Decentralized Metering and Billing of energy on Ethereum with respect to scalability and security

Aristotle University of Thessaloniki

Honda R&D Europe

Georgios Konstantopoulos

Contents

1 Abstract

Our design includes

2 Introduction

2.1 Motivation

Blockchain technology blabla

2.2 Problem Statement

How an entity can manage the energy consumed by a complex system of energy meters. The system should be able to bill and perform accounting on the metering data, based on an accounting model The system must be transparent, distributed, decentralized, easy-to-use and secure. Anyone in the network should be able to verify the validity transactions. It also needs to be scalable at reasonable cost.

2.3 Scope

The thesis is limited on blabla, scalability is explored blabla,

2.4 Outline

Chapter X describes Y

3 Blockchain Basics

3.1 Blockchain Fundamentals

3.1.1 Cryptographic Hash Functions

3.1.2 Transaction

A transaction is a bunch of data

3.1.3 Block

A block is a group of Transactions

3.1.4 Blockchain

A blockchain is a series of blocks linked to its previous one by referencing a hash pointer

3.2 Use Cases

3.2.1 Blockchain Types

Public Blockchain

Private Blockchain

3.3 Ethereum

3.3.1 EVM state machine

3.3.2 Transactions as State Transitions

3.3.3 Gas

3.3.4 Programming in Ethereum

Solidity

Contract oriented, explain ABI, bytecode

Tooling

Truffle, geth, docker, parity blabla

4 Blockchain Scalability

4.1 Bottlenecks in Scalability

Explain proof of work and how it limits transactions per second. A variety of Scalability Solutions have been thought of such as Plasma, blabla, Ethermint? We argue that there are two levels of scalability. Scalability on Contract and on Network Level. While the solutions described above are aiming at solving the Network level scalability, it is argued that scalability cna be also be improved by proper contract design, i.e better algorithm

4.2 Network Level Scalability

Long term solutions include. Due to the use case we will focus on shorter term solutions are shardign and casper pos Shorter term solutions include payment channels, sidechains and alternative consensus algorithms. Due to the corporate setup, we choose to use proof of authority networks and further analyze them along with a minimal implementation of payment channels

4.3 Contract Level Scalability

4.3.1 Gas Costs

4.3.2 Optimizing for gas

4.3.3 Encoding

5 Smart Contract Security

Smart contracts immutable, lack of tooling, and developer mistakes

5.1 Past Vulnerabilities

Past vulnerabilities had lots of money locked, business logic of application broken. Brief summary of some vulns. Refer to relevant literature for more in depth.

5.1.1 TheDAO

5.1.2 KotE

Online platforms for training and exploiting have been developed.

5.2 Security of deployed Smart Contracts

Talk about the literature which has already mapped through the deployed contracts. Which tools did they use? Oyente, Mythril etc

5.3 Best Practices

Contract oriented, keep it simple, no tx.origin, ktl as described in literature

5.4 Access Control

There is NO private data, just functions that can be called by certain individuals A proper access control model needs to be implemented so that only authorized users can access certain functions.

6 Blockchain and the Energy Market

Price of energy, consumer does not know what they pey, or what they gain from their renewables

Projects in energy?

6.1 Our Use-case

Transparency, full history of meter readings, price calculation, billing of inhouse energy departments. This can be extended for EV car payment microtransactions and so on.

7 Design and Implementation

7.1 Business Logic

Explain company structure

7.2 Smart Contracts

Explain the Smart Contracts suite

```
pragma solidity ^0.4.16;

contract TestContract {

  string private myString = "foo";
  uint private lastUpdated = now;

  function getString() view external returns (string, uint) {
    return (myString, lastUpdated);
  }

  function setString (string _string) public {
    myString = _string;
    lastUpdated = block.timestamp;
  }
}
```

7.3 Monitoring Server

Explain monitoring server

7.3.1 REST API

Explain rest api usage

7.3.2 Python Client

Explain python implementation of rest api

7.3.3 web3.py interaction

Explain how web3.py interacts with monitoring server and sends data to Smart Contracts

8 Conclusion

8.1 Results

8.2 Future Work

Final remarks include.