An E-voting System Based on Tornado Cash

Stefano Bistarelli[1], Bruno Lazo La Torre Montalvo[1], Ivan Mercanti[1], and Francesco Santini[1]

Department of Maths and Computer Science, University of Perugia, Perugia, Italy
`firstname.lastname@unipg.it`

Abstract. We propose a pseudo-anonymous e-voting platform based on the blockchain of Ethereum and a coin-mixer, that is Tornado Cash. After an online authentication and authorization phase, the user receives a fungible (i.e., pseudo-anonymous) voting token that can be deposited to a coin pool belonging to Tornado Cash (TC), together with an amount of Ether (ETH) A that will be used to pay successive fees. TC uses a smart contract that accepts token deposits that can be later withdrawn by a different address. In order to preserve privacy, a *relayer* contract can then be used to withdraw to a fresh ETH address (thus pseudo-anonymous) using A to pay fees. Relayers solve "fee payment dilemma", that is paying withdrawal fees by maintaining pseudo-anonymity. Finally, a further smart contract collects preferences and, after the closure of the elections, it automatically performs the counting of votes. All the front-end has been developed in a Web browser, by using Javascript and avoiding the voter performing any command-line operation to prepare transactions.

Keywords: E-voting · Ethereum · anonymity.

1 Introduction

The right to vote is one of the most important forms of manifestation of personal freedom and democratic expression. Thanks to the right to vote, citizens have the opportunity to intervene in important decisions for the collectivity: it is therefore important that this right is exercised freely, secretly, and with a wide opportunity of choice. E-voting (or electronic voting) refers to the use of electronic and computer technologies during the voting or counting process. As a first important observation, we would like to remark that the e-voting system we propose in this paper is not anyway considered as a substitute of traditional voting systems based on paper ballots, in election scenarios that support representative democracy in local, regional or national governments. We instead support the use of e-voting systems in case the application is less sensitive and attracts less interest in being massively attacked, thus invalidating the electoral process and causing disorder. For example, we think of elections concerning the administrative councils and boards of organizations and companies, especially those ones decentralized around the world.

Our favourite scenario for the e-voting system we propose consists of tens/hundreds voters (but not millions) who want/need to remain pseudo-anonymous, who can spend a few euro to vote, who cannot meet in presence, but need strong guarantees on the security and privacy of the elections result. Such warrants are in this case offered by a blockchain-based system, which we will explain in the following of the paper.

Some issues associated with "traditional" voting systems that uses paper ballots are represented by: their high cost derived from the use of equipment (for voting operations, tellers, polling chairmen); the time (in days) that can pass from the time the voting operation is completed to the time the count results are published; the low accessibility: statistically, having to physically reach the seat of the polling station is one of the major causes of voting abstention. The use of e-voting can mitigate these drawbacks, since the cost is in general lower, the voting operation results in an immediate counting (with no human error), a mobile device is enough to cast a preference vote from anywhere and, finally, filling in the ballot is guided through mouse clicks, thus eliminating null votes due to errors in filling in the ballot paper.

The purpose of this study is the implementation of an electronic voting system based on blockchain technology, which satisfies some classical properties of an e-voting system (see Sect. 2.3). For the implementation of such a system the following objectives have been identified: *i)* the distribution of a fungible ERC20 token dedicated to voting: we implemented the entire system within the Ethereum blockchain, creating a token that is distributed to an authorized user after identification. *ii)* User pseudo-anonymization: we implemented (readapting it from an existing solution, not oriented to e-voting) a coin-mixing service that allows a user to carry out deposit and withdrawal of the voting token in a decentralized and privacy-oriented manner. Finally, *iii)* the system is offered through a decentralized application (or *dApp*) dedicated to voting: we designed a dApp that allows the user to cast his vote by spending his voting token.

The application we propose in this paper elaborates on the Tornado Cash smart contract, which improves transaction privacy by breaking the on-chain link between source and destination addresses. This contract accepts ETH deposits that can be withdrawn by a different address. In order to preserve privacy, a *relayer* can be used to withdraw to an address with no ETH balance (i.e., a fresh address). Whenever ETH is withdrawn by the new address, there is no way to link the withdrawal to the deposit, ensuring complete privacy. Tornado Cash originally proposes itself as a coin-mixer service: these tools are intended to mix digital money with that of other users in order to obfuscate the source and destination of crypto assets. It is here used as a way to preserve pseudo-anonymity of preference votes.

The front-end of the application has been developed in Javascript, so that the user can easily perform all the operations by using a plain Web-browser, without interacting with the Ethereum blockchain with (more complicated) command-line tools.

The paper is organized as follows. Section 2 contains an overview of Ethereum, ERC20 Tornado Cash and e-voting system. Our e-voting model is presented in Sect. 3. Section 4 compares our proposal with other works in the literature and Sect. 5 draws some conclusions and discusses possible future work.

2 Background

In this section we partition the necessary background information to later describe how our e-voting proposal based on the blockchain of Ethereum works. First of all, Sect. 2.1 briefly introduces Ethereum, while Sect. 2.2 presents the main token standard in Ethereum, i.e, ERC20. Then we list the most important properties that a voting system should satisfy in Sect. 2.3. Finally, Tornado Cash is presented in Sect. 2.4.

2.1 Ethereum

Ethereum is a blockchain protocol created by Vitalik Buterin [5], which implements different features respect to the Bitcoin protocol. The main feature, which made it popular and second only to Bitcoin in terms of volume, is the possibility to create decentralized apps (*dApps*) via smart contracts.

