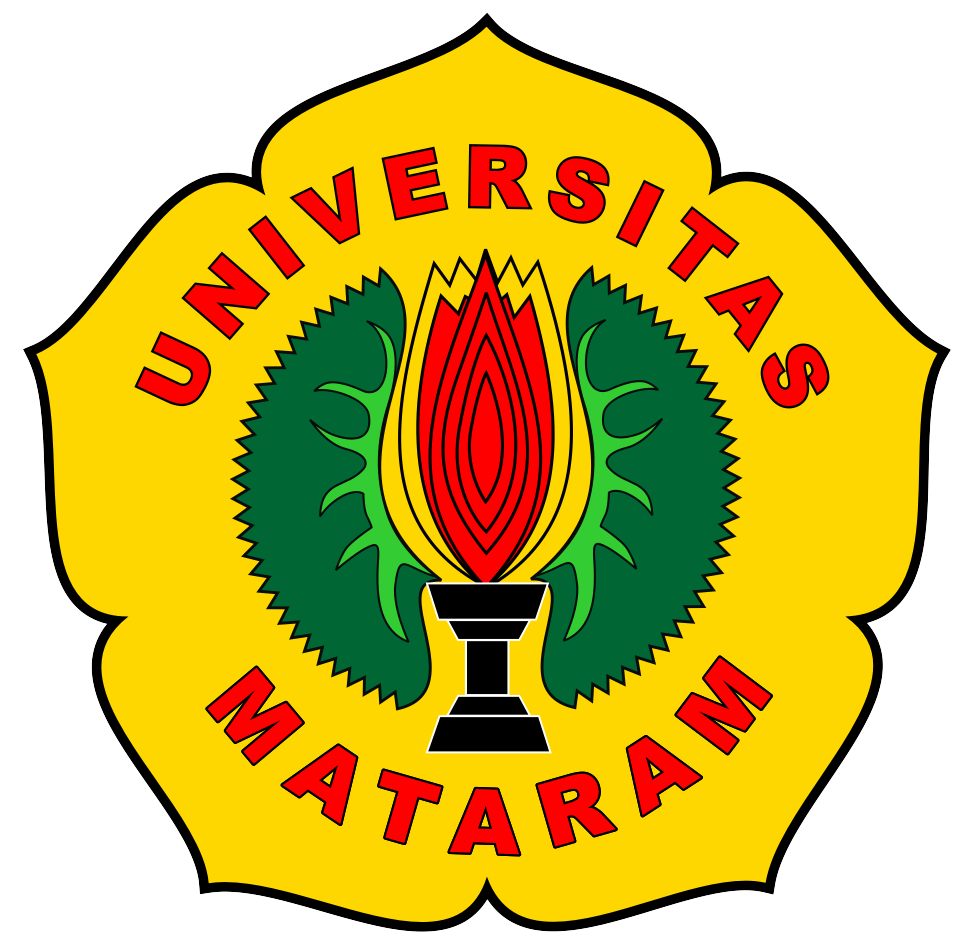
# MAKALAH SEMINAR HASIL TUGAS AKHIR

**IMPLEMENTASI *BLOCKCHAIN* SEBAGAI SISTEM KEAMANAN IoT DALAM MENDISTRIBUSIKAN DAN MENJAGA PRIVASI DATA**



**Oleh :   
Lalu Ocky Saktiya Luhung  
F1B 019 079**

**JURUSAN TEKNIK ELEKTRO**

**FAKULTAS TEKNIK**

**UNIVERSITAS MATARAM**

**2024**

# Makalah Seminar Hasil Tugas Akhir

**IMPLEMENTASI *BLOCKCHAIN* SEBAGAI SISTEM KEAMANAN IoT DALAM MENDISTRIBUSIKAN DAN MENJAGA PRIVASI DATA**

**Oleh:**

**Lalu Ocky Saktiya Luhung**

**F1B 019 079**

Telah diperiksa dan disetujui oleh:

1. Pembimbing Utama

Dr. Ir. Misbahuddin, S.T., M.T., IPU. Tanggal:

NIP: 19710607 199702 1 002

1. Pembimbing Pendamping

Giri Wahyu Wiriasto, S.T., M.T. Tanggal:

NIP: 19820904 201012 1 001

Mengetahui

Ketua Jurusan/Prodi Teknik …

Fakultas Teknik

Universitas Mataram

A. Sjamsjiar Rachman, ST., MT

NIP : 19711124 199903 1 004

# DAFTAR ISI

[MAKALAH SEMINAR HASIL TUGAS AKHIR i](#_Toc156203904)

[Makalah Seminar Hasil Tugas Akhir ii](#_Toc156203905)

[Fakultas Teknik ii](#_Toc156203906)

[Universitas Mataram ii](#_Toc156203907)

[DAFTAR ISI iii](#_Toc156203908)

[DAFTAR TABEL vi](#_Toc156203909)

[DAFTAR GAMBAR vii](#_Toc156203910)

[ABSTRAK ix](#_Toc156203911)

[*ABSTRACT* x](#_Toc156203912)

[BAB I 1](#_Toc156203913)

[PENDAHULUAN 1](#_Toc156203914)

[1.1. Latar Belakang 1](#_Toc156203915)

[1.2. Rumusan Masalah 3](#_Toc156203916)

[1.3. Batasan Masalah 3](#_Toc156203917)

[1.4. Tujuan Penelitian 3](#_Toc156203918)

[1.5. Manfaat Penelitian 3](#_Toc156203919)

[1.6. Sistematika Penulisan 4](#_Toc156203920)

[BAB II 6](#_Toc156203921)

[TINJAUAN PUSTAKA DAN DASAR TEORI 6](#_Toc156203922)

[2.1. Tinjauan Pustaka 6](#_Toc156203923)

[2.2. Dasar Teori 7](#_Toc156203924)

[2.2.1 *Internet of Things* (IoT) 7](#_Toc156203925)

[2.2.2 *Blockchain* 8](#_Toc156203926)

[2.2.3 *Blockchain*-IoT 9](#_Toc156203927)

[2.2.4 Algoritma *Consensus* 10](#_Toc156203928)

[2.2.5 Kriptografi 11](#_Toc156203929)

[2.2.6 Raspberry Pi 12](#_Toc156203930)

[2.2.7 MultiChain 12](#_Toc156203931)

[2.2.8 *Socket Programming* 13](#_Toc156203932)

[BAB III 14](#_Toc156203933)

[METODE PENELITIAN 14](#_Toc156203934)

[3.1. Tahapan Penelilitian 14](#_Toc156203935)

[3.2. Studi Literatur 15](#_Toc156203936)

[3.3. Analisis Kebutuhan 15](#_Toc156203937)

[3.4. Perancangan Topologi 15](#_Toc156203938)

[3.4.1 Topologi Jaringan 15](#_Toc156203939)

[3.4.2 Alur Komunikasi *Blockchain* 17](#_Toc156203940)

[3.4.3 Perangkat *Smart Home* 17](#_Toc156203941)

[3.4.4 *Node admin* 18](#_Toc156203942)

[3.4.5 Pengiriman Data 19](#_Toc156203943)

[3.4.6 Penerimaan Data 19](#_Toc156203944)

[3.5. Pengujian 21](#_Toc156203945)

[3.5.1 Koneksi dengan MultiChain 21](#_Toc156203946)

[3.5.2 Penyimpanan Data 21](#_Toc156203947)

[3.5.3 Validasi Data 21](#_Toc156203948)

[BAB IV 23](#_Toc156203949)

[HASIL DAN PEMBAHASAN 23](#_Toc156203950)

[4.1. Implementasi Multichain 23](#_Toc156203951)

[4.1.1. Proses *Create Blockchain* 23](#_Toc156203952)

[4.1.2. Proses *Run Blockchain* 24](#_Toc156203953)

[4.1.3. Proses Penelusuran *Blockchain* 24](#_Toc156203954)

[4.1.4. Proses Mencari *Permissions* 27](#_Toc156203955)

[4.2. Koneksi *Node Client* dengan *Node* Admin 28](#_Toc156203956)

[4.2.1. Proses Koneksi oleh *Node Client* 28](#_Toc156203957)

[4.2.2. Proses *Permissions* oleh *Node admin* 29](#_Toc156203958)

[4.2.3. Proses *Run* pada *Node Client* 30](#_Toc156203959)

[4.2.4. Proses Penelusuran pada *Node Client* 31](#_Toc156203960)

[4.3. Membuat *Stream* 37](#_Toc156203961)

[*4.3.1.* *Create Stream* 37](#_Toc156203962)

[4.3.2. Mencari *List Stream* 37](#_Toc156203963)

[4.3.3. Menyimpan Data pada *Stream* 38](#_Toc156203964)

[4.3.4. Melihat Detail Data pada *Stream* 38](#_Toc156203965)

[4.4. Penyimpanan Data *Node Client* pada *Stream* 39](#_Toc156203966)

[4.4.1. *Node client* Mencari Detail Item pada *Stream* 39](#_Toc156203967)

[4.4.2. *Node client* Gagal Menyimpan Data pada Stream 40](#_Toc156203968)

[4.4.3. *Permission* oleh *Node admin* 40](#_Toc156203969)

[4.4.4. *Node Client* Berhasil Meyimpan Data pada *Stream* 41](#_Toc156203970)

[4.4.5. Pencarian Detail Item pada *Stream* 41](#_Toc156203971)

[4.5. Program Perangkat *Smart Home* 42](#_Toc156203972)

[4.6. Program *Server* Untuk Terhubung dengan Multichain 44](#_Toc156203973)

[4.7. Pencarian Data pada Stream 46](#_Toc156203974)

[4.8. *Permissionless Device* 47](#_Toc156203975)

[4.8.1. *Permissionless Device* Mencoba Terhubung 47](#_Toc156203976)

[4.8.2. *Permissionless Device* Menyimpan Data 47](#_Toc156203977)

[4.8.3. *Permissionless Device* Mencari Detail Items 47](#_Toc156203978)

[BAB V 49](#_Toc156203979)

[KESIMPULAN DAN SARAN 49](#_Toc156203980)

[5.1. Kesimpulan 49](#_Toc156203981)

[5.2. Saran 49](#_Toc156203982)

[DAFTAR PUSTAKA 50](#_Toc156203983)

# DAFTAR TABEL

[Tabel 4.1 Info Blockchain yang Dapat diakses 26](#_Toc156167843)

[Tabel 4.2 Parameter dalam Struktur Block pada Node Admin 27](#_Toc156167844)

[Tabel 4.3 Alamat dan Jenis Node dalam Blockchain 28](#_Toc156167845)

[Tabel 4.4 Mekanisme handshake 29](#_Toc156167846)

[Tabel 4.5 Perintah Koneksi dengan Node 30](#_Toc156167847)

[Tabel 4.6 Parameter dalam Struktur Block pada Node Client 32](#_Toc156167848)

[Tabel 4.7 Keterangan mekanisme handshake 36](#_Toc156167849)

# DAFTAR GAMBAR

[Gambar 2.1 *Blockchain* dan Database 8](#_Toc150806172)

[Gambar 2.2 Struktur *Blockchain* 9](#_Toc150806173)

[Gambar 3.1 Diagram Alir Tahapan Penelitian 14](#_Toc150806175)

[Gambar 3.2 Topologi Jaringan 16](#_Toc150806176)

[Gambar 3.3 Alur Komunikasi *Blockchain* 17](#_Toc150806177)

[Gambar 3.4 Diagram Alir Program Simulasi Smart Home 18](#_Toc150806178)

[Gambar 3.5 Alur Penerimaan Data 20](#_Toc150806179)

[Gambar 4.1 Create Private blockchain 23](#_Toc150928900)

[Gambar 4.2 Running Blockchain 24](#_Toc150928901)

[Gambar 4.3 Mendapatkan Info Blockchain 25](#_Toc150928902)

[Gambar 4.4 Mendapatkan Detail Block 26](#_Toc150928903)

[Gambar 4.5 Mencari List Permission dalam Blockchain 27](#_Toc150928904)

[Gambar 4. 6 Node client Mencoba Terhubung dengan Jaringan Private blockchain 28](#_Toc150928905)

[Gambar 4.7 Pemberian Izin untuk Terhubung oleh Node admin 29](#_Toc150928906)

[Gambar 4.8 Node client Berhasil Terhubung 30](#_Toc150928907)

[Gambar 4.9 Mendapatkan Detail Blocks pada Node Client 31](#_Toc150928908)

[Gambar 4.10 Peer info pada Node admin 33](#_Toc150928909)

[Gambar 4.11 Peer Info pada Node Client 35](#_Toc150928910)

[Gambar 4.12 Create Stream 37](#_Toc150928911)

[Gambar 4.13 Mencari List Stream 37](#_Toc150928912)

[Gambar 4.14 Publish Data pada Stream 38](#_Toc150928913)

[Gambar 4.15 Gagal Melihat Detail Items 38](#_Toc150928914)

[Gambar 4.16 Proses Subscribe Stream 38](#_Toc150928915)

[Gambar 4.17 Detail Items dalam Stream 39](#_Toc150928916)

[Gambar 4.18 Node client Gagal Mendapat Detail Items 39](#_Toc150928917)

[Gambar 4.19 Node client Gagal Menyimpan Data 40](#_Toc150928918)

[Gambar 4.20 Pemberian Izin oleh Node admin 40](#_Toc150928919)

[Gambar 4.21 Node client Berhasil Meyimpan Data 41](#_Toc150928920)

[Gambar 4.22 Proses Subscribe dan Mendapatkan Detail Items 41](#_Toc150928921)

[Gambar 4.23 Hasil Simulasi Program Smart Home 43](#_Toc150928922)

[Gambar 4. 24 Menerima dan Publish Data oleh Server 45](#_Toc150928923)

[Gambar 4.25 Details Items dari Data Smart Home 46](#_Toc150928924)

[Gambar 4.26 Permissionless Device Mencoba Terhubung 47](#_Toc150928925)

[Gambar 4.27 Permissionless Device Gagal Menyimpan Data 47](#_Toc150928926)

[Gambar 4.28 Permissionless Device Gagal Mencari Detail Item 47](#_Toc150928927)

# ABSTRAK

Integrasi teknologi *blockchain* dengan *Internet of Things* (IoT) untuk meningkatkan keamanan dan privasi data. Dengan fokus pada keamanan IoT pada saat mendistribusikan data , risiko manipulasi data dapat dikurangi karena *blockchain* memungkinkan struktur terdesentralisasi dan terenkripsi. Integritas data ditingkatkan melalui verifikasi transaksi dan penyimpanan data yang diotorisasi dengan *hash value* dan bukti transaksi (txid). Penggunaan jaringan *private blockchain* mengontrol akses data, meningkatkan keamanan, privasi, dan memastikan akses terbatas pada pihak yang diizinkan. Hasilnya menunjukkan bahwa *blockchain* dapat meningkatkan keamanan data IoT serta menjaga integritas data dengan lebih baik, memperkuat otorisasi dan mengontrol akses data secara lebih selektif.

