Blockchain-based Electronic Voting System with Special Ballot and Block Structures that Complies with Indonesian Principle of Voting

Gottfried Christophorus Prasetyadi[1]
Graduate Program on Information System
Gunadarma University, Depok 16424, Indonesia

Achmad Benny Mutiara[2], Rina Refianti[3]
Faculty of Computer Science and Information Technology
Gunadarma University, Depok 16424, Indonesia

***Abstract*—Blockchain technology could be implemented not only in digital currency, but also in other fields. One such implementation is in democratic life, namely voting. This research focuses on designing a blockchain-based electronic voting system for medium to large-scale usage that complies with law, specifically voting principles in Indonesia. In this research, we proposed the following: a ballot design as block transaction employing UUID version 4, a modified block structure using SHA3-256 hash algorithm, and a voting protocol. The minimum length of a ballot is 43 bytes (excluding ECDSA signature) if one character is used as candidate's identifier and timestamp is stored as integer. We built a simulation program using Python-based Django web framework to cast 10,000 votes and mine them into blocks. Tampered transactions in each block could be detected and restored by synchronizing data with another node. We also evaluated the proposed system. By using this system, voters can exercise voting principles in Indonesia: direct, public, free, confidential, honest, and fair.**

***Keywords*—Blockchain; voting; design; simulation; Python**

I. Introduction

Information and communication technology is advancing rapidly. The performance and efficiency of Central Processing Unit (CPU) as the heart of a computer have continued to improve in the last few decades. Moore's Law, based on Gordon Moore's observation in 1965 and later adjustment in 1975, stated that the size of transistors were shrinking so fast that every two years, twice as many could fit onto a single computer chip [1]. This advancement has revolutionized many aspects in our social life and government. One such case that is going to be discussed in this study is voting.

Democratic countries, such as the Republic of Indonesia, guarantee the rights of their citizens to participate in decision-making, for example, to choose leaders by the mean of voting. By definition, voting (to vote) is "a formal indication of a choice between two or more candidates or courses of action, expressed typically through a ballot or a show of hands or by voice". In Indonesia, this right is listed on the state's constitution, namely Undang-undang Dasar Negara Republik Indonesia 1945 (UUD NRI 1945) article 28J paragraph 3: "everyone has the right to freedom of association, assembly, and issuing opinions".

Nowadays, voting process may be done electronically. Several electronic voting systems had been developed such as VOTAN (Votes Analyzer) for conducting electronic elections through the Internet securely. It is ideal for small communities such as organizations, universities and chambers [2]. It uses a centralized database, just like many other similar systems.

Centralized systems have common weaknesses. The data are stored centrally, so they have central point of failure, which can be exploited by computer crackers. Those systems are usually handled by single organization, so the data can be manipulated secretly by those who have administrative access to the database [3]. The recent development of blockchain technology can solve this problem.

The first work on cryptographically secured chain of blocks was published in 1991 in order to implement a system where documents' timestamps could not be modified [4]. In 2008, Satoshi Nakamoto, whose identity is still unknown, wrote about a "purely peer-to-peer version of electronic cash" known as Bitcoin [5]. Since then, blockchain made its public debut. Over time, people started to realize that blockchain could be used beyond cryptocurrency and they started to explore how blockchain could enhance many existing systems, including in voting process.

This study focuses on the design of several important components of the blockchain-based electronic voting system, and discusses the implementation of the proposed system for secure electronic voting to guarantee the rights of people, especially Indonesian citizens. The proposed system must follow the rules and principles recognized by the state.

This study is limited by the following. First, voters have to able to identify themselves using pseudonym. Second, the proposed system is intended for medium to large-scale usage, not small-scale (which often does not require costly effort). Third, node registration and public key storage are not discussed. Fourth, the simulation and testing are done on the local machine. Fifth and last, this study does not cover the solution for disabled people to access the system.

This paper is organized as follows. Section II contains comprehensive theoretical bases and proposed methods. Section III contains testing results and discussion. Section IV contains concluding remarks and possibilities of further improvements.

II. Related Works

Author in [3] proposed a blockchain-based system to record the results of elections using predetermined turn (flowchart shown in Fig. 1), instead of proof of work. Each node represented one voting place (called TPS/Tempat Pemungutan Suara) that produced one block containing one transaction comprising the sum of votes for each candidate. The researcher managed to generate 500,000 valid nodes in the simulation. However, it was not possible to identify and verify individual ballot, which this study tries to provide. Also, from their method, the predetermined turn system allowed only selected node (government-owned) to form the network.

