암호문 부채널 공격 및 대응 방안에 관한 연구

황윤성[1], 유준승[1], 강기봉[2], 백윤흥[3]
[1]서울대학교 전기정보공학부 석박통합과정, 반도체공동연구소
[2]서울대학교 전기정보공학부 석사과정, 반도체공동연구소
[3]서울대학교 전기정보공학부 교수, 반도체공동연구소
yshwang@sor.snu.ac.kr, jsyou@sor.snu.ac.kr, kbkang@sor.snu.ac.kr, ypaek@snu.ac.kr

A Survey on Ciphertext Side-Channel Attack and Countermeasures

Yunseong Hwang[1], Kibong Kang[1], Junseung You[1], Yunheung Paek[1]
[1]Dept. of Electrical and Computer Engineering and Inter-University Semiconductor Research Center (ISRC), Seoul National University

요 약

신뢰 실행 환경(TEE)은 클라우드 컴퓨팅에서 작업의 기밀성과 무결성을 보장하는 기술이다. 신뢰 실행 환경 기술들은 기밀성을 위해 암호화된 메모리를 사용해 외부의 공격자가 실제 값을 알지 못하도록 사용자의 데이터를 보호한다. 암호화 과정에서 AES 암호화를 사용하여 메모리를 암호화하며, 그 중에서 XEX와 XTS 방식을 사용한다. 그러나 이런 암호화 방식은 고정된 물리 주소의 메모리 블록에 대해 결정적으로 암호화되어 공격자가 암호문에 대한 평문을 유추하는 암호문 부채널 공격에 취약할 수 있습니다. 본 논문에서는 신뢰 실행 환경에서 발생할 수 있는 암호문 부채널 공격이란 무엇인지 분석하고, 그에 대한 대응 방안과 그 한계점을 제시한다.

1. 서론

신뢰 실행 환경(TEE)는 클라우드에서 동작하는 작업들에 대한 기밀성과 무결성을 보장하는 환경을 제공하는 기술이다. 클라우드 컴퓨팅의 수요가 늘어남에 따라 프로세서 제조사들(Intel, AMD 등)은 각자의 하드웨어에 맞는 신뢰 실행 환경 기술들(AMD SEV [6], Intel TDX [7], Intel SGX [8] 등)을 제공하고 있다.

각 신뢰 실행 환경 기술들은 기밀성을 보장하기 위해 암호화된 메모리를 지원한다. 메모리를 암호화하는 과정에 AES 암호화를 사용하며, 그중에서도 결정적이며 블록을 기반으로 하는 XOR-Encrypt-XOR(XEX) 방식과 XEX-based tweaked - code book mode with ciphertext stealing(XTS) 방식을 사용한다.

이러한 암호화 방식들은 암호문에 대한 부채널 공격의 가능성[2, 3]을 제공한다. 만약 민감한 자료들이 고정된 물리 주소를 가지고 있고 공격자가 암호문을 읽을 수 있는 권한을 가지고 있다면, 암호문에 해당하는 평문을 유추해낼 수 있게 된다.

암호문 부채널 공격으로 신뢰 실행 환경의 기밀성이 무너질 수 있다는 한계점을 극복하기 위해 같은 평문에 대한 암호문이 결정적이지 않게 하거나 그러한 위협을 감지하는 연구들이 진행되었다[1, 4, 5].

본 논문에서는 기존의 신뢰 실행 환경들에서 발생할 수 있는 암호문 부채널 공격을 분석하고, 그에 대한 대응 방안 및 그 한계점을 제시한다.

