Blockchain Based E-voting System: A Proposal

Completed Research

Jorge Lopes
Mestrado Integrado em Engenharia e Gestão de Sistemas de Informação, Universidade do Minho, Portugal
a73263@alunos.uminho.pt

José Luís Pereira
Departamento de Sistemas de Informação e Centro ALGORITMI, Universidade do Minho, Portugal
jlmp@dsi.uminho.pt

João Varajão
Departamento de Sistemas de Informação e Centro ALGORITMI, Universidade do Minho, Portugal
varajao@dsi.uminho.pt

Abstract

Blockchain is a distributed ledger technology that allows digital assets to be transacted in a peer-to-peer decentralized network. Those transactions are verified and registered by every node of the network, thus creating a transparent and immutable sequence of registered events whose veracity is provided by a consensus protocol. By enabling smart contracts to be deployed into a blockchain platform, the number of possible use cases for this technology improves considerably. Eliminating the need for third parties and, therefore, allowing many processes, in both the public and the private sectors, to become more efficient and economical. Electronic voting systems are one example of a use case that can be improved by the blockchain technology. In this paper, we discuss how blockchain technology can solve major electronic voting systems challenges. Additionally, the components and functionalities of our proposed blockchain based electronic voting system are presented and explained.

Keywords

Blockchain, Smart Contract, Voting, Electronic Voting System.

Introduction

Blockchain technology is an emergent and disruptive technology, which as the potential to improve the way individuals and organizations interact and operate, ultimately influencing almost all facets of our daily lives. Nowadays, besides the financial sector in which the blockchain concept was born, we are witnessing substantial efforts made by many organizations, from distinct economic sectors, which are using blockchain technology to develop very innovative applications (Lopes and Pereira 2019).

In simple words, the core of this technology is the use of a distributed and decentralized ledger for verifying and recording transactions, while allowing parties to send, receive, and record value or information through a peer-to-peer network of computers. A blockchain can be categorized into three different types, being them public, private and consortium. Their differences reside, basically, on the user's restrictions to interact with the network (Buterin 2015).

Recently, several blockchain platforms have evolved in order to support smart contracts. These are transactions or contracts converted into computer code that facilitate, execute and enforce agreements between two or more parties, having the potential to improve financial transactions and operational and counterparty risk associated with monitoring or enforcing contractual obligations.

The blockchain concept, enriched with smart contracts, has a very wide range of applications. In this paper, we propose a blockchain-based electronic voting system, which has the potential to solve the major limitations and issues associated with voting systems.

Regarding the structure of the paper, first, we very briefly review some basic concepts related to this technology. In the following, we describe the major requirements of an electronic voting system and the advantages of using blockchain technology to implement such a system. With those aspects in mind, we describe in detail the solution we propose. Finally, we draw some conclusions regarding the utilization of blockchain technology in the specific context of voting systems.

Blockchain Technology

Blockchain can be described as a distributed data structure that is replicated and shared among the members of a network, whose purpose is to record every transaction done. Each transaction is batched into timestamped blocks and each block is identified by its cryptographic hash. Each block stores the hash of the previous one, creating a link between the blocks or, as the name implies, a chain of blocks, thus generating a transparent and immutable history of records whose veracity is provided by a consensus protocol (Christidis and Devetsikiotis 2016).

Blocks are composed of a list of transactions recorded over a given period and a header (see Figure 1). The header contains: the previous Hash, which is the result of encrypting all data found on the previous block header; the nonce, which is a value adjusted by miners so that the hash of the block generated will be lower or equal to the current target of the network; finally, the Merkle tree root, which summarizes all the transactions in the block, making it possible to verify if a transaction is included or not (Jiang, Cao, Wu, Yang, Ma and He 2018).

