bit로 표기하여 저장한다. SMUAG-T128 기준 배열크기는 280byte이기에 구분자 표기를 위해 35byte가 할당되기에 비밀키 크기는 315 byte이다. Fig.13.은 SMAUG-T128을 기준으로 배열의 차수 정보에 맞게 bit 표기 배열 값을 나타낸 예시이다.

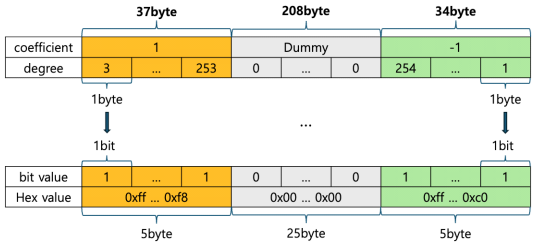


Fig. 13. Polynomial array structure of SMAUG-T128

Table 4.의 SMAUGT-192, SMAUG-T256 고정 횟수가 sparse 값과 다른 이유는 구분자 표기 배열 크기가 8배수를 만족하지 않기 때문이다. Table 5.는 모드별 다항식 한 개를 저장할 때 필요한 배열 크기를 나타낸다.

Table 5. Array size of one secret key polynomial for each mode (Unit : Byte)

Mode	TiMer	SMAUG-T128	SMAUG-T192	SMAUG-T256
byte	225	315	450	396

4.2.2 추가 취약점 완화

4.1절의 대응기법으로 연산의 상수 시간을 만족하여도 각 연산과 Dummy 연산이 식별가능할 경우 취약점은 여전히 발생한다. 따라서 덧셈, 뺄셈 연산과 Dummy 연산 사이의 차분이 없이 동일하게 수행되어야 한다. Fig.12.는 대응기법이 적용된 후의 비밀키 구조이다. Dummy 연산과 기존 연산 사이의 계수는 덧셈의 경우 253, 뺄셈의 경우 254로 Dummy 계수인 0과의 해밍웨이트 차이가 존재한다. Fig.13.은 임의로 작성된 예시이지만, 2.3절에서 알 수 있듯 계수 저장 시 1은 오름차순, -1은 내림차순으로 저장되기에 Dummy 연산 사이의 해밍웨이트 차가 클 수밖에 없다. 해밍웨이트 차이가 발생될수록 SPA 공격을 통해 Dummy 연산와 기존 연산을 구분이 쉬워진다. 따라서 계수의 해밍웨이트 차를 줄이고자 비밀키 저장 시 1은 내림차순, -1은 오름차순으로 저장한다. Fig.14.는 기존 Fig3.의 line 6를 수정하여 차수 저장 정렬을 변경하였고, line 8, 9를 통해 4.2.1항에서 언급한 bit 표기 배열 값을 저장하도록 수정하였다.

다항식 곱셈 알고리즘과 비밀키 생성 코드 외에도 비밀키 구조와 문자열 사이의 전환 코드, 덧셈과 뺄셈 코드 등을 변화된 비밀키 구조에 맞게 수정해야 한다. [12]는 해당 함수에 대한 자세한 코드를 나타낸다.

```
Input : res, res_length, op, op_len
Output : neg_start
1.  index = 0
2.  arr = [0, res_lengh-1]
3.  for i from 0 to op_len - 1 do
4.      index = (op[i] & 0x80) >> 7 & 0x01
5.      b = (-(uint64_t)op[i]) >> 63
6.      t = (-b)&(res[arr[index]])^(op_len-(i+1))
7.      res[arr[index]] ^= t
8.      t = (b & 1) << (7 - (arr[index] % 8))
9.      res[(arr[index] / 8) + HS_Q * 2] |= t
10.     arr[index] += (int8_t)op[i]
11. return arr[0]
```

Fig. 14. Modified Sparse ternary polynomial storage pseudocode

4.3 대응기법 SPA 안전성 검증

본 절에서는 4.2에서 제안한 대응기법이 3장에서 제안한 공격의 대응되는지 동일한 환경에서 검증 실험을 수행하였다. Fig.15.는 SMAUG-T128 기준으로 대응기법 적용 후 다항식 한 개에 대한 곱셈 연산 파형이다.

