동형암호에 대한 부채널 공격과 대응에 관한 연구

남기빈[1], 주유연[1], 하승진[1], 백윤흥[1]
[1]서울대학교 전기정보공학부, 서울대학교 반도체 공동연구소
{kvnam, yyjoo, sjha}@sor.snu.ac.kr, ypaek@snu.ac.kr

Side-Channel Attacks on Homomorphic Encryption and Their Mitigation Methods

Kevin Nam[1], Youyeon Joo[1], Seungjin Ha[1], Yunheung Paek[1]
[1]Dept. of Electrical and Computer Enginering and Inter-University Semiconductor Research Center(ISRC), Seoul National University

요 약

동형암호는 주목받는 차세대 프라이버시 보존 기술이다. 많은 기업들이 이를 활용한 서비스들을 제공하고 있다. 비록 동형암호가 수학적으로 안전성을 인정받았지만, 실행되는 프로그램으로써 동형암호는 부채널공격들에 취약하다는 연구 결과들이 보고되고 있다. 이 논문은 이런 부채널공격들에 대해 본석, 일반화하여 사용 가능한 gadget을 소개하며, 대응기법에 대한 가이드라인을 제안하고 그 효과와 한계에 대해 분석한다.

1. 서론

동형암호는 데이터를 암호화 상태로 연산할 수 있는, 주목받고 있는 차세대 프라이버시 보존 기술이다[1]. 많은 기업들이 이런 동형암호를 활용하여 여러 분야에 적용하려는 시도들을 이어왔으며, 이를 위해 보편적으로 사용할 수 있는 여러 라이브러리들을 만들어 배포하고 있다. 비록 동형암호는 수학적으로 증명된 우수한 보안성을 입증받았지만, 프로그램으로써 실행과정 중 민감한 정보의 유출에 대한 우려는 남아있다. 특히, 최근 동형암호 라이브러리들에 대한 부채널공격들을 다루는 연구들이 등장했다. 시간정보(timing), 전력(power) 등을 활용한 다양한 연구들이 소개되었으며, 완전히 성공한 경우는 없었으나, 공격들에 대한 위험성을 충분히 보여줬음에 의미를 지니고 있다.

이 논문은 동형암호에 대한 부채널공격들을 소개하고, 일반화하여 사용할 수 있는 부채널 공격 도구(gadget)를 소개하며 이에 대한 대응책에 대한 가이드라인을 제시할 것이다.

2. 이론적 배경

2.1 동형암호의 암호문 및 연산 구성

동형암호 체계들[2,3]은 큰 다항식 형태의 암호문을 활용한다. 이 때, 암호체계의 동형성을 유지하기 위해, 각 다항식들을 정해진 환(Ring) 안에 유지하는 Ring Theory를 활용하는데, 다항식에 대한 환은 연산 결과를 특정 다항식으로 나누고, 각 계수를 특정 정수로 나눈 나머지들로 유지하는 것으로 그 과정이 이루어진다. [그림 1]은 Ring $Z_{13}[x]/<x^4+1>$가 적용된 연산의 예시다. 실제 연산 결과는 6차 다항식이 나오지만, 이를 x^4+1으로 나누어 3차 이하로 유지한다. 또한 상수항을 살펴보면, 111이 나오나, 이를 13으로 나눈 나머지인 7로 만든다. 이 곱하고 나누는, 혹은 더하고 나누는 연산을 mult-mod, add-mod라고 부른다.

$$\begin{array}{ll} & (4x^3+x^2+11x+10) \\ \times & (6x^3+9x^2+11x+11) \\ + & (-x^2+1x+1) \\ \hline & (10x^3+5x^2+10x+7) \end{array}$$

그림 1 Ring Theory가 적용된 다항식 연산 예시

2.2 부채널 공격

프로그램과 동작하는 기기에 대한 시간, 온도, 전력 등 간접 정보들을 부채널이라고 한다. 부채널들을 활용해서 유출되지 않아야 하는 비밀에 대한 정보를 알아내는 공격을 부채널공격이라고 한다. 이때 공격자는 공격 대상 동작하는 기기의 부채널들에 접할 수 있다는 전제를 바탕으로 하며, Meltdown[4]이 하나의 예시이다.