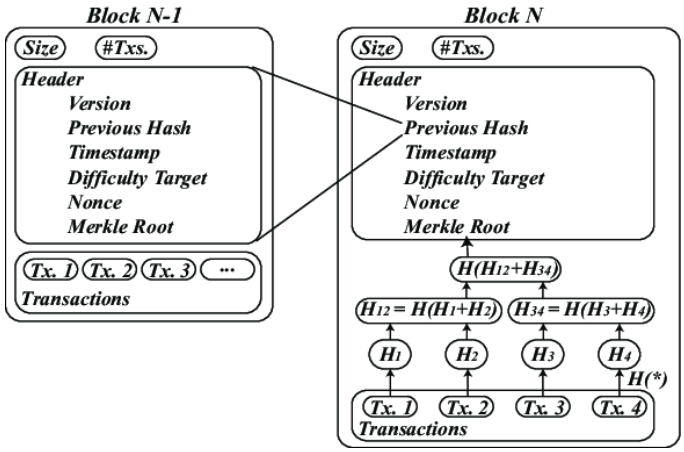

Figure 1: The Structure of a Block (Jiang et al. 2018)

Originally, blockchain was created in order to solve the double-spending problem, which means the result of successfully spending the same crypto or token more than once (Ofir 2018). Transactions represent transfer of value in a specific cryptocurrency or token. After a suitable confirmation, the transaction is included in a block and validated. In order to make a transaction a user needs the address (public key) of the receiver wallet and his/her own private key, a digital signature only known by the owner of that wallet (Christidis and Devetsikiotis 2016).

A blockchain is a linked list, linked by hash pointers, where each block contains a hash pointer that stores a cryptographic hash of the header of the previous block in the chain. If the content of a block in the chain is modified, the hash pointer stored on the next block will also change as well as all subsequent blocks in the chain ("E-learning Spot" n.d.). Therefore, a hash pointer provides the blockchain with a tamper-evident system in a simple way. For example, if someone changes the contents of one block the hash of the next block will not mash up and we will notice the inconsistency. Even if an attacker has enough computer power to change the block and all the hash pointers of the following blocks in the chain, he/she will arrive at a

dead-end because the last hash pointer is the value we remember as being the head of the list and the inconsistency will be noticed inevitably.

Consensus Protocols

The consensus protocol is the mechanism that allows a decentralized network to arrive at an agreement about the state of the blockchain and forces all nodes to behave accordingly to the network principles. By using a consensus protocol the blockchain eliminates the need to trust or rely upon a centralized authority (Ouattara, Ahmat, Ouédraogo, Bissyandé and Sié 2018).

The protocol rewards the community for properly maintaining the consensus, allowing greater recording and processing power in an incentive and typically competitive manner. The goal is to incentivize responsible and accurate recordkeeping, while reducing tampering.

Nowadays, two of the most commonnly used consensus protocols, exhibit some strengths but also suffer from important weaknesses (Baliga 2017):

- Proof-of-Work (PoW): rewards users who solve complicated cryptographical puzzles in order to validate transactions and create new blocks. The participants who compete to solve this puzzle are known as "miners". Currently, PoW is the most used consensus protocol, but it has some major drawbacks such as the need for a large amount of resources, the vulnerability to 51% attacks, etc.;
- Proof-of-Stake (PoS): forces the participants to compete not by computing power but with cryptocurrencies instead. The PoS algorithm randomly selects a validator for the block creation. The participants can increase their chances to be the one validating the block by increasing the amount of crypto they put on stake. PoS is a solution to the PoW energy consumption and computing power problem. Like the PoW centralization risk, this is also a possibility in PoS, where the richest stakeholders can have control of the consensus in the blockchain.

There are many other concensus mechanisms who try to address some limitations of the above mentioned ones, such as Delegated Proof-of-Stake (DPoS), Practical Byzantine Fault Tolerance (PBFT), Proof of Elapsed Time (PoET), among others (Baliga 2017).

Smart Contracts

A smart contract is like a program, which runs on the blockchain and has its correct execution enforced by the consensus protocol. A smart contract can encode any set of rules represented in its programming language. For instance, it can execute transfers when certain events occur. Accordingly, smart contracts can implement a wide range of applications, including financial instruments and self-enforcing or autonomous governance (Christidis and Devetsikiotis 2016).

A smart contract is linked to an account, it is identified by an address and its code resides on the blockchain. Once a contract is uploaded to the blockchain, it will react accordingly to the code implemented previously and is able to execute transactions like a normal user of the blockchain (Luu, Chu, Olickel, Saxena, and Hobor 2016).

The main objectives are to satisfy common contractual conditions, minimize both malicious and accidental events, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitrations and enforcement costs.

Electronic Voting Systems

Voting is a very crucial activity for any institution or organization that needs to elect a certain entity for a certain role. Historically, the most common way to vote, still largely used nowadays, is through a paper-based system. Of course, this is not the safest nor the most convenient or economical method to support the voting process.