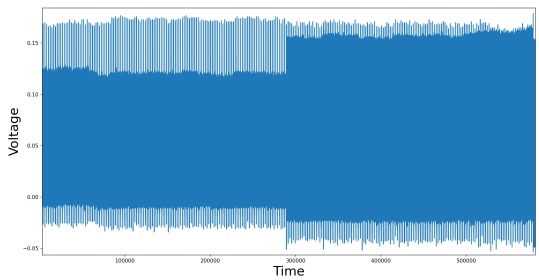


Fig. 15. Multiplication power consumption trace for one polynomial after applying the counter-measure

Dummy 연산으로 인해 수집 포인트 수는 기존 연산보다 약 430,000 포인트 증가하였다. 전체 포인트 수에 연산 당 포인트 수인 2,060을 나눈 결과 280으로 SMAUG-T128 다항식 배열 크기와 동일한 것을 확인할 수 있다.

대응기법 적용 후, 덧셈과 뺄셈 연산의 차분은 여

전히 구별 가능하지만 각 연산별 수행 횟수는 140회로 동일하기에 계수별 개수를 파악하기는 힘들다.

4.2.2항에서 언급했듯 덧셈, 뺄셈 연산과 Dummy 연산의 파형 차분이 발생하면 안되기에 파형 비교를 수행하였다. Fig.16.은 덧셈 연산과 덧셈 Dummy 연산의 파형을 나타낸다.

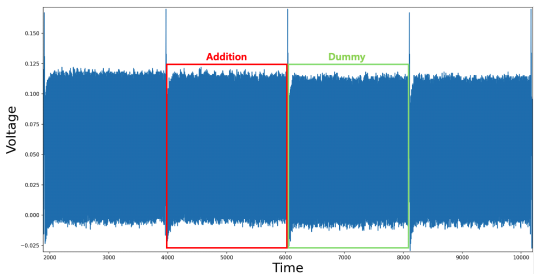


Fig. 16. Difference between addition and addition Dummy operation power conumption trace

두 연산의 파형을 비교해본 결과 다른 파형들과의 편차와 유사하기에 육안으로 판별하기 힘들다.

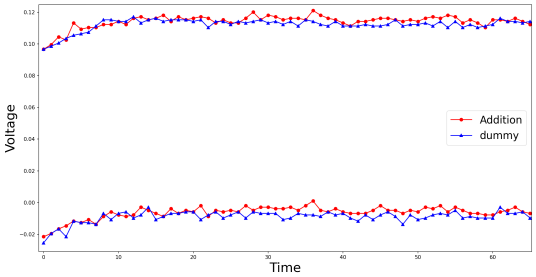


Fig. 17. local extremum point of power conumption trace overlapped with Dummy in addition

Fig.17.은 덧셈에서 Dummy 파형과의 비교를 위해 각 파형의 극점을 표시한 그래프이다. 그래프 확인 결과 덧셈 파형의 극점이 다소 더 높게 보이지만 해당 특징은 다른 덧셈 연산 간에서도 보이는 편차이기에 Dummy 파형과 덧셈 파형을 구분짓기는 어렵다.

Fig.18.은 뺄셈 연산과 뺄셈 Dummy 연산의 파형이고 Fig.19.는 Fig.18.에서 확인한 두 연산의 파형 극점을 함께 표시한 그래프이다.

뺄셈 연산 역시 파형 개요를 기반으로 연산을 구분 짓기는 어렵기에 개수를 파악이 어렵다. 실제 기존 SPA 공격에 사용되었던 공격 코드를 수행한 결과 Fig.20.과 같이 덧셈과 뺄셈 개수가 동일하게 측정되어 SPA 공격에 저항성을 가짐을 알 수 있다.

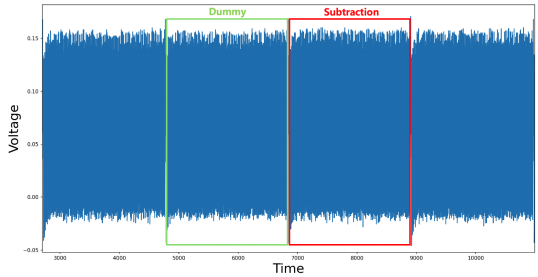


Fig. 18. Difference between subtraction and subtraction Dummy operation power conumption trace

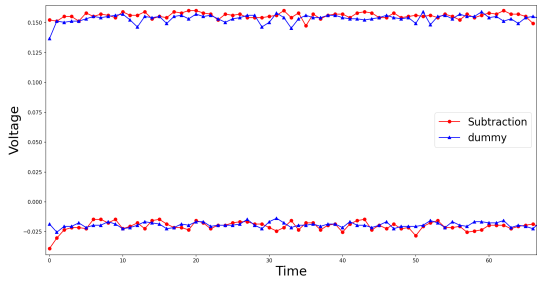


Fig. 19. local extremum point of power consumption trace overlapped with Dummy in subtraction