라이브러리명	배포사	암호체계
SEAL	Microsoft	BFV, CKKS
HELib	IBM	BGV, BFV
HEaaN	CryptoLab	CKKS

표 1 동형암호 라이브러리 종류

2.3 동형암호에 대한 부채널 공격 사례

동형암호 활용에 있어 프라이버시 유출은 암호화/복호화 과정에 쓰이는 사용자의 비밀키가 유출되는 것을 의미한다. 비밀키 역시 암호문들과 동일하게 다항식 형태를 갖고, 암/복호화는 이런 다항식들의 곱셈, 덧셈으로 이루어진다. 랜덤한 다항식 A, 비밀키 s, 그리고 보안성을 강화해주는 noise(error) 다항식 e와 암호화 대상 m이 있을 때, 암호문 c는 $c=(A,B)=(A,A\cdot s+m+e)$로 나타낼 수 있다. 이때 역시 mult-mod, add-mod 연산들이 활용된다. 복호화는 반대로, c가 주어졌을 때, A와 s를 곱해서 B에서 빼서 m+e를 추출하는 것으로 이루어진다. 이때 e를 정확히 제거하는 과정이 필요하지만, 이에 대해서는 동형암호 이론 논문들을 참고하길 바란다[1-3].

공격자는 이런 암호화, 복호화 과정에 연루되는 비밀키 s에 대한 정보를 추출하는 것을 목표로 한다. 여러 공격 연구들이 있었는데, 이들은 공통적으로 공격 대상 동형암호 라이브러리들은 <표1>에 나와있으며, 사례들은 다음과 같다. Aydin et.al.[5]은 SEAL에서 비밀키의 계수들이 -1,0,1들로 이루어져 있음에 주목했다. 암호화, 그리고 바로 직후 과정에서 곱셈중 발생하는 power trace를 분석하는 AI모델을 활용하여, 암호키의 각 계수가 무엇이었는지 맞추는 반복적인 공격을 통해, 98.3%의 정확도로 추출에 성공하였다. 하지만, AI모델에 대한 정보가 공개되지 않았고, 10,000,000회 이상의 공격을 시도한다는 점에서 현실성이 떨어지는 공격이다.

Cheng et.al.[6]은 같은 상황에서, mult-mod에서 발생하는 분기문의 횟수를 측정하는 timing 부채널 공격을 시도했다. 이들 역시 무수히 많은 공격시도가 필요하지만, 공격 성공의 필요충분조건을 증명했다는 점에서 의미가 있는 연구이다. 유사하게, Pal et.al.[7]은 여러 라이브러리들에 대한 timing 부채널 공격을 실시했으며, 그 결과 비밀키에 대한 정보 일부를 추출할 수 있음을 증명하였다. 이 외에도 많은 연구들이 존재하며, 이들에 대한 공통점과 일반화된 공격 기법에 대해 다음 섹션에서 설명하겠다.

3. 일반화된 동형암호에 대한 부채널 공격 및 대응

3.1 일반화된 공격 가젯(gadget)

언급된 공격 사례들은 공통적으로, mult-mod를

```
1 void RingMultiplier::mulModBarrett{
2   unsigned __int128 mul = (a) * b;
3   uint64_t abot = (mul);
4   uint64_t atop = (mul >> 64);
5   unsigned __int128 tmp = (abot) * pr;
6   tmp >>= 64;
7   tmp += (atop) * pr;
8   tmp >>= kbar2 - 64;
9   tmp *= p;
10  tmp = mul - tmp;
11  r = static_cast<uint64_t>(tmp);
12  if(r >= p) r -= p;
13 }
```

