Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.4, Aug. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.4.545

신뢰 실행 환경 어플리케이션 개발을 위한 상용 컨피덴셜 컴퓨팅 프레임워크 동향 및 비교 분석*

김 성 민†‡
성신여자대학교 (교수)

Analyzing Trends of Commoditized Confidential Computing Frameworks for Implementing Trusted Execution Environment Applications*

Seongmin Kim†‡
Sungshin Women's University (Professor)

요 약

신뢰 실행 환경(TEE, Trusted Execution Environment) 기술의 발전과 함께 신뢰성을 보장하지 못하는 클라우드 환경에서도 어플리케이션의 코드 및 데이터를 보호할 수 있는 컨피덴셜 컴퓨팅(confidential computing)이 차세대 클라우드 핵심 기술로 떠올랐다. 학계뿐만 아니라 산업계에서도 Intel SGX 기술을 중심으로 컨피덴셜 컴퓨팅 솔루션 상용화가 활발히 이루어졌다. 하지만 TEE 기술 기반 어플리케이션을 구현하고자 할 때, 다양한 선택지 중 어떠한 컨피덴셜 컴퓨팅 프레임워크를 활용하는 것이 효과적인지에 대한 명확한 기준이 존재하지 않는다. 본 논문에서는 현존하는 상용 컨피덴셜 컴퓨팅 프레임워크 기술들의 특성에 대한 심층적인 비교 분석을 수행하고, 각 프레임워크의 장단점을 파악할 수 있는 기준 지표들을 도출한다. 이를 바탕으로, 설계 및 운용 목적에 따라 어떠한 프레임워크를 선택하여 활용하는 것이 효과적인지에 대한 선택 기준을 제안한다.

ABSTRACT

Recently, Confidential computing plays an important role in next-generation cloud technology along with the development of trusted execution environments(TEEs), as it guarantees the trustworthiness of applications despite of untrusted nature of the cloud. Both academia and industry have actively proposed commercialized confidential computing solutions based on Intel SGX technology. However, the lack of clear criteria makes developers difficult to select a proper confidential computing framework among the possible options when implementing TEE-based cloud applications. In this paper, we derive baseline metrics that help to clarify the pros and cons of each framework through in-depth comparative analysis against existing confidential computing frameworks. Based on the comparison, we propose criteria to application developers for effectively selecting an appropriate confidential computing framework according to the design purpose of TEE-based applications.

Keywords: Trusted Execution Environment, Confidential Computing, Enclave, Intel SGX, Rust

Received(03. 30. 2021), Modified(06. 09. 2021), Accepted(06. 23. 2021)

* 이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원(NRF-2021R1G1A100632611)과 2021년도 정부(산업통상자원부)의 재원으로 한국산업기술진흥원의 지원(P0008703, 2021년 산업혁신인재성장 지원사업)을 받아 수행된 연구임.

† 주저자, sm.kim@sungshin.ac.kr
‡ 교신저자, sm.kim@sungshin.ac.kr(Corresponding author)

I. 서 론

최근, 다양한 분야의 IT 기업들이 서비스 효율성 향상 및 비용 절약을 위해 클라우드 플랫폼을 활용하고 있다[1, 2, 3]. 하지만 클라우드 환경을 활용하는 서비스 공급자는 서비스 운용을 위한 컴퓨팅 환경에 대한 접근 제어 또는 통제에 있어 명확한 제약을 가질 수밖에 없다. 예를 들어, 서비스 공급자는 클라우드 플랫폼 공급자가 악의적으로 주요 데이터 및 개인 정보들을 유출하는지에 대한 여부를 확인할 수 없으며, 클라우드 플랫폼 내에 취약점이 존재할 경우 이에 대한 직접적인 대응을 하는 것이 사실상 불가능하다. 클라우드 플랫폼 공급자 입장에서도 더 많은 서비스 공급자를 유인하기 위해서는 서비스 공급자의 소프트웨어가 무결성이 깨지지 않은 상태에서 데이터의 유출이 발생하지 않도록 안전하게 동작하는 것을 보장하는 것이 필요하다. 하지만 이는 근본적으로 신뢰성을 보장하지 못하는 클라우드 환경 본연의 특성으로 인해 발생하는 복잡한 문제이며[4, 5], 학계 및 산업계 전반에 걸쳐 중요한 화두로 떠오르게 되었다.

인텔(Intel) 사에서 개발한 상용 신뢰 실행 환경(TEE, Trusted Execution Environment) 기술인 SGX(Software Guard eXtension)[6]의 등장으로 클라우드 플랫폼의 생태계에는 많은 변화가 생기기 시작하였다. 기존 신뢰 플랫폼 모듈(TPM, Trusted Platform Module) 기반의 TEE 기술[7, 8]의 경우, 상용 소프트웨어에서 활용하기에 기능적 제약 및 성능적인 한계점을 가진 반면, SGX 기술은 범용 x86 아키텍처에서 활용할 수 있을뿐더러 네이티브(native)에 가까운 성능을 제공한다. SGX 기술은 *enclave*라고 하는 안전한 컨테이너 내에서만 평문 데이터를 저장하고 코드를 실행함으로써, 운영 체제나 하이퍼바이저 등과 같은 특권 계층의 권한을 가진 소프트웨어 및 하드웨어(CPU를 제외한)를 컨트롤하는 공격자들로부터 어플리케이션을 보호할 수 있다. 이를 통해, 근본적으로 신뢰성을 보장하지 못하는 클라우드 환경에서도 서비스 공급자들은 자신들의 소프트웨어를 안전한 격리된 환경에서 실행하는 것이 가능해졌다. 위와 같은 기술을 통상 컨피덴셜 컴퓨팅(confidential computing)이라 칭하며, 이는 차세대 클라우드에서 필수 불가결하게 갖추어야 할 핵심 기술로 자리매김하게 되었다.

실제로, 산업계에서는 SGX를 중심으로 컨피덴셜 컴퓨팅 솔루션 상용화가 활발히 이루어졌다. 마이크로소프트 Azure 클라우드 및 알리바바 클라우드에서는 SGX 기능이 탑재된 클라우드 인스턴스를 퍼블릭 클라우드에서 제공하기 시작하였다[9, 10]. 인텔, 구글, 마이크로소프트, 페이스북, 화웨이, 엔비디아 등 대표적 ICT 글로벌 기업들과 MIT, Berkeley 등 연구진이 참여하여 2019년에 구성된 CCC(Confidential Computing Consortium)[11] 또한 대표적 사례라고 볼 수 있다.

SGX 자체 기술과 관련 연구들 또한 활발히 수행되었는데, 특히 SGX 기술을 활용할 수 있도록 응용 소프트웨어를 개발하기 위한 프레임워크에 대한 연구[4, 12, 13, 14, 15]가 높은 주목도를 받았다. 이러한 연구들은 기존 라이브러리나 어플리케이션 코드를 수정 없이 SGX 환경에서 사용할 수 있도록 호환 가능케 해주거나[4, 12], SGX가 가지는 성능 오버헤드를 최소화하기 위한 최적화를 수행[13, 14, 15]하는 등을 통해 SGX 환경에서의 어플리케이션의 구현, 실행 및 배치 비용을 덜어주었다. 이러한 프레임워크 활용을 통해 서비스 공급자들은 실용적인 성능을 제공하면서도, SGX 기술을 바탕으로 본인들이 제공하는 서비스에 대한 보안성을 갖춘 어플리케이션을 개발하는 것이 가능해졌다.

하지만 SGX와 같은 TEE 기술을 활용하기 위한 어플리케이션을 개발하고자 할 때, 다양한 선택지 중 어떠한 컨피덴셜 컴퓨팅 프레임워크를 활용하는 것이 효과적인지에 대한 명확한 기준이 존재하지 않는다. 어플리케이션의 목적에 따라 어떠한 특성들이 기준으로 고려되어야 하는지, 현존하는 대표적 상용 TEE 어플리케이션 개발 프레임워크들은 해당 기준들에 대해 어떠한 차이를 갖는지, 각 기술 간의 장단점은 무엇인지에 대한 상호 비교 등의 객관적이고 통합적인 분석은 부족한 실정이다.

