Indian Institute of Technology, Kanpur
Undergraduate Project Report

# Distributed Academic Grading System Using Smart Contracts

Author: Shikhar Mahajan (150669)
Supervisor: Prof. Arnab Bhattarcharya

13 November 2017

## Abstract

*The prominent feature of blockchain development is commonly understood as the decentralization of financial systems. The main reason behind this is the mechanism of incentivising the block miners for preserving the trust and robustness in public blockchains at the cost of their computation power and storage. Private blockchains, on the other hand are not open to public mining estranging the direct monetary influence. This opens up the scope of this technology to all sectors where data mutability is an existing issue. We will discuss the genuine use-cases of this technology in detail and deploy a smart contract built on top of Ethereum to encounter a prevalent problem of centralization of academic grading systems. We will address the issues relating to data mutability, integrity and non-repudiability in existing systems and the ways in which it is tackled in the proposed blockchain based solution.*

## 1 Introduction

Considering the familiarization of the blockchain technology and smart contracts with the masses over the past couple of years, the world has widely misunderstood their absolute use cases. On one side, blockchains are rapidly overhauled by an increasing number of institutions building decentralized applications by coming up their own proofs of concept. Contrarily, on the other side some orthodox technicians are referring this technology as just the synchronization of databases limiting its practical applied functionality. Many open source and proprietary platforms have come up to adapt this technology with people such as Hyperledger, Multichain and Ethereum. Differences in exploiting these platforms are based on the underlying proofs of correctness for distributed consensus and the particular purpose computation. Smart contracts essentially express the business logic as computer programs and removes the ambiguity in interpretation of a legal contract. As stated earlier, we will discuss these concepts in detail and frame a decentralized solution to an existing problem of the centralization of academic grading systems that involves an intermediary between the professor and the student in maintaining and bearing the grade sheets. We choose Ethereum as a platform to build our decentralized application as it offers the smart contract scripting functionality and a general purpose computation scheme unlike others. Our smart contract is written in Solidity language and can be deployed in Remix [1], a web browser based IDE that provides basic UI and JS VMs to test the contract.

## 2 Blockchain under the hood

At the core, blockchains offer just another mechanism for the synchronization of distributed databases, but is not limited to it.

To maintain a database, two entries should not overlap with each other. If the database is distributed rather than centralized, it demands to a higher degree of compatibility leading to Multiversion Concurrency Control (MVCC) where each transaction basically interprets a steady snapshot of the data at a particular time. Hence in MVCC systems, there isn't any foundation of modifying an entry of the database. Rather, it essentially requires back-to-back extended operations of deleting and creating the entry. This is similar to the working of a distributed version control system of our daily use, Git.

The problems arising of updating the distributed databases in consensus has led to some variants like Master-Slave replication but they substantially demand very less write operations as compared to the read ones. To tackle this problem efficiently, Blockchains come in place offering a distributed MVCC in a multi-master replication system. This is implemented by allowing transactions in public space where each transaction block is chained in a time-stamped manner recording each modification to the database. A block is defined as a collection of hashed transactions floated in the network at a particular instant of time. The blocks are chained by storing the hash of the previously linked block address in the freshly created one. The identity hooked to these transactions follows the concept of asymmetric keys. The public address of the receiver of a transaction is embedded inside the output script which only allows the corresponding private key bearer to spend the asset transferred by the transaction.

## 3 Potential Use-cases [2]

Although blockchain technology is under development in nearly all fields today, there are a few provisions under which a project should fall to avoid an impotent blockchain application. These are listed as follows:

1. **Distributed database:** The project must involve a shared and distributed database. An entry corresponding to the database refers to the data/asset recorded in the name of the entity owning that entry. Any modifications to these entries are only entertained through transactions which are published globally (in the case of public blockchains) for acceptance by mining nodes.

2. **Multiple writers:** As transactions can be performed by any permissioned node (everyone in the case of public blockchains), it involves multiple entities transacting modifications to the database.

3. **Mistrust among the writers:** The concept of consensus is relevant only when there are multiple writers present with conflicting goals/records. This leads to the structure of forks in the blockchain, resulting into global acceptance of the longest chain in the fork.

