신뢰실행환경을 활용한 스마트 컨트랙트 실행 보호 기술 개발

팀 TrustForge

부산대학교 정보컴퓨터공학부

202155651 이준태

202155648 위재준

202055581 이준혁

지도교수 **권동현**

Contents

과제의 목표

1.1) 과제 배경

최근 블록체인 기술은 금융, 의료, 물류 등 다양한 분야에서 활용되고 있으며, 특히 Hyperledger Fabric과 같은 프라이빗 블록체인은 높은 확장성과 접근제어 기능으로 주목받고 있다. 그러나 Fabric의 체인코드는 일반 운영체제 환경에서 실행되어 보안상의 한계가 존재하며, 민감한 연산 결과의 위변조 위험 또한 존재한다.

1.2) 과제 세부 목표

본 과제는 WebAssembly 기반의 체인코드를 ARM 보드의 신뢰 실행 환경(TEE: Trusted Execution Environment)에서 실행하고, 실행 결과를 Hyperledger Fabric 블록체인과 연동하는 보안 실행 아키텍처를 구현하는 것을 목표로 한다. 이를 통해 체인코드 실행의 무결성과 신뢰성을 확보하고, 스마트 계약의 보안성을 향상시키고자 한다.

- TEE 환경에서 WebAssembly 체인코드 실행 구조 구현
- TEE 내부 실행 결과에 대한 서명 처리 및 전달
- Hyperledger Fabric과의 연동을 위한 Proxy 체인코드 구성
- 전체 시스템의 실험 및 보안성 검증

대상 문제 및 요구조건 분석

2.1) 유사 시스템 분석

현재 스마트 컨트랙트(Smart Contract)는 대부분 블록체인 환경에서 가상 머신(예: EVM) 상에서 실행되며, 코드의 투명성과 불변성을 기반으로 신뢰성을 확보한다. 그러나 이와 같은 특성은 악의적인 행위자에게 오히려 코드 분석과 공격 벡터 노출이라는 위험을 동반하기도 한다.

최근에는 Wasm (WebAssembly) 기반의 스마트 컨트랙트 플랫폼(예: Polkadot)이 주목받고 있으며, 이는 성능과 이식성 측면에서 기존 EVM 대비 우수한 평가를 받고 있다. 하지만 Wasm 자체는 실행 보안 보호 수단을 제공하지 않기 때문에, 스마트 컨트랙트 코드가 실행 중 민감 데이터를 다루거나 공격에 노출되는 문제는 여전히 해결되지 않고 있다.

이를 보완하기 위한 방안으로 ARM TrustZone 등 다양한 신뢰실행환경(Trusted Execution Environment, TEE) 기술이 개발되었지만, 이러한 기술을 Wasm 기반의 스마트 컨트랙트 실행 환경에 직접적으로 결합하려는 시도는 아직 미흡한 실정이다.

2.2) 문제점 분석

- WebAssembly 기반 스마트 컨트랙트는 성능 면에서는 뛰어나지만, 실행 시 민감한 데이터 보호에 취약하다.
- 현재 TEE를 활용한 보호 기술은 존재하나, 범용 Wasm 환경과의 호환 및 통합이 부족하다.
- TEE 기반 보호 기술 적용 시 성능 저하나 개발 복잡도 증가 문제가 발생할 수 있으며, 이에 대한 정량적 평가가 부족하다.

2.3) 시스템의 필요성

스마트 컨트랙트는 탈중앙화 애플리케이션에서 핵심적인 역할을 수행하므로, 코드 실행 시점에서의 보안성 확보는 필수적이다. 특히, 민감한 데이터를 처리하거나 프라이버시 보호가 필요한 애플리케이션에서는 기존의 블록체인 모델만으로는 보호 수준이 부족하다.

WebAssembly 기반 스마트 컨트랙트에 TEE를 연계하여 실행 환경을 보호하면, 데이터 무결성과 기밀성을 보장하면서도 높은 성능을 유지할 수 있다. 이러한 시스템은 향후 탈중앙화 금융(DeFi), 의료정보 처리, 기밀투표 시스템 등 다양한 고신뢰 응용 분야에 필수적인 기반 기술로 활용될 수 있다.

