

# Enforcing Confidentiality in Tornado Cash-based E-Voting Systems<sup>\*</sup>

Stefano Bistarelli<sup>1,†</sup>, Ivan Mercanti<sup>1,\*,†</sup> and Francesco Santini<sup>1,†</sup>

<sup>1</sup>*Dipartimento di Matematica e Informatica, University of Perugia, Perugia, Italy*

## Abstract

We propose a pseudo-anonymous and confidentiality-safe e-voting platform based on the blockchain of Ethereum and Tornado Cash. After completing an online authentication and authorization step, the user receives a fungible (i.e., pseudo-anonymous) voting token that can be used in Tornado Cash's (TC) coin pool. Users may deposit the token using the TC smart contract and may, after, withdraw it using a separate address. A *relayer* contract can then be used to withdraw funds to a new ETH address, keeping user information private. Finally, another smart contract collects preferences and encrypts the votes. The public keys of each candidate are used to encrypt the results. In this way, they can be maintained encrypted until the end of the election, when all candidates reveal their private keys. So, everyone can see the results only after the secret keys have been revealed, preserving confidentiality.

## Keywords

E-voting, Voting properties, Confidentiality, Tornado Cash

## 1. Introduction

The issue of electronic voting is a highly debated one, and various workable solutions have been developed over the years; in the last ten years, attention focused on systems based on blockchain systems (see Sect. 6). Permissioned and non-permissioned blockchains naturally satisfy the integrity property of votes, which is one of the most important requirements a voting scheme needs to address: once registered in a distributed ledger, the preference of users cannot be changed if the blockchain is robust enough (e.g., a large number of coin miners).

The most popular devices today are optical scan machines. With this technique, poll workers process voters' paper ballots using optical scanners, which register markings on a page and store the results electronically. Election authorities can compare the paper ballots that each voter fills out with the results of the scanner because this method is designed with a paper audit trail. Ballot marking devices, in which voters select their option on a screen, are a less prevalent form of voting machines. The system then outputs a paper ballot that can either be manually counted or scanned by a computer, rather than electronically recording the selections. In the USA, for example, *Direct-recording electronic* (DRE) machines are the only commonly

---


5th Distributed Ledger Technology Workshop (DLT 2023)


\*Corresponding author.

†These authors contributed equally.

✉ stefano.bistarelli@unipg.it (S. Bistarelli); ivan.mercanti@unipg.it (I. Mercanti); francesco.santini@unipg.it (F. Santini)

ORCID 0000-0001-7411-9678 (S. Bistarelli); 0000-0002-9774-1600 (I. Mercanti); 0000-0002-3935-4696 (F. Santini)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

used machines that do not require paper as a component of their design. Election forensics [1], which can assist in identifying election fraud, was developed in response to doubts about the reliability of DRE voting machines. The elections for Texas senator and governor of Georgia in 2018 highlighted the vulnerability of paperless systems.<sup>1</sup> Both times, it is claimed that paperless DRE machines converted votes for Democratic candidates into votes for Republicans. The lack of a paper audit record makes it unclear what the intended votes were and whether these claims were valid, even though this was probably a software error.

Due to all these issues, this paper does not consider any sort of political elections, but we curb to scenarios with “few” voters (e.g., tens or hundreds) and different election scopes. For example, we may think of stockholder voting procedures on matters of corporate policy in small and medium companies, which are however located in different countries or for which stockholders live and/or work in different countries. This would facilitate the makeup of the board of directors, dividends approval, or in general making substantial changes in the corporation’s operations.

The Tornado Cash smart contract, which enhances transaction privacy by removing the on-chain connection between source and destination addresses, serves as the foundation for the application that we suggest in this paper. Tornado Cash is a coin-mixer that employs a number of smart contracts that take deposits in varying amounts of ETH and ERC-20. A relayer can be used to withdraw funds to an address with no ETH balance in order to maintain privacy (i.e., a fresh address). The new address ensures perfect secrecy by making it impossible to connect any ETH withdrawals to deposits. We use this approach to enforce the election’s privacy by breaking the link between the voter and the voted candidate. How the election is going, that is the number of votes each candidate is receiving before the closing, represents indeed sensitive information that could impact the final outcome, being the Ethereum blockchain public and thus readable by anyone. Confidentiality during the election is implemented by using asymmetric cryptography. Moreover, the results are encrypted by all candidates’ public keys. In this way, they are encrypted until the end of the election, when all candidates reveal their private keys. So everyone can see the results only after the private keys revealing.

We elaborate on the first preliminary results proposed in [2], by enforcing the confidentiality of results during voting. The paper has the following structure: after this introduction, Sect. 2 overviews the necessary background information, such as Ethereum, the ERC20 token standard, some important e-voting properties, and finally, Tornado Cash. Then, Sect. 3 and Sect. 4 respectively describe the pre-voting (with the voting distribution phase) and the actual voting procedure. Section 5 describes the desired properties an e-voting system should respect. Section 6 reports similar works in the literature, while Sect. 7 draws 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.

