신뢰실행환경 기반 프록시 암호화를 활용한 동형암호 기반 클라우드 서비스의 성능 개선 연구

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

Proxy Encryption with Trusted Execution Environments to Reduce the Communication Overhead of Cloud Services over Encrypted Data

Youyeon Joo[1], Kevin Nam[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

동형암호 기반 클라우드 서비스는 평문대비 큰 동형암호문의 크기로 인해 통신부하가 발생한다. 본 논문은 이러한 통신부하를 줄이기 위한 방법으로 신뢰실행환경(Trusted Execution Environment, TEE) 기반 프록시 암호화를 활용하는 방법을 제안하고, 실험을 통해 큰 동형암호 파라미터에서의 유용성과 성능이 뛰어남을 보여주고자 한다.

1. 서론

클라우드 서비스 시장 규모와 함께 프라이버시 유출 문제도 증가함에 따라 사용자 데이터를 보호할 수 있는 기술들에 대한 관심이 증가했다. 특히 동형암호는 데이터를 암호화 상태로 연산할 수 있는 큰 장점을 지니고 있는 기술로, Google, Microsoft 등 많은 업체에서 주목하고 있는 기술이다.

동형암호는 암호문의 특성상 매우 많은 연산량을 유발하는데, 이를 극복하기 위한 다양한 측면에서의 연구들이 이루어져 현재 1초 내로 신경망 연산이 가능한 수준까지 도달했다[1]. 한편, 동형암호는 암호문의 크기가 매우 크다는 또다른 단점도 지니고 있다. 연산량에 비해 상대적으로 주목을 덜 받는 문제이며, 많은 연구가 통신에 대해 '이상적'인 전제를 바탕으로 접근하지만, 실제 클라우드 서비스에 있어 암호문의 크기는 큰 통신부하로 이어질 수 있기에 무시할 수 없는 문제점이다[2].

본 논문은 실제 클라우드 서비스에 동형암호를 적용할 때 발생할 수 있는 통신 부하에 대해 분석하고, 이를 극복하기 위해 하드웨어 기반 격리 기술인 신뢰실행환경 (Trusted Execution Environment, **TEE**)을 활용하는 방법을 소개한다.

<표 1> HEaaN 라이브러리의 대표 파라미터 (λ=128)

파라미터	N	logPQ	용량 (MB)
Par-A	131,072	2,070	64.7
Par-B	65,536	1,555	24.3

2. 이론적 배경

2.1. 동형암호의 암호문 크기

<표 1>은 대표적인 동형암호 라이브러리 중 하나인 HEaaN 라이브러리에서 사용되는 CKKS 체계의 동형암호 파라미터이다. λ는 보안 수준으로, 표준인 128-bit를 사용하였다. CKKS 암호문은 두 개의 다항식쌍으로 이루어져 있다. 각 다항식은 N차 다항식으로써, 각 항이 logPQ bit로 이루어진다. 표에 나온 파라미터의 경우 암호문 당 64.7, 24.3 MB라는 큰 크기를 확인할 수 있다. 이는 표준 암호인 AES와 RSA가 갖는 32 bytes, 1920 bytes 대비 매우 큰 암호문 크기이며, ResNet18 신경망 연산 시 동형암호를 활용하면 사용자가 클라우드에 1.3 GB의 암호문을 전송해야하는 결과로 이어진다[2]. 평문의 경우 512 bytes만 전송해도 된다는 점을 고려하면 매우 큰 부하이다.

2.2. 클라우드 네트워크 환경

문제는 이런 큰 암호문을 전송함에 있어 클라우드 네트워크 환경이 매우 열악하다는 점에 있다. 많은 연구들이 초고속 NIC와 유선망 수준의 속도가

가능하다는 전제로 동형암호 활용의 용이성을 주장한다. 그러나 실제 수치를 살펴보면, AWS는 문서상 최대 25Gbps를 제공한다고 하지만, 이는 '이상적'인 대역폭임을 명시하고 있다. 현실적으로는 클라우드 서비스 사용자 중 휴대폰, 노트북을 활용한 무선 환경을 활용하거나, 장거리에서 접속할 경우, 훨씬 낮은 대역폭을 할당받게 된다[3]. 또한, 일반적으로 사용자 측의 업로드 속도를 다운로드 속도의 20% 정도로 낮게 할당하는데[4], 이와 같은 환경에서 사용자가 자신의 데이터를 암호화하여 전송하면 큰 통신 부하가 발생할 수밖에 없다. 구체적으로는, 5G 통신을 활용하여 AWS에 접속할 때 1Gbps의 대역폭을 할당받게 되며, 이 중 사용자 업로드 대역폭은 약 200Mbps로 제한된다.

