E-Voting Using Blockchain

Suraj Pratap Gupta and Ankur Mani Tripathi
Computer Science & Engineering, IIT(ISM) Dhanbad, Jharkhand, India

suraj.18MT0086@cse.iitism.ac.in
ankur95.18mt0456@cse.iitism.ac.in

Abstract. A vote gives a citizen of a democratic country the power to elect a representative which forms a government which is of the people, by the people, and for the people. Voting forms one of the fundamental pillars of modern democracy. Conducting a transparent, verifiable, and unbiased election is a challenging task for the election commission. Earlier ballot papers were used for conducting elections but it has to face many issues like booth capturing or damaging of ballot papers of the booths where there were fewer chances of winning for a candidate. Currently, EVMs are used for conducting the election in India but there are many reports that EVMs are not fully tamperproof. EVM is not a universal acceptance across the world. Blockchain is a distributed tamperproof technology which can help in conducting election transparent and tamperproof. Here in this paper, an approach for voting using blockchain and Paillier encryption is proposed. It implies the tamperproof property of blockchain and additive homomorphic property of Paillier encryption to build a voting architecture that will make the election process transparent and tamperproof. Next, algorithms for registration, voting, and result declaration has been mentioned along with the results of Paillier encryption for voting.

Keywords: E-voting , Blockchain , Paillier Encryption.

1. Introduction

Blockchain has been used in various areas like Supply Chain, Cryptocurrencies, Data Sharing, and other areas. Bitcoin is one of the famous cryptocurrency which utilizes blockchain for its creation and payment among their users. Confidentiality of data is one of the vital factors for the secure transmission of data. To achieve confidentiality encryption of data is used as one of the tool. Encryption of message is a method in which the message is converted from plaintext to ciphertext so that only the intended recipient can read the message. The intended recipient has to decrypt the ciphertext with the help of a shared key to read the encrypted message. There are two types of Encryption techniques used in cryptosystem: Symmetric key or secret key encryption and Asymmetric key or public key encryption. Symmetric key or secret key encryption uses
a single secret key for encryption purpose and decryption purpose like AES which is same key for both the process whereas Asymmetric Key or public key encryption uses a different pair of keys for encryption purpose and decryption purpose, i.e., a public key for encryption and a private key for decryption, like RSA or Paillier Encryption. In the election, a vote of the voter has to be confidential. To maintain this confidentiality voter’s vote is encrypted using Paillier homomorphic encryption. Even though two or more voters vote for the same candidate then also the cipher text of each voter choice will be different. Using the homomorphic property of Paillier Encryption the sum of votes can be calculated without revealing voter choice.

Blockchain is used to make the voting process temper proof. Blockchain is simply a chain of blocks connected with the help of hash which is calculated from previous block. The base block of the blockchain is a special block known as the Genesis block which is also the first block.

An architecture of blockchain for conducting the voting process is proposed. It is a three-step process involving Registration Setup, Voting, and Vote result calculation.

2. Theory

2.1 E-Voting

E-Voting is a term coined for electronic voting. It uses an electronic device for casting or counting of votes. In this proposed model, it is a computer connected through an intranet. An intranet is a type of private network which can only be accessed by authorized users. For e-voting authorized users can be the election commission officers and polling booth officers.

2.2 Homomorphic Encryption

Homomorphic Paillier Encryption [3] is a kind of encryption which allows computations to be carried out on encrypted text, ciphertext, that results in the same result like it is performed on a plaintext. If two or more ciphertexts are multiplied they will result in the addition of their plain text values. Partial homomorphic encryption allows only some computations to be carried out on ciphertext like addition, multiplication, etc.

2.3 Homomorphic Addition

If two ciphertext is multiplied they will result in the addition of their plain text values. The public key \((n, g)\) and private key \((\lambda, \mu)\) [3]. The two ciphertexts will
be decrypt as:

$$D(E(m_{1}, r_{1}) \cdot E(m_{2}, r_{2}) \mod n^{2}) = (m_{1} + m_{2}) \mod n$$

The ciphertext multiplied with a plaintext raised on g will decrypt as:

$$D(E(m_{1}, r_{1}) \cdot g^{m_{1}^{2}} \mod n^{2}) = m_{1} + m_{2} \mod n$$

2.4 Paillier Algorithm

Paillier cryptosystem follows homomorphic properties. Users can calculate the encrypted message without decrypting it. Let there be two messages \( (m_{1} \) and \( m_{2} \) with encryption converted to \( E(m_{1}) \) and \( E(m_{2}) \). The system will perform some computations on ciphertexts to obtain the data of encrypted messages, where \( T[4] \) can be calculated as

$$T = \prod_{i=1}^{n} C_{i} \mod n^{2}$$

To get the final result of the calculated data, the system just needs to decrypt \( T[4] \).