Author in [7] proposed a protocol that employed blind digital signature and formulation of valid vote message, shown in Fig. 2. Their analysis did not involve simulation to prove their method, although they claim that the protocol can be implemented easily into Bitcoin network.

Author in [8] proposed a decentralized e-voting system that used seven roles to guarantee public and transparent voting process while ensuring voter's anonymity. Those roles are voters, registration server, authentication server, voting website, recording center, distributed data servers, and smart contract. Their proposed design involved Paillier homomorphic encryption, which we do not use as there is no arithmetic operation on cipher data in our solution.

III. The Material and Method

A. Electronic Voting and Election Law

Electronic voting refers to voting process that utilizes electronic devices and other modern technologies to cast and count the votes. Electronic voting can be held via internet, which the voters submit their votes to the voting organizer, from any location [9]. Organizers must employ any means necessary to ensure authentication and authorization for every cast ballot.

Specifically in Indonesia, Law (*Undang-undang*) number 7 year 2017 states in Article 2 that general election must comply with the following principles:

1) Direct: Each voter must cast his/her vote directly and not represented by other person or party.

2) Public: Every eligible member of society may participate in the voting, to cast his/her vote.

3) Free: A voter chooses candidate by his own will, not under threat or forced.

4) Confidential: Only the respective voter knows a voter's choice.

5) Honest: Every election and voting must comply with the regulation to guarantee the right of the voters, and that each vote cast has the same value.

6) Fair: All voters have equal right to vote, without any special privilege or discrimination.

Those principles formally applies to national election (such as electing president or regional representatives), although there is no reason not to use it as basis for any other type of voting in a democratic country such as Republic of Indonesia.

Fig. 1. Flowchart from [6].

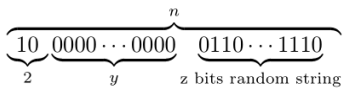

Fig. 2. Ballot Structure from [7] with Two First Characters used to Identify each Candidate.

B. Blockchain

Blockchain is a shared ledger of transactions. The transactions are ordered and grouped into blocks. Currently, the real-world model is based on private databases that each organization maintains, whereas the distributed ledger can serve as a single source of truth for all member organizations that are using the blockchain. Blockchain is also a data structure, a linked list that uses hash pointers instead of normal pointers. Hash pointers are used to point to the previous block [10].

Bitcoin cryptocurrency with chain of blocks as its basis was proposed by [5]. Blockchain employs consensus algorithm to achieve decentralization of control. Consensus provides a way for all peers to agree and accept a single version of truth on the blockchain network. Bitcoin itself uses proof-of-work consensus to prove that enough computational resources have been spent before proposing a truth to be accepted by peers, therefore solving the double spending problem and Byzantine General's problem.

C. Secure Hash Algorithm 3 (SHA-3)

SHA-3 is a latest member of secure hash algorithm standards. A cryptographic hash function is a one-way function that uses mathematical algorithm to map data of any size (message) to a fixed size bit string (hash). SHA-3 is meant to be an alternative to SHA-2, after successful attacks were proven on MD5 and SHA-1. SHA-3 uses Keccak algorithm. It is based on un-keyed permutations as opposed to other usual hash functions' constructions that used keyed permutations. A new approach called sponge and squeeze construction is used in Keccak, which is a random permutation model. The draft of SHA-3 (FIPS 202) was approved on 2015 by US National Institute of Standards and Technology [11]. SHA-3 is considered safe against quantum attack [12]. The performance is in par with SHA-2 [13].

The variant used in this study is SHA-3 with 256-bit of output (SHA3-256).

D. Universally Unique Identifier (UUID) Version 4

A UUID is 128 bits value that is used to identify a piece of data or information in computer systems. Every UUID is unique. The uniqueness of each value is guaranteed when it is generated using standard methods, and it does not depend on the parties that generate it. The protocol to generate UUID is specified in RFC 4122 [14].

UUID version 4 (UUID4) is generated randomly, not time-based or name-based like previous versions of UUID. Its probability of collision is so small that it can be safely ignored.

