**Federal State Autonomous Educational Institution for
Higher Education National Research University
Higher School of Economics**

**Faculty of Computer Science
BSc Applied Mathematics and
Information Science**

BACHELOR'S THESIS
Program project
«Developing Open-source Voting System Based on Blockchain»

**Submitted by Igor Iunash:
Student of group 186, 4th year of study**

**Approved by Supervisor:
Candidate of Sciences, Associate Professor, Oleg Sukhoroslov**

**Co-supervisor:
Researcher, Visiting Lecturer, Yash Madhwal**

**Consultant:
Candidate of Sciences, Associate Professor, Yury Yanovich**

Moscow, 2022

Table of Contents

Annotation

Nowadays, the problem of conducting electronic voting without usage of special equipment is crucial. A convenient way to cast your vote is to use a smartphone with Internet access. But for the participants, the honesty and transparency of the electoral process remain significant criteria. At the same time, it is necessary to maintain the privacy and anonymity of users of the electronic system.

This work presents a blockchain-based platform for conducting an electronic Teaching Quality Assessment (TQA) at the Higher School of Economics using a linkable ring signature, smart contracts on the Ethereum platform and a client application for the iOS platform.

Аннотация

В наши дни актуальна проблема проведения электронных голосований без использования специальной аппаратуры. Удобным способом отдачи голоса является использования смартфона с доступом в интернет. Но для участников все еще остаются важными критерии честности и прозрачности выборного процесса. При этом, необходимо сохранять приватность и анонимность пользователей электронной системы.

В этой работе представлена платформа для проведения электронной студенческой оценки преподавания (СОП) в Высшей школе экономики на основе блокчейна с использованием связываемой кольцевой подписи и умных контрактов на платформе Ethereum, а также реализовано клиентское приложение для платформы iOS.

Keywords

e-voting, blockchain, smart contract, ring signature, mobile application

1. Introduction

The issue of creating an open electronic voting (e-voting) system has become more crucial lately. The COVID-19 pandemic led to most common operations switching to remote format. However, there still remains the need to conduct state and municipal elections as well as corporate polls and other decision-making processes. Hereinafter the terms "polls", "voting" and "elections" imply any form of public will (limited list polls, multiple choice polls, elections) unless otherwise stated.

A traditional form of voting always has an authority responsible for organizing, running, controlling, and counting processes. Most often this role is assigned to the most trustworthy party. However, the process associated with this party is the most vulnerable to an attack aiming to influence the result. Therefore, the digitalization of these processes and distancing from any specific parties let the voting be more transparent, thus gaining more trust from the participants and other interested parties.

Traditional expenses, such as hiring employees, renting an office, etc., that are tied with conducting the above processes can also be decreased by integrating an e-voting system.

In this article, some theoretical approaches to implementing e-voting will be reviewed. The interconnection between selected voting specifications (anonymous/public, revoting, vote check, etc.) and corresponding technical limitations will be analyzed. The analysis of existing e-voting solutions will be carried out. In addition, an MVP featuring a specific case will be implemented.

For implementing any electronic system some components are required, such as a database (data storage) and an entity interacting with user input and the database itself. In recent years blockchain technologies have become widespread. These technologies can be used for implementing a database. A blockchain is a decentralized, distributed digital ledger consisting of records. It features the immutability and auditability of the records. Being based on blockchain, the Bitcoin made a significant contribution to its popularisation.

Smart-contracts technology can be used as the platform for the database interaction. This is a class of algorithms that can do operations on blockchain and data in it. They are triggered by making transactions (adding new blocks to a blockchain).

The most popular platform for launching smart contracts is Ethereum: it gives more flexible tools to create more complex flows in comparison with bitcoin smart contracts.

2. Literature review

The issue of organizing non-centralized online voting is not new. Some of the solutions were implemented plenty of years ago in the 20th century. They were generally based on several cryptography methods combined with each other. E.g., Fujioka – Okamoto – Ohta protocol [2] was invented in 1993. It gives an ability to build flow for digital voting with such tools as cryptography blind sign and flashing cipher.

One of the first e-elections, which was organized by the government, was the republican ballot in Geneva [3]: it is worth noting that the platform used for it is completely open-source and is available on GitHub [4].

Some people consider the platform used in Estonian elections a state-of-the-art approach for the e-voting digital system. It uses a citizen ID card to identify the user. Because of that, some independent researchers agree that it is not a completely anonymous or secret voting.

The other concern connected with digital elections is that any voting over the Internet cannot be 100% private. But this topic will be left out of this paper: it may be solved by ubiquitous Web3 spreading soon.

The other way to increase the level of trust is using an independent company that helps to host a voting [6]. In comparison with self-organized voting, it may be more private because such companies care about their reputation and will not collude with anyone. But from a research point of view, this approach definitely is not the best as we are trying to review the worst situation to make the platform as secure as possible.

There is a family of Direct Recording Electronic [19] machines (DRE/DRE-i/DRE-ip) which were used during offline electronic elections in different countries including the United Kingdom.

The main problem of all these solutions is that users have to rely on different black box implementations: web servers or tallying machines. It is not very different from trusting a human being. Users cannot be sure, that there is no impostor among organizers or developers of software.

The goal of this article is to create such a system, that will be completely transparent for users, in comparison with the related works above.

3. Possible problems

Before coding the system, we need to describe technical constraints because they will affect the final result. While organizing voting there is always truth preservation challenge and privacy leakage problem: the system must be constructed in such a way that nobody could ever influence the elections. The following lists possible ways of "hacking" a ballot:

Publication of fake results

The main idea is that if a malicious participant can collude with the organizer of the voting, the organizer may publish incorrect results based not on the real voters' choices but the changed ones. To solve this problem, the system may be constructed in a such way that all operations would be public and audible (with enough care about privacy). One way of doing this is using blockchain as a database and smart contracts on top of it.

