Implementasi Ethereum Blockchain dan Smart Contract pada Jaringan Smart Energy Meter

Article *in* MULTINETICS · October 2021

DOI: 10.32722/multinetics.v7i1

3 authors:

Anggun mugi Mabruroh
Telkom University
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Favian Dewanta
Telkom University
30 PUBLICATIONS **68** CITATIONS
SEE PROFILE

Aulia Arif Wardana
Telkom University
28 PUBLICATIONS **101** CITATIONS
SEE PROFILE

Implementasi Ethereum *Blockchain* dan *Smart Contract* pada Jaringan *Smart Energy Meter*

Anggun Mugi Mabruroh[1], Favian Dewanta[2], Aulia Arif Wardana[3]
[1, 2]Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[3] Informatika, Fakultas Informatika, Universitas Telkom
Telkom University, Jl. Telekomunikasi No. 1, Terusan Buahbatu – Bojongsoang, Sukapura, Dayeuhkolot, Bandung, Jawa Barat

Diterima: 31 Agustus 2021. Disetujui: 23 September 2021. Dipublikasikan: 18 Oktober 2021.

Abstract – Conventional databases on Internet of Things (IoT) technology have a centralized storage system on a database server. If the server is damaged and fails to work, then the service will also crash and there is also the possibility of data loss. Therefore, this study proposes the creation of an IoT device, namely a smart energy meter, by implementing a blockchain system as a distributed database. This blockchain system implements a private blockchain network that is realized using the Ethereum framework. The sensor data will be read by the Raspberry Pi 4B and sent to node 1 via MQTT. Node 1 will save the data to the block. Two Ethereum account nodes will validate the block. If accepted then the block will be stored on the blockchain and create a new block chain. In the storage process, each node will work according to a predefined smart contract created using solidity and accessed using the web3 API. Successfully saved data will be displayed to the user's web. Based on the results of testing the system delay from sensors to the website, the use of blockchain compared to conventional databases has an effect on increasing the overall delay, which is 8.5274 seconds, which is the average calculation of the submit transaction delay for transaction processing and transaction receipt for the data verification process. However, these results do not reduce the quality of the smart energy meter system because this system does not require a fast response in reporting the results of electric power usage. In addition, the number of nodes and block size do not affect the performance of the proof of authority consensus algorithm used in this blockchain system.

Keywords: Internet of Things, blockchain, smart contract, Ethereum, MQTT

Abstrak— Database* konvensional pada teknologi *Internet of Things* (IoT) memiliki sistem penyimpanan secara terpusat pada *database* server. Jika server tersebut rusak dan gagal bekerja, maka layanan pun akan ikut lumpuh dan ada kemungkinan juga kehilangan data. Oleh karena itu, penelitian ini mengusulkan pembuatan perangkat IoT, yakni *smart energy meter*, dengan menerapkan sistem *blockchain* sebagai *database* terdistribusi. Sistem *blockchain* ini menerapkan jaringan privat *blockchain* yang direalisasikan dengan menggunakan *framework* Ethereum. Data sensor akan dibaca oleh Raspberry Pi 4B dan dikirimkan ke *node* 1 melalui MQTT. *Node* 1 akan menyimpan data ke blok. Dua *node* akun Ethereum akan memvalidasi blok tersebut. Jika diterima maka blok akan disimpan pada *blockchain* dan membuat rantai blok baru. Dalam proses penyimpanan, setiap *node* akan bekerja berdasarkan dengan *smart contract* yang telah didefinisikan sebelumnya yang dibuat dengan menggunakan solidity dan diakses menggunakan web3 API. Data yang berhasil disimpan akan ditampilkan ke web pengguna. Berdasarkan hasil pengujian *delay* sistem dari sensor hingga ke *website*, penggunaan *blockchain* jika dibandingkan *database* konvensional berpengaruh terhadap penambahan *delay* secara keseluruhan, yakni 8,5274 detik yang merupakan kalkulasi rata-rata *delay submit transaction* untuk proses transaksi dan *transaction receipt* untuk proses verifikasi data. Namun, hasil ini tidak mengurangi kualitas sistem *smart energy meter* karena sistem ini tdak memerlukan respon yang cepat dalam melaporkan hasil penggunaan daya listrik. Selain itu, jumlah *node* dan ukuran blok tidak mempengaruhi kinerja algoritma konsensus *proof of authority* yang digunakan dalam sistem *blockchain* ini.

Kata kunci: *Internet of Things, blockchain, smart contract*, Ethereum, MQTT

I. PENDAHULUAN

Internet of Things (IoT) memberikan jaminan keamanan, kenyamanan, kemudahan dan peningkatan kualitas hidup pengguna, IoT menghubungkan antar perangkat yang dapat terhubung dengan internet[1]. IoT memungkinkan sensor dan *actuator* yang terhubung ke komputer untuk memfasilitasi produk dan layanan baru dengan mengurangi biaya, meningkatkan efisiensi, dan meningkatkan kegunaan sistem yang ada[2]. Implementasi IoT ada dipelbagai bidang dan tempat seperti *smart home*, *smart city*, *smart corporate* dan lain sebagainya. *Smart home* adalah salah satu dari penerapan IoT segala benda yang ada di rumah dapat saling berkomunikasi dan berinteraksi melalui internet. Sistem otomasi rumah saat ini sangat

populer bahkan sampai dikaitkan dengan tenaga listrik serta kebutuhan listrik juga sangat tinggi hampir semua peralatan rumah tangga menggunakan listrik[3].

Pengelolaan jaringan secara terpusat membutuhkan infrastruktur komunikasi dan informasi mahal. Selain itu, banyaknya ancaman serangan siber di internet menyebabkan pengelolaan *database* terpusat menjadi kurang aman. Sebagai solusinya, teknologi *blockchain* saat ini banyak diusulkan sebagai salah satu solusi untuk menambah keamanan, yakni dengan mendistribusikan *database* ke semua ledger dan juga dengan tambahan *hash* yang saling terhubung dengan blok data sebelumnya, sebagaimana dibahas di penelitian[4][5].

Teknologi pengukuran data listrik yang sudah ada masih menggunakan sistem terpusat di mana keamanan data listrik masih kurang aman yaitu disimpan pada suatu *database* pusat. Berdasarkan pada penelitian sebelumnya pembuatan sistem *smart energy* meter dengan menggunakan algoritma AES untuk pertukaran data pada jaringan IoT[6]. Data *blockchain smart energy* meter dibangun di *cloud storage*, setiap *cloud storage* memiliki salinan data lalu ketika blok data disimpan, setiap *cloud storage provider* harus verifikasi blok data tersebut[7]. Pada penelitian tersebut, jika administrator pusat suatu saat tidak bertanggung jawab dengan memanipulasi data maka pengguna listrik tidak mengetahui jika data mereka sudah dimanipulasi. Selain itu jika suatu saat server *database down* maka akan kehilangan data pada *database* tersebut.