Smart contracts are programs based on rule sets and deployed on the Ethereum blockchain, which allows functions to be executed if certain conditions are met.[1] Ethereum smart contracts can be written using several programming Turing-equivalent languages, but the most popular is Solidity created by Gavin Wood, one of Ethereum's co-founders. These contracts can execute transactions automatically, so there is no need for a third-party entity to take action.

dApps are in general applications that connects a smart contract with a front-end user interface.[2] To be defined as dApp, an application needs to be:

- *Decentralized*: operating within the blockchain, where no entity has control.
- *Deterministic*: a certain input always correspond to the same output.
- *Turing-complete*: every action can be performed with the required resource.
- *Isolated*: executing on a virtual environment called Ethereum Virtual Machine (EVM), so, in case of failure, the blockchain network will not be affected.

2.2 The ERC20 standard

ERC-20 is the technical standard for fungible tokens created using the Ethereum blockchain. A fungible token is interchangeable with another token, while the

[1] Smart contracts: `https://github.com/ethereum/ethereum-org-website/blob/dev/src/content/developers/docs/smart-contracts/index.md`.

[2] dApps: `https://github.com/ethereum/ethereum-org-website/blob/dev/src/content/developers/docs/dapps/index.md`.

well-known non-fungible tokens (NFTs) are not interchangeable.[3] ERC-20 offers several functions and events that a token must implement. The minimum of functions and information needed in an ERC-20 compliant token is:

– *TotalSupply*: the total number of tokens that will ever be issued.
– *BalanceOf*: the account balance of a token owner's account.
– *Transfer*: automatically executes transfers of a specified number of tokens to a specified address for transactions using the token.
– *TransferFrom*: automatically executes transfers of a specified number of tokens from a specified address using the token.
– *Approve*: allows a spender to withdraw a set number of tokens from a specified account, up to a specific amount.
– *Allowance*: returns the remaining number of tokens that spender will be allowed to spend on behalf of owner.
– *Transfer*: an event triggered when a transfer is successful.
– *Approval*: a log of an approved event.

2.3 Important voting properties

A good voting (and also e-voting) system has to satisfy the following properties [6, 10, 14], in particular:

– *Verifiability*: it is possible to verify that the counting of votes has been performed correctly.
– *Uniqueness*: a user is not allowed to vote more than once.
– *Integrity*: no one can change or delete a vote without revealing it.
– *Privacy*: it is not possible to determine the vote of a user.
– *Counting*: the vote count has to be verifiable by everyone.
– *Authentication*: only users who have correctly identified themselves can vote.
– *Confidentiality*: intermediate results cannot be obtained during the proceedings.
– *Lack of evidence*: users cannot prove for whom they voted.
– *Reliability*: the voting system must be reliable and stable.

2.4 Tornado Cash

Tornado Cash is an open-source, fully decentralized non-custodial[4] protocol implemented within the Ethereum blockchain. It improves transaction privacy by breaking the on-chain link between source and destination addresses. It uses a smart contract that accepts ether and other ERC20 tokens deposits from one address and enables their withdrawal from a different address. Tornado Cash

[3] `https://www.investopedia.com/news/what-erc20-and-what-does-it-mean-ethereum/`.
[4] Non-custodial wallet services are platforms that allow users to possess their private keys.

smart contracts are implemented on the Ethereum blockchain, so they can neither be modified nor tampered with. Mining smart contracts and administration smart contracts are implemented by the community in a decentralized way: any user can propose a smart contract and anyone can vote for or against it by locking *TORN*s (i.e., *Tornado Cash tokens*). After 5 days, if a minimum of 25000 TORNs has been reached and the proposal is voted by the majority of votes, it is approved. Hence, it changes the mining and the administration smart contracts.

Tornado Cash is currently operating with several different cryptocurrencies and layer-2 networks:

– Ethereum Blockchain : ETH (Ethereum), DAI (Dai), cDAI (Compound Dai), USDC (USD Coin), USDT (Tether) & WBTC (Wrapped Bitcoin),
– Binance Smart Chain: BNB (Binance Coin),
– Polygon Network: MATIC (Polygon),
– Gnosis Chain (former xDAI Chain): xDAI (xDai),
– Avalanche Mainnet: AVAX (Avalanche),
– Optimism, as a Layer-2 for ETH (Ethereum),
– Arbitrum One, as a Layer-2 ETH (Ethereum).

The user who wants to make a deposit[5] in the pool, will have to randomly generate a *secret* k and a *nullifier* r with $k, r \in \mathbb{B}^{248}$, and its hash called *commitment* C, such that $C = H_1(k||r)$.[6] Along with N tokens are then sent to the smart contract $\mathcal{C}$ interpreting C as a 256-bit unsigned integer. The contract will then accept the deposit of the N token and add C as a leaf of a tree, in case the tree is not full.[7]

To withdraw (Fig. 1), the user must select the recipient's address A with a transaction fee f such that $f \leq N$. Then, the user should provide proof that he/she possesses a secret to an unspent commitment from the smart contract's list of deposits. The *zkSnark*[8] technology allows doing that without revealing which exact deposit corresponds to this secret. The smart contract will check the proof, and transfer deposited funds to the address specified for withdrawal. An external observer will be unable to determine which deposit this withdrawal comes from.

To perform the withdrawal, two different options are available:

– The user links their wallet (*Metamask* or *WalletConnect*) to the Tornado Cash website, and they pay for the gas needed to withdraw the amount deposited.
– The user use a *relayer* to make the withdrawal to any Ethereum address without needing to make the wallet connection on the Tornado Cash website.

