BLOCKCHAIN DECENTRALIZED VOTING FOR VERIFIED USERS WITH A FOCUS ON ANONYMITY

Piotr Pospiech, Aleksander Marianski and Michal Kedziora

Department of Computer Science and Management, Wroclaw University of Science and Technology, Wroclaw, Poland

ABSTRACT

The paper presents decentralized voting scheme for verified users while maintaining their anonymity. A blockchain network was applied, which is a decentralized and distributed database based on the Peer-to-Peer architecture. During the implementation, the Ethereum network was used. Thanks to this, it is possible to code the terms of the contract required to perform the transaction. Ethereum and the use of smart contracts were also discussed in paper. The implementation uses the blind signature protocol by David Chaum and encryption with the Rivest-Shamir-Adleman (RSA) algorithm. Presented in this paper scheme for blockchain decentralized voting for verified users with focus on anonymity is then fully implemented and identified potential issues are analysed and discussed.

KEYWORDS

Blockchain, e-voting, Ethereum.

1. INTRODUCTION

With the development of blockchain technology, more decentralized systems and applications began to emerge. Bitcoin, as the first project to gain global popularity, introduced a Peer-to-Peer payment system where every user is equally important and has the same rights [8]. The lack of a transfer authorization unit increases the credibility of the system for users [9]. Nowadays, commonly used electronic storage and payment systems are those provided by banks. There is a central unit here which is self-interested and has unlimited power in the system. As a result, it can impose additional restrictions or new account rules on users who, due to the lack of alternatives, have to agree to them. An additional problem is the lack of trust in institutions. Central unit can lead to fraud [1].

User anonymity and the lack of a central unit to manage the system have gained popularity over time [13]. More blockchain-based systems have emerged that offer a variety of possibilities [15][18]. Bitcoin is no longer attractive for transactions due to high transaction fees and speed of transactions. In addition, methods for analyzing the network in search of the owner of the transaction were found, which enabled user identification [2]. The use of blockchain systems for payments is not the only possible use [20]. Many decentralized applications have been created that use the advantages of blockchain networks and get rid of the disadvantages of centralized systems [19]. The development of such applications made it possible to introduce smart contracts, which were used for the first time in the Ethereum network [16]. Decentralized applications are used where the central unit can be a weak point of the system. For example, electronic voting

David C. Wyld et al. (Eds): BIoT, DKMP, CCSEA, EMSA - 2021
pp. 55-63, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.110805

may be controlled and falsified by its organizer [7]. The topic of electronic voting itself is a very complex issue that is not easy to implement on a larger scale [10-12].

One of the first countries that introduced electronic voting was Estonia [3]. The system enabled remote voting, so users required verification. Citizens have owned modern ID cards. They had a chip containing electronic data, certificates and private keys. This information were protected with PIN codes. It allows confirming identity of voters online. Remote voting had an advantage over the traditional approach because it was possible to change the ballot. However, analysis of the voting system has identified many potential security gaps [5]. One server has stored data of users with their encrypted ballot. Another server was responsible for counting ballots. It was able only to decrypt the ballots without revealing the personal data of the voters. Notwithstanding, the cooperation of these two systems could potentially reveal voters. In this case, maintaining anonymity depends on the Trusted Third Party (TTP) and can be a vulnerable point of the entire system. Currently, many voting systems have been developed that use blockchain technologies [14]. One of them is the Voatz app [16]. Developers talk about the use of blockchain as follows: "All ballots are secured on multiple, restricted-access, geographically-distributed servers running on blockchain technology to ensure all election data remains tamperproof" [6]. It is a solution that is being introduced to an increasing number of elections in the United States. In 2018 it was used in the US Midterm Election in West Virginia, and in 2020 it made it possible to vote in the presidential election in Utah.

The objective of the paper is to develop and create a voting scheme. The purpose of the scheme and proof-of-concept application is to create and conduct voting. Participation in voting will require user verification, but the user himself will remain anonymous in the system and it will not be possible to check what a given user voted for. The results of the research can be used to verify the methods of maintaining user anonymity using the blockchain network.

Structure of the paper is as follows. The introduction contains a brief history of blockchain networks, cryptocurrencies and electronic voting. Research papers that deal with the same issue are discussed in the related works section. Some of the solutions proposed there were applied in this research. Then there is a section on blockchain technology, which introduces the topic of blockchain and its architecture, the Ethereum project and the blind signature protocol. The next section presents the technologies that were used in the application development process. Chapter Voting procedure presents in chronological manner the entire procedure of creating voting by its organizer, sending ballots, their verification and finally the presentation of the results. Then there is a chapter on the application itself. It presents the result of paper. The last chapter is a summary of the paper along with an indication of the direction of further work.