---

<sup>1</sup>“Why paper is considered state-of-the-art voting technology”, Raj Karan Gambhir and Jack Karsten Wednesday, August 14, 2019: [shorturl.at/cuA37](https://shorturl.at/cuA37).

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 [3], which implements different features with 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.<sup>2</sup> 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 connect a smart contract with a front-end user interface.<sup>3</sup> 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 corresponds 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 well-known non-fungible tokens (NFTs) are not interchangeable.<sup>4</sup> 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 the spender will be allowed to spend on behalf of the owner.

---

<sup>2</sup>Smart contracts: <https://github.com/ethereum/ethereum-org-website/blob/dev/src/content/developers/docs/smart-contracts/index.md>.

<sup>3</sup>dApps: <https://github.com/ethereum/ethereum-org-website/blob/dev/src/content/developers/docs/dapps/index.md>.

<sup>4</sup><https://www.investopedia.com/news/what-erc20-and-what-does-it-mean-ethereum/>.

- *Transfer*: an event triggered when a transfer is successful.
- *Approval*: a log of an approved event.

### 2.3. Important voting properties

In order to be considered secure, an electronic voting system must perform in a variety of situations without information flow or vulnerabilities, and be properly maintained and updated. Notwithstanding the difficulty of the issue, some standards for election systems appear to be universally acknowledged as essential requirements. A good voting (and also e-voting) system can be characterized by some properties [4, 5, 6]:

- *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.

The core requirements that are desirable in any election system are: *i*) Confidentiality, *ii*) Integrity, *iii*) Authentication, and *iv*) Verifiability [4, 5, 6].

### 2.4. Tornado Cash

Tornado Cash is an open-source, fully decentralized non-custodial<sup>5</sup> 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 token deposits from one address and enables their withdrawal from a different address. Tornado Cash 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 *TORNs* (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 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),

---

<sup>5</sup>Non-custodial wallet services are platforms that allow users to possess their private keys.

- 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<sup>6</sup> 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  $C = H(k||r)$ .<sup>7</sup> 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  $N$  tokens and add  $C$  as a leaf of a tree, in case the tree is not full.<sup>8</sup>

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*<sup>9</sup> 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 uses a *relayer* to make the withdrawal to any Ethereum address without needing to make the wallet connection on the Tornado Cash website. 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.<sup>10</sup>

### 3. Pre-voting phase

In this paper, we extended the model presented in [2], increasing his properties. To better understand our model, in the next sections, we refer to the various actors as follows:

- *Account1*: Ethereum account with ether and possibly other tokens. Not anonymous, linked to a voter.
- *Account2*: Ethereum account without any related transaction, which is not linkable to the voter's identity.
- *Admin*: The organizer of the election.

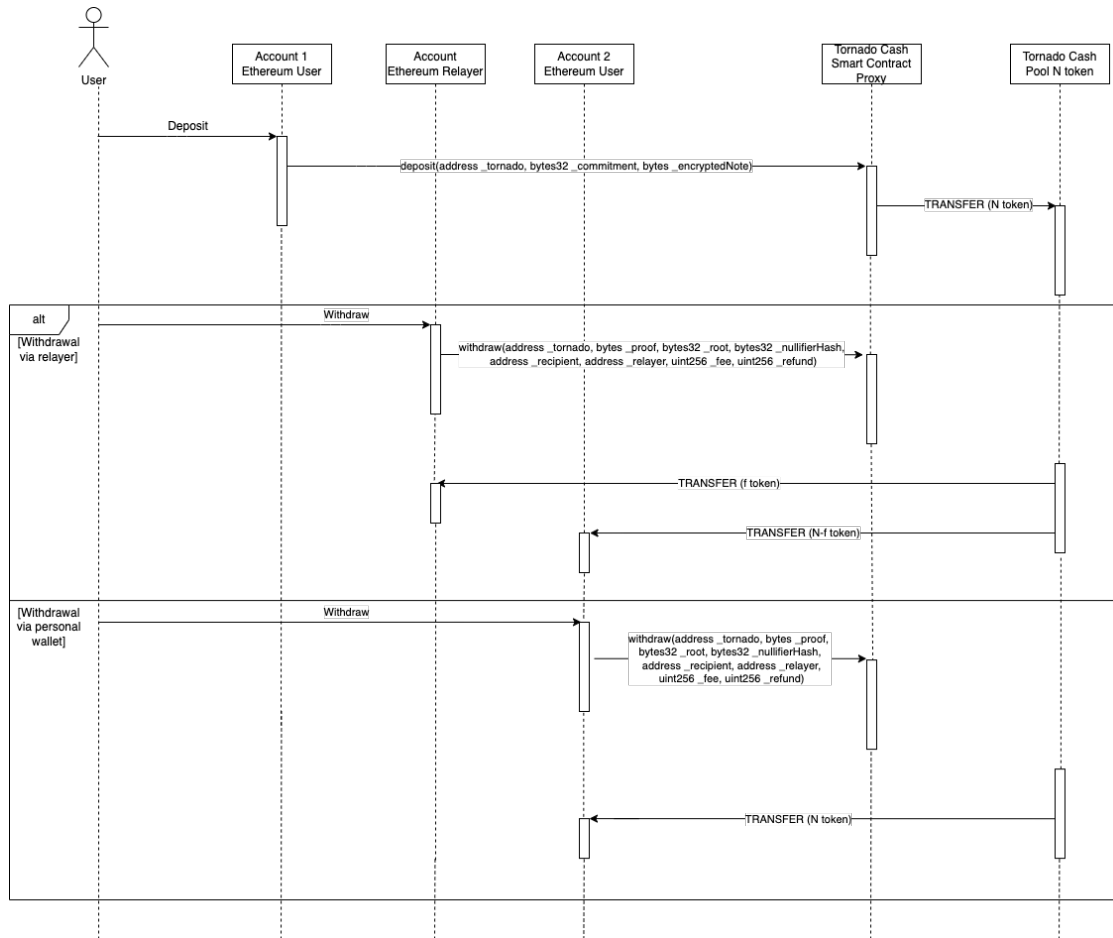
<sup>6</sup>Web-connection of Tornado Cash to Metamask or a private wallet: <https://tornadocash.eth.limo/>.