1. **Key Generation**: Paillier cryptosystem requires two keys namely public key and private key. The public key \( (n, g) \) is used for encryption and the private key \( (\lambda, \mu) \) is used for decryption. Steps for key generation are as follows:

   i. Choose two randomly different \( p \) and \( q \) which are two large prime numbers having: \( \gcd(pq, (p − 1)(q − 1)) = 1 \)

   ii. Calculate RSA modulus \( n = pq \) and compute Carmichael’s function as

   \[
   \lambda = \frac{(p − 1)(q − 1)}{\gcd(p − 1, q − 1)}
   \]

   iii. Select generator \( g \) where \( g \in Z_{n}^{*} \) such that:

   \[
   \gcd\left(\frac{g^{\lambda \mod n^{2} - 1}}{n}, n\right) = 1
   \]

   iv. Find multiplicative inverse \( \mu \) as:

   \[
   \mu = (L(g^{\lambda \mod n^{2}}))^{-1} \mod n
   \]

   where, function \( L \) is defined as \( L(k) = k^{-1} \mod n \)

After completion of all four steps mentioned above, the public key for encryption can be derived \( (n, g) \) and the private key for decryption as \( (\lambda, \mu) \).
2. *Encryption Process*: The encryption process using public key \((n, g)\) as follows:

i. Message to be encrypted is \(m\), where \(m \in \mathbb{Z}_n\)

ii. Random number is \(r\), where \(r \in n^2\)

iii. Cipher text can be calculated as:

\[
C = g^m r^n \mod n^2
\]

3. *Decryption Process*: The decryption process using private key \((\lambda, \mu)\) as follows:

i. Cipher text to be decrypted is \(C\), where \(C \in \mathbb{Z}_n^2\)

ii. To get plaintext \(m\) from ciphertext \(C\) by doing following computation:

\[
m = L(C^\lambda \mod n^2)^\mu \mod n
\]

where function \(L\) is defined as \(L(k) = k^{1 \mod n} - 1\)

2.5 *Blockchain*

Blockchain is simply a chain of blocks connected with the help of hash which is calculated from previous block. The base block of the blockchain is a special block known as the Genesis block which is also the first block. The blockchain is the platform to provide a decentralized way for computation and information sharing. The information sharing is more secure and is in a distributed manner with peer to peer network. The famous use of blockchain is bitcoin[1]. The bitcoin is a cryptocurrency which is used to exchange digital asset between individuals without intervening of the third party. Each transaction is protected through digital signatures. The blockchain is a peer to peer network with information shared is distributed over the network. Any information sharing or transaction done between any nodes will be recorded in the blockchain itself. So if in future it is required to verify the transaction then the blockchain can be viewed to check the validity of the transaction and also that transactions are also verified (that transaction is valid or not) before committing into the blockchain. As an example, Alice and Bob are two individuals, and Alice is sending 50 bitcoin to Bob over blockchain platform, after initiating the transaction from Alice of 50 bitcoin, the transaction created by Alice is submitted to the blockchain then this transaction will be verified by the nodes over blockchain if most of the nodes validate the transaction then the transaction will be accepted otherwise will be rejected. After accepting the transaction, the orderer will order the transactions (as there are many transactions) and will convert it into blocks and these blocks are mined by a miner and after mining the blocks the transaction is added into the blockchain. The node connected to the blockchain which helps in executing the transaction in return gets incentive is called a miner.
2.5.1. Block Structure: The proposed structure of a block as follows:

```c
struct block {
    blockHeight    int
    timestamp     int
    nonce         int
    targetDifficulty string
    hash          string
    previousHash string
    merkleRoot    string
    VoteCount     string
    ZoneId        int
    PreviousZoneBlock int
    TransactionCount int
    Transactions
}
```

2.5.2. Transaction Structure: The proposed structure of a transaction as follows:

```c
struct transaction {
    from        string
    to          string
    date        int
    time        int
    blockHeight int
    hash        string
    status      string
    vote        string
    ZoneId      int
}
```
2.6 Merkle Tree
Merkle tree[2] is a type of binary tree whose nodes represent a hash value. Hash of each transaction of a block is considered as leaf node. For calculation of a hash value SHA256 is applied twice and known as double SHA256. Merkle tree root is constructed by hashing a pair of nodes recursively until a single node is left known as Merkle Tree root as shown in figure 1. This Merkle Tree root is included in a block so that transaction verification can be done easily. Only a branch of Merkle Tree containing the transaction is required to verify that a particular transaction is present in the block or not.