[5] Web-connection of Tornado Cash to Metamask or a private wallet: `https://tornadocash.eth.limo/`.
[6] With || that stands for concatenation.
[7] `https://tornado-cash.medium.com/introducing-private-transactions-on-ethereum-now-42ee915babe0`.
[8] zk-SNARK: `https://z.cash/technology/zksnarks/`.

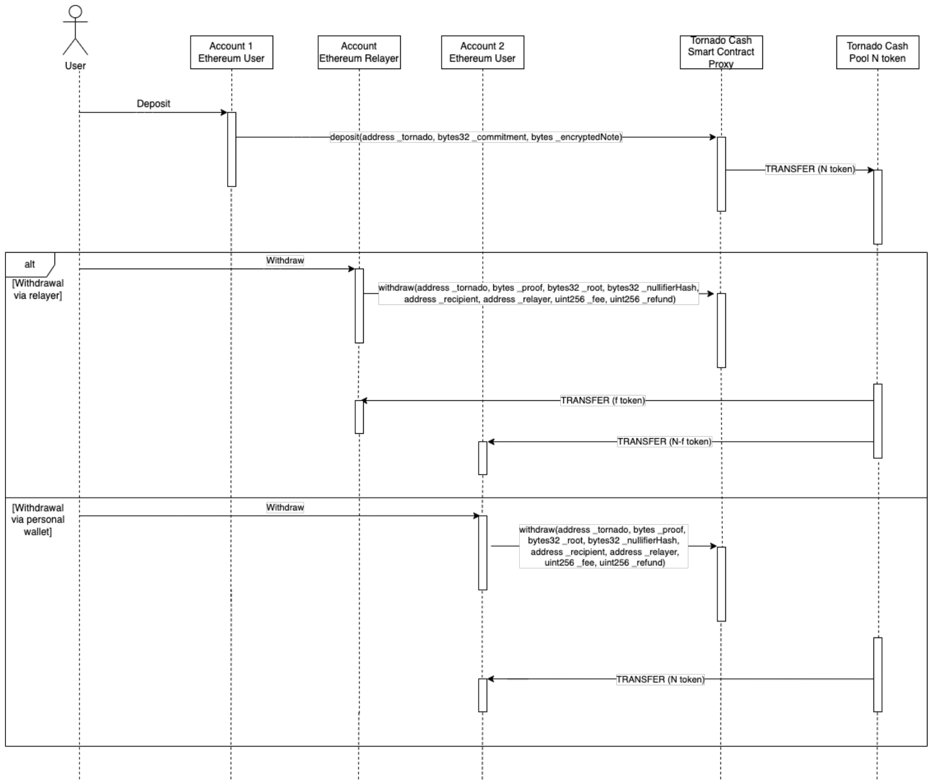

User
Account 1 Ethereum User
Account Ethereum Relayer
Account 2 Ethereum User
Tornado Cash Smart Contract Proxy
Tornado Cash Pool N token
Deposit
deposit(address _tornado, bytes32 _commitment, bytes _encryptedNote)
TRANSFER (N token)
alt
[Withdrawal via relayer]
Withdraw
withdraw(address _tornado, bytes _proof, bytes32 _root, bytes32 _nullifierHash, address _recipient, address _relayer, uint256 _fee, uint256 _refund)
TRANSFER (f token)
TRANSFER (N-f token)
[Withdrawal via personal wallet]
Withdraw
withdraw(address _tornado, bytes _proof, bytes32 _root, bytes32 _nullifierHash, address _recipient, address _relayer, uint256 _fee, uint256 _refund)
TRANSFER (N token)

Fig. 1. Sequence diagram: Deposit and Withdrawal operations.

Since the relayer is in charge of paying for the transaction gas, it will receive a small portion of the deposit.

The user's deposit and withdrawal actions are performed by interacting with the smart contract of the Tornado Cash Proxy.[9]

3 The E-voting model

We aim to design an e-voting model that satisfies the properties described in Sect. 2.3. Moreover, we want to achieve three other goals: the token distribution, the user pseudo-anonymization and the voting dApp distribution. To better understand our model, in this section we refer to the various actors as follows:

- *Account1*: Ethereum account with ether and possibly other tokens. Not anonymous, linked to a voter.

[9] Goerli Testnet Network: `https://goerli.etherscan.io/address/0x454d870a72e29d5e5697f635128d18077bd04c60`.

– Account2: Ethereum account without any related transaction, and for this reason not linkable to the identity of the voter.
– *Admin*: The organizer of the election.
– *SCP*: Smart contract pool.

The first step is a smart contract distribution of an ERC20 token (DVT[10]) by the Admin. The DVT will use to allow users to vote. Then the Admin sets a date by which users who want to vote will have to identify themselves (and consequently receive 1 DVT). The smart contract of the DVT token will be owned by the Ethereum account that issued it, and only this is authorized to mint and distribute new tokens. Listing 1.1 shows the contract of such a token.

Listing 1.1. The smart contract describing the DVT Token

```
pragma solidity ^0.8.0;
import "@openzeppelin/contracts/token/ERC20/ERC20.sol ";
import "@openzeppelin/contracts/access/Ownable.sol ";
contract DevToken is ERC20, Ownable{
        constructor() ERC20("DevToken", "DVT"){}
        function issueToken(address receiver) public onlyOwner{
                _mint( receiver , 1*10**18); }
}
```

The users also need to deposit 0.0015 ETH (currently equivalent to €1.5) and 1 DVT to the corresponding SCP, without withdrawing the token before the set date. Once the date expires, users who want to vote can proceed to withdraw the deposited tokens. The withdrawal of 0.0015 ETH is made via relayer to a new Ethereum wallet (Account2). This is to avoid in/out transactions and, therefore, to pseudo-anonymize the user. The user withdraws 1 DTV on the same wallet (Account2). Finally, the Admin makes public the dApp websites allowing users to vote by sending the DVT token to the smart contract dedicated to voting.