It leaves 122 of its 128 bits available for random data. The probability to find a duplicate within 103 trillion UUID4s is one in a billion.

E. Elliptic Curve Digital Signature Algorithm (ECDSA)

Digital signature is mathematical scheme for authenticating digital data and documents. If a signed data has valid digital signature, then the recipient could safely believe that it was created by a known sender (authenticity), which the sender cannot deny (non-repudiation), and that the data is intact and not altered (integrity). Digital signature employs asymmetric cryptography, which means two distinct keys are needed.

The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analogue of the Digital Signature Algorithm. The ECDSA is included in several standards, such as IEEE 1363-2000, ISO/IEC 15946-2, and FIPS PUB 186-4 (NIST). It is included in the cipher suites of the Transport Layer Security (TLS) protocol (RFC 4492) [15]. The elliptic curve is simply the set of points described by the equation (1) called Weierstrass normal form [16].

$$y^2 = x^3 + ax + b, \quad (1)$$

where $4a^3 + 27b^2 \neq 0$ to exclude singular curves.

ECDSA offers smaller key size than that of RSA-based ones for the same security level, allowing faster verification. Table I shows the comparison of RSA and ECC key sizes, while Table II shows the performance differences, which we measured by generating 1,000 signatures per algorithm to sign and verify 128 bytes message. The variant used in this study is ECDSA 256.

F. Methods

The right to vote is guaranteed by law in Indonesia. Voting process must follow voting principles: direct, public, free, confidential, honest, and fair. This study is aimed to provide a way for a voter to know whether his/her vote is recorded as-is, not just the summary of counts like in [6]. The proposed design will not implement Paillier cryptosystem, unlike [8]. The system will also provide a way to examine the counting process. Only valid voters may cast a vote.

The size of a transaction must be kept minimum, and the structure of the ballot must be simple and easy to understand. A block should contain as many transactions as possible, just like [5].

Blockchain is a distributed ledger. To ensure the accuracy and integrity of every record in a block, it must be sealed (mined) and chained with previous block. The sealing hash output is used as proof of work. Mining process requires effort, so the proposed system is intended for medium to large-scale usage.

In pre-voting phase, each potential voter generates UUID4 as pseudonym, a pair of public and private keys, and prepares legal documents. He/she then proceeds to validate his/her identity to the organizer, submit the public key, and keep his/her pseudonym and private key secret. It is up to the voting organizer to determine the best way to accomplish this. Fig. 3 shows the diagram of this phase.

TABLE. I. COMPARISON OF RSA AND ECC KEY SIZE [17]

RSA (bits)	ECC (bits)	Security Level (bits)
1024	160	80
2048	224	112
3072	256	128
7680	384	192
15,360	521	256

TABLE. II. PERFORMANCE DIFFERENCE BETWEEN ECDSA AND RSA-PSS

Component	ECDSA 256	RSA-PSS 3072
Avg. generation time (s)	0.085	1.161
Avg. signing time (s)	0.084	0.036
Avg. verification time (s)	0.169	0.002
Total time needed (s)	339.559	1,324.626

