Blockchain Based Electronic Ballot System

Mohammed Sanaullah[1], Tanmayi D[2], Adhnan Manzis[3], Adithi Mahesh[4]

[1,2,3,4]*Atria Institute of Technology, Bengaluru, Karnataka, India*

Abstract – ***Ever since the advent of the Internet, broadcasting one's thoughts and opinions throughout the world was made possible. This newfound power gave rise to a plethora of applications powered by the web; Skype, Gmail, YouTube are a few prominent examples. The Internet as we know it began its evolution, beginning with web of cognition (web1.0) followed by web of communication (web2.0) and now web3.0 the web of decentralization. The Ethereum blockchain brought forth the idea of decentralized applications (DApps) and paved way for zero-trust, peer-to-peer systems. Online voting, a controversial subject of discussion earlier could now see the light of day due to the web3.0 mindset. The paper provides an online alternative to the ballot.***

Key Words: Electronic Voting, Ethereum, Ganache, Cryptography, Blockchain, Decentralized Application, Voting, Anonymity, Elliptic Curve Cryptography, Hashing, Smart Contract, Proof-of-work, Security, Privacy, Transparency, End-to-end verifiable, Transactions.

1. INTRODUCTION

Online voting is a popular yet controversial concept in the modern world. It decreases the cost of physical infrastructure and participation of voters increases because of the ease. It also allows people to vote from the comfort of one's own home instead of travelling to voting booths and standing in the queues for long hours. Although online voting has many benefits, it is not widely adopted among people since a single vulnerability can lead to a large-scale manipulation of votes. Online voting system should be accurate, reliable, transparent, legitimate, and votes should be kept secret and immutable. Blockchain technology helps solve quite a few of the issues such as true decentralization, immutability, unanimous, consensus etc.,[1] Thus, we have a viable reason to utilize the Ethereum blockchain as a reliable data store due to its support for RPCs along with features such as Smart-Contracts [2] and Proof-of-stake which makes it much more sustainable [3]. By keeping the application software open source, trust of users can be upheld whilst any minor tweaks or bugs that may seep through the cracks can be identified and resolved by the populous.

1.1 Contribution

The proposed system contains interaction between 4 major entities, namely:

- Client
- Server
- Host
- Blockchain

The Database (DB) used for this system is relational in nature and contains the following tables:

- User
- Election
- Candidate
- Participate

The smart-contract (SC) code is written in solidity using the Remix-ide and deployed on the Ganache blockchain. There exists a Class which consists of the following fields:

- encryptionKey (private, string)
- decryptionKey (private, string)
- voterArray (public, array of addresses)
- votes (public, dictionary (address, vote))

The methods of the Class are as follows:

- setEncryptionKey (private)
- setDecryptionKey (private)
- castVote (private)
- addVoter (private)
- showEncryptionKey (public)
- showDecryptionKey (public)
- showVote (public)
- showVoters (public)

The "private" access-specifier ensures that only the owner of the smart-contract can modify/access such fields, methods.

1.2 Organization

The rest of the paper is organized as follows: In Section II, we have the terminologies and basic definitions related to the scheme are mentioned along with a symbol-definition table for quick reference. We have discussed our implementation in Section III. In the next Section IV, we've put the results and discussion of this system. Section V contains the conclusion of this system. Section VI wraps up the paper with the acknowledgment.

2. Terminologies

The definition of symbols used in this paper are mentioned in the following table. Rivest, Shamir, Adleman asymmetric-key encryption (RSA) will be used for encryption/decryption of votes.

Table -1: Symbol Table

Symbol	Type	Definition
Ser	Entity	Server
Cli	Entity	Client
Hos	Entity	Host
Blo	Entity	Blockchain
USR	Table	User-Table-DB
PAR	Table	Participate-Table-DB
ELE	Table	Election-Table-DB
uid	String/Number	Unique-ID
ipu	Address	User-Public-Address
ipr	Address	User-Private-Address
eid	String/Number	Election-ID
sig	Variable	Signature
enc	Variable	RSA-Public-Key
dec	Variable	RSA-Private-Key
vot	Variable	Vote
vts	Dictionary	votes-SC
var	Array	voterArray-SC
tok	Random Integer	Verification-Token

3. Procedures

The following are the pre-requisites:

- Every election has its own SC.
- Meta-data of the respective SC must be stored in ELE.
- The field enc of every active election's SC must be initialized.
- The candidates participating in an election must be stored in a different table with their references present inside ELE.