그림 2 HEaaN의mult-mod 코드

활용한다는 점이 있다. 이를 대상으로 하는 이유는 연산의 구조적 특징에 있다. [그림 2]는 HEaaN의 mult-mod코드이다. 줄 12는 mod 연산 수행에 필수적인 분기문이다. 이 분기문이 몇 번 발생하는지 여부를 계산하여, A를 하는 공격자는 s에 대한 대략적인 범위를 알 수 있다는 점을 활용하여 공격하는 것이다. 이 gadget이 공격에 활용될 수 있음은 매우 치명적인데, 모든 동형암호 알고리즘 전반적인 연산들이 mult-mod를 활용하기 때문이다. 즉, 공격자는 거의 무한한 부채널 공급원을 바탕으로, 시간만 충분하다면 공격 성공률이 매우 높을 수 밖에 없다. 따라서 이 gadget에 대한 대응 기법들은 필수적이다. 다음 섹션들은 이 논문에서 제안하는 대응 기법들에 대해 소개한다.

```
1 void RingMultiplier::mulModBarrett{
2   unsigned __int128 mul = (a) * b;
3   uint64_t abot = (mul);
4   uint64_t atop = (mul >> 64);
5   unsigned __int128 tmp = (abot) * pr;
6   tmp >>= 64;
7   tmp += (atop) * pr;
8   tmp >>= kbar2 - 64;
9   tmp += p;
10  tmp = mul - tmp;
11  r = static_cast<uint64_t>(tmp);
12  r -= 0.5*(sign(r-p)+1)*p;
13 }
```

(a) constant-time

```
1 void RingMultiplier::mulModBarrett{
2   unsigned __int128 mul = (a) * b;
3   uint64_t abot = (mul);
4   uint64_t atop = (mul >> 64);
5   unsigned __int128 tmp = (abot) * pr;
6   tmp >>= 64;
7   tmp += (atop) * pr;
8   tmp >>= kbar2 - 64;
9   tmp += p;
10  tmp = mul - tmp;
11  r = static_cast<uint64_t>(tmp);
12  wait(random(seed));
13  if(r >= p) r -= p;
14  wait(random(seed));
15 }
```

(b) random delay

그림 3 제안하는 대응 기법들

3.2 대응기법I - constant-time programming

[그림 3-a]의 줄 12 수정사항은 기존 분기문을 항상 실행되는 코드로 수정하여 수행시간이 항상 일정하도록 군현할 경우, timing 부채널공격으로부터 방어할 수 있다는 점을 고려한 대응 기법이다. 이는 또한 power trace의 길이를 측정하는 power 부채널 공격에 대한 대응도 가능하여, 여러 부채널공격들로부터 내성을 갖추어주는 기법이지만, 전반적인 수행시간을 증가시킨다는 점에서 성능부하가 발생한다.

3.3 대응기법II - random delay

[그림 3-b]의 줄 12와 줄 14의 수정사항은 random한 delay를 추가하여 수행시간을 감추는 기법이다. constant-time 기법과 같이 timing 부채널공격과 이와 관련 있는 power 부채널공격 등에 대한 내성을 갖추게 해주지만, random이라는 함수의 특

성이 공개되는 등 상황에 대한 취약점이 존재한다.

10 M 공격횟수	baseline	대응기법I	대응기법II
timing부채널공격 (A)	97.1%	측정 불가	61.2%
power부채널공격 (B)	95.1%	측정불가	71.3%
Hamming 공격 (C)	94.8%	98.7%	87.5%
추가 성능 부하	x	16.4%	6.9%

표 2 : 10 M회 공격시도에 대한 성공률 및 성능부하 (M=Million)

시도횟수	baseline	대응기법I	대응기법II
10 Million	97.1%	측정 불가	61.2%
100 Million	97.1%	측정 불가	72.6%
1 Billion	97.1%	측정 불가	84.4%

표 3: 시도횟수 증가에 따른 timing 부채널공격의 성공률

4. 대응기법의 효과 및 한계

제안한 대응기법들의 영향을 분석하기 위해, intel Xeon-Gold 6326와 DRAM 1TB가 탑재된 서버에서 HEaaN 라이브러리를 활용하여 암호화 과정을 공격 대상으로하는 비밀키 추출실험을 수행하였다. 총 10M회 시도를 했으며 그 결과는 <표2>와 <표3>과 같다. 이때 공격 성공률은, 비밀키 다항식의 길이 대비 추출에 성공한 계수의 숫자 비율이다(1000차 다항식에서 500개 추출 성공시 50% 성공률). Power 부채널공격의 경우 수학적 모델링을 통한 power 시뮬레이션 python프로그램을 활용했다. Hamming 공격(C)는, power 측정 후 발생하는 전력 진폭의 hamming weight 거리를 분석하는 부채널 공격이다.

