Secure Computations in Decentralized Environments

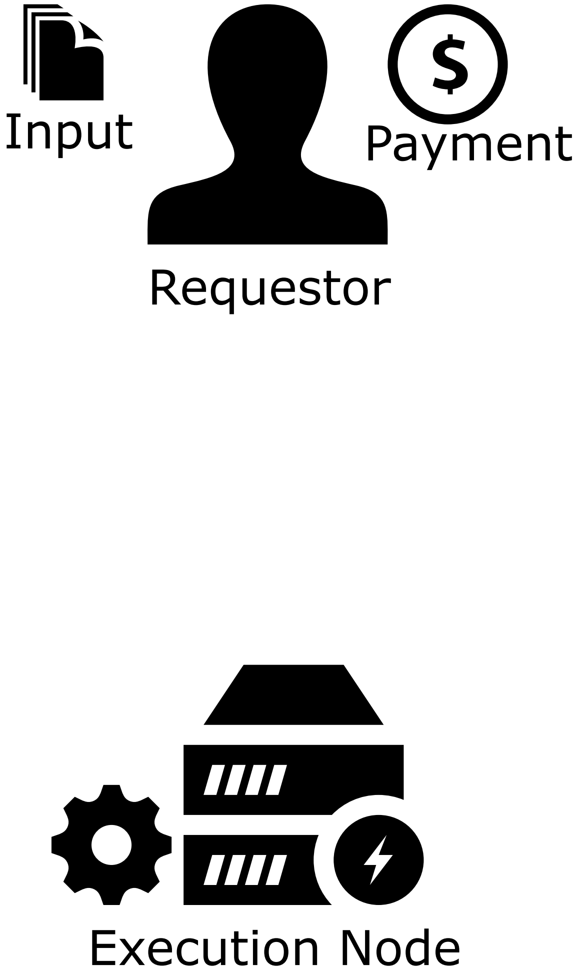
Michał Król **<m.krol@ucl.ac.uk>**, Alberto Sonnino **<a.sonnino@cs.ucl.ac.uk>**,
Mustafa Al Bassam **<m.albassam@cs.ucl.ac.uk>**, Ioannis Psaras **<i.psaras@ucl.ac.uk>**

University College London

<online version with animations>

Introduction

Scenario



Input

Payment

Requestor

Execution Node

Edge computing

To make edge computing a realistic alternative:

- Security and privacy must be built in the system design
- It must be easy to join the network to submit/execute tasks
- Nodes need to be rewarded for their work
- Fully decentralized without "trusted" 3rd parties
- Bulding Blocks
    - Rewards
    - Result Verification
    - Privacy

Rewards

- Nodes need to be rewarded for used resources
- It can be the main motivation for nodes to join
- Work need to be proved/verified before payment
- But when should be the payments done?

Result Verification

- Different types of tasks
- Cryptographic proof
    - High cost
    - Not available for every computation
- Parallel execution
    - Partial or complete
    - Highly innefficient
    - How to prevent colluding?

Input/Result Privacy

Input and Result Data must be hidden from the network **and** from the execution node

- Homomorphic encryption
  - Introduces overhead
  - Not always possible
- Trusted Execution Environment
  - Creates a trusted environment within an untrusted node
  - Low overhead
  - Requires dedicated hardware

Industry

- Golem, Somn
- Run on Ethereum Blockchain
- Payments using smart contracts
- No automatic, reliable result verification mechanism
- 3rd parties to resolve conflicts

Background

Intel SGX

- Trusted Execution Environment (TEE)
- Enclaves are protected by the CPU against access from other apps/OS/Hypervisor
- Used in Proof of Elapsed Time (PoET)
- Remote Attestation Protocol
    - Verifies the hardware
    - Verifies the code running on a remote node
    - Allows secure communication with the enclave

Blockchain Technologies

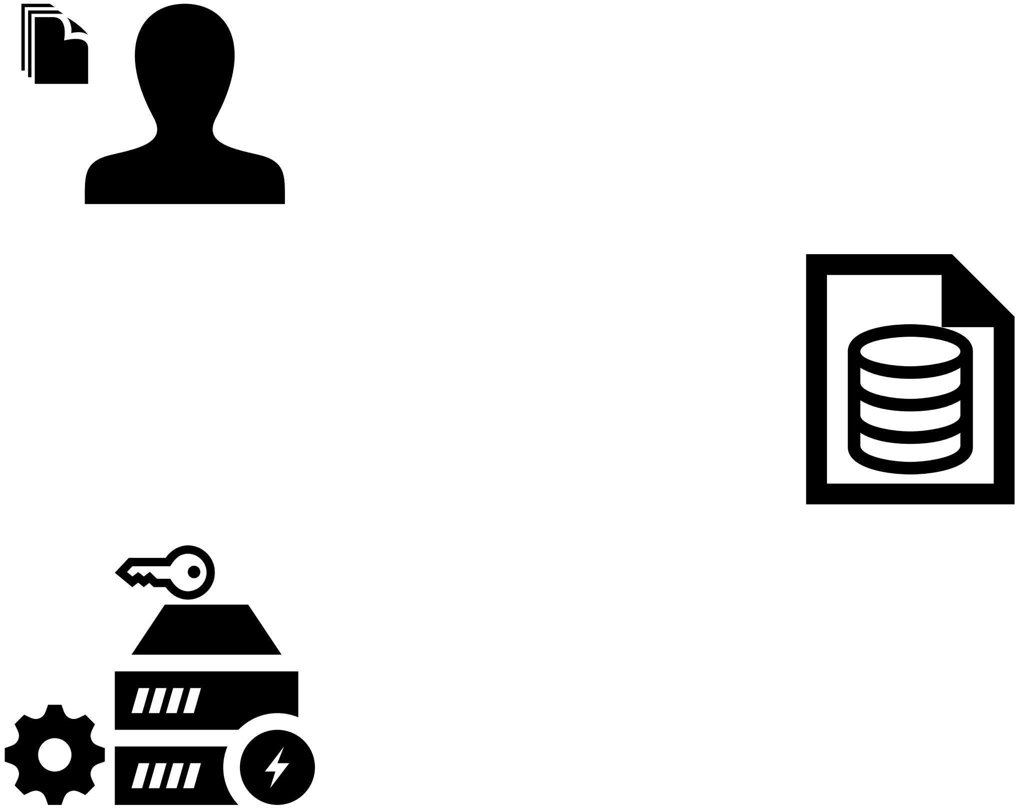
- Smart Contracts
    - Allow to logic on top of a blockchain
    - Turing complete language (Solidity)
    - Submitted data is publicly visible
- Payment Channels
    - Process off chain payments
    - Secured by deposits
- Oracles
    - Trusted data feed for Smart Contracts

Overview

Assumptions

- Both the Requestor and the Executing Node know the function and trust its behaviour (i.e. a function does images processing)
- The Requestor and the Executing Node mutually distrust one another
- Both the Requestor and the Executing Node trust the blockchain
- The Executing Node has complete control over its OS/Hypervisor

AirTnT



Conclusion

- System for result verification and payments
- Fully automated
- Orders of magnitude lower overhead than SoA
- No 3rd parties involved
- Limitations
  - Needs Intel hardware
  - Application size limit (up to 100MB)

Open Questions

- How to automatically dispatch tasks?
  - Concerning cost, node capacities price
  - How to define fairness?
- How to estimate cost of computations?
  - CPU, bandwidth, memory, QoS
  - Different node capacities
- How to provide full privacy?

Thank you