Desain dan Implementasi Sistem Pencatatan Pemungutan Suara dengan Teknologi *Blockchain* pada Jaringan *Peer-to-Peer*

(*Design and Implementation of Voting Recording System on Peer-to-Peer Network Using Blockhain*)

Rifa Hanifatunnisa[1], Muhammad Ismail[2]

***Abstract*—Voting is one of the important parts in democratic system, which is mostly done conventionally. The implementation of voting activities managed by one party or organization makes the results of voting results vulnerable to manipulation. Blockchain, as a technology that can secure data, can be used as one solution to overcome fraud in the voting process. Voting result data is recorded in the blockchain, then the data is sent to many parties. This distributed data also depends on one party about the results of the voting data. This study discusses the application of a voice recording system using blockchain technology and the process of distributing the data in a peer-to-peer network. In this system, TPS is a node on the network that will record, sign, and then transfer the data to other nodes that are connected to each other in a peer-to-peer network. That way, the data from the voting results of each node will be stored in all nodes. In addition, the results of the voting can be seen in real-time and the process of counting the results of the voting can be done faster**.

***Intisari*—Proses *voting* (pemungutan suara) merupakan salah satu bagian penting dalam sistem demokrasi, yang sampai saat ini masih banyak dilakukan secara konvensional menggunakan kertas. Pelaksanaan kegiatan *voting* yang dikelola oleh satu pihak atau organisasi membuat data hasil *voting* rentan manipulasi. *Blockchain* sebagai sebuah teknologi yang mampu menjaga keamanan data dapat dijadikan sebagai salah satu solusi untuk mengatasi kecurangan dalam proses *voting*. Data hasil *voting* dicatat di dalam *blockchain*, kemudian data tersebut didistribusikan ke banyak pihak. Data yang terdistribusi ini juga menghilangkan ketergantungan pada satu pihak mengenai data hasil *voting*. Makalah ini membahas penerapan sistem pencatatan *voting* menggunakan teknologi *blockchain* dan proses pendistribusian data tersebut dalam jaringan *peer-to-peer*. Pada sistem ini, TPS merupakan *node* pada jaringan yang akan mencatat, menandatangani, dan menyebarkan data *voting* di tempatnya ke *node* lain yang saling terhubung dalam jaringan *peer-to-peer*. Dengan begitu, salinan data hasil *voting* tiap *node* akan disimpan di semua *node*. Selain itu, hasil *voting* dapat dilihat secara *real-time* dan membuat proses penghitungan hasil *voting* menjadi lebih cepat.**

***Kata Kunci—Voting, Blockchain, Network, Peer-to-peer*.**

[1,2] *Jurusan Teknik Elektro, Politeknik Negeri Bandung, Jalan Gegerkalong Hilir, Ds. Ciwaruga, Bandung, INDONESIA (tlp: 022-2013789; fax: 022-2013889; e-mail: rifahani@polban.ac.id, mail.muhismail@gmail.com)*

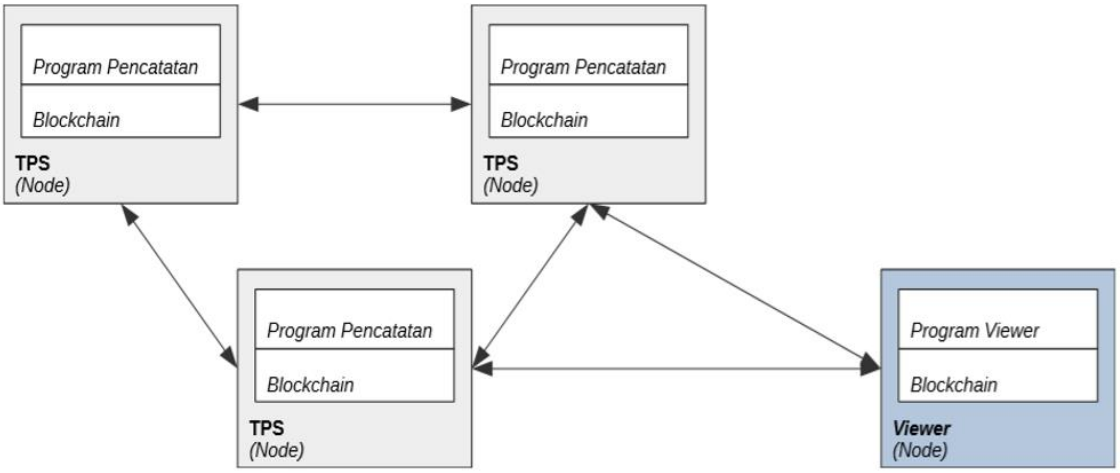
I. PENDAHULUAN

Pemungutan suara (v*oting*) merupakan bagian penting dalam proses demokrasi. Pemilih yang berpartisipasi menentukan keputusannya melalui *voting* nantinya akan diakumulasi dan dihitung jumlah suaranya untuk masing-masing pilihan.

Salah satu contoh pemanfaatan proses *voting* yang paling umum dalam suatu organisasi adalah memilih ketua atau pemimpin organisasi. Di Indonesia sendiri, kegiatan pemilihan umum (pemilu) dilaksanakan dengan melakukan *voting* untuk memilih Presiden dan Wakil Presiden, anggota Dewan Perwakilan Rakyat (DPR), anggota Dewan Perwakilan Daerah (DPD), dan anggota Dewan Perwakilan Rakyat Daerah (DPRD) [1].

Pemungutan suara pada pemilu di Indonesia masih dilakukan dengan menggunakan kertas. Proses pemilihan hingga proses penghitungan hasil pemilu dilakukan secara konvensional. Hal ini membuat proses penghitungan suara menjadi lambat, bahkan dapat memakan waktu hingga berminggu-minggu. Selain itu, proses penghitungan ini juga rentan terhadap kesalahan manusia seperti ketidaktelitian serta data hasil pemilu rentan terhadap manipulasi. Pengolahan data hasil pemilu secara terpusat (dikelola satu pihak) pun tidak menjamin integritas data. Pengecekan data untuk mencegah manipulasi pun akan sangat menyulitkan, terutama untuk kegiatan pemilu dalam skala masif dengan jumlah pemilih mencapai jutaan orang.

Blockchain sebagai teknologi yang digunakan pada Bitcoin untuk *database* transaksinya [2] dapat digunakan untuk mengurangi masalah dalam pemilu seperti yang telah dijelaskan. *Blockchain* ini merupakan sebuah *ledger* yang berisi semua transaksi yang telah dilakukan pada suatu jaringan dan disimpan secara aman [3], terdiri atas beberapa *block* yang saling terikat menggunakan *hash* dan didistribusikan pada sebuah jaringan terdesentralisasi, sehingga setiap *node* pada jaringan tersebut menyimpan salinan *blockchain*. Sifatnya yang terdesentralisasi membuatnya *blockchain* tidak bergantung dan tidak perlu diatur oleh satu otoritas. Beberapa kelebihan dari penggunaan *blockchain* adalah keamanan dan keandalannya terhadap serangan *Distributed Denial of Service* (DDoS); memungkinkan adanya transparansi data, sehingga masyarakat dapat berpartisipasi dalam melakukan audit data hasil pemilu; juga *immutability* atau tidak dapat dilakukan perubahan terhadap data [4].



Gbr. 1 Diagram blok sistem.

Oleh karena itu, dibuat sebuah sistem yang memungkinkan dilakukannya pengamanan data hasil pemungutan suara menggunakan teknologi *blockchain*. Pada sistem ini, data hasil pemungutan suara pada pemilu disimpan dalam sebuah *blockchain*. Data ini kemudian didistribusikan untuk memperlihatkan transparansi hasil pemilu. Sistem ini juga memungkinkan penghitungan hasil pemilu dilakukan dengan cepat karena perhitungan dapat dilakukan secara *real-time*. Sistem ini dapat mempermudah dan mempercepat proses rekapitulasi hasil pemungutan suara dalam kegiatan pemilu.

Makalah ini merupakan proses pengukuran implementasi dari penelitian sebelumnya, yaitu sistem fokus pada metode pengamanan pencatatan hasil *e-voting* yang telah dilaksanakan [5]. Berbeda dengan penggunaan *blockchain* pada Bitcoin, sistem ini menggunakan *permission blockchain*, yaitu hanya *node* tertentu saja yang dapat berpartisipasi (menambah *block*/data) dalam jaringan. *Node* pada sistem ini merupakan Tempat Pemungutan Suara (TPS) yang telah terdaftar pada sistem sebelum pelaksanaan pemilu.