**Kata Kunci** : *Blockchain*, *Internet of Things* (IoT), *Private blockchain*

# *ABSTRACT*

*Integration of blockchain technology with the Internet of Things (IoT) to improve data security and privacy. By focusing on IoT security when distributing data, the risk of data manipulation can be reduced because blockchain enables a decentralized and encrypted structure. Data integrity is enhanced through transaction verification and data storage authorized by hash value and proof of transaction (txid). The use of a private blockchain network controls data access, enhances security, privacy, and ensures access is limited to authorized parties. The results show that blockchain can improve IoT data security and better maintain data integrity, strengthen authorization and control data access more selectively.*

***Keyword*** : *Blockchain*, *Internet of Things* (IoT), *Private blockchain*

# BAB I

# PENDAHULUAN

## Latar Belakang

Dengan kemajuan teknologi komunikasi dan pengenalan jaringan 5G di mana-mana, teknologi *Internet of Things* mulai berkembang pada tingkat yang eksponensial. *Smart Home*, *Smart City*, *e-Health*, dan *Internet of Things* untuk perusahaan industri, intelijen terdistribusi, dan sistem lainnya adalah cara yang efektif dan akrab bagi masyarakat untuk meningkatkan banyak proses, misalnya, proses untuk monitoring keadaan rumah berdasarkan sensor dan proses lain yang dapat menjadi otomatis. Pendekatan proses seperti itu mengurangi pengaruh faktor manusia dan berkontribusi pada peningkatan efisiensi perusahaan, di mana ada semua prasyarat untuk penggunaan teknologi IoT. Terlepas dari semua efektivitas dan prevalensinya, teknologi *Internet of Things* memiliki banyak tantangan dan masalah yang terkait dengan keamanan dan konfigurasi perangkat IoT yang aman. Keberadaan sejumlah besar perangkat semacam itu penuh dengan bahaya, karena penyerang dapat mengendalikannya dan mengatur serangan DDoS dan manipulasi lalu lintas lainnya menggunakan perangkat IoT, yang mengirim perangkat ini ke *server*. Salah satu contoh serangan terpadu pada beberapa perangkat IoT adalah botnet. Botnet adalah kumpulan perangkat yang disusupi di bawah kendali penyerang. Mirai adalah worm dan botnet yang dibentuk oleh perangkat yang diretas (disusupi) seperti *Internet of Things* (pemutar video, webcam pintar, dll.). Botnet ini meretas perangkat dengan menebak kata sandi untuk port 23 (telnet). Dalam sistem IoT terpusat, terkadang cukup untuk meretas server atau mikrokontroler yang bertanggung jawab untuk komunikasi antara sekelompok besar perangkat agar dapat mengontrol semua perangkat yang berkomunikasi melalui protokol terpusat dengan *server* yang dikompromikan (Nesterenko dan Maslova, 2022).

Di lain sisi, perkembangan teknologi yang cukup maju memungkinkan untuk mengurangi dampak dari masalah yang ada, bahkan ada kemungkinan dapat menyelesaikan masalah yang ada. Teknologi yang dimaksud adalah *blockchain*.

Pendekatan penyimpanan yang terdesentralisasi untuk menyediakan penyimpanan data terdapat pada teknologi *blockchain* dan layanan berbagi. Untungnya, sifat teknologi *blockchain* dapat memberikan solusi yang baik untuk sistem penyimpanan yang terdesentralisasi. *Blockchain* terdiri dari blok-blok individual yang dihubungkan oleh fungsi hash, dan setiap blok berisi nilai hash dari blok sebelumnya, *time-stamp*, dan data transaksi. *Blockchain* dapat dianggap sebagai *database* buku besar terdistribusi, yang terdesentralisasi, terbuka dan transparan, anti-rusak, dan dapat dilacak, serta menyediakan metode penyimpanan yang aman dan andal untuk data.

Pada dasarnya *Blockchain* adalah buku besar basis data yang terdesenralisasi, terdistribusi saling berbagi dan sangat sulit untuk diubah yang menyimpan daftar aset dan transaksi di jaringan *peer-to-peer*, serta telah merantai blok data yang telah diberi cap waktu dan divalidasi oleh *miners*. Teknologi *blockchain* telah diramalkan oleh industri dan komunitas penelitian sebagai teknologi yang sangat menyita perhatian yang siap memainkan peran utama mengelola, mengendalikan, dan yang paling penting mengamankan perangkat IoT. *Blockchain* menggunakan algoritma *hashing* SHA-256 untuk memberikan bukti kriptografi yang kuat untuk otentikasi dan integritas data. *Blockchain* memiliki riwayat penuh dari semua transaksi dan memberikan kepercayaan terdistribusi global. Salah satu tujuan penggunaan *blockchain* adalah untuk menghilangkan pihak ketiga atau *Trusted Third Parties* (TTP). TTP atau otoritas dan layanan terpusat dapat diganggu, ditembus kemanannya, dan diretas. Mereka juga dapat berbuat jahat dan berprilaku korup di masa depan, meskipun mereka dapat dipercaya sekarang (Khan, dkk, 2018).

## Rumusan Masalah

Berdasarkan latar belakang diatas dapat diidentifikasi beberapa permasalahan yaitu :

1. Bagaimana cara menjaga keamanan data dalam sistem IoT dengan menggunakan *blockchain*?
2. Bagaimana *blockchain* dapat diimplemetasikan pada sistem keamanan IoT untuk meningkatkan keamanan dan privasi data pada saat mendistribusikan data?
3. Bagaimana *blockchain* dapat memfasilitasi proses autentikasi dan otorisasi pada sistem keamanan IoT?

## Batasan Masalah

Penelitian yang dilakukan memiliki batasan masalah dan ruang lingkup sebagai berikut:

1. Menjaga keamanan dan privasi data dalam sistem IoT berfokus dengan menggunakan jaringan *private blockchain*.
2. Perancangan dan pengembangan sistem keamanan IoT berbasis *blockchain*.
3. Analisis keamanan dan privasi data saat mendistribusikan data oleh perangkat IoT kepada *blockchain*.

## Tujuan Penelitian

Tujuan penelitian ini adalah mengimplementasi dan mengintegrasi teknologi IoT dengan teknologi *blockchain* dalam mendistribusikan data pada IoT dengan menggunakan *blockchain*, dan berbagai hal lainnya seperti berikut:

1. Menjaga keamanan dan privasi data pada sistem IoT dengan *blockchain*.
2. Meningkatkan integritas data, memberikan otentikasi dan otorisasi yang aman, memfasilitasi transaksi yang aman, serta meningkatkan privasi data.
3. Menggunakan jaringan *private blockchain* dalam memberi akses secara khusus.

## Manfaat Penelitian

Manfaat dari hasil pembuatan Tugas Akhir ini adalah :

1. Dapat meningkatkan keamanan sistem IoT yang sudah ada dengan mengintegrasikan dengan teknologi *blockchain*.
2. Mendapatkan pemahaman implementasi *blockchain* pada sistem keamanan IoT.
3. Menggunakan dan mengintegrasikan IoT dengan *Blockchain* dapat meningkatkan keamanan dan privasi data.

## Sistematika Penulisan

Adapun sistematika penulisan dari Laporan Tugas Akhir ini terbagi menjadi beberapa bab, sesuai dengan ketentuan yang ditetapkan di Fakultas Teknik, Universitas Mataram sebagai berikut:

**BAB I PENDAHULUAN**

Pada bab ini berisi Gambaran umum dan penjelasan mengenai latar belakang pemilihan judul penelitian, perumusan masalah yang dimana merupakan tulisan singkat berisi pertanyaan tentang topik penelitian yang nantinya akan dijawab oleh penulis sehingga penelitian yang dilakukan memiliki suatu kesimpulan dari hasil analisis dan visualisasi data penelitian yang dilakukan, selain itu terdapat pula batasan masalah yang merupakan batas-batas dari topik penelitian yang sedang dikaji atau diteliti, serta terdapat juga tujuan penelitian dan manfaat penelitian yang merupakan keuntungan yang didapat atau diperoleh oleh berbagai pihak dari penelitian yang dilakukan.

**BAB II TINJAUAN PUSTAKA DAN DASAR TEORI**

Pada bab ini berisi mengenai tinjauan pustaka dan landasan teori, yang dimana dalam tinjauan pustaka mengulas beberapa penelitian sebelumnya yang sejenis mengenai *blockchain* untuk sistem keamanan pada IoT, sedangkan pada dasar teori membahas mengenai teori-teori yang berkaitan dengan *blockchain* untuk sistem keamanan pada IoT yang akan dilakukan pada penelitian yang didapatkan melalui beberapa sumber-sumber seperti jurnal, buku, dsb.

**BAB III METODE PENELITIAN**

Pada bab ini menjelaskan tentang langkah-langkah dalam pelaksanaan penelitian yang dimana terdiri dari alur penelitian yang membahas mengenai Gambaran umum dalam bentuk diagram terkait alur penelitian yang dilakukan dan uraian metodologi yang membahas tahapan yang dilakukan pada alur penelitian diantaranya yaitu, pengambilan data, pengiriman data, pembuatan jaringan *blockchain* pada sistem IoT, penerimaan data, validasi dan otentikasi keamanan data dalam *blockchain*.

**BAB IV HASIL DAN PEMBAHASAN**

Pada bab ini membahas mengenai hasil dari penelitian yang telah dilakukan yang dimana terdiri dari hasil analisa, perancangan jaringan *blockchain*, dan pengujian pada perangkat IoT yang menyimpan data pada jaringan *private blockchain*.

**BAB V KESIMPULAN DAN SARAN**

Pada bab ini berisi kesimpulan yang merupakan jawaban dari masalah yang dirumuskan dalam bentuk kalimat tanya di rumusan masalah dan berisi saran yang membangun untuk pengembangan yang lebih baik di masa depan.

# BAB II

# TINJAUAN PUSTAKA DAN DASAR TEORI

## Tinjauan Pustaka

Pada tinjauan pustaka ini akan membahas penelitian yang pernah dilakukan sebelumnya atau yang relevan dengan penelitian yang akan dilakukan sebagai acuan dalam pengerjaan penelitian.

Penelitian pertama adalah "*Blockchain for IoT Security and Privacy: The Case Study of a* Smart Home".Studi ini mempelajari penggunaan *blockchain* untuk meningkatkan keamanan dan privasi pada lingkungan *Smart Home*, yang merupakan salah satu aplikasi penting dari IoT. Pendekatan studi kasus untuk mengevaluasi keefektifan penggunaan *blockchain* dalam meningkatkan keamanan dan privasi pada *Smart Home*. Penggunaan *blockchain* pada *Smart Home* dapat memberikan beberapa keuntungan dalam hal keamanan dan privasi, seperti mengurangi risiko serangan peretas dan memastikan privasi data pengguna. Secara keseluruhan, penelitian ini menyimpulkan bahwa penggunaan *blockchain* dapat memberikan solusi yang efektif untuk mengatasi tantangan keamanan dan privasi pada *Smart Home*. Studi ini menunjukkan bahwa implementasi *blockchain* pada *Smart Home* dapat meningkatkan tingkat keamanan, privasi, dan pengelolaan data yang lebih efektif pada lingkungan IoT (Dorri dkk., 2017).

Penelitian kedua adalah "*A secure and private data sharing architecture for IoT based on blockchain and smart contract*".Penelitian ini bertujuan untuk memperkenalkan arsitektur berbagi data yang aman dan pribadi untuk *Internet of Things* (IoT) menggunakan *blockchain* dan *smart contract*.Penelitian ini mengusulkan arsitektur berbagi data yang terdiri dari beberapa komponen, yaitu penyedia layanan, pengguna, dan *blockchain*. Dalam arsitektur ini, *blockchain* digunakan untuk memastikan keamanan dan privasi data, sedangkan *smart contract* digunakan untuk mengatur dan mengelola data yang dibagikan.Hasil dari penelitian ini menunjukkan bahwa arsitektur berbagi data dapat memberikan tingkat keamanan dan privasi yang lebih tinggi dibandingkan dengan arsitektur yang tidak menggunakan *blockchain*. Selain itu, arsitektur yang diusulkan juga lebih efisien dalam hal penggunaan sumber daya dan waktu respons (Mingyu, dkk.,2020).