Pada penelitian ini, penulis melakukan perancangan, pembuatan dan penelitian mengenai perangkat IoT *smart energy meter* untuk pemantauan listrik yaitu sensor daya akan mendeteksi data daya pada suatu beban. Kemudian data daya tersebut akan dikirimkan dari Raspberry Pi ke *node* 1 menggunakan *Message Queuing Telemetry Transport* (MQTT) karena *communication overhead* yang rendah dan efisiensi sumber daya[8]. Data yang diterima kemudian diproses untuk disimpan ke *database blockchain*, lalu data akan ditampilkan pada web pengguna.

II. TINJAUAN PUSTAKA

A. *MQTT*

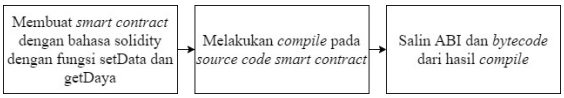
Protokol MQTT atau *Message Queuing Telemetry Transport* adalah protokol yang dirancang khusus untuk komunikasi "mesin ke mesin" di mana bekerja dengan *overhead* yang rendah (> 2 *byte*) sehingga konsumsi *power supply* cukup kecil[9]. MQTT membutuhkan *node publisher*, *subscriber* dan broker.

B. *Blockchain*

Blockchain adalah sistem *cryptocurrency* pertama yang diterapkan oleh *bitcoin* di mana *blockchain* mengatasi permasalahan keamanan data dan keselamatan data IoT yang rentan[10].

Blockchain adalah *database* yang terdistribusi untuk mendistribusikan struktur data dan dibagikan ke seluruh *node* anggota[11]. Struktur data tersebut berupa daftar *record* yang disebut blok, di mana di dalam setiap blok tersebut memiliki *timestamp*[11] dan kode *hash* yang terhubung dengan *blockhash* sebelumnya, sehingga setiap blok saling terkait satu sama lain menggunakan *hash* masing-masing blok tersebut.

C. *Smart Contract*



Gambar 1. Proses Compile untuk Mendapatkan *Bytecode* dan ABI

Smart contract yaitu kontrak yang dibuat untuk perjanjian antara beberapa *node* berdasarkan jenis algoritma *consensus.* Kontrak tersebut dalam bentuk *source code* di mana penulis menggunakan bahasa solidity. Solidity adalah bahasa pemrograman mirip dengan c++ yang memiliki ekstensi .sol di mana solidity berorientasi obyek untuk merancang *smart contract* agar dapat berjalan di Ethereum Virtual Machine. Di dalam *source code* solidity dapat diprogram untuk mendapatkan data dan menyimpannya serta menampilkan data. File .sol tersebut akan dikompilasi, kemudian menghasilkan *bytecode* yang digunakan untuk referensi fungsi dan kontrak untuk dieksekusi di EVM. Selain *bytecode*, hasil *compile* yang kedua adalah ABI (*Application Binary Interface*) yang merupakan daftar fungsi dan kontrak berbentuk JSON (*Javascript Object Notation*)[12]. Proses *compile smart contract* dapat dilihat pada Gambar 1. *Smart contract* dapat menentukan aturan dan menghasilkan kode untuk diterapkan pada jaringan *blockchain*. *Smart contract* tidak dapat dihapus secara *default*, dan interaksi dengan *smart contract* tidak dapat dihilangkan.

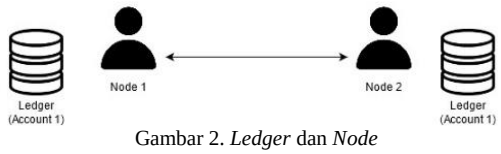
D. *Ethereum*

Ethereum merupakan *platform* untuk menyediakan atau membuat *smart contract* pada teknologi *blockchain*[13]. Ethereum memiliki komputasi virtual yaitu Ethereum Virtual *Machine* (EVM) di mana setiap *node* yang ada di jaringan mengirimkan permintaan agar EVM ini mendapatkan wewenang melakukan komputasi. *Node* lain melakukan verifikasi, validasi dan

melaksanakan komputasi. Hal ini menyebabkan perubahan status di EVM, yang akan disebarkan ke seluruh jaringan. Permintaan komputasi disebut permintaan transaksi di mana semua catatan transaksi serta status EVM disimpan di *blockchain*, kemudian disimpan dan disetujui oleh semua *node*[14]. EVM mengeksekusi kode ABI dan dapat diakses publik oleh pengguna manapun dan *permissionless*. Hasil dari *blockchain* Ethereum merupakan aplikasi terdistribusi. Pada Ethereum, pembuatan *smart contract* menggunakan bahasa solidity. *Smart contract* Ethereum memiliki *contract* yang kuat, publik dan transparan. Ethereum menggunakan algoritma konsensus *proof of work* dan *proof of authority*[15].

E. Proof of Authority

Algoritma Konsensus *Proof of Authority* memiliki jaringan yang bergantung pada sekelompok *node* otoritas yang merupakan *validator* yang telah disetujui sebelumnya. Kemudian melakukan verifikasi transaksi dan membangun blok baru. *Validator* harus mematuhi serangkaian aturan agar dapat dipercaya. Salah satunya mengharuskan mereka untuk terdaftar di *database blockchain* dengan identitas yang sama dengan yang mereka miliki. Agar jaringan berfungsi, lebih banyak aturan harus diikuti. Pada penelitian ini, penulis menggunakan konsensus *proof of authority* dimana *authority* atau *ledger node* bekerja sebagai *validator* dan melakukan verifikasi publik. Protokol ini dapat digunakan di jaringan publik atau pribadi[16]. Gambar 2 merupakan ilustrasi sederhana mengenai letak *ledger account* 1 pada masing-masing *node*.



Gambar 2. *Ledger* dan *Node*

F. Web3.py API

Web3.py adalah *library* python untuk berinteraksi dan terhubung dengan sistem Ethereum. Web3.py memiliki beberapa API untuk menggunakan fungsi Ethereum diantaranya yaitu untuk terhubung dengan jaringan Ethereum pribadi, memanggil fungsi *smart contract* dan melakukan transaksi yang menghasilkan *blockhash*. Sebelumnya hanya terdapat web3 API berbasis javascript tapi sekarang sudah berkembang dan berevolusi untuk kebutuhan dan kenyamanan pengembang python[17].

