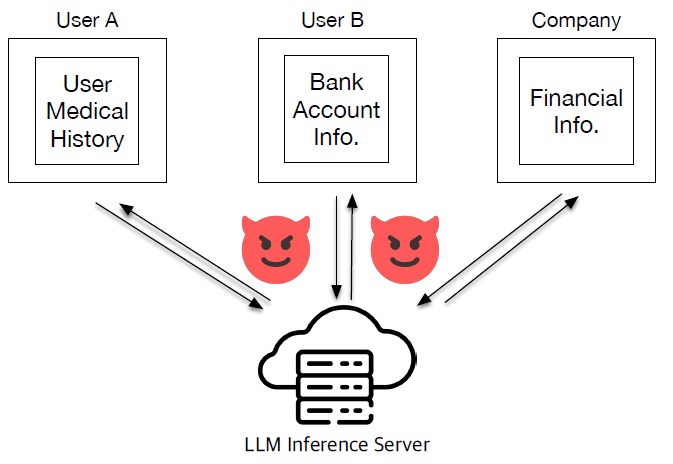
**Introduction**

2017년 “Attention Is All You Need”라는 논문 발표 이후 OpenAI, Google, 그리고 Meta와 같은 수 많은 기업들이 자신만의 거대언어모델(Large Language Model; LLM)을 개발해 출시해오고 있다. LLM의 추론 성능을 향상시키기 위해 다방면으로 수행된 연구들의 결과를 통해 확립된 사실은 추론 성능이 LLM의 모델 파라미터 개수와 비례한다는 점이다. 이에 따라 추론 성능을 향상시키기 위해 LLM의 크기는 나날이 커지고 있는 실정이다.

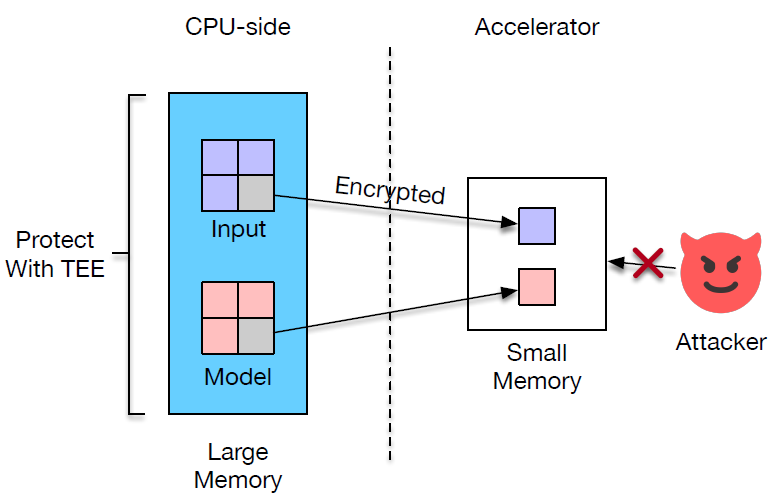
반면에 LLM의 크기가 커진다는 것은 연산장치 및 메모리와 같은 컴퓨팅 자원이 그만큼 더 요구된다는 점이다. 현재 LLM의 크기는 이미 상당히 커져 일반적으로 추론 요청을 하는 사용자의 로컬 컴퓨팅 머신에서는 효율적인 연산이 어렵기에 컴퓨팅 자원이 풍부한 클라우드 서버에 추론 요청을 보내 연산한 뒤 결과물만 돌려받는 방식(Computation offloading model)을 채택한다. 이러한 방식을 통해 LLM 추론 서비스 사용자는 고성능 로컬 컴퓨팅 머신 없이도 서버의 컴퓨팅 자원을 사용해 쾌적한 추론 서비스를 누릴 수 있다.

하지만 위와 같은 Offloading model은 사용자의 민감정보를 포함하고 있는 추론 요청을 공격자가 존재할 수 있는 클라우드 서버에 어떠한 보호장치도 없이 보내기 때문에 보안적인 측면에서는 안전하지 않다. 아래 그림은 개인 사용자나 회사의 민감정보들을 포함한 LLM 추론 요청이 클라우드 서버로 내보내질때 공격자에 의한 민감정보 탈취 가능성을 보여준다.