Figure 2 shows how the distribution of a token works. The Admin creates a smart contract. The smart contract implementation of a new ERC20 token (DVT) is used to secure voting rights. It is deployed on the Goerli test network (or briefly *testnet*) via the Hardhat[11] development environment. The smart contract is owned by the Ethereum account that released it (Admin), and only that account will be able to generate new tokens. It is possible to control from the smart contract the number of tokens in circulation and the Ethereum accounts that own them. Admin also authenticates the user and then gives one DTV token to them. To authenticate users, Admin creates a web page with a form. The user fills this form with their personal data. Admin checks that the data are valid and the user did not already receive the DTV token. If everything is correct, the user is authenticated and receives 1 DTV token. Due to its decentralized nature, it is possible to control from the smart contract the number of coined tokens and the Ethereum accounts that own them.

[10] DevToken.
[11] `https://hardhat.org/`.

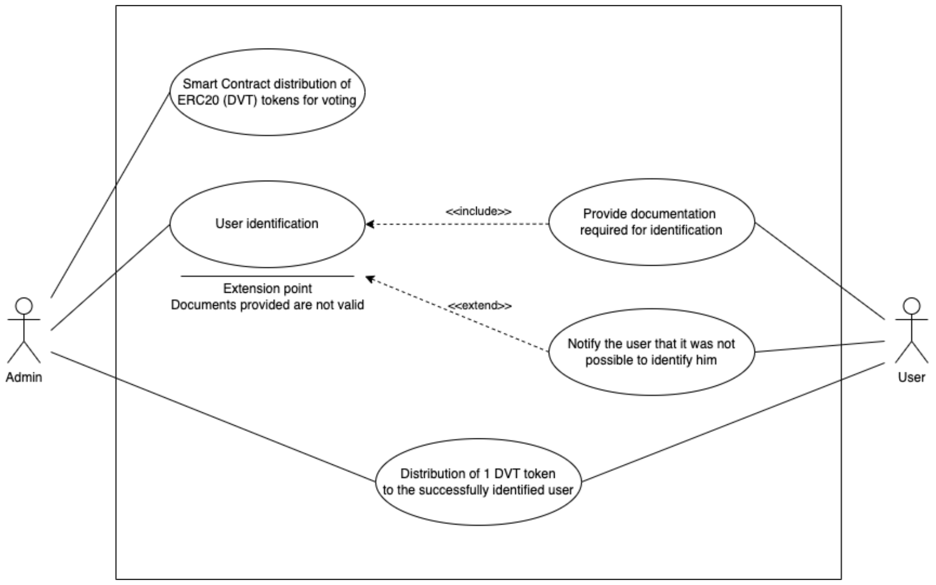

Smart Contract distribution of ERC20 (DVT) tokens for voting
User identification
<<include>>
Provide documentation required for identification
Extension point
Documents provided are not valid
<<extend>>
Notify the user that it was not possible to identify him
Admin
User
Distribution of 1 DVT token to the successfully identified user

Fig. 2. Use case diagram: distribution of a vote token.

To ensure the pseudo-anonymization of users we extend in our Smart Contract the Tornado Cash protocol. In particular, we implement the deposit and withdrawal of ETH and DTV via relayer. Figure 3 and Fig. 4 describe the Sequence diagram of the tokens distribution via relayer. By executing the instructions in these figures, the SCP distributes: *i)* the smart contract hasher, which calculates the hash when a deposit is made (*2_deploy_hasher.js*), *ii)* the smart contract verifier, which verifies that the withdrawal proof is valid (*running 3_deploy_verifier.js*), *iii)* the ETH or DTV SCP using *4_deploy_eth_tornado.js* and *5_deploy_erc20_tornado.js* respectively. The Admin distributes SCP to deposit and withdraw ETH and DTV. Moreover, the SCP allows the users to deposit from Account1 and withdraw in Account2. So the user has to use SCP for deposit and withdraw operations before voting to remain pseudo-anonymous.

The user, after finalizing the deposit, receives in output a private note that they necessarily have to memorize in order to later withdraw the deposit. As has already been described, Admin sets an expiration date for making deposits. Once this date expired, the voter withdraws ETH from Account2. To pay the fee for this transaction they can not use Account1 (if the user does so, this would create a correlation between Account1 and Account2, and thus pseudo-anonymity would be lost) nor Account2 (this one does not have any ether). An Ethereum account called *relayer*, configured by the Admin, is necessary to pay the fee for an ETH withdrawal transaction. To configure the relayer, we refer