Trace Num	Difference	Operation	Counts
1	0.08006493	Addition	140
...	...	...	
140	0.08033981	Addition	
141	0.16051251	Subtraction	140
...	...	...	
280	0.16796501	Subtraction	

Fig. 20. Power consumption SPA analysis results after applying countermeasure

V. 성능 평가

본 절에서는 4절에서 소개한 대응기법 여부에 따른 알고리즘의 성능 평가를 비교하고 시간과 메모리 측면에서의 오버헤드를 설명한다. 본 논문에서 소개한 대응기법은 SMAUG-T의 키 생성과 복호화 과정에서 적용되기에 해당 과정의 오버헤드만 설명한다. 성능 측정 환경은 CPU 8코어, GPU 10코어, RAM 16GB의 Apple Silicon M3에서 수행하며, Clock Cycles 측정은 SMAUG-T에서 제공하는 Benchmark 코드를 사용하였다. 4.2절에서 소개한

Table 6. Median Clock Cycle counts of 10000 executions for SMAUG-T

Mode	Operate	Clock Cycles		Overhead(%)
		Classic	CounterMeasure	
TiMER	Key Gen	26,227	26,388	0.61
	Decap	11,269	12,910	14.56
SMAUG-T128	Key Gen	26,933	26,899	-0.12
	Decap	10,393	13,666	31.49
SMAUG-T192	Key Gen	46,990	46,546	-0.94
	Decap	16,222	24,290	49.73
SMAUG-T256	Key Gen	94,741	96,606	1.96
	Decap	29,907	42,586	42.39

대응기법은 Dummy 연산을 위해 비밀키 구조의 크기를 증가시며, 모드에 따라 설정되는 다항식 한개의 크기는 Table 5.에서 확인할 수 있다.

Table 6.은 SMAUG-T의 모드별 대응기법 적용 여부에 따른 키 생성과 복호화 시 평균 Clock Cycles이다. 성능 측정 결과 모든 모드에서 키 생성의 경우 오버헤드는 존재하지 않는다. 키 생성 과정에서는 키 크기만 증가할 뿐 추가 연산은 수행되지 않기에 기존 연산과 유사한 Clock Cycles을 보인다. 하지만 복호화의 경우 Dummy 연산이 수행되기에 TiMER의 경우 약 15%, SMAUG-T128에서는 약 31%, SMAUG-T192, 256에서는 각각 약 50%, 42%의 오버헤드가 발생된다. SMAUG-T128 대비 192와 256에서는 비밀키 다항식 개수가 하나 더 많기에 복호화 Clock Cycles 오버헤드가 더 크게 발생된다. 또한 SMAUG-T192, SMAUG-T256에서는 sparse 값이 더 작은 SMAUG-T256에서 더 낮은 오버헤드가 측정된다.

메모리 관점에서는 Dummy 저장을 위한 비밀키 크기만 증가할 뿐, 공개키나 알고리즘 측면에서 오버헤드는 존재하지 않는다. 대응기법 적용 시, 4.2.1항에서 설명했듯 개인키의 sparse 성질을 고려하여 메모리를 할당해야 하고, Dummy 구분자를 저장할 메모리 역시 추가로 할당되어야 한다. Table 7.은 모드별 대응기법 적용 여부에 따른 비밀키 크기와 오버헤드를 나타낸다.

Table 7. Maximum Secret Key memory Usage to SMAUG-T for each mode

Mode	Byte		Overhead (%)
	Classic	Counter Measure	
TiMER	100	450	350
SMAUG-T128	140	630	350
SMAUG-T192	198	1350	581.8
SMAUG-T256	172	1188	590.6

Table 5.는 대응기법 적용 여부에 따른 비밀키 다항식 한 개의 크기이고, SMAUG-T128 기준 비밀키 크기는 315byte이다. 따라서 다항식 개수는 2개이기에 대응기법을 적용한 비밀키 크기는 630byte이다. 다른 모드에서도 동일한 방법으로 측정한 결과 TiMER와 SMUAG-T128는 약 350%, SMAUG-T192는 약 581%, SMAUG-T256에서는 약 590%의 오버헤드가 존재한다.

본 대응기법은 비밀키의 sparse 분포가 균일하지 않은 환경까지 고려한 크기이다. 따라서 sparse 분포를 고려한다면 Clock Cycles과 키 크기가 현재보다 줄어들 것으로 예상된다.