II. *VOTING* DAN TEKNOLOGI *BLOCKCHAIN*

E-voting saat ini banyak digunakan oleh beberapa negara di dunia, contohnya di Estonia. Negara ini telah menggunakan sistem *e-voting* sejak tahun 2005 dan pada tahun 2007 telah melaksanakan *voting* secara *online*. Estonia juga merupakan negara pertama di dunia yang melaksanakan *online voting* [6]. Sejak itu, sistem *voting* secara *online* yang mengikat secara hukum diterapkan di berbagai organisasi dan negara lainnya, seperti *Austrian Federation of Students*, Swiss, Belanda, dan Norwegia [7], tetapi masih memiliki masalah keamanan yang cukup besar dan pemilihan sering dibatalkan [8]. Meskipun mendapatkan banyak perhatian, sistem *voting* secara *online* masih belum banyak dilakukan di berbagai negara di dunia, termasuk di Indonesia, yang masih menggunakan sistem konvensional. Sistem *voting* tradisional memiliki beberapa masalah yang dihadapi ketika dikelola oleh satu organisasi yang memiliki kontrol penuh terhadap sistem dan *database*, karena organisasi tersebut dapat mengubah *database* dan ketika *database* berubah, maka jejak pun dapat dengan mudah dihilangkan [9].

Solusi dari masalah yang ada adalah dengan membuat *database* menjadi bersifat publik, yaitu salinan *database* dimiliki oleh banyak pengguna, yang berguna sebagai pembanding apabila terdapat kejanggalan. Solusi untuk sistem *e-voting* tersebut cocok dilakukan menggunakan teknologi *blockchain*. Teknologi *blockchain* memungkinkan dalam mendukung aplikasi *e-voting*. Suara tiap pemilih dijadikan sebagai transaksi yang dapat diciptakan menjadi *blockchain* yang dapat berfungsi untuk melacak perhitungan suara. Dengan cara seperti ini, setiap orang dapat menyetujui perhitungan akhir karena jejak audit *blockchain* yang terbuka, perhitungan suara dapat diverifikasi bahwa tidak data yang diubah atau dihapus, juga tidak ada data tidak sah yang dimasukkan dalam *blockchain.*

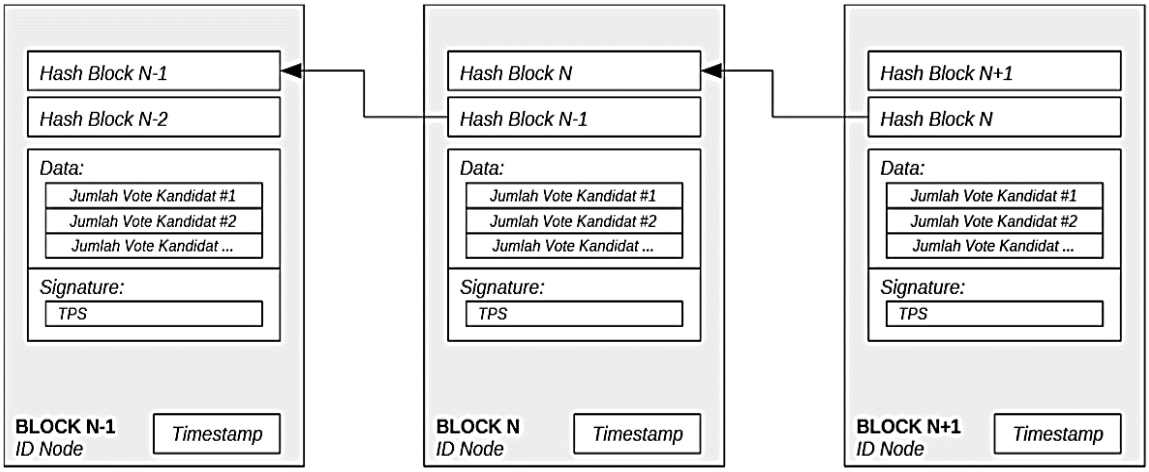
Teknologi *blockchain* memiliki beberapa keunggulan, yang membuatnya menjadi alternatif yang kuat dan aman untuk *database*, sebagai berikut [8].

- *High Availability*. Data didistribusikan sepenuhnya ke seluruh *node* dan disimpan dalam *database* secara lengkap.
- *Verifiability and integrity*. Setiap *block* diverifikasi dan ditambahkan ke dalam *blockchain*. Oleh karena itu, data di dalamnya akan sulit diubah, karena seluruh *block* menjadi harus ikut diubah nilainya.
- Mudah dalam menentukan satu *common starting point*, tempat untuk menyimpan data – yang selalu ditambahkan ke *block* terakhir dalam rantai terpanjang.

Keunggulan ini membuat *blockchain* menarik untuk digunakan dalam *database* pada sistem *e-voting*.

III. METODOLOGI

Diagram blok pada Gbr. 1 menunjukkan dua jenis *node*, yaitu *node* TPS dan *node viewer*. Perbedaan antara kedua *node* ini terletak pada akes yang dimiliki tiap *node* terhadap *blockchain*. *Node* TPS memiliki akses untuk melakukan perubahan pada *blockchain*, yaitu menambahkan *block* baru. *Node viewer* tidak memiliki akses untuk melakukan perubahan pada *blockchain*, tetapi memiliki akses untuk mendapatkan dan melihat data *blockchain*. Dengan kata lain,



Gbr. 2 *Blockchain* sistem pencatatan pemungutan suara.

node ini merupakan pengawas atau pemantau yang merupakan peran masyarakat umum. Masing-masing *node* dapat memiliki salinan *blockchain*, terutama untuk *node* TPS yang diharuskan untuk memiliki salinan *blockchain*.

Manajemen *blockchain* ini diatur oleh program, yang pada *node* TPS adalah program pencatatan dan pada *node viewer* adalah program *viewer*. Program pencatatan, sesuai dengan namanya, merupakan program yang bertugas untuk mencatat hasil pemungutan suara pada TPS tersebut dan mem-*broadcast*-kannya ke *node* lainnya. Program ini juga bertugas melakukan proses penandatanganan digital, melakukan verifikasi, dan mendengarkan *broadcast* dari *node* TPS lainnya. Sedangkan program *viewer* bertugas untuk mendengarkan *broadcast* dari *node* lainnya dan kemudian melakukan verifikasi *blockchain*. Program *viewer* ini merupakan aplikasi berbasis web yang juga dapat menampilkan data *blockchain* yang diterimanya.

Isi dari *block* yang digunakan pada sistem ini terdiri atas ID *node*, *timestamp*, *hash block*, *hash block* sebelumnya, data hasil *voting* tiap kandidat, serta tanda tangan (*signature*) saksi-saksi dan TPS. Struktur dan isi dari *block* dan gambaran dari *blockchain* pada sistem ini diperlihatkan dalam Gbr. 2.

A. Perancangan Algoritme

Sebelum pemilu dimulai, masing-masing TPS membuat sebuah *private key* dan *public key*. *Public key* ini kemudian disebar ke setiap *node*/TPS sehingga setiap *node* memiliki daftar lengkap *public key* masing-masing TPS. Karena *public key* dan *private key* dapat dibuat dengan mudah dan gratis, perlu adanya kesepakatan mengenai kevalidan semua *public key* yang ada.

Setelah pemilu selesai dan proses penghitungan suara dilaksanakan, setiap TPS mencatat jumlah *vote* yang diterima tiap pilihan/kandidat. Data yang berisi hasil penghitungan *vote* tiap kandidat tersebut kemudian ditandatangani (*sign*) oleh TPS tersebut dengan menggunakan *private key*-nya. Setelah proses penandatanganan selesai, TPS/*node* menyimpan semua data ke dalam *block* dan melakukan proses *hashing* terhadap *block* tersebut. Akhirnya, *block* di-*broadcast*-kan oleh TPS yang nantinya akan diterima oleh *node* lainnya untuk dilakukan verifikasi dan dimasukkan ke dalam *blockchain* masing-masing *node*.