Figure 1. Merkle Tree

3. Proposed Architecture
The architecture is proposed to demonstrate the e-voting process. Election Commission will set up Registration Booth, Mining Booth, and Polling Booth at different zones and subzones. One Registration Booth will setup per zone. One Mining Booth per district and per zone. One Polling Booth will be set up per subzone. Each district will be divided into various zones and each zone will be divided into many subzones. Polling booth will be setup at various subzones. Election Commission will choose different government schools as Polling Booths and provide various resources required for voting process.

3.1 Registration Setup
There will be one Registration Booth per zone per district. This booth will be responsible for the registration of candidates competing in the election, voter, polling officers, mining officers, and district officers. All registered officers will be provided with a login id and password. Election Commission is responsible for setting up Registration Booth, Pooling Booth, and Mining Booth. There will be one Registration Booth and Mining Booth per zone in every district. One Mining Booth will be set up at every district also. Mining Officers will be responsible for collecting voting transactions from Vote pool to form a block. After the successful mining of block, it can be added to the blockchain.
3.1.1. Candidates Registration: Candidates can be registered by providing their party ticket and other pieces of information like NID, name, address, etc. The candidate has to choose a password and a secret word. This secret word will be an answer to one of the security questions like What is your hobby?, What is your nickname? etc. Secret words will help the candidate to retrieve his password in case he forgets it. After a successful registration candidate will be allotted a candidate ID abbreviated as CID. CID and password can be used for login purposes. It will be the responsibility of the candidate to secure their secret words and password.

**Algorithm 1: Candidate Registration**

**Result:** Generation of candidate id, CID, and storage of Candidate Data into Dc

**Input:** National ID NIDᵢ, Nameᵢ, Addressᵢ, ZoneId ZIDᵢ, PartyId PIDᵢ, password passᵢ, seek words seekᵢ

**Initialization:** V Msg[ZIDᵢ][PIDᵢ], base b

1. HNIDᵢ = Hash(NIDᵢ);
   if HNIDᵢ not in Dc then
     2. Hpassᵢ = Hash(passᵢ);
     3. ENIDᵢ = Encrypt(NIDᵢ, Hpassᵢ);
     4. Esecretᵢ = Encrypt(passᵢ ⊕ seekᵢ, HNIDᵢ);
     5. Eᵢ = Encrypt(Nameᵢ, Addressᵢ, Hpassᵢ);
     6. pkᵢ, pri = peerdeploy();
     7. PeerAddressᵢ = getpeeraddress(pkᵢ);
     8. CIDᵢ = Hash(pkᵢ);
     9. V Msg[ZIDᵢ][PIDᵢ] = bPIDᵢ;
     10. EV Msgᵢ = Encrypt(V Msg[ZIDᵢ][PIDᵢ], PeerAddressᵢ);
     11. Dc ← Dc ∪ (ENIDᵢ, Eᵢ, CIDᵢ, Esecretᵢ, EV Msgᵢ, PeerAddressᵢ);
     return CIDᵢ, PeerAddressᵢ;
   else
     2. User already exist;
   end
Figure 2. Voting Process

Table 1. Description for Notations Used

| Notation   | Description                                      |
|------------|--------------------------------------------------|
| PeerAddressi | Unique address of user i                        |
| Encrypt()  | Encrypting with key                             |
| getpeeraddress() | Get the peer address                         |
| peerdeploy() | Deploy peer into blockchain                     |
| Hash()     | Hash of the data with SHA256                    |
| pk_i       | Public key of user_i                            |
| pr_i       | Private key of user_i                           |
| DC         | Candidate Database                              |
| DV         | Voter Database                                   |
| DP         | Polling Officer Database                        |
| DM         | Mining Officer Database                         |
| NZ         | Number of Zones                                 |
| Np         | Number of Political Parties                     |
| Nv         | Number of Voters                                |
3.1.2. Voters Registration: Each voter will visit the Registration booth for registration. At the registration booth voter has to provide his National ID like Aadhar Card along with data like name, mobile number, address, etc. After successful verification of National ID, his biometric will be captured and stored so that voter authentication can be done on the day of voting. The voter has to choose a password and a secret word. This secret word will be an answer to one of the security questions like What is your hobby?, What is your nickname ?, etc. Secret words will help the voter to retrieve his password in case he forgets it. After successful registration, a Voter ID is provided to the voter which he has to bring along with a password or secret word at the time of voting. The voter can vote by visiting the Polling Booth of his area. Security of passwords and secret words is the responsibility of voters himself. All these data will be stored in encrypted form in a distributed database i.e. Orbit database.