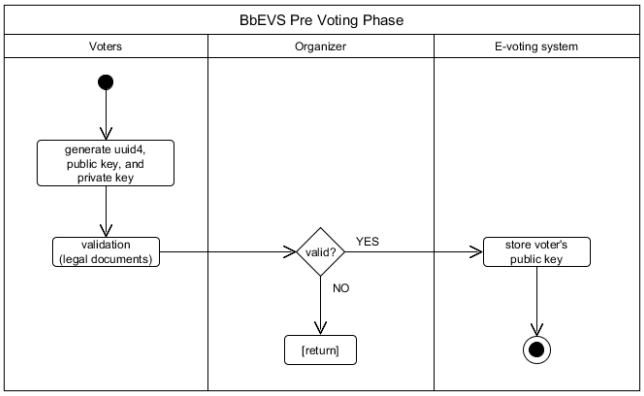

Fig. 3. Diagram of pre-voting phase

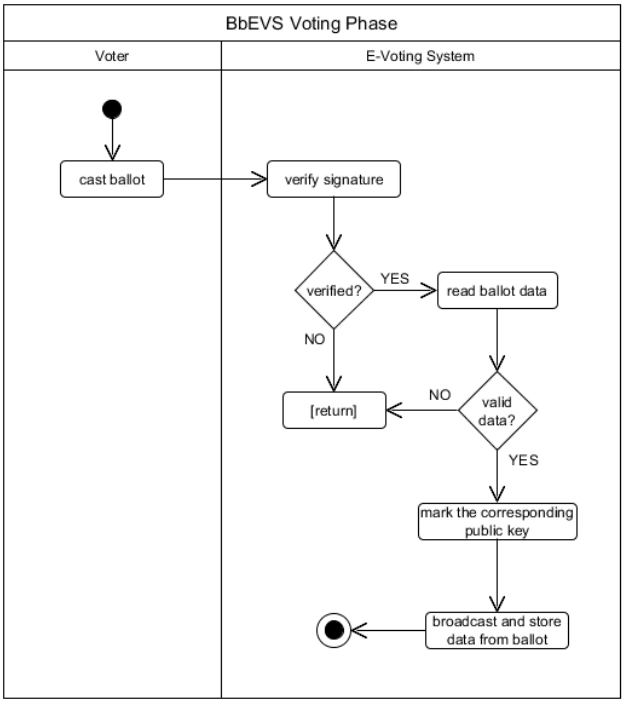

Fig. 4. Diagram of vote casting phase

Fig. 4 shows the diagram of vote casting phase. In this phase, each voter casts his/her own ballot after signing it. Once accepted by server, the ballot will be verified for authenticity and integrity before being relayed to all nodes. Data from a verified ballot are considered valid if the pseudonym is unique, candidate identifier is valid, and (optionally) the timestamp is considered reasonable.

Now we discuss the recording and counting phase. Ideally, a block is created when certain numbers of transactions (ballots) have been relayed to all nodes, and then that block is broadcasted. Finally, the voting result can be counted. The counted votes $V_{counted}$ should be less than or equal to the total votes cast, shown in (2).

$$V_{\text{counted}} = V_{\text{total}} - V_{\text{unmarked}} \quad (2)$$

The vote is 'unmarked' if the corresponding public key is never used for verification, or the data in the ballot are invalid.

In the proposed system, the ballot has the following structure:

$b_{\text{v\_id}} + b_{\text{c\_id}} + \text{t},$

where b_{v_id} is the UUID as voter's ID (32 bytes), b_{c_id} is the candidate ID (length may vary), and t as timestamp (can be either integer or float value). Timestamp value may be either the ballot creation time or the time the ballot is received by electronic voting system. Thus, the minimum length of a ballot is 43 bytes. One or more fields may be added or modified depending on voting requirements. The following is an example of valid ballot:

ae19033a1d9a4f6cbaed53c6d2de1f730011540300734.584385.

As comparison, the size of a Bitcoin transaction is approximately 267 bytes. The structure of the block used in this study does not contain block version and difficulty target.

To study and analyze how the proposed system works, we created a simulation program. The software is a web-based application with the following details:

- Programming language: Python version 3.6.0,
- Framework: Python-based Django framework version 2.1.2,
- Database engine: SQLite 3,
- GUI: web-based (browser) (debug message and other details are printed on the system's shell),
- Server: Django integrated development server.

In this simulation, transactions are broadcasted to two nodes. One of the nodes acts as an always-honest node so all blocks and transactions can be compared later. The number of transactions, transactions per block, and puzzle difficulty can be adjusted to compensate the performance of the computer that runs the simulation.

The simulation comprises two sections: "block" and "chain" (Fig. 5).

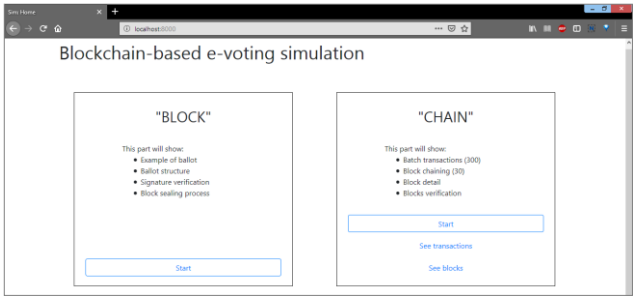

Fig. 5. Front Page of the Simulation Program.

1) "Block": This section shows the example of ballot and demonstrates signature verification and mining processes.

2) "Chain": This section demonstrates the batch generation of transactions, sealing (mining) process, detail of each block, verification, and data synchronization process.