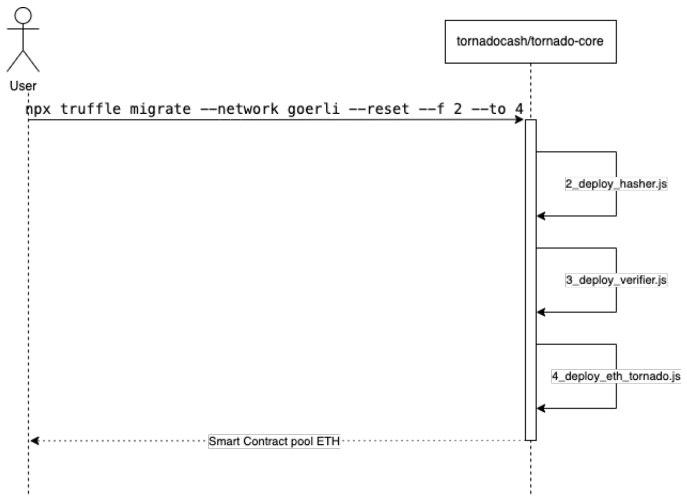


Fig. 3. Sequence diagram: distribution smart contract pool ETH.

to Tornado Cash's *tornado-relayer* repository on GitHub[12]. In particular, we extend it with the possibility of operating in Goerli with our SCP and to modify the transaction gas and fee value.

Figure 5 shows what a user can do with our SCP. The user can deposit either ETH or DTV from a wallet and get it in a relayer wallet in order to break their connection with the coin, i.e. with the purpose to pseudo-anonymize the user. The withdrawal function:

```
withdraw ( bytes _proof , bytes32 _root ,
bytes32 _nullifierHash , address _recipient , address _relayer ,
uint256 _fee , uint256 _refund )
```

Where we have the recipient's address (parameter _recipient), a transaction fee (parameter _fee), the proof (parameter _proof) to make a withdrawal, a root (parameter _root) selected from those stored on the contract and the hash of the nullifier (parameter _nullifierHash).

Having performed the withdrawal transaction, the user has 0.0015 ETH and 1 DTV in their Account2. ETH value is for pay transaction fees and the DVT, instead, is for voting. At this point, the admin creates a web dApp based on smart contract. The contract can be created using the constructor:

```
constructor(string[] memory proposalNames, uint256 tokenQuantity_, address token)
```

The smart contract constructor uses:

[12] https://github.com/tornadocash/tornado-relayer/tree/c838316436a9f87f8655087c34764b46e4b1491b.

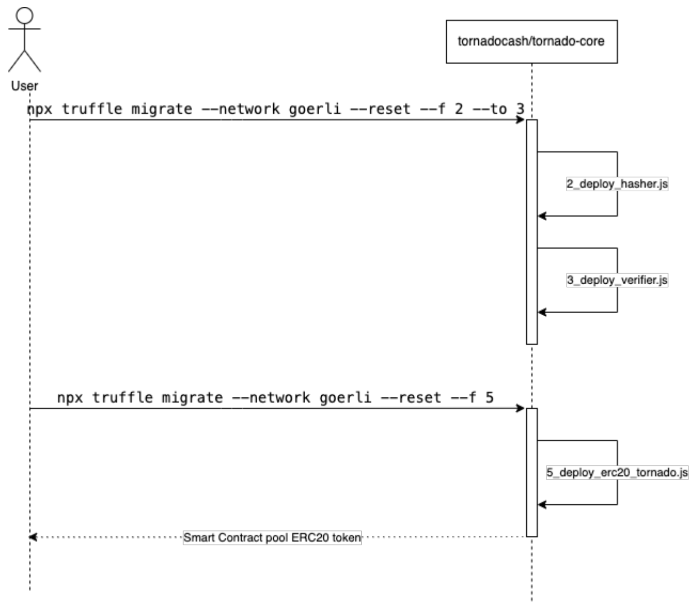
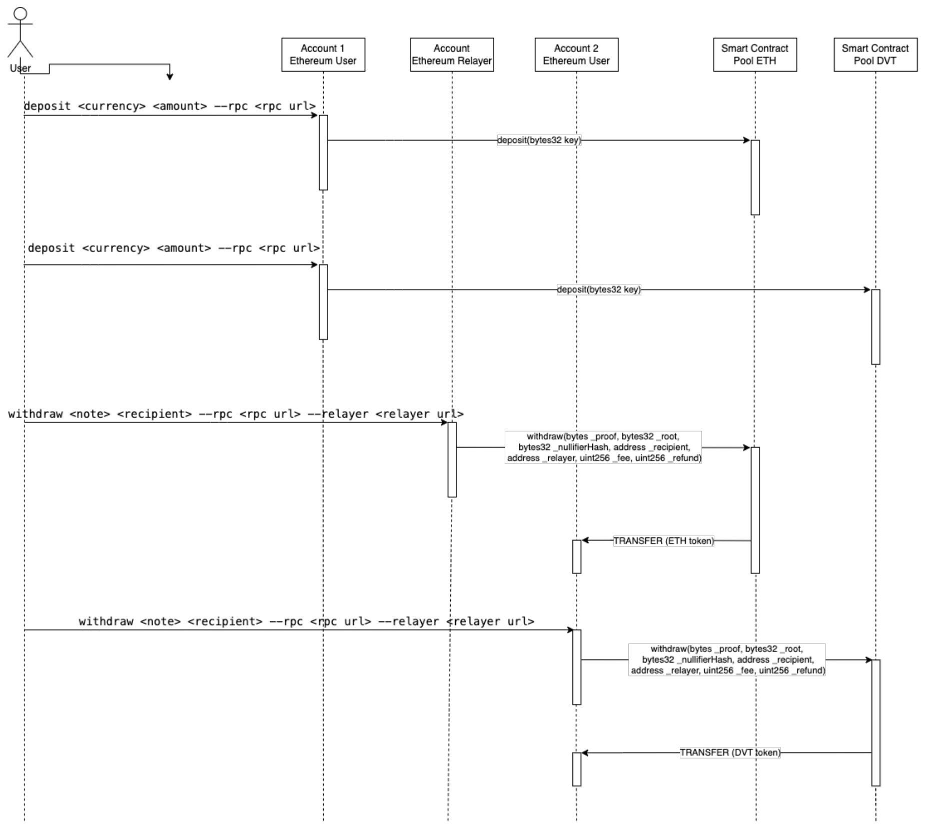

User
tornadocash/tornado-core
npx truffle migrate --network goerli --reset --f 2 --to 3
2_deploy_hasher.js
3_deploy_verifier.js
npx truffle migrate --network goerli --reset --f 5
5_deploy_erc20_tornado.js
Smart Contract pool ERC20 token

Fig. 4. Sequence diagram: distribution smart contract pool (ERC20).