<sup>7</sup>With  $||$  that stands for concatenation.

<sup>8</sup><https://tornado-cash.medium.com/introducing-private-transactions-on-ethereum-now-42ee915babe0>.

<sup>9</sup>zk-SNARK: <https://z.cash/technology/zksnarks/>.

<sup>10</sup>Goerli Testnet Network: <https://goerli.etherscan.io/address/0x454d870a72e29d5e5697f635128d18077bd04c60>.



**Figure 1:** Sequence diagram: Deposit and Withdrawal operations.

- SCP: Smart contract pool.

Using a smart contract, the admin will first distribute an ERC20 token (DVT, or "DevToken"). The DVT will then make voting possible for users. The administrator then decides on a deadline for users to register in order to vote (and consequently receive 1 DVT). Only the Ethereum account that issued the DVT token will be able to mint and distribute new tokens because that account will hold the smart contract for that token. In the meantime, the Admin also registers all the candidates. Every candidate will provide a public key (related to a private key they own) that will be used to encrypt the votes in the voting step.

The users must also deposit 1 DVT and 0.0015 ETH (which is now equivalent to about €1.5) to the appropriate SCP without withdrawing the token before the deadline. Those who wish to cast a vote may proceed to withdraw their deposited tokens once the deadline has passed. Relayer is used to transfer 0.0015 ETH to a new Ethereum wallet (Account2). This helps to pseudo-anonymize the user by preventing in/out transactions. The user then takes 1 DTV on the same wallet (Account2). Finally, the Admin releases the voting dApp websites allowing

users to vote by sending the DVT token to the smart contract dedicated to voting.

The administrator makes a smart contract to use a new ERC20 coin (DTV). Voting rights are preserved by using it. It is deployed using the Hardhat<sup>11</sup> development environment on the Goerli test network (also known as *testnet*). Just the Ethereum account that launched the smart contract is the contract's owner (Admin) and can create additional tokens. The smart contract and the Ethereum account that own the tokens can be used to regulate the number of tokens that are in use. The admin additionally authenticates the user before receiving one DTV token. Admin develops a web page with a form so users can authenticate. The user enters their information in this form. The administrator then confirms that the data are accurate and that the user still needs the DTV token. If everything is correct, the user gets authorized and receives 1 DTV token. Due to its decentralized structure, the smart contract can regulate the number of created tokens and the Ethereum accounts that hold them.

We extend the Tornado Cash protocol in our smart contract to ensure user pseudo-anonymization. Moreover, we specifically incorporate the use of relayers for ETH and DTV deposits and withdrawals. 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 for ETH and DTV deposits and withdrawals. Moreover, the SCP permits users to withdraw from Account2 and deposit from Account1. In order to maintain their pseudo-anonymity, the user must perform their deposit and withdrawal activities via SCP before voting.