Adding non-existing votes

Influencing elections is also possible via throwing additional votes into the system, to archive an expected result. In this case, while the audition process true and published results would be the same, but would differ with votes, that were given by real participants, because except them also 'dead souls' have participated in the voting. To deal with this challenge organizers might publish the list of users who will live their vote through the system, thus the maximum number of answers will be known before the election and it will be possible to check that only the correct people are participating.

Votes censoring

This "hack" may take place after fixing the previous one: some part of the real votes may be ignored and some other data may be added to the system instead of them. To fix this problem, we can allow users to see their vote any time before or after the ballot ends, so that they would be able to check its correctness and complain if it is not correct.

Double voting

There is another problem that must be taken into consideration in the beginning of infrastructure constructing. It is the possibility of several votes being sent by one voter.

While creating the system that solves all the problems above, we need to care about the user's privacy (if it is not public-answers voting). Hence, we need to implement a system featuring several properties:

- Only the voter can see their vote
- It is not possible to determine a voter by any vote
- It is not possible to determine if a voter has already cast a vote

Generally, to protect privacy, systems use different cryptography methods. It also increases computation complexity and, consequently, hardware requirements. All of this can lead to a DOS attack possibility. So, it is necessary to find a balance in this trade-off.

4. Results anticipated

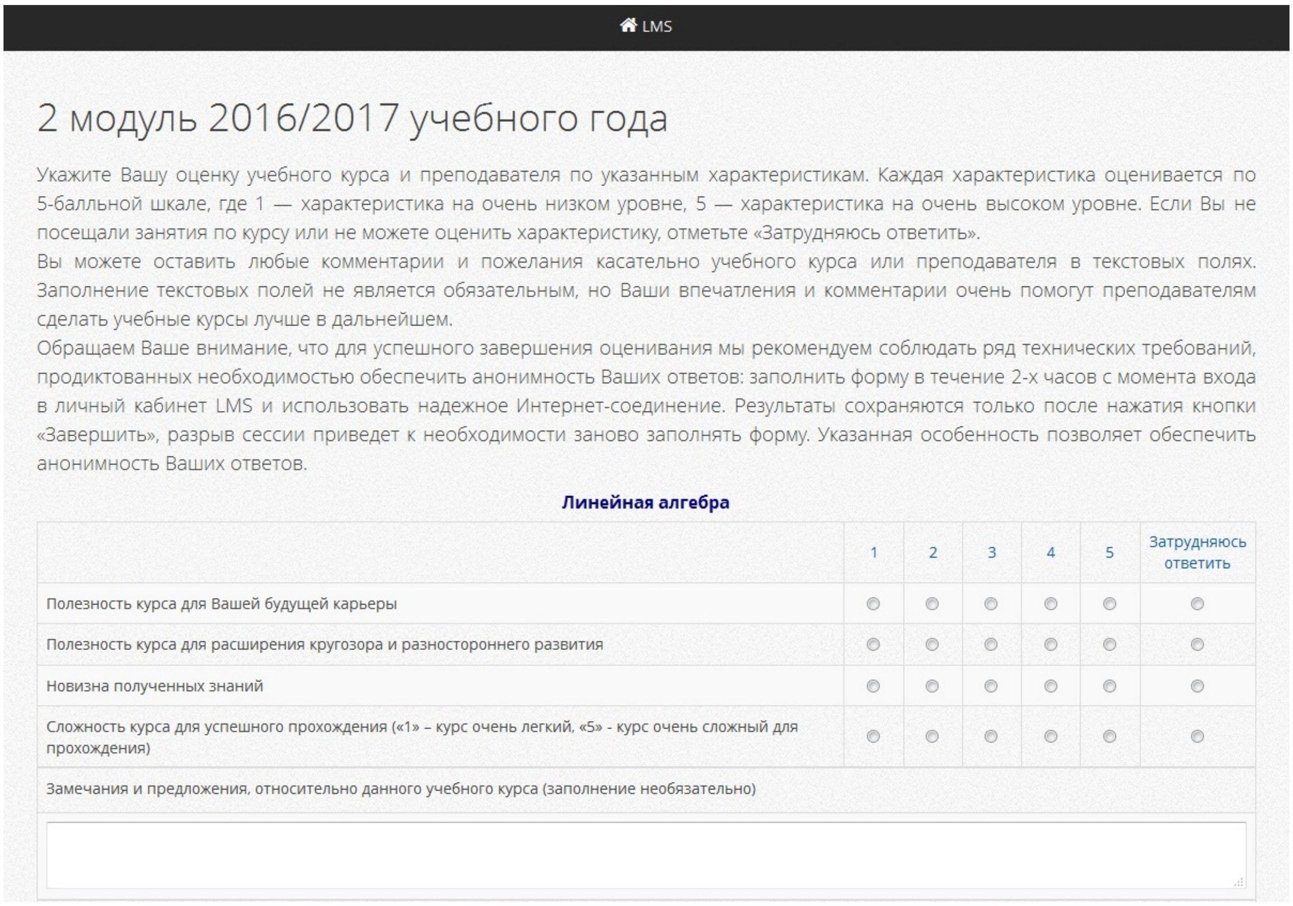
As a result of this article, it is planned to create an MVP of a platform that will make it possible to organize Teaching Quality Assessment (TQA/СОП) in NRU HSE. The platform will use blockchain and smart contracts.

Currently within TQA students can rate their courses and teachers from 1 to 5 on the following criteria [7]:

Course evaluation criteria	**Teacher evaluation criteria**
The practical value of the course for a student’s future career	Clarity of requirements to students
The practical value of the course for broadening a student’s horizons and diversified personal development	Clarity and consistency of study materials
Novelty of knowledge gained in the course	Communication between teachers and their audience
Difficulty of successfully completing the course	Teachers’ availability for extracurricular discussion of any academic issues

Table №1

In such form, TQA has already existed for more than 6 years. Officially, this system is fully anonymous and none of the organizers can determine the caster of any vote. On the other hand, this system is fully hosted and controlled by the department of information technologies and nobody can check the real system code.