2.4) 요구사항 분석

기능적 요구 사항

- WebAssembly 기반의 스마트 컨트랙트를 TEE 내에서 실행 가능하도록 구성
- TEE 내에서 스마트 컨트랙트 실행 시, 외부의 데이터 및 상태 접근 제어 구현
- TEE 외부와의 통신 채널 확보
- 기존 블록체인 노드와 연계 가능하도록 인터페이스 구현

비기능적 요구사항

- 성능 측정: 스마트 컨트랙트를 TEE에서 실행했을 때의 실행 시간 및 자원 분석
- 보안성 평가: 기존 환경 대비 정보 노출 가능성
- 이식성: 다양한 플랫폼 및 하드웨어 환경에서의 실행 가능성 확보
- 개발 편의성: 기존 Wasm 기반 개발 환경과의 호환성 유지

현실적 제약 사항 분석 결과 및 대책

3.1) 제약 사항 정리

제약 사항	설명
Arm TrustZone의 제한된 리소스	- 메모리 용량이 매우 제한적이며, OP-TEE의 기본 설정도 보수적임 - 시스템 콜 불가능, 일반적인 OS 기능 미지원 (파일 시스템, 네트워크 등)
TrustZone의 원격 인증 미지원	- TrustZone은 SGX와 달리 RA(Remote Attestation) 기능이 없어서 외부 서비스는 해당 디바이스의 신뢰성이나 실행 중 코드의 무결성을 검증할 수 없음
OP-TEE의 실행 파일 제약	- 신뢰할 수 있는 애플리케이션은 반드시 미리 서명되어야 하며 실행 중에 코드를 동적으로 바꾸기 어려움
성능 저하 우려	- TEE는 일반 실행 환경보다 느릴 수 있음
TLS 기능 미제공	- OP-TEE는 TLS(SSL) 스택 미구현으로 인해 일반적인 보안 채널을 구축하기 어려움

3.2) 대책 정리

제약 사항	대응 방안
Arm TrustZone의 제한된 리소스	- WAMR(WebAssembly Micro Runtime): 작고 효율적인 Wasm 실행 환경으로, TEE에 적합 - 메모리 확장: OP-TEE 커널을 수정하여 공유 메모리 9MB, 보안 메모리 27MB 확보 - WASI(WebAssembly System Interface) 일부 지원: Wasm 앱이 POSIX처럼 시스템 기능을 요청하도록 함
TrustZone의 원격 인증 미지원	- Intel SGX의 RA 프로토콜을 단순화 및 수정하여 TrustZone에 적용 - Wasm 앱에서 직접 RA 요청 가능하도록 WASI-RA 확장 구현 - Wasm 코드의 해시값을 측정해 서명된 증거(evidence)로 생성
OP-TEE의 실행 파일 제약	- WATZ는 AOT(Ahead-of-Time) 컴파일된 Wasm 바이트코드를 런타임에 로드해 실행 - 바이트코드는 기존에 서명된 실행파일이 아니어도 TEE 내에서 샌드박스 내에서 안전하게 실행 가능
성능 저하 우려	- AOT 컴파일 → 실행 중 JIT나 인터프리터 없이 즉시 실행, 성능 28배 향상 - Wasm 앱은 네이티브보다 평균 1.6 2.1배 느리지만 보안성이 보완됨
TLS 기능 미제공	- TLS를 쓰지 않고도 보안을 유지하는 자체 RA 프로토콜 개발 - ECDHE로 키 합의, AES-GCM으로 데이터 암복호화, MAC으로 메시지 무결성 보장