4. **Interactive transactions:** Transactions essentially involves deleting and creating entries in a collective bilateral fashion. This accounts to modifying existing entries in the database linking each transaction that involves a particular entry.

5. **Disintermediation:** This is a fundamental fracture point of many projects aiming to exploit blockchains. It is often the case that the project doesnt actually need the removal of a trusted central party. Disintermediation leads to delays in acceptance of transactions and involves the cost of applied computation power and storage of all the verifiers involved.

6. **Permissioned miners:** The provision of permissions in blockchains opens up its scope to many fields. This is a primary pointer to the development of private blockchains where an authority in control of the blockchain permits some particular nodes to mine blocks. This margins the protocols adopted by private banks with a public protocol like Bitcoin.

If all of the above conditions are not met, then there is a good chance that implementing the existing relational databases model rather than a blockchain model will work fine to achieve a viable business miniature.

It should also be noted that in public blockchains, although a direct personal identity is not linked with the public address, an adversary can track transactions involving a particular public address as all transactions blocks are broadcasted to the network and are accessible by every node. This leads to a trade-off of decentralization achieved at the price of confidentiality.

## 4 DAGS(Distributed Academic Grading System)

DAGS stands for Distributed Academic Grading System. Through this system, we aim to encounter the following challenges to keep the application consistent with the existing constraints:

1. Each course is identified by an unique course ID like CS395.

2. The instructor of the course and maximum number of students that can get enrolled in the course is set by the chairperson primarily.

3. The instructor is allowed to add students in the course via their public addresses. These public addresses of students are also needed in order to obtain a their past grade sheet.

4. A student cannot fake his/her identity to get enrolled in the course twice.

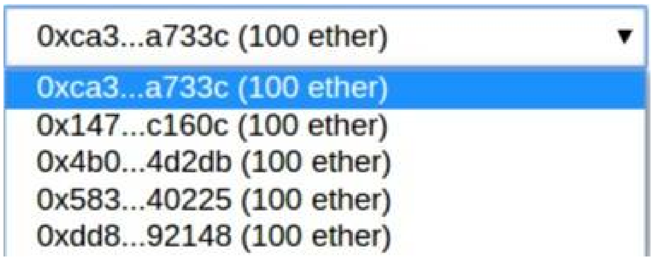
5. A grade can only be assigned by the professor and once assigned, cannot be changed by him/her at any later point of time. In case of any discrepancies, the chairperson is the only authority who can alter the grade.

6. Each student has the permission to view his/her own grade but not someone elses.
7. Everyone in the network can do statistics on the course grading scheme such as percentage of students getting an A grade, failing the course etc.
8. Professor is able to justify the grading scheme if questioned later(Off-chain data).

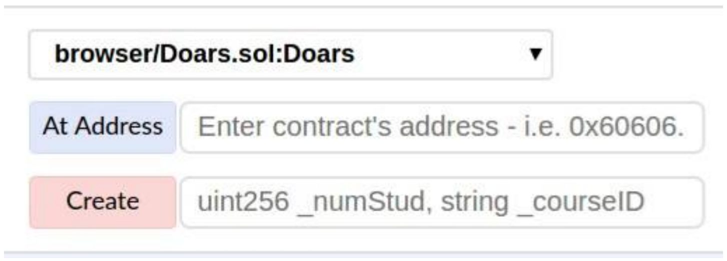
We choose Ethereum as a platform for our decentralized application demonstration as it offers the smart contract scripting functionality and a general purpose computation scheme unlike others. The smart contract used for this demonstration is written in Solidity language and is deployed in Remix, a web browser based IDE that provides basic UI and JS VMs to test the contract.

## 4.1 Deployment

Remix gives us 5 accounts with 100 ether each that can be accessed by the smart contract for testing purposes. We distribute these 5 addresses into chairperson, instructor and 3 students getting enrolled in the course.

