Developing a universal, decentralized and immutable Erasmus credit transfer system on blockchain

Domantas Pelaitis  
Department of Software Engineering, Vilnius University, Lithuania  
Email: domantas.pelaitis@mif.stud.vu.lt

Georgios Spathoulas  
Department of Computer Science and Biomedical Informatics, University of Thessaly, Greece  
Email: gspathoulas@dib.uth.gr

Abstract—Blockchain is one of the most hyped technologies of recent years. It promises to restructure the way applications are developed and to replace the widely client-server model used today. While the technology is promising the ecosystem of tools and development methodologies are still in their early stages. In this paper a proof of concept blockchain application is developed, regarding the exchange of data between different universities in different countries, with respect to the Erasmus program. A decentralized application has been designed from ground up and is proposed to be installed in multiple nodes located in all participating universities. Ethereum platform has been chosen and the required smart contracts have been implemented. Additionally an easy to use web interface has been developed in order to be usable for end users. We have evaluated both the development process and the efficiency of the resulting implementation.

I. INTRODUCTION

Blockchain technology is getting more and more attention nowadays. It solves the consensus problem between multiple nodes, by combining various cryptographic techniques. The main innovation that attracts developers is that it eliminates the need for a trusted third party between the nodes. Research community is working on the employment of the technology and is researching for appropriate use cases to test it on. While using a blockchain instead of a regular database has a lot of advantages such as no single point of failure or data immutability, it comes with drawbacks in terms of scalability and cost.

The blockchain technology has been introduced through bitcoin, a successful distributed payments system that has disrupted the economic transactions domain. While blockchain is still a young technology it has matured a lot through recent years and many different approaches have been proposed regarding the mechanism that ensures the integrity between multiple equivalent nodes. Additionally the community have intensively worked on using the bitcoin design principles, in order to decentralize other services, apart from payment systems. This initiative has driven us today to discuss and test the development of applications, named dapps (distributed apps), on various blockchain infrastructures, in order to replace centralized services adhering to the client server model.

The Erasmus Program [1] (stands for EuRopean community Action Scheme for the Mobility of University Students) is a European Union (EU) student exchange program established in 1987. Students who join the Erasmus Program study at least three months in another European country. The period spent abroad is recognized by their university when they come back, as long as they abide by terms previously agreed. Students do not pay extra tuition fees to the university that they visit. It has become very popular among European students as it provides a great learning experience along with a chance to socialize with students from other countries. An important aspect is that students can use the courses (along with the corresponding credit units) they have taken while participating in the program, when they return to the University they originally study. This practically generates the requirement for an information exchange system between different Universities across Europe.

In this paper a proof of concept Erasmus student management application is built, by using blockchain for storing and managing data. The main functional requirement for such a system is the management and the exchange of personal student data between semi-trusted parties; the Universities participating in the program. No single party is completely responsible for the management of data, but each party shares the same amount of responsibility. Using blockchain to store and manage data solves this problem in a decentralized fashion and helps to avoid involving third parties with the related risks. Overall the proposed solution feels much more appropriate, than using a regular database hosted at the infrastructure of a specific party.

In order to test the usability of the proposed framework and asses the performance and the running costs, an initial implementation has been conducted. The code of the project described herein is open source and is available on Github.1

1https://github.com/DomantasP/ErasmusChain
Blockchain have been used in various domains. The most promising applications have been proposed for finance [2], [3], [4], supply chain management [5], [6], [7] and sharing economy [8], [9], [10].

Finance is the most natural domain for the application of blockchain. The first and most successful application of the technology is bitcoin, a distributed payments system. Blockchain solves important issues for financial or assets transactions, such as trust, integrity and availability. It enables participating nodes to exchange valuable tokens without the need for any specific node (e.g. a bank) to validate these transactions.

