Project Plan

Final Year Project 2017-2018, Department of Computer Science, The University of Hong Kong

Topic: Blockchain research and implementation (fyp17064)

Name: NG King Pui

UID: 3035178820

Date of Submission: 10/2017

Table of Content

1. Background

Blockchain is the backbone of Bitcoin, which is now gaining more and more attention in the world. The principle behind blockchain is to chain up blocks of data and let everyone have a copy of the chain, a.k.a. distributed ledger system. The blocks of data are chained up with hash, which is mathematically-proven secure and used in many other security measures. Every block will have its own unique hash value, which is generated based on its own data and the hash value of the last block. That is the reason why it is called blockchain, from chaining up blocks of data with hash. Therefore, if someone try to manipulate one block of data, e.g. change his own account balance, to fake others, his own copy of the chain will be different with others in terms of has value. Others can then tell his cop is not valid hence revoking that block. With the power of the crowd, it is difficult to create fake record in blockchain. Also as everyone has a copy, every transaction record is transparent to everyone. It then prevents under-the-table deals. With these 2 features, blockchain breaks the gap of trust. It can then be utilized in many areas, especially monetary transaction.

Smart contract is one of the typical examples of application of blockchain. Built on top of blockchain, smart contracts share the feature of security and transparency. Besides, smart contracts allow users to run their own script to make the contracts self-enforcing and self-executing. With the scripts, the application of smart contracts can be much wider across different industries. Apart from that, smart contracts reduce the cost and increase the efficiency significantly when compared with traditional contracts. When creating a traditional contract, legal consultancy is usually required to fit the interests of both sides and the regulations. This process will induce huge cost, in terms of both money and time. But with smart contracts, the consultancy cost can be eliminated. Additionally, based on the self-executing feature of smart contracts, the contract can become effective once the condition set is met.

For example, there is a contract between A and B, which is about A has to pay B $1 million to buy a house. With the traditional approach, A and B have to seek for relevant legal services in order to make the transaction. This induces a huge cost to both A and B. However, with smart contract, the script can check whether A has enough balance to pay B; and B really has the ownership of the house. Once both conditions above are met, the contract will

execute automatically. Also this contract is appended in a blockchain, which can be served as a proof of the fact that A owns the house after this transaction.

With such features, smart contract is expected to be more commonly used in the future. However, smart contract still has some flaws that may hinder its application. In this project, the focus will be on its security loopholes as security is the first concern of every digital transaction.

2. Objectives

The objectives of this projects are:

- Studying the basic working principle of blockchain and smart contracts
- Conducting research on the existing security loopholes of smart contracts
- Knowing how to detect security loopholes of smart contracts
- Building a web-based checking tool which can detect security loopholes of smart contracts

3. Methodology

In this section, the platform chosen Ethereum and the language of smart contracts Solidity are discussed. Moreover, the framework Geth and Web3.js, which are the major modules of developing the checking tool, are discussed and their roles in the tool are explained.

3.1 Ethereum

Ethereum is the chosen platform of development in this project. Ethereum is open-source blockchain platform. It is the most prominent platform for smart contracts (Buterin, 2013). It allows users to create their own scripts written in Solidity, a contract-oriented programming language for writing smart contracts. It is designed to create smart contracts on Ethereum. However, Solidity is also considered as one of the reasons why the implementation of smart contracts particularly prone to errors in Ethereum (Atezi, Bartoletti, Cimoli, 2017). In order to investigate these errors caused by Solidity, this is chosen as the language to use.

Ethereum is a blockchain platform. To create incentive for others to compute the result together, a token Ether is created to pay for those who have computed. For example, to validate a block of data about A buying a house from B, the transaction need to be validated by the crowd. To pay for the computing power, both A and B have to pay in Ether, or namely gas, for their effort. Currently one Ether token equals around US$300.

