Matthew Dippel | Spring 2022 | Capstone 1 | Dr. Bogert | Dr. Sanft

11/6/21 - Draft 0
4/10/22 - Draft 1

Ethereum E-voting System

Abstract

The aim of this project is to create a scalable, blockchain-based, auditable, private and secure voting system. Crucial to this is the public smart contract, deployed on the Ethereum blockchain. This is ultimately what allows a vote to be auditable, private and secure, because all vote counting happens on a public resource, and all vote casting uses digital signatures for authentication and authorization. Each vote is represented as a transaction on a public ledger, as such, the origin and contents of this transaction can be verified by the one who cast the vote. Working with this smart contract is a client app, which allows users to register themselves and cast their vote. In addition, a voting administrator companion app allows someone to conduct the registration and voting processes. In the near term, this system may be appropriate for use in local elections and referendums. Looking far further into the future, this project has an enormous potential to transform the democratic process worldwide. Ideally, in a return to direct democracy, where and when this makes sense. From this it follows that some of the goals of this project cannot be separated totally from politics, thus this connection is deserving of some critical analysis. This discussion of politics should not distract from the project itself, as it is certainly apolitical in nature. Its only purpose is to provide user auditability, and inspire increased public trust in the democratic process, something which currently stands at an all time low. Ultimately, politics itself will have to deal with its own problems before this kind of voting system could ever hope to attain wide usage.

Introduction

Voting is an important part of modern civilization. Without a trustworthy voting system, democracy becomes farcical. It is a fact that paper ballot voting systems have many advantages and a robust track record that is hard to top. They also contain aspects that are difficult to recreate in electronic form. Promisingly, recent advancements in cryptographic techniques as well as data structure design and implementation offer a possible electronic and online solution with advantages over the legacy paper ballot system. Unlike the paper ballot system, a blockchain-based system allows each voter to cryptographically verify that their own vote was counted accurately and that the final tally is accurate. Yet one unsolved issue remains in this setup: threat actors registering under multiple aliases; in effect, “ballot stuffing.” But, this is just as much a problem in general elections, extremely so in “poorer” nations, and increasingly an issue in “richer” nations. In addition, the code is open source, which is a security advantage. In a paper ballot system, the voter must trust the ones who are counting the votes, along with all others with ties to the process itself and governmental politics as a whole. In this new system,

some trust must still be placed in the one who is administering the vote, but because the results can be thoroughly verified, this is not as much of an issue. Trust must also be placed in the Ethereum blockchain itself. As a proof-of-concept, this application is not suitable for large elections. In the near term, this application could be suitable for local referendums or the like. However, since this prototype runs on an Android device, it is inherently less secure. Ultimately, this application would need to be run on devices similar to today's voting machines.

In order to understand how this voting system works, it is necessary to give a primer on blockchain. This system relies upon the Ethereum blockchain and Ethereum Virtual Machine, so all descriptions will apply solely to it, as described in Wood[9]. Other types of blockchains do exist, notably the bitcoin blockchain (Nakamoto[4]), from which the Ethereum blockchain essentially evolved. In a slight oversimplification, a blockchain can be thought of as an immutable, time-ordered, distributed public transaction ledger. Each blockchain also has a unit of transaction associated with it, in this case, ether. The property of immutability is what makes blockchain technology suitable for voting applications. Once a transaction has been recorded on the blockchain, it is almost impossible to reverse. The way this is accomplished is through the process of "mining." Miners represent nodes in the Ethereum network whose task it is to validate potential transactions, order them into blocks, and attempt to include them into the main chain. If they are successful, they receive a reward in ether (~2 ether, currently about $9000). This process occurs every 15 seconds. Furthermore, the block which gets chosen to be included in the main chain is chosen based on a random lottery system, where computational power translates into increased chances of winning. This is achieved by the network protocol stating that a block must be found whose hash (plus a special value called a nonce) is equal to or greater than a certain value. This ensures that as long as the hashing power of the network is always mostly benevolent, the ledger is immutable. Conceivably, an adversary could submit illegal transactions to the network, but unless the attacker could control more than 51% of the computing power of the network, this is quite unlikely. For further information, refer to Nakamoto[4].

In addition to the ability to order and record transactions in its native currency, ether, the Ethereum blockchain offers something additional: the Ethereum Virtual Machine. This essentially treats the blockchain as a global state machine or world computer (but slow: one packet of instructions every 15 seconds!). The state of a program, known as a smart contract, with its public and private variables, transition functions and so on, exists on the blockchain. This program is able to perform calculations, provide public access to variables and validate messages. This setup comes with a catch, however. All function calls to the smart contract cost real money. However, at current prices, this only equates to $.13 per voter, something which should come down in the future anyways.