The Python code to generate valid block hash is shown below:

```
# Try to seal the block and generate valid hash
nonce = 0
timestamp = datetime.datetime.now().timestamp()
while True:
    block_hash = SHA3_256.new(prev_hash + merkle_h + nonce + timestamp).hexdigest()
    # Check whether this hash satisfies puzzle requirement
    if block_hash[:pcount] == puzzle:
        break
    nonce += 1
```

The `prev_hash`, `merkle_h`, `nonce`, and `timestamp` are all encoded in binary so they are concatenated using '+' operator. `pcount` and `puzzle` are defined in separate configuration file.

The Python code to generate a single transaction with valid, random values is shown below:

```
# generate random, valid values
v_id = str(uuid4())
v_cand = _get_vote()
v_timestamp = _get_timestamp()
# directly fill the values and the block id for simulation purpose
new_vote = Vote(id=v_id, vote=v_cand, timestamp=v_timestamp)
new_backup_vote = VoteBackup(id=v_id, vote=v_cand, timestamp=v_timestamp)
# "Broadcast" to two nodes
new_vote.save()
new_backup_vote.save()
print("#{} new vote: {}".format(i, new_vote)) # debug
```

A database in each node is used to record and synchronize all transactions (ballots) and blocks. Table III and Table IV show the model structure of ballot and block, respectively. Fig. 6 shows their relationship.

TABLE. III. STRUCTURE OF VOTE (BALLOT) MODEL

Field	Type	Description
id	char	UUID as primary key
vote	int	Candidate id
timestamp	float	
block_id	int	Foreign key

TABLE. IV. STRUCTURE OF BLOCK MODEL

Field	Type	Description
id	int	Auto-increment, used for simulation
prev_h	char	Previous block hash
merkle_h	char	Merkle root hash
h	char	Block hash
nonce	int	An arbitrary number
timestamp	float	Block creation timestamp

Fig. 6. Entity Relationship Diagram.

Some tests must be run to ensure the proposed system meets the following requirements. First, each user can examine their cast ballots after voting is over. Second, users can examine the detail of each block (hashes, nonce, number of transactions it contains, total number of blocks, etc.). Third, in case a node gets corrupted, it must be able to sync with majority of nodes aka "the agreed truth". A reasonably great number of dummy, valid ballots must be generated to run this test, i.e., 10,000. In our study, they are generated programmatically. The built-in user interface, i.e., web UI, is used to confirm the result.

The simulation was run and benchmarked on the computer with the following specifications:

- CPU: 4 Cores, up to 3.6 GHz,
- RAM: 8 (2x4) GB, 1,600 MHz,
- Hard drive: 466 GB capacity, Read 74.45 MB/s, Write 64.18 MB/s,
- Operating system: Windows 8.1 Pro, 64-bit.

IV. RESULTS AND DISCUSSION

A. Results

In the "Block" section of the simulation program, we cast a signed ballot using the web user interface (Fig. 7) as a single transaction and then sealed it into a block. After several trials, the block was mined successfully after a valid hash had been generated. In our test, the nonce was 71,252 and the puzzle difficulty required hash with four leading zeros. It took approximately 7.166 seconds (Fig. 8).

In the "Chain" section, we generated 10,000 votes (Fig. 9) in approximately 2,585 seconds (43 minutes 5 seconds), and broadcasted them to transaction pool. Each block comprised 20 votes as transactions. Thus, the maximum size of each block was 1,000 bytes. This size was decided to make sure that each block was small enough so all transactions in the block could be verified quickly. Finally, 500 blocks were created successfully; each one correctly contained 20 transactions. In this round, candidate #2 won by 3,416 votes (Fig. 10).

We tampered some records (transactions) on the database using a database management tool. All blocks and the transactions were then checked for integrity, and the system successfully detected blocks on the main node with tampered data, shown in Fig. 11. The troubled node had to synchronize its data with the majority of nodes on the network.

