Digital e-Voting with Blockchain Technology

Vivek Sharma **Alexander Hart** **Tzipora Halevi**
vsharma@gradcenter.cuny.edu alexanderhart97@gmail.com tzipora.halevi@brooklyn.cuny.edu

Department of Computer Science CUNY Brooklyn College

***Abstract*- The current system of pencil and paper for acquiring/verifying student signatures in regards to campus activity petitions is flawed. From long wait times to laggardness of adopting modern technology has made the current process seem lacking and has made the students desire an easier way. Through incorporating blockchain into the voting process, we are able to create a proof-of-concept for turning the analog process into digital.**

I. INTRODUCTION

The inspiration for creating a digital voting application was conceived last semester when Brooklyn College's two campus newspapers were in the process of merging [1] As part of the process, it was called for students to wait in line and sign their name and EMPLID to signify their support. It was during that time, I questioned if there was a more modern method that would achieve the same results.

II. PROPERTIES OF VOTING

When implementing voting in digital applications; it is important to preserve as many beneficial and expected properties [or features] as possible - such as those described below [2].

(i) Eligibility: Only allowing desired participants that meet certain criteria i.e. age, location, etc.

(ii) Privacy: The ability for a voter and their decision to be known only by the voter and no one else.

(iii) Immutability: A vote should have integrity such that the vote can not be changed by anyone [without proper consent]

(iv) Coercion resistance: No voter should feel compelled/forced to cast a vote in any specific way; it should be of their own decision-making.

(v) Forgiveness: The ability for a voter to alter their vote after they have casted it.

(vi) Verifiability: The ability for results to be transparent and factual, such that the public can verify the outcome of the event.

(vii) Physical verification: Physical verification is necessary to know whether the actual voter voted or the bogus person voted on the identity of another voter.

III. PROPERTIES OF BCEV

The implementation of BCeV took into consideration some of what could be considered the most important properties, i.e. eligibility, privacy, immutability, and verifiability.To elaborate - for eligibility; only those that had official Brooklyn College email addresses were able to register for an account. Furthermore, to protect the privacy of students; all the email addresses were encrypted with SHA-256 such that even the administrator(s) of BCeV

would not be able to take note of who exactly was voting. In regards to immutability, the fact that blockchain's nature is to be immutable handled that property - with the caveat that the ancillary database that stores certain information remains in-tact.

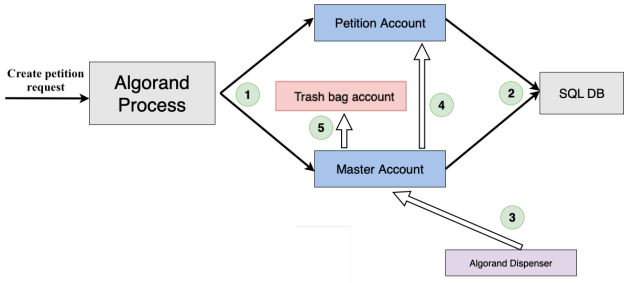
IV. DATA STORAGE

BCeV makes use of both traditional database and blockchain. Furthermore, encrypted user information is stored within SQL tables, in addition to petition account addresses on Algorand to be utilized by the blockchain. In regards to data storage on the Algorand network; JSON data representing the encrypted voter hash, timestamp and relevant petition address are all stored on-chain.

Figure I: Example of JSON structure stored in transaction Note

```
{
  "email": "YWRAbWluLmNvbQo..."
  "timeStamp": "12/05/19 14:30:09"
  "publicPK": "QA75IQ755GBY..."
}
```

V. SYSTEM ARCHITECTURE

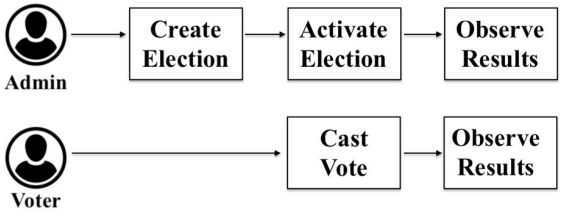


1. Algorand process creates two account addresses within the Algod node].
2. Insert the amount addresses for newly-created petition into SQL table.
3. Administrator manually uses Algorand dispenser to deposit tokens into master account
4. Transfer fixed predetermined amount of tokens from master account to petition account if a user votes YES for a specific petition - checking that the user hasn't already voted for the specific petition.
5. Transfer fixed predetermined amount of tokens from master amount to trash bag account when a user votes NO for any given petition, checking that the user hasn't already voted for the specific petition.

VI. CHALLENGES(S) AND SOLUTIONS(S)

One challenge was designing the voting process without the typical blockchain conventions, *e.g. smart contracts*; as seen in other networks like Ethereum. The solution implemented made use of a transaction "note field" that allowed 1KB of arbitrary data to be stored - in our case; JSON data that contain relevant information to the voting process. However, important to state the fact that our network of choice recently implemented smart contract functionality [3][4].