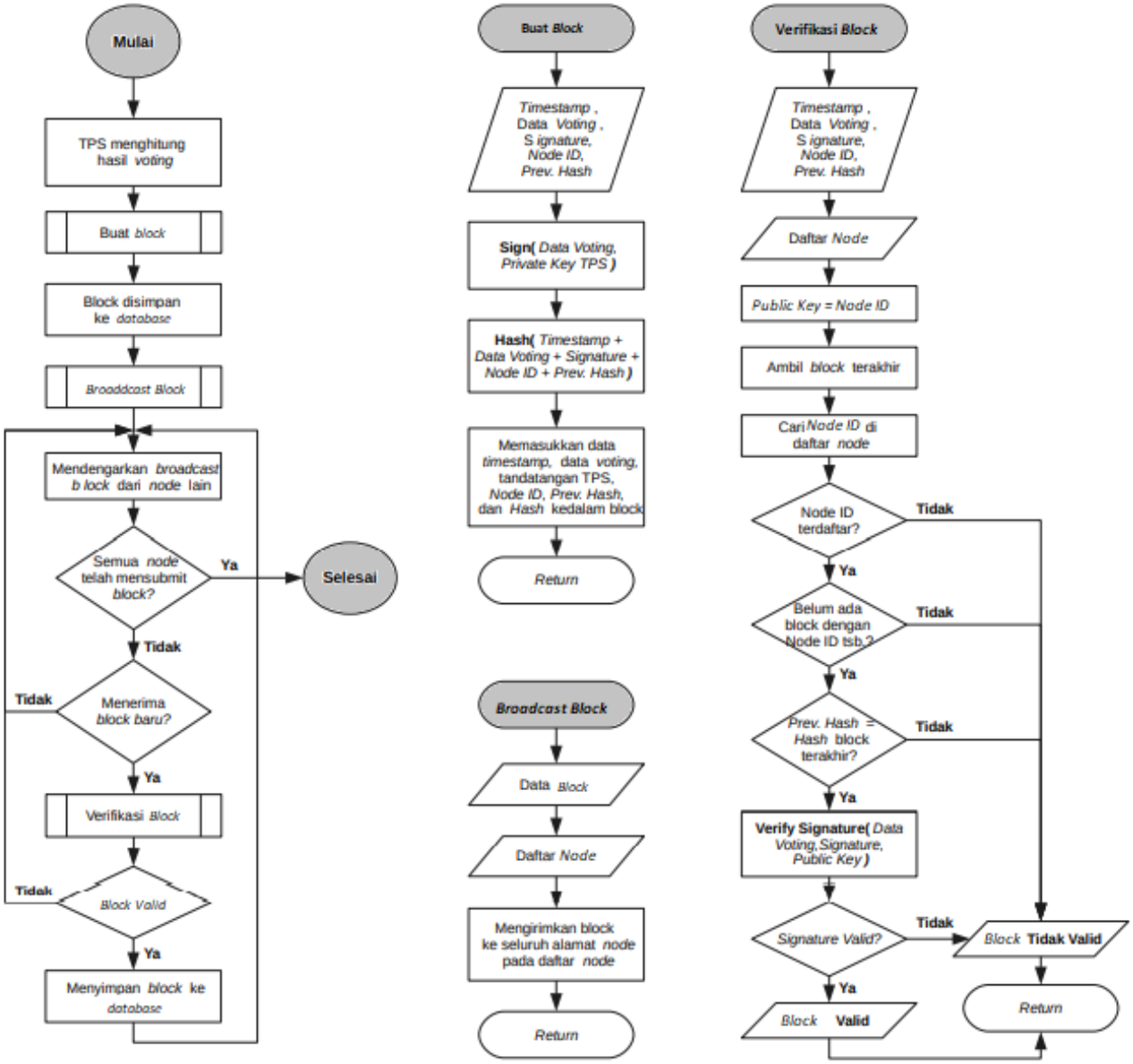
Setelah semua *block* terkumpul, proses penghitungan jumlah *vote* atau rekapitulasi akhir dapat dilakukan. Sebelumnya, dilakukan terlebih dahulu proses verifikasi untuk mengecek sah atau tidaknya *blockchain* tersebut. Apabila *blockchain* sah atau valid, hasil perhitungan dapat dilakukan dan jumlah *vote* masing-masing pilihan/kandidat dapat dilihat.

Diagram alir pada Gbr. 3 terdiri atas satu diagram alir utama dan tiga diagram alir proses turunan. Digram alir ini menunjukkan kerja sistem saat pembuatan *block*, *broadcast block*, dan mendengarkan *block* dari *node* lain. Sistem ini akan dimulai setelah proses penghitungan hasil *voting*, yaitu ketika TPS akan melakukan pembuatan *block*.

1) Pembuatan Block: Pembuatan *block* dimulai dengan memasukkan data hasil *voting* dari TPS tersebut untuk kemudian dilakukan proses penandatanganan (*signing*). Proses penandatanganan ini membutuhkan *private key* yang dimiliki oleh TPS tersebut. Setelah proses penandatangan selesai, *block* akan mengambil data-data berikut.

- *Timestamp*, yaitu waktu pembuatan *block* dalam format *UNIX time* yang merupakan jumlah detik yang telah berlalu sejak 00:00:00 Kamis, 1 Januari 1970 [10].
- Data *voting*, yaitu jumlah *vote* setiap kandidat yang didapat dari hasil perhitungan TPS.
- *Signature*, yakni tanda tangan digital yang dibuat oleh TPS menggunakan *private key*. Data yang ditanda tangan adalah data *voting* TPS.
- *Node ID*, yaitu identitas *node* yang merupakan *public key* dari *node*/TPS.
- *Prev. Hash* merupakan *hash* dari *block* sebelumnya.

Data-data tersebut kemudian dimasukkan ke dalam fungsi *hash* untuk mendapatkan nilai *hash* dari *block* yang akan dibuat. Fungsi *hash* yang digunakan adalah SHA256. Hasil *hash* beserta kelima data yang dimasukkan ke dalam fungsi *hash* tadi merupakan sebuah *block* data yang kemudian disimpan ke dalam *database*.

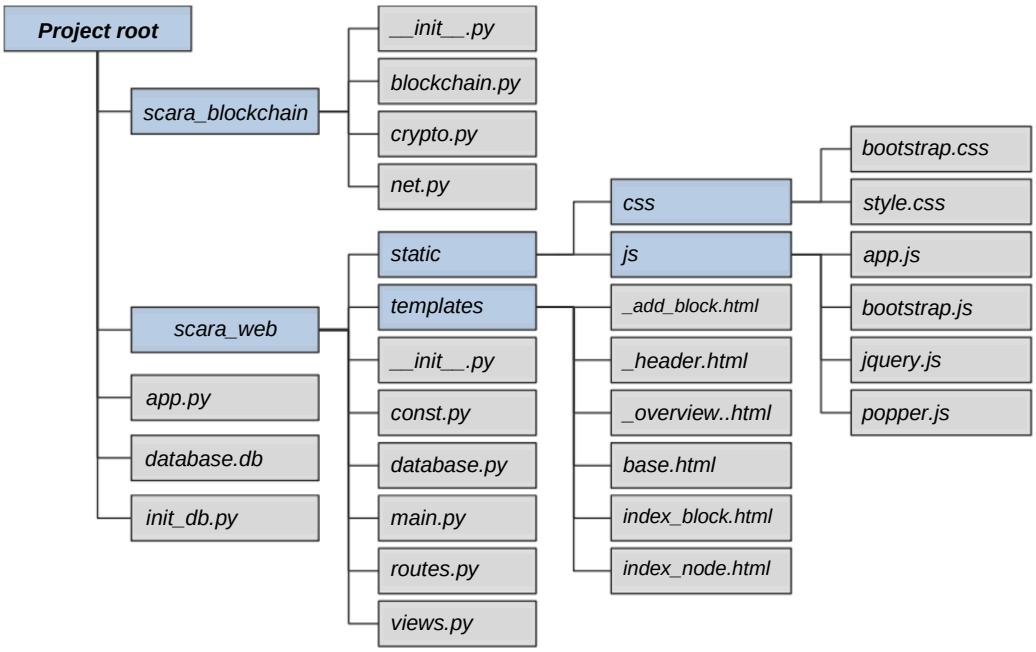


Gbr. 3 Diagram alir sistem.

2) Broadcast Block: Block yang telah dibuat kemudian akan di-*broadcast*-kan ke semua *node* yang terdaftar dalam sistem. Daftar *node* ini disimpan dalam *database* sistem. Masing-masing *node* memiliki alamat IP atau alamat *host* dan juga *port*-nya. Data yang di-*broadcast*-kan ini akan diterima oleh *node* tujuan untuk kemudian dilakukan verifikasi oleh *node* tujuan. Setelah proses *broadcast* selesai, *node*/TPS ini akan mendengarkan *broadcast* dari *node* lainnya.