After completing the deposit, the user receives a confidential message in the output that he/she must memorize to withdraw the funds subsequently. Admin establishes an expiration date for making deposits, as was already mentioned. After that time has passed, the voter removes ETH from Account2 to cash out. To pay the fee for this transaction, they can not use Account1 (if the user does so, this will create a correlation between Account1 and Account2, and thus pseudo-anonymity would be lost) nor Account2 (this one does not have any ether). Consequently, to cover the charge for an ETH withdrawal transaction, the Admin must configure an Ethereum account called *relayer*. To configure the relayer, we refer to Tornado Cash's *tornado-relayer* repository on GitHub<sup>12</sup>. In particular, we extend it with the possibility to operate in Goerli with our SCP and to modify the transaction gas and fee value.

Figure 2 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 to break their connection with the coin, i.e., 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*).

---

<sup>11</sup><https://hardhat.org/>.

<sup>12</sup><https://github.com/tornadocash/tornado-relayer/tree/c838316436a9f87f8655087c34764b46e4b1491b>.

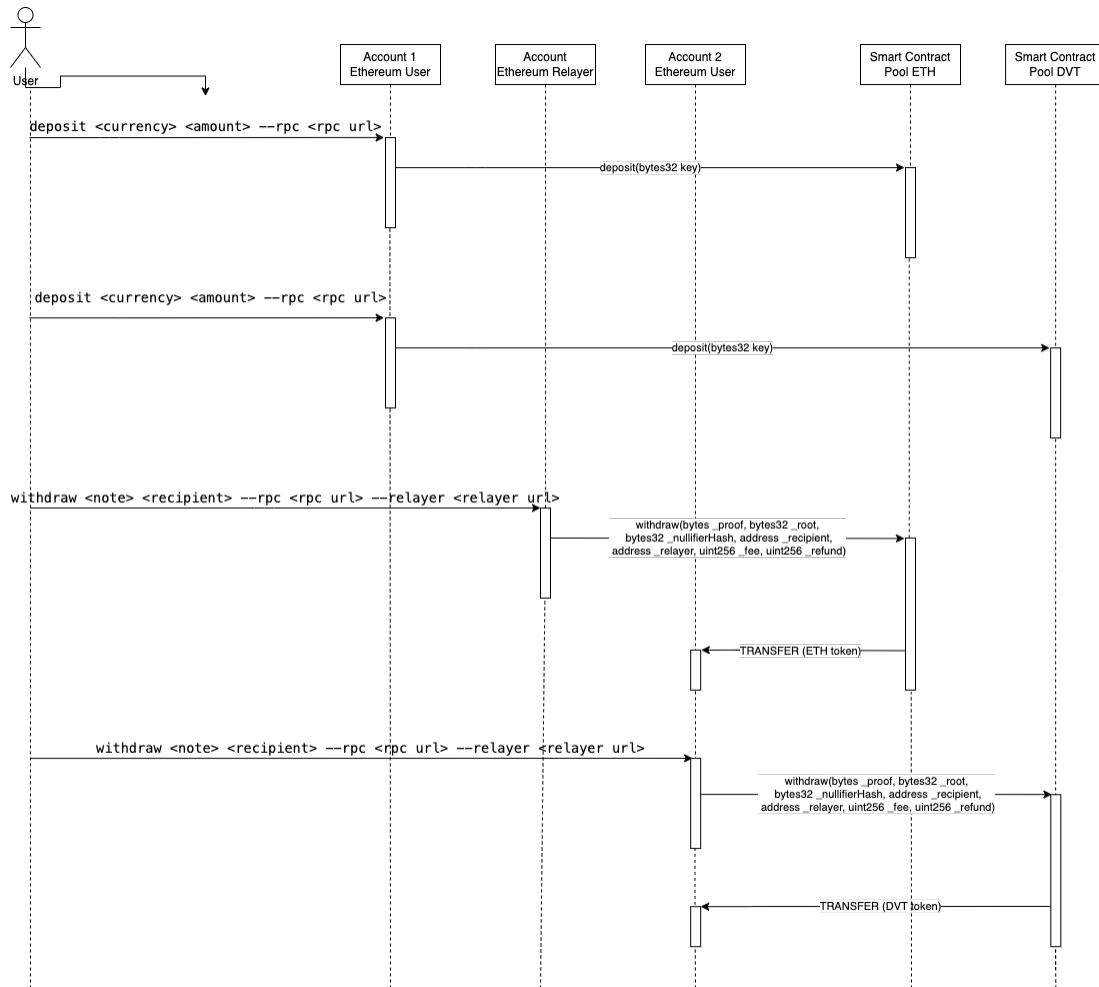


Figure 2: Sequence diagram: interaction via command line tool.