3.1 User Registration

The Cli must first generate an Ethereum account (ipu, ipr) by using Meta-Mask [4], borrow from Ganache or generate using web3-library [5].

The next step begins with Cli sending (uid, ipu) to the Ser, if ipu doesn't exist in USR, (uid, ipu) is added to it. Else, the Ser throws an error to Cli stating that, "User already exists!".

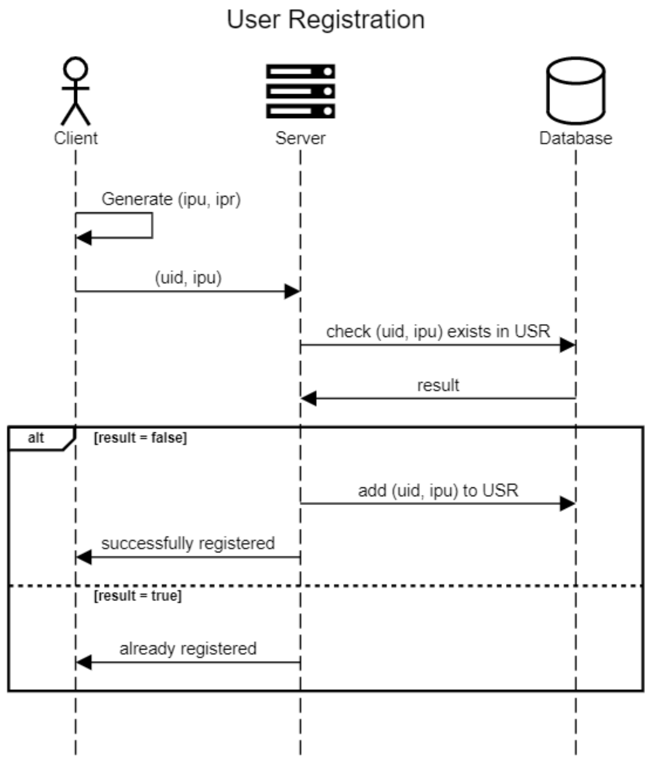

Fig -1: User registration sequence diagram

3.2 Voting

Cli first chooses an election and gets eid and its Hos. Cli signs uid with ipr to make sig [6] and sends (sig, uid) to the Hos. Hos extracts ipu from the pair [7] sent by Cli. Hos sends (ipu, eid) to Ser and waits for a response.

Ser first verifies whether ipu is registered by looking up USR. If ipu is registered, then it checks PAR for (ipu, eid) pair's existing. If the pair (ipu, eid) is unique and not existing in PAR, Ser will add the pair (ipu, eid) to PAR and sends "success" to Hos. If any of the conditions fail, the Hos is greeted with "failed".

Hos upon receiving "success" from Ser will proceed with adding ipu to var [8]. Hos will generate a random integer called nonce and records it as (nonce, ipu) [9]. Finally voting access is granted to Cli by sending it (enc, nonce).

Cli will choose the preferred candidate-id and encrypt it using enc [10]. The cipher generated is vot and the pair (ipu, vot, nonce) is sent to Hos.

Hos will verify nonce in its records, if it exists the corresponding ipu is extracted from its records and the entry is deleted [8]. Hos then proceeds to add (ipu, vot) as key-value pair in vts [11]. Finally, Hos sends "success" to Cli. If failure of any condition occurs, the Cli is greeted with "failure".