- the PROPOSALNAMES parameter to set up a dynamic array that will contain the names of the candidates on the ballots;
- the TOKENQUANTITY_ parameter to set the amount of tokens needed for each user to be able to vote;
- the TOKEN parameter to indicate the address of the ERC20 token that will be required to vote.

Instead, to handle voting in the smart contract, function VOTE(UINT PROPOSAL) is used. The PROPOSAL parameter indicates the candidate selected by the voting user.

Finally, the Admin implements a web application that interacts with the smart contract of voting (Fig. 6). The user who wants to vote has to log in to their Metamask account (connected to Account2) from its extension on a browser. Then the user can access the dedicated voting web page indicated by the administrator. When this page loads, Metamask will ask the user for the smart contract permission to receive 1 DVT token from the user's wallet (GETVOTES() function in Fig. 6). Notice that it is possible only if the token is present in the user's wallet (Account2) and the user wants to vote. Finally, the user needs to wait a few seconds for the transaction to be mined on the Blockchain. Afterwards, the user can select the candidate they wish to vote for from the drop-down menu. After clicking on the button that says "Vote", Metamask will prompt the user for confirmation to call the VOTE(UINT PROPOSAL) function of the smart contract. Once this transaction is mined, the vote count received by the selected candidate will be increased.


User
Account 1 Ethereum User
Account Ethereum Relayer
Account 2 Ethereum User
Smart Contract Pool ETH
Smart Contract Pool DVT
deposit(bytes32 key)
deposit(bytes32 key)
withdraw(bytes _proof, bytes32 _root, bytes32 _nullifierHash, address _recipient, address _relayer, uint256 _fee, uint256 _refund)
TRANSFER (ETH token)
withdraw(bytes _proof, bytes32 _root, bytes32 _nullifierHash, address _recipient, address _relayer, uint256 _fee, uint256 _refund)
TRANSFER (DVT token)

```
deposit <currency> <amount> --rpc <rpc url>
deposit <currency> <amount> --rpc <rpc url>
withdraw <note> <recipient> --rpc <rpc url> --relayer <relayer url>
withdraw <note> <recipient> --rpc <rpc url> --relayer <relayer url>
```

Fig. 5. Sequence diagram: interaction via command line tool.

3.1 Satisfied Properties

The proposed model satisfy the properties presented in Sect. 2.3 except for *Confidentiality* and *Lack of evidence*.

Verifiability: Transactions in the Ethereum blockchain are public, so it is always possible to verify that the counting of votes has been performed correctly.

Uniqueness: Double-voting is prevented by the fact that double-spending is not possible with the blockchain technology [5].

Integrity: When a transaction is in a confirmed block, to modify that block is computationally hard by design [5], since it is required to also modify all the successive blocks. Moreover, it is not possible to change or delete a transaction (vote) without revealing it.

Privacy: Voters' Account2 cannot be associated with their identity because the *Token Distribution* is implemented via relayer using the Tornado Cash protocol. In this way, the users have an Ethereum account without any transaction; therefore, it is not possible to identify the voter.

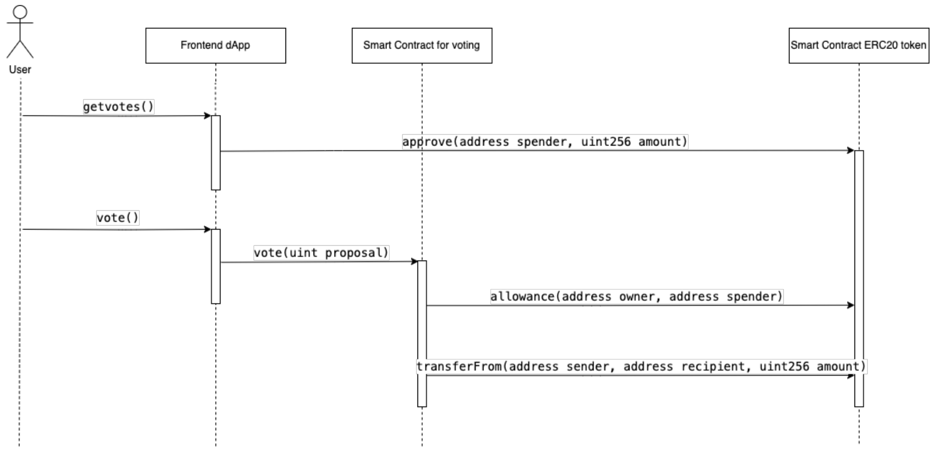

User
Frontend dApp
Smart Contract for voting
Smart Contract ERC20 token
getvotes()
approve(address spender, uint256 amount)
vote()
vote(uint proposal)
allowance(address owner, address spender)
transferFrom(address sender, address recipient, uint256 amount)

Fig. 6. Sequence diagram: user voting.