VII. ROLES



VIII. BLOCKCHAIN EVALUATIONS

	Algorand	*Bitcoin*	Ethereum	*HyperLedger*
Consensus Algorithm	PPoS	PoW	PoW	Various
Smart Contract	Yes	No	Yes	Chaincode
Smart Contract Language	TEAL	C++	Solidity	C++
Block Rate	5 seconds	10 minutes	10 seconds	Chaincode
Type	Permissionless	Permissionless	Permissionless	Permissionless/Permissioned

PPoS: *can tolerate malicious users, achieving consensus without a central authority, as long as a supermajority of the stake is in non-malicious hands. [6]*

PoW: *the algorithm rewards participants who solve cryptographic puzzles in order to validate transactions and create new blocks (i.e. mining). [7]*

Permissionless: *no permission is required to become part of this blockchain network and contribute to its upkeep. [8]*

Permissioned: *require permission to join. As a result, the owner of a permissioned blockchain has the ability to dictate who can and cannot become part of its network. [9]*

TEAL is a bytecode based stack language that executes inside Algorand transactions to check the parameters of the transaction and approve the transaction as if by a signature. [10]

VIV. EXISTING VOTING SYSTEMS

As pointed out by other academic surveys in relation to blockchain voting, we will show how our web application compares to the others:

	Polys	*Follow My Vote*	BCeV	*Bit Congress*
Eligibility	✔	✔	✔	✘
Anonymity	✔	✔	✔	✔
Verifiability	✔	✔	✔	✔
Integrity	✔	✔	✔	✔
Physical	✘	✘	✘	✘
Verification	✘	✘	✘	✘
Forgiveness	✘	✘	✘	✘
Coercion Resistance	✘	✘	✘	✘

Figure I: Example of a public transaction (Algorand)

TxID	**Status**	**Block**	**Amount**	**Fee**	**...**	**Note**
mxf..	Complete	3621..	10000	0.01	...	cXhs..
yzqd..	Complete	3622..	10000	0.01	...	kfasc..

X. Implementation

Below, is source code written in Python that checks if a voter has already participated in a petition:

```
def DoubleVoteChecker(curUser, txArray, yesNo):
  if yesNo == "Yes":
```

```
    if txArray is None:
       return False

    for i in range(0,len(txArray)):
      curTx   = txArray.get("transactions")[i]
      noteb64 = curTx.get("noteb64")
      note    = base64.b64decode(noteb64)
      email   = json.loads(note)["email"]
      if email == str(curUser):
         return True
         break
      return False
  elif yesNo == "No":
     if txArray is None:
        return False

     for key in txArray:
        email = key
        if email == str(curUser):
            return True
            break
        return False
```

Below, is pseudo code that cast a vote for a petition after the user clicks the appropriate button:

```
if vote button clicked:
       -Retrieve account address from
button value.

       -Retrieve transactions from
blockchain for given account address.

       -Retrieve transactions from
blockchain for trashbag account address.

       -Invoke DoubleVoteChecker method
with both sets of transactions and current
user email to confirm whether to proceed
with vote casting.

       -Create JSON structure with
appropriate data.

       -Create,sign,and send new
transaction to be appended to the
blockchain.

       -Redirect to [optional] feedback
form.
```

XI. USEABILITY TRIAL

As of this writing, a trial has been conducted on a sample size of thirty (30) subjects from a WhatsApp group chat. Out of the 30, nine (9) subjects took the time to actually trial the web application and participate in the optional feedback survey which is part of user design (UX). Furthermore, some subjects made comments directly to the developer stating their pleasure with the creation of the web app. See below for both instances of feedback; respectively.

Do you know what blockchain technology is?	Was BCeV easy to use?	Do you value privacy/anonymity when voting?	Do you prefer digital voting in comparison to pen and paper?	Do you think the future of voting should be similar to BCeV?

100% of the feedback was answered with "Yes"

"This is such a great idea!"
-November 18th, 2019

"This is amazing - I am so happy we have this!"
-November 18th, 2019

Additionally, in the future it could be a possible option to amend the web application to include traditional elections with multiple candidates. Perhaps utilize the

newly-added smart contracts in Algorand. Furthermore, in order to make our proof-of-concept less centralized in regards to only running a single node; it is possible to explore options that would increase the amount of nodes *i.e.* using services such as PureStake. [9]

XII. FUTURE SCOPE

In the future it could be a possible option to amend the web application to include traditional elections with multiple candidates. Perhaps utilize the newly-added smart contracts in Algorand. Furthermore, in order to make our proof-of-concept less centralized in regards to only running a single node; it is possible to explore options that would increase the amount of nodes *i.e.* using services such as PureStake. [9]

XIII. CONCLUSION

In a world that is increasingly becoming more dependent on technology to improve the process of nearly every cornerstone of society; voting is no exception. By designing, implementing, and testing our proof-of-concept - we were able to showcase that the analog can be transformed to digital via blockchain technology to make the petition process more convenient .