LMS

2 модуль 2016/2017 учебного года

Укажите Вашу оценку учебного курса и преподавателя по указанным характеристикам. Каждая характеристика оценивается по 5-балльной шкале, где 1 — характеристика на очень низком уровне, 5 — характеристика на очень высоком уровне. Если Вы не посещали занятия по курсу или не можете оценить характеристику, отметьте «Затрудняюсь ответить».
Вы можете оставить любые комментарии и пожелания касательно учебного курса или преподавателя в текстовых полях. Заполнение текстовых полей не является обязательным, но Ваши впечатления и комментарии очень помогут преподавателям сделать учебные курсы лучше в дальнейшем.
Обращаем Ваше внимание, что для успешного завершения оценивания мы рекомендуем соблюдать ряд технических требований, продиктованных необходимостью обеспечить анонимность Ваших ответов: заполнить форму в течение 2-х часов с момента входа в личный кабинет LMS и использовать надежное Интернет-соединение. Результаты сохраняются только после нажатия кнопки «Завершить», разрыв сессии приведет к необходимости заново заполнять форму. Указанная особенность позволяет обеспечить анонимность Ваших ответов.

Линейная алгебра

	1	2	3	4	5	Затрудняюсь ответить
Полезность курса для Вашей будущей карьеры	○	○	○	○	○	○
Полезность курса для расширения кругозора и разностороннего развития	○	○	○	○	○	○
Новизна полученных знаний	○	○	○	○	○	○
Сложность курса для успешного прохождения («1» – курс очень легкий, «5» - курс очень сложный для прохождения)	○	○	○	○	○	○

Замечания и предложения, относительно данного учебного курса (заполнение необязательно)

Image №1. Actual UI of TQA.

To make the system more transparent and to increase the level of trust, this article presents an open-source platform that will allow organizing electronic TQA and will meet several requirements:

1) Answers are available only to a predefined number of users
2) Only the voter can see their vote
3) It is not possible to determine a voter by any vote
4) It is possible to get the list of the users who voted
5) All process of sending and processing votes is transparent
6) The platform has good UX (non-formal requirement)

Any of these requirements may be prioritized depending on the importance level.

5. Implementation

Naive approach

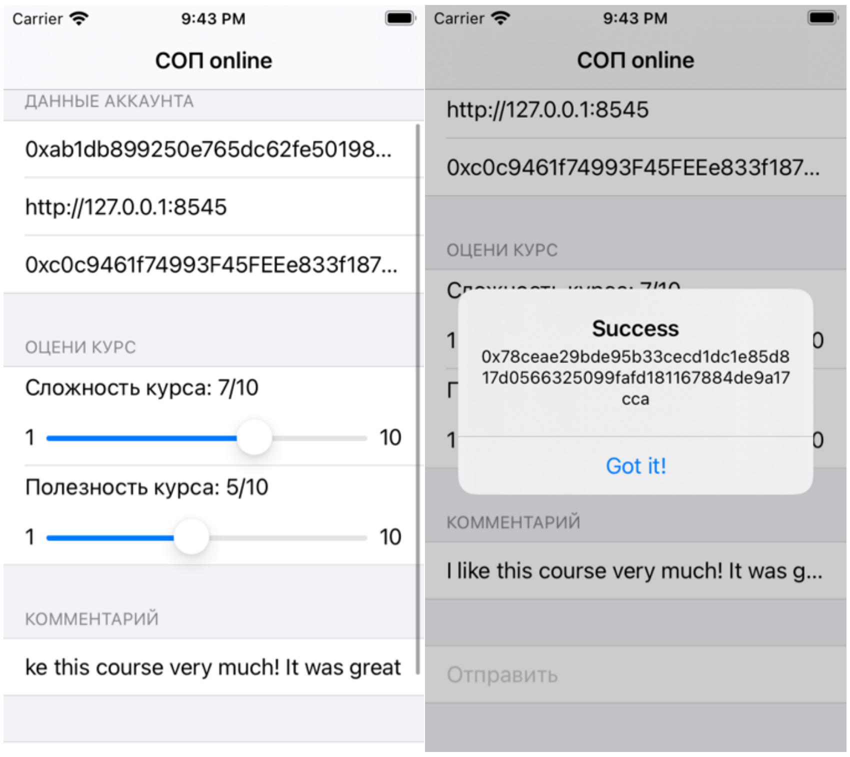
The final implementation has to match several criteria listed above. As mentioned, to avoid the problem of publishing non-existent votes, you need to announce the list of participants in advance. TQA is organized separately for each academic stream and information about students in this stream is public. Therefore, we can determine the maximum number of participants in a particular vote in advance.

At the same time, students of one stream should not be allowed to participate in the TQA of the other stream. Hence, it is necessary to check the voting permissions of every user in the voting smart contract. At the same time, it is necessary to preserve the privacy property in which it is impossible to determine the user who cast the vote.

The proposed solution is to generate pairs of private and public keys in advance (before a voting begins). Make the entire list of public keys available to everybody (without telling which public key belongs to which user). Send private keys confidentially to each student via confidential transmission channel. Thus, each student will be able to check if they are to participate in the voting and to make sure that no one from the street participates in it. In this case, if one of the fellow students does not find his public key, they will file an appeal. However, no one knows to whom exactly each of the public keys belongs.

Also, before voting starts, a smart contract is created, in which the creator adds general information about voting (start date and time, name of the voting, etc.) and a list of participants' public keys. A pair of public and private keys for the smart contract *PK* and *SK*, respectively, is also generated. *PK* is added to the contract, and SK is stored off-chain. With *PK*, users will encrypt their votes before they are sent to the smart contract. And with the help of *SK*, the votes will be decrypted after the end of the voting.