2. 암호문 부채널 공격에 대한 배경이론

2.1. AES 암호화

AES-XEX 방식의 경우, 각 정렬된 128-bit 메모리 블록이 독립적으로 암호화된다. 암호화 과정은 먼저, 메모리 블록의 물리 주소인 P를 사용해 tweak 값 T(P)를 계산한다. 메모리 블록의 평문 m은 tweak 값 T(P)와 XOR된 뒤 암호화된다. 암호화된 결과에 다시 tweak 값 T(P)를 XOR하여 최종 암호문 c를 얻는다. 따라서, 평문 m에 대한 암호문 c는 식 1과 같이 계산된다.

$$c = ENC(m \oplus T(P_m)) \oplus T(P_m) \quad (1)$$

이때, tweak 값은 메모리 블록의 물리 주소를 기반으로 계산되기 때문에 만약 같은 평문이 고정된 물리 주소에 존재한다면 항상 동일한 암호문으로 암호화된다는 문제가 존재한다. AES-XTS는 AES-XEX를 기반으로 하며 임의의 길이의 메모리 블록을 암호화 및 복호화할 수 있는 ciphertext stealing 방법

을 제공하고 있다. 따라서 AES-XTS 역시 AES-XEX와 같이 고정된 물리 주소에 존재하는 평문이 항상 동일하게 암호화된다는 문제가 존재한다.

2.2. 부채널 공격

부채널 공격이란 프로그램 실행 과정 중 발생하는 물리적인 현상들을 바탕으로 특정 데이터를 유추해내는 공격 기법을 의미한다. 이러한 물리적인 현상들에 의한 정보를 얻기 위해 한 번 혹은 그 이상의 반복적인 실행을 수행시킨다. 이를 통해 암호 알고리즘 상에서 비밀 키를 복원하거나 민감한 정보를 얻어내는 것을 목표로 한다.

2.3. 암호문 부채널 공격

암호문 부채널 공격[2, 3]은 2.1에서 서술된 고정된 물리 주소의 메모리 블록이 결정적으로 암호화됨을 이용하는 공격이다. 공격자는 특정 메모리 블록에 해당하는 암호문의 변화를 관찰하고, 기존에 알고 있는 실행 코드와 연관 지어 그에 대한 평문을 유추해낸다. 이를 통해 최종적으론 민감한 정보들의 평문을 알아내는 것을 목표로 한다.

암호문 부채널 공격은 크게 Dictionary 공격과 Collision 공격으로 분류할 수 있다. 두 공격 모두 고정된 물리 주소를 갖는 메모리 블록에 대한 반복적인 메모리 쓰기 동작을 악용한다. 먼저, Dictionary 공격은 공격자가 고정된 물리 주소의 메모리 블록에 대해 평문-암호문 쌍을 모아 dictionary를 만드는 공격을 의미한다. 이러한 dictionary를 바탕으로 이후 어떤 암호문이 관찰되면 그에 해당하는 평문을 유추할 수 있게 된다.

Collision 공격은 여러 번의 메모리 쓰기에 의한 암호문의 변화를 관찰해 민감한 자료와 관련된 정보를 추출하는 공격을 의미한다. 그림 1은 collision 공격을 발생시키는 과정을 보여주고 있다.

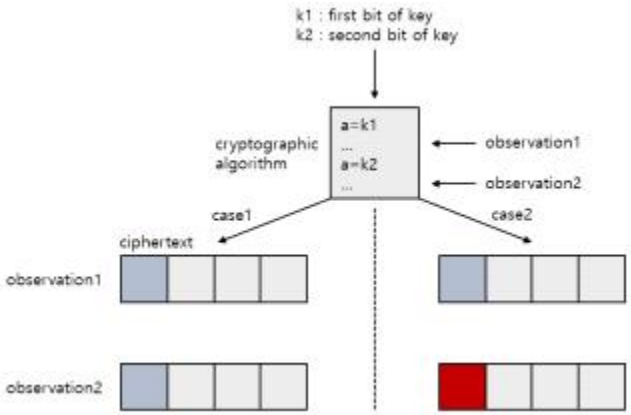

그림 1 . Collision 공격을 통해 key bit들을 유추하는 과정.

k1과 k2는 각각 private key의 첫 번째와 두 번째 bit를 나타내며, 변수 a에 연속적으로 쓰이고 있다. 두 번째 메모리 쓰기 동작이 끝났을 때 암호문이 변화했는지를 관찰하여 공격자는 k1과 k2가 같은지 아닌지를 유추할 수 있다. 공격자는 1번의 경우에는 암호문에 변화가 없었으므로 k1과 k2가 같음을, 반대로 2번의 경우에는 암호문에 변화가 있었으므로 k1과 k2가 다름을 유추할 수 있다.