XIV. REFERENCES

[1] kingsmanbc118, Author: et al. “Spring 2019 Issue 12.” *The Kingsman*, 9 May 2019, https://kingsmanbc.home.blog/2019/05/09/spring-2019-issue-12/.

[2] Sayyad S.F. “Features of Blockchain Voting”

[3] Nelson, Danny. “Algorand 2.0's New Non-Turing-Complete Smart Contracts Are a Feature, Not a Bug.” *CoinDesk*, CoinDesk, 22 Nov. 2019, https://www.coindesk.com/algorand-2-0s-new-non-turing-complete-smart-contracts-are-a-feature-not-a-bug.

[6]Ethereum. “Ethereum/Wiki.” *GitHub*, https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ.

[7]https://www.algorand.com/Algorand%20Protocol.pdf

[8]Beedham, Matthew. “Here's the Difference between 'Permissioned' and 'Permissionless' Blockchains.” *Hard Fork | The Next Web*, 15 Nov. 2018, https://thenextweb.com/hardfork/2018/11/05/permissioned-permissionless-blockchains/.

[9]“Hyperledger Blockchain Performance Metrics White Paper.” *Hyperledger*, https://www.hyperledger.org/resources/publications/blockchain-performance-metrics#transaction-latency.

[10]“Algorand Services for Participants and Developers.” *PureStake*, https://www.purestake.com/technology/algorand-services/.

[11]“Transaction Execution Approval Language (TEAL).” *Transaction Execution Approval Language (TEAL) | Algorand Developer*, https://developer.algorand.org/docs/teal.

XV. SCREENSHOTS

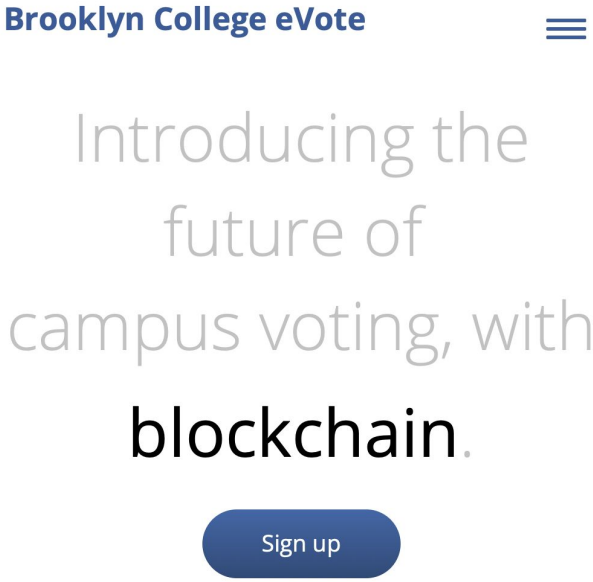


Figure 1: Landing page for non-signed users.

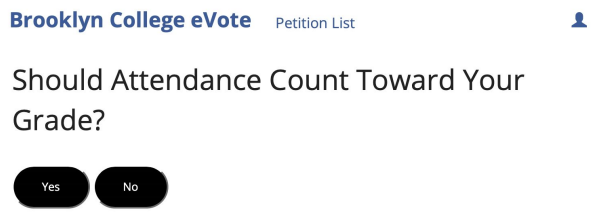


Figure 2: Petition list with vote buttons.

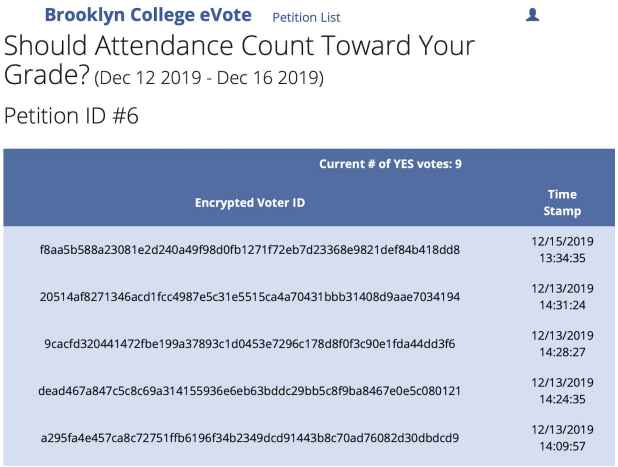


Figure 3: Vote details for specific petition; gathered from blockchain.