

ZoKrates

- Scalable Privacy-Preserving Off-Chain Computations

Jacob Eberhardt, Stefan Tai
TU Berlin

2020.05.06
이수연

Contents

1. Introduction
2. Background
3. Off-chaining Computations
4. Zokrates Implementation
5. Evaluation
6. Discussion & Outlook
7. Related Work
8. Conclusion

Introduction

블록체인은 여러 분야에서 쓰이기에 걸림돌이 되는 challenge 존재

이 중에 확장성과 privacy는 실제로 활용되기 위해서는 극복해야하는 문제

확장성 { 블록체인 거래 >> 네트워크의 모든 노드에서 검증되고 처리
유효성 검증 및 처리에 필요한 모든 데이터가 모든 노드에서 사용가능

privacy { public 블록체인 네트워크에서는 모든 데이터가 공개
기밀 정보 및 privacy 보호가 필요한 경우 충돌이 발생

해결 방법

off-chain 제안

확장성과 privacy 를 모두 해결하기 위함

◆ 주요 아이디어

블록체인 외부 리소스를 사용하여 **블록체인의 계산 노력과 데이터 저장을 줄이는 것**

➡ **블록체인의 기본 속성은 유지되어야 함**

◆ 목표

검증에 드는 비용을 줄여, 같은 비용으로 처리할 수 있는 거래량을 늘림

데이터는 공개되어 있지만 암호화되어 있어 privacy는 보호됨

그러면서도 이 데이터는 네트워크 상의 트랜잭션과 상호작용 가능

Contribution

기여 1 Non-interactive zero-knowledge proof 기반의 off-chain 처리 모델을 도입
블록 체인 시스템의 트랜잭션 처리량 및 privacy 보호를 향상

- ➡ off-chain에서 계산을 한 후 결과를 블록체인에 기록
- ➡ off-chain 계산 결과의 정확성을 증명하는 proof를 제공
- ➡ proof를 이용해 블록 체인에서 검증

proof 시스템이 만족해야하는 조건

- a) proof 검증은 원래 계산의 (재)실행보다 더 빨라야 함
- b) proof는 zero knowledge를 이용 즉, private 데이터를 공개하지 않고 증명할 수 있어야 함

Contribution

기여 2

zkSNARK를 proof 시스템으로 사용해 off-chain 처리 모델을 지원하는 소프트웨어 도구 set 인 [ZoKrates](#)를 소개

등장 원인

검증 체계에 적합한 암호화 기법이 있지만 블록 체인에 적용이 어려움

low level이고, 사용하기 어려운 환경에 적합한 연산이 필요

on-chain 검증은 매우 복잡 & 체계에 대한 해박한 지식 요구

Contribution

ZoKrates

- ◆ 개발자가 high level로 off-chain 계산을 편리하게 하는 도메인 별 언어를 정의
- ◆ proof 시스템의 low level 프로그래밍을 몰라도 연산 가능
 - ➡ domain-specific 코드를 검증 가능하도록 제한된 시스템으로 변환하는 컴파일러가 포함되어 있기 때문
- ◆ off-chain 프로그램을 실행하고 정확성을 증명하는 proof를 생성하는 데 도움
- ◆ 편리한 on-chain검증을 위해 Verification Smart Contract를 지원
 - ➡ off-chain 에서 생성 된 proof를 검증하고 off-chain 계산의 정확성을 확인
- ◆ Ethereum 블록 체인 대상이지만 설계 상으로는 다른 블록 체인 시스템과 호환 가능

2) zkSNARKs

Verifiable computation schemes 를 통해 신뢰할 수 없는 prover에게 계산을 위탁
prover : 계산을 수행하고 결과가 정확하다는 증거를 포함하여 결과를 반환