2. RELATED WORK

In order to increase the anonymity of users, many works introduced a blockchain network to the voting protocol [14]. The decentralization of the system solves the problem of entrusting all responsibility to one entity. It also enables transparency of the elections but imposed the necessity to implement the protocol responsible for anonymity. The authors of paper [4], propose to use the blind signature protocol. A similar solution is in the paper of Yi Liu and Qi Wang [6]. The blockchain network is used to maintain transparency and the ability to verify the voting procedure by users, and the blind signature protocol enables hiding the identity of voters. The authors divide the participants of the voting process into three groups: voters, organizers and inspectors. Organizers verify voters and are responsible for collecting ballots. Inspectors are limiting the control of the organizers by additional interaction with the voters in the voting process. The system is resistant to ballot manipulation and forgery. As long as there is one honest organizer or inspector, an attack will not succeed. In the paper [17] by Qixuan Zhang, Bowen Xu,

Haotian Jing and Zeyu Zheng, the authors presented a solution that is a simplified version of the previous paper. The system has only two types of users: voters and voting organizer. Each vote has only one organizer. The Ques-Chain smart contract acts as an inspector. It has a public key, a judge function to verify ballots and a ballot box to store valid votes. Our research also uses a blind signatures protocol and is improvement due to Ques-Chai work. In addition, the system was based on the Ethereum network due to the use of smart contracts.

3. E-VOTING SCHEME

Voting scheme starts with voters list preparation. The organizer prepares a list of public keys of cryptocurrency wallets of users who will be entitled to participate in the vote. This is an activity that is not performed in the scheme and the method of user verification depends on the organizer and the type of voting. The user list will be needed when verifying user blinded ballot, therefore this process can be done until then.

```
1 await contract.methods.addVoteOption.cacheSend(name.trim(), {
2     from: drizzleState.accounts[0],
3 });
```

Listing 1. Adding vote option

Second step is generating keys by organizer using RSA algorithm. Before the organizer starts voting, the public and private key must be generated. The keys will be used for digital signatures of ballot and for their subsequent verification.

Third step is vote creation. The organizer can create a vote by creating a list of voting options. Candidates are signed up on a smart contract, which makes them visible to all users of both applications.

Fourth step is Sending a ballot for verification. The user selects an option to vote and votes. The ballot is then encrypted with the public key. The user receives a scrambled ballot and a blinding factor that will be needed later to decrypt the signed ballot. The next step is to send your ballot to the organizer.

```
const keyE = await Organization.methods.keyE().call();
const keyN = await Organization.methods.keyN().call();

const { blinded, r: keyR } = BlindSignature.blind({
    message: this.state.selectedOption,
    N: keyN,
    E: keyE,
});

const userAddress = drizzleState.accounts[0];

await Organization.methods
    .sendBlindedBallot(userAddress, blinded.toString(), date)
    .send({
        from: userAddress,
    });
```

Listing 1. Ballot signing and sending it to the Organization contract

Fifth step is ballot verification. The organizer must verify if the user is allowed to vote, having access to a list of all sent encrypted ballots. For this purpose, it compares the address from which the ballot was sent with the addresses on the list of users that was created earlier. After positive verification, the ballot is signed by the organizer and sent back to the user.

Sixth step is Sending a ballot. The user, using part of the public key and the blinding factor, can decrypt his ballot already signed by the organizer. The user then changes their wallet address. Thanks to that, the user remains anonymous after sending the ballot.

```
await QuesChain.methods
    .sendBallot(
        drizzleState.accounts[0],
        unblindedBallot,
        localStorage.getItem("voteOption")
    )
    .send({
        from: drizzleState.accounts[0],
    });
```

Listing 2. Sending a ballot to the Ques-Chain contract

Seventh step is results verification. The organizer has the opportunity to count the ballots and share the results. All verified valid ballots will be recorded on the Ques-Chain contract.

```
const verifiedBallots = ballots
    .filter((ballot) =>
        BlindSignature.verify({
            unblinded: ballot.unblindedSignature,
            key,
            message: ballot.content,
        })
    )
    .map((ballot) => ({
        id: ballot.id,
        userAddress: ballot.userAddress,
        unblindedSignature: ballot.unblindedSignature,
        content: ballot.content,
    }));
```

Listing 3. Verifying and filtering ballots

In final step the user has access to all verified and approved ballots on the Ques-Chain contract. It can display the number of ballots cast for each voting option. In addition, he can search for his ballot from the list of ballots taken into account in the election results. To do this, it searches for a ballot in terms of the address from which it was sent.