Using this smart contract functionality allows the entire voting process to happen in a public, verifiable manner. With paper ballots, auditing requires recounting votes by hand. With this system, this is totally unnecessary, since the source code can be verified to know that it is doing exactly what it purports to.

Background

In order to understand current developments in this area, it is necessary to take a look at previous efforts and critiques of those efforts. It is probably best to start at the beginning, with the reference blockchain implementation, Bitcoin (Nakamoto[4]), designed by Satoshi Nakamoto. This pseudonymous programmer came up with the proof-of-work consensus mechanism which enables blockchain to work as an immutable ledger in the first place. Technically, this is known as solving the “double-spend” problem, because if some adversary tries to “double-spend,” i.e. spend money they don’t have, this will be detected by the network due to transaction validation and the proof-of-work mechanism which has the computing power of millions of computers behind it (which would have to be matched, compute power for compute power, in order to broadcast a ‘fake’ transaction).

Something else to understand about the Bitcoin network (which also applies to Ethereum) is the concept of wallets and addresses. Since a blockchain exists to record transactions in its native currency, a concept is required for users to prove ownership over their share. This is done through asymmetric cryptography. In order to provide a “destination” for coins (used here as a generic term for a unit of currency) to reside at, a user generates a public/private key pair, similar to the ones used for digital signatures. Essentially, the public key acts as a public address which the protocol assigns a balance to. When a user wants to send coins to another address, they can only do so by signing a transaction with the corresponding private key. Otherwise, the transaction is rejected by the network. This functionality is key to the voting process. By keeping the private key secret, that is, only generated and accessed while the program is running and never saved to disk, one can be certain that only the people with access to that device could have sent that transaction. Furthermore, because a public key has no personal identifying information associated with it, the voter’s identity is protected to some extent.

After Bitcoin came Ethereum, which adds the Ethereum Virtual Machine (EVM) to the picture. Similar to the way a regular virtual machine functions, the EVM accepts bytecode instructions which can change the state of a program which exists on the blockchain. In the case of a smart contract, the bytecode-compiled program provides an interface, through public facing functions and variables, which enables the user(s) to transact with it. By assigning a “gas price” to various instructions (per the reference implementation, Wood[9]), such as the EVM’s stack operations, all procedures ultimately have a cost, priced in ether, representing a fee for that transaction/function call. For example, an instruction like ADD or PUSH only costs 3 units of gas, whereas an instruction like SSTORE, which manipulates the associated smart contract’s assigned storage structure, costs between 5000 and 20000 units, depending upon whether data is being written or deleted (Wood[9] p. 26). Thus, this cost must be considered when designing applications that use the Ethereum blockchain.

In order to call one of the smart contract’s functions, the following process occurs. First, a client application references a function from the compiled smart contract’s ABI (Application Binary Interface). After passing the appropriate arguments, this data is broadcast like a normal

transaction to a node in the Ethereum network. This node first validates the transaction itself, including making sure that sufficient gas fees were included in the transaction. Then, the node running the Ethereum client attempts to execute the function. If an error is thrown, all state changes are reverted. Only upon a successful return do these state changes get included in a block. Once they do, however, they are permanent as long as the Ethereum network is secure. In the context of this project, once a vote is cast, once it is recorded and tallied, this change in state can be publicly verified.

Description

Hardware

- My Android phone
    - Samsung Galaxy S9+
    - Android version 10
    - Kernel version 4.9.186
    - Knox version 3.4.1
- Another android phone for each voter
    - Currently running a Pixel 2 emulator
- An ethereum node
    - Currently using <*https://infura.io*>
    - Infura provides up to 100,000 free daily requests (smart contract function calls)
    - In the future, will be run on a self-hosted node (initial setup is done, but needs to be updated and connected/secured)

Software

- Client voting app (`client`)

```
_loadContract()
```

    - Loads the contract ABI (from a json) into memory
    - Uses a contract address (might need to be implemented as a parameter)
    - Returns a `Future<DeployedContract>`

```
query(String functionName, List<dynamic> args)
```

    - Wrapper for calling a smart contract function that will not change the blockchain's state (thus, it's free of transaction costs)
    - Returns a `Future<List<dynamic>>` (whatever the smart contract function returns)

```
submit(String functionName, Credentials credentials, List<dynamic> args)
```