본 논문에서는 SGX 기술을 중심으로, TEE 어플리케이션을 개발하기 위해 학계 및 산업계에서 제안된 상용 컨피덴셜 컴퓨팅 프레임워크 기술들의 특성에 대한 심층적인 기술 분석을 수행한다. 이후 분석한 내용을 통해 상용 프레임워크의 장단점을 파악할 수 있는 기준 지표들을 정의한다. 이를 바탕으로, 설계 및 운용 목적에 따라 어떠한 프레임워크를 선택하여 활용하는 것이 효과적인지에 대한 선택 기준과 해당 기준에 따른 우선순위를 제시한다. 본 논문에서의 연구 결과는 향후 상용 TEE 어플리케이션 생태계를 확장하는 데 도움이 될 것으로 기대한다.

II. 배경 지식: TEE 기술

TEE 기술은 TPM 기반에서 점차 CPU의 명령어 집합체(ISA, Instruction Set Architecture)를 확장, 새로운 명령어를 추가함으로써 안전하면서도 격리된 실행이 가능한 TCB(Trusted Computing Base)를 구축하는 형태로 발전하였다. 현재 인텔, AMD, ARM과 같은 메인 CPU 공급업체는 각각의 건축에 맞는 TEE 기술을 설계 및 상용화하였다[6, 16, 17]. 나아가 FPGA 기반으로 TEE 기능을 제공하기 위한 칩을 제작하는 연구 또한 수행되고 있다[18]. 본 장에서는 각각의 TEE 기술에 대한 배경 지식과 함께, 본 논문에서 SGX 기술을 중심으로 TEE 개발 프레임워크를 분석하는 이유에 대하여 설명한다.

2.1 Intel SGX

SGX는 격리 실행(isolated execution) 및 암호화를 통해 어플리케이션의 코드 및 데이터를 보호한다. Intel 스카이레이크(Skylake) CPU 모델부터 해당 기능이 탑재되었으며, 물리 메모리 일부 중에서 EPC(Enclave Page Cache)라고 하는 영역을 최대 128MB까지 예약하여 사용할 수 있다. 해당 영역은 암호화가 되어 있으며 가상 메모리 내 안전한 컨테이너인 enclave에 매핑이 된다. 이는 CPU 패키지 내에 있는 MEE(Memory Encryption Engine)에 의해서만 복호화 및 실행이 이루어진다. 따라서, 운영 체제 및 하이퍼바이저 등과 같은 특권 계층의 소프트웨어들도 enclave 내 코드 및 데이터 평문에 접근할 수 없다. Fig.1은 SGX가 제공하는 격리 실행 흐름을 나타낸다.

또 하나의 주요 기능은 원격 검증(remote attestation)인데, 이를 통해 enclave 영역에 대한 무결성과 enclave 개발자가 사인한 인증서에 대한 검증을 원격 호스트에서 수행할 수 있다. 이 과정에서 실제로 enclave 프로그램이 실제 SGX 하드웨어가 탑재된 CPU에서 SGX 모드로 실행되고 있는지도 확인 함으로써, SGX CPU를 에뮬레이션하여 가장한 공격 또한 탐지할 수 있다. 참고로 원격 검증을 과정에서 디피-헬만 키 교환(Diffie-Hellman key exchange)을 함께 사용하여 신뢰할 수 있는 채널을 수립할 수 있다.

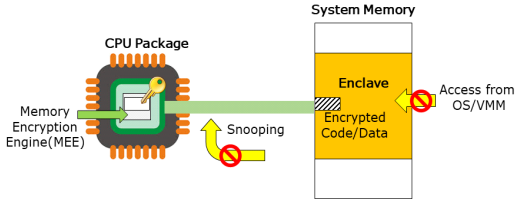

Fig. 1. Work flow of isolated execution of SGX

2.2 기타 TEE 아키텍쳐

산업계에서 활발히 사용되는 또 다른 상용 TEE 기술로는 ARM TrustZone[16]이 있다. 대표적 TrustZone 기반 솔루션으로는 Samsung Knox가 있으며, SGX와 달리 신뢰할 수 있는 secure world(TEE)와 신뢰성을 보장하지는 못하나 일반 운영 체제 및 하이퍼바이저와 같이 기능적 제약이 없는 normal world(REE, Rich Execution Environment)로 구분되어 있다. 각각의 world는 하이퍼바이저부터 어플리케이션 계층까지 가질 수 있으며, 부팅 단계에서 TEE에서 동작하는 소프트웨어의 무결성 검사를 통해 신뢰성을 보장한다.

SGX와 같이 x86 아키텍처를 대상으로 등장한 AMD SEV(Secure Encrypted Virtualization)[17]는 VM의 메모리 영역을 개별 암호키로 암호화함으로써, 하이퍼바이저로부터 guest VM의 데이터를 보호하는 기술이다. 산업계에서 TEE 기술의 상용화에 노력을 기울일 때, 학계에서는 버클리 연구진을 중심으로 개발한 RISC 기반의 개방형 ISA인 RISC-V[18]를 TEE로써 활용하기 위한 연구가 수행되었다. 대표적인 RISC-V 기반 TEE 연구로는 MIT 연구진이 제안한 Sanctum[19]이 있다.

SGX 기술 대비 타 상용 TEE 아키텍처들은 아래의 제약 사항 및 한계점을 가진다. 첫째로, TrustZone은 ARM 건축 기반이기 때문에 임베디드 시스템을 표적으로만 활용이 제한된다. 즉, x86 기반 서버로 구성된 클라우드 환경에 적용하기에 근본적으로 한계가 있다. AMD SEV 기술은 보안 측면에서 명확한 단점을 가진다. 원격 검증을 통해 데이터에 대한 무결성 및 기밀성을 보장하는 SGX와 달리, 데이터에 대한 기밀성은 보장하나 무결성을 보장하지 않기 때문에 재전송 공격(replay attack) 등에 취약할 수 있다. RISC-V의 경우 명령어 집합체의 개발이 아직 초동단계이며, 어플리케이션 생태계 또한 성숙하지 않았다. 따라서, 본 연구에서는

SGX 기술을 중심으로 TEE 개발 프레임워크에 대한 분석을 수행한다.

2.3 관련 연구

컨피덴셜 컴퓨팅 기술의 발전에 따라 TEE 기반 어플리케이션 개발을 위한 프레임워크가 등장하고 있지만, 프레임워크 별 특성, 우수성 및 한계점을 심층 분석하고 비교한 연구는 부족한 실정이다. Intel SGX 기술과 AMD SEV 기술을 직접 비교하거나[31], 현존하는 다양한 TEE 기술들에 대해 격리(isolation)와 검증(attestation) 측면에서의 디자인 특성을 비교한 연구[32]들은 존재한다. 하지만 이러한 연구들은 하드웨어 기반 TEE 기술에 대한 비교 분석만을 제공할 뿐, 실제 어플리케이션을 구현하는 개발자에게 어떠한 TEE 개발용 프레임워크를 선택하여 활용하는 것이 효과적인지에 대한 선택 기준을 제공하지 못한다.

최근 SGX 어플리케이션 개발을 위한 프레임워크에서 사용되는 API 및 ABI 설계에 따라 메모리 취약점이 존재할 수 있다는 연구 결과를 통해, 현존하는 SGX 기반 TEE 개발 프레임워크들의 보안성에 대해서는 상대적인 평가가 가능해졌다[30]. 하지만 보안성이라는 기준 외에 배치 비용, 개발 비용, 성능 등과 같은 다양한 요소에 대한 복합적인 비교 및 평가를 수행한 연구는 전무하다. 본 연구에서는 현존하는 TEE 개발 프레임워크들을 심층 분석하여 어플리케이션 개발에 고려되어야 할 다양한 요소들을 제안하고, 제안한 평가 기준을 바탕으로 한 다차원적인 비교 분석을 수행한다.