Penelitian ketiga adalah "*Security threats and solutions to IoT using Blockchain: A Review*".Penelitian ini mengevaluasi penggunaan teknologi *blockchain* sebagai solusi untuk mengatasi ancaman keamanan pada lingkungan *Internet of Things* (IoT). Penelitian ini mengidentifikasi beberapa ancaman keamanan yang dihadapi oleh lingkungan IoT, seperti serangan DDoS, pencurian data, dan manipulasi data. Kemudian, penelitian ini membahas bagaimana teknologi *blockchain* dapat membantu mengatasi masalah-masalah keamanan ini. Dengan penggunaan *blockchain*, sistem IoT dapat dilindungi dari serangan peretas dan manipulasi data. Selain itu, penelitian ini juga membahas tentang tantangan dan kelemahan penggunaan *blockchain* untuk sistem IoT, seperti keterbatasan kapasitas transaksi dan biaya operasional yang tinggi. Namun, penelitian ini menyimpulkan bahwa penggunaan *blockchain* dapat memberikan solusi yang efektif untuk mengatasi ancaman keamanan pada lingkungan IoT. Secara keseluruhan, penelitian ini memberikan informasi yang berguna tentang penggunaan teknologi *blockchain* untuk mengamankan sistem IoT. Studi ini memberikan wawasan tentang bagaimana *blockchain* dapat digunakan untuk mengatasi ancaman keamanan dan memberikan solusi yang efektif untuk mengamankan data pada lingkungan IoT (Alam dkk., 2021).

## Dasar Teori

### *Internet of Things* (IoT)

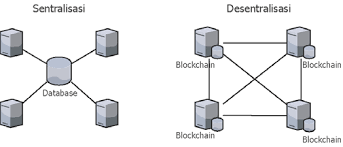
*Internet of Things* (IoT), merupakan sebuah konsep yang bertujuan untuk memperluas manfaat dari konektivitas internet yang tersambung secara terus menerus yang memungkinkan kita untuk menghubungkan mesin, peralatan, dan benda fisik lainnya dengan sensor jaringan dan aktuator untuk memperoleh data dan mengelola kinerjanya sendiri, sehingga memungkinkan mesin untuk berkolaborasi dan bahkan bertindak berdasarkan informasi baru yang diperoleh secara independen. *Internet of Things* atau sering disebut IoT adalah sebuah gagasan dimana semua benda di dunia nyata dapat berkomunikasi satu dengan yang lain sebagai bagian dari satu kesatuan sistem terpadu menggunakan jaringan internet sebagai penghubung. Misalnya sebuah *Smart Home* yang dapat dikelola lewat *smartphone* dengan bantuan koneksi internet. Pada dasarnya IoT bila mendapatkan sambungan internet sebagai media komunikasi dan *server* sebagai pengumpul informasi yang diterima untuk dianalisa (Efendi, 2018).

Dengan meningkatnya penggunaan perangkat pintar, kemampuan bertahan dan pengorganisasian mandiri jaringan menjadi sangat menantang. Namun, beberapa paradigma pengorganisasian mandiri yang diusulkan, dapat meningkatkan kekuatan jaringan. Dalam IoT, asosiasi dari berbagai perangkat heterogen mengurangi kemampuan jaringan sumber daya, yang menarik perhatian para peneliti ke arah bidang yang sedang berkembang ini.Perkembangan IoT mengarah pada sejumlah besar pembuatan konten, yang memperoleh unit pemrosesan besar, penyimpanan konten (*cache*), dan penyediaan *bandwidth*. Hal ini disebabkan karena fakta bahwa jumlah *node* kecil yang terhubung ke Internet akan mencapai 27 miliar pada tahun 2021. Beberapa aplikasi membutuhkan transmisi yang aman, sementara beberapa aplikasi lainnya membutuhkan penyimpanan lokal untuk transmisi cepat dan waktu respons yang rendah. Sejumlah besar konten ini dengan pemrosesan lokal akan membutuhkan teknik yang canggih untuk administrasi lokal (Ikram dkk.,2019).

### *Blockchain*

Sejak Bitcoin diperkenalkan pada tahun 2009, penggunaan *blockchain* telah meledak melalui penciptaan berbagai mata uang kripto, *decentralized finance* (DeFi) *applications*, *non-fungible tokens* (NFT), dan *smart contract*.

*Blockchain* adalah sebuah *database* terdistribusi atau buku besar yang dibagikan di antara *node*-*node* jaringan komputer. *Blockchain* terkenal dengan peran pentingnya dalam sistem mata uang digital untuk menjaga catatan transaksi yang aman dan terdesentralisasi, tetapi tidak terbatas pada penggunaan mata uang digital. *Blockchain* dapat digunakan untuk membuat data dalam industri apa pun menjadi tidak dapat diubah, istilah yang digunakan untuk mengGambarkan ketidakmampuan untuk diubah. Karena tidak ada cara untuk mengubah sebuah blok, satu-satunya kepercayaan yang dibutuhkan adalah pada saat pengguna atau program memasukkan data. Aspek ini mengurangi kebutuhan akan pihak ketiga yang terpercaya, yang biasanya adalah auditor atau manusia lain yang menambah biaya dan membuat kesalahan.



Gambar 2.1 Blockchain dan Database

*Blockchain* terdiri dari program yang disebut skrip yang melakukan tugas-tugas yang biasanya dilakukan dalam *database*: Memasukkan dan mengakses informasi serta menyimpan dan menyimpannya di suatu tempat. *Blockchain* didistribusikan, yang berarti banyak salinan disimpan di banyak mesin, dan semuanya harus cocok agar valid. *Blockchain* mengumpulkan informasi transaksi dan memasukkannya ke dalam sebuah blok, seperti sebuah sel pada *spreadsheet* yang berisi informasi. Setelah penuh, informasi tersebut dijalankan melalui algoritma enkripsi, yang menciptakan angka heksadesimal yang disebut *hash*. *Hash* tersebut kemudian dimasukkan ke dalam *header* blok berikutnya dan dienkripsi dengan informasi lain dalam blok tersebut. Hal ini menciptakan serangkaian blok yang dirantai bersama. (Hayes,2023).

*Blockchain* adalah sebuah inovasi yang telah menarik banyak sekali pertimbangan dari para ahli dan *Blockchain* adalah sebuah blok catatan yang diikat Setiap blok berisi dua bagian penting yaitu pertukaran dan *header* blok. Dalam *blockchain*, sebuah pertukaran membentuk korespondensi penting yang mengizinkan dua hub untuk memperdagangkan data satu sama lain. Struktur esensial dari pertukaran muncul di sisi kiri. Perhatikan bahwa berbagai peluncuran *blockchain* mungkin memiliki sedikit variasi dalam struktur pertukaran seperti pada Gambar dibawah ini (Premkumar and Sathya 2021).



Gambar 2.2 Struktur Blockchain

### *Blockchain*-IoT

*Blockchain* menyediakan penyimpanan data yang terdesentralisasi untuk sistem Teknologi Informasi (TI) umum. *Blockchain* diperkenalkan pada akhir tahun 2008 dan berfungsi sebagai platform untuk dan pemrosesan transaksi anonim menggunakan desentralisasi terdesentralisasi menggunakan jaringan komputer biasa. Biasanya, *Blockchain* dibangun sebagai sebuah daftar yang terhubung dari blok data, di mana mengubah satu bit di salah satu blok yang disimpan sebelumnya dapat segera ditemukan oleh rekan-rekan yang berpartisipasi. Dalam *Blockchain*, penambang adalah aktor utama utama yang memverifikasi validitas data yang disimpan. Kasus penggunaan yang terintegrasi dengan IoT telah meningkatkan perhatian yang tinggi dalam dekade terakhir, karena pemantauan rantai pasokan, pemantauan lingkungan, kota pintar, industri pintar, dan perawatan kesehatan berfokus pada kekekalan data dan membutuhkan sistem IoT untuk pengukuran, pengumpulan data, dan kontrol aktif. Dengan demikian, maka integrasi *Blockchain* dan IoT dapat mendukung BIoT menanggapi tuntutan penyimpanan yang persisten dan data sangat aman, di mana pengumpulan data otomatis menjadi kunci untuk menawarkan transparansi dan keandalan. BIoT yang sangat menuntut membutuhkan elaborasi dan analisis protokol IoT yang mendasari, yang membentuk dasar komunikasi untuk sistem IoT. Dengan demikian, studi tentang jangkauan komunikasi, kecepatan data, *maximum transmission units* (MTU), keandalan protokol komunikasi dan efisiensi energi diperlukan untuk mendukung dengan tepat Penerapan IoT (Andersson dkk., 2019).

### Algoritma *Consensus*

Algoritma *consensus* adalah sebuah teknik untuk mencapai sebuah kesepatan bersama di dalam sebuah kelompok. Seperti sebuah kelompok yang terdiri dari sepuluh *node* yang perlu menentukan pilihan tentang tugas yang menguntungkan mereka semua. Semuanya dapat mengusulkan sebuah pemikiran, namun yang dominan akan disetujui oleh salah satu yang paling mendorong mereka. Namun, perhitungan konsensus tidak hanya setuju dengan bagian yang dominan memberikan suara, namun juga menyetujui yang menguntungkan semua pihak. Konsensus *Blockchain* adalah strategi untuk membuat keseragaman dan kesopanan di dunia *online*. Kerangka kerja konsensus yang digunakan untuk ini pemahaman ini dikenal sebagai hipotesis *consensus* (Pahlajani, Kshirsagar, and Pachghare, 2019). *Blockchain* menerapkan beberapa algoritma *consensus* seperti berikut :

1. *Proof of Work* (PoW)

*Proof of Work* (PoW) adalah strategi konsensus yang digunakan pada Bitcoin. Jika sebuah *node* ingin mencatat sebuah blok, banyak usaha yang harus dilakukan oleh *node* tersebut untuk membuktikan bahwa *node* tersebut tidak memiliki keinginan untuk menyerang jaringan *blockchain* yang ada, hal ini yang mendasari cara kerja konsensus ini. Konsensus ini membutuhkan nilai *hash* yang dihitung tersebut untuk sama dengan atau lebih kecil dari nilai yang telah ditentukan sebelumnya. Ketika salah satu *node* dalam jaringan berhasil mencapai nilai yang ditentukan, maka blok tersebut akan disebarkan ke jaringan dan semua *node* dalam jaringan masing-masing mengkonfirmasi kebenaran nilai *hash* itu, dan setelah itu blok dinyatakan valid. Setelah itu semua *node* harus menambahkan blok ini ke *blockchain* mereka. *Nodes* yang menghitung nilai *hash* ini disebut dengan *miners* dan proses pengerjaan PoW ini disebut *mining* dalam Bitcoin (Frankenfield, 2019).

1. *Proof of Stake* (PoS)

*Proof of Stake* (PoS) adalah protokol yang lebih ramah energi dibandingkan dengan *Proof of Work* (PoW). *Miner* dalam PoS harus membuktikan kepemilikan dengan memiliki sejumlah uang (*cryptocurrency* yang dibuat pada *blockchain* tertentu). Pemilihan dengan melihat jumlah saldo cukup tidak adil karena orang yang paling kaya di jaringan tersebut akan mendominasi. Oleh karena itu, terdapat beberapa solusi yang diajukan untuk mengombinasi jumlah saldo dan hal lain untuk menambah blok baru pada jaringan. Contoh, pada *blockchain*, dimana membuat blok selanjutnya akan diacak dengan menggunakan rumus yang mencari nilai *hash* uang paling kecil lalu dikombinasikan dengan saldo orang tersebut (Frankenfield, 2019).

### Kriptografi

Teknologi kriptografi adalah kemampuan untuk bertukar pesan yang hanya dapat dibaca oleh penerima yang dituju dan digunakan untuk beberapa tujuan untuk mengamankan berbagai transaksi yang terjadi di jaringan, untuk mengontrol pembuatan unit mata uang baru, dan untuk verifikasi transfer aset digital dan token.Mata uang digital meniru konsep tanda tangan dunia nyata dengan menggunakan teknik kriptografi dan kunci enkripsi. Metode kriptografi menggunakan kode matematika tingkat lanjut untuk menyimpan dan mengirimkan nilai data dalam format yang aman yang memastikan hanya mereka yang memiliki data atau transaksi tersebut yang dapat menerima, membaca, dan memproses data, serta memastikan keaslian transaksi dan partisipan, seperti halnya tanda tangan di dunia nyata.

Pada teknologi *blockchain* kriptografi menggunakan metode *Hashing*, yang digunakan untuk memverifikasi integritas data transaksi di jaringan secara efisien. *Hashing* menjaga struktur data *blockchain*, mengkodekan alamat akun seseorang, merupakan bagian integral dari proses enkripsi transaksi yang terjadi antar akun, dan memungkinkan penambangan blok. Selain itu, tanda tangan digital melengkapi berbagai proses kriptografi ini, dengan mengizinkan partisipan asli untuk membuktikan identitas mereka ke jaringan (Seth, 2022).

### *Raspberry Pi*

*Raspberry Pi* adalah sebuah komputer kecil yang mumpuni yang memiliki ukuran sebesar kartu ATM. Perangkat ini dapat digunakan untuk proyek elektronik karena memiliki *input*, *output*, port digital dan dapat melakukan banyak hal layaknya PC desktop atau komputer. *Raspberry Pi* dapat menghubungkannya ke TV atau layar komputer dan *keyboard*. *Raspberry Pi* dibuat di Inggris oleh *Raspberry Pi Foundation* (Fernando, Meyliana, and Surjandy 2019). *Raspberry Pi* tidak menggunakan *hard disk*, namun menggunakan SD Card untuk proses *booting* dan penyimpanan data jangka-panjang. *Raspberry Pi* memiliki beberapa tipe, yaitu Model A dan Model B, dengan perbedaan spesifikasi seperti harga, SoC, CPU, GPU, RAM, dan lain-lain (Na and Hipertensiva n.d.). Pada penelitian ini, menggunakan Raspberry Pi 3 Model B (Fernando, Meyliana, and Surjandy 2019).