Another domain that has drawn the focus of blockchain community is supply chain management monitoring. Blockchain technology is able to monitor the flow of goods in supply chains. It enables all actors in such chains, such as producers of goods, brokers or end consumers to be sure about the identity of goods, their ingredients and past related transactions.

Sharing economy is a model where users that posses assets are able to share those with others in exchange for some sort of payment. Such services mainly regard houses or cars and are based on a centralized structures. The provider of the service charges users with significant fees and in exchange establishes trust between them. Blockchain technology can substitute the service provider, offer trust through cryptography and significantly reduce charges.

### III. BLOCKCHAIN TECHNOLOGY

#### A. Bitcoin

In 2008 Satoshi Nakamoto introduced bitcoin, a technology that would enable immutable transactions of digital assets from one party to another, without any intermediate party being responsible for this operation [11]. The underline mechanism of bitcoin, blockchain can fully solve problems related to the trustful interaction between multiple parties that do not trust each other by default.

In simple words, blockchain is a shared ledger of transactions. Committed transactions are periodically grouped into a block which is then hashed (mined) by one of the parties and appended to a chain of such blocks. If a malicious adversary wants to change any transaction that happened in the past he should change that transaction, re-mine the block in which that transaction was included and then re-mine all the blocks that appeared after this block. Because of the fact that mining procedure is very demanding in resources, this is not feasible.

Another way to describe blockchain is as a decentralized database, where every party has a copy of the same ledger. A blockchain network is a network of nodes, where each node runs software that adheres to a specific protocol which defines the rules of how those nodes communicate and how the data is verified and stored. Every node has its own copy of the blockchain. Every block in the blockchain has a block header and a list of transactions. A block header consists of previous block header hash, the root of a Merkle tree of the blocks transactions, the timestamp and a nonce. Previous block hash is a way to chain blocks and this is how every block references to a previous block.

A Merkle tree is a tree data structure, where every leaf node is labelled with the hash of a transaction and every non-leaf node is labelled with the hash of the labels of its child nodes. It enables the integrity check of the block by only checking the value of the root node of the tree.

A nonce is a pseudo-random number that is used in proof of work procedure. It is chosen randomly in order to vary the input to a hash function, until a hash result that fulfills specific conditions is obtained. Proof of work is a mechanism that obligates miners to commit significantly high processing power, along with the corresponding power consumption, before mining a block. The difficulty of this work is adjusted so as to limit the rate at which new blocks can be generated by the network.

Due to the very low probability of successful mining, it is hard to predict which node in the network will be able to generate the next block. For a block to be valid, its hash value must be less than the current target. Each block contains the hash of the preceding block, thus each block has a chain of blocks that together contain a large amount of work. Changing a block (which can only be done by making a new block containing the same predecessor) requires regenerating all successors and redoing the work they contain. This protects the blockchain from tampering.

#### B. Blockchain

Although the concept of blockchain was first introduced by Satoshi Nakamoto, nowadays there exist a lot of different blockchains, serving various purposes. To name a few: Ethereum that allows not only to transfer digital assets from one address to another but also to execute smart contracts [12], IOTA that is used for the security of internet of things data [13] or Monero that tries to keep transaction very secure and anonymous [14]. Some of these blockchains solve different problems, others solve the same problems in a different way, but they are all based on similar underlying technologies.

For example Ethereums main purpose is to be a decentralized application platform that allows to build and use real applications on top of blockchain. It enables such functionality by allowing anyone to deploy and use smart contracts on its blockchain. Smart contracts are account holding objects on the Ethereum blockchain. They contain code functions and can interact with other contracts, make decisions, store data, and send tokens to others. Contracts are defined by their creators, but their execution, and by extension the services they offer, is provided by the Ethereum network itself. They will exist and be executable as long as the whole network exists, and will only disappear if they were programmed to self destruct. Because the Ethereum Virtual Machine is Turing complete it means you can write contracts that theoretically can solve any reasonable computational problem.
As it happens in bitcoin a cryptocurrency (a virtual token) is used in order to enable the functioning of Ethereum platform. This is named Ether and is the next most important cryptocurrency after bitcoin regarding both popularity and market cap [15]. Specifically there are two kinds of accounts:

- **external accounts** that just hold a balance of Ether
- **smart contract accounts** that hold a balance of Ether but also contain the code of the contract

The main language used for developing smart contracts is Solidity. The usual structure of such contracts consists of some attributes and some methods. Nodes of the Ethereum blockchain can either deploy such contracts or interact with them by calling their methods. The main atomic action that is used in order to enable the deployment and use of smart contracts is the transaction. Each external account may deploy a contract by committing a transaction that contains the compiled code of the contract. After the deployment of a contract, which is then identified by a unique address, external accounts and smart contract accounts can interact with it by calling its methods.

This functionality that resembles traditional non-decentralized applications comes at a cost. All nodes of the system have to execute every single transaction and the account that sends the transaction pays a corresponding fee in Ether, which is calculated upon the complexity of the code to be executed.

### IV. Design of the Application

![Use case diagram](image.png)

Due to technology and design limitations it was very hard to build truly decentralized applications before the proposal of blockchain. Even when the application built was intended to be used by equivalent nodes, a central point was needed in order to implement the client-server model. The usual design and development approach was to either use one of the nodes as a central node or to even make up an additional central node by utilizing a trusted third party. This centralization has led to various issues concerning power given by controlling others’ data and responsibility for handling such data in a privacy aware fashion. There are use cases for which the problem to be solved does not by default set a node as central. All nodes are equal and no node has the required trust to act as central node. These use cases are suitable for testing blockchain technology’s applicability, as the decentralized nature of blockchain serves them in a better way and diminishes the requirement for setting any node as a powerful central node. The use case discussed herein is courses credit transfer for Erasmus program.

We identified the main actors and the work-flow of the specific use case as follows. A **student** takes part into the Erasmus program and studies at a foreign University for a semester. The University he comes from is identified as **originating institution** and the University he travels to is identified as **receiving institution**. The courses and the marks of the student have to be securely sent from the receiving to the originating institution.

The proposed decentralized application allows to map the usage requirements and the actors of the use case in a better way than a centralized solution would do so. In practice there is no central entity that is responsible for Erasmus students and the inter-institution registration of their courses. Originating institution usually has to communicate with the receiving institution or vice versa to exchange the required data. The proposed application tries to solve this problem and simplify the whole process by establishing trust between different parties.

As described in Section I the functional requirements of the use case are:

- The originating institution needs to add students and set the corresponding receiving institution.
- The originating institution needs to set the courses each student will take in the receiving institution.
- The receiving institution needs to register the marks for each course.

Figure 1 depicts the use case diagram and Figure 2 shows the robustness diagram of the implemented application. Originating institution maintains a list of its students and should add the receiving institution for each one taking part into Erasmus program. It can also add the courses that the student is going to study in the receiving institution along with all related information such as course title, course id and course credits. On the other hand receiving institution is responsible only for validating that the student passed each course and registering his mark.

Each University is registered in the decentralized application by associating an Ethereum address with its name. If every participating University advertises its Ethereum address then all other participants can verify the authenticity of the committed
transactions by checking the address of the account that sends
the transaction. Additionally this one to one mapping between
Universities and Ethereum addresses enables a simple access
control mechanism to be established in the smart contract
itself.

Any University can act both as an originating institution for
its own students or as a receiving institution for the Erasmus
students that are currently studying or going to study in it.
The initial implementation of this distributed application, apart
from the smart contracts, also includes a web front end with all
the functionality mentioned. Practically each University may
host an instance of the system consisting of a geth client
(Ethereum node) and the web front end to be used by its
employees. Through this a University can be signed up and
create an account using an Ethereum wallet. After registration
this account is identified by its public address and the relevant
University name. After registering an account, the University
can simply log in with the same wallet, so no username or
password is needed.