III. 학계 및 산업계에서의 기술 동향

SGX 기능에 대한 스펙이 공개된 이후로, 현재까지 SGX와 관련된 다양한 연구들이 수행되고 있으며 산업계에서도 SGX 기술이 적용된 상용 솔루션을 공개하고 있다. SGX 관련 연구는 크게 세 가지 줄기로 구분되어 수행되고 있는데, 첫 번째는 본 논문에서 탐구하는 내용과 같이 SGX 기술을 기존 소프트웨어 코드에 쉽고 안전하게 적용 가능케 하고, 이에 대한 성능을 향상하기 위한 연구가 있다. 다음으로 SGX를 대상으로 한 부채널 공격(side-channel attack) 등의 취약점과 이에 대한 방어 기법 등 자체 보안에 관한 연구가 있으며, 마지막으로 SGX 기술을 다양한 컴퓨팅 분야에 적용하는 연구들이 있다. 국내 SGX 관련 연구의 경우 초동 단계이며 SGX 기술을 적용할 수 있는 시나리오에 대한 아이디어 차원의 제안[20, 21]에 머물러 있다.

3.1 TEE 개발 프레임워크 학술 연구 동향

인텔에서 공식적으로 SGX 프로그램 개발용 SDK(Software Development Kit)를 공개하기 이전부터, 학계에서는 SGX 프로그램 개발을 위한 프레임워크에 관한 연구가 수행되었다[4, 22]. 하지만 초기 TEE 개발 프레임워크의 연구는 실제 하드웨어 모드로 동작하는 것이 아닌 시뮬레이션 또는 에뮬레이션 환경에서 개략적인 성능 프로파일링만을 제공할 뿐이었다.

Intel SGX SDK[23] 공개 이후, 학계에서는 크게 2가지 목표를 달성하기 위한 개발 프레임워크에 관한 연구가 수행되었다. 첫 번째는 기존 코드를 별도의 수정 없이 SGX 환경에서 사용할 수 있도록 해주는 것을 목표로 수행된 연구들로[4, 12, 13], 개발자의 편의성 및 코드 호환성 측면에서 기여하였다. 한편, SGX 성능을 최적화하고 TCB를 줄여 공격 표면(attack surface)을 최소화하는 목표를 달성하기 위한 연구[13, 14, 15]들 또한 활발히 수행되었다. 위 목표들을 달성하기 위해 수행된 대표적 연구로는 SCONE과 Graphene-SGX가 있으며, SGX 관련 연구들에서 활발하게 활용되고 있다.

3.1.1 SCONE

2016년 USENIX OSDI 학회에서 발표된 SCONE[13]은 SGX enclave 기술을 Docker 컨테이너에서 활용할 수 있도록 제안한 최초의 기술이다. Docker가 제공하는 높은 이식성 및 개발/배포 단계의 효율성을 가지면서도, SGX가 제공하는 TEE 내에서 어플리케이션을 구동할 수 있게 해준다. SCONE은 Docker 컨테이너 내에서 소프트웨어를 수정하지 않고 구동할 수 있는 것을 가능케 한다. 또한, 보안성을 고려하여 공격 표면을 줄이고 TCB를 최소화하기 위해, 표준 C 라이브러리인 libc 중 일부만을 사용할 수 있도록 최소한의 인터페이스만을 노출(expose)한다.

SCONE은 성능적인 측면에서도 최적화 기법들을 많이 제안하였는데, 스레드 동기화 과정에서 발생하

는 성능 오버헤드를 줄이기 위해 SGX enclave 내에서 사용자 레벨 스레딩(user-level threading) 및 비동기적 시스템 호출 처리가 가능하도록 프레임워크를 설계 및 구현하였다. 이를 통해, 소프트웨어 코드에 대한 수정 없이 기본 성능 대비 0.6x에서 1.2x의 성능을 달성할 수 있었다. 현재 상용화 서비스로 제공되고 있으며, 오픈 소스는 아니지만 무료 버전의 라이센스를 통해 활용할 수 있다.

3.1.2 Graphene-SGX

Graphene-SGX는 선행 개발되었던 라이브러리 운영 체제(libOS)인 Graphene을 SGX에 포팅하고, 내부적인 구현을 최적화하여 성능을 개선한 연구로, 2017년 USENIX ATC 학회에 발표되었다. 라이브러리 운영 체제를 SGX에 포팅했기 때문에 기본적으로 glibc와 같은 표준 라이브러리 인터페이스를 포함, 운영 체제 기능에 대한 대부분의 호환성을 제공할 뿐만 아니라, fork 등과 같은 시스템 호출을 지원하기 때문에 멀티프로세스 어플리케이션을 구동할 수 있다. 이는 SCONE과 차별성을 가진 특성으로, Graphene-SGX만의 강점이라고 볼 수 있다.

Graphene-SGX는 동적 라이브러리에 대한 로딩 또한 지원한다. enclave 컴파일 과정에서 정적 라이브러리에 대한 링크만을 제공하는 것이 아닌, 동적 라이브러리를 실행 중에 enclave 내에 로드하여 사용할 수 있다. 이때, 로딩 과정에서의 보안 이슈를 고려하여 명세서(manifest) 기반으로 동적 라이브러리들에 대한 무결성 검증을 수행한다. Graphene-SGX는 어플리케이션을 수정 없이 구동하는 데 필요한 제약 사항이 SCONE에 비해 완화되었다고 볼 수 있지만, 그만큼 TCB 크기가 늘어나게 되어 공격 표면이 증가한다는 단점을 가진다. Enclave 프로세스 구동(launch) 시 라이브러리 운영 체제 전체를 로드해야 하기 때문에 상대적으로 enclave 생성 시 매우 큰 오버헤드를 가진다는 한계점 또한 존재한다. Graphene-SGX의 경우, 전체 코드가 오픈 소스로 공개되어 있다.

3.2 상용 TEE 개발 프레임워크

Intel SGX가 실제 하드웨어에서 사용 가능해짐과 동시에, 인텔에서는 SGX 어플리케이션 개발이 가능하도록 SGX 드라이버 및 SGX 개발을 위한 SDK를 공개하였다. 인텔이 공개한 SDK를 기반으로, 글로벌 소프트웨어 기업들은 자체적으로 TEE 어플리케이션을 개발하기 위한 프레임워크들을 앞다투어 출시하고 있다. 대표적인 상용 솔루션으로는 구글사의 Asylo, 마이크로소프트사의 OpenEnclave가 있다.

3.2.1 Intel SGX SDK

SGX 어플리케이션 개발을 가능케 해준 최초의 프레임워크는 인텔에서 공개한 SGX SDK이다. 윈도우 환경과 리눅스 환경에서 활용할 수 있는 버전을 순차적으로 공개하였으며, 윈도우 버전의 경우 리눅스 버전과 다르게 페이징을 지원하지 않아 enclave 영역으로 최대 128MB까지만 사용할 수 있다. SDK의 컴포넌트는 크게 SGX를 빌드 대상으로 안전하게 포팅된 신뢰할 수 있는 라이브러리(암호 라이브러리, 표준 라이브러리 등), 개발을 위한 툴체인, 샘플 프로젝트로 구성되어 있다. 개발을 위한 툴체인으로는 enclave에 서명할 때 사용하는 서명 툴과 디버깅 툴 등이 있다.