Electronic voting systems might be the solution to the disadvantages mentioned above. The first electronic voting system was introduced in the early eighties (Ayed 2017). Since then, the method of voting has evolved

over time to improve its speed, security, flexibility, availability and cost during its three main steps, being them the authentication of the voter, the casting of the vote and lastly, the exposing of the results.

Requirements of Electronic-voting Systems

To judge the adequacy and security parameters of a voting system it is necessary to evaluate twelve core requirements (Table 1). These requirements are presented by order of relevance, from the most important to the least important (Nogueira and Sá-Soares 2012).

Requirement	**Description**
Authenticity	Only users with the right to vote should be able to cast a vote
Singularity	Each voter should be able to vote only once
Anonymity	It should not be possible to associate a vote to a voter
Integrity	Votes should not be able to be modified or destroyed
Uncoercability	No voter should be able to prove the vote that he/she has casted
Verifiability	Anyone should be able to independently verify that all votes have been correctly counted
Auditability and Certifiability	Voting systems should be able to be tested, audited and certifiable by independent agents
Mobility	Voting systems should not restrict the voting place
Transparency	Voting systems should be clear and transmit accuracy, precision, and security to voter
Availability	Voting systems should be always available during the voting period
Accessibility and Convenience	Voting systems should be accessible by people with special needs and without requiring specific equipment or abilities
Detectability and Recoverability	Voting systems should detect errors, faults and attacks and recover voting information to the point of failure

Table 1: E-voting Systems Requirements

From the order of importance shown above, it is possible to observe that the requirements with higher impact involve the security and anonymity of the voting system, while the requirements with lower impact encompass the usability and transparency of the voting infrastructure.

Benefits of Blockchain regarding E-voting Systems

Blockchain has the potential to improve electronic voting systems by solving some of their major limitations and issues. Next, each e-voting systems requirement is addressed in the same order as presented in Table 1, in order to verify if a blockchain solution might fulfil them.

- Authenticity: Every user of the network is identified by a public key, which can only be accessed by its own private key. Assuming that every voter will keep its own private key secret, then the authenticity requirement is fulfilled;
- Singularity: Every vote cast is linked to the public key of the voter on the blockchain. By allowing each public key to cast only one vote, the singularity requirement is assured;
- Anonymity: since every user is identified by a public key and the stored vote is encrypted, it is impossible to associate a voter with a vote;
- Integrity: since the hash pointer provides the blockchain with tamper-evident properties, every stored vote has the same properties, meaning that it can't be adulterated;

- Uncoercability: Every vote cast will be encrypted before being stored on the blockchain, making it impossible for anyone to know the content of that vote;
- Verifiability: since the blockchain is transparent to every node of the network, everyone could confirm that the number of votes casted and counted are the same;
- Auditability and Certifiability: equally to the verifiability requirement, the transparence property of blockchain allow for any node in the network to audit the blockchain. Since the system code is open source and visible on the blockchain, means that the used application could be also audited;
- Mobility: the only requirements to access the network is a device with internet connection and an address in the blockchain platform, meaning that it is not required any kind of special infrastructure or voting machines;
- Transparency: identically to the Auditability and Certifiability requirement, transparency is one of the blockchain properties, and every application implemented in the blockchain inherits this same property;
- Availability: since blockchain is a distributed network, as long as the network has the required number of nodes to achieve consensus, the voting system will be always available;
- Accessibility and Convenience: equal to the mobility requirement, blockchain does not require any kind of infrastructure or voting machines. Only a device with an internet connection is required;
- Detectability and Recoverability: If any malicious action is made on the blockchain, it will be detected by the system and considered invalid, which fulfils the detectability requirement. In the case of recoverability, as soon as some data is stored on the blockchain, no longer can be deleted, meaning that data can always be recovered.

Regarding uncoercability and verifiability, while they might seem impossible to be obtained simultaneously, (in the sense that if the content of a vote stored in the blockchain cannot be known, then the election results would be impossible to verify afterwards) the fact is the smart contract code used to update the results during the election process is visible and auditable. Thus, it can be certified as producing correct results. Indeed, a blockchain-based voting solution may satisfy every e-voting requirement, thus having the potential to improve our current e-voting systems.

Proposed Solution

This section addresses the voting solution we are proposing, describing the chosen blockchain platform; the architecture of the system; the smart contracts that were deployed on the blockchain; the interface, API and functionalities that are implemented; and lastly, it describes the encryption server, explaining how it can ensure the privacy of the vote.