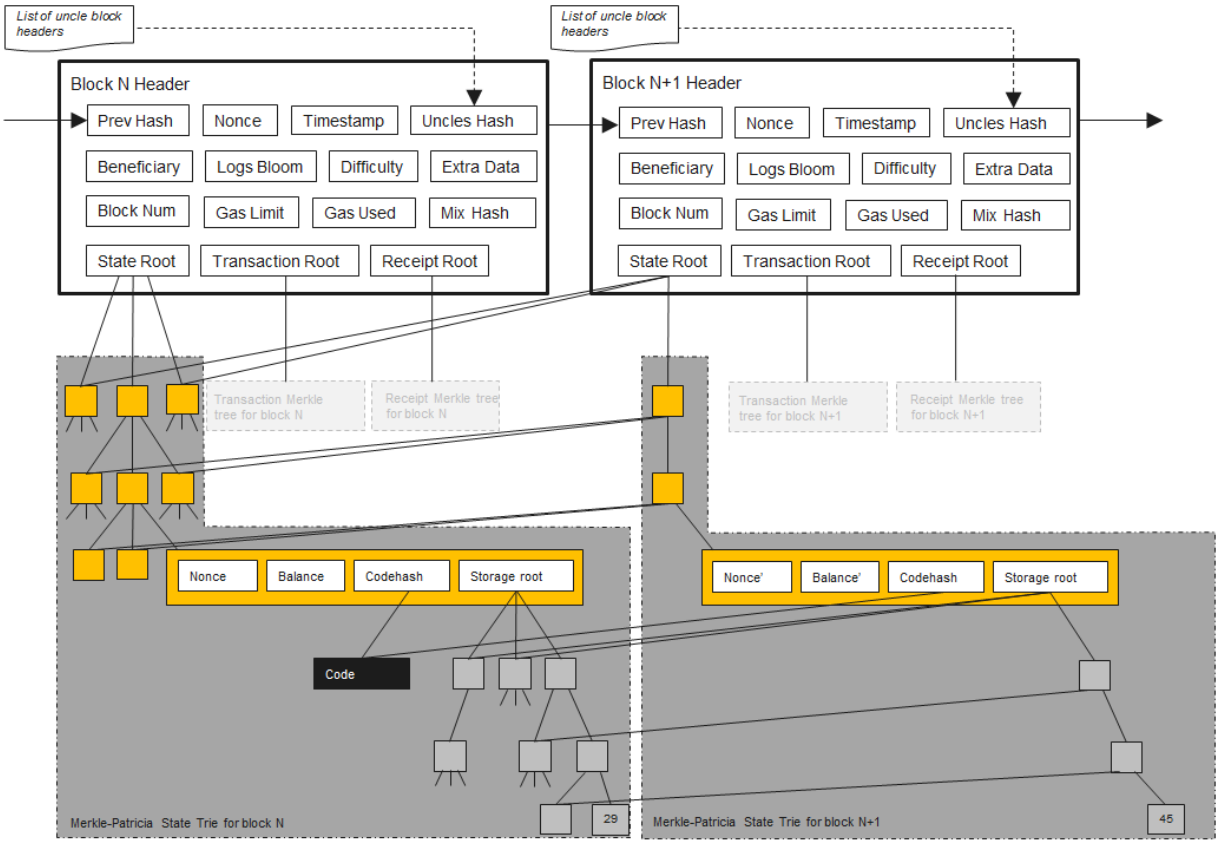

Figure 1: Architecture of Ethereum

Figure 1 shows the architecture of Ethereum. The header consists of many data which help identifying blocks. The PrevHash will help when chasing back the chain of data. This part makes Ethereum adopt the structure of blockchain, which make them share the same features mentioned in Background section. The state is the data stored in the block. These data can be balance or other data as shown in Fig.1.

3.2 Solidity

Solidity is a programming language designed for writing smart contracts. Fig. 2 shows how does a simple wallet object is created with Solidity. Owner contains the unique address of the owner of the wallet. It initializes the owner as who creates this object. Also the Pay function takes in the amount to send and the address of recipient. Before the owner actually pays, the function will first check does the owner call this function and if the owner has enough balance to pay that amount. If these conditions are all met, the function will deduct the amount from the owner and call recipient.send(amount) to increase the balance of recipient. It is only a simple example. More functions can be added if required.

```
contract AWallet{
    address owner;
    mapping (address => uint) public outflow;

    function AWallet(){ owner = msg.sender; }

    function pay(uint amount, address recipient) returns (bool){
        if (msg.sender != owner ||  msg.value != 0) throw;
        if (amount > this.balance) return false;
        outflow[recipient] += amount;
        if (!recipient.send(amount)) throw;
        return true;
    }
}
```

Figure 2: Simple demo code of a wallet in Solidity (Atezi, Bartoletti, Cimoli, 2017)

3.3 Geth

Besides researching on the potential attacks on Ethereum smart contracts, another key deliverable of this project is a tool checking smart contracts whether the identified security loopholes exist in those contracts. In order to check those contracts, a private blockchain is needed. With a private blockchain, a isolated testing environment is created so that no real monetary transaction is involved. Additionally, all the factors in a private testing blockchain are under my control. In a public blockchain, due to the nature of decentralized system, it is impossible to control the behavior of every single user. Therefore, some extreme cases cannot be tested. On the other hand, virtual users can be created to simulate different scenario, including extreme cases. Due to these two reasons, a private blockchain is a desired testing environment of Ethereum smart contracts.