이러한 공격 과정을 통해 메모리에 적힌 민감한 정보뿐만 아니라 레지스터 값 역시 추출해낼 수 있다. 구체적으로, 최근 연구[3]에서 AMD SEV에 의해 보호되는 VM에서 동작하는 운영 체제에 대해 context switch 과정에서 user space 레지스터 값이 stack에 적히는 점을 악용한 공격이 제시되기도 했다. 이는 공격자가 의도적으로 context switch를 발생시키고 그에 따라 변화하는 암호문을 관찰해 레지스터 값을 유추한다.

3. 암호문 부채널 공격의 대응 방안 및 한계점

암호문 부채널 공격을 막는 대응 방안은 공격이 발생하게 된 원인인 결정적인 암호화를 막는 것에서 시작한다.

식 1에서 알 수 있듯, 어떤 평문 m에 대한 암호문 c를 바꾸기 위해선 암호화 과정을 바꾸거나, 평문을 바꾸거나, tweak 값을 바꿔야 한다. 암호화 과정 자체를 바꾸는 것은 그에 따라 성능, 호환성, 메모리 사용 패턴 등 추가로 고려해야 할 사항들이 매우 많으므로 대응 방안으론 고려하지 않았다.

나머지 두 방법 중 평문을 바꾸는 방안에 관한 연구들[1, 4]에선 무작위의 mask 값 또는 counter 값을 사용했다. Cipherfix[1]에선 평문과 같은 크기의

mask buffer를 두어 평문과 XOR하는 방법을 사용했다. 해당 평문에 새로운 값이 쓰인다면 pseudo random number generator(PRNG)를 바탕으로 새로운 mask 값을 만들어 mask buffer에 저장한다. 반대로 평문을 읽는 경우엔 masking된 평문을 해독해 기존의 평문 값을 얻는다. 이러한 방법을 바탕으로, 특정 평문에 동일한 값이 반복해서 쓰이더라도 그에 해당하는 mask 값이 바뀌기 때문에 평문에 대한 암호문이 매번 다르게 나타나게 한다. Obelix[4]에서는 Oblivious RAM(ORAM)에서 counter 값을 사용해 데이터를 보호했다. 이후 ORAM에 write back이 발생할 때마다 counter 값을 적절히 증가시킨다. 이러한 counter 값을 바탕으로 암호문이 동일하지 않게 만들었다.

하지만, mask를 기반으로 한 평문을 바꾸는 방법에는 문제점들이 존재한다. 먼저 mask가 한정된 길이를 갖기 때문에 collision이 발생할 수 있다. 예를 들어 8-bit mask를 사용한다고 하면, mask collision은 16번의 메모리 쓰기 만에 발생하게 될 것이다. 따라서 안전하다고 생각되는 길이 이상의 mask를 사용해야 할 것이다. counter를 기반으로 한 Obelix에서는 데이터를 쪼개고 나뉜 데이터에 각각 counter를 할당하는 방식으로 collision 문제를 해결했다. 다음으로, mask를 사용하는 경우에는 평문과 동일한 크기의 mask buffer를 두어야하기 때문에 그에 따른 메모리 overhead가 발생하게 된다. 마지막으로, 모든 메모리 쓰기 동작에 PRNG를 사용한 임의의 mask 값을 만들어야 하기 때문에 추가적인 performance overhead도 발생하게 된다.

tweak 값을 바꾸는 방법은 tweak 값을 만드는 함수에 입력으로 들어가는 메모리 블록의 물리 주소를 바꾸는 것이다. 이 방법에 관해선 구체적으로 진행된 연구를 확인하지 못했다. 이를 위해선 민감한 정보들을 보관할 별도의 메모리 영역을 만들고, 그 영역 내에서 물리 주소를 이동시키는 방법을 고려해볼 수 있다. 예를 들어, 이러한 메모리 영역을 queue의 형태로 구현해, 영역 내의 메모리 쓰기가 발생하면 queue update를 수행하도록 할 수 있다. 하지만 이런 방법은 몇 가지의 고려해야 할 사항들이 존재한다. 먼저 queue를 관리하는 overhead가 존재한다. 만약 메모리 영역이 커지면, 그에 따라 영역 내에 존재하는 메모리 object도 늘어나 queue를 관리하는 데 더 많은 비용이 들게 될 것이다. 또한, 메모리 영역의 크기를 설정하는 문제도 있다.