설계 문서

4.1) 1) 시스템 구성도

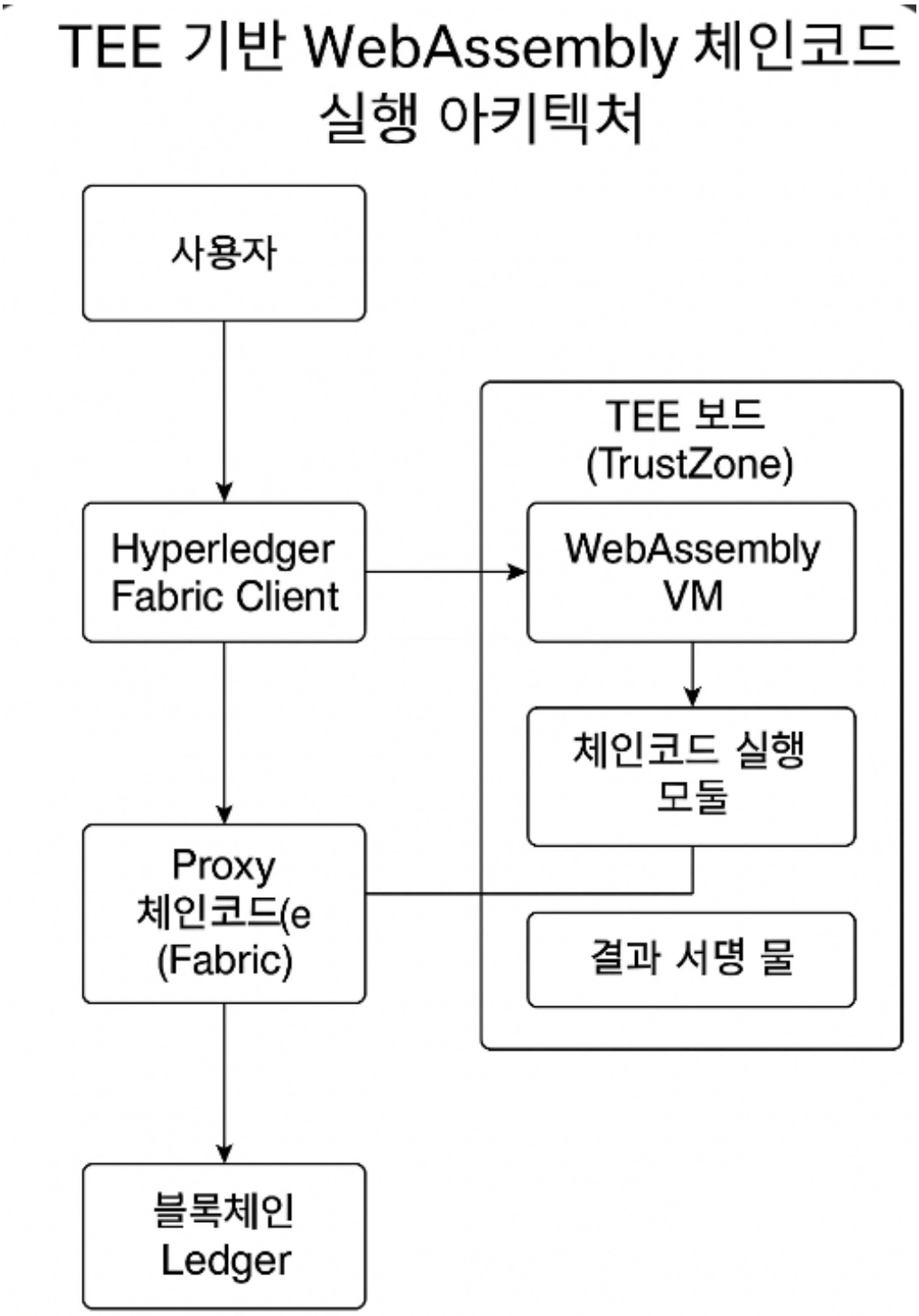

Figure 1: TEE 기반 WebAssembly 체인코드 실행 아키텍처

4.2) 모듈별 구성 설명

본 시스템은 체인코드(스마트 계약)를 보안 환경에서 안전하게 실행하고, 그 결과를 Hyperledger Fabric 블록체인에 저장하는 구조이다. 이를 위해 아래와 같은 모듈들이 협력한다:

- **1. 사용자 클라이언트**
  사용자가 체인코드 실행을 요청하는 프로그램이다. 블록체인 네트워크에 "이 계약을 실행해달라"는 트랜잭션을 보낸다.

- **2. 프록시 체인코드 (Proxy Chaincode)**
  Hyperledger Fabric 내부에 설치되며, 체인코드 실행 요청을 TEE 보드로 전달한다. 실행 결과를 받아 블록체인에 저장하는 역할을 하며, 사용자 입장에서는 체인코드가 내부에서 실행된 것처럼 보인다.

- **3. TEE 보드 (보안 실행 환경)**
  ARM TrustZone 기반의 보안 공간이다. 일반 시스템과 분리되어 있어 안전하게 체인코드를 실행할 수 있다. 다음과 같은 하위 모듈로 구성된다:
  - **Wasm 실행기**: WebAssembly 형식으로 작성된 체인코드를 불러와 실행한다.
  - **체인코드 처리기**: 실제 스마트 계약 로직을 수행한다. 예: 잔고 계산, 조건 비교 등
  - **결과 인증 모듈**: 실행 결과에 디지털 서명을 하여 조작되지 않았음을 보장한다.

- **4. 검증 서버 (Verifier)**
  TEE에서 보낸 결과가 진짜인지 확인한다. 검증이 끝나면 민감한 정보(예: 암호키)를 전달하며, 이 과정을 원격 증명(Remote Attestation)이라고 한다.