2.3. Transciphering를 활용한 극복방법

최근 이런 메모리 부하로 인한 통신량을 감소하기 위한 해결책으로 transciphering이라는 기술이 제안되었다. 먼저 사용자는 데이터 d를 저용량 암호체계 c의 암호키 ck를 활용하여 암호화한 $Enc_c(d,ck)$를 클라우드로 전송한다. 이후, ck를 동형암호 (HE)로 암호화한 $Enc_{HE}(ck)$를 클라우드로 전송한다. 클라우드는 $Enc_c(d,ck)$ 를 $Enc_{HE}(ck)$로 복호화하는 연산을 동형암호로 수행한다. 이는 즉, $Dec(Enc_c(d,ck), Enc_{HE}(ck))$를 수행, $Enc_{HE}(d)$를 만든다 (복호화 과정 역시 하나의 연산으로써, 동형암호 상태로 수행한다). 사용자는 동형암호로 저용량 암호체계의 키 ck를 한번만 암호화해서 전송하고, 다른 데이터는 모두 저용량 암호체계로 암호화하기에, 동형암호로 데이터를 암호화하고 전송할 때마다 훨씬 적은 양의 통신량이 발생한다.

하지만 이 방법 역시 동형암호 상태로 저용량 암호체계의 복호화 연산을 수행해야 한다는 한계가 있다. 동형암호 transciphering의 대표적인 연구로는 HERA[5]가 있는데, <표1>의 Par-B를 활용할 경우 대략 130초 이상 필요할 만큼 큰 연산부하가 발생해 이를 고려한 효율적인 암호체계 선택이 중요하다.

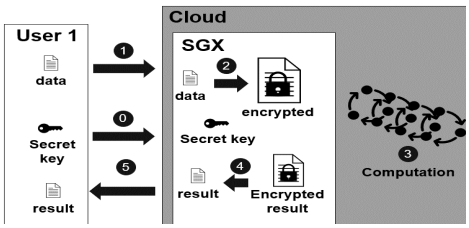
2.4. TEE와 클라우드 서비스에의 적용 한계점

SGX, SEV와 같은 TEE는 하드웨어 격리기술을 활용한 원격 계산 기술이다. 즉, 사용자가 자신의 데이터와 코드를 원격 환경에서 수행할 때 사용할 수 있는 기술로, 많은 클라우드 서비스들이 TEE를 활용한 인스턴스를 제공한다. 개별 세부 사항은 다르지만, 큰 틀에서 TEE는 외부에서 바라보았을 때 AES와 같은 암호화된 상태로 보이며, 내부를 보기 위해서는 권한인증을 받아야 한다. 이런 TEE는, 믿을 수 없는 원격 계산 환경에서 사용자의 프라이버시를 지킬 수 있다는 점에서 프라이버시 보호 기술로 활용할 수 있다. 하지만 클라우드 서비스에 활용 시, 신경망을 활용한 음성 서비스 등 여러 기능이 클라우드의 민감 데이터를 활용하는 경우가 있으며, 반대로 클라우드의 민감 데이터가 사용자의 TEE 내에서 동작할 경우 사용자에게 역으로 유출되는 위험이 발생한다는 문제가 있다.

3. TEE를 활용한 프록시 암호화 기술

3.1. TEE 기반 프록시 암호화 과정

우리는 사용자의 프라이버시만 관여할 경우 TEE 사용에 데이터 유출 문제가 없다는 점을 주목했다. 데이터의 암/복호화 연산의 경우, 클라우드의 민감 데이터나 코드는 연산에 관여하지 않으며, 사용자만 관여하는 연산이기 때문에 TEE에서 수행해도 무관하며, 사용자의 암호키들 역시 TEE에 보관하여 클라우드 다른 환경으로부터 격리되어 있으니 프라이버시 유출 문제가 발생하지 않는다.