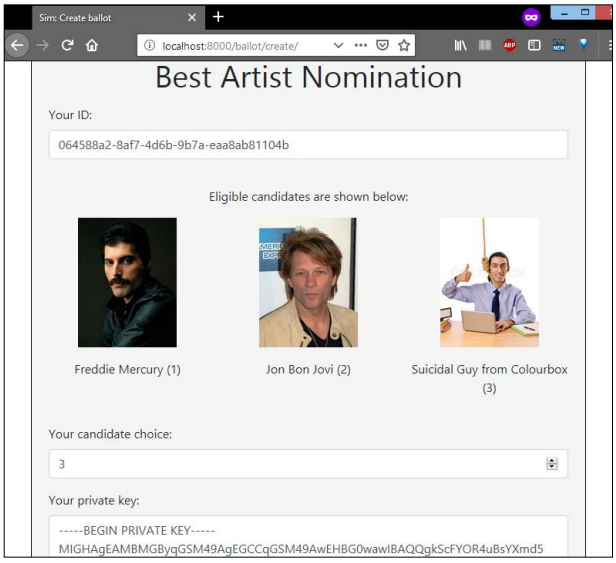

Fig. 7. A Ballot Example.

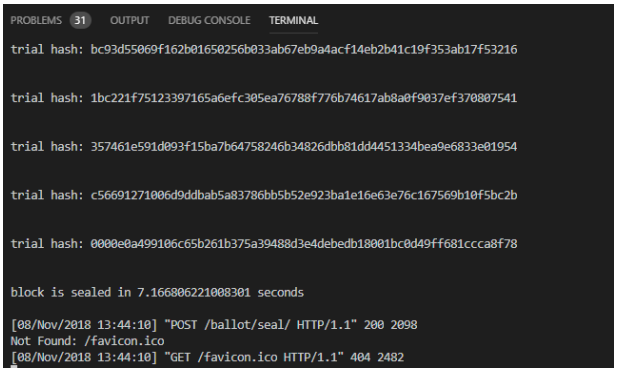

Fig. 8. Terminal Output Showing Mining Process.

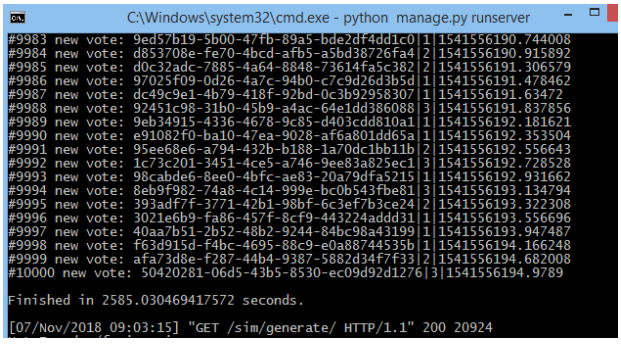

Fig. 9. Successfully Generated 10,000 Votes.

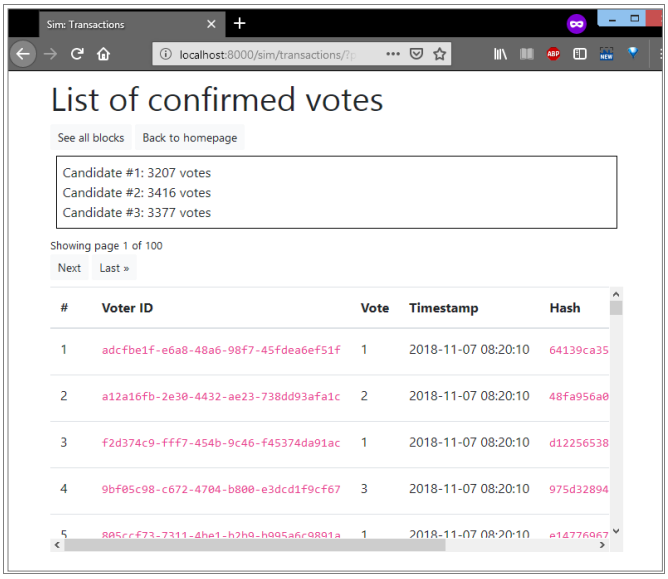

Fig. 10. Voting Result in the Simulation.