4. PROOF OF CONCEPT E-VOTING IMPLEMENTATION

The project consists of two separate applications dedicated to two groups of users. First is Voter which is a person taking part in voting. The second is voting organizer which is a person responsible for creating and conducting a vote. The application for voters will be available at the selected internet address. Any user can use the application and vote. Only verified users' ballots will be counted towards the voting results. The application for the voting organizer may be run locally and may not be available from any internet address. Two smart contracts were created for the purpose of voting. First is organization contract - used to create voting and verify users ballots, second is Ques-Chain contract - collects verified users ballots and allows you to check the voting result. The voting organizer can create a new vote by adding voting options to the Organization contract. Then the voter sends his ballot for verification. Also, the verification and return of the signed ballot takes place under the same contract. Only sending the verified ballot is done through the Ques-Chain contract. The organizer has access to them and can count the ballots and share the voting result.

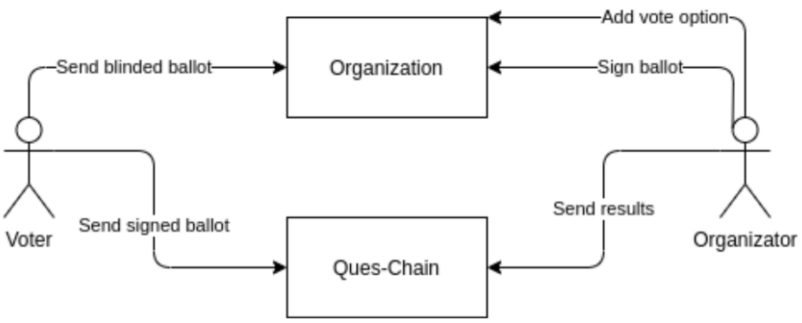

Figure 1. Traffic sample

The Organization contract was written in the 0.5.16 version of Solidity. The contract has three implemented data structures: First one is ballot - a voter's ballot that is created when an option is selected, and the ballot is sent for verification. It has a unique identifier, address of the person sending the ballot, content, i.e., the selected option, date of creation and information whether the ballot has already been verified. Second one is SignedBallot - Verified ballot of the voter, signed by the organizer. It has the same data as a regular voter's ballot, but without verification information, and the content is already encrypted information. Third is VoteOption - A voting option that is added by the voting organizer. It has a unique identifier and name to be displayed to the users of the application. Voting options are stored in the mapping structure. The function of adding options to a contract takes one parameter - the name of the option. The contract stores the organizer's public key data that is needed during the ballot-blinding process. There is one function that sets both used variables. Blinded ballot are held in the mapping structure. The function to send a ballot to a contract has three arguments: the address of the user who sends the ballot, the blind ballot, and the sending date. Ballots verified and signed by the organizer are kept in the mapping structure. The ballot signing function has four arguments: selected ballot identifier, user address, blind ballot, and date of signature.

```
handleSaveButton = async (event) => {
    event.preventDefault();

    const generationDate = moment()
        .format("DD/MM/YYYY HH:mm:ss");

    localStorage.setItem("privateKey", this.state.privateKey);
    localStorage.setItem("keyDate", generationDate);

    const key = new NodeRSA(this.state.privateKey);

    const { drizzle, drizzleState } = this.props;
    const contract = drizzle.contracts.Organization;

    const e = key.keyPair.e.toString();
    const n = key.keyPair.n.toString();

    await contract.methods.setKeyData(e, n).send({
        from: drizzleState.accounts[0],
    });

    this.setState({ saved: true });
};
```

Listing 4. Generating RSA key by organizator

The contract has one voting structure. It contains a unique identifier, user address, unblinded ballot and the selected option. User ballots are stored in the mapping structure. The user sends a ballot by specifying three parameters: user's address, unblinded signature and the selected option in voting. The voting results are serialized to a string variable. The share result function takes only one argument. The "Private" key page is used to generate the RSA key. The key can be generated using the PEM string. Above the field for entering the private key, the date of generating the current key is displayed. The date is generated and formatted using the moment library. The date and private key are saved in local memory. An RSA key is then generated using

the key entered. The two components of the RSA key (e and n) will be used to encrypt the ballots. Therefore, they are included in the Organization contract.