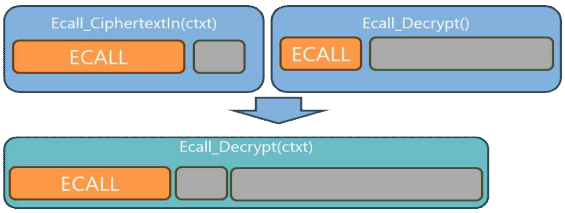
<그림 1> TEE 기반 동형암호 프록시 암호화 (Azure SGX 활용)

이런 관찰을 바탕으로, 우리는 <그림 1>과 같은 TEE 기반 동형암호 프록시 암호화 과정을 생각해볼 수 있다. 본 논문에서는 Azure사에서 제공하는 SGX를 TEE로 사용했는데, 이와 관해서는 후에 5장에서 더 서술하겠다. 먼저 사용자는 동형암호 암호키를 클라우드의 TEE에 보관한다 (⓪). 그 후, 자신의 민감 데이터를 TEE 내부로 전송하고 (①), TEE 내에서 동형암호 암호키를 활용하여 암호화하여 (②) 클라우드에게 전달해준다. 클라우드에서 연산이 끝난 후 (③) 다시 TEE로 결과를 돌려받고, 이를 복호화하여 (④) 사용자에게 전송하여 복호화한 결과를 확인할 수 있다 (⑤).

3.2. TEE 기반 프록시 암호화 과정 최적화

통신량을 줄인 대신, 이 방법은 사용자 환경이 아니라 TEE 내에서 암/복호화를 수행하기에 추가 부하가 발생할 가능성이 있다. TEE는 내부로 함수 혹은 데이터를 call하고 내보내는 데에 있어 인증

등의 연산이 필요하여 이로 인한 부하가 발생한다. 따라서 성능 개선을 위해서는 단순히 동형암호 암/복호화 코드를 포팅하는 것이 아닌, 추가 부하를 일으키는 요소들을 고려하여 최적화해야 한다.

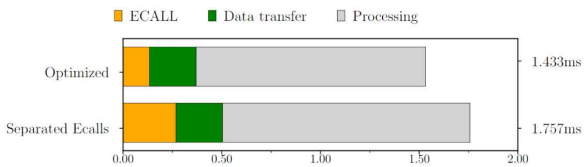


<그림 2> SGX 내부 HEaaN 포팅 시 최적화 과정

<그림 2>는 SGX의 특징을 고려하여 HEaaN 포팅을 최적화하는 과정을 나타낸다. 단순히 포팅할 경우, HEaaN에서 함수로 구현된 CiphertextIn과 Decrypt 등 모든 함수마다 Ecall이라는 SGX 함수 호출 과정이 추가되어, 성능 부하가 발생하지만, 이들을 하나로 합쳐 한 번의 Ecall으로 암호문을 들여와서 복호화하도록 최적화할 수 있다. 다른 TEE들도 이와 유사하게 단순 포팅이 아닌 맞춤형 최적화를 통해 성능 개선을 이룰 수 있다.

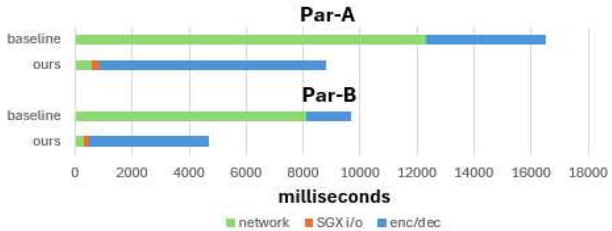
4. 실제 클라우드에서의 실험 결과

구체적인 성능 수치 측정을 위해, Azure 사의 DCsv2 인스턴스를 활용했다. 이 인스턴스는 Xeon E-2288G CPU를 탑재하고 있으며, SGX 기능을 제공한다. 이 인스턴스에 HEaaN 라이브러리를 포팅하였으며, <표 1>의 두 파라미터를 사용했다. 프록시 암호화 과정 적용 전/후를 비교하기 위해 사용자 단에서 암호화하여 클라우드에 전송하는 접근과 비교하였다. 일반적으로, 사용자 단 성능이 클라우드보다 낮지만, 우리는 비교군에 유리하도록 사용자 역시 서버급 성능을 지닌 (Xeon Gold 6336) 환경에서 실험을 수행하였다 (TEE 프록시 암호화 없이 사용자가 직접 암/복호화를 수행하므로 사용자 성능이 빠를수록 비교군 성능이 좋아진다).