G. Geth

Geth atau go-ethereum adalah salah satu implementasi dari protokol Ethereum yaitu bahasa pemrograman *open-source* yang dapat dipasang pada semua sistem operasi[18]. Pada Ethereum terdapat tiga jenis *node* yaitu *full-node*, *light node* dan *node* arsip. *Full-node* memiliki beberapa tugas untuk menyimpan data *blockchain* lengkap, berpartisipasi dalam validasi blok, memverifikasi semua blok dan status. Semua *state* dapat diturunkan dari *full node*. *Full-node* juga melayani jaringan dan menyediakan data berdasarkan permintaan. Selanjutnya *light node* yang memiliki beberapa tugas untuk menyimpan rantai *header* dan meminta yang lainnya. *Light-node* dapat memverifikasi validitas data terhadap akar status di *header* blok. *Light-node* berguna untuk perangkat berkapasitas rendah, seperti perangkat tertanam atau ponsel, yang tidak mampu menyimpan *gigabyte* data *blockchain*. Dan yang terakhir adalah *node* arsip yang memiliki beberapa tugas untuk menyimpan semua yang disimpan dalam simpul penuh dan membangun arsip status historis. *Node* arsip diperlukan jika ingin menanyakan sesuatu seperti saldo akun pada blok. Data ini mewakili unit *terabyte* yang membuat *node* arsip kurang menarik bagi pengguna rata-rata tetapi dapat berguna untuk layanan seperti penjelajah blok, *vendor* dompet, dan analitik rantai[19]. Kedua *node* yang penulis buat yaitu *node* 1 dan *node* 2 menggunakan *full-node*.

III. METODE PENELITIAN

A. Metodologi

Penelitian diawali ketika ada masalah muncul saat meninjau jurnal dan *paper* konferensi tentang bagaimana mengamankan data *Internet of Things*[2]. Beberapa sistem telah mengantisipasi masalah ini dengan menerapkan algoritma AES untuk melakukan enkripsi dan dekripsi data[6]. Namun, jika *administrator database* pusat memanipulasi data maka data tersebut akan berubah. Selain itu jika server *database down* maka akan kehilangan data pada *database* server. Sehingga memerlukan *database* yang tidak terpusat. *Blockchain* merupakan *database* yang desentralisasi dan terdistribusi. Maka dari itu, penulis melakukan penelitian yaitu menerapkan *blockchain* pada sistem penyimpanan data *smart energy meter*.

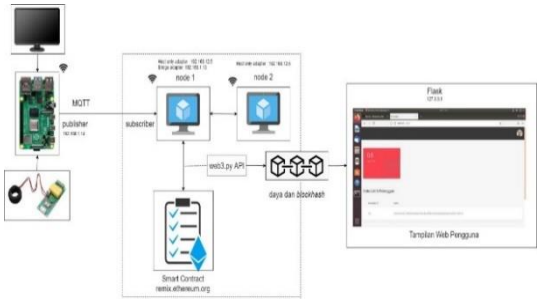
B. Metode DSRM (Design Science Research Methodology)

DSRM adalah metode dalam membangun sistem pada penelitian. DSRM cocok untuk *software developer*, sedangkan pada sistem yang penulis buat membangun perangkat lunak dan perangkat keras. Sehingga DSRM diterapkan pada pembuatan sistem prototipe *blockchain*. Fase pertama adalah identifikasi masalah, yang dibahas pada bagian pendahuluan melalui frasa awal dari metode penelitian. Tahap kedua adalah desain dan pengembangan sistem dibahas pada bagian Desain Sistem, Desain Perangkat *Smart Energy Meter* dan

Desain Perangkat Lunak. Tahap demonstrasi dan evaluasi dibahas pada bagian Hasil dan Pembahasan. Komunikasi dibahas pada bagian Kesimpulan. Metode pengujian yang digunakan untuk menguji prototipe adalah pengukuran *delay* dan performa sistem. Pertama, pengukuran *delay* untuk menguji apakah semua komponen yang digunakan dalam prototipe bekerja sebagaimana mestinya, sedangkan pengukuran performa sistem untuk menentukan apakah prototipe akan menjadi solusi yang tepat dengan menguji kualitas algoritma konsensus *Proof of Authority*, web pengguna dan CPU *node* 1.

C. Desain Sistem

Pada Gambar 3 adalah desain sistem *smart energy meter* yang menerapkan *blockchain* dirancang untuk memberikan keamanan pada penyimpanan *database* dimana menggunakan *blockchain*. Terdapat beberapa tahap untuk data bisa ditampilkan ke web pengguna. Tahap pertama yaitu menghubungkan sensor PZEM-004T dengan Raspberry Pi 4B. Untuk proses pembacaan sensor di Raspberry Pi 4B menggunakan pemrograman python. Setelah data terbaca maka tahap kedua yaitu pengiriman data daya dari Raspberry Pi 4B ke *node* 1 di *virtual machine* virtualbox. Pengiriman tersebut menggunakan protokol komunikasi yaitu MQTT dimana Raspberry Pi 4B sebagai *publisher* atau yang mengirim data dan *node* 1 sebagai *subscriber* yaitu yang menerima data. Dalam penelitian ini penulis menggunakan *library* paho-mqtt yaitu library MQTT dari python. Setelah itu tahap ketiga, data yang diterima di *node* 1 kemudian disimpan ke *database blockchain* dengan memanggil fungsi setData pada *smart contract* menggunakan API web3py. Setelah tersimpan, data tersebut akan ditampilkan di web pengguna memanggil fungsi *get*Daya pada *smart contract* menggunakan API web3py.