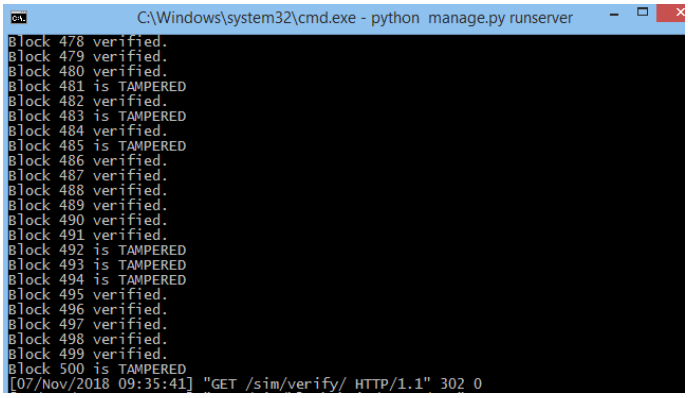

Fig. 11. The System Detected Blocks with Tampered Transactions.

We then synchronized the blocks in the main node by comparing them with the second (always-honest) node in the simulation. All the data were successfully restored, shown in Fig. 12 (for block #1). Fig. 13 shows block #1, including its header, status, and transactions (votes).

In our simulation, the timestamps were stored as float and each candidate was identified by only one character. The average ballot size was 47.42 bytes. Fig. 14 shows the approximated size of database for up to 9,999 transactions. That many transactions should require 474.152 KB of database.

B. Discussion

Each voter has to submit a public key after verification involving legal documents, so this system does not need fingerprint schema for digital signature. In the proposed system, after a signed ballot has been proven valid, the corresponding public key is then marked or deleted. This way, a voter could only vote once.

For the pseudonym, hash digest may also be used to replace UUID. The public key portion of ECDSA could possibly be used as the hash input. The reason of using UUID instead of hash digest in this study is that some hash algorithms that produce reasonably small digest output are prone to collision (such as MD5 and SHA1), and trimming long output could mean wasting computational power. Each UUID is unique and exposed only when a vote has been cast and verified. If somehow there are UUIDs with the same value in the pool, the first to be proven valid is the counted one (though this will never happen, if the protocol has been implemented correctly).

To ensure that a voter's location cannot be tracked, he/she can send the ballot via proxy, such as Tor. Inputting private key into web form to sign the vote is not recommended method; the voter should generate the valid ballot by his/her own and then send it via secure API. Due to the design of the proposed ballot, Paillier homomorphic encryption is not needed, contrary to the research by [8].

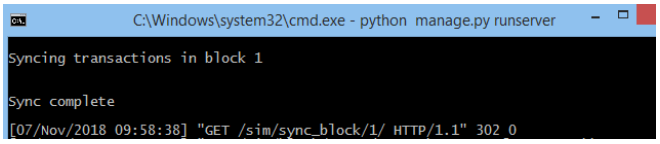

Fig. 12. All Transactions in a Block had been Synchronized.

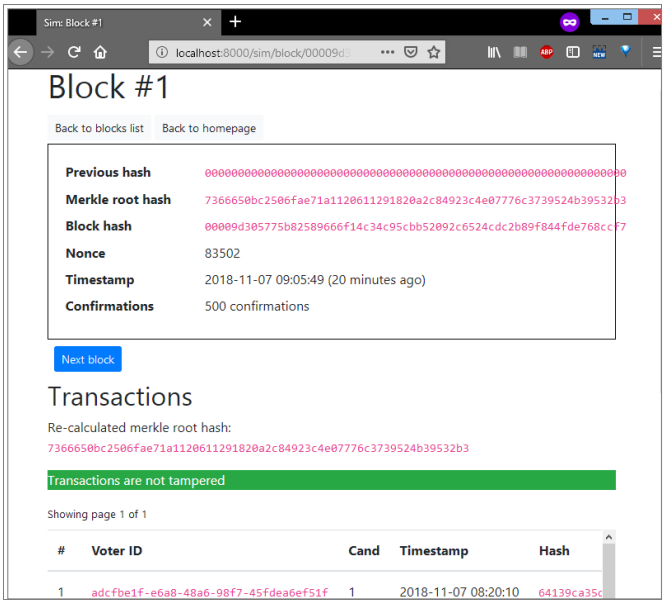

Fig. 13. Details of a Block.