<그림 3> SGX의 Ecall 최적화 전/후 실행시간 비교

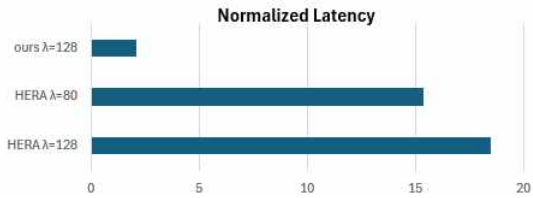
<그림 3>은 SGX 맞춤형 최적화 진행 전/후 실행시간 비교이다. SGX 내부로 HEaaN의 각 함수를 불러오는 Ecall 횟수를 최소화함으로써, 약 10.33% 소요시간 개선 효과를 얻을 수 있었다. <그림 4>는 TEE 프록시 암호화 적용 전/후 성능을



<그림 4> TEE 프록시 암호화 적용 전/후 동형암호 활용 성능 비교

비교한 수치를 나타낸다. 데이터는 각 파라미터가 패킹할 수 있는 float 벡터를 활용했으며, baseline은 사용자가 암호화 후 전송, 돌려받은 후 복호화하는 시간을 측정했고, TEE 프록시 암호화 (ours)는 데이터를 TEE로 전송, TEE 내부에서 암호화 후 복호화, 사용자에게 재전송하는 시간을 측정했다.

실험 결과, TEE 프록시 암호화 사용 시 Par-A와 Par-B 각각 2.06배, 1.88배 성능 개선을 이룰 수 있음을 확인할 수 있다. 약간의 SGX I/O 시간이 추가되었고, SGX 내부에서 암/복호화하는 데에 있어, TEE를 사용함으로써 발생하는 추가 연산으로 인해 1.88-2.63배의 시간이 추가되지만, 통신량을 줄임으로써 통신 부하가 많이 감소함에 따라 전체적으로 성능 향상을 누렸음을 확인할 수 있다.



<그림 5> HERA와의 성능 비교 (Normalized Latency)

<그림 5>는 transciphering 기술인 HERA를 사용하는 경우와의 성능 비교를 나타낸 것이다. 동일한 보안 수준 (λ=128-bit)의 경우 TEE 프록시 암호화가 18.78배 빨랐다. 심지어 더 낮은 보안 수준인 λ=80-bit인 세팅의 HERA를 사용하는 것보다도 15.04배 빠른 것을 확인할 수 있다.

5. 토론 및 고찰

실험을 통해, TEE 프록시 암호화 기법은 기존 동형암호나 transciphering을 활용하는 접근보다 성능이 뛰어남을 확인할 수 있다. 본 장에서는 이런 기능을 사용함에 필요한 논의점들을 다루고자 한다. 근본적으로, TEE를 사용하고자 할 시에 사용하는 시스템의 CPU (혹은 사용 device) 제조사에 따라 사용할 수 있는 TEE 종류가 제한된다. 동형암호와 같이 프로그램이 아닌, 하드웨어 장비에 대한 의존성이 있기에, TEE를 사용한다는 큰 전제에

의존적이라고 할 수 있다. 하지만 과거와 다르게 TEE는 보편화된 기술로, 대다수의 클라우드 서비스들이 제공하고 있는 기능이다. 본 논문이 활용한 SGX를 제공하는 Azure[6], Nitro와 SEV-SNP를 제공하는 AWS[7], 그리고 CPU 뿐 아니라 GPU 기반 TEE를 제공하는 구글까지[8], 주요 클라우드 서비스 기업은 TEE를 포함한 인스턴스를 제공하고 있다. 따라서, 클라우드 서비스에서의 TEE 활용은 다분히 현실적인 접근이라 할 수 있다.