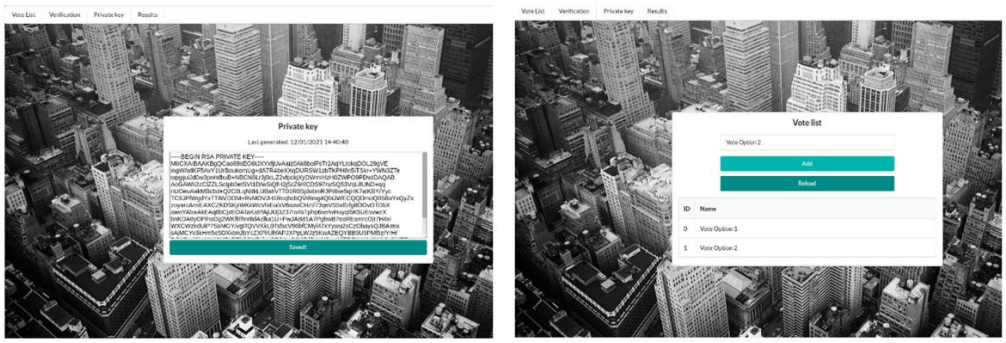

Figure 2. GUI of implemented solution. Page to set RSA key and Voting list

Adding a new voting option requires an Ethereum transaction. For this purpose, a wallet in the MetaMask application was used. There is a transaction fee to complete the transaction. When you press the "Add" button, the current voting options from the contract are downloaded and then checked if the option name is unique. If the option name is not unique, an error message will be displayed. Otherwise, the name is sent to the contract and a new voting option is added to the list. If the list does not contain any options from the beginning, a message informing about it is displayed. The "Reload" button is used to refresh the list after adding a new voting option. The data is taken from the Organization contract. The "Verification" page is used to confirm the identity of voters and sign their encrypted ballots. The "Reload" button is used to download the entire list of ballots requiring confirmation. Each record has information about the wallet address from which the transaction of sending the ballot was made and the transaction date. After verifying the address, the organizer can choose a ballot and sign it. The ballot is then added to the list of verified ballots on the Organization's contract. When the "Count votes" button is pressed, all ballots from the Ques-Chain contract are downloaded. Then they are verified, i.e. the decrypted ballot is compared with the selected voting option. All verified ballots are added up and listed in the scoreboard. The organizer can publish the results after clicking the "Send results" button.

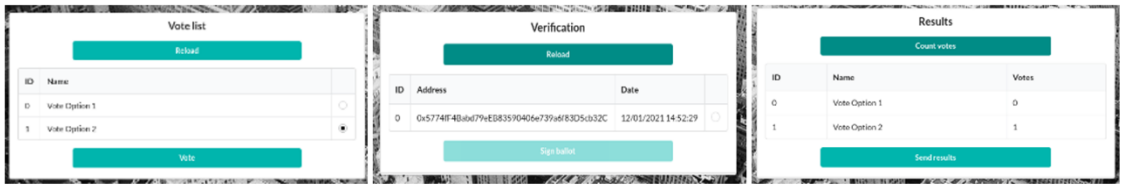

Figure 3. GUI of implemented solution. sending a ballot, ballot verification and vote results with verified ballot.

As for the voter implementation. The user can view the current voting list that is available on the Organization contract. After selecting an option from the list and pressing the "Vote" button, the ballot is encrypted and sent for verification. The "Signature" page displays information about the user's ballot verification process. When the ballot is verified by the organizer, the information will be displayed along with the exact approval date. Additionally, the verified ballot is checked if it has not been counterfeited and still holds information about the same ballot.

If the organizer has made the results available, the user can view them. The result table contains information about the name of a voting option and the total number of ballots cast for that option. In addition, the user can check if his ballot is on the list of ballots that make up the score. If the ballot has been counted, its data is displayed. Thanks to this, the user can see if his ballot is cast for the option chosen by him.

5. CONCLUSIONS

The research required an analysis of existing solutions in the field of anonymity and blockchain technology. The research also concerned theoretical works that had not yet been implemented. The result of the analysis was the creation of an architecture that was based on the solution proposed in the paper [17]. During creation of the paper, a blockchain-based voting scheme was presented and proof of concept application was developed. It uses a blind signature protocol to encrypt messages and keep voter anonymity.