Algorithm 2: Voter Registration

Result: Generation of Voter ID, VID, and storage of Voter Data into \( D_v \)
Input: National ID \( NID_i \), Name\(_i\), Address\(_i\), password pass\(_i\), seek words seek\(_i\);

1. \( H_{NID_i} = \text{Hash}(NID_i) \);
   if \( H_{NID_i} \) not in \( D_v \) then
   2. \( H_{\text{pass}_i} = \text{Hash(\text{pass}_i)} \);
   3. \( E_{NID_i} = \text{Encrypt}(NID_i, H_{\text{pass}_i}) \);
   4. \( E_{\text{secret}_i} = \text{Encrypt(\text{pass}_i \oplus \text{seek}_i, H_{NID_i})} \);
   5. \( E_i = \text{Encrypt(\text{Name}_i, \text{Address}_i, H_{\text{pass}_i})} \);
   6. \( p_i, p_{ri} = \text{peerdeploy()} \);
   7. \( \text{PeerAddress}_i = \text{getpeeraddress}(p_{ki}) \);
   8. \( \text{VID}_i = \text{Hash}(p_{ki}) \);
   9. \( \text{VoteToken}_i = 1 \);
   10. \( D_v \leftarrow D_v \cup (E_{NID_i}, E_i, \text{VID}_i, E_{\text{secret}_i}, \text{VoteToken}_i, \text{PeerAddress}_i) \);
   return \( \text{VID}_i, \text{PeerAddress}_i \);
else
   2. User already exist;
end
3.1.3 Polling Officers Registration: Election Commission will provide a list of polling officers for registration. Polling Officer will provide details like NID, name, address, etc. for registration. These officers will be provided with a username and password after a successful registration. It will be the responsibility of officers to secure their usernames and password. These usernames and passwords will be used by the polling officer to log in at the polling booth. Polling officers will be distributed to various subzones of the district. These officers will be responsible for authentication and verification of voter at the polling booth.

3.1.4 Mining Officers Registration: Election Commission will provide a list of mining officers for registration. Mining officers will provide details like NID, name, address, etc. for registration. These officers will be provided with a username and password after a successful registration. It will be the responsibility of officers to secure their usernames and password. These usernames and passwords will be used by mining officers to log in to the mining center. One mining center will be set up at each zone of the district as well as at the district. Miner at the zone mining center will be responsible for collecting voting transactions from the voting pool to mine a block and add it to the blockchain.

3.2 Voting Process
On the day of voting Voter will visit the polling booth according to his name in the voting list along with his VID and NID. In the polling booth, the voter will be authenticated with the help of his biometric. After authentication voter will be forwarded for login to vote.

Algorithm 3: Polling Officer Registration

Result: Generation of Polling Officer ID, POID, and storage of Polling Officer Data into $D_p$
Input: National ID $NID_i$, Name$_i$, Address$_i$, password pass$_i$, seek words seek$_i$
1. $HNID_i = \text{Hash}(NID_i)$;
   if $HNID_i$ not in $D_p$ then
     2. $Hpass_i = \text{Hash}(pass_i)$;
     3. $ENID_i = \text{Encrypt}(NID_i, Hpass_i)$;
     4. $Esecret_i = \text{Encrypt}(pass_i \oplus \text{seek}_i, HNID_i)$;
     5. $E_i = \text{Encrypt}(\text{Name}_i, \text{Address}_i, Hpass_i)$;
     6. $pki, pri = \text{peerdeploy}()$;
     7. $\text{PeerAddress}_i = \text{getpeeraddress}(pk_i)$;
     8. $POID_i = \text{Hash}(pki)$;
     9. $D_p \leftarrow D_p \cup (ENID_i, E_i, POID_i, Esecret_i, \text{PeerAddress}_i)$;
        return $POID_i$, PeerAddress$_i$
   else
     2. User already exist;
end
Algorithm 4: Mining Officer Registration