} 계산력이 약한 verifier에게 도움

ZkSNARK (zero-knowledge Succinct Non-interactive ARguments of Knowledge) 주요 특성

- ◆ Set of Verifiable computation schemes
- ◆ proof는 짧고 non-interactive
 - 즉, prover는 상호작용하지 않아도 verifier에게 확신을 줄 수 있음
- ◆ prover는 proof를 생성할 때 private 정보를 사용할 수 있으며 verifier는 해당 정보에 대해 알 수 없음
- ◆ 검증 비용은 입력의 계산 복잡성과 관계없음

2) zkSNARKs

- ◆ zkSNARK verification scheme 는
다항식에 기반한 abstraction인 QAP (Quadratic Arithmetic Programs)에 정의
→ Arithmetic Circuits 와 Rank-1-Constraint-Systems (R1CS)가 있음
- ◆ 산술 회로를 R1CS 및 QAP로 변환 할 수 있기 때문에 circuit이라고 부름

R1CS 프로그램을 미지수에 대한 set of condition 으로 인코딩
실행이 올바르게다는 것은 조건을 만족하는 variable assignment을 찾는 것과 같음
이러한 임무를 witness 이라고 함

초기 단계로, proof creation 및 verification 에 사용되는 common reference string (CRS)을 작성하기 위해
신뢰할 수 있는 일회성 setup을 수행해야 함 ?

Off-chaining Computations

- ◆ off 체인을 이용하면 성능,비용적인 측면과 개인정보보호 측면에서 이점이 있음
- ◆ 암호화 해시를 블록체인에 올리고 그 데이터를 이용하는게 적절한 방법이지만 정보의 내용을 유지하면서 외부 노드로 연산을 옮기는게 쉽지 않음
 - ➡ 결과를 신뢰 할 수 없기 때문

Off-chaining Computations

해결 방법

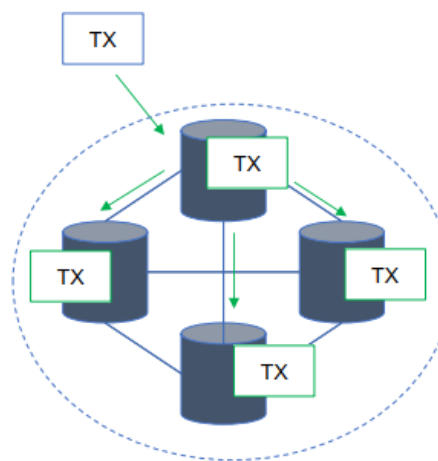
연산과 연산 결과가 올바름을 증명하는 proof를 게시 할 수 있도록
Verifiable computation schemes 를 채택 할 것을 제안