This implementation could be found in the GitHub repository of this article [8] in the "naive_approach" branch. This branch contains the code of a smart contract on Solidity language. It also uses the *Brownie* Python library [9] for testing and deploying the smart contract. The repository also contains code of a simple iOS app on SwiftUI to make transactions for the deployed smart contract.


Carrier 9:43 PM

СОП online

ДАННЫЕ АККАУНТА

0xab1db899250e765dc62fe50198...

http://127.0.0.1:8545

0xc0c9461f74993F45FEEe833f187...

ОЦЕНИ КУРС

Сложность курса: 7/10

1 10

Полезность курса: 5/10

1 10

КОММЕНТАРИЙ

ke this course very much! It was great

Carrier 9:43 PM

СОП online

http://127.0.0.1:8545

0xc0c9461f74993F45FEEe833f187...

ОЦЕНИ КУРС

Success

0x78ceae29bde95b33cecd1dc1e85d817d0566325099fafd181167884de9a17cca

Got it!

КОММЕНТАРИЙ

I like this course very much! It was g...

Отправить

Image №2. UI of basic app for TQA

Of course, this approach does not meet all the listed requirements and has a few other disadvantages (for example, we cannot verify the correctness of a user's vote because of the encryption and a user can corrupt their bulletin).

Requirement	**Comment**
Answers are available only to a predefined number of users	Implemented Only users with private key have access.
Only the voter can see their vote	Implemented A user can check their transaction by Tx ID.
It is not possible to determine a voter by any vote	Half implemented. We have partially decoupled a user and their account.
It is possible to get the list of the users who voted	Not implemented. Even though we can get information about the accounts which did not vote, we cannot list real people.
All process of sending and processing vote is transparent	Implemented by using the smart contract technology.

The platform has good UX (non-formal requirement)	Not implemented. The iOS app has poor UI and UX.

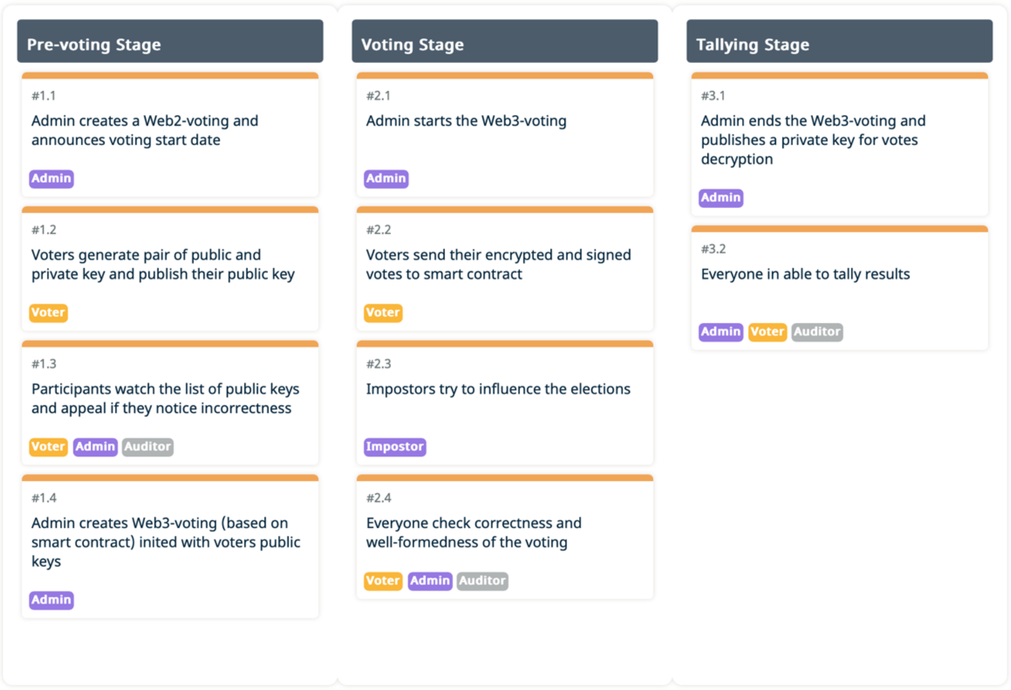
Table №2

Better approach

In a naive approach to solving the problem, a certain "confidential transmission channel" was used to transfer the private key. But the creation of such a channel is a separate non-trivial task for solving. It would be much better to have a way in which the number of opportunities for data leakage would be minimized.

A ring signature is a cryptographic mechanism that makes it possible to verify that a certain message was signed by one of the participants from a predetermined list of participants, while we cannot know which one exactly by. In the first approximation, this is what we would like to use when implementing electronic voting: we compile a list of students' public keys that is accessible to everyone. Next, the student signs his vote with a ring signature, while no one knows which of the participants signed the vote. The problem is that when using a simple ring signature, each of the participants will be able to send an unlimited number of votes, but due to complete anonymity, we will not be able to prevent this.

Here come linkable [10] or traceable [11] ring signatures, which allow you to understand that several votes were left by one participant and even determine which one. In this case, the organizer or anyone who wishes will be able to check at any time that none of the two votes were left by the same participant.



Pre-voting Stage

#1.1
Admin creates a Web2-voting and announces voting start date
Admin

#1.2
Voters generate pair of public and private key and publish their public key
Voter

#1.3
Participants watch the list of public keys and appeal if they notice incorrectness
Voter Admin Auditor

#1.4
Admin creates Web3-voting (based on smart contract) inited with voters public keys
Admin

Voting Stage

#2.1
Admin starts the Web3-voting
Admin

#2.2
Voters send their encrypted and signed votes to smart contract
Voter

#2.3
Impostors try to influence the elections
Impostor

#2.4
Everyone check correctness and well-formedness of the voting
Voter Admin Auditor

Tallying Stage

#3.1
Admin ends the Web3-voting and publishes a private key for votes decryption
Admin

#3.2
Everyone in able to tally results
Admin Voter Auditor

Image №3. Diagram of the approach.