- **5. Hyperledger Fabric 블록체인**
  프록시 체인코드가 전달한 실행 결과를 블록에 저장한다. 이 결과는 변경할 수 없으며, 감사 및 검증 용도로 사용된다.

4.3) 사용 기술

1. **WebAssembly (Wasm)**

   웹 브라우저뿐 아니라 서버, IoT, 블록체인 등 다양한 환경에서 실행 가능한 **범용 바이트코드 형식**이다.

   체인코드를 작은 크기의 코드로 만들어 **빠르게 실행**하고, **보안도 확보**할 수 있다.

   본 과제에서는 체인코드를 Go 언어로 작성한 후, Wasm 형식으로 변환하여 실행한다.

2. **ARM TrustZone**

   스마트폰, IoT 기기에서 널리 사용되는 ARM CPU의 **하드웨어 보안 기능**이다.

   일반 영역과 분리된 **보안 세계 (secure world)**를 제공하여, 민감한 연산을 안전하게 처리할 수 있다.

   외부 공격자가 접근할 수 없는 공간에서 체인코드를 실행한다.

3. **OP-TEE**

   TrustZone용으로 개발된 **오픈소스 보안 운영체제**이다.

   TEE 환경에서 프로그램을 실행하고, 보안 API를 제공한다.

   본 과제에서 사용된 WATZ 시스템은 OP-TEE 위에서 실행된다.

4. **Hyperledger Fabric**

   기업용으로 설계된 **프라이빗 블록체인 플랫폼**이다.

   참가자와 권한을 제어할 수 있으며, 체인코드를 통해 자동화된 거래 처리를 수행한다.

   본 시스템에서는 체인코드 실행 결과를 블록에 기록하는 용도로 사용된다.

5. **Remote Attestation (원격 증명)**

   "이 결과는 보안 환경에서 실제로 실행되었다"는 것을 검증하는 기술이다.

   TEE 보드에서 체인코드 실행 결과에 대해 **디지털 서명**을 만들고, 외부 서버(Verifier)가 이를 검증한다.

   사용자는 결과를 신뢰할 수 있으며, 민감한 데이터를 안전하게 전달받을 수 있다.

추진 체계 및 일정

5.1) 개발 일정

월	계획
5월: 논문 분석 및 배경 학습	- 논문 전체 구조 파악 - TrustZone, Wasm, OP-TEE, RA 기본 개념 정리 - 착수보고서 및 지도확인서 제출
6월: 실험 환경 구축 및 기초 구현	- OP-TEE 설치 및 실습 - Wasm 실행 실습 - WaTZ 구조 파악
7월: WaTZ 기능 구현 및 수정 실습	- WAMR + WASI 통합 - AOT 실행 실습 - RA 프로토콜 구조 파악 - 중간보고서 및 중간평가표 제출
8월: 원격 인증 구현 실습	- RA 구현: 메시지 흐름 실험 - evidence 생성 및 검증 - WASI-RA 호출 테스트
9월: 성능 테스트 및 정리	- SQLite, Genann 실험 - 성능 수치 재현 및 시각화 - 최종보고서,최종평가표 제출
10월: 최종 정리 및 발표	- 졸업과제 발표심사 진행

구성원 역할 분담

이름	역할
이준태	- OP-TEE 환경 구축, WaTZ 실행, RA 기능 구현 - SQLite/Genann 테스트, 성능 측정 및 분석 - 발표자료 제작, 최종 발표 준비
이준혁	- OP-TEE 환경 구축, WaTZ 실행, RA 기능 구현 - SQLite/Genann 테스트, 성능 측정 및 분석 - 발표자료 제작, 최종 발표 준비
위재준	- OP-TEE 환경 구축, WaTZ 실행, RA 기능 구현 - SQLite/Genann 테스트, 성능 측정 및 분석 - 발표자료 제작, 최종 발표 준비

참고문헌

References

[1] Michele Rossi, Christian Platzer, and Frank Kargl, *WaTZ: A Trusted WebAssembly Runtime Environment with Remote Attestation for TrustZone*, NDSS, 2023.

[2] Linaro, *OP-TEE: Open Portable Trusted Execution Environment*, `https://www.op-tee.org/`

[3] Andreas Haas et al., *Bringing the Web up to Speed with WebAssembly*, ACM PLDI, 2017.

[4] Hyperledger Foundation, *Hyperledger Fabric Documentation*, `https://hyperledger-fabric.readthedocs.io/`