VI. 결 론

본 논문은 KpqC 2라운드 PKE/KEM 알고리즘 중 SMAUG-T 복호화의 다항식 곱셈 연산이 비밀키에 따른 연산 순서를 수행하는 취약점을 제시했다. 이 취약점을 이용해 복호화 전력 파형의 SPA를 통해 비밀키의 일부 정보를 복구하는 부채널 공격을 소비전력 제안한다. 추가적으로 제안한 SPA 공격에 대한 대응기법과 그에 따른 오버헤드를 제시하며 실험을 통해 제안한 공격에 저항성을 가짐을 증명한다. 그러나 본 논문에서 제안한 공격은 다른 다항식 곱셈 방법에서는 적용할 수 없으므로 다항식 곱셈 연산 구조를 변경함으로써 대응하는 것도 가능하다. 향후 제안한 대응기법의 최적화 연구와 다항식 곱셈 방법을 변경하여 취약점을 제거하는 방향의 연구가 필요하다.

References

[1] A. Karlov and N.L de Guertechin, "Power analysis attack on Kyber,", IACR Cryptology ePrint Archive 2021-1311, Sep. 2021.

[2] R. Primas, P. Peter and S. Mangard, "Single-trace side-channel attacks on masked lattice-based encryption," Cryptographic Hardware and Embedded Systems-CHES 2017, pp. 25-28, Sep. 2017.

[3] P. Pessl and R. Primas, "More practical single-trace attacks on the number theoretic transform," Progress in Cryptology - LATINCRYPT 2019, pp. 2-4, Oct. 2019.

[4] L. Backlund, K. Ngo, J. Gärtner and E. Dubrova, "Secret key recovery attack on masked and shuffled implementations of CRYSTALS-Kyber and Saber," Applied Cryptography and Network Security, pp. 159-177, Oct. 2023.

[5] Z. Xu, O. Pemberton, S.S. Roy, D. Oswald, W. Yao and Z. Zheng, "Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber," IEEE Transactions on Computers, vol. 71, no. 9, pp. 2163-2176, Sep. 2022.

[6] M. Hamburg, et al, "Chosen ciphertext k-trace attacks on masked CCA2 secure Kyber,", IACR Transactions on Cryptographic Hardware and Embedded Systems 2021-04, Aug. 2021.

[7] Jeong-hwan Lee, Gyu-sang Kim, Hee-seok Kim, and Seok-hie Hong, "A key recovery side-channel attack on KpqC 1 round candidate SMAUG," CISC-S'23, pp. 553-556, June 2023

[8] Jung-hee Cheon, et al, "SMAUG-T: the Key Exchange Algorithm based on Module-LWE and Module-LWR" KPQC, 2024.

[9] L. Ducas, et al, "CRYSTALS-Dilithium: A lattice-based digital signature scheme,", IACR Transactions on Cryptographic Hardware and Embedded Systems 2018-01, Feb. 2018.

[10] N. Sendrier, "Secure sampling of constant-weight words-application to bike,", IACR Cryptology ePrint Archive 2021-1631, Dec. 2021.

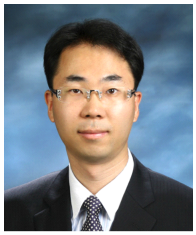
[11] Github, https://github.com/hmchoe0528/SMAUG-T_public.git, Jan. 2025.

[12] Github, https://github.com/ISCMSHY/SMAUG-T-SCA-CounterMeasure.git, Jan. 2025.

〈저자소개〉

유 성 환 (Sung-Hwan Yoo) 학생회원
2019년 3월~현재: 국민대학교 정보보안암호수학과 학사과정
〈관심분야〉 정보보호, 양자 내성 암호, 부채널 분석 및 대응법 설계

한 재 승 (Jaeseung Han) 학생회원
2020년 2월: 국민대학교 정보보안암호수학과 학사
2022년 2월: 국민대학교 금융정보보안학과 석사
2022년 3월~현재: 국민대학교 금융정보보안학과 박사과정
〈관심분야〉 부채널 분석 및 대응법 설계, 오류 주입 공격, 대칭키 암호, 양자 내성 암호



한 동 국 (Dong-Guk Han) 종신회원
1999년 2월: 고려대학교 수학과 학사
2002년 2월: 고려대학교 수학과 이학석사
2005년 2월: 고려대학교 정보보호대학원 공학박사
2004년 4월~2005년 4월: 일본 Kyushu Univ., 방문연구원
2005년 4월~2006년 4월: 일본 Future Univ.-Hakodate, Post.Doc.
2006년 6월~2009년 2월: 한국전자통신연구원 정보보호연구단 선임연구원
2009년 3월~현재: 국민대학교 정보보안암호수학과 정교수
〈관심분야〉 공개키 암호시스템 안전성 분석 및 고속 구현, 부채널 분석 및 대응법 설계, IoT 정보보호 기술