In order to create a private blockchain to simulate attacks on Ethereum smart contracts, Geth, or Go-Ethereum, is used to create such a isolated environment. Geth is the official Ethereum implementation written in a programming language Go. It is widely used to communicate with the Ethereum network. Its functionalities include managing different accounts, mining cryptocurrency Ether and executing smart contracts. In this project, the functionalities to be used are mainly managing different dummy accounts and executing smart contracts under testing.

In order to create a private chain without communicating with the main public Ethereum chain, all the nodes in the private chain should not be connected to the public chain and be discovered by other users. A network ID different from the main network is needed to make sure that the nodes out of our testing environment are not connected to our private chain. The network ID of the main chain is 1. Therefore, if the network ID of the private chain is not 1, the nodes in our private chain cannot connect to the main chain where many real monetary transactions are made.

Moreover, the maximum number of peer can be set to zero to further confirm that no other peer is connected to the private network. The corresponding command is:

```
--maxpeers 0
```

0 here is the parameter indicating the desired maximum number of peers. As we look for an isolated testing environment, 0 is the desired number of peers.

3.4 Web3.js

With Geth, an isolated testing environment is created. The next step is to deploy the smart contracts. Web3.js is the library to use. Web3.js is JavaScript API (application programming interface) which can compile smart contracts written Solidity and execute them in the private chain.

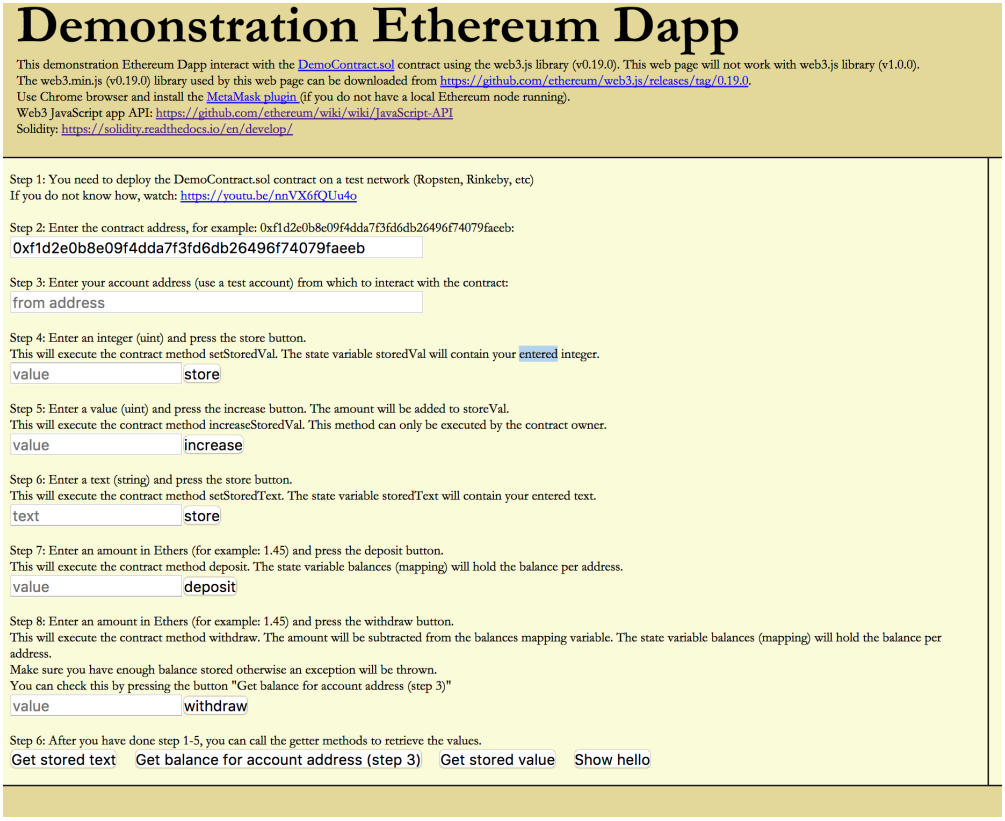

Figure 3: A screenshot of web-based application in Web3.js by mobilefish.com

Additionally, as it is a module in Node.js, another framework building web application, this tool can be launched as a web application. It makes the tool more user-friendly as no command input is needed to use tool. The entry barrier of using this tool is lowered by a simple web interface. The above figure is a screenshot of a web-application of Ethereum. It is easier to use compared with traditional command line tool.