V. APPLICATION IMPLEMENTATION

All application features are offered by two smart contracts.
The first one Authentication.sol has as sole purpose to register
new users and let them subsequently log in. The other contract
Erasmus.sol is used to store and access data. The main data
entities are University, Student and Course. These are listed in
Listing 1, where the main data structure of the Erasmus.sol
contract is depicted. Each University has its own list of local
students, students that originally study there, and an Erasmus
students list, a list of students that are going to study in this
University for the Erasmus project. Every Student instance
holds some typical data such as id, first name, last name and
courses that he is going to have. Finally, each Course instance
has a course id, a title, credits, a mark and a flag to recognize
it as completed course or a course in progress.

There are lot of methods for setting and getting data from
the contract, along with some private helper methods. It should
be noted that most of the methods use Solidity require function
to implement some kind of validation. In this way every
account can only access and manipulate the data it licensed to
do so. For example, only the account of the University noted
as the Erasmus University of a student can set his courses as
finished or register marks for those. No other account in the
system can commit such transactions.

```solidity
pragma solidity ^0.4.11;

contract Authentication {
    struct User {
        address owner;
        bytes32 salt;
    }

    mapping(address => User) users;
}
```

Listing 1. Data entities in Authentication.sol smart contract

For the interaction of users with the application a web
front end has been implemented. The simplistic user interface
makes the service accessible instantly, without requiring the
installation of additional software. In the front-end React [16]
and Redux [17] javascript libraries have been employed to build a single page application. Redux actions communicate with the smart contract through an instance of Web3 - ethereum javascript API [18]. Data that is retrieved from smart contract goes to React-Redux containers which then display information to the user. All of it is built on top of react-redux truffle box. Truffle is a framework and is used for compiling, migrating and testing smart contracts [19].

VI. EXPERIMENTAL RESULTS

The experiments procedure used, was to fully implement the proposed application and deploy the smart contracts on TestNet, in order to test their functionality and the relevant costs. In Ethereum the main public network is named MainNet and real Ether fees are required to interact with it. For testing purposes there is an analogous network named TestNet that functions exactly in the same way. The only difference is that Ether on TestNet do not have real world value and can be easily obtained either by mining or through faucet services.

The deployment of every instance of the smart contract is a transaction the fees of which are related to the complexity of the code of the smart contract. Additionally every smart contract’s function call is a new transaction the fees of which are related to the code it includes. Every such transaction costs some gas, Ethereum internal pricing unit for running a transaction or contract. Gas is necessary to cover computational costs. The more difficult the computation is, the more gas it costs. When committing a transaction the node sets the gas price, which relates to the amount of Ether the node is willing to pay for each gas unit of the transaction. Miners collect these fees as a mining reward, so they choose which transactions to include into the next block according to their gas prices. The higher the gas price is the fastest the transaction will go through, as the probability to be chosen for inclusion into the next block is higher. Users have to find a cost/speed balance according to how critical the transactions is and how much they want to spent. The proper gas price is affected by the size of the mining pool and the volume of transactions. According to the current state of MainNet, a safe choice for the gas price, in order to have a transaction executed without significant delays is 2 GWei (1 ether = 10^9 GWei).

In order to assess the cost of deploying and using the implementation described herein, we have deployed the contracts to TestNet in order to measure the exact amount of gas needed. Then the methods of the contracts have been used in order to emulate all different use cases of the implementation. The deployment of the Authentication.sol contract needs 352,968 gas units while the deployment of the Erasmus.sol contract needs 3,073,820 gas units. These deployment costs are analyzed in Table I. Additionally the gas units needed for each of the use cases are presented in Table II.

The gas units in the results have been transformed in Ether and fiat (€) by using a gas price equal to 2 GWei and the value of 1 Ether = 569 €, as this is the exchange rate on 11th of March of 2018.