## 4. A confidential voting phase

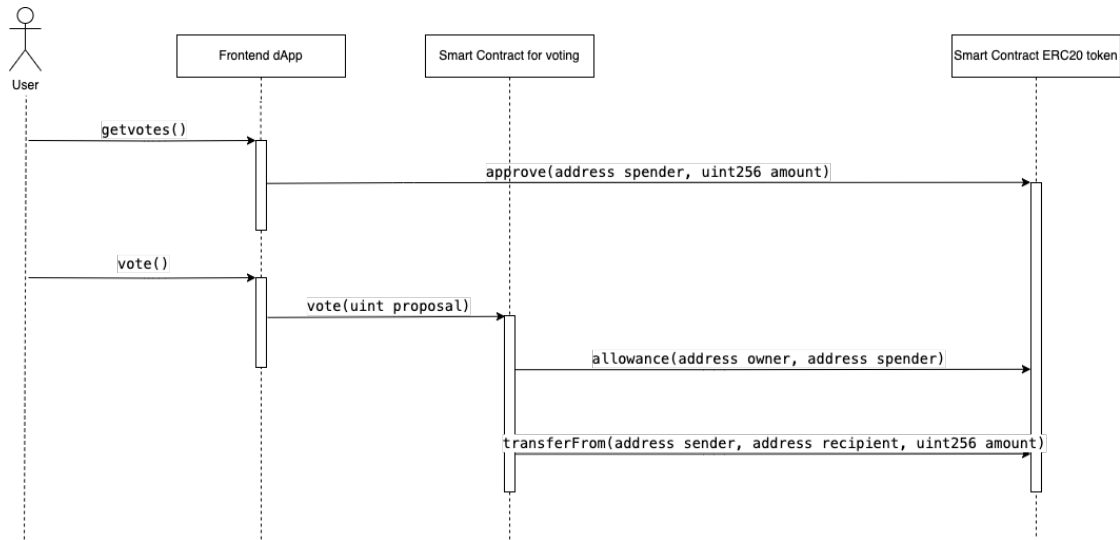
Having performed the withdrawal transaction, the user has 0.0015 ETH and 1 DTV in their Account2. ETH value is to pay transaction fees and the DVT, instead, is for voting. At this point, the admin creates a web dApp based on a smart contract. The contract can be created using the constructor:

```
CONSTRUCTOR(String[] MEMORY PROPOSALNAMES, uint256 TOKENQUANTITY_, address TOKEN)
```

The smart contract constructor uses:

- 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 number of tokens needed for each user to be able to vote;





**Figure 3:** Sequence diagram: user voting.

- 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. The function is used to send the DTV token to a specific address<sup>13</sup> and store in the smart contract the encrypted selected candidate. The encryption is done using all the candidates' public keys; in this way, the election results will be available only when all the candidates reveal their private keys. In particular we used these scheme:  $EncryptedVote = (... (candidateName + randomNonce)_{candidatesPubKey_0} \dots)_{candidatesPubKey_n}$ . The candidate's name concatenated with a random nonce<sup>14</sup> is encrypted with all the public keys of the candidates. Finally, the Admin implements a web application that interacts with the smart contract of voting (Fig. 3). 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. 3). 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 must 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. The user's vote will be stored in the Blockchain and encrypted using the candidates' public keys.

<sup>13</sup>Since the vote is stored in the blockchain is not necessary to send the token to different addresses according to the selected candidate.

<sup>14</sup>Notice that the nonce is used to avoid an attacker, simply encrypting all the candidates' names, could decrypt the votes

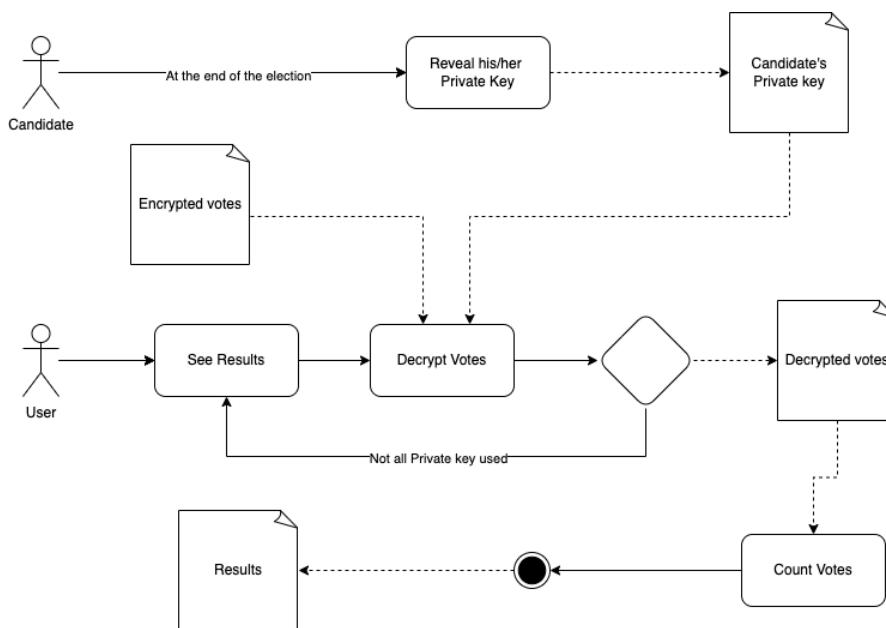


Figure 4: Activity diagram: Voting count.