SGX SDK를 활용하여 SGX 프로그램을 개발하는 절차는 다음과 같다. 먼저 enclave의 힙 영역, 스택 영역의 크기 등의 명세를 xml 파일 형식으로 작성한다. 다음으로 enclave를 서명하기 위한 enclave 개발자의 키를 생성한 뒤, SGX 프로그램을 개발한다. 이때, SGX 프로그램의 경우 일반 C/C++ 프로그램과 같은 역할을 하는 app(untrusted)과 enclave로 구분되어 있다. 이후 app 및 enclave에 해당하는 소스 코드를 작성 후, 컴파일한 뒤 실행한다. Fig.2는 실제 SDK에서 제공하는 app 부분의 샘플 소스 코드이다. main 함수가 해당 부분에 포함되어 있으며, initialize_enclave()와 같은 SGX 런타임 라이브러리 함수들을 활용하여 enclave를 생성하는 것이 가능하다. 이후 enclave 영역으로 진입하여 보호되어야 할 코드 및 데이터를 처리한다.

Intel SGX SDK에서는 enclave 영역으로 진입하고 빠져나가기 위한 인터페이스로 ECALL과 OCALL을 제공한다. ECALL은 enclave 영역으로 들어가기 위해서, OCALL은 enclave 영역에서 빠져나와 시스템 권한이 필요한 코드(시스템 호출 등)를 처리하기 위해 각각 활용된다. OCALL 및 ECALL을 사용할 때는 보안성을 위해 파라미터의

```
/* Global EID shared by multiple threads */
sgx_enclave_id_t global_eid = 0;

/* Application entry */
int SGX_CDECL main(int argc, char *argv[])
{
    (void)(argc);
    (void)(argv);

    /* Initialize the enclave */
    if(initialize_enclave() < 0){
        printf("Enter a character before exit ...₩n");
        getchar();
        return -1;
    }

    Your code here!

done:
    /* Destroy the enclave */
    sgx_destroy_enclave(global_eid);

    printf("Info: SampleEnclave successfully returned.₩n");

    printf("Enter a character before exit ...₩n");
    getchar();
    return 0;
}
```

Fig. 2. Sample code of main() function provided by Intel SGX SDK

버퍼 크기를 메모리 포인터와 함께 EDL(Enclave Definition Language) 파일에 명확하게 명시해야 한다.

인텔에서 자사 CPU 아키텍처 기능인 SGX와 함께 SDK를 개발하였기 때문에, SGX 버전 2 명령어에 대한 패치와 같은 SGX 기술 업데이트에 대한 반영이 그 어떤 프레임워크보다 빠르게 이루어진다는 장점이 있다. 하지만 SGX SDK의 경우, SGX 프로그래밍 경험이 없는 C 또는 C++ 개발자가 어플리케이션 또는 라이브러리를 구현하는데 명확한 진입 장벽이 존재한다. ECALL, OCALL, EDL 등과 같은 SGX SDK를 활용한 프로그래밍을 위한 사전 지식이 필요하며, enclave 영역과 app 영역을 구분하여 구현해야 하는 디자인 또한 직관적이지 않아 단점으로 작용한다.

3.2.2 Google Asylo

구글에서 개발한 오픈 소스 프레임워크인 Asylo[24]는 C 또는 C++ 개발자가 익숙한 POSIX 모델을 기반으로 enclave 어플리케이션을 손쉽게 구현할 수 있도록 해준다. 다양한 CPU 아키텍처를 백엔드 타겟으로 활용할 수 있도록 설계되었다고는 하나, 오픈 소스 저장소의 코드를 확인해 본 결과 현재는 SGX 기술에 대해서만 시뮬레이션 모드 및 하드웨어 모드로의 동작을 백엔드로 선택할 수 있었다. 데모 enclave를 도커 이미지 형태로 배포하는 등 컨테이너 환경에서도 원활히 사용될 정도로 Asylo는 여타 프레임워크와 비교했을 때 성숙 단계에 진입하였다.

Asylo는 Intel SGX SDK가 제공하는 EDL 형식 및 ECALL/OCALL 인터페이스를 사용하지 않고, 자체적인 클래스와 API를 정의하여 enclave 프로그램을 구동할 수 있도록 한다. 다만, 신뢰 영역과 비신뢰(untrusted) 영역을 구분하여 프로그램을 구현하는 형태의 Intel SGX SDK의 디자인은 그대로 차용하였다. 또한, 자사에서 개발한 메시지 전달 기술인 프로토콜 버퍼(Protocol Buffers, protobuf) 및 gRPC(google Remote Procedure Call) 기술을 활용할 수 있도록 추상화(asbtraction) 계층을 구축하여 enclave 간 통신에 적용하였다. 이를 통해, protobuf가 제공하는 고성능 직렬화 및 역직렬화의 이점을 enclave 프로그래밍 시 얻을 수 있다.

3.2.3 Microsoft OpenEnclave

또 하나의 대표적 오픈 소스 프레임워크로는 마이크로소프트 사의 OpenEnclave[25]가 있다. 아마존 다음으로 높은 클라우드 점유율을 자랑하는 마이크로소프트 사는 SGX 기술 출시 이후 SGX 기반 ACC(Azure Confidential Computing) 솔루션을 출시하였다. 나아가 ACC 기술을 기반으로 컨피덴셜 컴퓨팅 생태계에서의 시장 점유율을 높이기 위해 OpenEnclave SDK를 공개하였는데, 이는 C 또는 C++ 프로그래밍 언어로 TEE 기반 어플리케이션을 구현할 수 있게 해주는 enclave 추상화 제공을 목표로 하였다. 신뢰 영역과 비신뢰(untrusted) 영역을 구분한 디자인과 자체 API들을 정의했다는 점은 Asylo와 유사하다.

다양한 TEE 환경을 백엔드로 지원할 수 있도록 통합 enclave 개발 환경을 제공한다는 목표는 Asylo와 동일하나, Asylo를 포함한 다른 컨피덴셜 컴퓨팅 프레임워크들과 비교했을 때 OpenEnclave는 상호 운용성(interoperability) 측면에서 강점을 가진다고 평가할 수 있다. Intel SGX SDK와 마찬가지로 윈도우와 리눅스 환경을 모두 지원하며, 통합 TEE 개발 프레임워크 관점에서 중요한 요소인 이기종(heterogeneous) 아키텍처에 대한 백엔드 지원 측면에서 가장 앞서있다고 볼 수 있다. 현재 OpenEnclave는 SGX와 ARM TrustZone에 대

한 어플리케이션 개발을 지원한다.

IV. Rust 기반 TEE 개발 프레임워크

보안 분야에서 높은 주목을 받는 프로그래밍 언어인 Rust[26]가 등장한 이후, Rust를 활용하여 TEE 어플리케이션을 개발할 수 있도록 해주는 프레임워크들이 제안되었다. Rust는 기존 프로그래밍 언어들과 다르게 메모리 안전성(memory safety)을 보장한다. 포인터 개념을 사용하지 않고 객체에 대한 소유권(ownership) 및 일생(lifetime) 개념을 도입함으로써 경계를 벗어난(out-of-bound) 메모리 접근을 컴파일 타임에 근본적으로 차단한다. Rust는 객체의 사용이 끝나면 해당 메모리 영역을 바로 해제하기 때문에 자바 등의 프로그래밍 언어와 다르게 런타임에 쓰레기 수집(garbage collection)이 필요하지 않아, 성능 측면에서도 우수성을 가진다.

이러한 Rust의 장점은 SGX 기술에서도 효과적으로 작용한다. 만약 enclave 내의 소스 코드가 메모리 취약점이 존재할 경우 SGX 기술은 무력해진다. Rust가 보장하는 메모리 안전성을 SGX와 같은 TEE 기술과 결합시킬 경우 이를 차단할 수 있을 뿐만 아니라, 성능 측면에서도 큰 오버헤드를 가지지 않는다. 대표적인 Rust 기반 TEE 프레임워크로는 Fortainx 사에서 개발한 EDP(Enclave Development Platform)[27], 바이두 사에서 개발한 Rust SGX SDK[28], 마지막으로 최근 CCC에서 핵심 디자인 컨셉이 공개되고 활발히 개발 진행 중인 레드햇 사의 Enarx[29]가 있다.