VII. CONCLUSIONS

A proof of concept decentralized application has been implemented for the use case of data exchange between Universities regarding Erasmus program. The ease of development, the cost of deploying and using the application and the advantages of such an implementation are discussed in this Section.

Modern tools allow fast and easy development of decentralized applications. Connecting and communicating with smart contracts on Ethereum blockchain can easily be implemented through APIs like Web3. With such tooling, developing client side applications does not differ so much from developing applications that use traditional databases. Blockchain development ecosystem have matured through recent years and developers can use the same skill set they already have to solve the problems in a new way. That is an important factor for the adoption of blockchain technology as the development of multiple applications is a prerequisite for adoption of the technology by end users.

The cost for the deployment of the two smart contracts is 3.9 €, but this is payed once upfront. The usage cost that have been induced for the required functionality is approximately 2.1 € per student, if the student registers for the usual number

### Table I

<table>
<thead>
<tr>
<th>Contract</th>
<th>Deployment Gas</th>
<th>Transaction fee (ETH)</th>
<th>Transaction fee (Fiat)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication.sol</td>
<td>352968</td>
<td>0.00071 €</td>
<td>0.402 €</td>
</tr>
<tr>
<td>Erasmus.sol</td>
<td>3073820</td>
<td>0.00615 €</td>
<td>3.498 €</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>Function</th>
<th>Gas used</th>
<th>Transaction fee (ETH)</th>
<th>Transaction fee (Fiat)</th>
<th>Estimated usage per student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add new student</td>
<td>183682</td>
<td>0.00037 €</td>
<td>0.256 €</td>
<td>1</td>
</tr>
<tr>
<td>Add course</td>
<td>198596</td>
<td>0.00040 €</td>
<td>0.276 €</td>
<td>5</td>
</tr>
<tr>
<td>Registration</td>
<td>123442</td>
<td>0.00025 €</td>
<td>0.172 €</td>
<td>-</td>
</tr>
<tr>
<td>Set student’s erasmus university</td>
<td>69919</td>
<td>0.00014 €</td>
<td>0.097 €</td>
<td>1</td>
</tr>
<tr>
<td>Register course mark</td>
<td>66649</td>
<td>0.00013 €</td>
<td>0.093 €</td>
<td>5</td>
</tr>
<tr>
<td>Remove course</td>
<td>283887</td>
<td>0.00057 €</td>
<td>0.395 €</td>
<td>-</td>
</tr>
</tbody>
</table>

Modern tools allow fast and easy development of decentralized applications. Connecting and communicating with smart contracts on Ethereum blockchain can easily be implemented through APIs like Web3. With such tooling, developing client side applications does not differ so much from developing applications that use traditional databases. Blockchain development ecosystem have matured through recent years and developers can use the same skill set they already have to solve the problems in a new way. That is an important factor for the adoption of blockchain technology as the development of multiple applications is a prerequisite for adoption of the technology by end users.
of 5 courses. The cost is regarded low, when you take into account what are the specifications of the service. The marks of the student are registered on the blockchain theoretically for unlimited time. Additionally there is no chance of any integrity violation, while there is not any downtime as the Ethereum EVM is always available. The cost of the service cannot be directly compared to that of traditional services, because the model of billing is completely different.

There are some important qualitative advantages of the implementation proposed. There is no need for nodes (Universities) to trust each other or even to put trust on a central node responsible for storing or managing data. The integrity of the student data stored in the platform is cryptographically ensured. No one has the power to manipulate the data or the software of the service, in order to either take advantage or harm others. As soon as the contract is deployed in the Ethereum network no one, not even the user that deployed that, has the power to stop it alter it or change data stored in it.

As future work we plan to work on enhancing the framework in a way that we can also protect the privacy of the students. This first implementation practically exposes courses and marks of each student on the blockchain. Cryptographic techniques may be employed though, in order to encrypt the data of-chain before submitting those to the smart contract.

REFERENCES