### Multichain

Multichain adalah platform yang membantu pengguna untuk membangun *Blockchain* pribadi tertentu yang dapat digunakan oleh organisasi untuk transaksi. API sederhana yang disediakan MultiChain membantu untuk mengatur rantai. Tujuan MultiChain membuat visibilitas *blockchain* harus selalu secara aktif disimpan dalam peserta yang dipilih untuk menghindari kebingungan untuk memastikan stabilitas dan kontrol atas transaksi, dan proses penambangan (*mining*) dapat dilakukan dengan lebih aman. Model *blockchain* ini hanya mentransaksikan akun yang divalidasi ke peserta rantai ini. Dalam MultiChain terdapat Proses *hand-shaking* dimana terjadi ketika *node* dalam *blockchain* terhubung satu sama lain. MultiChain terjadi ketika dua *node blockchain* terhubung. Identitas setiap *node* mewakili dirinya sendiri dengan alamat dengan daftar izin. Oleh karena itu, setiap *node* yang diwakilinya mengirimkan pesan ke pengguna lain. Koneksi *Peer to Peer* (P2P) dibatalkan jika mereka tidak menerima hasil yang memuaskan dari proses tersebut (Rumah, 2023).

### *Socket Programming*

*Socket* adalah sebuah cara untuk berkomunikasi dengan program atau *node* lain dengan menggunakan komunikasi antara *client* dan *server*. Agar suatu *socket* dapat berkomunikasi dengan *socket* lainnya, maka socket butuh diberi suatu alamat untuk identifikasi. Alamat *socket* terdiri dari alamat IP dan nomer *port*.Alamat IP dapat menggunakan alamat jaringan lokal (LAN) maupun internet.

# BAB III

# METODE PENELITIAN

## Tahapan Penelilitian

Perancangan alur dari tahapan penelitian ini merupakan Gambaran umum dari tahapan penelitian yang akan dilakukan dalam pengerjaan tugas akhir mulai dari awal hingga akhir. Tahapan penelitian tugas akhir diGambarkan seperti berikut:



Gambar 3.1 Diagram Alir Tahapan Penelitian

## Studi Literatur

Pada penelitian ini dilakukan studi literatur sebagai tahap awalan untuk melakukan penelitian, tujuan dari studi literatur ini adalah untuk memahami dasar teori dan konsep yang akan menjadi pendukung dalam melakukan penelitian yang berjudul ”Implementasi *Blockchain* Sebagai Sistem Keamanan IoT dalam Mendistribuskan dan Menjaga Privasi Data”.

## Analisis Kebutuhan

1. *Hardware*
2. Laptop HP Notebook 14-am013TU. Intel® Celeron® CPU N3060@ 1.60GHz *with* 8.0GB RAM
3. Raspberry Pi 3a *with* 1.0 GB RAM
4. *Software*

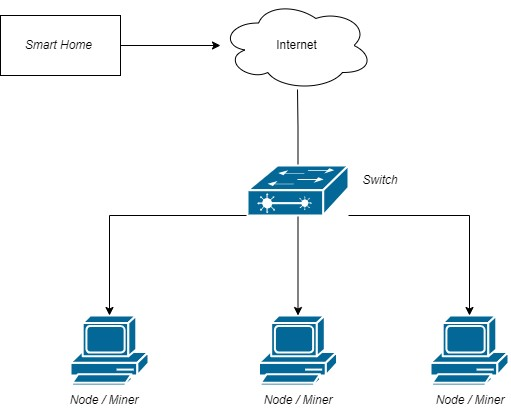
*Software* yang digunakan pada penelitian ini sebagai berikut:

1. Visual Studio Code sebagai *text* *editor* yang membantu proses develop program simulasi *Smart Home* IoT dan program *server*.
2. Multichain sebagai platform yang dapat memberikan akses ke *server* *private blockchain*.
3. Ubuntu 22.04.3 desktop sebagai OS yang digunakan untuk menjalankan *private blockchain*.

## Perancangan Topologi

### Topologi Jaringan

Pada penelitian ini menggunakan topologi jaringan, yaitu terdapat jaringan internet untuk mejadi jaringan lokal antara perangkat *smart home* dengan PC (*node* / *miner*) serta terdapat 3 *node* yang dapat berperan sebagai *miner* dan saling terhubung antara *node* yang satu dengan *node* yang lain. Perangkat *smart home* yang akan disimulasikan dalam bentuk program *Python* yang dijalankan pada perangkat *Raspberry Pi* yang akan mengirimkan data kepada PC yang berperan sebagai *miner*. Topologi jaringan yang dibuat dapat dilihat pada Gambar 3.2 berikut ini :

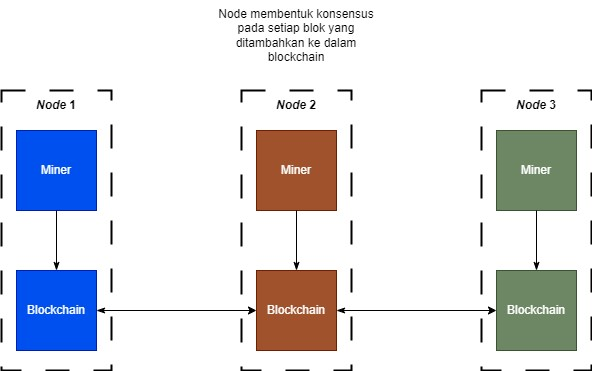


Gambar 3.2 Topologi Jaringan

PC berperan sebagai *node* dalam jaringan dan dapat juga berperan sebagai *miner*. *Miner* bertanggung jawab untuk melakukan pengecekan dan validasi terhadap aktivitas yang terjadi di dalam sistem, misalnya *smart home devices* yang hendak mengirimkan dan menyimpan data ke dalam sistem apakah diizinkan atau tidak dan apakah data yang dikirimkan merupakan data yang valid atau tidak. Penelitian ini menggunakan platform Multichain yang akan dipasang pada semua *node* yang terdapat di dalam sistem.

### Alur Komunikasi *Blockchain*

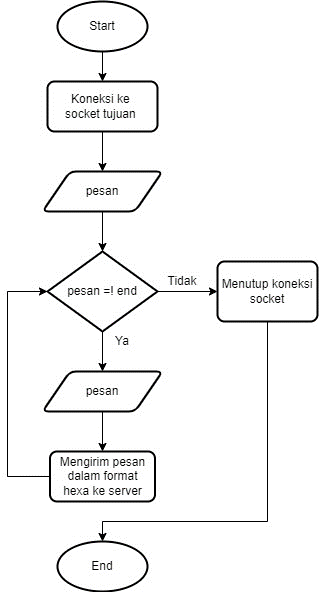
Peran *node* dan *miner* dalam jaringan *blockchain* yaitu *node* dapat saling berkomunikasi dengan sesama *node*, dimana masing-masing *node* terdapat *miner* untuk memvalidasi transaksi dan menambahkan ke blok baru ke dalam *blockchain* seperti yang terlihat pada gambar dibawah ini :



Gambar 3.3 Alur Komunikasi Blockchain

### Perangkat *Smart Home*

Pada penelitian ini dilakukan pembuatan perangkat *smart home* berbasis program yang dimana pembuatan program menggunakan bahasa *python*. *Smart Home* berbasis program ini akan dijalankan pada perangkat *Raspberry Pi* dengan tujuan untuk dapat mengirimkan data ke *node admin* yang kemudian akan diproses dan akan didistribusikan ke *blockchain* apabila data sudah di dicek dan divalidasi.



Gambar 3.4 Diagram Alir Program Simulasi Smart Home

### *Node admin*

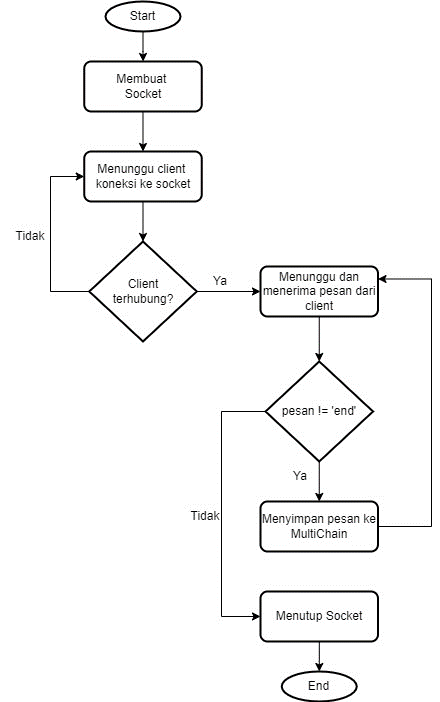
Berdasarkan Gambar.3.2 terdapat tiga *node* yang dimana akan dipilih satu *node* yang akan berperan sebagai *node admin*. *Node admin* akan berperan untuk menginisiasi *blockchain*, seperti memberi izin *node* yang diperbolehkan untuk terhubung ke dalam *blockchain*, memberi izin *node* untuk melakukan *mining* ke dalam *blockchain*, menjalankan sebuah program yang berfungsi sebagai penguhubung antara *blockchain* pada Multichain dengan perangkat *Smart Home* berbasis program yang akan mengirimkan paket data kemudian meneruskannya ke dalam *blockchain*. Untuk dapat menerima paket yang dikirim dari perangkat *smart home* perlu dibuat jalur komunikasi dengan menggunakan metode *socket programming* yang berfungsi sebagai penghubung antara *node admin* dengan perangkat *smart home* berbasis program *python* tersebut.

### Pengiriman Data

Data atau pesan dikirimkan dari perangkat *smart home* yang telah dihubungkan dengan *node admin* kemudian perangkat *smart home* akan disimulasikan dengan menggunakan program *Python* yang mengirimkan data kepada *node admin*. Setelah terhubung, perangkat *smart home* dapat mengirim data ke *node* Multichain menggunakan *socket*. Data akan diserialisasi menjadi format yang sesuai .JSON atau format khusus, lalu kirim data melalui koneksi *socket* yang sudah dibuat. Dengan menggunakan perintah send() untuk mengirim data melalui *socket*. Data yang dikirimkan akan memilki waktu jeda tiap pengiriman data adalah 25 detik karena harus menunggu proses *mining* dari *blockchain* terlebih dahulu.

### Penerimaan Data

Data yang telah dikirim oleh perangkat *smart home* kemudian akan diterima oleh sebuah program yang dijalankan pada *node admin*. Program ini memiliki fungsi untuk menerima data yang dikirim oleh perangkat *smart home* kemudian meneruskannya kepada Multichain.



Gambar 3.5 Alur Penerimaan Data

## Pengujian

### Koneksi dengan Multichain

Setelah pesan atau data diterima oleh *node admin*, kemudian data akan diteruskannya ke dalam *blockchain* dengan menggunakan API yang telah disediakan oleh Multichain sehingga pesan tesebut dapat disimpan ke dalam *blockchain*. Untuk dapat berkomunikasi dengan API milik MultiChain dengan menggunakan bahasa pemrograman *python*, maka dapat memanfaatkan *library* Savoir.

### Penyimpanan Data

Setelah terhubung dengan API milik Multichain, maka untuk selanjutnya data dapat diterima dan disimpan ke dalam *blockchain*. Data yang telah diterima dan masih dalam format hexadesimal akan diteruskan ke dalam *blockchain* untuk disimpan ke dalam blok baru. Multichain memiliki fitur *stream* yang berfungsi sebagai tempat untuk menyimpan data secara umum, dan istilah menyimpan data di dalam *stream* dikenal dengan istilah *publish*.

### Validasi Data

Ada dua jenis media yang akan mencoba mengirimkan data ke dalam *blockchain* yaitu *permissioned device* dan *permissionless device*. *Permissioned devices* adalah perangkat yang diizinkan oleh sistem untuk menyimpan dan melihat data. Sedangkan *permissionless devices* adalah perangkat yang berada di dalam maupun di luar sistem yang tidak memiliki izin apapun untuk melakukan aktivitas yang sama seperti *permissioned devices*.

Proses validasi data dalam Bitcoin biasa disebut dengan proses *mining*. Bitcoin menggunakan konsep yang bernama PoW (*Proof of Work*). Proses vaidasi didalam multichain menggunakan skema kombinasi antara jumlah *miner* dengan parameter bernama *mining-diversity* yang dibatasi dengan 0 ≤ *mining diversity* ≤ 1. *Miner* adalah sebutan untuk komputer atau *node* yang bertugas untuk melakukan proses validasi data pada *blockchain*.

Menerapkan semua perubahan izin yang ditentukan oleh transaksi dalam blok secara berurutan.

Menghitung jumlah *miner* yang diizinkan yang ditentukan setelah menerapkan perubahan, misalkan jumlah *miner* adalah 2.

Mencari nilai spacing degan mengkalikan jumlah *miner* dengan nilai *mining-diversity*. Misalkan nilai *mining-diversity* = 0.3, maka diperoleh nilai *spacing* = 2 x 0.3 = 0.6. Kemudian nilai 0.6 dibulatkan ke atas menjadi 1 sehingga nilai *spacing* yang sebenarnya adalah 1.

Jika *miner* dari *block* ini telah melakukan validasi pada salah satu dari (nilai *spacing* -1) *block* sebelum ini, maka proses validasi tidak sah. Jika tidak, maka *miner* yang bersangkutan akan didelegasikan untuk melukakn proses validasi pada *block* tersebut dan *block* akan dianggap valid dan dapat disimpan ke dalam *blockchain*.