따라서 tweak 값을 바꾸기 위해 물리 주소를 바꾸는 방법은 작은 메모리 영역에 대해서만 적용할 수 있을 것이다. 또한, 물리 주소를 바꾸며 발생하는 overhead 문제에 관해 별도의 메모리 영역을 더 작게 구획화하여 구획별로 물리 주소를 변경하거나, 물리적인 주소는 고정한 채 전달되는 논리적인 주소를 변경시키는 등의 더욱 효율적으로 동작할 수 있는 연구를 통해 해결해야 할 것이다.

4. 결론

본 논문에서는 신뢰 실행 환경 기술들의 기밀성을 위한 메모리 암호화 과정에서 사용되는 AES-XEX, AES-XTS 방식에 고정된 물리 주소에 존재하는 메모리 블록이 결정적으로 암호화된다는 취약점이 있음을 확인했다. 또한, 암호문 부채널 공격은 결정적인 암호화 방식을 악용하는 공격 기법임을 확인하고 구체적인 공격이 어떻게 이루어지는지에 대한 분석을 진행했다. 이러한 공격을 막기 위해 연구된 대응 방안 및 그의 한계점을 분석하고, 추가로 고려해볼 수 있는 해결 방안을 제시했다. 기존의 대응 방안들의 한계점과 제시된 해결 방안은 개선 가능한 여지가 있기에 이에 관한 연구가 계속 진행되어야 할 것이다.

ACKNOWLEDGEMENT

이 논문은 2024년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(RS-2023-00277326). 이 논문은 2024년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(IITP-2023-RS-2023-00256081). 이 논문은 2024년도 BK21 FOUR 정보기술 미래인재 교육연구단에 의하여 지원되었음. 본 연구는 반도체 공동연구소 지원의 결과물임을 밝힙니다. 이 논문은 2024년도 정부(산업통상자원부)의 재원으로 한국산업기술기획평가원의 지원을 받아 수행된 연구임(No. RS-2024-00406121, 자동차보안취약점기반위협분석 시스템개발(R&D)). 이 논문은 2024년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구 결과임(No.RS-2024-00438729, 익명화된 기밀실행을 이용한 전주기적 데이터 프라이버시 보호 기술 개발).

참고문헌

[1] Wichelmann, Jan, et al. "Cipherfix: Mitigating Ciphertext {Side-Channel} Attacks in Software." 32nd USENIX Security Symposium (USENIX Security 23). 2023.

[2] Li, Mengyuan, et al. "{CIPHERLEAKS}: Breaking Constant-time Cryptography on {AMD}{SEV} via the Ciphertext Side Channel." 30th USENIX Security Symposium (USENIX Security 21). 2021.

[3] Li, Mengyuan, et al. "A systematic look at ciphertext side channels on AMD SEV-SNP." 2022 IEEE Symposium on Security and Privacy (SP). IEEE, 2022.

[4] Wichelmann, Jan, et al. "Obelix: Mitigating Side-Channels through Dynamic Obfuscation." 2024 IEEE Symposium on Security and Privacy (SP). IEEE Computer Society, 2024.

[5] Deng, Sen, et al. "{CipherH}: Automated Detection of Ciphertext Side-channel Vulnerabilities in Cryptographic Implementations." 32nd USENIX Security Symposium (USENIX Security 23). 2023.

[6] Kaplan, David, Jeremy Powell, and Tom Woller. "AMD 메모리 encryption." White paper 13 (2016).

[7] Intel, “Intel Trust Domain Extensions (Intel TDX) Module Base Architecture Specification,” January 2023.

[8] Intel, “Intel Software Guard Extensions,” 2016.