Blockchain Platform

The chosen blockchain platform was Ethereum (Ethereum n.d.). The advantages of this blockchain platform include a suitable protocol for developing decentralized applications, by building a blockchain with a built-in turing-complete programming language (Solidity), allowing users to write smart contracts with their own rules for transactions formats, rules for ownership and state transitions functions. Moreover, Ethereum is a public blockchain which philosophy is based on principles that are considered ideal to electronic voting systems.

Decentralized Application

This implementation consists of an HTML interface for the application users, a cryptographic server that will encrypt/decrypt the votes, three contracts deployed on the Ethereum blockchain that are coded in the Solidity language and an API to act as a bridge between all the components mentioned before. The architecture of the system is represented in Figure 2.

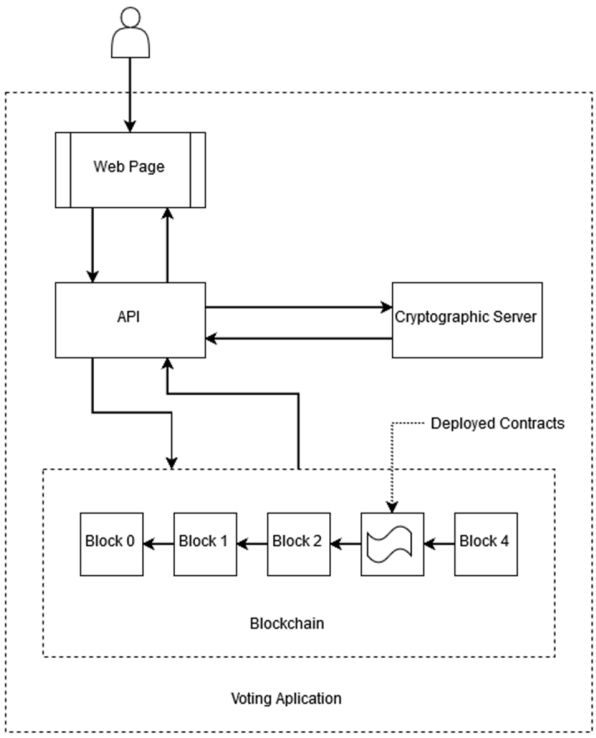

Figure 2: Voting Application Architecture

Users

There are three types of users who can interact with the application, being them the administrator, the creator and the voter. The administrator has the responsibility for the initial deployment of the contracts and grants or revokes the permission for a user to create ballots. The creator is a user who was granted permission for creating new ballots. The voter, as the name indicates, is the user who can vote for a candidate in a certain ballot.

Interface and API

The user interface is a simple HTML page that allows users to access the functionalities of the application. The API is responsible to react to actions made on the interface and interact with the encryption server and the blockchain.

For each request made in the interface, it will interact with the encryption server by server calls to encrypt, decrypt or add votes. To interact with the blockchain, transactions or web3.eth.calls are used, in order to store or retrieve information, respectively.

The web3.eth.call executes a message call transaction, which is directly executed in the virtual machine of the node, but it is never mined into the blockchain, by doing so it is possible to retrieve information from the blockchain without paying startgaz.

Contracts

In order to reach a higher performance level, the application functionalities are divided into three contracts, being them Record, Creator and Election. Each one with a specific purpose, but all of them working together to achieve the same goal. In Figure 3, it is possible to analyze the memory field structure of each smart contract, the lines between fields represent relational data.

