Blockchain Development Workshop

Wallets, Tokens & zk-SNARKs

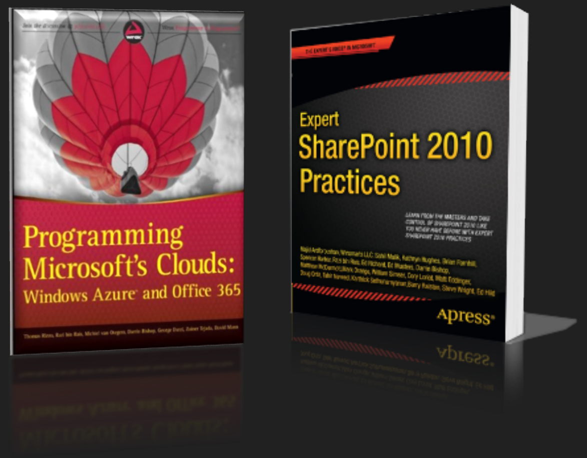
Razi Rais | www.razibinrais.com | @razibinrais

Who am I?

Enterprise Services – Microsoft | Blockchain & Identity

15+ Years | Architecture | Design | Development | Training

Web: www.razibinrais.com
Twitter: @razibinrais
LinkedIn: www.linkedin.com/in/razirais
Git: github.com/razi-rais


Programming Microsoft's Clouds: Windows Azure and Office 365
Expert SharePoint 2010 Practices
Apress

Agenda

- EOA, Contract Accounts, Payable Methods
- Working with Multi-Signature Wallets
- Tokens : Direct transfer, Delegate transfer to a contract
- zk-SNARKS and using them in Ethereum
- Q/A
- Demos & Labs

Hands-on Labs

Hands-on Labs: https://github.com/razi-rais/blockchain-workshop

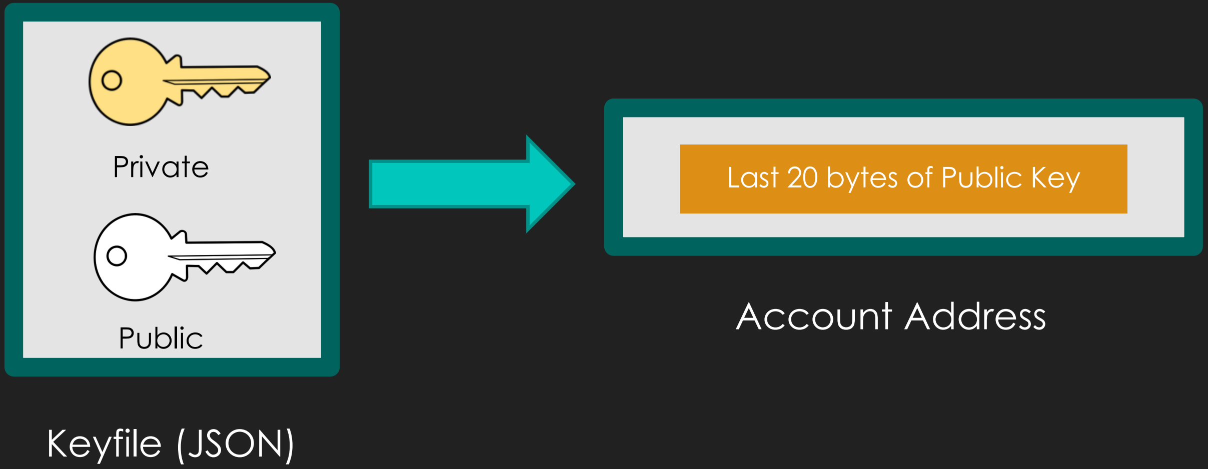
```
git clone https://github.com/razi-rais/blockchain-workshop.git

cd blockchain-workshop
```

Ethereum Accounts

Externally Owned Accounts | Contract Accounts

Externally Owned Accounts (EOA)



Contract Accounts

Every **contract** has an unique address

Contract is created by sending a transaction to 0x00 address

A contract registration transaction should contain **no ether value**
but a **data payload** containing compiled bytecode of the contract

How to manage value (Ether) in a contract?