4.1 Fortanix EDP

Fortanix 사에서 개발한 EDP는 Rust로 구현된 프로그램이 SGX 환경에서 실행될 수 있도록 해준다. C 또는 C++ 프로그래밍 언어 중심이었던 기존 컨피덴셜 컴퓨팅 프레임워크들과 다르게, Rust 표준 라이브러리를 SGX 위에서 사용할 수 있도록 지원함으로써 사실상 프로그래머가 SGX의 존재 여부에 관계없이 일반 Rust 프로그램을 구현하는 것과 차이가 없도록 해준다. Fig.3에서 볼 수 있듯이, 간단한 Hello world 프로그램이기는 하나 실행 과정에서 빌드 대상을 "x86_64-fortanix-unkown-sgx"로 주는 것 외에는 SGX 환경에서 실행된다고 판단할 수 있는 부분이 없음을 알 수 있다.

```
in hello-world/src/main.rs

fn main() {
    println!("Hello world!");
}

# Run your enclave!
cd hello-world
cargo run --target x86_64-fortanix-unknown-sgx
```

Fig. 3. Hello world code example of EDP and commands to run the program

또 다른 차이점은 신뢰, 비신뢰 영역을 구분하지 않고 enclave 내에서 동작할 코드에 대해서만 구현하면 된다는 것이다. 이는 앞서 Fig.2에서 살펴본 Intel SGX SDK의 경우와 다르게, SGX에 대한 배경 지식이 없는 개발자도 손쉽게 SGX 환경에서 동작할 프로그램을 구현할 수 있다는 장점을 갖는다.

EDP는 기존 SGX 환경에서 문제가 되었던 여러 보안 취약점들을 해결하기 위해 Rust를 사용하는 것 외에도 다양한 보안 기법들을 적용하였다. 특히 주목할만한 점은 Intel SGX SDK가 제공하는 ABI를 따르지 않고, 부채널 공격들에 강건한 디자인을 위해 자체 ABI를 정의하였다는 것이다. 여기서 그치지 않고, API 또한 usercall이라는 자체 인터페이스를 정의하였고, 이 인터페이스 개수를 20개 이하로 제한함으로써 공격 표면을 최소화하였다. 구체적으로, EDP는 보안성 강화를 위해 환경 변수 설정, 타임아웃, 파일 시스템, 프로세스 생성 등과 관련된 Rust 표준 API 사용을 근본적으로 허용하지 않도록 설계되어 있다. EDP는 자체 ABI를 포함하여 전체 코드가 오픈 소스로 공개되어 있으며 리눅스 환경과 윈도우 환경에서 사용할 수 있다.

위와 같은 우수성을 갖지만, EDP 역시 상용 TEE 개발 프레임워크들과 비교했을 때 상대적 단점이 존재한다. 첫 번째로, 백엔드 TEE로는 오직 SGX만을 지원하며, 추가적인 백엔드 아키텍처를 지원하기 위한 확장이 용이하지 않은 디자인을 갖는다. 두 번째로, 사용 가능한 표준 API에 대해 제한함으로써 보안성을 강화하는 디자인을 선택했기에 그에 대한 트레이드 오프로 어플리케이션 구현 시 불가피한 기능적 제약이 존재한다. 마지막으로 Rust 프로그래밍 언어가 고급 프로그래밍 언어 중에서는 낮은 런타임 오버헤드를 가지나, C/C++ 프로그래밍 대비 성능적으로 뒤처진다는 근본적인 한계가 있다.

4.2 Rust SGX SDK

바이두에서 개발한 Rust SGX SDK는 Fortanix EDP와 마찬가지로 Rust로 구현된 어플리케이션을 SGX 환경에서 안전하게 구동 가능케 해준다. EDP와 마찬가지로 Rust 프로그래밍 언어로 어플리케이션을 새롭게 구현해야 한다는 구현 비용이 존재하지만, Rust가 보장하는 메모리 안전성을 가지게 된다. 다만 Rust SGX SDK는 EDP와 비교했을 때 디자인 측면에서 매우 큰 차이를 보이는 요소가 있는데, 바로 Intel SGX SDK의 프로그래밍 모델과 인터페이스를 그대로 따른다는 것이다. Intel SGX SDK와 어플리케이션 사이에 Rust로 구현된 API를 중간 계층에 제공함으로써, C와 C++로 구현된 Intel SGX SDK의 API들과 Rust 어플리케이션 코드를 안전하게 바인딩해주는 효과를 가진다.

Rust SGX SDK의 경우, Intel SGX SDK와 동일한 ABI(Application Binary Interafce)를 사용하기 때문에 SGX 기술의 업데이트에 빠르게 대응할 수 있는 등 SGX 기술에 대한 높은 호환성을 갖는다. 하지만 Intel SGX SDK 코드상에 보안 취약점이 존재할 경우, 해당 취약점에 동일하게 노출된다는 단점이 있다. 또한, Rust를 단순히 Intel SGX SDK의 API들을 래핑(wrapping)하는 용도로만 활용하였기 때문에, Rust 프로그래밍 언어가 갖는 고유의 강점인 소유권 등과 같은 코드 패턴들이 EDP와 비교하면 효과적으로 적용되지 못했다는 한계를 갖는다.

4.3 Redhat Enarx

2020년 CCC 컨소시엄에서 공개된 레드햇 사의 Enarx 프레임워크는 앞서 소개한 두 Rust 기반 프레임워크와는 다르게, 궁극적으로 하이브리드 클라우드 또는 멀티 클라우드를 대상으로 플랫폼 애그노스틱(platform-agnostic)하게 TEE를 활용할 수 있게 하겠다는 목표를 갖는다. 이는 enclave 개발자가 클라우드 환경 내에서 어떠한 운영 체제나 프로세서의 조합인지에 대한 인지가 없더라도 TEE 기능을 수행할 수 있게 됨을 말한다.

이러한 특성을 달성하기 위해 Enarx에서는 기반 기술로 웹어셈블리(Web Assembly, WASM)를 활용한다. 웹어셈블리는 높은 이식성과 네이티브에 가까운 수준의 성능을 제공한다는 장점으로 인해, 웹

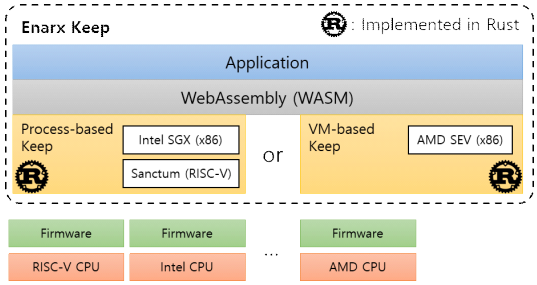

Fig. 4. Overall Architecture of Enarx

브라우저에서 활발히 활용되고 있다. 최근 자바 가상 머신(JVM, Java Virtual Machine)과 유사하게 웹어셈블리 코드가 실행될 대상 아키텍처를 알 수 없음에도 실행될 수 있는 애그노스틱 특성이 주목받으면서, 시스템 분야로의 적용에 대한 가능성이 검토되고 있으며 Enarx의 디자인이 이러한 배경에서 탄생하게 되었다.