Counting: Each valid transaction is permanently stored in the blockchain, where it is possible to repeat the counting phase when needed. Any Ethereum node can repeat this phase as needed.

Authentication: This is accomplished by the authentication phase, when the Admin distributes the DTV tokens to the authenticated users.

Confidentiality: Unfortunately, this property is not satisfied by our implementation: in fact, by checking the candidates' accounts it is possible to read intermediate results.

Lack of evidence: Users, revealing the possess of their Account2, can prove for whom they voted. This means we cannot satisfy this property.

Reliability: Clearly, the reliability properties depends on many factors and it is not easy to be measured (e.g., with a simulation). However, Ethereum already proves to be a reliable and largely used infrastructure. Indeed, it is required to use transactions with a high fee in order not to lose votes; nevertheless, a voter can check if their votes has been included in the blockchain. Clearly, the size of the peer-to-peer network and the number of miners mitigate such problems.

3.2 Cost estimation

In this section, we provide an estimation of the costs needed to cast a vote on the *Mainnet* of Ethereum. Alternatively, a public testnet can be used where ETH has no real value; the testnet used in this paper, for example, i.e., Goerli, is a *proof-of-authority* (PoA) Ethereum testnet.[13] However, a project with a mainnet has undoubtedly more credibility than a project without one, since transactions

[13] The PoA is a consensus method that gives a small and designated number of blockchain actors the power to validate transactions or interactions with the network and to update its registry. Goerli is said to switch to proof-of-stake soon.

are not simulated. Moreover, the number of miners is higher in a mainnet and the stake is real, thus reducing the chance of an attack. A third possibility would be to run an ad-hoc e-voting blockchain based on Ethereum, but still, the reduced number of nodes in the network would offer fewer security guarantees.

To calculate the average fee cost of each type of transaction in ETH, the values reported following a test phase are considered. The average fees cost of all the needed transaction is:

- 0.0010805 ETH for the deposit of 0.0015 ETH from the Account1 to the SCP.
- 0.001291 ETH for the deposit of 1 DTV from the Account1 to the SCP.
- 0.000448857 ETH for the withdrawal of 1 DTV from the SCP to the Account2.
- 0.000063152 ETH the permission for the voting smart contract to spend 1 DVT in the Account2.
- 0.000107894 ETH the sending of 1 DTV to the voting smart contract from the Account2 to execute the voting.

To sum up, the Account1 spend in mean 0.0023715 ETH, the Account2, instead 0.000512009 ETH. The total result is 0.0028835 ETH, around €4.20 at the time of writing.[14]

4 Related Work

Nowadays, the most spread voting schemes consists in paper-based elections. However, paper-based systems are not completely secure and they may suffer from frauds, even in today's democratic countries,[15] where controversies are very frequent.[16] Estonia became the first nation to hold general elections over the Internet with a pilot project for the municipal elections in 2005. The e-voting system withstood the test of reality and was declared a success by Estonian election officials [2]. Despite this, e-voting systems have not experienced a breakthrough in Europe, since most of the diffidence resides in the general level of trust in government, but also the level of trust in the corporations that supply the machines use in the electoral process [8].

Some proposals have been already opened in the direction outlined by this paper. The most noticeable reference is the *Bitcongress.org project*,[17] which already offers a voting platform based on Bitcoin. However, the software is offered as a broker between the voter and Bitcoin. An evidence is the presence of a "Smart Contract Blockchain": quoting the project white-paper, *"A vote token is sent by a legislation creation tool with combined cryptocurrency wallet. The vote is sent to a smart contract based election holding yay, nay and candidate*

[14] Using gas price to compute fees on 23rd of July 2022.
[15] `http://news.bbc.co.uk/2/hi/uk_news/4410743.stm`.
[16] `http://news.bbc.co.uk/2/hi/europe/4904294.stm`.
[17] Web-site of the Bitcongress.org project: `http://www.bitcongress.org`.

addresses". On the contrary, in our implementation a vote is directly sent to the address of a candidate, without any intermediary. Moreover, still quoting the white-paper, "*The election logs then changes, the vote count is recorded and displayed within Axiomity (a decentralised application) using Bitcongress onto the Smart Contract Blockchain*". In our solution the counting is directly performed in the blockchain. Other commercial systems are *Follow my vote*[18] and *TIVI*[19].

Envisioning the use of blockchains for voting purposes has been already proposed in [11, 12, 16] for example. In the following we present related scientific works. All of them propose solutions without the help of coloured coins or permissioned ledgers, which have been use to respectively simplify the counting process and satisfy further properties, as shown in Sect. 2.3: properties as data confidentiality and uncoercibility seem not be be addressed in all such proposals; with *MultiChain*,[20] or in general permissioned ledgers, it is possible instead. The proposal in [3] simply consists in an electronic voting system based on the Bitcoin block-chain technology.

In [7, 1] the author propose an e-voting scheme, which is then implemented in the Ethereum blockchain. The implementation and related performance measurements are given in the paper along with the challenges presented by the blockchain platform to develop a complex application like e-voting. In general, special attention must be paid to the debugging and verification steps on (Ethereum) smart-contracts. In [13] the authors show as a blockchain-based e-voting system with Ethereum and Metamask can serve as a solution to security and trust issues in the e-voting system.

Even the execution of the protocol in [9] is enforced by using the consensus mechanism that also secures the Ethereum blockchain. However, by using a permissionless blockchain, public verifiability does not provide any coercion resistance.