Result: Generation of Mining Officer ID, MID, and storage of Mining Officer Data into Dm

Input: National ID NID_i, Name_i, Address_i, password pass_i, seek words seek_i

1. HNID_i = Hash(NID_i);
   if HNID_i in D_p then
      2. Hpass_i = Hash(pass_i);
      3. ENID_i = Encrypt(NID_i, Hpass_i);
      4. Esecret_i = Encrypt(pass_i ⊕ seek_i, HNID_i);
      5. E_i = Encrypt(Name_i, Address_i, Hpass_i);
      6. pk_i, pr_i = peerdeploy();
      7. PeerAddress_i = getpeeraddress(pk_i);
      8. MID_i = Hash(pk_i);
      9. TerminatingToken_i = 1;
      10. D_m ← D_m ∪ (ENID_i, E_i, MID_i, Esecret_i, PeerAddress_i, TerminatingToken_i);
          return MID_i, PeerAddress_i;
   else
      2. User already exist;
end

3.2.1. Voter Login: The voter will provide his VID and NID to the polling officer. Polling Officer will enter VID and NID into the system then a window will be displayed at voter end. He will enter his password if he remembers it and if the password is correct he will be login successfully. If the voter does not remember the password then he will choose the security question and will enter the answer. On the correct answer, voter will be login successfully.
Algorithm 5: Voter Login

Result: Generation of Session $S_i$
Input: National ID $NID_i$, password $pass_i$ or seek words $seek_i$;
1. $HNID_i = \text{Hash}(NID_i)$;
   if $HNID_i \in D_v$ then
     if Voter $i$ does not remember $pass_i$ then
       $pass_i = \text{Decrypt}(E_{\text{secret}_i}, HNID_i) \oplus seek_i$;
     end
   2. $Hpass_i = \text{Hash}(pass_i)$;
   if $(NID_i \equiv \text{Decrypt}(ENID_i, Hpass_i))$ then
     3. $pk_i, pr_i = \text{peerdeploy}()$;
     4. $VPeerAddress_i = \text{peerAddress}(pk_i)$;
     5. $S_i = 300$;
     6. return $(pk_i, pr_i, VPeerAddress_i, S_i)$;
   else
     3. Unauthorized Voter
   end
else
2. Voter does not exist
end

3.2.2 Vote: After a successful login, a window displaying all parties will be shown. The voter
will choose the desired party and click on the vote icon. The vote transaction will be submitted
to the voting pool. This vote will be digitally signed with voter private key $pr_i$ and voter vote
token value will be decremented and he will be marked as voted. Mining Officer will collect
voting transactions from the voting pool in every ten minutes and will form the block by using
proof of work. Miner will add VoteCount to block header from voting transactions of the block
which can be calculated as follows:

$$\text{VoteCount} = \prod_{i=1}^{N_v} C_i \mod n^2$$

This block will be propagated to all the miners and they will append it to their copy of the
blockchain after recalculating it to verify its value. If VoteCount value does not matches then it
will be added to Orphan pool. Miner at district
level will remine this block after calculating correct VoteCount value. After the completion of voting time, the miner at zone mining center will create a special transaction known as terminating transaction which will contain the block height of the last block mined by the miner and will add it to the transaction pool. This terminating transaction will be collected by a miner at the mining center at the zone level. This miner will mine a special block known as terminating block and will add it to the blockchain. The addition of this terminating block will signify the end of the voting process. This terminating block contains special transactions known as terminating transactions which contains the last block height of the mined block of each zone.

**Algorithm 6: Vote**

**Result:** Generation of Vote Transaction  
**Input:** Voter PeerAddress $V_{PeerAddress_i}$, Candidate PeerAddress $C_{PeerAddress_i}$, Session $S_i$;  
if ($S_i \equiv 300$) then
  1. SetTimer($S_i$);  
  if (VoteToken$_i == 1$) then
    2. $Msg_i = \text{Decrypt}(EV_{msg_i}, C_{PeerAddress_i})$;  
    3. $r_i = \text{random}()$;  
    4. $Vote_i = \text{PaillierEncryption}(Msg_i, r_i)$;  
    5. $Tx_i = \text{CreateTransaction}(V_{PeerAddress_i}, C_{PeerAddress_i}, Vote_i)$;  
    6. VoteToken$_i --$;  
    7. return $Tx_i$;  
  else
    2. Voter already Voted;  
  end else  
  1. Session Expired  
end