Each of the voting stages will be analyzed in more detail below:

Pre-voting Stage

Before starting the main voting process, it is necessary to carry out some preparatory procedures: create smart contracts for voting with a list of public keys of participants, and for this, participants need to generate and report them. Let's analyze this stage in more detail:

1) Before the start of the evaluation period for educational courses, the faculty administrator creates a record in the database with information about the need for certain students for each study group to pass a certain vote. This step is running on Web2.

   In the mobile applications department in HSE, we already have a cluster of Kubernetes microservice, so it was easy to make another service for this task.

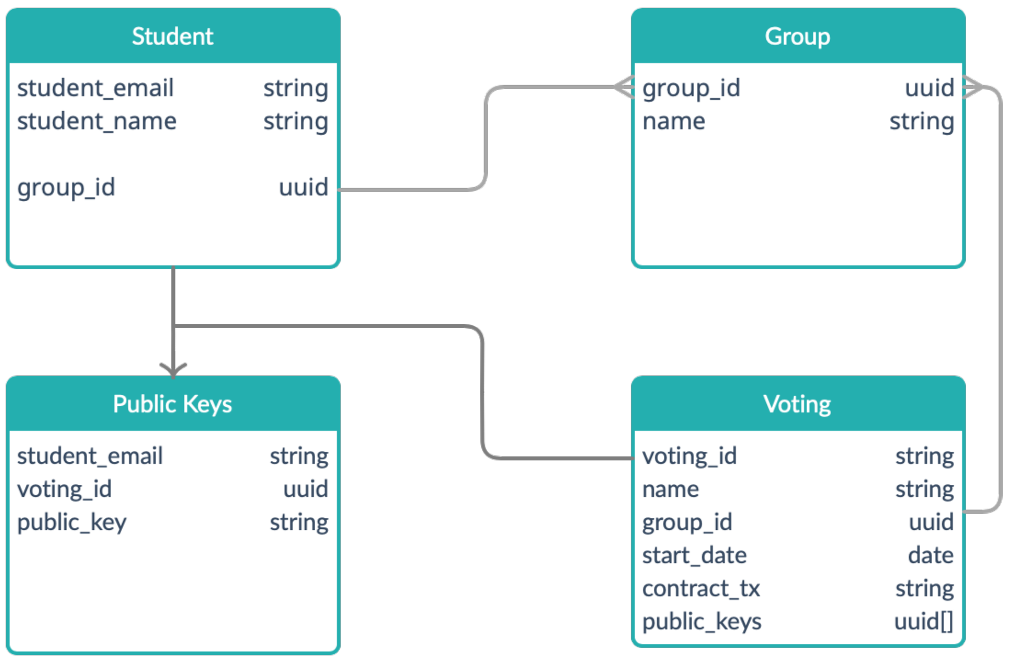


Image №4. ER-diagram of database

2) Voters open the mobile app and authorize using their corporative email address. Next, they can generate a pair of public and private keys and send them to the database. This step is also running on Web2.

   To implement this stage for the TQA, we have added a new section to the official application of the university called HSE App X. The application already had the functionality of OAuth authorization using corporate email based on Microsoft ADFS technology, as well as a mechanism for interacting with the backend of mobile applications. To identify the user, JWT keys are used, inside of which the user's email is located, so we can unambiguously determine which study group the user belongs to using the students table from image № 4.

   For those users who did not have time to create public and private keys, they will be created automatically. The public key will be published and the private key will be available via a one-time link: this mechanism is not private enough, since the organizers can keep access to the private key, but the only alternative is to completely prevent the user from voting, which is a bad option.

3) Due to the fact that we do not need to hide the relationship between public keys and students, any student can check that only his classmates will

participate in the voting (or rather, their corresponding public keys). If anyone notices an error, he will be able to declare the incorrectness of the process before the voting begins.

4) After the end of the registration stage, the administrator creates a smart contract for a specific vote and adds a list of public keys of participating students to it. The structure of the smart contract will be described separately below.

The administrator also adds the smart contact address to the voting object so that the participants know which address to contact on the next stage.

Voting Stage

During this stage voters (students) send their votes to the smart contract.

1) Admin starts the voting in the smart contract so that voters are able to send their votes.
2) Students open the voting section in HSE App X. In this section, they can rate their courses and teachers.

   After they have left their feedback, the application will encrypt their vote using a public encryption key from the smart contract. Next, a student will have to paste the private signing key, which will be used to sign their vote. They also need to provide a private key for their Ethereum account. This private key must be different from the signature private key to decouple the user from his account. It does not matter which account a student uses for sending a transaction, because the user anonymous identification is done with the ring signature mechanism.

   The last thing they need is to send the transaction.

The application gives an opportunity to make everything smooth and provides a convenient user experience. But users are always able to make everything by themselves (encrypt, sign, and send a vote), if they do not trust the app for some reason.

3) If any impostors existed, they would also try to participate in the voting. They could try to influence the results of the voting in several ways: to pretend to be some other person or to try to send several votes. They would successfully send a transaction with the wrong data because on the fly sign checking in

Ethereum virtual machine is a very expensive operation and costs a lot of gas. So, we would detect any errors asynchronously.