Once this transaction is mined, the vote will be finalized. At the end of the election, as shown in Figure 4, all the candidates will reveal their private keys corresponding to the public keys used to encrypt the votes. With all the private keys, everyone can check the election results. In fact, using the revealed private keys, users could easily decrypt all the votes. Once all votes are decrypted, the admin can perform the count and share the results that everyone can check.

## 5. Satisfied Properties

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

**Verifiability:** Ethereum blockchain is public, and so after the candidates reveal their private keys, it is always possible to verify that the counting of votes has been performed correctly.

**Uniqueness:** Double-voting is prevented because double-spending is impossible with blockchain technology [3]. Moreover, every user receives just one DTV token to vote.

**Integrity:** When a transaction is in a confirmed block, modifying that block is computationally hard by design [3] since it is also required to modify all the successive blocks. Moreover, changing or deleting a transaction (vote) without revealing it is impossible.

**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 transactions; therefore, it is not possible to identify the voter.

**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. Only users with a DTV token can vote.

Notice that we know it could be possible to send the DTV token to another person using Tor-nado cash. However, we designed this protocol to be used in small elections, e.g. representatives in companies. Therefore, we think that vote trading is a minor problem in this kind of election.

**Confidentiality:** It is implemented by using asymmetric cryptography. The results are encrypted until all candidates reveal their private keys. In this way, they are encrypted until the end of the election. So everyone can see the results only after the private keys revealing.

**Lack of evidence:** Users revealing the possession of their Account2 can prove for whom they voted, but they could do it only after the end of the election, because before it is impossible to determine for whom they voted.

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

## 6. Related Work

Nowadays, the most spread voting schemes consist of paper-based elections. However, paper-based systems are not completely secure and they may suffer from fraud, even in today's democratic countries,<sup>15</sup> where controversies are very frequent.<sup>16</sup> Estonia became the first nation to hold general elections over the Internet with a pilot project for municipal elections in 2005. The e-voting system withstood the test of reality and was declared a success by Estonian election officials [7]. 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*,<sup>17</sup> which already offers a voting platform based on Bitcoin. However, the software is offered as a broker between the voter and Bitcoin. 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 a combined cryptocurrency wallet. The vote is sent to a smart contract-based election holding yay, nay, and candidate 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 decentralized 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*<sup>18</sup> and *TIVI*<sup>19</sup>. *Semaphore*<sup>20</sup> is an Ethereum-based

<sup>15</sup>[http://news.bbc.co.uk/2/hi/uk\\_news/4410743.stm](http://news.bbc.co.uk/2/hi/uk_news/4410743.stm).

<sup>16</sup><http://news.bbc.co.uk/2/hi/europe/4904294.stm>.

<sup>17</sup>Web-site of the Bitcongress.org project: <http://www.bitcongress.org>.

<sup>18</sup>FollowMyVote.com: <https://followmyvote.com>.

<sup>19</sup>TIVI: <https://tivi.io>.

<sup>20</sup><https://docs.zkproof.org/pages/standards/accepted-workshop3/proposal-semaphore.pdf>.

architecture for zero-knowledge signalling. It enables users to publicly declare their support for any string without disclosing their identity or needing permission. It is meant to be used as a base layer for signaling-based applications, mixers, and anonymous DAOs.

Envisioning the use of blockchains for voting purposes has been already proposed in [9, 10, 11] for example. In the following, we present related scientific works. The proposal in [12] simply consists of an electronic voting system based on the Bitcoin blockchain technology. In [13, 14] the author proposes 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 [15] 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 [16] is enforced by using the consensus mechanism that also secures the Ethereum blockchain.

All of them propose solutions without the help of colored coins or permissioned ledgers, which have been used to respectively simplify the counting process and satisfy different properties, as shown in Sect. 2.3: properties as data confidentiality and lack of evidence seem not to be addressed in all such proposals; with *MultiChain*<sup>21</sup> [17], or in general permissioned ledgers, it is possible instead. Unfortunately, to do so, they lost the high reliability of a system like Bitcoin or Ethereum and full decentralization.

In the literature, several confidentiality-safe remote e-voting solutions based on blockchain technology have been proposed [18]. Below we describe and compare them with ours.