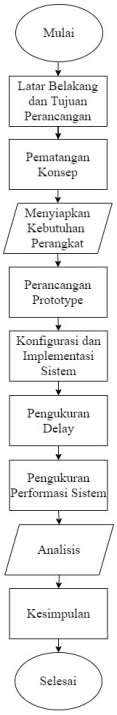
Gambar 3. Desain Sistem

a. Diagram Alir Pembuatan Perangkat dan Sistem *Blockchain*

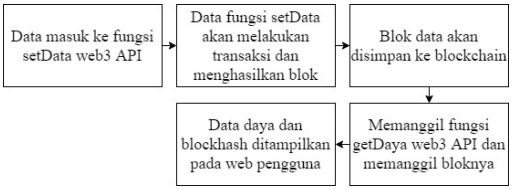
Pada Gambar 4 adalah diagram alir tahapan pengerjaan perangkat *smart energy meter* dan penerapan sistem *blockchain* meliputi beberapa tahap yaitu tahap pertama adalah menentukan latar belakang dan tujuan perancangan implementasi sistem *blockchain* pada jaringan *smart energy meter*. Tahap kedua yaitu pematangan konsep dengan mencari referensi dan informasi pembuatan sistem *blockchain* untuk lalu lintas data *internet of things* dengan tujuan agar nantinya dalam pembuatan dapat berjalan lancar dengan hasil yang diinginkan. Tahap ketiga yaitu menyiapkan kebutuhan perangkat seperti sensor PZEM-004T dan Raspberry Pi 4B untuk membuat perangkat *smart energy meter*. Tahap keempat dan kelima perancangan *prototype* adalah percobaan perangkat *smart energy meter* menerapkan sistem *blockchain* dengan menggunakan fungsi-fungsi yang ada pada program solidity *smart contract*. Tahap keenam adalah pengecekan apakah sistem *blockchain* dapat diterapkan pada jaringan *smart energy meter* atau tidak. Setelah itu ke tahap tujuh yaitu pengujian performa di mana pengujian pada data daya yang diterima *node* 1 dapat disimpan dengan memanggil fungsi setData dari program solidity *smart contract* dan diambil atau ditampilkan kembali dengan memanggil fungsi getDaya serta menghasilkan *blockhash* dari setiap transaksi penyimpanan data daya tersebut. Selain itu pengukuran *delay* pada pengiriman data daya ke *node* 1, pengukuran *delay* pada transaksi dan penampilan kembali data daya tersebut ke web pengguna. Kemudian tahap analisis dan kesimpulan yaitu menganalisis dari beberapa hasil pengukuran *delay* dan membuat kesimpulan dari analisis tersebut.

b. Diagram Blok

Gambar 5 adalah diagram blok proses kerja secara keseluruhan sistem *smart energy meter* yaitu mulai dari sensor yang terhubung dengan sumber tegangan dari beban yang dideteksi dayanya dan dapat dibaca pada monitor yang terhubung dengan Raspberry Pi 4B dengan menggunakan program dari python agar dapat mengakses sensor dan membaca datanya. Setelah itu mengirimkan data daya tersebut ke *node* 1 dengan menggunakan protokol komunikasi MQTT yaitu data daya tersebut disimpan pada *database blockchain* dan ditampilkan kembali di web pengguna.

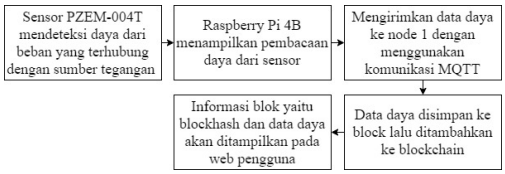


Gambar 4. Diagram Alir Pembuatan Perangkat dan Sistem *Blockchain*



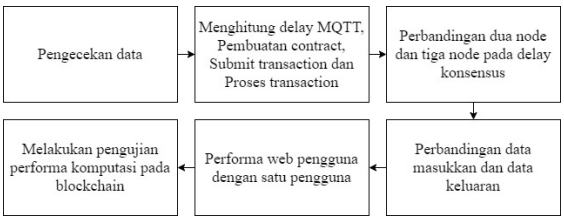
Gambar 5. Diagram Blok Proses Kerja Sistem *Smart Energy Meter*

Gambar 6 menunjukkan alur data dari sensor PZEM-004T hingga ke web pengguna yang sebelumnya melalui proses pengolahan data pada sistem *blockchain.* Data yang diterima oleh *node* 1 disimpan ke *database* dengan menggunakan fungsi setData yang dipanggil dengan web3.py API. Setiap data yang disimpan ke *database blockchain*, *node* 1 mengajukan transaksi dan menghasilkan blok. Data diambil dengan fungsi getDaya yang dipanggil dengan web3.py API. Data daya yang diambil tersebut ditampilkan ke web pengguna.



Gambar 6. Diagram Blok Proses *Blockchain*

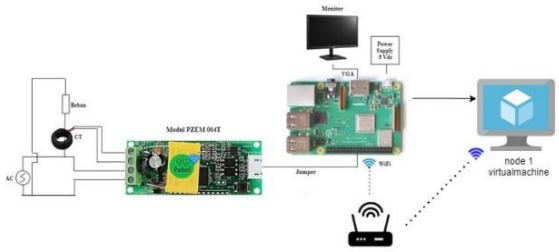
Selanjutnya Gambar 7 menunjukkan diagram blok pengujian performa sistem *smart energy meter*. Pengujian performa dilakukan dengan pengecekan antara data yang dikirim dari Raspberry Pi ke *node* 1 dengan yang ditampilkan pada web pengguna, menghitung *delay* MQTT, *delay* pembuatan *contract*, *delay submit transaction*, *delay* proses transaksi, perbandingan *delay* konsensus antara dua *node* dan tiga *node* dan performa komputasi *blockchain.*