To compile a Solidity smart contract, the command to use is:

```
web3.eth.compile.solidity(sourceString [, callback])
```

sourceString is the contract itself in string form. The callback is the function called after the contract is successfully compiled. After that an instance of contract object is created to execute the contract. The contract can then be executed and be tested under different potential attacks.

3.5 Summary of checking tool

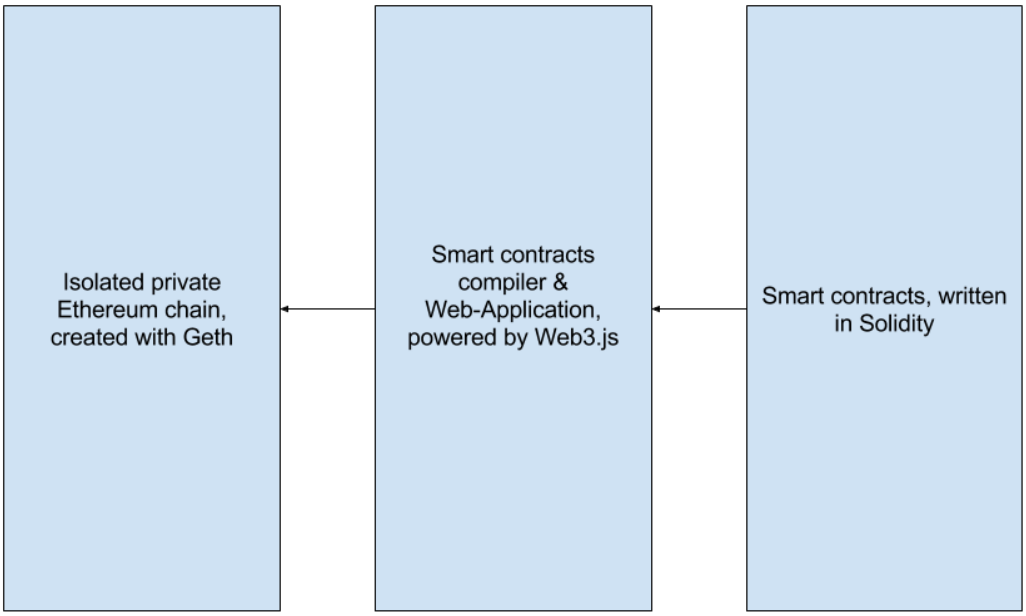

Figure 4: Architecture and logic flow of the checking tool

To conclude, the architecture of the checking tool is shown as above. The smart contracts in Solidity are the input of the system. They will be compiled with Web3.js and be executed on an isolated private chain created with Geth. The identified security loopholes are then tested under that environment. The tool is launched as a web-based application powered by Web3.js and Node.js to make the tool more user-friendly.

4. Upcoming Milestones

In this project, there are two main phases: research phase of the topic, i.e. blockchain and smart contracts, and development phase of the checking tool.

The research phase will mainly focus on the working principle of blockchain and smart contracts, and the existing security loopholes of smart contracts. A research paper "A Survey Attacks on Ethereum Smart Contracts" will be one of the papers or articles that will be studied. Also the articles from other blockchain developers will be taken into consideration. It is expected to be done by November 2017.

The development phase is about building the web-based checking tool, one of the final deliverables. It consists of the design of user interface and the checking mechanism of different attacks and security loopholes. It is expected to be done by January 2018.

After the checking mechanism is implemented, different sample contracts will be put under testing for debugging the checking tool. The rest of second semester will be spent on this phase.

The schedule of expected upcoming milestones is:

Time	Project Status
Oct 2017 – Nov 2017	Research on smart contracts and its security loopholes
Dec 2017 – Jan 2018	Designing the UI of the tool and checking mechanism
Feb – Mar 2018	Developing the tool and different attacks scenarios
Mar - Apr 2018	Testing and fixing bugs of the tool, Writing the final report

5. Conclusion

With the rise of blockchain and the convenience brought by smart contracts, the coverage of smart contracts is expected to grow rapidly. That is the reason why security loopholes of smart contracts are concerned. In this project, the loopholes are studied and a tool checking the vulnerability of smart contracts from those security holes will be developed as the final deliverable.