Skema ini memberlakukan teknik penjadwalan *round-robin*, dimana *miner* yang diizinkan harus membuat *block* secara bergiliran untuk menghasilkan *blockchain* yang valid. Parameter *mining-diversity* mendefinisikan ketatnya skema Nilai 1 pada *mining-diversity* memastikan bahwa setiap *miner* akan masuk ke dalam rotasi *round-robin*, sedangkan nilai 0 menunjukkan tidak ada batasan sama sekali. Secara umum nilai yang lebih tinggi akan lebih aman, namun nilai yang terlalu dekat dengan 1 dapat menyebabkan *blockchain* membeku jika beberapa *miner* menjadi tidak aktif.

# BAB IV

# HASIL DAN PEMBAHASAN

## Implementasi Multichain

Implementasi multichain dilakukan pada setiap *node* dengan melakukan konfigurasi multichain terlebih dahulu kemudian melakukan proses *install*. Untuk proses pembuatan jaringan *private blockhain* dapat dilakukan dengan perintah yang dijalankan pada terminal dengan CLI (*Command Line Interface*). Pada implementasi multichain ini akan dilakukan skenario percobaan sebagai berikut:

1. *Permissioned device* melakukan koneksi kedalam jaringan *private blockchain*.
2. *Permissionless device* melakukan koneksi kedalam jaringan *private blockchain*.

*Permissioned device* digambarkan sebagai perangkat yang memiliki izin untuk dapat melakukan penyimpanan data dan melihat data sedangkan *permissionless device* digambarkan sebagai perangkat yang tidak dapat melakukan aksi seperti *permissioned device*.

### *Create Blockchain*

Proses *create* atau pembuatan jaringan *private blockchain* dapat dilakukan dengan menjalankan perintah dengan CLI.

|  |
| --- |
| multichain-util create *blockchain*\_name. |

Pada script diatas digunakan untuk pembentukan blockchain yang apabila perintah diatas dijalankan akan membuat jaringan private blockchain pada node dan kemudian akan secara default akan memberi peran admin kepada node yang melakukan pembentukan private blockchain. multichain-util adalah perintah dasar multichain yang digunakan pada saat akan membuat blockhain, create adalah perintah agar dapat membuat nama blockchain, *blockchain*\_name adalah nama blockchain yang akan digunakan yaitu iotchain.

* Proses Pembentukan *Blockchain*

|  |
| --- |
| Blockchain parameter set was successfully generated.  You can edit it in **/home/jarkom/.multichain/iotchain/params.dat** before running multichaind for the first time.­  To generate blockchain please run "**multichaind iotchain -daemon**". |

*Blockchain* akan membentuk parameter dan secara otomatis terbentuk folder sesuai dengan nama *blockchain* pada /home/jarkom/.multichain/iotchain/params.dat. Sebelum melakukan *run blockchain* terlebih dahulu dapat dilakukan edit beberapa parameter pada lokasi folder *blockchain* berada. Secara *default* apabila *node* yang membentuk jaringan *private blockchain* akan menjadi *node admin*, yang semua inisiasi dan aksi dikontrol penuh oleh *node admin*.

### *Run Blockchain*

|  |
| --- |
| multichaind *blockchain*\_name -daemon |

Pada *script* diatas melakukan *run blockchain* pada jaringan *private blockchain* dimana pada saat pembuatan jaringan *private blockchain* pertama kali akan secara otomatis juga terbentuk struktur *blockchain*. multichaind adalah perintah dasar pada multichain yang digunakan pada saat akan menjalankan blockchain, *blockchain*\_name adalah nama blockchain yaitu iotchain, -daemon adalah perintah pada multichain untuk menjalankan blockchain.

* Membentuk struktur *blockchain*

|  |
| --- |
| Starting up node ...  **Looking for genesis block ...**  **Genesis block found**  other nodes can connect to this node using:  **multichaind iotchain@192.168.9.21:4287**  This host has multiple IP addresses, so from some networks:  multichaind iotchain@10.26.48.32:4287  Listening for API requests on port 4286 (local only - see rpcallowip setting)  **Node ready**. |

*Blockchain* dijalankan kemudian terbentuk *genesis block*. *Genesis block* merupakan *block* pertama dari *blockchain*. Terdapat perintah multichaind iotchain@192.168.9.21:4287 yang merupakan cara agar *node* lain dapat terhubung sebagai anggota dalam jaringan *private blockchain* yang telah dibuat.

### Penelusuran *Blockchain*

Penelusuran informasi *blockchain* dilakukan untuk mendapatkan struktur blockchain yang berada pada jaringan private blockchain dan informasi nilai address dan port dari node.

|  |
| --- |
| multichain-cli blockchain\_name [option] |

Pada script diatas dilakukan untuk mendapatkan detail info didalam node yang sudah menjalankan jaringan private blockchain dengan menggunakan perintah multichain-cli iotchain getinfo kemudian akan diberikan parameter dan nilai dari info blockchain didalam node. multichain-cli adalah perintah dasar pada multichain yang digunakan untuk mendapatkan informasi, blockchain\_name adalah nama blockchain yaitu iotchain, getinfo adalah perintah untuk mendapat detail info dari blockchain.

* Detail *blockchain*

|  |
| --- |
| {  "version" : "2.3.3",  "nodeversion" : 20303901,  "edition" : "Community",  "protocolversion" : 20013,  "chainname" : "iotchain",  "description" : "MultiChain iotchain",  "protocol" : "multichain",  **"port" : 4287**,  "setupblocks" : 60,  **"nodeaddress" : "iotchain@192.168.9.21:4287",**  "burnaddress" : "1XXXXXXWoqXXXXXXBzXXXXXXVcXXXXXXWiAJWE",  "incomingpaused" : false,  "miningpaused" : false,  "offchainpaused" : false,  "walletversion" : 60000,  "balance" : 0,  "walletdbversion" : 3,  "reindex" : false,  **"blocks" : 3**,  "chainrewards" : 0,  **"streams" : 1**,  "timeoffset" : 0,  "connections" : 0,  "proxy" :  "difficulty" : 5.96046447753906e-8,  "testnet" : false,  "keypoololdest" : 1705221318,  "keypoolsize" : 2,  "paytxfee" : 0,  "relayfee" : 0,  "errors" : "#  } |

Detail info dari *blockchain* dengan menjalankan perintah multichain-cli blockchain\_name getinfo. Kemudian didalam *blockchain* terbentuk *blocks* sebanyak 3, yang dimana jumlah *block* akan terus bertambah apabila terjadi penambahan data pada jaringan *blockchain*.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Nilai** | **Keterangan** |
| Port | 4287 | nilai *default* port |
| *Setup blocks* | 60 | Jumlah *default* *setup block*s pada parameter |
| *Node address* | iotchain@192.168.9.21:4287 | Alamat *node* |
| *Blocks* | 3 | Jumlah sementara *blocks* dalam *blockchain* |
| *Streams* | 1 | Jumlah *default streams* saat *blockchain* dibuat pertama kali |

Tabel 4.1 Info Blockchain yang Dapat diakses

Pada Tabel 4.1 terdapat beberapa parameter yang dapat diakses secara manual jumlah maupun informasi di dalam *blockchain* yang telah dibuat, nilai dan parameter pada Tabel 4.1 didapatkan berdasarkan Gambar 4.3 mengenai info *blockchain*.

* Mendapatkan struktur block

|  |
| --- |
| multichain-cli blockchain\_name getblock block\_value |

Pada script diatas merupakan perintah untuk mendapatkan struktur block didalam blockchain agar dapat mengetahui nilai yang ada pada block. multichain-cli adalah perintah dasar pada multichain yang digunakan untuk mendapatkan informasi, blockchain\_name adalah nama blockchain yaitu iotchain, getblock adalah perintah untuk mendapatkan detail block, block\_value adalah nilai dari block didalam *blockchain*.

* Struktur block

|  |
| --- |
| {  **"hash":"00d91c6b42286933cfc55c15c4f75d664b77332bbbe10d05abb14269413b4365"**,  **"miner": "1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn"**,  **"confirmations": 3**,  "size": 264,  **"height": 3**,  "version": 3,  "merkleroot":"eb89583608a6]7866642ffaba87da528bae3ca58d3823f27ed2b9923c1791a8f",  "tx": [  "eb89583608a697866642ffaba87da528bae3ca58d3823f27ed2b9923c1791a8f"  "time": 1705221337,  **"nonce": 721**,  "bits": "2000ffff",  "difficulty": 5.96046447753906e-8,  **"previousblockhash":"00f2ff1f6c38d12ff1ee1af6b802540baff1fb9dfa82184ede9bdd6cef603447",**  **"nextblockhash":"001183756dfd69b85e80a37f9ca3e3a3984e044e9264e26043b481f41ffc040f**  } |

Mendapatkan detail dari struktur block yang berada pada *blockchain* dengan multichain-cli iotchain getblock 3. Dimana “*hash*” adalah nilai *hashing* dari *block* yang secara *runtime* akan terus berubah apabila *block* terus bertambah, “*miner*” adalah alamat yang terbentuk secara acak yang khusus digunakan untuk *node*.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Nilai** | **Keterangan** |
| *Hash* | 00d91c6b42286933cfc55c15c4f75d664b77332bbbe10d05abb14269413b4365 | *hash* dari *block* |
| *Miner* | 1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn | Alamat dari *node* |
| *Nonce* | 712 | Jumlah konsensus yang berjalan saat menambahkan *block* |
| *Previous block hash* | 00f2ff1f6c38d12ff1ee1af6b802540baff1fb9dfa82184ede9bdd6cef603447 | Nilai *hash* *block* sebelumnya |
| *Next block hash* | 001183756dfd69b85e80a37f9ca3e3a3984e044e9264e26043b481f41ffc040f | Nilai *hash block* selanjutnya |

Tabel 4.2 Parameter dalam Struktur Block pada Node Admin

Pada Tabel 4.2 terdapat beberapa parameter dan nilai yang diberi keterangan yang menjelaskan juga bentuk struktur *block* di dalam *blockchain* yang didapat berdasarkan pada Gambar 4.4 mengenai detail *block*.

### Mencari *Permissions*

Mencari permissions merupakan perintah yang dijalankan untuk mendapatkan jenis izin aksi yang dapat dilakukan node didalam jaringan private blockchain.

|  |
| --- |
| multichain-cli blockchain\_name listpermissions |

Pada script diatas digunakan untuk mengetahui jenis izin aksi dari node yang ditampilkan secara list. multichain-cli adalah perintah dasar pada multichain yang digunakan untuk mendapatkan informasi, blockchain\_name adalah nama blockchain yaitu iotchain, listpermissions adalah perintah untuk memberikan detail list permissions dari alamat node untuk mengetahui peran node diadalam jaringan private blockchain.

* *List permissions* dalam *blockchain*

|  |
| --- |
| {  **"address" : "1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn"**,  "for" : null,  **"type" : "mine"**,  "startblock" : 0,  "endblock" : 4294967295  },  {  **"address" : "1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn"**,  "for" : null,  **"type" : "admin"**,  "startblock" : 0,  "endblock" : 4294967295  }, |

Perintah *list permissions* untuk mengetahui *address* dan *type* agar dapat mengetahui peran dari *address*, yang dilihat berdasarkan nilai *type* dengan menjalankan perintah multichain-cli iotchain listpermissions. Kemudian didapatkan *address* dari *node* memiliki *type* sebagai *mine* dan admin.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Nilai** | **Keterangan** |
| *Address* | 1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn | Alamat dari *node* |
| *Type* | *Mine* dan admin | Jenis dari alamat *node* yang menjelaskan peran *node* didalam *blockchain* |

Tabel 4.3 Alamat dan Jenis Node dalam Blockchain

Pada Tabel 4.3 terdapat parameter dan nilai yang dijelaskan dengan diberi keterangan yang di dapat berdasarkan pada Gambar 4.5 mengenai *list permission* dalam *blockchain*.

## Skenario *Permissioned Device*

Koneksi antar *node* didalam jaringan *private blockchain* dapat terjadi dengan proses koneksi menggunakan mekanisme *hand shaking*, dimana *node* yang ingin terhubung (*client*) akan diberi alamat secara khusus, alamat tersebut digunakan untuk terhubung dengan *node* yang kemudian akan memberikan *permission* (admin).

### Koneksi oleh *Node Client*

|  |
| --- |
| multichaind blockchain\_name@<seed-node-ip>:<seed-node-port>. |

Pada script diatas adalah cara koneksi node client kedalam jaringan private blockchain. multichaind adalah blockchain\_name adalah, <seed-node-ip>:<seed-node-port> adalah alamat IP dan port dari node admin.

* Proses koneksi *node client*

|  |
| --- |
| **Retrieving blockchain parameters from the seed node 192.168.9.21:4287**  ...  Blockchain successfully initialized.  Please ask blockchain admin or user having activate permission to let you connect and/or transact:  **multichain-cli iotchain grant 1Ef2MZGZBj7vC36g7bGtmXopDsAnLfpCjuHGeP connect**  **multichain-cli iotchain grant 1Ef2MZGZBj7vC36g7bGtmXopDsAnLfpCjuHGeP connect, send, receive** |

*Node client* yang mencoba terhubung dengan jaringan *private blockchain* harus mendapatkan *permissions* oleh *Node admin* terlebih dahulu. Seperti yang terlihat dimana terdapat aksi yang dapat dilakukan *Node client* dapat berupa *connect*, *send*, *receive*.

|  |  |
| --- | --- |
| **Parameter** | **Keterangan** |
| iotchain | Nama *private blockchain* |
| grant | Perintah untuk melakukan proses *handshake* antar *node* |
| *connect, send, receive* | Aksi yang akan dapat dilakukan oleh *node* |

Tabel 4.4 Mekanisme handshake

Pada Tabel 4.4 terdapat parameter dengan keterangan yang dapat menjelaskan fungsi dari perintah yang *node client* dapat lakukan kemudian akan meminta *permissions* kepada *node* admin terlebih dahulu.