    - Wrapper for calling a smart contract function that *will* change the blockchain's state
    - Requires a gas fee

    - Requires a private key (corresponds to the client voter's Ethereum address)
- Essentially, the client app:
    - Loads the contract
    - Generates a voter address (public/private key pair)
    - Generates a QR code of the public key, which is entered into the voter roll
    - Sets up a listener with a loop that queries the total vote tally every 15 seconds (whenever there is a new block)
    - When the voter presses the vote button, calls `submit()` to cast their vote
- Voting administrator app (`admin`)
    - Loads Contract
    - Loads Administrator address (this must have enough ether to conduct the vote i.e. have enough ether to cover the cost of every function call)
    - Scans the QR code of each voter's address(public key) in order to register them
    - Sends the list of voters' addresses to the smart contract to disallow unregistered voters from casting votes (partially implemented, in progress)
    - Funds each voter's address with enough ether to cover the cost of casting one vote (this final step begins the voting process)
- Smart contract "dapp" (decentralized application)
    - Current address: `0x93790eCe73A285ddd1C271455aDa4e908A1B243B`
    - <https://rinkeby.etherscan.io/address/0x93790eCe73A285ddd1C271455aDa4e908A1B243B>
    - Rinkeby Testnet
    - Ether has no value
    - Holds final vote tally
    - Holds list of registered voters
    - Increments this tally appropriately upon the receipt of a valid ballot
    - Makes sure only registered voters and the voting administrator can interact with it

Methodology

0.1.0 - Sept 2021

- Ethereum Smart Contract which has a public function that allows a user of the client app to send either a 0 or a 1 (representing one vote/ballot) which is added to the public results tally.
- Client Android app that can interact with the smart contract through a remote procedure call.

0.2.0 - Oct 2021

- Client Android app which enables the user/voter to generate an ethereum address (public/private key pair) and corresponding QR code (public key). This is then scanned by the administrator's app, registering the voter and beginning the voting process.
- Administrator Android app which enables the user/administrator to register voters by collecting voter public keys through scanning their QR codes. These addresses are then funded with enough ether to cast a vote.

0.3.0 - Nov 2021
- Add checks to smart contract which allow only registered voters to cast a vote.

0.4.0 - March 2022
- Cleanup!
- Video Demo

0.5.0 - April 2022
- GUI
- Return to registration issues

References

1. Antonopoulos, Andreas M. and Gavin Wood. *Mastering Ethereum: Building Smart Contracts and DApps*. O'Reilly, 2019, github.com/ethereumbook/ethereumbook.

2. Au, Man Ho, et al. "Short Linkable Ring Signatures Revisited." *Lecture Notes in Computer Science*, edited by A. S. Atzeni and A. Lioy, vol. 4043, pp. 101-115, Springer-Verlag, 2006, doi:10.1007/11774716_9.

3. Bartolucci, Silvia, et al. "SHARVOT: Secret SHARe-Based VOTing on the Blockchain." *2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pp. 30-34, Association for Computing Machinery, 2018, doi:10.1145/3194113.3194118.

4. Nakamoto, Satoshi. "Bitcoin: a peer-to-peer electronic cash system." *Bitcoin White Paper*, 2008, bitcoin.org/en/bitcoin-paper.

5. Park, Sunoo, MIT, et al. "Going from bad to worse: from internet voting to blockchain voting." *Journal of Cybersecurity*, vol. 7, no. 1, 2021, doi:10.1093/cybsec/tyaa025.

6. Shah, Akhil, et al. "Blockchain Enabled Online-Voting System." *ITM Web Conf.,* vol. 32, 03018, International Conference on Automation, Computing and Communication, 2020, doi: 10.1051/itmconf/20203203018.

7. Sharma, Trishie, et al. "A Cost-Efficient Proof-of-Stake-Voting Based Auditable Blockchain e-Voting System." *IOP Conf. Series: Materials Science and Engineering*, vol. 1099, 012038, IOP Publishing, 2021, doi: 10.1088/1757-899X/1099/1/012038.

8. Volkhausen, T. "Paillier cryptosystem: A mathematical introduction." *Seminar Public-Key Kryptographie*, 2006.

9. Wood, Gavin. "Ethereum: a secure decentralised generalised transaction ledger." *Ethereum Project Yellow Paper*, 11 July 2021. github.com/ethereum/yellowpaper.

10. Yu, Bin, et al. "Platform-independent Secure Blockchain-Based Voting System." *Cryptology ePrint Archive*, 2018, eprint.iacr.org/2018/657.