3.3 Voting Result

On the day of the result declaration, the representative of each party will be present at the mining center of the district. At the mining center, the result of each zone of the district will be calculated by the mining officer in presence of representatives of all parties. The mining officer will scan the terminating block and will find the terminating transaction of his zone. In the terminating transaction there will be the last block height of the block of the zone. The mining officer will provide this terminating transaction as input to the result calculation algorithm. The algorithm will calculate the result and will display the winning candidate of that zone.
**Algorithm 7: Voting Result**

**Result:** Declaration of Voting Result  
**Input:** Terminating Transaction \( T_{xi} \), Zone Id \( ZID_i \), base \( b \);  
**Initialization:** \( Result[Nz][Np] = 0 \);  
1. current block height = \( T_{xi}.LastZoneBlock \);  
   while current block height != 0 do  
   2. Go to current block height in \( B_L \);  
   3. next block height = current block height. PreviousZoneBlock;  
   4. \( CurrentVoteCount = CurrentVoteCount \times current\ block\ height. VoteCount \);  
   5. current block height = next block height;  
end  
6. \( N = \text{ConvertToBase}(CurrentVoteCount, b) \);  
7. max id = 0;  
8. max value = 0;  
9. c = 0;  
while \( N != 0 \) do  
10. \( r = N \mod 10 \);  
   if (max value > r) then  
       max value = r;  
       max id = c;  
   end  
11. \( Result[ZID_i][c] = r \);  
12. \( N = N/10 \);  
13. c++;  
end  
14. WinningPartyId = max id;  
15. WinningVoteCount = max value;  
16. return (Result[Nz][Np], WinningPartyId, WinningVoteCount);

---

**Figure 3. Voting Result**

| Voter Table | | | | |
|---|---|---|---|---|
| No. | Voter | VoteNo | RandomNumber | Encrypted Vote |
|---|---|---|---|---|
| 1. | A | 100 | 5112 | 1022005063 |
| 2. | B | 10 | 2865 | 12721503102 |
| 3. | C | 100 | 1434 | 3898663966 |
| 4. | D | 10 | 6465 | 33243394 |
| 5. | E | 10 | 1932 | 8725477329 |
| 6. | F | 100 | 72448 | 4642798975 |
| 7. | G | 100 | 41196 | 1816507353 |
| 8. | H | 100 | 76955 | 18024869710 |
| 9. | I | 1 | 28037 | 8097659087 |

Encrypted Vote Count = 36075674679  
Decrypted Vote Count = 1431
4. Result And Analysis

Experiment was performed on Paillier Algorithm by taking different number of voters and 3 candidates. The experiment was performed on a system having 2GB RAM and processor is Intel(R) Pentium(R) CPU G2020 @2.90GHz. The running time of algorithm is shown for different number of voters in figure 4. Figure 3 shows the result of encryption and decryption values of 9 voters and 3 candidates using Paillier Algorithm. Here, the candidates are named as P,Q,R and there is one option of NOTA, i.e., none of the above and Voters are named as A,B,C,...,I. VoteMsg shows the vote choice of each voter where LSB represents NOTA and MSB represents candidate P. Encrypted vote is calculated by providing VoteMsg and random number as an input to Paillier algorithm. After that Encrypted vote count is calculated by multiplying the encrypted votes. This value is decrypted to get the voting result. As shown in figure 3, Q got most numbers of vote. Hence Q wins.

5. Conclusion

A unique architecture for e-voting using blockchain is proposed in this paper. This approach ensures confidentiality of voter’s vote by the help of Paillier encryption. Blockchain tamperproof property makes it very difficult to alter vote.

References

[1] Bitcoin: A Peer-to-Peer Electronic Cash System Satoshi Nakamoto satoshin@gmx.com www.bitcoin.org Available at: https://bitcoin.org/bitcoin.pdf
[2] R.C. Merkle, "Protocols for public key cryptosystems," In Proc. 1980 Symposium on Security and Privacy, IEEE Computer Society, pages 122-133, April
[3] S. M. Anggriane, S. M. Nasution and F. Azmi, "Advanced e-voting system using Paillier homomorphic encryption algorithm," 2016 International Conference on Informatics and Computing (ICIC), Mataram, 2016, pp. 338-342.
[4] Sansar Choinyambuu," Homomorphic tallying with paillier cryoptsystem", HSR Hochschule für Technik Rapperswil, 2009.