3) Mendengarkan Broadcast Block: Node akan menerima *block* dari *node*/TPS lainnya. Proses ini dilakukan sampai semua *block* dari setiap *node* yang terdaftar pada sistem (terdaftar pada *database* milik *node*) telah diterima semuanya. Setiap kali *node* menerima *block* dari *node* lain, *block* akan melakukan verifikasi untuk menguji validitas *block* yang diterimanya. Apabila *block* yang diterima adalah *block* yang valid, *block* tersebut akan diterima dan akan disimpan pada *database*. Jika tidak, *block* tersebut akan ditolak dan *node* akan melanjutkan mendengarkan *broadcast* dari *node* lainnya.

4) Verifikasi Block: Setiap *block* yang diterima dari *node* lain akan dicek keabsahan atau validitasnya. Pertama adalah dengan menguji, data *node ID* pada *block* yang diterima merupakan *node ID* yang terdaftar pada *database* sistem atau tidak. Jika *node ID* yang diterima benar, *node ID* ini kemudian diperiksa, *node* ini telah membuat *block* sebelumnya atau belum, dengan memeriksa daftar *block* di *database*. Proses selanjutnya adalah memeriksa nilai *previous hash* dari *block* yang diterima dengan *hash* pada *block* terakhir yang ada pada *database*, untuk memastikan *block* yang diterima terikat dengan *block* sebelumnya. Proses pemeriksaan yang terakhir adalah validitas data *voting* pada *block* yang diterima tersebut. Pengecekan validitas tanda tangan dapat dilakukan dengan



Gbr. 4 Struktur *file* program.

memeriksa tanda tangan yang ada pada *block* yang diterima. Proses verifikasi tanda tangan ini juga membutuhkan *public key* dari *node* yang melakukan tanda tangan data *voting* tersebut. *Public key* ini sama dengan *node ID* pada *block*. Apabila data pada *block* yang diterima berhasil melewati semua proses pemeriksaan, *block* tersebut dapat dikatakan valid. Sebaliknya, jika tidak, *block* tersebut tidak valid dan tidak dapat diterima dan disimpan pada *database*.

B. Realisasi

Sistem dibuat menggunakan bahasa pemrograman Python. Program dijalankan pada beberapa komputer yang saling terkoneksi, yang komputer-komputer tersebut berada pada jaringan yang sama. Aplikasi ini merupakan aplikasi berbasis web. Bagian antarmuka (bagian *front-end*) dibuat dengan menggunakan HTML, CSS, dan Javascript, sedangkan bagian *back-end* dibuat menggunakan bahasa pemrograman Python.

Bagian *back-end* ini dibuat dengan memanfaatkan beberapa *library*, seperti *aiohttp*, yang merupakan *library* yang dapat digunakan untuk membuat sebuah server yang dapat berjalan secara asinkron; *library Pycryptodome* sebagai library yang menyediakan fungsi-fungsi kriptografi; dan *aiosqlite* sebagai *library* yang menyediakan fungsi untuk melakukan pengolahan data pada *database* SQLite 3 secara asinkron.

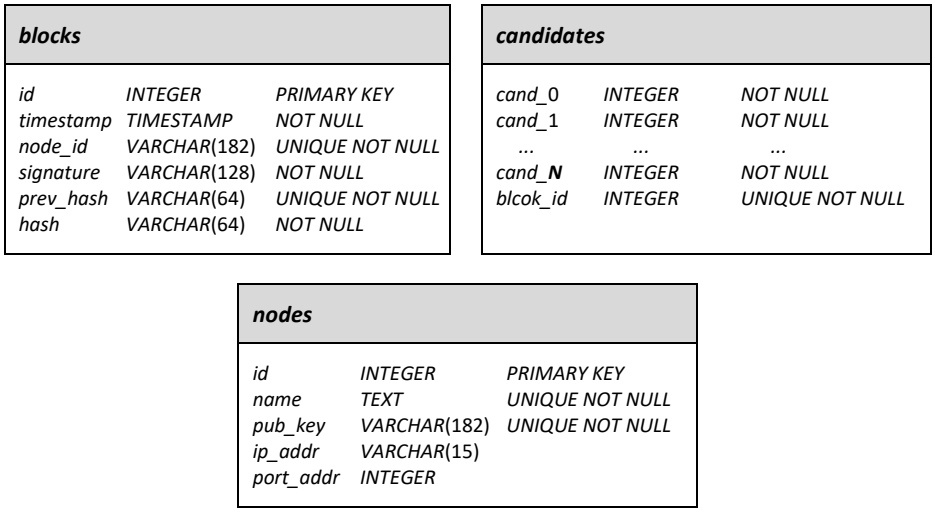
Data pada aplikasi ini disimpan pada database SQLite. Database ini disimpan pada sebuah *file* sehingga membuat program lebih portabel.

1) Struktur Program: Terdapat dua modul yang dibuat dalam realisasi program ini. Pertama adalah modul *blockchain* berisi fungsi-fungsi yang berkaitan dengan pemrosesan *blockchain* itu sendiri. Modul *blockchain* ini disimpan dalam *folder scara_blockchain*. Modul web sendiri merupakan modul yang berkaitan dengan aplikasi web, yang digunakan sebagai antarmuka program. Modul web ini disimpan pada *folder scara_web*. *File app.py* merupakan program *bootstrap*. *File* ini akan mengintegrasikan program dengan memanggil *file* atau modul lain. Untuk lebih jelasnya, struktur program atau struktur *file* pada program ini ditunjukkan pada Gbr. 4.

1) Realisasi Antarmuka Pengguna (User Interface): Antarmuka pengguna dibuat dengan menggunakan teknologi web. Artinya, pengguna dapat membuka *browser* untuk berinteraksi dengan program. Program web ini dibuat dengan menggunakan *library aiohttp* sebagai *backend*-nya, yang bertugas untuk "melayani" permintaan/*request* dari pengguna. Program ini terdapat pada modul web (*folder scara_web*).

Program ini terdiri atas beberapa *file* dengan *file* utamanya adalah *file main.py*. *File main.py* akan melakukan inisialisasi program web ini dengan fungsi *init()* yang juga berfungsi untuk memanggil fungsi-fungsi lain, seperti fungsi *jinja_setup()* untuk melakukan pengaturan *library Jinja*, fungsi *db_setup()* untuk melakukan pengaturan *database*, dan fungsi *routes_setup()* untuk melakukan pengaturan *routing* aplikasi. Hal-hal yang berkaitan dengan *database* didefinisikan di dalam *file database.py*. Di dalamnya terdapat fungsi-fungsi untuk berinteraksi dengan *database*, seperti mengambil data dari *database* atau menambahkan data ke dalam *database*. Hal-hal yang berkaitan dengan proses *routing* aplikasi terdapat pada *file routes.py*. Dari *file routing* ini proses kemudian akan diteruskan ke *file views.py* yang memproses *request* dari pengguna dan memberikan hasil dari pemrosesan tersebut kepada pengguna. Tampilan aplikasi didefinisikan/dijelaskan pada *file* yang terdapat dalam *folder templates*. Misalnya, *file index_block.html* merupakan *file* untuk tampilan halaman daftar *block*.

2) Realisasi Blockchain: Dalam merealisasikan *blockchain*, terdapat beberapa fungsi yang dibuat, di antaranya sebagai berikut.



Gbr. 5 Struktur *database.*

- Fungsi untuk melakukan proses *hash* data *block*,
- fungsi untuk menandatangani data *voting* pada *block*,
- fungsi untuk melakukan verifikasi tanda tangan pada *block*,
- fungsi untuk memverifikasi *block*, dan
- fungsi untuk melakukan verifikasi kumpulan *block* atau *blockchain*.

3) Realisasi Pengiriman Block: Proses pengiriman *block* atau *broadcasting block* didefinisikan oleh fungsi *broadcast_block()* yang ada pada *file net.py*. Fungsi ini dipanggil ketika pengguna telah menambahkan *block* yang valid ke dalam *database*-nya. Isi dari fungsi *broadcast_block()* ini adalah sebagai berikut.

```
# Bagian dari file:
# scara_blockchain/net.py
import asyncio as aio
import json

nodeAddr = Tuple [str, int]

async def broadcast_block (address : NodeAddr,
data: Block)
loop = aio.get_event_loop()
(host, port) = address

Reader, writer = await aio.open_connection(host,
port, loop=loop)

block_data = json.dumps ({
'timestamp' : data[0], timestamp(),
'node_id' : data[1],
'signature' : data[2],
'prev_hash' : data[3],
'hash' : data[4],
'candidates' data[5] })

Writer.write(block_data.encode() + b'\n')

# T000 : Add confirmation from destination node
# data = await reader.read(100)
# print ('Received : %r' % data.decode ())
Writer.close()
```

Fungsi ini akan mengirimkan data *block* ke satu *node* sesuai dengan alamat IP dan *port* yang diberikan. Utntuk pengiriman ke banyak *node*, digunakan fungsi lain yang dibuat asinkron agar prosesnya dapat berjalan di *background* dan dapat berjalan bersamaan dengan bagian program lainnya. Dengan demikian, tidak ada program yang terhambat ketika proses pengiriman data *block* ini dilakukan.