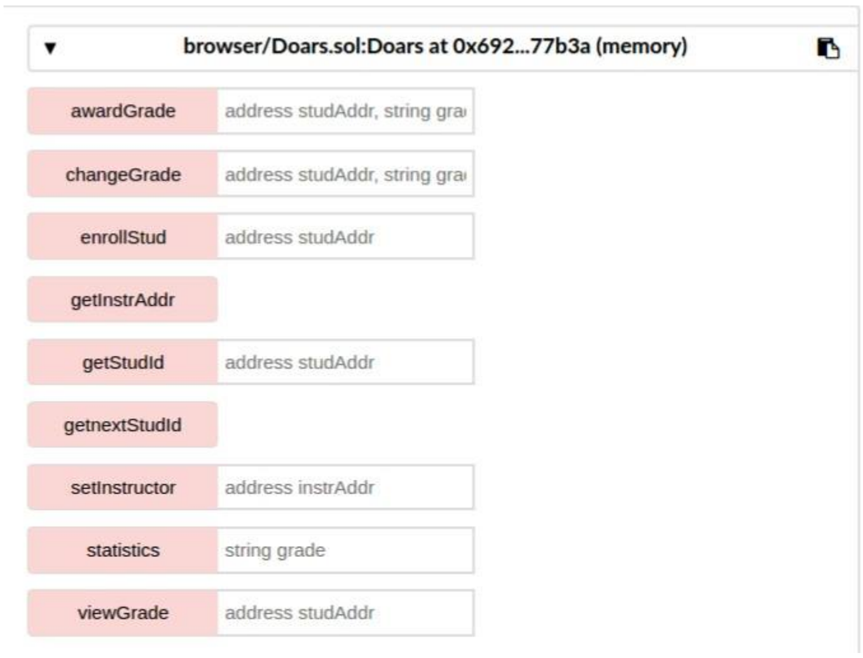

0xca3...a733c (100 ether)
0xca3...a733c (100 ether)
0x147...c160c (100 ether)
0x4b0...4d2db (100 ether)
0x583...40225 (100 ether)
0xdd8...92148 (100 ether)

The chairperson creates the smart contract by specifying the course ID and the maximum number allowable students in the course. The chairperson can also bind the contract to a special account address if needed. By default if this field is left empty, it is set arbitrarily to any available contact account address.


browser/Doars.sol:Doars
At Address
Enter contract's address - i.e. 0x60606.
Create
uint256 _numStud, string _courseID

## 4.2 Functions

Following is the image of the UI provided by the Remix IDE to perform the contract's functions:



The description of each function is as follows:

1. **setInstructor:**
   (a) Takes as input the public address corresponding to the instructor
   (b) Requires the transaction sender to be the chairperson
2. **getInstrAddr:** Outputs the course's instructor's public address
3. **enrollStud:**
   (a) Takes as input the public address corresponding to a student
   (b) Each student enrolled in the course is assigned an unique number ID starting from 0 in the order of enrolling students
   (c) Requires the transaction sender to be the instructor
4. **getStudId:**
   (a) Takes as input the public address corresponding to a student
   (b) Outputs the student's ID
5. **getnextStudId:**
   (a) Outputs the number which will be assigned as the ID of next enrolling student of the course
   (b) Also equals the total number of students enrolled in the course so far
6. **awardGrade:**
   (a) Takes as input the public address corresponding to a student and the grade string assigned to him/her