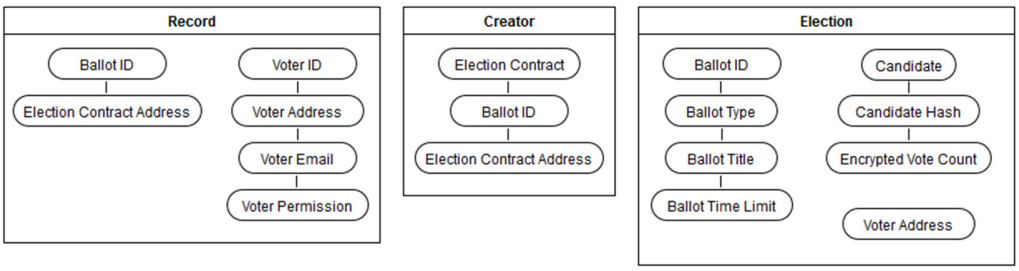

Figure 3: Contracts Information Flow

Record: this contract will store information related to every voter registered and ballots created, being the owner of this contract the administrator. When deploying this contract, the administrator needs to specify the user's email domains that have permission to register. Posteriorly, he/she can add new domains if required. Every user who is able to register on the system will have his/her information stored on this contract, this is necessary for validation functions. When registering the user can require creating permissions. If granted by the administrator, the user will be able to create new ballots, this means a new Election contract. For every ballot created, its ID and Ethereum address are linked and stored.

Creator: this contract will spawn new Election contracts when required by a user with permissions to do it. The parameters of the new ballot need to be delivered by the creator. The owner of this contract is the administrator.

Election: this contract represent an election, storing its parameters, candidates and votes. The owner of this contract is its creator. When deploying this contract, the creator needs to specify all the parameters regarding the election contract, being them the ID, the type, the deadline and the title of the ballot. After introducing and validating the candidates, assuming that the deadline was not exceeded, it is possible to vote. First, the voter information will be verified and, if valid, the vote will be encrypted, and the current vote count updated. When reached the deadline, in case of an election type, the vote count is decrypted and the result revealed. While on a poll type, the current result is always available.

Functionalities

Regarding the functionalities that are available to users, the use case diagram in Figure 4 represents which users can access each one of the functionalities. In the following, some details about them:

Register Voter: For a user to be able to register it is required for him/her to insert on the interface his/her email, institutional number and inform if he/she demands creator permission. The API will receive this information from the interface and will send an eth.call to the Record contract to verify the user information by checking if the domain of the email is equal to the one specified when the contract was deployed and if the user was not already registered. If all verifications are validated, the user will receive a notification on the interface and the API will send a transaction to the contract to register the new voter and store his/her information;

Create Ballot: In order to create a new ballot a user, with permissions to do it, has to insert his/her email and the parameters of the new ballot in the interface. It is necessary to specify the new ballot title, the deadline date and time, the candidates and if it will be a poll or an election. In the first option, the number of votes of each candidate is always available, while on the second option the results will be available after the deadline imposed by the creator. After submitting this information, the interface sends it to the API that proceeds by sending two eth.call to the Record contract to verify if the creator email and Ethereum address are valid. If so, the API sends a third eth.call to verify if the user has permissions to create ballots. After confirming that all verifications are valid, the API sends a transaction to the Creator contract with a request to create a new Election contract with the parameters previously inserted by the creator along with a new 32-bit ballot ID provided by a random number generator, replicated to all nodes. Once the new Election contract has been deployed it returns its address to the Creator contract. Next, the API sends another eth.call to the Creator

contract to retrieve the address of the Election contract that has been deployed and sends a transaction to the Record contract to store that address along with the corresponding ballot ID. After this process is completed the ballot ID is displayed on the interface, which must be written down by the creator and shared with the users who want to vote on that ballot;

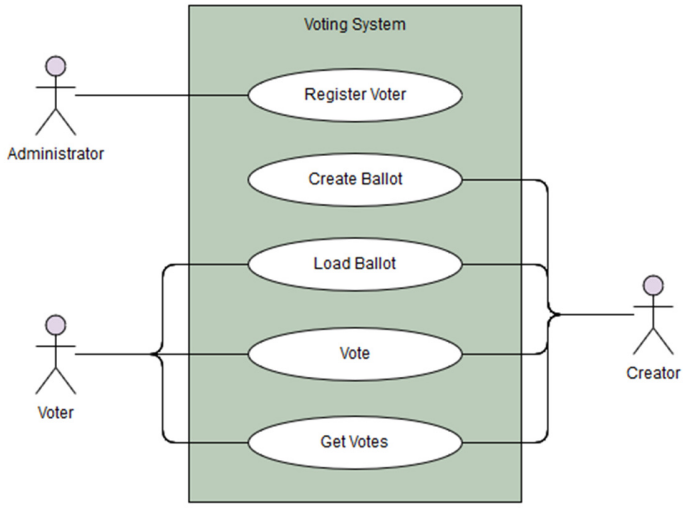

Figure 4: Blockchain Based E-voting System Functionalities

Load Ballot: By inserting the ballot ID provided by the Election contract creator on the interface, the API sends an eth.call to the Record contract to verify if the ID is valid. If so, it loads the respective ballot and show the title, the name of the candidates and the respective votes if the ballot type was poll, if not, the votes will remain blank until the deadline. In order to present the number of votes of each candidate, the API needs to send a request to the Encryption server to decrypt the vote count;