4) Realisasi Peneirimaan Block: Block baru diperoleh dengan mendengarkan *broadcast* data *block* dari *node*/*peer* lain. Proses mendengarkan atau *listening broadcast* ini merupakan *background task* atau proses yang berjalan di "belakang". Proses ini dijalankan ketika program utama dimulai.

5) Realisasi Database: Database pada program ini menggunakan SQLite3 dan disimpan pada *file* bernama *database.db*. Struktur dari *database* tersebut diperlihatkan pada Gbr. 5.

Database ini terdiri atas tiga tabel, yaitu tabel *blocks*, tabel *candidates*, dan tabel *nodes.* Tabel *blocks* merupakan tabel tempat menyimpan data *block*, tabel *candidates* merupakan tabel untuk menyimpan data *vote* bagi masing-masing kandidat di setiap *block*, dan tabel *nodes* adalah tabel yang digunakan untuk menyimpan informasi dari *nodes*. Pembuatan *database* ini dilakukan oleh program pada *init_db.py*. *File init_db.py* ini dapat dijalankan oleh pengguna sebelum menjalankan program utama untuk membuat *database*.

IV. HASIL

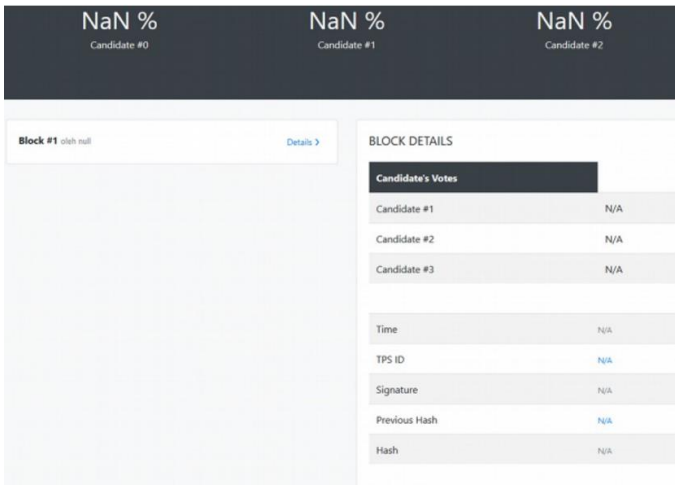
A. Pengujian

Terdapat beberapa parameter yang diuji pada sistem yang dibuat. Parameter-parameter ini dijadikan sebagai sebuah acuan dalam pengambilan hasil atau target yang terukur. Berikut adalah parameter-parameter yang diuji.

- Keberhasilan dalam pembuatan *block* baru oleh *node* beserta tanda tangan dan juga *hash* yang dihasilkan.
- Verifikasi *block* yang tersimpan pada *database*.
- Penghitungan atau akumulasi jumlah *vote* tiap kandidat.
- Pengiriman data *block* dari satu komputer ke komputer lain (*peer-to-peer*).

```
id          timestamp                  node_id     signature   prev_hash   hash
----------  -------------------------  ----------  ----------  ----------  ----------------------------------------------------------------
1           2019-06-09 02:05:38.700631  0          0           0           b9c72ac92ef5ba239a1ad3214ad0d29b5e9277c4d94d9d3edbe0415cc0963f3e
sqlite>
sqlite> SELECT * FROM candidates;
cand_0      cand_1      cand_2      block_id
----------  ----------  ----------  ----------
0           0           0           1
sqlite>
sqlite> SELECT * FROM nodes;
id          name        pub_key
----------  ----------  ----------------------------------------------------------------------------------------------------------------------
1           TPS 1       3059301306072a8648ce3d020106082a8648ce3d03010703420004d53a71d7f8c493c3b7c5c29c626efadcf8206ca7e28ea1660b79681e1a74d35a7877
2           TPS 2       3059301306072a8648ce3d020106082a8648ce3d03010703420004d70878a3c6dc14822e7c85bdaa815fc9a6c5735ad2df8a22c6ae478c88cd0ac7b538
3           TPS 3       3059301306072a8648ce3d020106082a8648ce3d03010703420004d0643c9a8199e00d85ff0d272395fae1aeb128fad8691264f0bb8fbb3d01fca9dd8
4           TPS 4       3059301306072a8648ce3d020106082a8648ce3d03010703420004b4bc5244a6cc50681fea5c77bd5a8177a18dc35a0ebfc6e8b2fb6753384ea002f9b6
5           TPS 5       3059301306072a8648ce3d020106082a8648ce3d03010703420004b8fbb34e472de6f5401f0e744485d58843e62c9e389acdfc1591e6349f81e78cb642
6           TPS 6       3059301306072a8648ce3d020106082a8648ce3d03010703420004863 9c44108a0e8836af7c1ddfe9b41362a56ac3c59307e5ebfef13d2684d87ad93a5
```

Gbr. 6 Isi tabel *blocks*, *candidates*, *nodes*.



Gbr. 7 Tampilan halaman *blocks*.

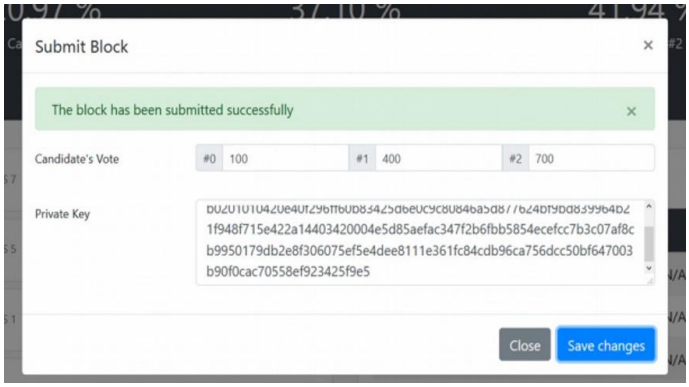
Pengujian dilakukan pada komputer PC dan pada laptop. Keduanya terhubung secara *wireless* ke jaringan Wi-Fi yang sama, yang disediakan oleh *router*. Komputer PC dan laptop yang digunakan telah terpasang Python 3 dan telah menginstal beberapa *library* yang dibutuhkan oleh program. Karena antarmuka program berbasis web, pada masing-masing komputer tersebut telah diinstal aplikasi *browser*.

Setiap komputer membuat sebuah *virtual environment* untuk memisahkan pemakaian *library* dengan versi berbeda dengan program lainnya. *Virtual environment* ini dibuat dengan menggunakan *virtual envwrapper* yang merupakan *library* atau program Python yang dapat diinstal menggunakan *Python package manager*, *pip*. *Virtual environment* yang telah dibuat tersebut kemudian diaktifkan. *Library-library* yang dibutuhkan oleh program kemudian diinstal pada *virtual* e*nvironmen*t tersebut dengan mengacu pada *file requirements.txt* yang terdapat pada *folder* program. Proses instalasi ini dilakukan dengan memberikan perintah: "pip *install* -r requirements.txt" pada *command prompt*. Setelah *library* terinstal, program membutuhkan sebuah *database* yang dapat dibuat dengan menggunakan program pada *file init_db.py* yang telah dibuat. Dengan memasukkan perintah *python init_db.py* pada *command prompt*, *file database* SQLite dengan nama *database.db* akan dibuat pada *folder* program. *Database* tersebut berisi tiga buah tabel, yaitu *blocks*, *candidates*, dan *nodes*. Awalnya tabel *blocks* dan *candidates* hanya berisi satu baris data, yaitu *data block genesis* dan tabel *nodes* tidak memiliki data sama sekali. Untuk itu, dibuat data sampel untuk pengujian dengan menggunakan fungsi yang ada pada program *init_db.py*, yaitu: *create_test_nodes()*. Setelah tabel *nodes* terisi, seperti pada Gbr. 6, akan muncul *file* dengan nama *tps_keys.json* yang berisi data mengenai *private key* dan *public key* tiap *nod*e yang dibuat. Kedua *key* ini nantinya akan digunakan dalam proses pembuatan *block.* Program kemudian dapat dijalankan dengan menjalankan *file app.py* pada terminal atau *command prompt*. Setelah *file app.py* dijalankan, alamat *http://localhost:8080/blocks* dibuka pada aplikasi *browser*. Tampilan pada alamat tersebut ditunjukkan pada Gbr. 7. Tulisan "NaN %" merupakan hasil akumulasi jumlah *vote* tiap kandidat. Tulisan "NaN" sendiri muncul karena data pada *database* masih kosong.