### *Permissions* oleh *Node admin*

|  |
| --- |
| multichain-cli grant address\_node-client connect,send,receive |

Pada script diatas merupakan pemberian *permission* untuk tergabung pada jaringan private blockchain yang berada pada *node client dan* dapat diulangi untuk dijalankan pada *node admin*. multichain-cli adalah perintah dasar pada multichain yang digunakan untuk mendapatkan informasi, grant adalah perintah untuk melakukan mekanisme handshaking antar *node*, blockchain\_name adalah nama blockchain yaitu iotchain, address\_node-client adalah alamat khusus dari *node client* yang diberikan oleh *node* admin, connect,send,receive adalah jenis aksi yang dapat dilakukan *node client* didalam jaringan *private blockchain*.

|  |
| --- |
| {"method":"grant","params":["1Ef2MZGZBj7vC36g7bGtmXopDsAnLfpCjuHGeP", "connect, send, rece  ive"],"id":"86226335-1705221510", "chain\_name":"iotchain"}  **0a4d46c91be3662e0491a85609e5ca932c7557cc16c72e153c599b2568ee5e6** |

Perintah *grant* diikuti dengan alamat dari *node client*, yang digunakan untuk memberikan *permission* agar *node client* dapat tergabung dalam jaringan *private blockchain* yang dikelola dan segala aksinya diatur oleh *node admin*. Kemudian terdapat *public key* yang didapatkan dari pemberian *permission* oleh *node admin*.

### *Run* pada *Node Client*

|  |
| --- |
| multichaind *blockchain*\_name -daemon |

Pada *script* diatas melakukan *run blockchain* pada jaringan *private blockchain* dimana pada saat pembuatan jaringan *private blockchain* pertama kali akan secara otomatis juga terbentuk struktur *blockchain*. multichaind adalah perintah dasar pada multichain yang digunakan pada saat akan menjalankan blockchain, *blockchain*\_name adalah nama blockchain yaitu iotchain, -daemon adalah perintah pada multichain untuk menjalankan blockchain.

* Membentuk struktur *blockchain*

|  |
| --- |
| Starting up node ...  **Looking for genesis block ...**  **Genesis block found**  **Retrieving blockchain parameters from the seed node 192.168.9.21:4287**  **...**  other nodes can connect to this node using:  **multichaind iotchain@192.168.9.23:4287**  This host has multiple IP addresses, so from some networks:  multichaind iotchain@10.26.48.22:4287  Listening for API requests on port 4286 (local only - see rpcallowip setting)  **Node ready**. |

Kemudian dapat dijalankan perintah dengan multichaind iotchain -daemon, *node client* berhasil terhubung pada jaringan *private blockchain* setelah diberikan *permission* oleh *node admin*. Kemudian secara otomatis parameter yang berada pada *node admin* di *generate* pada *node client*.

|  |  |
| --- | --- |
| **Nilai** | **Keterangan** |
| Multichaind iotchain -daemon | Perintah untuk menjalankan *blockchain* |
| Multichaind iotchain@192.168.9.23:4287 | Alamat IP *node* yang terhubung dengan kabel LAN yang dapat digunakan untuk *node client* lainnya terhubung |
| Multichaind iotchain@10.26.48.22:4287 | Alamat IP *node* yang terhubung dengan jaringan WiFi yang dapat digunakan untuk node client lainnya terhubung |

Tabel 4.5 Perintah Koneksi dengan Node

Pada Tabel 4.5 terdapat nilai dan keterangan yang dimana dapat menjelaskan fungsi dari masing-masing perintah yang dapat digunakan.

### Penelusuran pada *Node Client*

|  |
| --- |
| multichain-cli iotchain getblock 18 |

Pada script diatas. multichain-cli adalah perintah dasar pada multichain yang digunakan untuk mendapatkan informasi, blockchain\_name adalah nama blockchain yaitu iotchain, getblock adalah perintah untuk mendapatkan detail block, block\_value adalah nilai dari block didalam *blockchain*.

* Struktur block

|  |
| --- |
|  |



Gambar 4.9 Mendapatkan Detail Blocks pada Node Client

Pada Gambar 4.9 dijalankan perintah multichain-cli blockchain\_name getblock 25 dan didapatkan jumlah *block* yang bertambah dan memberikan juga “*hash*” dari *block* itu sendiri dan nilai “*miner*” adalah alamat dari *node admin*.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Nilai** | **Keterangan** |
| *Hash* | 00ae807458948b6874f0098477923b4cc3618067c2b9d17c1b2c807d10e86fac | *hash* dari *block* |
| *Miner* | 1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn | Alamat dari *node* |
| *Nonce* | 213 | Jumlah konsensus yang berjalan saat menambahkan *block* |
| *Previous block hash* | 00715195c5641a994c86ea224ebbdef6ab030782fcc1e53f826842d13ec9df7d | Nilai *hash block* sebelumnya |
| *Next block hash* | 005e6dc228d87e39964dd7047de89e9389e5a5757843b254ae2fbd53bc96e902 | Nilai *hash block* selanjutnya |

Tabel 4.6 Parameter dalam Struktur Block pada Node Client

Pada Tabel 4.6 terdapat beberapa parameter dan nilai yang diberi keterangan yang menjelaskan juga bentuk struktur *block* di dalam *blockchain* yang didapat berdasarkan pada Gambar 4.9 mengenai detail *block*.

* Peer info node admin

|  |
| --- |
| multichain-cli blockchain\_name getpeerinfo |

Pada script diatas merupakan perintah untuk mendapat info koneksi *peer-to-peer* antar *node* admin dan *node client* dalam jaringan *private blockchain*. multichain-cli adalah perintah dasar pada multichain yang digunakan untuk mendapatkan informasi, blockchain\_name adalah nama *blockchain* yaitu iotchain, getpeerinfo adalah perintah untuk mendapatkan address dan handshake node yang melakukan koneksi *peer-to-peer*.

|  |
| --- |
| [  {  "id" : 3,  "addr" : "192.168.9.23: 40930",  "addrlocal" : "192.168.9.21:4287",  "services" : "0000000000000001",  "lastsend" : 1705221642,  "lastrecv" : 1705221642,  "bytessent" : 11070,  "bytesrecv" : 6319,  "conntime" : 1705221541,  "pingtime" : 0.006231,  "version" : 70002,  "subver" : "/MultiChain: 0.2.0.13/",  "handshakelocal" : "1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn",  "handshake" : "1Ef2MZGZBj7v36g7bGtmXopDsAnLfpCjuHGeP",  "inbound" : true,  "encrypted" : false,  "startingheight" : 0,  "banscore" : 0,  "synced\_headers" : -1,  "synced\_blocks" : -1,  "inflight" : [  ],  "whitelisted" : false  },  {  "id" : 4,  "addr" : "10.26.48. 22:4287",  "addrlocal" : "10.26.48.32: 41894",  "services" : "0000000000000001",  "lastsend" : 1705221644,  "lastrecv" : 1705221644,  "bytessent" : 3039,  "bytesrecv" : 3170,  "conntime" : 1705221542,  "pingtime" : 0.211778,  "version" : 70002,  "subver" : "/MultiChain:0.2.0.13/",  "handshakelocal" : "1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn",  "handshake" : "1Ef2MZGZBj7vC36g7bGtmXopDsAnLfpCjuHGeP",  "inbound" : false,  "encrypted" : false,  "startingheight" : 1,  "banscore" : 0,  "synced\_headers" : 15,  "synced\_blocks" : -1,  "inflight" : [  ],  "whitelisted" : false  }  ] |

Menggunakan perintah multichain-cli iotchain getpeerinfo dapat mengetahui jumlah *node* yang terhubung dengan jaringan *private blockchain* yang telah dibuat dan mendapatkan detail informasi dari setiap node. Pada Gambar 4.10 merupakan *peer* info pada *node* admin. Nilai addr dan addrlocal pada id:3 dan id:4 berbeda, dimana pada id:3 nilai addr:192.168.9.23:40930 merupakan alamat dari *node* *client* yang nilai IP *address* didapatkan karena node terhubung dengan kabel LAN dan nilai addrlocal:192.168.9.21:4287 merupakan alamat dari *node* admin yang nilai IP *address* didapatkan karena *node* terhubung dengan kabel LAN sedangkan pada id:4 nilai addr: 10.26.48.22:4287 merupakan alamat dari *node client* yang nilai IP *address* didapatkan karena terhubung jaringan internet dan nilai addrlocal: 10.26.48.32:41894 merupakan alamat dari *node* admin yang nilai IP address di dapatkan karena terhubung jaringan internet.

* Peer info node client

|  |
| --- |
| [  {  "id" : 2,  "addr" : "192.168.9.21:4287",  "addrlocal" : "192.168.9.23:40930",  "services" : "0000000000000001",  "lastsend" : 1705221632,  "lastrecv" : 1705221632,  "bytessent" : 6255,  "bytesrecv" : 11006,  "conntime" : 1705221541,  "pingtime" : 0.041175,  "version" : 70002,  "subver" : "/MultiChain: 0.2.0.13/",  "handshakelocal" : "1Ef2MZGZBj7vC36g7bGtmXopDsAnLfpCjuHGeP",  "handshake" : "1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn",  "inbound" : false,  "encrypted" : false,  "startingheight" : 15,  "banscore" : 0,  "synced\_headers" : 23,  "synced\_blocks" : 23,  "inflight" : [  "whitelisted" : false  },  {  "id" : 3,  "addr" : "10.26.48. 32:41894",  "addrlocal" : "10.26.48.22:4287",  "services" : "0000000000000001",  "lastsend" : 1705221634,  "lastrecv" : 1705221634,  "bytessent" : 3106,  "bytesrecv" : 2975,  "conntime" : 1705221542,  "pingtime" : 0.424476,  "version" : 70002,  "subver" : "/Multichain:0.2.0.13/",  "handshakelocal" : "1Ef2MZGZBj7vC36g7bGtmXopDsAnLfpCjuHGeP",  "handshake" : "1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn",  "inbound" : true,  "encrypted" : false,  "startingheight" : 15,  "banscore" : 0,  "synced\_headers" : 23,  "synced\_blocks" : 23,  "inflight" : [  "whitelisted" : false  }  ] |

Menggunakan perintah multichain-cli blockchain\_name getpeerinfo dapat mengetahui jumlah *node* yang terhubung dengan jaringan *private blockchain* yang telah dibuat dan mendapatkan detail informasi dari setiap node. Pada Gambar 4.11 merupakan *peer* info dari *node client*.

Dimana pada Gambar 4.10 untuk *node admin* dan 4.11 untuk *node client*, kedua *node* dapat terhubung dengan menggunakan mekanisme *handshake*. Pada Gambar 4.11 nilai *handshake local* adalah alamat dari *node* dan nilai dari *handshake* adalah alamat *node* yang terhubung.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Nilai** | **Keterangan** |
| *addr* | 192.168.9.21:4287 | Nilai IP *node* admin dan *default port network* |
| *addrlocal* | 192.168.9.23:40930 | Nilai IP *node client* dan *port handshake* |
| handshake | 1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn | Alamat *node* admin |
| *handshakelocal* | 1Ef2MZGZBj7vC36g7bGtmXopDsAnLfpCjuHGeP | Alamat *node client* |

Tabel 4.7 Keterangan mekanisme handshake

## Membuat *Stream*

*Stream* adalah tempat menyimpan semua transaksi yang dilakukan di dalam jaringan *private blockchain* dan perintah untuk menyimpan transaksi data pada multichain menggunakan perintah *publish*.

### *Create Stream*

|  |
| --- |
| multichain-cli blockchain\_name create stream stream\_name “{\”restrict\”:\”write\”}”. |

*Create streams* dapat dilakukan dengan mengatur *mode restrict* dengan *write* dimana dapat dilakukan proses penyimpanan data secara langsung. multichain-cli adalah perintah dasar pada multichain yang digunakan untuk mendapatkan informasi, blockchain\_name adalah nama *blockchain* yaitu iotchain, create stream adalah perintah untuk membuat streams, stream\_name adalah pemberian nama streams, “{\”restrict\”:\”write\”}” adalah perintah agar pada streams dapat dilakukan peyimpanan data melalui terminal secara langsung.

* Membuat streams

|  |
| --- |
| {"method":"create","params":["stream","iotdata", {"restrict":"write"}],"id":"37075076-17  05221834","chain\_name":"iotchain"}  **0d1077ce26d418ce0c8ad22c28374e6d26642f784d7d379df2037d08f47059f3** |

Proses membuat *stream* berhasil, kemudian didapatkan nilai *txid* yang akan menjadi nilai tetap dalam *stream*. Nilai *txid* merupakan bukti transaksi setiap melakukan pembuatan fitur pada multichain

### Mencari *List Stream*

|  |
| --- |
| multichain-cli blockchain\_name liststreams |

Pada script diatas adalah untuk mencari *stream* yang telah dibuat dapat menggunakan perintah multichain-cli blockchain\_name liststreams. multichain-cli adalah perintah dasar pada multichain yang digunakan untuk mendapatkan informasi, blockchain\_name adalah nama *blockchain* yaitu iotchain, liststreams adalah perintah untuk mendapatkatkan infomasi streams yang berada didalam *blockchain.*

* List Streams

|  |
| --- |
| [  {  **"name" : "iotdata"**  **"createtxid":"0d1077ce26d418ce0c8ad22c28374e6d26642f784d7d379df2037d08f47059f**  "streamref" : "36-265-4109",  **"restrict" : {**  **"write" : true,**  "read" : false,  "onchain" : false,  "offchain" : false  "details" : {  },  **"subscribed" : false**  }  ] |

*Stream* yang telah di buat dengan nama iotdata dengan parameter createtxid sesuai dengan yang didapatkan pada saat membuat *stream* yang dapat juga dilihat pada saat proses pembuatan *stream*.