In [19], authors proposed an end-to-end verifiable, confidentiality-safe, and secure Blockchain-based online e-voting protocol (LOKI). LOKI Vote is based on Monero<sup>22</sup>, a public distributed ledger with privacy-enhancing technologies that obfuscate transactions to achieve anonymity and fungibility. Like us, they encrypted all votes using a group of public keys belonging to Tallying authorities (TAs). These TAs, unlike our model, are trusted people outside the election.

The work in [20] presents a protocol that stores all submitted votes in a blockchain database, which all users can access. Each vote is encrypted before submission and remains encrypted at all times. Moreover, the eligibility of voters and their submissions can be verified by anyone without revealing the contents of the votes. The proposed verification and self-tallying algorithms allow any voter to verify the correctness of the final result. Unlike our protocol, this one does not guarantee the voters' privacy: everyone can see the sender of each vote in the blockchain.

The authors of [21] show a model many times more complex and computationally expensive than ours to ensure confidentiality. Each vote is represented as a distribution, divided into many parts, and distributed to many randomly selected peers. Each part can be verified independently, and a small collection of these does not reveal the overall vote preference. Tallying of votes is distributed among all peers, with each peer tallying a small portion. Each peer's tally can be verified and corrected, and all correct partial tallies will be aggregated to get the final result.

In [22], we can see a smart contract implementation for the Open Vote Network that runs on Ethereum. The authors show that the implementation can be used with minimal setup for

---

<sup>21</sup>MultiChain is a bridging platform for cryptocurrencies and NFTs across blockchains.

<sup>22</sup><https://www.getmonero.org/>.

elections. The main problem of this implementation is that it works just with elections with two options, e.g., yes or no.

Finally, the work in [16] proposes a blockchain-based voting system that ensures voters' privacy and voting correctness by homomorphic encryption, linkable ring signature, and proofs of knowledge (PoKs) between the voter and blockchain. However, the confidentiality of the protocol is guaranteed just by the administrator, that knows the private key to encrypt the votes. So he/she could check the result at any time.

## 7. Conclusion

This paper presents the creation of an electronic voting system based on the Ethereum blockchain, employing the Tornado Cash coin-mixer protocol to guarantee voter privacy, smart contracts to guarantee process transparency, and asymmetric cryptography encryption to preserve confidentiality.

First, we created a Web application that authenticates the user before delivering an ERC20 token symbolizing a ballot. Then, using the Tornado Cash protocol as a guide, the following steps were taken: we created smart contract pools to enable the deposit and withdrawal of DVT and ETH; we created an application to enable user interaction with the aforementioned smart contracts; we used a relay to withdraw ETH on behalf of the authorized user; we designed a dApp to collect and encrypt the votes as a last step. The votes are encrypted using all candidates' public keys. In this way, they are encrypted until the end of the election. So everyone can see the results only after the private keys revealing.

In the future, we would like to implement a Web dApp, which allows users to connect their MetaMask account and interact with the smart contract pools by performing deposits and withdrawals. We would also like to simplify the vote encryption part of the system, for example, using a distributed public key [23]. This will allow generating a public key using multiple private keys so that we may have just one public key corresponding to all the candidates' private keys. Moreover, we could use just one public key to encrypt all the votes. In addition, we would like to enforce *Lack of evidence* (see Sect. 5); one option could be using some obfuscation procedure on the code of the smart contract [24], for example. Finally, we want to incorporate stronger authentication and authorization mechanisms like OAuth and OpenID into the application. Another way could be using Self-Sovereign Identity to verify the credential and avoid Admin ad point of trust. Moreover, paying something to call a smart contract function that performs authentication will also reduce DoS attacks.

## Acknowledgement

The authors are members of the INdAM Research group GNCS and Consorzio CINI. This work has been partially supported by: INdAM - GNCS Project, CUP E53C22001930001; 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 FICO: funded by Ricerca di Base 2021, University of Perugia; Project GIUSTIZIA AGILE, CUP: J89J22000900005.