1. Make sure contract has a **payable** function

2. Make sure contract can send/spend Ether, otherwise Ether will be locked/lost forever

If you want to burn Ether: Send them to special address 0x000000000000000000000000000000000000dEaD

MultiSigWallet

Ability to transfer ether to MultiSigWallet

Ability to add owners, set minimum signature count etc.

Ability to transfer ether from MultiSigWallet to EOA contract

Caveats: Security!

MultiSigWallet - Demo

Working with MultiSigWallet

MultiSigWallet - Lab

Working with MultiSigWallet

https://github.com/razi-rais/blockchain-workshop/tree/master/multi-sig-wallet

Tokens

Token defines an abstraction that can be owned (in Ethereum by EOA or contract)

Example: Utility, Assets, Equity, Identity, etc.

Tokens can be fungible or non-fungible

Caveats: Security, Regulatory challenges

Fungible & Non-Fungilble Tokens

A token is fungible when you can substitute any single unit of that token for another without any difference in its value or function

Example: Majority of ICOs (Utility or Equity) tokens

Standards: EIP20, EIP 777 (draft), EIP223 (draft)

Fungible & Non-Fungilble Tokens

A token is non-fungible when each represent a unique item and therefore is not interchangeable

Example: CyrptoKitty

Standards: ERC 721

Ethereum Tokens - Demo

Working with EIP20 Token

Token - Lab

Working with EIP20 Token

https://github.com/razi-rais/blockchain-workshop/tree/master/tokens

zk-SNARKs

Zk-SNARK → Zero-Knowledge Succinct Non-Interactive Argument of Knowledge

Zero-knowledge → Allows *prover* to prove to the *verifier* that a statement is true without revealing any information beyond the validity of the statement itself

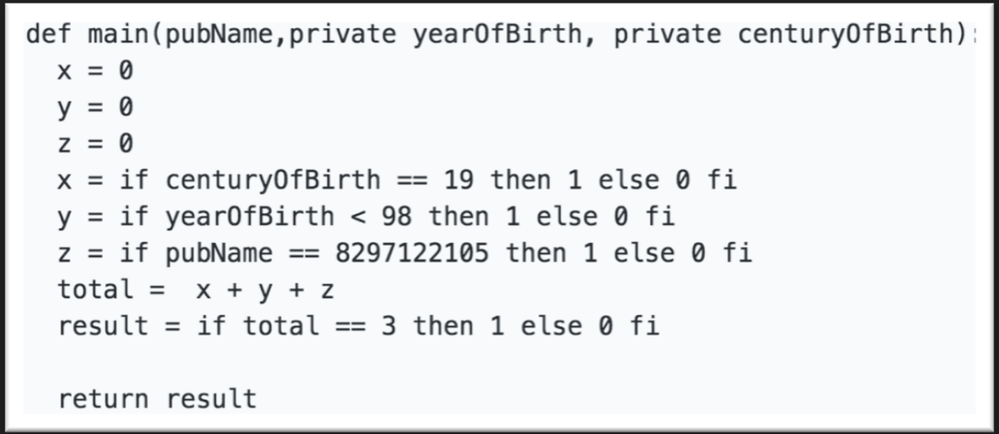
Succinct → Proof is short and easy to verify

Non-interactive → Proof does not require back-and-forth interaction between the prover and the verifier

Argument of knowledge →Proof attests not just that the statement is true, but also that the prover knows why its true

zk-SNARKs – How it works

1. program/circuit has public input (x) and private input (witness or w)



```
def main(pubName,private yearOfBirth, private centuryOfBirth):
  x = 0
  y = 0
  z = 0
  x = if centuryOfBirth == 19 then 1 else 0 fi
  y = if yearOfBirth < 98 then 1 else 0 fi
  z = if pubName == 8297122105 then 1 else 0 fi
  total =  x + y + z
  result = if total == 3 then 1 else 0 fi

  return result
```

Program written to work with ZoKrates

2. key generator (lambda , program/circuit) → proving key (pk) , verification key (vk)

3. prover (pk , x , w) → proof

4. verifier(vk , x , proof) → { true | false }

zk-SNARKs - Demo

Working with zk-SNARKs in Ethereum

zk-SNARKs - Lab

Working with zk-SNARKs in Ethereum

https://github.com/razi-rais/blockchain-workshop/tree/master/zk-SNARKs