Enarx는 Rust와 웹어셈블리, TEE가 결합한 기술이다. 어플리케이션과 런타임으로 구성된 "Keep"이라는 단위의 모듈로 실행이 되며 이 모듈이 백엔드 아키텍쳐에 해당하는 TEE 기술 위에서 동작한다. Enarx는 기본적으로 Rust로 구현되어 있으며 웹어셈블리로 컴파일되기 때문에, 메모리 안전성을 보장한다. 웹어셈블리를 활용함에 따라, Fig.4에 나타나듯이 Intel SGX나 Sanctum으로 대표되는 프로세스 기반 TEE와 AMD SEV와 같이 VM 기반 TEE를 모두 백엔드 아키텍처로 활용할 수 있다. 이는 기존 TEE들과는 차별화된 새로운 특성으로, CPU 하드웨어 공급자 또는 클라우드 공급자의 종류에 무관하게 해당 기술을 활용할 수 있다는 점에서 독보적인 강점을 가진다. 하지만 해당 기술은 논문 작성 시점 기준으로 아직 프로토타입조차 없을 정도로 성숙하지 않았다.

V. TEE 개발 프레임워크 비교분석

본 절에서는 앞서 분석한 TEE 개발 프레임워크들을 객관적으로 비교하기 위한 여러 가지 척도들을 분류하고, TEE 개발 프레임워크를 선택하는 기준을 수립하는 데 있어 지표로 활용하기 위해 각 척도에 대한 비교 분석을 수행한다. 본 연구와 유사하게 TEE 개발 프레임워크에 대한 비교 분석을 수행한 연구는 2019년 CCS 학회에 발표된 enclave의 메모리 취약점 분석 논문[30]이 있는데, 메모리 취약

Table 1. An overview of the differences in existing TEE frameworks

Categories	Aspects	Intel SGX SDK	Graphene -SGX	Asylo	Open Enclave	EDP
Overview	Programming Language	C, C++	C	C++	C, C++	Rust
	Vendor	Intel	Academia (Oscalab)	Google	Microsoft	Fortanix
	Intel SDK dependency	-	No	Yes	Yes	No
	familiarity with SGX	Required	Required	Required	Required	Not required
	Docker/Cloud support	Yes (Docker)	Yes (Docker)	Yes (Docker)	Yes (Azure)	No
Security	Remote attestation	Yes	Yes	Yes	Yes	Yes
	ABI-level attacks	1	2	1	1	0.5
	API-level attacks	0.5	5	4	4	1
	Memory safety	No	No	No	No	Yes
Neutrality	OS support	Linux, Windows	Linux	Linux	Linux, Windows	Linux, Windows
	Supporting TEE (arch)	SGX (x86)	SGX (x86)	SGX, SGX simulation (x86)	SGX (x86), TrusZone (ARM)	SGX (x86)
Functionality	Standard library	own libc (SGX libc)	glibc	own libc	own libc (oelibc)	Rust std
	System call restriction	No	No	No	No	Yes
	Library linking	static only	both static & dynamic	static only	static only	static only
	3rd party attestation	Yes	No	No	No	Yes
	Serialization support	No (needs porting)	No (needs porting)	Yes (protobuf)	No (needs porting)	Yes (Rust serde)

점과 관련된 보안성 관점에서 각 TEE 프레임워크들에 대한 분석을 수행하였다. 해당 연구는 보안 측면에서 기존 프레임워크들의 API 및 ABI가 얼마나 안전하게 설계되었는가에 대해서만 초점을 맞추었다. 다시 말하면, 프레임워크의 사용성이나 상호 운용성, 성능 등과 같은 다각도의 지표들에 대한 비교 분석이 수행되었다고 보기 어렵다. 본 논문에서는 보안성뿐만 아니라, 각 프레임워크의 기반 기술에 대한 심층적이고 객관적인 비교 분석을 통해 목적에 맞는 TEE 개발 프레임워크 선정을 가능케 하고자 한다.

비교에 앞서 분석한 8개의 TEE 어플리케이션 개발 프레임워크 중, Enarx, Rust SGX SDK와 SCONE을 제외한 5개의 프레임워크를 기준으로 심층 분석을 수행하고자 한다. Enarx의 경우, 아직 생태계가 형성되지 않았다고 봐도 무방할 정도로 개발 초기 단계이므로 본 비교에서 제외한다. Rust SGX SDK 또한 Rust API로 구현되었다는 점 외에는 기타 특성이 Intel SGX SDK에 수렴하기 때문에 비교 분석 대상에서 제외하였다. 마지막으로 SCONE은 소스 코드에 대한 접근이 어려워 코드

레벨 분석에 제약이 존재하였기 때문에 제외한다.

Table 1은 각 프레임워크 비교를 위한 기준 지표들과 해당 항목들에 대한 지원 여부, 특성 및 예시 등을 정리한 것이다. 상위 분류 항목은 크게 각 프레임워크에 대한 개요, 보안성, 중립성, 기능성 총 4가지로 구분하였다. 4가지 상위 항목에 대한 세부 하위 항목 중 보안성에서 ABI 레벨 공격 및 API 레벨 공격에 대해 취약성을 갖는지에 대한 여부는 기존 enclave 취약점 분석 논문[30] 결과를 참고하였으며, 그 외의 항목들은 소스 코드, 관련 논문 및 기술 문서로부터 파악하였다.

기능성 항목에서 라이브러리 링킹(linking)은 어플리케이션을 빌드할 때 링킹 시 활용할 수 있는 라이브러리의 종류를 말하며, Graphene-SGX를 제외하고는 모두 정적 라이브러리(static library)에 대해서만 허용한다. Graphene-SGX의 경우, 앞서 설명한 명세서 기반 무결성 검증을 통해 런타임에 동적 라이브러리에 대한 안전한 로딩이 가능하도록 구현하였다. 또한, 서드 파티(3rd party) 검증은 Intel에서 SGX 사용자들의 요구 사항을 반영하여 새롭게 개발한 기술로, 원격 검증 시 Intel에서 관리하는 검증 서버로부터 검사를 받는 것이 아닌 서드 파티에서 자체 원격 검증 서비스를 구축할 수 있도록 해주는 기술이다. 이는 현재 Intel SGX SDK와 EDP에서만 사용할 수 있다. 마지막으로 직렬화(serialization) 지원 여부는 protobuf와 같은 데이터 직렬화 통신 기술이 프레임워크에 적용되었는지를 비교한 항목이다.

VI. TEE 개발 프레임워크 선택 기준 제안

본 절에서는 앞서 비교 분석한 내용을 바탕으로, 컨피덴셜 컴퓨팅을 위한 어플리케이션을 개발함에 있어 목적에 맞는 TEE 프레임워크를 알맞게 선택하기 위한 기준을 제안한다. 이후, 제안한 기준에 대해 위에서 비교 분석한 5가지 프레임워크에 대한 상대 평가를 통해 기준별 우선순위를 파악한다. 제안한 상위 기준 지표는 총 5가지로, 다음과 같다.

- 성능(Performance)
- 보안성(Security)
- 배치 비용(Deployment cost)
- 개발 비용(Implementation cost)
- 애그노스틱(Agnostic)

Table.1에서 성능과 관련된 보조 지표로는 직렬화 지원 여부와 구현 프로그래밍 언어가 있다. 기본적으로 Rust 프로그래밍 언어가 계속 발전 중이기는 하나, C 또는 C++ 프로그래밍 언어에 비교했을 때 런타임 오버헤드가 크며, 직렬화 지원을 통해 enclave 간 데이터 통신 및 SGX 관련 시스템 서비스와 enclave 간 통신에서 성능적 우수성을 가질 수 있다. 이때, Graphene의 경우 예외적으로 라이브러리 운영 체제 기반이기 때문에 초기 enclave 생성을 위한 오버헤드가 압도적으로 클 수밖에 없다. 따라서 어플리케이션에서 성능이 심각한 요소로 작용할 경우, Asylo 〉 Intel SGX SDK=OpenEnclave 〉 EDP 〉 Graphene-SGX의 순서로 선택하는 것이 효과적이라고 볼 수 있다.