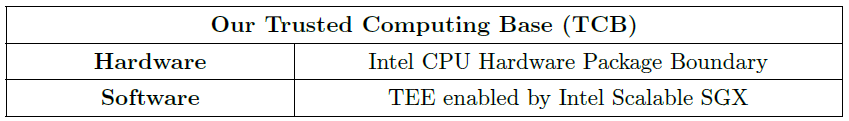
**연구목표 및 필요성:** 정리하자면 매우 거대해진 LLM 추론 연산을 위해서 추론 요청을 클라우드 서버에 내보내 연산 후 결과를 되돌려 받는 오프로딩 방식은 효율적인 측면에서 필수적이다. 반면에 사용자 민감정보가 담긴 요청을 외부로 오프로딩 시키는 것은 공격자로 인한 탈취 및 위변조와 같은 공격으로부터 안전하지 않다. 이에 따라 본 연구의 목표는 추론 요청을 외부 가속기에 오프로딩 시켜 효율적인 추론 연산이 가능하게 하면서도 가능성 있는 공격자로부터 사용자 민감정보를 안전하게 보호할 수 있는 보호 메커니즘 연구를 목표로 한다.

**Background**

**Trusted Computing Base (TCB):** 본 연구에서 타겟하는 연산 모델은 앞서 언급하였듯이 Offloading scheme을 따른다. 이러한 연산 모델에서는 평소에 추론 요청에 필요한 사용자 입력을 공격자로부터 안전한 신뢰 가능한 영역에 가지고 있다가 행렬곱셈 연산과 같은 많은 컴퓨팅 자원을 요구하는 연산 처리시에만 외부 가속기에서 추론 요청을 부분적으로 처리한다.

본 연구에서는 TCB를 TEE (Trusted Execution Environment; CPU-side에 안전구역을 형성하여 공격자로부터 Code와 Data를 보호함) 및 TEE에서 동작하는 소프트웨어로 한정한다. 또한 외부 가속기(우리의 예제에서는 GPU가 외부 가속기) 사용시에는 사용자 입력을 적적히 암호화 후 내보내 공격자가 의미를 알 수 없도록 한다. 아래 그림에서는 Offloading model을 시각화 한다. 평소에 추론을 위한 사용자 입력과 인공지능 모델 가중치는 CPU-side에서 안전하게 보호된다. 외부 가속기 사용시에 사용자 입력은 기밀성 보장을 위해 암호화 후 외부로 내보낸다. (\*만약 모델 가중치가 보호의 대상이 아닌 경우에는 TEE에 갖고 있지 않는 것이 성능에 좋음)

**Threat Model:** 바로 앞서 지금까지 설명해온 내용들을 정리하자면 우리의 위협 모델(Threat model)이 아래와 같이 정의된다.



하드웨어 측면에서는 인텔 CPU 하드웨어 패키지 내부는 외부로부터 관측이 불가능하다고 가정한다. 또한 소프트웨어 측면에서는 보호되어야할 대상은 모두 TEE내부에서 안전하게 관리되어지고 있다고 가정한다.

**기존 연구:** 기존 연구는 LLM 이전의 인공지능 모델(CNN, DNN, …)들에 대해 외부 가속기 사용시 기밀성을 보장하는 암호화 기법에 대해 제안한다. 기존의 연구 중 하나인 Slalom (ICLR 2019)에서는 연산을 위해 외부 가속기로 내보낼 때 사용자 입력에 CPRNG을 통해 생성된 난수를 더해 얻은 결과물을 생성해 외부로 유출 시 공격자가 의미를 알아볼 수 없게 감춘다. 예를 들어, 사용자 입력이 X이고 모델 가중치가 W라 할 때 우리가 최종적으로 가속기를 통해 얻고자 하는 결과물을 Z=XW이다. 하지만 기밀성 보장을 Slalom에서는 위해 사용자 입력 X에 R을 더해 을 내보낸다. 가속기에서는 Z대신 를 돌려보낸다. 원래 구하려면 값 Z를 구하려면 를 해주어야 한다. 다행히도 사용자 입력 암호화를 위한 R은 사전에 미리 선정할 수 있고 모델 가중치인 W도 학습 이후에 고정된 값이기 때문에 복호화를 위한 는 미리 계산해 가지고 있다가 가 돌아오면 단순히 빼주면 된다. Slalom에서 제안한 암호화 기법은 DNN 및 CNN과 같이 사용자 입력과 모델 가중치를 행렬곱하는 Linear layer연산에는 매우 효과적이나 추후 자세히 설명하겠지만 LLM 추론시에는 사용할 수 없게 된다. 아래 그림은 Slalom에서 Linear layer연산을 하는 시각화 한다. 1) 암호화키 R은 사전에 선정하여 모델 가중치 W와 미리 행렬곱셈해두어 복호화키 U를 저장해둔다. 이후 2)에서 사용자 입력 X가 발생하면 미리 선정해둔 R을 더해 로 암호화한다. 3)에서는 외부 가속기에서 를 구한다. 4)에서는 암호화된 상태에서 연산된 결과물을 다시 CPU-side TEE로 되돌려 받는다. 마지막으로 5)에서는 원래 구하고자 하던 값 Z를 구하기 위해 에서 미리 구해 저장해 두었던 U를 뺀다.