### Menyimpan Data pada *Stream*

|  |
| --- |
| multichain-cli blockchain\_name publish stream\_name key\_value data\_value |

Pada script diatas adalah menyimpan data ke dalam *stream* dapat dilakukan dengan menggunakan perintah *publish*, dengan dapat menjalankan perintah multichain-cli blockchain \_name publish stream\_name key\_value data\_value. multichain-cli adalah perintah dasar pada multichain yang digunakan untuk mendapatkan informasi, blockchain\_name adalah nama *blockchain* yaitu iotchain, publish adalah perintah untuk menambahkan data kedalam streams, stream\_name adalah pemberian nama pada streams yaitu iotdata, key\_value adalah pemberian nilai kunci dari data, data\_value adalah nilai data yang akan disimpa kedalah streams.

* Publish data pada streams

|  |
| --- |
| **multichain-cli iotchain publish iotdata key1 "{\"json\":{\"Suhu\":\"31C\", \"Kelembaban\ " :\"50%\"}}"**  {"method":"publish","params":["iotdata", "key1",{"json":{"Suhu":"31C","Kelembaban":"50%"  }}],"id": "29449075-1705221957", "chain\_name":"iotchain"}  **376a6f146c0eea975fd719f698bcef87f5c3ef83c73516f9bbf74893ee3a8fa6** |

Penyimpanan data ke dalam *stream* dilakukan dengan mode *write*, yang dimana dilakukan *input* data secara langsung dengan CLI. Kemudian didapatkan nilai txid dari proses *publish* data tersebut.

### Melihat Detail Data pada *Stream*

|  |
| --- |
| multichain-cli blockchain\_name liststreamitems stream\_name |

Pada script diatas untuk menampilkan detail data pada *stream* dapat menggunakan perintah multichain-cli blockchain\_name liststreamitems stream\_name. multichain-cli adalah perintah dasar pada multichain yang digunakan untuk mendapatkan informasi, blockchain\_name adalah nama *blockchain* yaitu iotchain, liststreamitems adalah perintah untuk mendapatkan detail data didalam stream, stream\_name adalah nama streams yaitu iotdata.

* Gagal melihat data dalam streams

|  |
| --- |
| **multichain-cli iotchain liststreamitems iotdata**  {"method":"liststreamitems","params":["iotdata"], "id": "41440439-1705221995", "chain\_name  ":"iotchain"}  error code: -703  error message:  **Not subscribed to this stream** |

*Node* tidak dapat melihat data pada *stream* karena belum melakukan proses *subscribe* pada *stream* yang telah dibuat.

* Proses subscribe streams

|  |
| --- |
| **multichain-cli totchain subscribe iotdata**  {"method":"subscribe", "params": ["iotdata"], "id": "93920725-1705222017", "chain\_name":"iotchain"} |

Perintah *subscribe* dapat dijalankan dengan multichain-cli blockchain\_name subscribe stream\_name.

* Detail items dalam streams

|  |
| --- |
| **multichain-cli iotchain liststreamitems iotdata true 1**  {"method":"liststreamitems", "params":["iotdata", true, 1], "id": "28473826-1705222062", "cha  in\_name":"iotchain"}  [  {  **"publishers" : ["1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn"**  **"keys" : ["key1"**  "offchain" : false,  "available" : true,  **"data" : {**  **"json" : {**  **"Suhu" : "31C",**  **"Kelembaban" : "50%"**  **"confirmations" : 7,**  **"blockhash" : "008cfe63fe8334c8d864a82528753f4121b3906916e6e873af645726e06ce38d**  **"blockheight" : 44**,  "blockindex" : 1,  "blocktime" : 1705221965,  **"txid":"376a6f146c0eea975fd719f698bcef87f5c3ef83c73516f9bbf74893ee3a8fa6"**,  "vout" : 0,  "valid" : true,  "time" : 1705221957,  "timereceived" : 1705221957  }  ] |

Perintah multichain-cli iotchain liststreamitems iotdata dapat dijalankan kembali, maka didapatkan detail item yang disimpan didalam *stream* beserta nilai txid yang didapatkan sesuai demgan yang diberikan pada saat proses *publish* data.

## Penyimpanan Data *Node Client* pada *Stream*

Penggunaan istilah *node client* pada pengujian yang dilakukan menggambarkan juga *node client* dapat disebut sebagai *permissioned device* karena sudah diberikan *permissions* oleh *node* admin untuk tergabung kedalam jaringan *private blockchain*.

### *Node client* Mencari Detail Item pada *Stream*

|  |
| --- |
| **multichain-cli iotchain liststreamitems iotdata**  {"method":"liststreamitems","params":["iotdata"], "id": "44762638-1705222128", "chain\_name  ":"iotchain"}  error code: -703  **error message:**  **Not subscribed to this stream** |

*Node client* tidak dapat melihat detail *item* pada *stream* karena belum melukan proses *subscribe* pada *stream*.

### *Node client* Gagal Menyimpan Data pada Stream

|  |
| --- |
| **multichain-cli totchain publish totdata key2 "{\"json\":{\"Suhu\":\"**  **32C\", \"Kelembaban\ ":\"40%\"}}"**  {"method":"publish","params":["iotdata","key2",{"json":{"Suhu":"32C","Kelembaban":"40%"}}],"id":"888031051705222168","chain\_name":"iotchain"}  error code: -704  **error message:**  **This wallet contains no addresses with permission to write to this stream and global send permission.** |

*Node client* tidak dapat menyimpan data pada *stream* karena belum memiliki *permission* untuk melakukan *publish* data ke dalam *stream*.

### *Permission* oleh *Node admin*

|  |
| --- |
| **multichain-cli totchain grant 1Ef2MZGZBj7vC36g7bGtmXopDsAnLfpCjuHGeP send**  {"method":"grant","params":["1Ef2MZGZBj7vC36g7bGtmXopDsAnLfpCjuHGeP", "send"], "id":"3477  9544-1705222223","chain\_name":"iotchain"}  **cfd98f9ce492ff718591a48abb33d1b368749dca0a3f36549b1d845c7aba146b**  **multichain-cli iotchain grant 1Ef2MZGZBj7vC36g7bGtmXopDsAnLfpCjuHGeP iotdata.write**  {"method":"grant","params":["1Ef2MZGZBj7vC36g7bGtmXopDsAnLfpCjuHGeP", "iotdata.write"],"  id":"33022464-1705222260","chain\_name":"iotchain"}  **18ef9e88a7dd78b0a9423e502ef79fc34cfe1646cf2d9a0286a8d375a578bc2c** |

Perintah untuk memberikan *permissions* oleh *node admin* dengan menjalankan perintah multichain-cli blockchain\_name grant node\_client\_address send, dimana diberi *permissions* dapat melakukan *send* data ke dalam *stream*. Kemudian pada baris kedua dijalankan perintah multichain-cli blockchain\_name grant node\_client\_address iotdata.write untuk *node client* dapat melaukan *publish* data dengan mode *write* ke dalam *stream* iotdata. Keduanya diikuti dengan nilai *hash* dari proses pemberian izin oleh *node admin*.

### *Node Client* Berhasil Meyimpan Data pada *Stream*

|  |
| --- |
| **multichain-cli iotchain publish iotdata key2 "{\"json\":{\"Suhu\":\"**  **32C\", \ "Kelembaban\ " :\ "40%\"}}"**  {"method":"publish","params":["iotdata","key2", {"json":{"Suhu":"32C","Kelembaban":"40%"}}],"id":"25397248-1705222278", "chain\_name":"iotchain"}  **432650340bc94bce322b070d491368326a0772567d201b69f42782c6bded0dbf** |

*Node client* berhasil menyimpan data pada *stream* setelah diberi *permissions* oleh *node admin*. Kemudian proses peyimpanan data disertai dengan txid diberikan setelah melakukan *publish* data ke dalam *stream*.

### Pencarian Detail Item pada *Stream*

|  |
| --- |
| **multichain-cli iotchain subscribe iotdata**  {"method": "subscribe", "params": ["iotdata"], "id": "10187870-1705222314", "chain\_name":"iot  chain"}  **multichain-cli iotchain liststreamitems iotdata**  {"method":"liststreamitems","params":["iotdata"],"id":"50650521-1705222323","chain\_name":"iotchain"}  [  ]  **"publishers" : ["1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn"**  **"keys" : ["key1"**  "offchain" : false,  "available" : true,  **"data" : {**  **"json" : {**  **"Suhu" : "31C",**  **"Kelembaban" : "50%"**  **}**  },  **"confirmations" : 23**,  "blocktime" : 1705221965,  **"txid":"376a6f146c0eea975fd719f698bcef87f5c3ef83c73516f9bbf74893ee3a8fa6"**  },  {  **"publishers" : ["1Ef2MZGZBj7vC36g7bGtmXopDsAnLfpCjuHGeP"**  **"keys" : ["key2"**  "offchain" : false,  "available" : true,  **"data" : {**  **"json" : {**  **"Suhu" : "32C",**  **"Kelembaban" : "40%"**  **}**  },  **"confirmations" : 3**,  "blocktime" : 1705222279,  **"txid":"432650340bc94bce322b070d491368326a0772567d201b69f42782c6bded0dbf"**  }  ] |

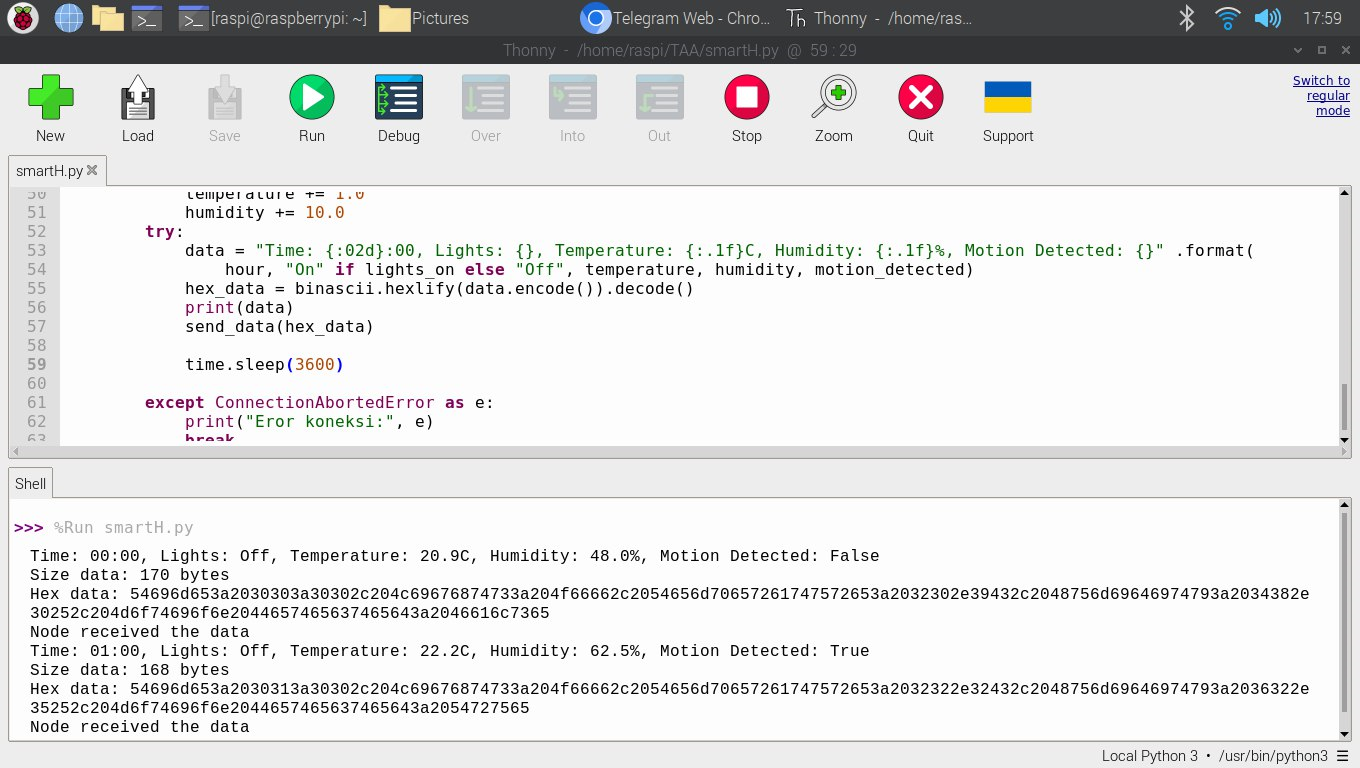
Proses diatas dilakukan terlebih dahulu proses *subscribe* dengan perintah multichain-cli blockchain\_name subcribe iotdata oleh *Node client* agar dapat melihat detail item pada *stream*. Kemudian dilakukan poses mencari detail items dengan perintah multichain-cli blockchain\_name liststreamitems iotdata dan didapatkan dengan key1 adalah data dari hasil *publish* oleh *node admin* dan key2 adalah hasil *publish* oleh *node client*.