Gambar 7. Diagram Blok Pengujian Performa *blockchain*

D. Desain Perangkat Smart Energy Meter

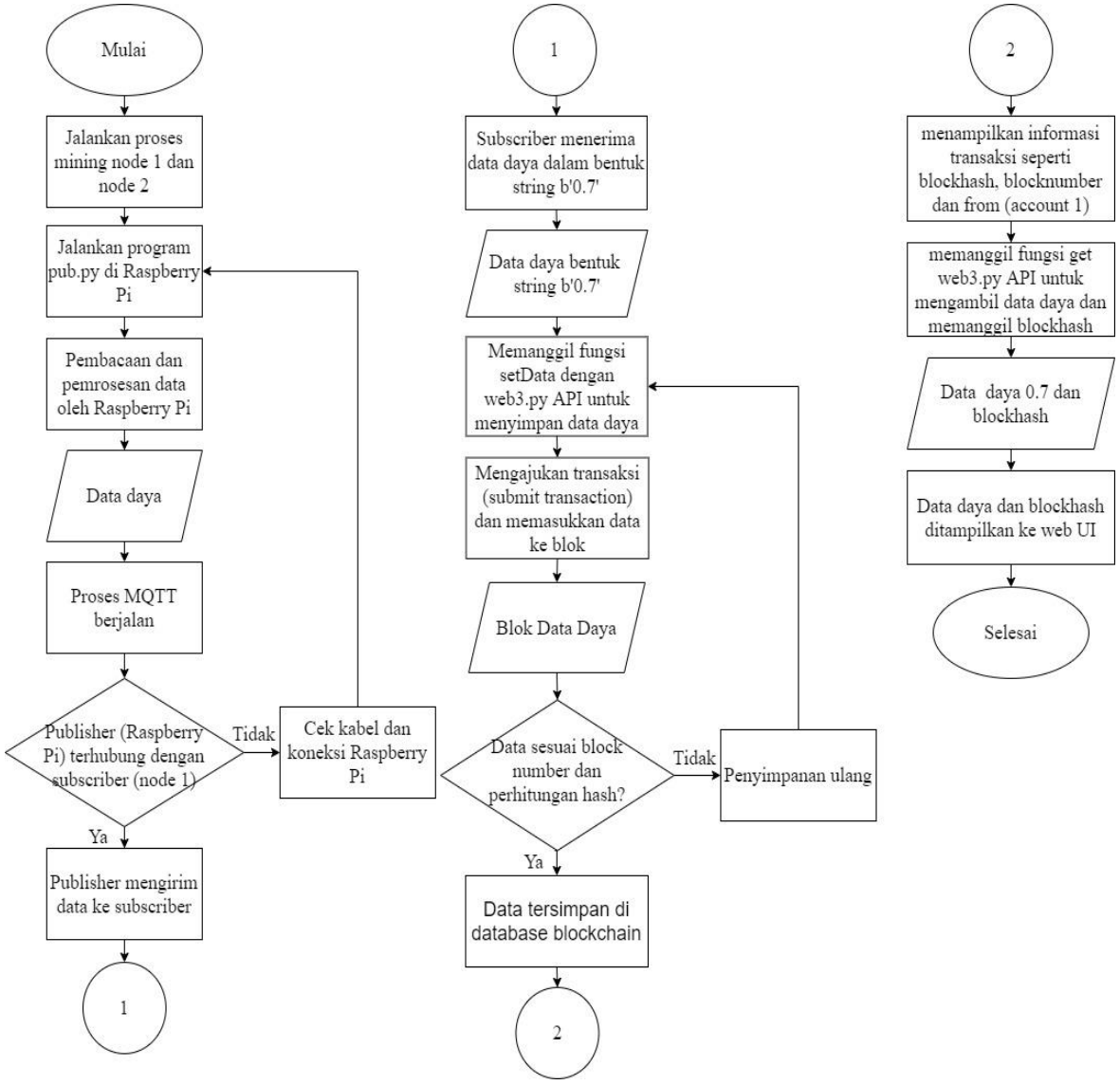
Gambar 8 menunjukkan rancangan desain perangkat *smart energy* meter yang terdiri dari sensor PZEM-004T, Raspberry Pi 4B dan Monitor. Pada Raspberry Pi 4B sudah terintegrasi modul WiFi dan sudah terpasang sistem operasi raspberry debian agar dapat digunakan. Sehingga Raspberry Pi 4B dapat dihubungkan ke monitor yang nantinya penulis dapat membaca data dari sensor PZEM-004T. Pembacaan data sensor tersebut diprogram menggunakan bahasa python yang sudah terintegrasi di raspberry debian.



Gambar 8. Desain Perangkat *Smart Energy Meter*

E. Desain Perangkat Lunak

Gambar 9 menunjukkan diagram alir sistem *blockchain* yang dimulai dari proses *mining node* 1 dan *node* 2 agar nantinya dapat melakukan transaksi. Lalu data dikirimkan dari *publisher* Raspberry Pi ke *subscriber node* 1 ubuntu virtualbox menggunakan MQTT di mana alamat IP yang diatur sebagai broker yaitu 192.168.1.14 (merupakan IP *address* dari Raspberry Pi). Setelah *node* 1 menerima data selanjutnya proses penyimpanan dengan memanggil fungsi set menggunakan web3.py API yang mengakses fungsi-fungsi setData dari *smart contract* yang sudah dibuat lalu *node* 1 mengajukan transaksi (*submit transaction*) dan menghasilkan blok. Dalam proses penyimpanan blok ke *blockchain* terdapat



Gambar 9. Diagram Alir Sistem *Blockchain*

validator (*account* 1) yang memvalidasi penyimpanan tersebut, jika sesuai dengan *block number* dan hasil perhitungan *hash* selanjutnya maka berhasil disimpan dan *account* 1 menyetujui transaksi dan menyimpan blok ke *blockchain*. Setiap proses penyimpanan dilakukan transaksi antara *account* 1 dan *account* lainnya. Transaksi tersebut menghasilkan beberapa informasi yang penting dan sangat digunakan pada penelitian penulis adalah *blockhash*, *blocknumber*, *from* atau pengirim dalam transaksi tersebut dan *hash* transaksi. Setelah data berhasil disimpan pada *database blockchain* maka selanjutnya proses pengambilan data daya dan *blockhash* dengan memanggil fungsi getDaya menggunakan web3.py API untuk mengaksesnya. Untuk memanggil dan membaca data daya dan *blockhash* tersebut ke Web UI.

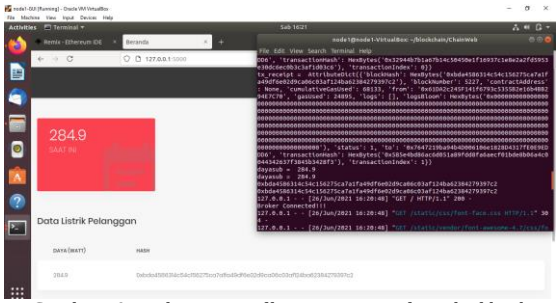
IV. HASIL DAN PEMBAHASAN

Pada pengujian ini, langkah pertama menjalankan sistem yaitu dengan menjalankan program flask sekaligus web3. Lalu setelah flask sudah berjalan, maka mengakses URL web flask yaitu http://127.0.0.1:5000/. Selanjutnya menjalankan program di sisi Raspberry Pi untuk menghubungkan Raspberry Pi dengan *node* 1 melalui protokol MQTT. Berikut skenario pengujiannya:

A. Pengecekan Data

Pengecekan data dilakukan antara data yang diterima dan disimpan dengan data yang ditampilkan pada web pengguna. Data tersebut adalah daya dan *blockhash*. Hasil dari pengecekan data yang diterima dan data yang ditampilkan sama, sehingga data berhasil disimpan dan ditampilkan pada web

pengguna. Hasil tersebut dapat dilihat pada Gambar 10.