필요한 요구사항

짧은 proof와 저렴한 verification

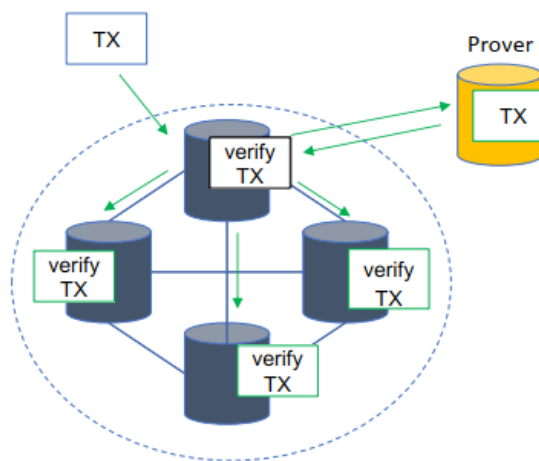
zero- knowledge

On-chain processing



Blockchain Network

Delegated computation



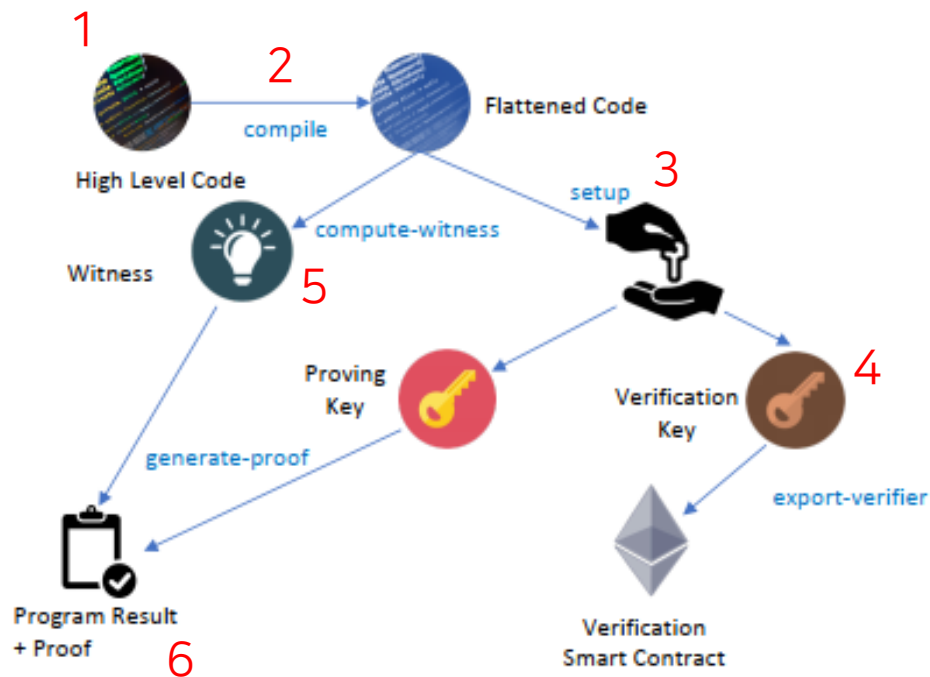
Blockchain Network

Zokrates Implementation

- ◆ specifying a provable program 부터 proof 생성 및 블록 체인에 대한 정확성 검증 까지 전체 프로세스를 지원
- ◆ 코드는 모두 오픈 소스, GitHub 공개
- ◆ proof verification을 위해 Ethereum 블록 체인을 사용

Zokrates Implementation

1. 개발자는 domain specific language (DSL)로 ZoKrates 프로그램을 지정
2. 프로그램은 R1CS로 쉽게 변환 될 수 있도록 flatten 코드로 컴파일
→ zkSNARK proof systems에서 쓰일 수 있음
3. flatten 코드를 기반으로 setup 단계가 수행
→ 긴 proving 키와 짧은 verification 키라는 두 개의 public 키가 생성됨
4. verification 키를 기반으로 verification smart contract이 생성되어 블록 체인에서 proof를 검증 가능
5. proof를 생성하기 위해 prover 는 "compute-witness"단계를 통해 flatten 프로그램을 실행
→ flatten 프로그램에 대한 솔루션을 찾는 것
6. proving 키와 솔루션을 사용하여 proof를 작성하고 이를 verification contract 에 전송하여 정확성을 검증



A. The Zokrates Language

◆ zkSNARK을 이용한 Verifiable computation schemes

: QAP (Quadratic Arithmetic Programs) 또는 Rank-1-Constraint-Systems (R1CS) 와 같은 수학 방정식을 이용

→ verification schemes을 구축하는 데 적합하지만 복잡한 계산 방식을 지정하는 것은 어렵고 불편

Zokrates

high level domain-specific language를 정의한 다음 R1CS로 변환하여
보다 편리하고 가까운 방식으로 오프 체인 계산을 지정할 수 있음