대응기법I은 timing 부채널공격에 대해 완벽한 내성을 지니고 있다(측정 불가). 이는 항상 수행시간이 동일하여 입력에 대한 출력 패턴을 분석할 수 없기 때문이다. 하지만 (C)에 대해 취약한데, 이는 [그림 3-a]에 나온 sign함수에 따른 hamming weight 거리가 더 두드러지게 나타나서 오히려 새로운 취약점으로 작용했기 때문이다. 총 성능부하는 16.4%가 발생했다. 대응기법II는 비교적 적은 성능부하로 그 효율성을 보였으나, 모든 부채널공격에 대해 61.2% 이상의 성공률을 허용하였다. 대응기법I에 비해(C)에 대한 내성은 조금 더 좋았다. 하지만, <표3>에 나타나듯, 공격 횟수가 증가하면, 내성이 떨어진다는 단점을 지니고 있다. 이는 random함수의 패턴이 공격 횟수 증가와 함께 노출되어 평균적으로 baseline과 비슷해지기 때문이라고 판단할 수 있다.

5. 결론

동형암호에 대한 부채널공격들을 분석해본 결과, mult-mod라는 gadget을 활용한다는 점을 파악했으며, 이에 해당 gadget에 대한 대응기법들을 제안하고 이들의 효과에 대해 알아보았다. 두 대응기법은 비교적 간단했으나, 각기 서로 다른 부채널공격들에 대해 뛰어난 효과를 보이기도 했으나, 또 서로 다른 점에서 약점을 보이기도 했다.

동형암호는 그 시장규모를 키워가며 차세대 암호 체계로 성장하고 있다. 하지만 이런 취약점들에 대한 충분한 대응책이 마련되지 않는다면, 치명적인 프라이버시 유출로 이어질 수 있다. 이 논문이 소개한 기법들의 장점을 살리고 서로의 단점을 극복하는 등 복합적인 대응기법의 개발을 통해 다양한 부채널 공격으로부터 안전히 지켜주는 연구가 이루어져야 할 것이다.

6. ACKNOWLEDGEMENT

이 논문은 2023년도 BK21 FOUR 정보기술 미래인재 교육연구단, 반도체 공동연구소 지원의 결과물이며, 연구 수행에 있어 2023년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받았으며(NRF-2020R1A2B5B03095204) 정보통신기획평가원의 지원을 받아(No.2021-0-00528, 하드웨어 중심 신뢰계산기반과 분산 데이터보호박스를 위한 표준 프로토콜 개발) 수행된 연구이다. 또한, 연구장비를 지원하고 공간을 제공한 서울대학교 컴퓨터연구소에 감사드린다.

참고문헌

[1] GENTRY, Craig. “Fully homomorphic encryption using ideal lattices”, ACM symposium on Theory of computing. 2009. p. 169-178.

[2] CHEON, Jung Hee, et.al. “Homomorphic encryption for arithmetic of approximate numbers”, ASIACRYPT 2017, Hong Kong, China, December 3-7, 2017, p. 409-437.

[3] CHILLOTTI, Ilaria, et.al. “TFHE: fast fully homomorphic encryption over the torus”, Journal of Cryptology, 2020, 33.1: 34-91.

[4] LIPP, Moritz, et.al. “Meltdown”, arXiv preprint arXiv:1801.01207, 2018.

[5] AYDIN, Furkan et.al. “Exposing Side-Channel Leakage of SEAL Homomorphic Encryption Library”, Workshop on Attacks and Solutions in Hardware Security. 2022. p. 95-100.

[6] CHENG, Wei, et.al. “Cache-Timing Attack on the SEAL Homomorphic Encryption Library”, 11th International Workshop on Security Proofs for Embedded Systems2022 (PROOFS 2022).

[7] PAL, Asmita, et al. Characterizing Memory Side Channels in FHE Applications.