    (b) Requires the transaction sender to be the instructor

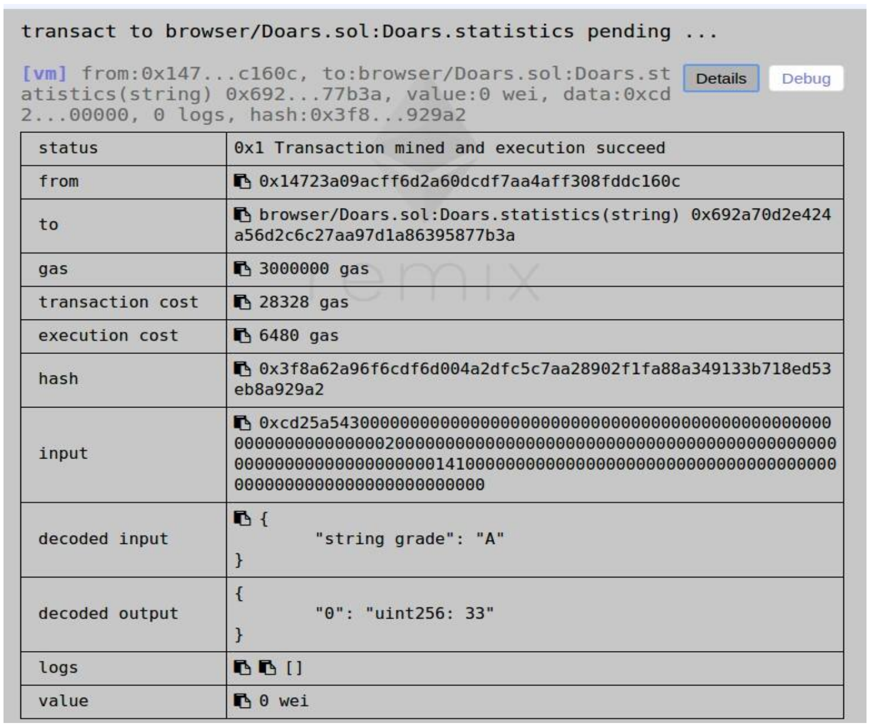
7. **viewGrade:**
    (a) Takes as input the public address corresponding to a student
    (b) Requires the transaction sender to be the instructor or the student himself/herself

8. **changeGrade:**
    (a) Takes as input the public address corresponding to a student and the changed grade string assigned to him/her
    (b) Requires the transaction sender to be the chairperson

9. **statistics:**
    (a) Takes as input a grade string
    (b) Outputs the percentage of students in the course getting that grade

## 4.3 Transactions

We will look at how a transaction is interpreted by the smart contract upon execution of a function. Take the function *statistics* for instance. Here is an image showing the detailed description the transaction created by this function when given the input string as "A":



```
transact to browser/Doars.sol:Doars.statistics pending ...

[vm] from:0x147...c160c, to:browser/Doars.sol:Doars.st
atistics(string) 0x692...77b3a, value:0 wei, data:0xcd
2...00000, 0 logs, hash:0x3f8...929a2
```

| | |
|---|---|
| status | 0x1 Transaction mined and execution succeed |
| from | 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c |
| to | browser/Doars.sol:Doars.statistics(string) 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a |
| gas | 3000000 gas |
| transaction cost | 28328 gas |
| execution cost | 6480 gas |
| hash | 0x3f8a62a96f6cdf6d004a2dfc5c7aa28902f1fa88a349133b718ed53eb8a929a2 |
| input | 0xcd25a54300000000000000000000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000000000001410000000000000000000000000000000000000000000000000000000000000000 |
| decoded input | { "string grade": "A" } |
| decoded output | { "0": "uint256: 33" } |
| logs | [] |
| value | 0 wei |

- *status* shows the current status of the transaction in the blockchain
- *from* shows the transaction sender's public address

- *to* shows the contract's account address in the blockchain
- *gas* shows the upper gas limit spent on the transaction specified by the sender
- *transaction cost* shows the gas utilized in mining the block associated with the transaction and appending it to the blockchain
- *execution cost* shows the gas utilized by the contract in computing the result and returning it as the output
- *input* shows the hashed input script
- *decoded input* shows the argument name, type and its value given to function, "A" in this case
- *decoded output* shows the return result index, type and value of the function, 33 in this case.

## 4.4 Future Work

1. We intend to scale the current distributed model to accommodate many courses instead of one in a single blockchain.
2. Modifying the current model such that a student gets an unique ID among all students which is consistent in all courses that he/she gets enrolled in.
3. Improving the UI of the application by using web development tools like CSS and JS to make the front-end more interactive.
4. Introduce off-chain data to include hashes of the soft-copies of all the answer scripts.
5. Migrate the application completely to the Truffle framework from browser based Remix IDE to make the system adaptability easier.

## References

[1] Wikipedia. Remix-Wikipedia. `https://en.wikipedia.org/wiki/Remix`.

[2] Gideon Greenspan. Avoiding the pointless blockchain project. `https://www.multichain.com/blog/2015/11/avoiding-pointless-blockchain-project/`, 2015.