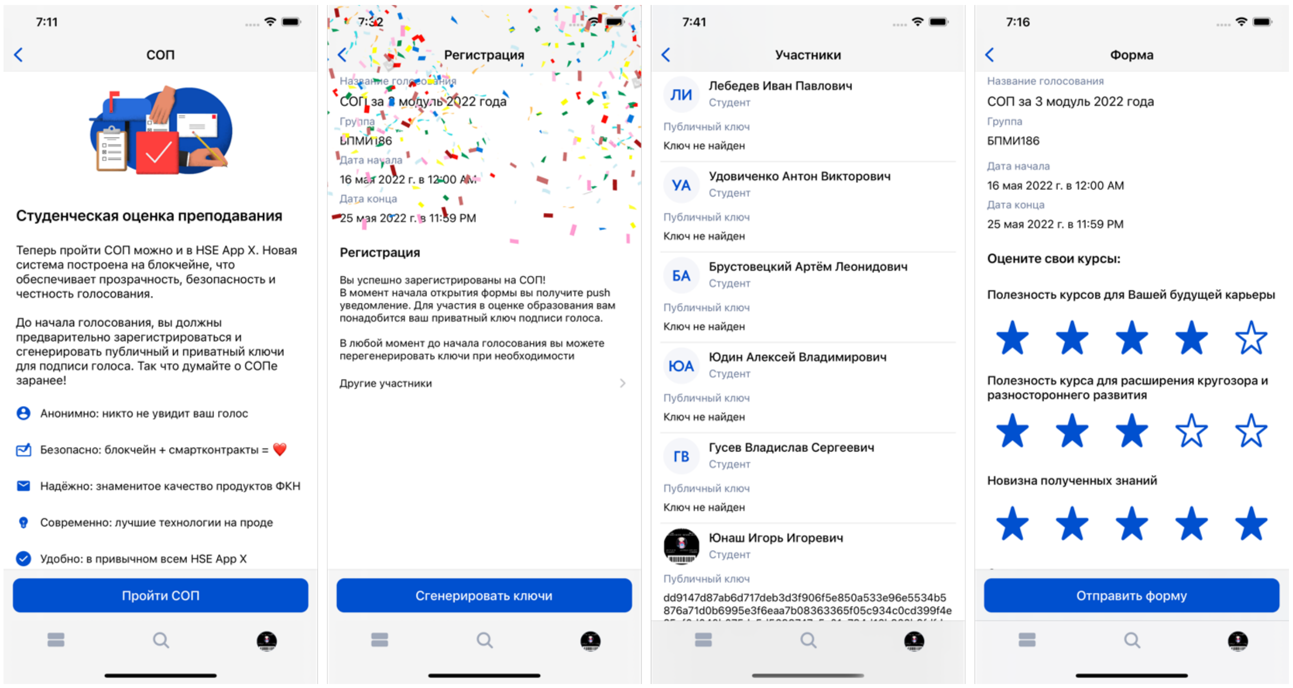
4) Any person is able to audit votes in a particular smart contract: all public keys are available and all encrypted and signed votes are available, so, anyone can verify signatures and check that only one vote is cast by one person.

Tallying Stage

1. Admin of the voting publishes the private key, which will be used to decrypt votes. It is necessary to keep the private key secret before the end of voting. But if it leaks in some way, it will only lead to the recorded votes revealing. It will not be able to change them due to the ring signature.
2. After the private key publication, everyone is able to tally the results. During tally, it is necessary to verify the signatures of votes and check if any two votes are not linked in terms of the linked ring signatures.

iOS Application

As it was said earlier, this approach is realized in HSE App X – the official university mobile app. It uses UIKit to create an interface.



Студенческая оценка преподавания

Теперь пройти СОП можно и в HSE App X. Новая система построена на блокчейне, что обеспечивает прозрачность, безопасность и честность голосования.

До начала голосования, вы должны предварительно зарегистрироваться и сгенерировать публичный и приватный ключи для подписи голоса. Так что думайте о СОПе заранее!

- Анонимно: никто не увидит ваш голос
- Безопасно: блокчейн + смартконтракты = ❤️
- Надёжно: знаменитое качество продуктов ФКН
- Современно: лучшие технологии на проде
- Удобно: в привычном всем HSE App X

Название голосования
СОП за 3 модуль 2022 года
Группа
БПМИ186
Дата начала
16 мая 2022 г. в 12:00 AM
Дата конца
25 мая 2022 г. в 11:59 PM

Регистрация

Вы успешно зарегистрированы на СОП!
В момент начала открытия формы вы получите push уведомление. Для участия в оценке образования вам понадобится ваш приватный ключ подписи голоса.

В любой момент до начала голосования вы можете перегенерировать ключи при необходимости

Другие участники

ЛИ Лебедев Иван Павлович
Студент
Публичный ключ
Ключ не найден

УА Удовиченко Антон Викторович
Студент
Публичный ключ
Ключ не найден

БА Брустовецкий Артём Леонидович
Студент
Публичный ключ
Ключ не найден

ЮА Юдин Алексей Владимирович
Студент
Публичный ключ
Ключ не найден

ГВ Гусев Владислав Сергеевич
Студент
Публичный ключ
Ключ не найден

Юнаш Игорь Игоревич
Студент
Публичный ключ
dd9147d87ab6d717deb3d3f906f5e850a533e96e5534b5
876a71d0b6995e3f6eaa7b08363365f05c934c0cd399f4e

Название голосования
СОП за 3 модуль 2022 года
Группа
БПМИ186
Дата начала
16 мая 2022 г. в 12:00 AM
Дата конца
25 мая 2022 г. в 11:59 PM

Оцените свои курсы:

Полезность курсов для Вашей будущей карьеры
★ ★ ★ ★ ☆

Полезность курса для расширения кругозора и разностороннего развития
★ ★ ★ ☆ ☆

Новизна полученных знаний
★ ★ ★ ★ ★

Image №5. Screenshots of the TQA in HSE App X

We use an open-source Linkable Ring Signature (LRS) realization [12] for Node.js. It is not a trivial task to work with Node.js packages in Swift environment.

First of all, we need to give an ability for the package to run out of Node.js environment. It is possible to do with the help of *browserify* library [13]. It makes one big JS bundle from all Node.js requirements. In this case, we need to create a file called 'pre-bundle.js', which import LRS library, and run a bash command:

browserify pre-bundle.js --standalone lrs > lrs-bundle.js

As the result, we get a pure JS file which we can run in browser.