성능에 관해서, TEE는 그 종류별로 필요한 추가 연산 (e.g. Ecall 등)과 보장하는 보안 수준이 상이하다. 예를 들어, Intel의 SGX는 코드 동작 메모리 용량이 128MB를 넘을 경우 매우 큰 부하가 발생하고, AMD의 SEV의 경우 integrity 보장 기능을 사용할 경우 큰 성능부하가 발생한다.

본 **논문에서 SGX를 사용한 이유는**, 다른 TEE에 비해 제한사항과 성능 부하가 크기 때문이다. 대표적으로 비교되는 SEV와 비교했을 때 SGX가 19.31배 느리다[9]. 앞서 사용자 단을 서버급 성능으로 실험한 이유와 동일하게, 제한사항이 많고 성능이 느린 SGX를 사용했음에도 뛰어난 성능을 보임을 증명하기 위해 선택했다. 추후 보다 정확한 분석을 위해, 미래의 연구로써 SEV에 동형암호를 포팅하는 연구를 생각해볼 수 있을 것 같다.

한편 TEE가 동형암호와 같은 수준의 보안성을 보장하는지에 대해서는 정확히 결론짓기 어렵다. 두 기술의 기반 보안 기술이 다르기에 (하드웨어와 수학) 정확히 주장할 수 없기 때문이다. 하지만 확률 기반 보안 수준을 사용하는 MPC와 계산적 보안성 (λ)과 차이가 있음에도 교차 사용되듯이[2], 각 기술에서 표준처럼 다루어지는 보안 수준이 지켜졌을 때를 안전하다고 볼 수 있고 이에 하드웨어적으로 충분한 격리기술을 사용하는 TEE 기술을 교차 사용하는 것을 수용할 수 있을 것이다.

6. 결론

본 논문은 동형암호 기반 클라우드 서비스를 활용함에 있어 동형암호의 큰 암호문 크기로 인해 발생할 수 있는 통신부하를 개선하기 위해 TEE 기반 프록시 암호화 기술을 제안한다. 실험 결과, 일반 사용 대비 약 2배 가속이 이루어지며, 통신량 감소에 초점을 둔 다른 접근과 비교 시에도 15배 넘는 가속을 이룰 수 있음을 확인할 수 있다. 미래 연구로 SGX 외 다른 TEE를 사용한 시스템 구현들이 촉진되었으면 한다.

7. ACKNOWLEDGEMENT

이 논문은 2024년도 BK21 FOUR 정보기술 미래 인재 교육연구단에 의하여 지원되었음. 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 2023년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임. (No.2021-0-00528, 하드웨어 중심 신뢰계산 기반과 분산 데이터보호박스를 위한 표준 프로토콜 개발). 이 논문은 2023년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.RS-2023-00277060, 개방형 엣지 AI 반도체 설계 및 SW 플랫폼 기술개발).

참고문헌

[1] Prasetiyo, A. Putra and J. -Y. Kim, "Morphling: A Throughput - Maximized TFHE - based Accelerator using Transform - domain Reuse," 2024 IEEE HPCA, Edinburgh, UK, 2024, pp. 249-262

[2] Garimella, Karthik, et al. "Characterizing and optimizing end-to-end systems for private inference." Proceedings of the 28th ASPLOS, Volume 3. 2023.

[3] https://docs.aws.amazon.com/ko_kr/AWSEC2/latest/User Guide/ec2-instance-network-bandwidth.html

[4] Hu, Wenjin, Tao Yang, and Jeanna N. Matthews. "The good, the bad and the ugly of consumer cloud storage." ACM SIGOPS Operating Systems Review 44.3 (2010): 110-115.

[5] Cho, Jihoon, et al. "Transciphering framework for approximate homomorphic encryption." International Conference on the Theory and Application of Cryptology and Information Security. Cham: Springer International Publishing, 2021.

[6] https://learn.microsoft.com/en-us/azure/confidential-computing/virtual - machine- solutions-sgx

[7] https://docs.aws.amazon.com/ko_kr/AWSEC2/latest/UserGuide/sev-snp.html

[8]https://cloud.google.com/confidential-computing

[9] Mofrad, Saeid, et al. "A comparison study of intel SGX and AMD memory encryption technology." Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy. 2018.