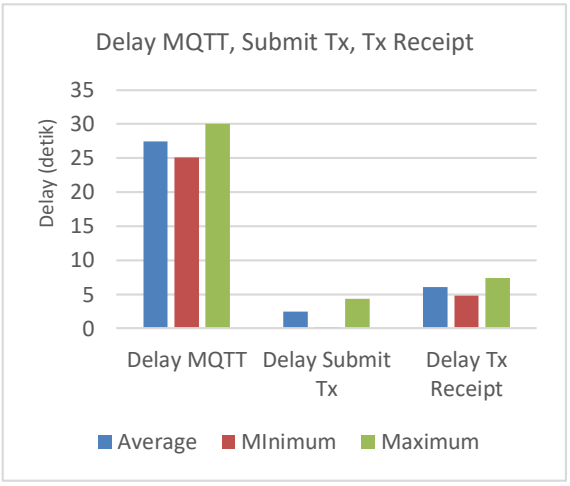
Gambar 10. Web Menampilkan Data Daya dan *Blockhash*

B. Pengukuran Delay Proses MQTT, Submit Transaction dan Transaction Resecipt

Pada bagian ini, penulis melakukan pengujian *delay* terhadap tiga proses sub-sistem seperti yang dijelaskan berikut ini.

1. Proses MQTT
   Pengukuran *delay* MQTT menggunakan wireshark untuk mengambil waktu pemrosesan MQTT. Proses MQTT dapat dilihat pada wireshark yaitu dimulai saat *node* 1 dan Raspberry Pi terhubung, subscriber (*node* 1) melakukan *request* hingga *publisher* (Raspberry Pi) mengirimkan data.
2. Proses *Submit Transaction*
   Pengukuran *delay submit transaction* dilakukan pada saat data daya diterima hingga keluar *submit transaction*. Pengukuran tersebut dengan menggunakan *library time* dari python. Penulis menambahkan baris *code* start_time sebelum potongan program data daya diterima dan end_time setelah potongan program pemanggilan fungsi setData.
3. Proses *Transaction Receipt*
   Pada proses *transaction receipt*, *node validator* melakukan validasi, jika berhasil maka membentuk tanda transaksi diterima. Pengukuran *delay* menampilkan data sama seperti pengukuran *delay submit transaction* yaitu dengan menggunakan *library time* dari python di mana menambahkan baris *code* start_time sebelum potongan program menampilkan tx_receipt dan end_time setelah potongan program untuk menampilkan *transaction receipt*.

Pengukuran *delay* pada protokol komunikasi MQTT didapatkan *delay* maksimal sebesar 30,07 detik. Hasil tersebut lebih besar 0,07 daripada interval waktu yang ditentukan oleh penulis pada *source code* python untuk *publish* data yaitu 30 detik. Namun hal tersebut tidak mempengaruhi hasil penerimaan data karena data sukses ditampilkan pada web pengguna. Hasil dari pengukuran *delay* pada proses *submit transaction* menunjukkan nilai maksimal yaitu 4,369 detik. Hasil pengukuran *delay* yang selanjutnya adalah pada proses *transaction receipt*. Hasil pengukuran tersebut menunjukkan nilai maksimal 7,431 detik. Jika dilakukan perhitungan total dari proses *submit transaction* hingga transaksi diterima yaitu 11,8 detik, sehingga pengaturan waktu interval pada program python MQTT di Raspberry Pi cukup baik yaitu mengirimkan setiap 30 detik, di mana data sukses dikirim dan disimpan. Namun jika dibandingkan dengan proses penyimpanan pada *database* tradisional yang bukan *blockchain* maka *delay* proses penyimpanan ke *database blockchain* lebih besar[20] dan tidak bisa digunakan untuk data *realtime*, hanya dapat digunakan untuk penyimpanan data dengan waktu proses sekitar 30 detik. Hasil pengukuran *delay* MQTT, *submit transaction (Submit Tx)* dan transaksi yang diterima (Tx *Receipt*) dapa dilihat pada Gambar 11.



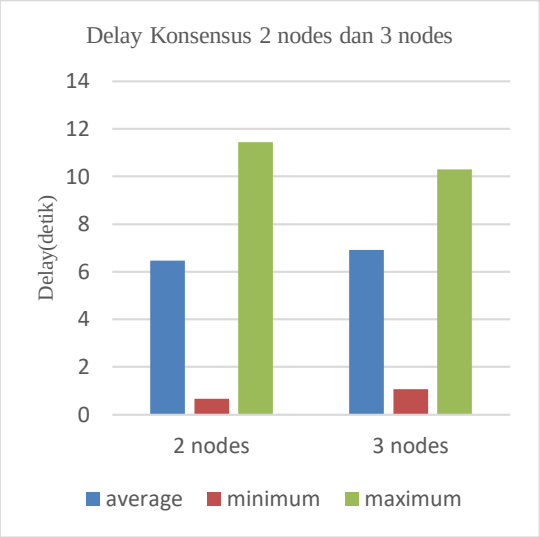
Gambar 11. Nilai Minimal, Rata-rata dan Maksimal dari Pengukuran *Delay*

C. Proses Konsensus pada Dua Node dan Tiga Node

Pengukuran dilakukan dengan memberi baris program start_time sebelum baris program pemanggilan fungsi set dan end time sebelum baris program menampilkan tx_receipt. Pengukuran *delay* konsensus ada dua pengukuran yaitu pada dua dan tiga *node*.

Pada pengukuran yang pertama dua *node* memiliki rata-rata *delay* konsensus lebih tinggi yaitu 3,154 detik dan pada tiga *node* rata-rata *delay* yaitu 2,349 detik. Sedangkan pada pengukuran yang kedua, tiga *node* memiliki rata-rata *delay* yang lebih tinggi yaitu 6,916 detik dan dua *node* memiliki rata-rata *delay* 6,468 detik. Hasil tersebut menunjukkan bahwa jumlah *node* tidak berpengaruh pada waktu konsensus. Dikarenakan menurut dari beberapa

referensi yaitu jumlah *node* tidak terlalu mempengaruhi kinerja konsensus untuk beberapa konsensus seperti PoW dan PoA[21]. Yang mempengaruhi proses validasi adalah nilai komputasi atau CPU dan kualitas jaringan pada masing-masing *node*[22]. Berikut hasil pengukuran pada dua *node* dan tiga *node* pada Gambar 12.



Gambar 12. Pengukuran *Delay* Konsensus

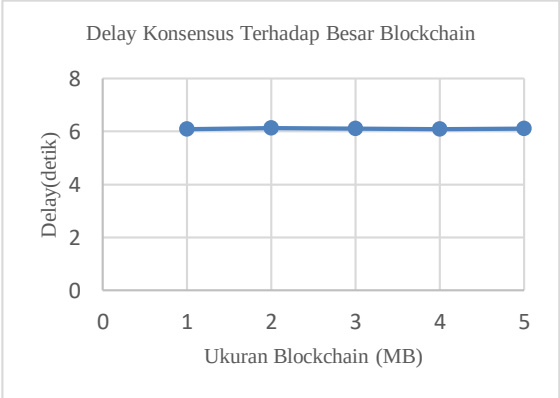
D. *Proses Pembuatan Smart Contract*

Pembuatan *smart contract* ini terjadi pada awal program dijalankan. Pengukuran *delay* pembuatan *smart contract* dilakukan dengan cara memantau pada geth *console node* 1, ketika memulai program web3 maka membuat *smart contract.* Pembuatan *smart contract* terjadi dua kali yang pertama dengan *delay* 0,001 detik dan yang kedua dengan *delay* 1,051 detik. Jika dijumlahkan total *delay* pembuatannya adalah sekitar 1 detik. Hasil tersebut tidak berpengaruh pada proses sistem karena pembuatan *smart contract* hanya pada awal proses saja.