Vote: After loading the ballot, the user can vote by providing his/her email and choosing one of the available candidates. The API receives this information from the interface and sends an eth.call to the Record contract to verify if the voter is valid. If the voter passes the verification it sends an eth.call to the Election contract to verify if the chosen candidate is valid and if the vote was done before the ballot deadline. If the current block timestamp is greater than the deadline the vote is not considered valid. If all verifications are validated the vote is stored as an array, being 1 for the chosen candidate and 0 for the remaining options. The vote is then sent to the Encryption server to be encrypted. As soon as the vote has been encrypted, the previously encrypted vote count will be retrieved from the Election contract by the API using an eth.call and sent to the Encryption server to be homomorphically added together, resulting in the new encrypted vote count;

Get Votes: Every time a voter loads the ballot or votes the API sends an eth.call with the hashed candidates to retrieve the current encrypted vote count. Depending on the deadline and if it is a poll or an election, it would either decrypt the votes and display them or it will wait for the end of the deadline.

Encryption Server

In order to ensure the privacy of the votes, it is necessary to prevent unauthorized access to the votes. To achieve that, each vote needs to be encrypted before being transacted to the blockchain. The definition of homomorphism is a structure-preserving map between two algebraic structures. Homomorphic encryption is a public key encryption scheme with properties that allow specific types of computations to be carried out on ciphertext, generating an encrypted result, when decrypted the result will be equal to perform the same operation on the plaintext. Making possible to operate on messages without ever releasing their content (Yi, Paulet, and Bertino 2014). The Paillier scheme inherits the same properties as the Homomorphic encryption, such as the homomorphic addition and multiplication of plaintexts. These properties make possible to manage votes without revealing it.

Other E-voting Systems Proposals

A review of the scientific literature regarding blockchain based e-voting systems reveals several efforts already done in this area. For instance, authors such as (McCorry, Shahandashti and Hao 2017) have implemented an e-voting application using smart contracts, called The Open Vote Network (OVN), which they claim is a voting protocol with maximum voter privacy. The OVN was developed using smart contracts for the Ethereum blockchain. In a similar effort, Alexander, Landers and Howerton (2018), developed a blockchain-based e-voting platform, also over the Ethereum blockchain, which they called Netvote. In Netvote, each election is represented by a set of smart contracts which are deployed on the blockchain by the so-called election administrators.

Example Application

To serve as an example application of our proposal a prototype of a blockchain based E-voting system has been developed. The developed system provides an easy and intuitive E-voting platform designed to satisfy the requirements of a very simple context (elections in our own institution), by enabling the creation of multiple Election contracts with different characteristics.

The prototype aims to simplify all the processes involved in performing an election. Thus, providing the opportunity to decrease the costs of infrastructure and maintenance, decreasing the amount of time required to organize an election, improve its security by benefiting from the blockchain properties and, lastly, potentially increasing voters participation by using a solution that can be accessed from everywhere and which requires a minimum effort to vote.

Future Improvements

While it is a functional prototype, in order for our solution to be used in real election scenarios there are some concerns that need to be taken into account. First of all, to prevent the possibility of fraud or incorrect data being delivered to the contract, the process of registering needs to ensure that the institutional email belongs to the person who is registering. The same is true with the processes of validating a request to create an election, as it needs to be conducted by someone with permissions to do so. These actions are crucial to reduce the possibility of occurring the so-called "oracle problem". Secondly, to cast a vote, a user has to possess a certain amount of Ether, in order to pay for the transaction gaz. As a possible solution, the creator of the election must be able to initially send the required amount for the intended users. With this transaction, it can be attached a data field with the ballot ID of the respective Election, creating a more secure and dynamic process of sharing the ballot ID.

Accessing the Application Code

The developed application project can be found on the following GitHub repositorium along with a "read me" file specifying all the dependencies and tools and explaining how to run it on a local host machine.

https://github.com/Jorge-Lopes/Blockchain_E-voting_Proposal

Conclusion

Blockchain is an emergent technology that is evolving on a daily basis. Similarly to other new technologies, blockchain adoption by organizations will evolve step by step, through small efforts, some failures, many successes and, hopefully, widespread adoption. The challenge is to assure the right balance between ensuring the governance, safety and resilience of the blockchain developed solutions while not infringing the innovation and development of this fast-evolving technology.