## References

- [1] C. Stewart III, Voting technologies, *Annual Review of Political Science* 14 (2011) 353–378.
- [2] S. Bistarelli, B. L. La Torre Montalvo, I. Mercanti, F. Santini, An e-voting system based on tornado cash, in: A. Saracino, P. Mori (Eds.), *Emerging Technologies for Authorization and Authentication*, Springer Nature Switzerland, Cham, 2023, pp. 120–135.
- [3] V. Buterin, *Ethereum white paper: A next generation smart contract & decentralized application platform* (2013). URL: <https://github.com/ethereum/wiki/wiki/White-Paper>.
- [4] L. Fouard, M. Duclos, P. Lafourcade, 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].
- [5] C. D. Mote, Report of the national workshop on internet voting: issues and research agenda, in: *Proceedings of the 2000 annual national conference on Digital government research*, Digital Government Society of North America, 2000, pp. 1–59.
- [6] A. Schneider, C. Meter, P. Hagemeister, Survey on remote electronic voting, arXiv preprint arXiv:1702.02798 (2017).
- [7] R. M. Alvarez, T. E. Hall, A. H. Trechsel, Internet voting in comparative perspective: the case of estonia, *PS: Political Science & Politics* 42 (2009) 497–505.
- [8] L. Loeber, D. E. Council, E-voting in the netherlands; from general acceptance to general doubt in two years, *Electronic voting* 131 (2008) 21–30.
- [9] S. Omohundro, Cryptocurrencies, smart contracts, and artificial intelligence, *AI matters* 1 (2014) 19–21.
- [10] M. Pilkington, *11 blockchain technology: principles and applications*, *Research handbook on digital transformations* (2016) 225.
- [11] M. Swan, *Blockchain: Blueprint for a new economy*, "O'Reilly Media, Inc.", 2015.
- [12] A. B. Ayed, A conceptual secure blockchain-based electronic voting system, *International Journal of Network Security & Its Applications* 93 (2017).
- [13] F. Hardwick, R. N. Akram, K. Markantonakis, E-voting with blockchain: An e-voting protocol with decentralisation and voter privacy, *CoRR abs/1805.10258* (2018). URL: <http://arxiv.org/abs/1805.10258>.
- [14] B. Ahn, Implementation and early adoption of an ethereum-based electronic voting system for the prevention of fraudulent voting, *Sustainability* 14 (2022). URL: <https://www.mdpi.com/2071-1050/14/5/2917>. doi:10.3390/su14052917.
- [15] D. Pramulia, B. Anggorojati, 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)*, 2020, pp. 18–23. doi:10.1109/ICIMCIS51567.2020.9354310.
- [16] B. Yu, J. K. Liu, A. Sakzad, S. Nepal, R. Steinfeld, P. Rimba, M. H. Au, Platform-independent secure blockchain-based voting system, in: L. Chen, M. Manulis, S. A. Schneider (Eds.), *Information Security - 21st International Conference, ISC 2018*, Guildford, UK, September 9-12, 2018, *Proceedings*, volume 11060 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 369–386. URL: [https://doi.org/10.1007/978-3-319-99136-8\\_20](https://doi.org/10.1007/978-3-319-99136-8_20). doi:10.1007/978-3-319-99136-8\_20.
- [17] S. Bistarelli, I. Mercanti, P. Santancini, F. Santini, End-to-end voting with non-permissioned

- and permissioned ledgers, *J. Grid Comput.* 17 (2019) 97–118. URL: <https://doi.org/10.1007/s10723-019-09478-y>. doi:10.1007/s10723-019-09478-y.
- [18] A. Benabdallah, A. Audras, L. Coudert, N. E. Madhoun, M. Badra, Analysis of blockchain solutions for e-voting: A systematic literature review, *IEEE Access* 10 (2022) 70746–70759.
- [19] M. Chaieb, S. Yousfi, LOKI vote: A blockchain-based coercion resistant e-voting protocol, in: EMCIS, volume 402 of *Lecture Notes in Business Information Processing*, Springer, 2020, pp. 151–168.
- [20] X. Yang, X. Yi, S. Nepal, A. Kelarev, F. Han, Blockchain voting: Publicly verifiable online voting protocol without trusted tallying authorities, *Future Gener. Comput. Syst.* 112 (2020) 859–874.
- [21] W. Zhang, Y. Yuan, Y. Hu, S. Huang, S. Cao, A. Chopra, S. Huang, A privacy-preserving voting protocol on blockchain, in: 11th IEEE International Conference on Cloud Computing, CLOUD 2018, San Francisco, CA, USA, July 2-7, 2018, IEEE Computer Society, 2018, pp. 401–408. URL: <https://doi.org/10.1109/CLOUD.2018.00057>. doi:10.1109/CLOUD.2018.00057.
- [22] P. McCorry, S. F. Shahandashti, F. Hao, A smart contract for boardroom voting with maximum voter privacy, in: A. Kiayias (Ed.), *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*, volume 10322 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 357–375. URL: [https://doi.org/10.1007/978-3-319-70972-7\\_20](https://doi.org/10.1007/978-3-319-70972-7_20). doi:10.1007/978-3-319-70972-7\_20.
- [23] R. Gennaro, S. Goldfeder, Fast multiparty threshold ecdsa with fast trustless setup, in: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS '18*, Association for Computing Machinery, New York, NY, USA, 2018, p. 1179–1194. URL: <https://doi.org/10.1145/3243734.3243859>. doi:10.1145/3243734.3243859.
- [24] S. Suegami, Smart contracts obfuscation from blockchain-based one-time program, *IACR Cryptol. ePrint Arch.* (2022) 549. URL: <https://eprint.iacr.org/2022/549>.