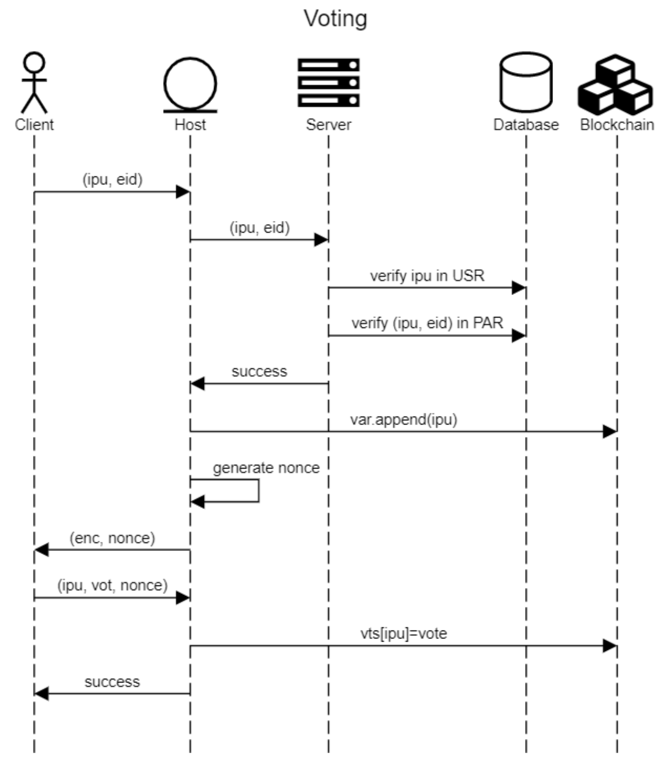

Fig -2: Voting sequence diagram

3.3 Voting Results

The respective election SC has certain methods with public access, utilizing these methods the list of participating voters can be fetched (showVoters). For each of these voters, their respective votes can be fetched (showVote). The value of dec if null results in displaying encrypted votes.

Thus, when Hos initializes dec equates to ending the election. Interested parties may use dec to verify results of the election, knowing the fact that data fetched or stored on the blockchain is immutable and non-fungible.

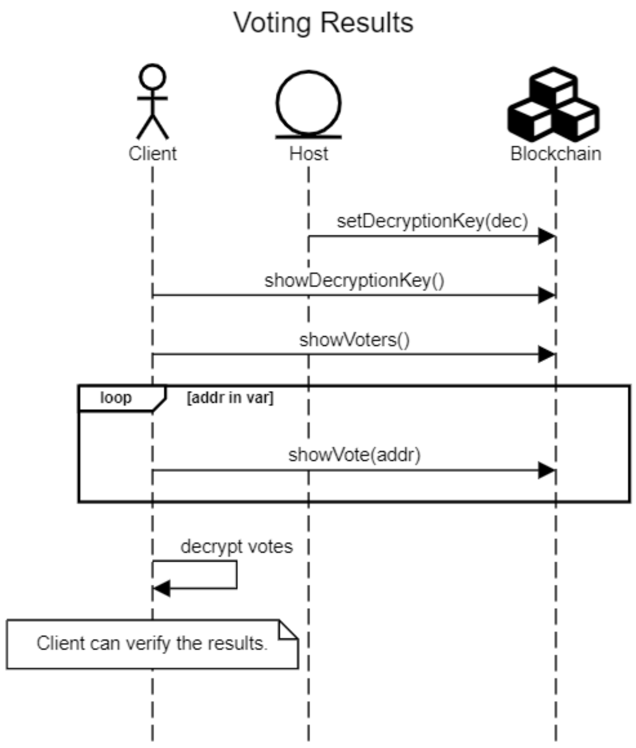

Fig -3: Voting results sequence diagram

4. RESULTS AND DISCUSSION

RSA is mathematically easy to understand and implement compared to other algorithms. Since RSA is relatively old [12], it has withstood the test of time ever since and has become a de facto standard for asymmetric-key encryption in modern times [12]. RSA is second only to ECC in terms of efficiency and security [comparison between ECC, RSA]. Due to lack of proper library or documentation for ECC based encryption compared to ECC based signatures, RSA was a much more enticing and suitable choice.