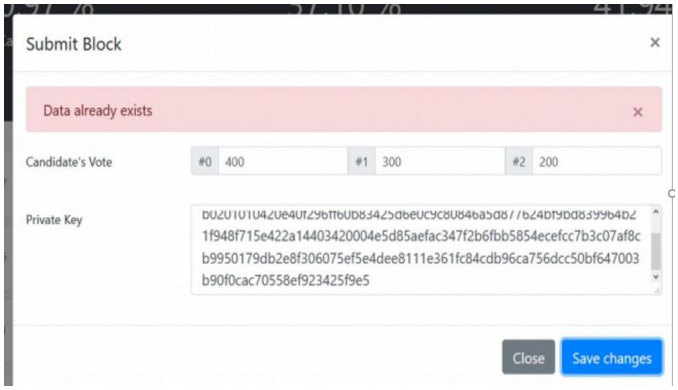
B. Hasil Pengujian

1) Pembuatan Block Baru: Pada pengujian ini ditampilkan dua kondisi: (1) kondisi saat *block* berhasil dibuat atau ditambahkan; dan (2) kondisi ketika *block* gagal ditambahkan atau *error*.

Pembuatan *block* baru yang berhasil ditunjukkan oleh Gbr. 8 Ketika *block* berhasil ditambahkan, akan muncul pesan bahwa *block* telah berhasil ditambahkan. Selain itu, *block* yang telah berhasil ditambahkan pun akan segera muncul pada daftar *bloc*k yang ada pada *database*, tanpa perlu dilakukan muat ulang (*reload*). Ketika terjadi kegagalan saat pembuatan *block*, akan muncul pesan *error* seperti pada Gbr. 9 (di dalam kotak berwarna merah). Pesan *error* yang muncul dapat berbeda-beda, tergantung pada jenis *error* yang terjadi. Pesan *error* yang ditampilkan pada Gbr. 9 sendiri merupakan pesan *error* yang terjadi karena telah terdapat *block* yang di-*submit* oleh *node* dengan *private key* yang dimasukkan. Hal ini merupakan ketentuan dari *block* pada sistem ini, yaitu satu *node* hanya dapat melakukan *submit* satu *bloc*k saja.



Gbr. 8 Pembuatan *block* baru (kondisi berhasil).



Gbr. 9 Pembuatan *block* baru (kondisi gagal - 1).

Gbr. 10 Pembuatan *block* baru (kondisi gagal – 2).

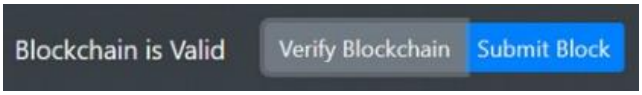
Kondisi lainnya diuji ketika *private key* yang dimasukkan salah akibat jumlah karakter *private key* yang lebih pendek atau lebih panjang dari yang seharusnya. Kesalahan karena *private key* ini akan memunculkan pesan *error*: "*Wrong private key*" seperti yang ditunjukkan pada Gbr. 10.

Pengujian-pengujian yang dilakukan pada proses penambahan *block* terangkum pada Tabel I. Pengujian tersebut dilakukan pada kasus ketika *block* harus berhasil dibuat dan pada kasus ketika *block* harus gagal dibuat.

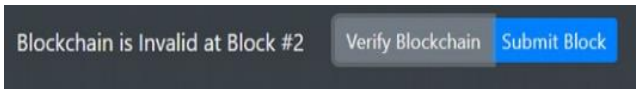
Ada beberapa hal yang dipertimbangkan dalam pembuatan *block* baru, yaitu *private key* yang dimsukkan harus benar dan ketentuan bahwa satu *node* hanya dapat membuat satu *block*. *Private key* yang dimasukkan pada pembuatan *block* ini adalah *private key* dengan tipe *Elliptic-Curve Cryptography* (ECC) dengan besar *private key* adalah 256 bit. *Private key* ini

TABEL I
PENGUJIAN PADA PROSES PENAMBAHAN *BLOCK*

Pengujian	Respons Hasil		
	***Block* Berhasil Dibuat**	***Block* Baru Muncul**	**Jenis Pesan**
Penambahan *block* baru dengan *private-key* yang valid	Ya	Ya	*Success*
Penambahan block baru dengan *private-key* yang tidak valid	Tidak	Tidak	*Wrong Private Key*
Penambahan *Block* baru dengan *private-key* milik node tidak terdaftar	Tidak	Tidak	*Wrong Private Key*
Penambahan *block* dengan memasukkan *private-key* yang digunakan pada pembuatan *block* sebelumnya	Tidak	Tidak	*Data Already Exist*



Gbr. 11 Verifikasi *blockchain* (kondisi valid).



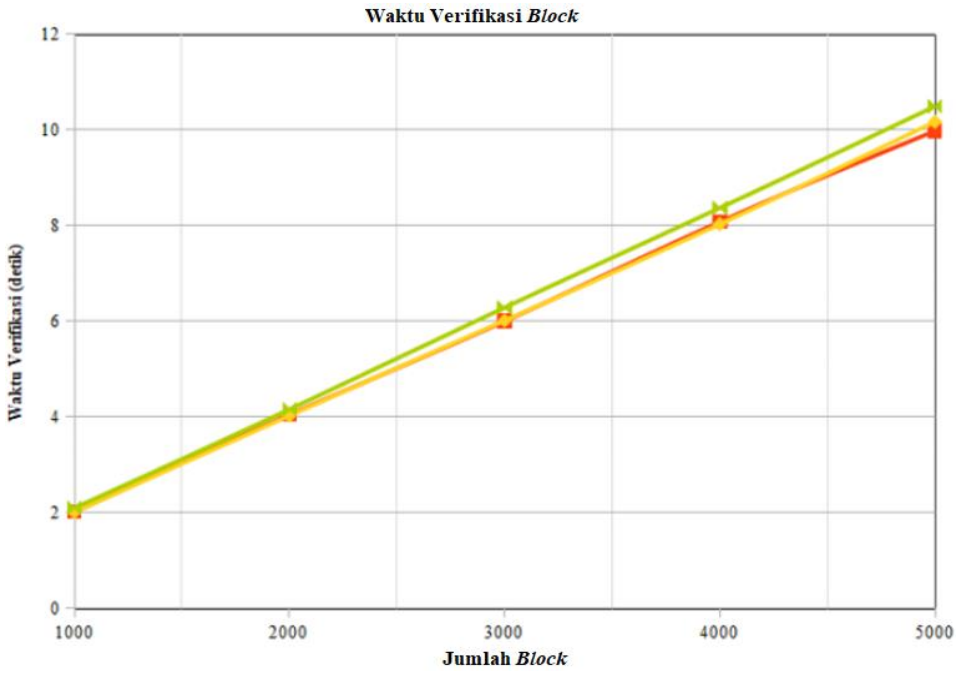
Gbr. 12 Verifikasi *blockchain* (kondisi invalid).

nantinya akan digunakan untuk melakukan proses penandatanganan data pada *block*. Selain itu, dari *private key* ini juga dapat diperoleh *public key* yang kemudian akan menjadi ID dari *node*. *Public key* yang didapat dari *private key* yang dimasukkan akan digunakan untuk mencari *block* pada *database* dengan *node ID* yang sama dengan *public key* ini. Jika ditemukan, *block* baru dengan *public key* ini akan ditolak untuk mencegah kepemilikan lebih dari satu *block* oleh satu *node*. Alasan dari satu *node* hanya boleh membuat satu *block* adalah untuk mencegah proses manipulasi jumlah *vote* oleh *node* yang *valid*/legal. Satu *node* yang *valid* atau terdaftar mungkin saja akan membuat *block* sebanyak mungkin, sehingga jumlah *vote* dapat ditambah semakin banyak, baik itu karena dilakukan secara sengaja atau tidak sengaja.

2) Verifikasi Blockchain: Verifikasi *blockchain* dilakukan dengan menekan tombol "*Verify Blockchain*". Apabila *blockchain* yang dimiliki valid, maka akan muncul tulisan "*Blockchain is Valid*" seperti pada Gbr. 11.