5 Conclusion

This study describes the process of creating an e-voting system based on the Ethereum blockchain, using the protocol implemented by the Tornado Cash coin-mixer to ensure the privacy of the voter and via smart contracts to ensure the transparency of the entire process.

First, we developed a Web application authenticating the user and subsequently distributing an ERC20 token which represents a vote. Following this, by referring to the Tornado Cash protocol: *i)* we developed smart contract pools with the purpose to allow the deposit and withdrawal of the voting tokens (DVT) and ETH, *ii)* we implemented an application through which the user can interact with the aforementioned smart contracts, and *iii)* we used a relayer for withdrawing ETH on behalf of the authorized user. Finally, *iv)* a dApp was developed to collect votes.

[18] FollowMyVote.com: `https://followmyvote.com`.
[19] TIVI: `https://tivi.io`.
[20] MultiChain is a bridging platform for cryptocurrencies and NFTs across blockchains.

Possible future developments of what described in this paper concern the improvement of some of already incorporated functionalities: For instance, we would like to implement a Web dApp, which allows a user to connect their MetaMask account and then interact with the smart contract pools by performing deposits and withdrawals. In addition, we would like to enforce more properties from those presented in Sect. 2.3 *Confidentiality* and *Lack of evidence*, for example by implementing the project through the use of a permissioned blockchain, where the right to read the blockchain is granted only to certain users at certain times, so that they can count the votes once the voting is over [4]. Since we are not die-hard supporters of permissioned ledgers, one more option could be to obfuscate sensitive data, as the variables representing the number of currents votes for each candidate are [15].

Moreover, we plan to integrate the application with stronger schemes of authentication and authorization, such as OAuth and OpenID protocols. Finally, while on-chain confidentiality is ensured by Tornado Cash itself, before and after transactions are executed, the privacy of a voter may not be ensured when transactions are sent over the Internet. For this reason, we plan to extend the application by running it in an overlay network such as TOR.[21]

Acknowledgement

The authors are a member of the INdAM Research group GNCS and Consorzio CINI. This work has been partially supported by:

- GNCS-INdAM, CUP E55F22000270001;
- Project RACRA - funded by Ricerca di Base 2018-2019, University of Perugia;
- Project BLOCKCHAIN4FOODCHAIN: funded by Ricerca di Base 2020, University of Perugia;
- Project DopUP - REGIONE UMBRIA PSR 2014-2020;
- Project GIUSTIZIA AGILE, CUP: J89J22000900005.

References

1. Ahn, B.: Implementation and early adoption of an ethereum-based electronic voting system for the prevention of fraudulent voting. Sustainability **14**(5) (2022). https://doi.org/10.3390/su14052917, `https://www.mdpi.com/2071-1050/14/5/2917`
2. Alvarez, R.M., Hall, T.E., Trechsel, A.H.: Internet voting in comparative perspective: the case of estonia. PS: Political Science & Politics **42**(03), 497–505 (2009)
3. Ayed, A.B.: A conceptual secure blockchain-based electronic voting system. International Journal of Network Security & Its Applications **93** (2017)
4. Bistarelli, S., Mercanti, I., Santancini, P., Santini, F.: End-to-end voting with non-permissioned and permissioned ledgers. J. Grid Comput. **17**(1), 97–118 (2019)

[21] `https://docs.tornado.cash/general/how-to-use-tornado-cash-with-tor`.

5. Buterin, V.: Ethereum white paper: A next generation smart contract & decentralized application platform (2013), https://github.com/ethereum/wiki/wiki/White-Paper
6. Fouard, L., Duclos, M., Lafourcade, P.: Survey on electronic voting schemes. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.295.7959&rep=rep1&type=pdf (2007), [Verimag technical report, online; accessed 28-January-2018]
7. Hardwick, F., Akram, R.N., Markantonakis, K.: E-voting with blockchain: An e-voting protocol with decentralisation and voter privacy. CoRR **abs/1805.10258** (2018), http://arxiv.org/abs/1805.10258
8. Loeber, L., Council, D.E.: E-voting in the netherlands; from general acceptance to general doubt in two years. Electronic voting **131**, 21–30 (2008)
9. McCorry, P., Shahandashti, S.F., Hao, F.: A smart contract for boardroom voting with maximum voter privacy. In: Financial Cryptography and Data Security. Lecture Notes in Computer Science, vol. 10322, pp. 357–375. Springer (2017)
10. Mote, C.D.: Report of the national workshop on internet voting: issues and research agenda. In: Proceedings of the 2000 annual national conference on Digital government research. pp. 1–59. Digital Government Society of North America (2000)
11. Omohundro, S.: Cryptocurrencies, smart contracts, and artificial intelligence. AI matters **1**(2), 19–21 (2014)
12. Pilkington, M.: 11 blockchain technology: principles and applications. Research handbook on digital transformations p. 225 (2016)
13. Pramulia, D., Anggorojati, B.: Implementation and evaluation of blockchain based e-voting system with ethereum and metamask. In: 2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS). pp. 18–23 (2020). https://doi.org/10.1109/ICIMCIS51567.2020.9354310
14. Schneider, A., Meter, C., Hagemeister, P.: Survey on remote electronic voting. arXiv preprint arXiv:1702.02798 (2017)
15. Suegami, S.: Smart contracts obfuscation from blockchain-based one-time program. IACR Cryptol. ePrint Arch. p. 549 (2022), https://eprint.iacr.org/2022/549
16. Swan, M.: Blockchain: Blueprint for a new economy. " O'Reilly Media, Inc." (2015)