Currently, a transaction fee is payable when making a transaction. If this is not a problem for the voting organizer, the user should not be charged with an additional fee for using this service. In addition, the user would have to have Ether on two wallets, which makes the voting process very difficult. The solution to this problem would be to transfer the obligation to pay the transaction cost to the voting organizer.

The application requires the user to send the verified ballot from one wallet and then send the verified ballot from the other wallet. The goal is to have a different address so that the user remains anonymous. The application does not automate this process and it is the user's responsibility to change the address. The application could use MetaMask to simplify this process.

The implemented ballot encryption protocol uses RSA encryption. Only the public key and the selected voting option are needed for the blinding process. The ballot encrypted in this way can be easily decrypted. Just compare it with the encryption results of all voting options. It would be necessary to introduce an additional method of securing an encrypted message.

The application allows only one election. Additionally, it does not provide the ability to edit voting options before publishing them. Currently, the organizer ends the voting when the ballots are counted, and the results are made available. The application should allow the creation of many different votes by different organizations for commercial use. Also, a system should be implemented that prevents voting after a given time. The results would be made available automatically without the intervention of the organizer.

The application fully implements the proposed architecture and enables voting. However, in order to be able to implement this system, many improvements for users should be introduced. These improvements, possible fixes and changes are listed below. The application requires some changes to be commercially applicable. Many of them facilitate the use of the application for users or adapt the use of Ethereum transactions to reduce the requirements for the user.

REFERENCES

[1] Li Y. Alvarez R., Levin I. Fraud, convenience, and e-voting: how voting experience shapesopinions about voting technology, May 2018.
[2] Chaum D. Blind signatures for untraceable payments, 1983.
[3] Maaten E. Towards remote e-voting: Estonian case, January 2004.

[4] Akram R. Markantonakis K. Hardwick F., Gioulis A. E-voting with blockchain: An e-votingprotocol with decentralisation and voter privacy, July 2018.
[5] Chowdhury M. Jabed M. Comparison of e-voting schemes: Estonian and norwegian solutions,September 2013.
[6] Wang Q. Liu Y. An e-voting protocol based on blockchain, 2017.
[7] Embele K. Ndu A. Awodola O. Mawutor J., Enofe A. Fraud and performance of deposit moneybanks, May 2019.
[8] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system, October 2008.
[9] Chen M. Jia C. Liu C. Wang Z. Shao W., Li H. Identifying bitcoin users using deep neural network,December 2018.
[10] KSHETRI, Nir; VOAS, Jeffrey. Blockchain-enabled e-voting. IEEE Software, 2018, 35.4: 95-99.
[11] Specter, Michael A., James Koppel, and Daniel Weitzner. "The ballot is busted before the blockchain: A security analysis of voatz, the first internet voting application used in us federal elections." 29th {USENIX} Security Symposium ({USENIX} Security 20). 2020.
[12] Park, Sunoo, et al. "Going from bad to worse: from internet voting to blockchain voting." Journal of Cybersecurity 7.1 (2021):
[13] Kedziora, Michal, and Wojciech Wojtysiak. "Practical Analysis of Traceability Problem in Monero's Blockchain." ENASE. 2020.
[14] Yang, Xuechao, et al. "Blockchain voting: Publicly verifiable online voting protocol without trusted tallying authorities." Future Generation Computer Systems 112 (2020): 859-874.
[15] Trojanowska, Natalia, et al. "Secure Decentralized Application Development of Blockchain-based Games." 2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC). IEEE, 2020.
[16] Lam, Peter. "From Helios to Voatz: Blockchain Voting and the Vulnerabilities It Opens For The Future.", 2017
[17] Jing H. Zheng Z. Zhang Q., Xu B. Ques-chain: an ethereum based e-voting system, May 2019
[18] Kedziora, Michal, et al. "Anti-Cheat tool for detecting unauthorized user interference in the unity engine using Blockchain." Data-Centric Business and Applications. Springer, Cham, 2020. 191-209.
[19] Kedziora, Michal, Patryk Kozlowski, and Piotr Jozwiak. "Security of Blockchain Distributed Ledger Consensus Mechanism in Context of the Sybil Attack." International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer, Cham, 2020.
[20] Dimitriou, Tassos. "Efficient, coercion-free and universally verifiable blockchain-based voting." Computer Networks 174 (2020): 107234.

© 2021 By AIRCC Publishing Corporation. This article is published under the Creative Commons Attribution (CC BY) license.