Sebaliknya, apabila terdapat kesalahan pada *blockchain* dikarenakan adanya kesalahan tanda tangan, *hash*, atau data yang tidak sesuai dengan *hash* maupun tanda tangan, akan muncul tulisan "*Blockchain is Invalid at Block #N*". Pada Gbr. 12 terlihat bahwa *blockchain* yang dimiliki invalid beserta dengan nomor *block* yang invalid. Dalam gambar tersebut, *block* nomor 2 invalid. Ini dikarenakan sebelumnya data *vote* untuk salah satu kandidat pada *block* nomor 2 telah diubah dan membuat tanda tangan menjadi salah.

Proses verifikasi ini membutuhkan waktu yang bervariasi, bergantung pada jumlah *block* yang ada pada *database* dan



Gbr. 13 Rata-rata waktu verifikasi *block*.

jumlah kandidatnya. Data hasil pengukuran waktu verifikasi ini diperlihatkan pada Gbr. 13.

Berdasarkan Gbr. 13 yang menunjukkan lama waktu yang dibutuhkan untuk melakukan proses verifikasi *blockchain*, terlihat bahwa lamanya waktu verifikasi terhadap jumlah *block* bersifat linier. Terdapat selisih waktu hampir 1 detik antara *block* yang berisi dua kandidat per *block* dengan *block* yang berisi tiga puluh kandidat per *block* pada proses verifikasi sebanyak 5.000 *block*. Oleh karena itu, dapat diperkirakan bahwa pada jumlah *block* yang besar, misalnya 800.000 *block*, waktu yang dibutuhkan untuk proses verifikasi ini adalah sebesar 1.600 detik, atau 26,67 menit.

Block yang disimpan pada *database* hanya *block* yang *valid* saja. Apabila terdapat *block* yang tidak *valid*, keseluruhan *block* tidak akan *valid* dan proses selanjutnya, yaitu penghitungan *vote*, tidak dapat dilakukan. Verifikasi ini juga untuk mencegah terjadinya manipulasi atau perubahan data pada *block* yang akan memengaruhi hasil penghitungan *vote* akhir.

Proses verifikasi ini dimulai dari *block* terendah ke *block* tertinggi atau dimulai dari *block* ke-2 (*block* ke-1 adalah *block genesis* dan tidak diperhitungkan dalam proses verifikasi). Oleh karena itu, apabila verifikasi ini mendapatkan *block* yang tidak *valid*, proses verifikasi menuju *block* seterusnya akan dihentikan karena *block* setelahnya tidak dapat dianggap valid lagi. Karena terdapat pemberitahuan *block* spesifik yang tidak *valid*, dapat diketahui juga pemilik *block* tersebut dan bagian yang tidak *valid*, *hash*-nya atau tanda tangannya.

3) Penghitungan Jumlah Vote Tiap Kandidat: Jumlah *vote* akan ditampilkan secara *real-time* dalam persen. Setiap ada perubahan pada *database* dikarenakan adanya *block* baru yang bertambah, nilai persen *vote* tiap kandidat akan dihitung dan ditampilkan ulang.

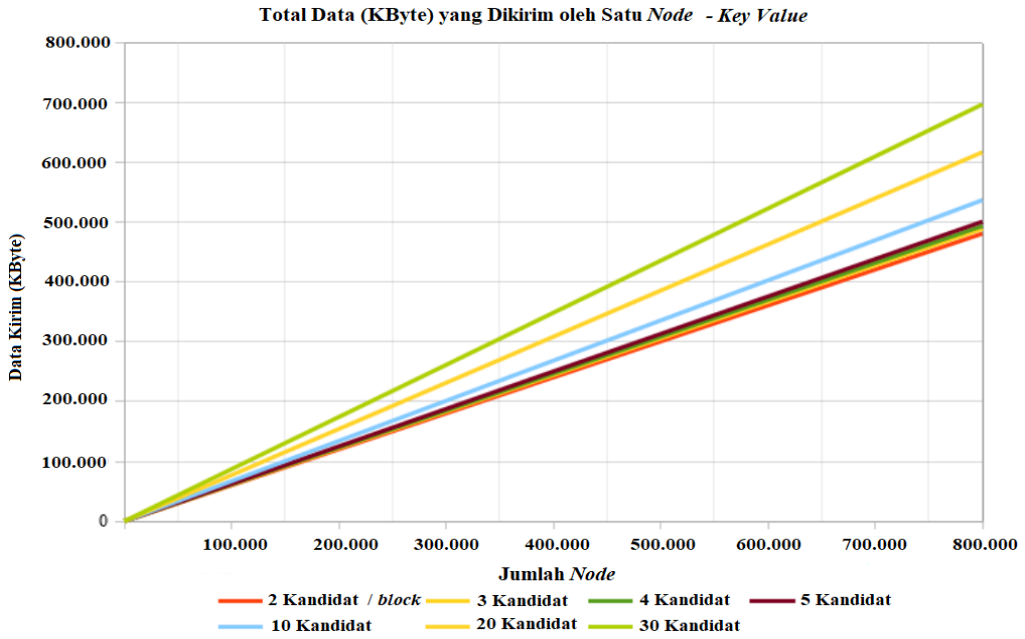
Gbr. 14 Penghitungan jumlah *vote* tiap kandidat.

```
sqlite> SELECT * FROM candidates;
cand_0      cand_1      cand_2      block_id
----------  ----------  ----------  --------
0           0           0           1
100         200         300         2
450         550         300         3
100         400         700         4
```

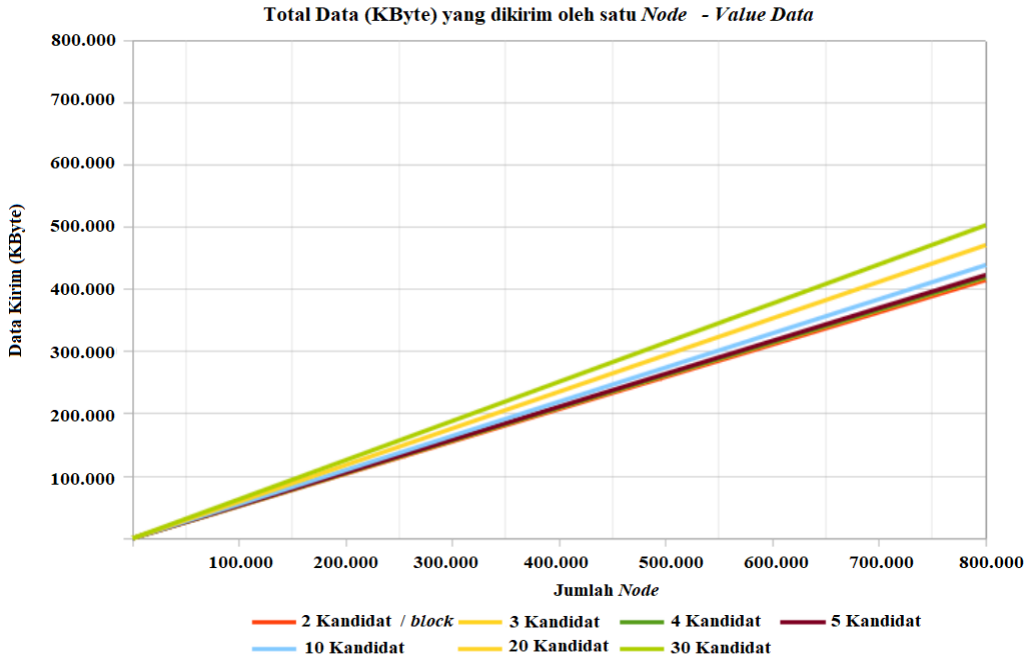
Gbr. 15 Data *vote* tiap kandidat pada *database*.

Untuk membuktikan bahwa penghitungan tersebut benar, dapat dibandingkan dengan jumlah *vote* yang ada pada *database*. Contohnya, hasil penghitungan *vote* pada Gbr. 14 menunjukkan 20,97% *vote* untuk kandidat #0; 37,10% *vote* untuk kandidat #1; dan 41,94% *vote* untuk kandidat #2. Jika dibandingkan dengan data yang ada pada *database* seperti yang diperlihatkan pada Gbr. 15, kandidat #0 memiliki total 650 *vote*, kandidat #1 memiliki total 1.150 *vote*, dan kandidat #2 memiliki total 1.300 *vote*. Dalam persen akan menghasilkan nilai yang sama dengan yang ditampilkan pada Gbr. 14, yaitu kandidat #0 sebesar 20,97%, kandidat #1 sebesar 37,10%, dan kandidat #2 sebesar 41,94%.