The next step is to run this code in some way on iOS apps. Apple provides a native way to run JavaScript code – JavaScriptCore [14]. It is helpful with simple scripts, but after many hours of attempts to run LRS library in JavaScriptCore, it did not lead to any result.

The other way to run JavaScript code in Swift is to make a fake WebView [15] – an in-app version of the browser – and inject JS code inside it. This approach has worked for the chosen LRS bundle!

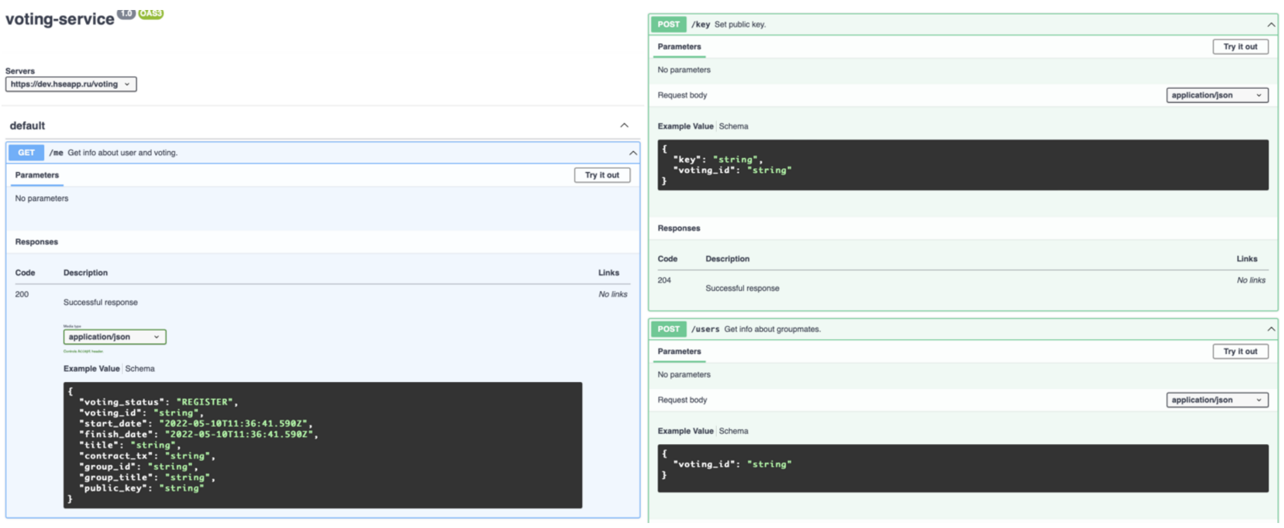
As it was mentioned earlier, HSE App X has OAuth authentication. It was created using an open-source HSEAuth library [16]. After logging in users get a JWT access token, which is used for all network requests to HSE mobile backend. It also encapsulates a token refreshing mechanism, that helps to increase the duration of a user session.

The Web2 network layer is done with *Alamofire* library. Backend API represents canonical REST API with GET, POST, and DELETE methods.

To communicate with a smart contract a *Web3.Swift* library [18] is used. It allows the application to call smart contract methods and make transactions. It is possible to call its methods in the modern async/await paradigm or classical closure approach.

Web2 Backend

Before communicating with the smart contract, a user has to register to the voting and get a contract address. It is done in a Web2 environment and with the help of REST API. HSE Mobile department already has two Kubernetes clusters with many microservices. It also has a dev and production environment to save real users from not verified code.


voting-service 1.0 OAS3

Servers
https://dev.hseapp.ru/voting

default

GET /me Get info about user and voting.

Parameters

No parameters

Responses

Code	Description	Links
200	Successful response	No links

Media type: application/json

Controls Accept header

Example Value | Schema

```
{
  "voting_status": "REGISTER",
  "voting_id": "string",
  "start_date": "2022-05-10T11:36:41.590Z",
  "finish_date": "2022-05-10T11:36:41.590Z",
  "title": "string",
  "contract_tx": "string",
  "group_id": "string",
  "group_title": "string",
  "public_key": "string"
}
```

POST /key Set public key.

Parameters

No parameters

Request body: application/json

Example Value | Schema

```
{
  "key": "string",
  "voting_id": "string"
}
```

Responses

Code	Description	Links
204	Successful response	No links

POST /users Get info about groupmates.

Parameters

No parameters

Request body: application/json

Example Value | Schema

```
{
  "voting_id": "string"
}
```

Image №6. Swagger for API requests

Node.js framework is used for the voting microservice. MongoDB is used as a storage for all data about users, groups, voting, and corresponding public keys. To document and specify all API requests, Swagger is used: image №6 illustrates a part of this documentation.

Smart Contract

The smart contract has not changed a lot from the first version: Solidity is still used as a development language; Brownie library is used for testing and deployment purposes.

```
enum VotingStatus {TOO_EARLY, ACTIVE, FINISHED}
VotingStatus public voting_status;

string public name;
address creator;

bytes[] public votes;
string[] public signs;

string[] public voters_public_keys;

string public public_encryption_key;
string public secret_decryption_key;
```

Listing №1. Variables of the smart contract

The main logic of the smart contract is still quite simple:

- Create a smart contract
- Fill all the necessary fields
- Start a voting
- Vote
- End the voting

- Tally the results

Note: public variable *secret_decryption_key* must be set only after *VoingStatus* became *FINISHED*.

Better approach evaluation

Requirement	**Comment**
Answers are available only to a predefined number of users	Implemented Only users with a private key have access.
Only the voter can see their vote	Implemented. A user can check their transaction by Tx ID.
It is not possible to determine a voter by any vote	Implemented. We completely separated the user, their account and their vote using the ring signature.
It is possible to get a list of the users who voted	Half-implemented. A user can mark that they have voted via Web2 environment but it is not verifiable.
All process of sending and processing vote is transparent	Implemented by using the smart contract technology.
The platform has good UX (non-formal requirement)	Implemented. The iOS app has good UX and pretty design.