성능 평가에 있어 보조 지표로 언급한 직렬화 지원 여부 및 구현 프로그래밍 언어와 같은 정성적인 보조 지표 외에도, 정량적인 보조 지표들에 대한 기준을 마련하는 것이 필수적이다. SGX 어플리케이션의 end-to-end throughput에 직접적인 영향을 미치는 정량적 지표는 크게 3가지로 볼 수 있다. 이는 각각 enclave 생성 시간, 메모리 접근 오버헤드, 그리고 enclave 출입 오버헤드이다. enclave 생성 시간은 enclave 영역 내에 포함된 코드 및 데이터 페이지의 크기에 비례하여 증가하는데, 앞서 언급했듯이 라이브러리 운영 체제를 enclave 내에 넣는 Graphene-SGX은 타 프레임워크들에 비해 큰 enclave 생성 시간을 가진다.

SGX의 경우 현재 하드웨어적인 제약으로 인해 EPC 물리 메모리의 크기를 최대 128MB까지만 설정할 수 있어 메모리 footprint에 따라 불가피한 페이징 오버헤드가 발생한다. 또한, 보안성을 위해 시스템 호출(system call)과 같은 운영 체제 권한이 필요한 명령어를 enclave 내에서 실행할 수 없으므로 해당 명령에 대한 처리를 위해서는 enclave 출입이 발생하며 이에 따른 오버헤드가 발생한다. 두 요소에 대해서는 시스템 호출 빈도 등과 같이 어플리케이션의 특성에 영향을 받기 때문에 프레임워크 간 비교 및 평가를 통한 일반화된 결과를 도출하기 어렵다. 따라서 개발자가 구현하고자 하는 어플리케이션 또는 클라우드 서비스를 직접 포팅하여 비교를 수행하는 과정이 필요하다.

두 번째로, 보안성 지표는 기존 enclave 런타임 취약점 분석 연구[30]를 바탕으로 평가를 수행하였다. 해당 연구에서는 ABI 레벨 취약점 3개, API

레벨 취약점 7개에 대해 프레임워크에 대한 취약성 평가를 수행하였다. 구체적으로, 실제 취약점 공격(exploit)을 수행이 가능했던 케이스와 improper sanitization에 의해서만 취약점 공격이 가능한 경우, 이론상으로 취약성이 파악된 경우로 구분하여 분류하였다. 본 논문에서는 10개의 취약점에 대해 동일 가중치를 부여하여 평가하였으며, improper sanitization에 의해서만 조건부 취약점 공격이 가능한 경우는 절반의 가중치로 평가하였다. 이에 따르면 EDP 〉 Intel SGX SDK 〉 Asylo = OpenEnclave 〉 Graphene-SGX 순서로 높은 보안성을 갖는다. 각 프레임워크 별 취약점 개수는 Table.1에 표기하였다. 이때, EDP를 Intel SGX SDK보다 높은 보안성을 갖추었다고 평가한 이유는 Rust로 구현되어 enclave 내의 소스 코드에 존재하는 메모리 취약점에 대해서도 강인하기 때문이다.

배치 비용과 관련된 보조 지표는 컨테이너/클라우드 지원 여부, Intel SGX SDK 의존성 여부, 지원 운영 체제의 개수가 있다. Docker 컨테이너 등의 형태로 미리 생성된 바이너리를 지원함으로써, 높은 재현성(reproducibility)을 제공할 경우 개발자에게 편의성을 제공할 수 있다. 또한, 새로운 SGX 기능이 추가되면, Intel SGX SDK가 가장 먼저 업데이트가 될 것이고 이에 의존성을 가진 프레임워크들이 최신 기술에 대한 패치를 선제적으로 수행할 수 있다. 마지막으로 다양한 운영 체제를 지원하면 서비스 공급자가 다양한 실행 환경에서 클라우드 서비스를 운용하는 것이 가능하다.

본 지표의 평가는 클라우드/컨테이너 지원 여부 및 SDK 의존성 여부를 동일 가중치로 놓고 합산한 뒤, 지원하는 운영 체제의 수를 곱하여 산출하였다. 사실상 프레임워크 별 지원 운영 체제의 경우 리눅스만을 지원하거나 윈도우 및 리눅스 환경을 모두 지원하는 경우 두 가지로 구분될 수 있다. 특정 프레임워크가 2개의 운영 체제를 모두 지원한다면 클라우드 지원 여부 및 SGX 의존성 여부에 따른 장점을 각 환경에서 모두 누릴 수 있다. 이에 따라, Intel SGX SDK 〉 OpenEnclave 〉 Asylo = EDP 〉 Graphene-SGX 순으로 배치 비용 측면에서 효과적이다.

SGX에 대한 지식이 필요한지와 동적 라이브러리 지원은 개발 비용과 관련이 있다. SGX 지식이 필요하지 않다면, TEE 개발에 대한 문턱이 낮아 개발 인력 확보 측면에서 비용이 절감된다. 두 번째로 상용 어플리케이션들은 동적 라이브러리를 사용하는 경우가 비일비재한데, 만약 동적 라이브러리에 대한 링킹을 지원하지 않는다면 이를 SGX에서 동작 가능한 정적 라이브러리로 컴파일하는 포팅 과정이 필요하다. 위 두 가지 보조 지표의 가중치를 동일하게 놓았을 때, Graphene-SGX 〉 EDP 〉 Intel SGX SDK = Graphene-SGX = Asylo 순서로 개발 비용이 적다. SGX 관련 지식이 필요하지 않은 두 개의 프레임워크 중, Graphene-SGX를 EDP보다 높은 우선순위로 선정한 것은 비교적 최근에 개발된 Rust에 비해 C/C++ 프로그래밍에 대한 친숙도가 더 높다는 점을 감안했기 때문이다.

마지막으로 애그노스틱 항목은 SGX 이외의 TEE 기술을 지원할 수 있는지와 관련된 기준이다. 이는 x86 기반 서버와 많은 양의 IoT 기기가 공존하며 상호 통신하는 차세대 에지 컴퓨팅 플랫폼에서 활용되기 위해 매우 중요한 요소이다. 애그노스틱 지표의 우수성은 지원하는 TEE 기술의 개수를 바탕으로 평가를 수행하였다. 현시점 기준 이종의 TEE를 지원하는 것은 OpenEnclave 뿐이며, 기술 문서에 따르면 Asylo 또한 플랫폼 애그노스틱 보장을 목표로 하기에 OpenEnclave 〉 Asylo 〉 Intel SGX SDK = Graphene-SGX = EDP 순으로 애그노스틱 기술에 친화적이다.

실제 어플리케이션 개발 시, 하나의 기준만을 고려하여 활용할 프레임워크를 선정하기보다는 복합적인 요소를 고려하는 것이 일반적이다. 따라서 본 연구에서 제안한 5가지 기준이 복합적으로 조합된 우선순위 평가 방법이 필요하다. 예를 들면, 성능과 배치 비용의 기준에 대해 높은 우선순위를 갖지만, 개발 비용에 대해서는 낮은 우선순위를 가지는 시나리오를 가정할 수 있다. 이 경우, 본 연구에서 제안한 5가지 기준에 대해 분석적 계층화(AHP, Analytic Hierarchy Process) 기법을 적용함으로써 복합적인 평가가 가능하다. 본 논문에서 제안한 5가지 지표 및 보조 지표를 바탕으로 계층화가 가능하며, 서비스의 요구 사항에 따라 5가지 지표의 상대적 중요도를 개발자의 수요에 맞춰 결정하여 평가를 수행하면 된다. 이때 보조 지표의 가중치 또한 요구 사항에 따라 변경할 경우, 더 정교한 프레임워크 선정 평가가 가능하다.