Penghitungan ini dilakukan setiap ada *block* baru. Artinya program tidak melakukan proses penghitungan tiap 1 detik, tetapi proses penghitungan dipicu oleh munculnya *block* yang telah diterima. Ini dilakukan untuk meringankan beban aplikasi, karena untuk data yang sangat banyak, proses ini dapat memakan sumber daya dan waktu yang banyak tiap detiknya. Proses penghitungan ini masih memiliki kelemahan, yaitu penghitungan masih tetap dilakukan meskipun *blockchain* tidak *valid*. Nilai persentase *vote* tiap kandidat tetap berubah saat *blockchain* tidak *valid*.



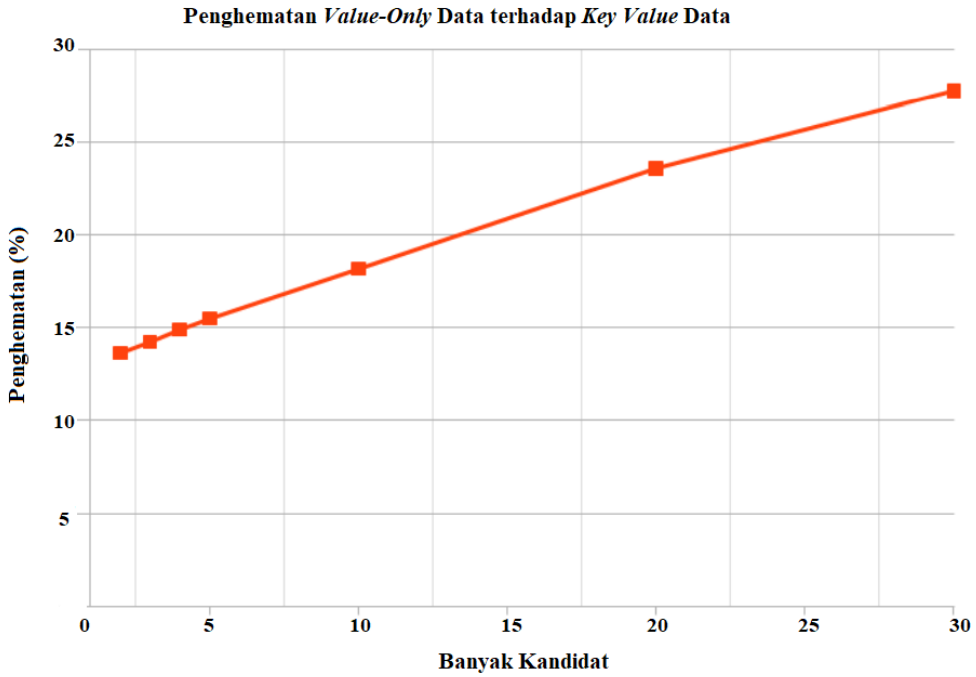
Gbr. 16 Grafik ukuran data *block* yang dikirim (*key-value*).



Gbr. 17 Grafik ukuran data *block* yang dikirim (*value* saja).

4) Pengiriman Block: Node tujuan akan mendengarkan *broadcast block* dari *node-node* lainnya. *Block* yang diterima dicek terlebih dahulu sebelum *node* tujuan ini menambahkan *block* tersebut pada *database blockchain*-nya. Apabila *block* yang diterima tidak *valid*, *block* tersebut akan ditolak atau diabaikan. Sebaliknya, apabila *block* yang diterima merupakan *block* yang *valid*, *block* ditambahkan ke dalam *database blockchain*.

Pengujian lainnya adalah dengan mencatat besar ukuran data *block* yang dikirim dan diterima *node*. Struktur data *key-value* memiliki *key* seperti *timestamp*, *hash*, *prev.* dan *hash*, beserta *value* atau isi dari *key* tersebut. Dengan struktur ini (*key-value*), suatu *block* yang memiliki dua kandidat dengan jumlah *vote* maksimal adalah 300 *vote* memiliki ukuran/besar data 601 byte. Selain dengan dua kandidat, diuji juga dengan beberapa jumlah kandidat. Jumlah *node* pun divariasikan, yaitu dimulai dari 100.000 *node* hingga 800.000 *node*. Hasilnya ditunjukkan pada grafik Gbr. 16 yang bersifat linier. Sedangkan Gbr. 17 menunjukkan ukuran data *block* yang dikirim berupa *value* saja.



Gbr. 18 Penghematan *value-only* data terhadap *key-value* data.

Grafik pada Gbr. 18 memperlihatkan besarnya pengurangan (dalam persen) besar data yang dikirim menggunakan data dengan *value* saja terhadap data yang dikirim menggunakan data *key-value*. Terlihat bahwa penghematan dapat mencapai 30% pada data dengan jumlah kandidat setiap *block* sebanyak tiga puluh kandidat.

Berdasarkan data pada pengujian pembuatan *block* (Gbr. 16 sampai Gbr. 18), besar data yang harus dikirim oleh satu *block* terhitung wajar untuk jumlah *node* yang sangat banyak (500 MB hingga 700 MB pada 800.000 *node*) ditambah dengan data yang harus diunduh oleh masing-masing *node* dari *node* lainnya, sebesar 500 MB hingga 700 MB juga. Besarnya data ini juga dapat dikurangi dengan menurunkan bit pada algoritme *hash* dan tandatangan digital yang digunakan, tentunya dengan mengorbankan aspek keamanan.

V. KESIMPULAN

Pembuatan dan pengimplementasian sistem pencatatan hasil pemungutan suara menggunakan *blockchain* telah berhasil dilakukan. Sistem ini memungkinkan proses pembuatan *block*, verifikasi *block*, dan juga penghitungan hasil *voting*. Penggunaan *blockchain* ini dapat mencegah terjadinya perubahan data *voting* dan karena sistem ini bersifat digital, proses penghitungan suara dapat dilihat secara *real-time*. Pembuatan dan pengimplementasian sistem pencatatan hasil pemungutan suara menggunakan *blockchain* pada jaringan *peer-to-peer* telah dilakukan dengan berhasilnya pengiriman data *block* oleh satu *node* ke *node* lainnya pada jaringan *peer-to-peer*. Aplikasi berbasis web sebagai antarmuka aplikasi ini juga telah berhasil diimplementasikan. Aplikasi ini dapat diakses melalui *browser* setelah server lokal aplikasi ini dijalankan. Server yang melayani pemrosesan data *block* dibuat secara asinkron yang memungkinkan pembuatan program *non-blocking*. Hal ini membuat program tidak saling menunggu selesainya satu proses untuk melanjutkan ke proses selanjutnya.

REFERENSI

[1] "Undang-Undang No. 7 Tahun 2017 tentang Pemilihan Umum," Pemerintah Republik Indonesia, 2017.

[2] S. Nakamoto (2009) "Bitcoin: A Peer-to-Peer Electronic Cash System," [Online], https://bitcoin.org/bitcoin.pdf, tanggal akses: 18-Jan-2017.

[3] A. Narayanan, J. Bonneau, E. Felten, A. Miller, dan S. Goldfeder, *Bitcoin and Cryptocurrency Technologies*, Princeton, USA: Princeton University Press, 2016.

[4] F.S. Hardwick, A. Gioulis, R.N. Akram, dan K. Markantonakis, "E-Voting with Blockchain: An E-Voting Protocol with Decentralisation and Voter Privacy," *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, hal. 1561-1567.

[5] R. Hanifatunnisa dan B. Rahardjo, "Blockchain Based E-Voting Recording System Design," *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, 2017, hal. 1-6.

[6] A. Barnes, C. Brake, dan T. Perry, "Digital Voting with the Use of Blockchain Technology," Plymouth University, Course Report, Plymouth, UK, hal. 1-19, 2016.

[7] S. Heiberg dan J. Willemson, "Verifiable Internet Voting in Estonia," *2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE)*, 2014, hal. 1-8.

[8] C. Meter (2017) "Design of Distributed Voting Systems," [Online], http://arxiv.org/abs/1702.02566, tanggal akses: 18-Jan-2017.

[9] (2016) "Follow My Vote (n.d.): Why Online Voting," [Online], https://followmyvote.com/, tanggal akses: 1-Jan-2017.

[10] "The Open Group Base Specifications Issue 7, 2018 edition - IEEE Std 1003.1-2017 (Revision of IEEE Std 1003.1-2008)," IEEE and The Open Group, 2018.