E. *Perbandingan Panjang Blockchain Terhadap Delay Consensus*

Pengukuran dilakukan dengan memberi baris program start_time sebelum baris program pemanggilan fungsi set dan end time setelah baris program menampilkan *blockhash*. Ukuran blok pada *blockchain* didapat dari memanggil API web3 eth.getBlock(‘latest’). Pengukuran dilakukan pada ukuran data 36 byte dan ukruan blok yang tersimpan data yaitu 818 byte, sedangkan blok yang kosong 608 byte. Sekitar 17,5 jam di mana data akan dikirimkan setiap 15 detik. Total panjang *blockchain* yang dihasilkan selama pengukuran adalah 5,987MB.

Pada hasil pengukuran *delay* konsensus dilakukan perhitungan rata-rata setiap 1 MB. Rata-rata *delay* konsensus yang dihasilkan pada ukuran panjang *blockchain* pada 1 MB sebesar 6,085 detik, pada ukuran panjang *blockchain* 2 MB sebesar 6,119 detik, pada ukuran panjang *blockchain* 3 MB sebesar 6,105 detik, pada ukuran panjang *blockchain* 4 MB sebesar 6,087 detik dan pada ukuran panjang *blockchain* 5 MB sebesar 6,11 detik. Berdasarkan hasil tersebut panjang *blockchain* tidak mempengaruhi *delay* konsensus karena grafik rata-rata *delay* hampir memiliki besar *delay* yang sama yaitu sekitar 6 detik. Selain itu, mengapa panjang *blockchain* tidak mempengaruhi *delay* konsensus? karena proses validasi dilakukan pada blok yang akan disimpan dengan satu blok sebelumnya, Berikut hasil pengukuran dapat dilihat pada Gambar 13.



Gambar 13. Rata-rata *Delay* Konsensus Terhadap Panjang *Blockchain*

F. *Perbandingan Ukuran Data Input dan Blok Output*

Penulis menggunakan data *random* atau *dummy* untuk mendapat ukuran 500 *byte*, 400 *byte* dan 300 *byte*. Ukuran data masukkan didapat dari memasukkan *library* sys python untuk menggunakan sys.getsizeof() agar dapat menampilkan informasi ukuran data yang ingin ditampilkan. Informasi ukuran blok didapatkan dari API web3 yaitu eth.getBlock(‘latest’) di mana akan menampilkan informasi blok yang terakhir melakukan transaksi. Berikut informasi masing-masing ukuran data kirim dan ukuran bloknya pada Tabel 1.

TABEL I. INFORMASI UKURAN DATA KIRIM, DATA TERIMA DAN UKURAN BLOK

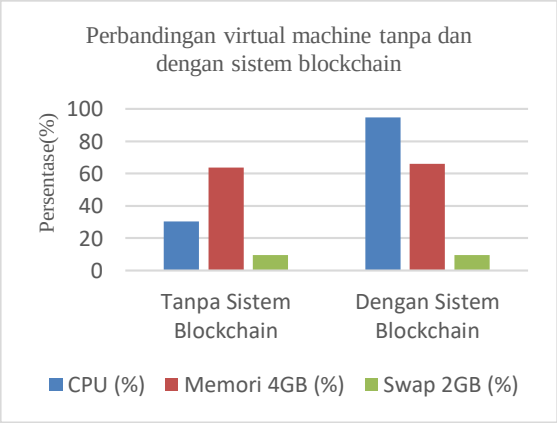
Ukuran Data	**Ukuran Blok**
300 *byte*	1461 *byte*
400 *byte*	1685 *byte*
500 *byte*	1910 *byte*

Penulis melakukan pengambilan data sebanyak 174 blok di mana 144 blok kosong dan 30 terisi data

masukkan. Informasi masing-masing ukuran data dan ukuran blok dapat dilihat pada tabel 3.1 yaitu pada ukuran 500 *byte* memiliki ukuran blok 1910 *byte* maka untuk informasi *blockhash*, *transaction hash* dan informasi blok lainnya menghabiskan 1410 *byte*. Pada ukuran 400 *byte* memiliki ukuran blok 1685 *byte* maka untuk informasi blok lainnya menghabiskan 1285 *byte*. Dan pada ukuran 300 *byte* memiliki ukuran blok 1461 *byte* sehingga untuk informasi *block* selain data menghabiskan 1161 *byte*. Terdapat blok kosong yang tidak menyimpan data yaitu memiliki ukuran blok 608 *byte*.

G. Pengukuran Komputasi Sistem Blockchain

Menurut beberapa referensi jumlah *node* tidak terlalu mempengaruhi biaya komputasi proses konsensus pada skema PoW dan PoA[21]. Yang mempengaruhi proses validasi adalah nilai komputasi atau CPU dan kualitas jairngan pada masing-masing *node*[22]. Maka penulis melakukan pengujian performa komputasi pada *blockchain* dilakukan dengan membandingkan nilai proses CPU tanpa *mining blockchain* dengan proses pengiriman data dengan *mining blockchain.* Berdasarkan hasil performa dengan memantau monitor sistem komputasi pada ubuntu, maka hasilnya sistem operasi lebih berat dengan menjalankan *mining blockhain* karena *mining* tersebut digunakan juga untuk menyimpan data daya dari sistem IoT dan juga untuk mendapatkan data daya dan *blockhash* tersebut untuk ditampilkan pada web pengguna. Pemakaian CPU mencapai 100% di mana itu nilai pemakaian tertinggi. Pemakaian *memory* mencapai lebih dari setengah kinerja *memory* yaitu 68,8%. Untuk pemakaian partisi swap tidak digunakan karena nilai sama antara sistem operasi tanpa dan dengan sistem *blockchain* yaitu 9,7%. Maka sistem *blockchain* harus menggunakan spesifikasi CPU, *Harddisk* dan *Memory* yang cukup tinggi. Gambar 16 merupakan grafik rata-rata nilai persentase CPU, *memory* dan partisis swap sebelum dan saat sistem *blockchain* berjalan.