VII. 결 론

본 연구는 차세대 클라우드 보안 핵심 기술로 떠오르고 있는 컨피덴셜 컴퓨팅 환경에서의 어플리케이션 개발을 위해 고려해야 할 기준을 제시한다. 이를 위해 클라우드 환경에서 생태계 구축이 빠르게 이루어지고 있는 Intel SGX 기술을 중심으로 학계 및 산업계에서 제안된 8개의 대표적인 상용 TEE 개발 프레임워크들에 대한 분석을 수행하였다. 이 중, 오픈 소스로 공개되었고 기술적으로 성숙한 5개의 프레임워크에 대해 기준 지표 및 보조 지표를 설정하고 이에 대한 비교를 통해 TEE 개발 프레임워크의 선택에 있어 고려해야 할 기준들을 비교 분석 내용을 바탕으로 정립하였다.

앞으로 개발될 컨피덴셜 컴퓨팅 기반 어플리케이션들은 각자 서비스 모델, 규모, 참여 가능 인력 등의 요인들로 인해 SGX와 같은 TEE 기술을 적용하기 위해 선택할 프레임워크가 서비스 개발자마다 다를 수 있다. 본 연구에서 제안한 선택 기준들은 이러한 상황에서 목적에 맞는 적절한 프레임워크를 선정하는 데 가이드라인이 될 것으로 판단된다. 향후 동일한 어플리케이션 또는 클라우드 서비스를 다양한 TEE 프레임워크들로 직접 개발해봄으로써, 본 연구에서 제안한 정성적 지표들뿐만 아니라 성능 측면에서 정량적 지표 기준들을 마련하고 직접 비교해보는 연구를 진행하여 프레임워크 선택 기준을 다채화하고 선명히 할 수 있는 방법론을 연구할 예정이다.

References

[1] H. Li, K. Ota, and M. Dong, "Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing," IEEE Network, vol. 32, no. 1, pp. 96-101, Jan.-Feb. 2018.

[2] J.Han, S. Kim, T. Kim, and D. Han, "Toward scaling hardware security module for emerging cloud services," Proceedings of the 4th Workshop on System Software for Trusted Execution, pp. 1-6, Oct. 2019.

[3] R. Mijumbi, J. Serrat, and J.L. Gorricho, "Network Function Virtualization: State-of-the-Art and Research Challenges," IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 236-262, Firstquarter 2016.

[4] A. Baumann, M. Peinado, and G. Hunt, "Shielding applications from an untrusted cloud with haven," Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation, pp. 267-283, Oct. 2014.

[5] A.J. Feldman, W.P. Zeller, M.J. Freedman, and E.W. Felten, "SPORC: Group Collaboration using Untrusted Cloud Resources." Proceedings of the 9th USENIX Symposium on Operating Systems De-sign and Implementation, pp. 337-350, Oct. 2010.

[6] F. McKeen, I. Alexandrovich, A. Berenzon, and C.V. Rozas, "Innovative Instructions and Software Model for Isolated Execution," Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, pp.1-1 Jun. 2013.

[7] J.M. McCune, B.J. Parno, A. Perrig, M.K. Reiter, "Flicker: an execution infrastructure for tcb minimization," Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems, pp.315-328, Apr. 2008.

[8] J. McCune, Y. Li, N. Qu, Z. Zhou, and A. Datta, "TrustVisor: Efficient TCB reduction and attestation," Proceedings of the 2010 IEEE Symposium on Security and Privacy, pp.143-158, May, 2010.

[9] Azure Confidential Computing, https://azure.microsoft.com/en-us/blog/introducing-azureconfidential-computing, Accessed: Feb 2021. [Online]

[10] Install SGX, Alibaba Cloud ECS,

https://www.alibabacloud.com/help/doc-detail/108507.htm, Accessed: Mar. 2021. [Online].
[11] Confidential Computing Consortium, https://confidentialcomputing.io/, Accssed: Mar. 2021. [Online].
[12] C.C. Tsai, D.E. Porter, and M. Vij, "Graphene-sgx: A practical library OS for unmodified applications on SGX." Proceedings of the 2017 USENIX Annual Technical Conference, pp. 645-658, Jul. 2017.
[13] S. Arnautov, B. Trach, F. Gregor, and T. Knauth, "SCONE: Secure linux containers with intel SGX." Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, pp.689-703, Nov. 2016.
[14] S. Shinde, D.L. Tien, S. Tople, and P. Saxena, "Panoply: Low-TCB Linux Applications With SGX Enclaves." Proceedings of the Network and Distributed System Security Symposium, Feb. 2017.
[15] M. Orenbach, P. Lifshits, and M. Minkin, "Eleos: ExitLess OS services for SGX enclaves." Proceedings of the Twelfth European Conference on Computer Systems. pp.238-253, Apr. 2017.
[16] ARM. "Building a secure system using trustzone technology", PRD29-GENC-009492C, Dec. 2008.
[17] D. KAPLAN, J. POWELL, and T. WOLLER, "AMD memory encryption.", White paper, AMD Inc, Apr. 2016.
[18] K. ASANOVIC and D. PATTERSON, "Instruction sets should be free: The case for risc-v". UCB/EECS-2014-146, EECS Department, University of California, Berkeley, Aug. 2014.
[19] V. COSTAN, I. Lebedev, and S. Devadas, "Sanctum: Minimal hardware extensions for strong software isolation." Proceedings of the 25th USENIX Security Symposium, pp-857-874, Aug. 2016.
[20] Su-In Kang and Huy-Kang Kim, "Improvemnet of Online Game Security using Intel SGX", Review of Korea Institute of Information Security and Cryptology, 27(4), pp. 22-26, Aug. 2017.
[21] Dongyeob Kim, Hoorin Park, and Wonjun Lee, "Improvement of System Log Security using Intel SGX", Korea Software Congress, 45(2), pp. 1085-1087, Dec. 2018.
[22] P. Jain, S.J. Desai, S. Kim, M.W. Shih, J. Lee, C. Choi, Y. Shin, T. Kim, B.B. Kang, and D. Han, "OpenSGX: An open platform for SGX research," Proceedings of the Network and Distributed System Security Symposium, Feb. 2016.
[23] Intel Software Guard Extensions SDK. https://software.intel.com/en-us/sgx-sdk, Accessed: Mar. 2021. [Online]
[24] Google Asylo, https://asylo.dev/, Accessed: Mar. 2021. [Online]
[25] Open Enclave SDK, Microsoft, https://openenclave.io/sdk/, Accessed: Mar. 2021. [Online]
[26] N. D. Matsakis and F. S. Klock, "The rust language," Proceedings of the 2014 ACM SIGAda annual conference on High integrity language technology, pp. 103 - 104, Oct. 2014.
[27] Fortanix EDP, https://edp.fortanix.com/, Accessed: Mar. 2021. [Online]
[28] Rust SGX SDK, https://github.com/apache/incubator-teaclave-sgx-sdk, Accessed: Mar. 2021. [Online]
[29] ENARX, Redhat, https://enarx.dev/, Accessed: Mar. 2021. [Online]
[30] J.V. Bulck, D. Oswald, E. Marin, and

A. Aldoseri, "A tale of two worlds: Assessing the vulnerability of enclave shielding runtimes." Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp.1741-1758, Nov. 2019.

[31] S. Mofrad, F. Zhang, S. Lu, and W. Shi, "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, pp.1-8, Jun. 2018.

[32] M. Pieter, J. Gotzfried, and R.D. Clercq, "Hardware-based trusted computing architectures for isolation and attestation." IEEE Transactions on Computers, vol. 67, no. 3, pp.361-374, Mar. 2018.

〈저자소개〉

김 성 민 (Seongmin Kim) 정회원
2012년 2월: 한국과학기술원 전기 및 전자공학과 졸업
2014년 2월: 한국과학기술원 전기 및 전자공학과 석사
2019년 2월: 한국과학기술원 정보보호대학원 박사
2019년 9월~2020년 8월: 삼성전자 삼성리서치 Staff Engineer
2020년 9월~현재: 성신여자대학교 융합보안공학과 조교수
〈관심분야〉 신뢰 실행 환경, 클라우드 컴퓨팅, 시스템 보안