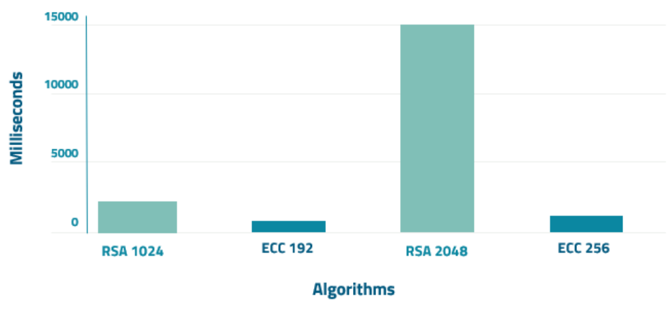

Chart -1: Key generation time of algorithms [13]

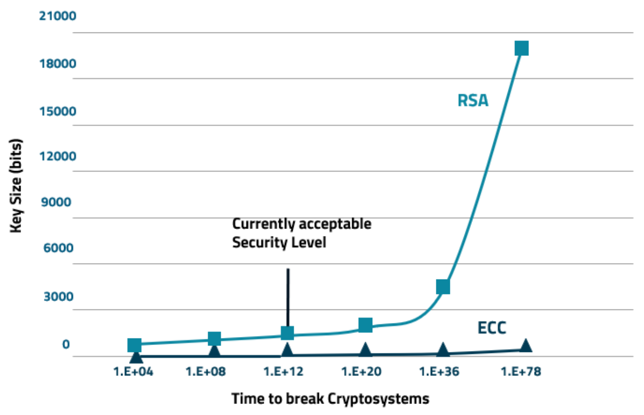

Chart -2: Compromise tolerance [13]

	Public Key Size			Key Size Ratio	Protection from
Security Bits	DSA	RSA	ECC	ECC to RSA/DSA	Attack
80	1024	1024	160-233	1:6	Until 2010
112	2048	2048	224-225	1:9	Until 2030
128	3072	3072	256-383	1:12	Beyond 2031
192	7680	7680	384-541	1:20	

Table -2: Security comparison [13]

5. CONCLUSION

The evidence is quite clear, blockchain approach is the best possible tool we have at present to resolve the challenge that is Online Electronic Voting System. The paper hopes to stand testament to this statement and provide ample evidence to the efficacy of blockchain and a good starting point for building much secure, scalable and efficient solutions. Ethereum 2.0 brings about much more promise by implementing proof-of-stake over the proof-of-work for consensus, which could have energy savings of up to 99% [].

6. ACKNOWLEDGEMENT

Special thanks and gratitude to Prof. G Srinivasachar at Atria Institute of Technology, who generously provided constructive discussions on paper and helped us with prototyping the application.

REFERENCES

[1] S.Nakamoto, “Bitcoin: A peer-to-peer electronic cash system”, 2008.

[2] Vitalik Buterin, “Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform”, 2014.

[3] Ralph Giles and Hannah Jones, “Proof of Stake - The Environmental Impact”, 2022.

[4] Joseph Lubin, “meta-mask: trailblazing tool enabling user interactions and experience on Web3.”, 2016

[5] Matthew Dangerfield, “web3js: The Ethereum JavaScript API which connects to the Generic JSON-RPC spec.”, 2014

[6] Web3.js, https://web3js.readthedocs.io/en/v1.7.4/web3-eth-personal.html#sign, 2014.

[7] Web3.js, https://web3js.readthedocs.io/en/v1.7.4/web3-eth-personal.html#ecrecover, 2014.

[8] Changxia Sun, Xia Zeng, et al., “Research and Application of Data Exchange based on JSON”, 2020

[9] Phillip Rogaway, “Nonce-Based Symmetric Encryption”, 2004

[10] Mohit Gupta, “RSA Algorithm in Cryptography”, 2022

[11] Web3.js, https://web3js.readthedocs.io/en/v1.7.4/web3-eth-contract.html#web3-eth-contract, 2014.

[12] Shireen Nisha, Mohammed Farik, “RSA Public Key Cryptography Algorithm – A Review”, 2017

[13] Parwinder Kaur Dhillon and Sheetal Kalra, “Elliptic curve cryptography for real time embedded systems in IoT networks”, 2016.