In this paper we aim to contribute to the consolidation of the blockchain applications ecosystem by developing a blockchain-based electronic voting platform. It is widely recognized that electronic voting solutions have the potential to increase the level of participation in elections, due to the convenience of remote voting, while simultaneously reducing costs by eliminating the need for ballot printing or electronic voting machines purchase and maintenance. By using blockchain technology in the development of an

electronic voting solution we expect to enhance the security of those solutions and provide a transparent and auditable electronic voting platform.

With the project coming to its end, it is already possible to conclude that blockchain has the potential to improve electronic voting systems by inherently solving their major limitations and issues. While our developed electronic voting solution remains only a functional prototype, the trials it was subjected to allow us to conclude about the viability of the use of blockchain technology in the development of this kind of systems. In the future, we intend to test our blockchain-based e-voting system in real conditions in order to evaluate its performance under heavy loads.

Acknowledgements

The authors would like to thank the editors and reviewers for their helpful comments and suggestions.

This work has been supported by FCT - Fundação para a Ciência e Tecnologia, within the Project Scope: UID/CEC/00319/2019.

REFERENCES

Alexander, J., Landers, S., and Howerton, B. 2018. "Netvote: A Decentralized Voting Network," Retrieved from https://netvote. io/wp-content/uploads/2018/02/Netvote-White-Paper-v7.pdf

Ayed, A. B. 2017. "A Conceptual Secure Blockchain-based Electronic Voting System," *International Journal of Network Security & Its Applications* (9:3), pp. 1–9.

Baliga, A. 2017. "Understanding Blockchain Consensus Models," Retrieved from https://www.persistent.com/wp-content/uploads/2017/04/WP-Understanding-Blockchain-Consensus-Models.pdf

Buterin, V. 2015. "On Public and Private Blockchains." Retrieved October 10, 2018, from https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/

Christidis, K., and Devetsikiotis, M. 2016. "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, 4, pp. 2292–2303.

E-learning Spot. (n.d.). "Hash Pointers and Data Structures," Retrieved October 11, 2018, from http://learningspot.altervista.org/hash-pointers-and-data-structures/

Ethereum. (n.d.). "A Next-Generation Smart Contract and Decentralized Application Platform," Retrieved November 8, 2018, from https://github.com/ethereum/wiki/wiki/White-Paper

Jiang, S., Cao, J., Wu, H., Yang, Y., Ma, M., and He, J. 2018. "BlocHIE: A BLOCkchain-Based Platform for Healthcare Information Exchange," in *Proceedings of the 2018 IEEE International Conference on Smart Computing,* pp. 49–56.

Lopes, J., and Pereira, J.L. 2019. "Blockchain Projects Ecosystem: A Review of Current Technical and Legal Challenges," in *Proceedings of the 7th World Conference on Information Systems and Technologies*, AISC 931, pp. 83–92.

Luu, L., Chu, D.-H., Olickel, H., Saxena, P., and Hobor, A. 2016. "Making Smart Contracts Smarter," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 254–269.

McCorry P., Shahandashti S.F., and Hao F. 2017. "A Smart Contract for Boardroom Voting with Maximum Voter Privacy," in *Financial Cryptography and Data Security*, : Kiayias A. (eds) Lecture Notes in Computer Science, vol 10322. Springer, Cham

Ofir, B. 2018. "What is Double Spending?" Retrieved November 7, 2018, from https://99bitcoins.com/double-spending/

Ouattara, H. F., Ahmat, D., Ouédraogo, F. T., Bissyandé, T. F., and Sié, O. 2018. "Blockchain Consensus Protocols," in *e-Infrastructure and e-Services for Developing Countries*, V. Odumuyiwa, O. Adegboyega and C. Uwadia (eds.), pp. 304–314.

Nogueira, J., and Sá-Soares, F. 2012. "Trust in e-voting systems: A case study," in Lecture Notes in Business Information Processing, Vol. 129, pp. 51–66.

Yi, X., Paulet, R., and Bertino, E. 2014. *Homomorphic Encryption and Applications*, Springer International Publishing.