Gambar 16. Grafik Rata-rata Nilai CPU, *Memory* dan partisi Swap Sebelum Menjalankan Sistem *Blockchain*

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Penelitian ini mengusulkan sistem *blockchain* sebagai *database* terdistribusi pada *smart energy meter* untuk meningkatkan kehandalan layanan *database.* Sistem *blockchain* ini diimplementasikan pada jaringan privat dengan menggunakan *framework* Ethereum. Hasil eksperimen menunjukkan bahwa komponen *delay* terbesar dalam sistem adalah pengiriman data dari Raspberry Pi ke *node* 1 yang menggunakan protokol MQTT. Pada eksperimen pembuatan *blockchain*, *delay* konsensus terpantau stabil terhadap ukuran *blockchain.* Namun, pada pengukuran performa komputasi, sumber daya komputasi jelas sangat terpengaruh dengan penggunaan *blockchain* yang mengakibatkan bebannya naik cukup *signifikan* mendekati kapasitas 100% dan 68,8% untuk CPU dan RAM pada *node* yang melakukan konsensus.

B. Saran

Agar lebih informatif untuk pengguna, maka sistem *smart contract* pada *blockchain* Ethereum perlu ditambah fitur untuk memodifikasi atau menambah parameter listrik lainnya. Server web-hosting juga perlu ditambahkan ke dalam *smart energy meter* agar dapat diakses oleh banyak pengguna di internet.

REFERENSI

[1] M. Moniruzzaman, S. Khezr, A. Yassine, and R. Benlamri, "Blockchain for smart homes: Review of current trends and research challenges," *Comput. Electr. Eng.*, vol. 83, p. 106585, 2020, doi: 10.1016/j.compeleceng.2020.106585.

[2] J. E. Siegel, S. Kumar, and S. E. Sarma, "The future internet of things: Secure, efficient, and model-based," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2386–2398, 2018, doi: 10.1109/JIOT.2017.2755620.

[3] N. A. Prasetyo, A. G. Prabawati, and Suyoto, "Smart home: Power electric monitoring and control in Indonesia," *Int. J. Interact. Mob. Technol.*, vol. 13, no. 3, pp. 143–151, 2019, doi: 10.3991/ijim.v13i03.10070.

[4] M. B. Mollah *et al.*, "Blockchain for Future Smart Grid: A Comprehensive Survey," *IEEE Internet Things J.*, vol. X, no. vi, pp. 1–1, 2020, doi: 10.1109/JIOT.2020.2993601.

[5] F. Dewanta and M. Mambo, "BPT Scheme : Establishing Trusted Vehicular Fog Computing Service for Rural Area Based on Blockchain Approach," vol. 70, no. 2, pp. 1752–1769, 2021, doi: 10.1109/TVT.2021.3051258.

[6] A. I. I. MuhammadRakha Laayu, Rendy Munadi, "Analisis Algoritma Advanced Encryption Standard (AES) Untuk Sistem Pemantauan Konsumsi Daya Listrik," *e-Proceeding Eng.*, vol. 7, no. 3, pp. 8827–8833, 2020, [Online]. Available: https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/download/13956/13696.

[7] Y. Ren *et al.*, "Multiple cloud storage mechanism based on blockchain in smart homes," *Futur. Gener. Comput. Syst.*, vol. 115, pp. 304–313, 2021, doi: 10.1016/j.future.2020.09.019.

[8] G. S. Ramachandran *et al.*, "Trinity: A Byzantine Fault-Tolerant Distributed Publish-Subscribe System with Immutable Blockchain-based Persistence," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, May 2019, pp. 227–235, doi: 10.1109/BLOC.2019.8751388.

[9] R. A. Atmoko, R. Riantini, and M. K. Hasin, "IoT real time data acquisition using MQTT protocol," *J. Phys. Conf. Ser.*, vol. 853, no. 1, 2017, doi: 10.1088/1742-6596/853/1/012003.

[10] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," *2017 IEEE Int. Conf. Pervasive Comput. Commun. Work. PerCom Work. 2017*, pp. 618–623, 2017, doi: 10.1109/PERCOMW.2017.7917634.

[11] L. Arief and T. A. Sundara, "Studi atas Pemanfaatan Blockchain bagi Internet of Things (IoT)," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 1, no. 1, p. 70, 2017, doi: 10.29207/resti.v1i1.26.

[12] D. A. Badawi, "Sistem Verifikasi Dokumen Hasil Investigasi Digital Berbasis Teknologi Blockchain." 2019.

[13] S. Ferretti and G. D'Angelo, "On the Ethereum blockchain structure: A complex networks theory perspective," *Concurr. Comput.* , vol. 32, no. 12, 2020, doi: 10.1002/cpe.5493.

[14] P. Wackerow, "Intro to Ethereum," 2021. https://ethereum.org/en/developers/docs/intro-to-ethereum/.

[15] P. Sajana, "On Blockchain Applications : Hyperledger Fabric And Ethereum," vol. 118, no. 18, pp. 2965–2970, 2018.

[16] S. Al-Saqqa and S. Almajali, "Blockchain technology consensus algorithms and applications: A survey," *Int. J. Interact. Mob. Technol.*, vol. 14, no. 15, pp. 142–156, 2020, doi: 10.3991/IJIM.V14I15.15893.

[17] J. C. Piper Merriam, "Introduction Web3.py," 2018. https://web3py.readthedocs.io/en/stable/.

[18] T. go-ethereum Authors, "Go Ethereum," 2021. https://geth.ethereum.org/.

[19] P. Wackerow, "Nodes and clients," 2021. https://ethereum.org/en/developers/docs/nodes-and-clients/.

[20] M. J. M. Chowdhury, A. Colman, M. A. Kabir, J. Han, and P. Sarda, "Blockchain Versus Database: A Critical Analysis," *Proc. - 17th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. 12th IEEE Int. Conf. Big Data Sci. Eng. Trust. 2018*, no. August, pp. 1348–1353, 2018, doi: 10.1109/TrustCom/BigDataSE.2018.00186.

[21] Y. Wu, P. Song, and F. Wang, "Hybrid Consensus Algorithm Optimization: A Mathematical Method Based on POS and PBFT and Its Application in Blockchain," *Math. Probl. Eng.*, vol. 2020, 2020, doi: 10.1155/2020/7270624.

[22] B. Cao *et al.*, "Performance analysis and comparison of PoW, PoS and DAG based blockchains," *Digit. Commun. Networks*, vol. 6, no. 4, pp. 480–485, 2020, doi: 10.1016/j.dcan.2019.12.001.