사용하기 쉽고, provable constraint systems으로 오버헤드가 거의 없이 변환됨

static scoping 기능이 있는 간단한 명령형 언어

다른 circuit 생성 언어 와 호환되며 DSL에 프로그램을 지정할 필요가 없음

1) code structure

- ◆ ZoKrates 프로그램의 entry point는 main function
 - public 및 private input을 argument로 가질 수 있으며 하나 이상의 값을 return
- ◆ 프로그램의 올바른 실행을 증명하는 proof가 블록 체인에 전송 될 때 public input은 public 정보가되지만, private input은 공개되지 않으며 prover의 비밀로 유지
- ◆ 임의의 함수를 정의하고 호출 가능
- ◆ 함수에는 자체 static scope가 있으며 함수 definition가 호출되기 전에 있어야함

2) Types

- ◆ ZoKrates의 기본 데이터 type은 prime field element
- ◆ This is an positive integer modulo a fixed prime number
- ◆ prime number의 크기는 254 bit
 - ➡ 이 크기로 인해 개발자는 대부분의 경우 prime field element를 단순한 부호 없는 정수 유형으로 생각
- ◆ binary type을 나타내기 위해 필드 요소가 단순히 값 0과 1로 제한될 수 있음

3) Operators

- ◆ 일반적인 산술 연산자가 지원되며 (예 : $+$, $-$, $*$, $/$) overflow 또는 나머지가 있는 나눗기가 없는 한 양의 정수 semantic을 가짐
- ◆ 일반적인 비교 연산자가 지원 (예 : $==$, $<=$)
- ◆ $==$ 연산자를 사용하여 assertion 을 정의
- ◆ Bool 연산자는 정의되어 있지 않지만 $AND(a; b) = ab$ 및 $OR(a; b) = 1 - (1 - a)(1 - b)$ 와 같이 함수로 묶인 산술 표현식으로 나타낼 수 있음

4) Control Flow

- ◆ For-Loops는 지원되지만 Verifiable computation schemes과 맞추어 줘야 해서 컴파일 프로세스 에서 loop가 statement로 바뀔 때 반복 횟수에 최댓값이 필요
- ◆ 같은 이유로 재귀는 지원되지 않음
- ◆ If-Else-Statements를 이용해 조건문 사용 가능

B. The ZoKrates Toolbox

ZoKrates DSL에 프로그램이 작성된 후, 올바른 실행을 증명하는 proof가 블록 체인에 전송되고 검증 될 때까지 아래 도구를 거침

지원하는 도구의 종류

- 1) Compiler
- 2) Witness Generator
- 3) Circuit Import
- 4) Setup and Proof Generation
- 5) Proof Generation
- 6) Contract Generator

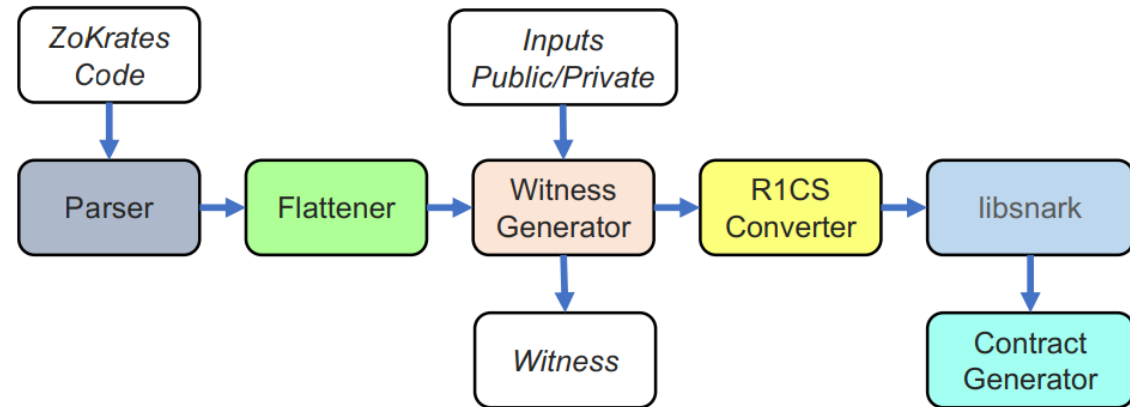


Fig. 3. ZoKrates Architecture.