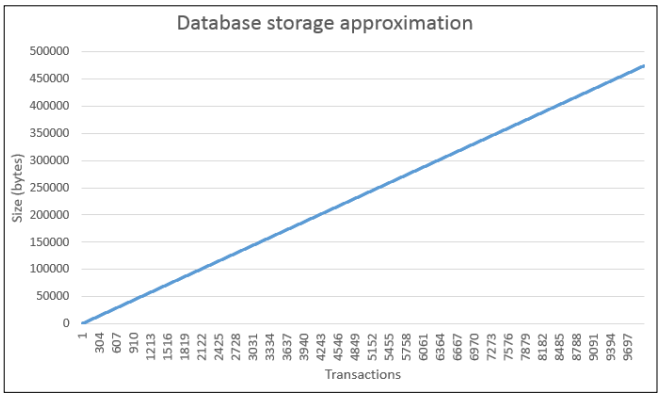

Fig. 14. Line Chart of Database Capacity Measurement.

V. Conclusions

Based on the simulation and testing results, the proposed system worked well and complied with Indonesian voting principles. Received ballots could be checked for authenticity and integrity using ECDSA, and then all the data in each ballot were checked for validity. Only valid ballots could be recorded as transactions, which were then mined into blocks. If a node had one or more tampered transactions, they could be detected and restored by comparing the data with other node. The usage of SHA-3 and ECDSA (instead of RSA signature) were meant to speed up the process of generating key pair and keep overall data size to minimum. In the future, the system could be tested inter-device on the local network or internet using secure API.

Acknowledgment

We would like to thank the Gunadarma Foundation for financial support of this research.

References

[1] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, Burlington: Morgan Kaufmann, 2017.

[2] S. Valsamidis, S. Kontogiannis, T. Theodosiou and I. Petasakis, "A Web e-voting system with a data analysis component," Journal of Systems and Information Technology, vol. 20, no. 1, pp. 33-53, 2018.

[3] The Economist, "The great chain of being sure about things," 31 October 2015. [Online]. Available: https://www.economist.com/briefing/2015/10/31/the-great-chain-of-being-sure-about-things. [Accessed 8 October 2018].

[4] A. Narayanan, J. Bonneau, E. Felten, A. Miller and S. Goldfeder, Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction, Princeton: Princeton University Press, 2016.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[6] R. Hanifatunnisa, "Design and Implementation of Blockchain Based E-Voting Recording System," Master's Program Thesis, Institut Teknologi Bandung, Bandung, 2017.

[7] Y. Liu and Q. Wang, "An e-voting protocol based on blockchain," IACR Cryptol. ePrint Arch., vol. 1043, p. 2017, 2017.

[8] J. Hsiao, R. Tso, C. Chen and M. Wu, "Decentralized E-Voting Systems Based on the Blockchain Technology," in Advances in Computer Science and Ubiquitous Computing, Singapore, Springer, 2017, pp. 305-309.

[9] D. Zissis and D. Lekkas, "Securing e-Government and e-Voting with an open cloud computing architecture," Government Information Quarterly, vol. 28, no. 2, pp. 239-251, 2011.

[10] I. Bashir, Mastering Blockchain, Birmingham: Packt Publishing Ltd., 2017.

[11] NIST, "Announcing Approval of Federal Information Processing Standard (FIPS) 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, and Revision of the Applicability Clause of FIPS 180-4, Secure Hash Standard," 5 August 2015. [Online]. Available: https://www.gpo.gov/fdsys/pkg/FR-2015-08-05/pdf/2015-19181.pdf. [Accessed 26 October 2018].

[12] M. Amy, O. Di Matteo, V. Gheorghiu, M. Mosca, A. Parent and J. Schanck, "Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3," in International Conference on Selected Areas in Cryptography, Cham, 2016.

[13] G. Bertoni, J. Daemen, M. Peeters, G. Assche and R. Keer, "Is SHA-3 slow?," 12 June 2017. [Online]. Available: https://keccak.team/2017/is_sha3_slow.html. [Accessed 15 November 2018].

[14] P. Leach, M. Mealling and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace," Internet Engineering Task Force, July 2005. [Online]. Available: https://tools.ietf.org/html/rfc4122.html#section-4.1. [Accessed 15 November 2018].

[15] H. Delfs and H. Knebl, Introduction to Cryptography: Principles and Applications, Springer, 2015.

[16] A. Corbellini, "Elliptic Curve Cryptography: a gentle introduction," 17 May 2015. [Online]. Available: http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/. [Accessed 15 November 2018].

[17] M. Bafandehkar, S. Yasin, R. Mahmod and Z. Hanapi, "Comparison of ECC and RSA algorithm in resource constrained devices," in International Conference on IT Convergence and Security, 2013.