Table №3

6. Testing and Deployment

Development environment

For the development environment, ganache-cli tool is used for running Ethereum and a local network for communication.

Ganache gives an ability to test smart contracts without an overhead of a real network. It does not have any proof-of-stake/proof-of-work mechanism, so it is strictly not recommended to use ganache in the production environment.

To be able to communicate, the mobile application and ganache must be on the same network. The other way is to publish the necessary port with ngrok command line utility: it will give the public available *HTTP(S)* address for your local ganache.

For the Web2 network layer, we have a separate development environment (located at https://dev.hseapp.ru). It has an individual Kubernetes cluster, its own Mongo Database, etc. The CI/CD process is done with a Google Skaffold tool, which helps to deploy Kubernetes containers, and with Gitlab CI, which helps to run tasks on commit events.

The mobile application also has different environments: the development version can be downloaded from the TestFlight app (Apple's service for pre-release testing).

Production environment

When the development process is done, we can move to the production environment. To deploy a smart contract, we can use a public test environment (for example, Rinkeby), production Ethereum main network, or set up our own private Ethereum network.

The table №4 shows a comparison among these approaches.

Criterion	**Rinkeby**	**Mainnet**	**Private network**
Time	Fast enough Test network does not use proof-of-work protocol, so all operations are fast	Slow To make a transaction it must be mined, and it takes some time.	Fast There are not many transactions and the admin is able to choose which protocol to use
Fiat money is required	No Users can get ETH from test faucet	Yes Users (or admins) have to buy some ETH to make transactions	No Admins can create private faucet or create plenty of pre-filled wallets.
Secure	Not really It uses proof-of-authority protocol which is not as secure as proof-of-work	Yes It has many nodes and uses proof-of-work protocol.	It depends on the protocol which admin may choose for private network.

Table №4

In this article Rinkeby network is used, because it is the easiest way to test smart contracts in the production environment. Later, when all the parts of the system are completely finished and approved, it will be possible to launch HSE private Ethereum network for better security.

7. Conclusion

In this work the issue of organizing e-voting and known approaches are described. New system for Teaching Quality Assessment proposed: existing solution was discussed and technical requirements for the new one were described.

A smart contract for e-voting was created and deployed in development and production environments: it uses Linkable Ring Signature for preserving anonymity and truthfulness. It was integrated with the HSE App X iOS application and Web2 backend on Node.js and MongoDB.

Results are publicly available on GitHub [8] and can be used by everyone. It almost completely meets all the listed criteria and desired results. But there still are things to develop and improve:

- Change linkable ring signature to traceable ring signature
- Test approach with ring signature validation inside smart contract and not outside
- Implement TQA in the Android version of HSE App X

8. References

[1] L. Cranor, “Electronic Voting – Computerized polls may save money, protect privacy.” XRDS: Crossroads, The ACM Magazine for Students, Volume 2, Issue 4, March 1996, pp 12–16

[2] Fujioka, T. Okamoto, K. Ohta, “A practical secret voting scheme for large scale elections,” Seberry J., Zheng Y. (eds) Advances in Cryptology — AUSCRYPT '92. AUSCRYPT, 1992, Lecture Notes in Computer Science, vol 718. Springer, Berlin, Heidel

[3] “Electronic voting in Switzerland,” Wikipedia, May 17, 2020. https://en.wikipedia.org/wiki/Electronic_voting_in_Switzerland.

[4] “CHVote,” GitHub, Mar. 23, 2022. https://github.com/republique-et-canton-de-geneve/chvote-1-0 (accessed Mar. 20, 2022).

[5] S. Heiberg and J. Willemson, "Verifiable internet voting in Estonia," 2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE), 2014, pp. 1-8, doi: 10.1109/EVOTE.2014.7001135.

[6] “E-voting: проведение электронных голосований,” E-vote, https://www.e-vote.ru (accessed Mar. 15, 2022)

[7] “TQA Methodology,” NRU HSE, https://www.hse.ru/en/evaluation/method (accessed Mar. 24, 2022).

[8] Article GitHub Repository, https://github.com/Igralino/CourseWork

[9] “Brownie: A python-based development and testing framework for smart contracts”, GitHub, https://github.com/eth-brownie/brownie (accessed Mar. 27, 2022).

[10] Liu, Joseph K.; Wong, Duncan S. (2005). Linkable ring signatures: Security models and new schemes. ICCSA. Lecture Notes in Computer Science. Vol. 2. pp. 614–623.

[11] Fujisaki, Eiichiro; Suzuki, Koutarou (2007). "Traceable Ring Signature". Public Key Cryptography: 181–200.

[12] “Linkable Ring Signature JS library”, GitHub, https://github.com/parabirb/lrs14 (accessed May 6, 2022).

[13] “browserify,” GitHub, Dec. 08, 2021. https://github.com/browserify/browserify

[14] “JavsScriptCore,” developer.apple.com. https://developer.apple.com/documentation/javascriptcore (accessed May 05, 2022).

[15] “WKWebView,” developer.apple.com. https://developer.apple.com/documentation/wkwebview (accessed May 05, 2022).

[16] “HSEAuth-iOS,” GitHub, Sep. 30, 2021. https://github.com/hseapp/HSEAuth-iOS (accessed May 06, 2022).

[17] “Alamofire/Alamofire,” GitHub, May 14, 2022. https://github.com/Alamofire/Alamofire (accessed May 06, 2022).

[18] “argentlabs/web3.swift,” GitHub, May 13, 2022. https://github.com/argentlabs/web3.swift (accessed May 06, 2022).

[19] "DRE-ip: A Verifiable E-Voting Scheme Without Tallying Authorities," Shahandashti, Siamak & Hao, Feng. (2016), Computer Security – ESORICS 2016.