## Program Perangkat *Smart Home*

Program simulasi *smart home* dijalankan dari perangkat lomputer, *output* dari simulasi program *smart home* kemudian akan disimpan kedalam jaringan *private blockchain*.

|  |
| --- |
| import socket  import random  import time  import binascii  server\_ip = "192.168.9.20"  server\_port = 4000  addr = (server\_ip, server\_port)  def send\_data(data):      try:          client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)          client\_socket.connect(addr)          size = len(data.encode())          client\_socket.send(data.encode())          print("Bytes size :", size, "Bytes")          print("Hex data:", data)          response = client\_socket.recv(1024).decode("utf-8")          print(response)          client\_socket.close()      except Exception as e:          print("Gagal mengirim data ke server:", str(e))  if \_\_name\_\_ == "\_\_main\_\_":      lights\_on = False      temperature = 20.0      humidity = 50.0      motion\_detected = False      for hour in range(24):          temperature += random.uniform(-1.0, 1.0)          humidity += random.uniform(-5.0, 5.0)          motion\_detected = random.choice([True, False])          if hour >= 6 and hour < 8:              lights\_on = True          elif hour >= 18 and hour < 22:              lights\_on = True          else:              lights\_on = False          if motion\_detected:              temperature += 1.0              humidity += 10.0          try:              data = "Time: {:02d}:00, Lights: {}, Temperature: {:.1f}C, Humidity: {:.1f}%, Motion Detected: {}" .format(                  hour, "On" if lights\_on else "Off", temperature, humidity, motion\_detected)              hex\_data = binascii.hexlify(data.encode()).decode()              print(data)              send\_data(hex\_data)              time.sleep(3600)          except ConnectionAbortedError as e:              print("Eror koneksi:", e)              break |

Pada *script* diatas adalah program simulasi yang bertindak sebagai klien yang dijalankan pada perangkat Raspberry pi. Simulasi program akan terhubung dengan program *server* karena digunakan *socket programming* yang memberi nilai alamat IP dan *Port* dari program *server*. Proses simulasi *smart home* diatur dengan data akan dikirim ke program *server* dengan time.sleep(3600) atau dikirim setiap jam.



Gambar 4.23 Hasil Simulasi Program Smart Home

Pada Gambar 4.23 adalah *output* dari simulasi program *smart home* yang akan dikirim ke program *server* dimana nilai asli dari simulasi diubah terlebih dahulu kedalam format *hexadecimal* agar dapat disimpan oleh program *server* ke dalam multichain.

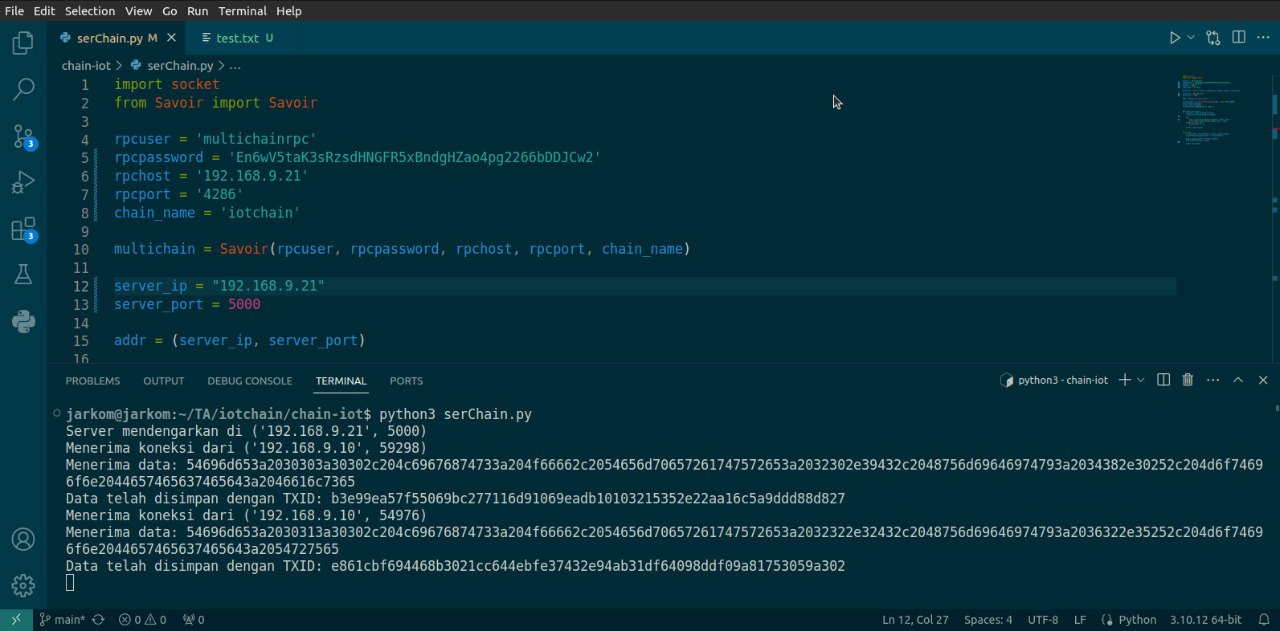
## Program *Server* Untuk Terhubung dengan Multichain

|  |
| --- |
| import socket  from Savoir import Savoir  rpcuser = 'multichainrpc'  rpcpassword = 'hq5xxpMVQB771DZtUhmFZMmkbhrDJv9y9eqHCWXzELB'  rpchost = '192.168.9.20'  rpcport = '7196'  chain\_name = 'iotchain'    multichain = Savoir(rpcuser, rpcpassword, rpchost, rpcport, chain\_name)    server\_ip = "192.168.9.20"  server\_port = 4000  addr = (server\_ip, server\_port)  server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  server\_socket.bind(addr)  server\_socket.listen(5)  print(f"Server mendengarkan di {addr}")  def handle\_recv\_data():      response = 'Node received the data'      client\_socket.send(response.encode())      client\_socket.close()  while True:      client\_socket, client\_address = server\_socket.accept()      print("Menerima koneksi dari", client\_address)      data = client\_socket.recv(1024).decode()      print("Menerima data:", data)      handle\_recv\_data()      try:          txid = mc.publish("iotdata", "key2", data)          print("Data telah disimpan dengan TXID:", txid)      except Exception as e:          print(f"Error: {e}") |

Pada *script* diatas merupakan program *server* yang akan menerima koneksi dari klien untuk mendapat data hasil simulasi hasil program *smart home* kemudian akan dilakukan proses *publish* data ke dalam *stream*.

|  |
| --- |
| from Savoir import Savoir  rpcuser = 'multichainrpc'  rpcpassword = 'hq5xxpMVQB771DZtUhmFZMmkbhrDJv9y9eqHCWXzELB'  rpchost = '192.168.9.20'  rpcport = '7196'  chain\_name = 'iotchain'    multichain = Savoir(rpcuser, rpcpassword, rpchost, rpcport, chain\_name) |

Pada bagian kode diatas adalah konfigurasi program *server* multichain dengan menggunakan API JSON RPC. Konfigurasi program *server* dengan multichain menggunakan *library Savoir* dimana untuk mengambil nilai dari API JSON RPC dari multichain agar program server dapat terhubung dengan multichain.



Gambar 4. 24 Menerima dan Publish Data oleh Server

Pada Gambar 4.24 adalah hasil dari program *server* menerima koneksi dengan program klien dan menerima data hasil simulasi dari program *smart home* yang nilai asli data telah di konversi kedalam format *hexadecimal*, kemudian dilakukan proses *publish* data ke dalam *stream* pada multichain dengan diberi nilai txid data yang tersimpan ke dalam *stream* multichain.

## Pencarian Data pada Stream

|  |
| --- |
| **multichain-clt totchain liststreamitems iotdata false 2**  {"method":"liststreamitems","params":["iotdata", false, 2], "id":"57908529-1705227673", "ch  ain\_name":"iotchain"}  [  **"publishers" : ["1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn"**  **"keys" : ["key3"**  "offchain" : false,  "available" : true,  **"data" : "54696d653a2030303a30302c204c69676874733a204f66662c2054656d70657261747572653a2032302e39432c2048756d69646974793a2034382e30252c204d6f74696f6e2044657465637465643a2646616c7365",**  **"confirmations" : 22**,  "blocktime" : 1705223485,  **"txid":"b3e99ea57f55069bc277116d91069eadb10103215352e22aa16c5a9ddd88d827"**  **"publishers" : ["1BV1dZvbDfowz4ctGKYjFAtj3acdsNQLxJNEkn"**  **"keys" : ["key3"**  "offchain" : false,  "available" : true,  **"data": "54696d653a2030313a30302c204c69676874733a204f66662c2054656d70657261747572653a2032322e32432c2048756d69646974793a2036322e35252c204d6f74696f6e2044657465637465643a2054727565"**  **"confirmations" : 11,**  "blocktime" : 1705227088,  **"txid":"e861cbf694468b3021cc644ebfe37432e94ab31df64098ddf09a81753059a302"**  ], |

Pada Gambar 4.25 adalah pencarian data simulasi *smart home* yang disimpan melalui program *server* yang melakukan *publish* data ke dalam *stream* pada multichain. Nilai data yang sudah dikonversi ke dalam format *hexadecimal* dengan nilai txid yang didapatkan bernilai sama seperti pada Gambar 4.24 dan data diberikan juga keterangan *timestamp*.

## Skenario *Permissionless Device*

*Permissionless device* adalah perangkat yang tidak dapat melakukan aksi yang sama seperti *permissioned device* dan beberapa perilaku *permissionless device* digambarkan seperti berikut:

### *Permissionless Device* Mencoba Terhubung

|  |
| --- |
| **multichaind iotchain@10.26.48.32:4287**  Retrieving blockchain parameters from the seed node 10.26.48.32:4287  Blockchain successfully initialized.  Please ask blockchain admin or user having activate permission to let you connect and/or transact:  multichain-cli iotchain grant 1asswQNAzmAPtwAes7mUE7wdDm27dMJ4uxkJkq connect  multichain-cli iotchain grant 1asswQNAzmAPtwAes7mUE7wdDm27dMJ4uxkJkq connect, send,receive |

*Permissionless device* mencoba terhubung dengan jaringan blockchain dan didapatkan pemberitahuan untuk meminta izin kepada node admin pada saat menjalankan perintah multichaind blockchain\_name@<seed\_ip\_node>:<sedd\_port\_node> .

### *Permissionless Device* Menyimpan Data

|  |
| --- |
| **multichain-cli iotchain publish iotdata key4 "{\"json\":{\"Suhu\":\"32C\", \"Kelembaban\":\"49%\"}}"**  error: Could not connect to the server 127.0.0.1:4286 (error code 1 - "EOF reached")  **Make sure the multichaind server is running and that you are connecting to the correct RPC port.** |

*Permissionless device* mencoba menyimpan data pada multichain dengan menjalankan perintah multichain-cli blockchain\_name publish Streams\_name key\_value data\_value dan didapatkan error: Could not connect to the server karena permissionless tidak dapat terhubung dengan *server* multichain tetapi melakukan penyimpanan data.

### *Permissionless Device* Mencari Detail Items

|  |
| --- |
| **multichain-cli iotchain liststreamitems iotdata true 1**  error: Could not connect to the server 127.0.0.1:4286 (error code 1 - "EOF reached")  **Make sure the multichaind server is running and that you are connecting to the correct RPC port.** |

*Permissionless device* mencoba mendapatkan detail item dari percobaan *permissionless device* saat menyimpan data pada Gambar 4.27 dengan menjalankan perintah multichain-cli blockchain\_name liststreamitems dan didapatkan error: Could not connect to the server karena *permissionless device* tidak dapat terhubung dengan *server* multichain tetapi mencoba melakukan pencarian detail item.

## Evaluasi Kinerja

Evaluasi kinerja dilakukan untuk dapat mengetahui nilai yang dapat dihasilkan oleh blockchain yang diimplementasikan pada proses distribusi data IoT kedalam jaringan private blockchain dengan menggunakan platform multichain dan

# BAB V

# KESIMPULAN DAN SARAN

## Kesimpulan

Berdasarkan penelitian yang usdah dilakukan diperoleh beberapa kesimpulan sebagai berikut:

1. *Blockchain* dapat menjadi salah satu cara untuk meningkatkan keamanan data IoT, dengan dilakukan integrasi *blockchain* dengan IoT data pada sistem IoT menjadi lebih terlindungi dari terjadinya manipulasi karena dengan didukungya oleh *blockchain* yang memiliki struktur sistem yang terdesentralisasi dan terenkripsi.
2. *Blockchain* memberikan integritas data dapat dijaga dengan lebih baik, setiap dilakukan proses transaksi data IoT dapat diverifikasi dan diotorisasi dengan aman dengan adanya hash value yang diberikan setiap melakukan transaksi dan juga bukti *txid* pada setiap dilakukannya penyimpanan data.
3. Menggunakan Jaringan *private blockchain* dimana akses untuk tergabung ke dalam jaringan yang terkendali, yang berarti bahwa pihak-pihak tertentu saja yang memiliki otorisasi yang dapat mengakses dan terlibat dalam jaringan tersebut. Menjadikan tingkat keamanan dan privasi serta memastikan bahwa akses data hanya dibuka bagi *node* yang memiliki izin.

## Saran

Berdasarkan penelitian ini, terdapat beberapa kekurangan dan potensi pengembangan yang dapat menjadi referensi bagi penelitian selanjutnya, diantaranya sebagai berikut:

1. Pada penelitian selanjutnya diharapkan dapat menggunakan platform atau aplikasi *blockchain* yang lebih ampuh dalam menjaga integritas data dan meningkat keamanan data pada lingkungan IoT, serta mempelajari lebih banyak tentang penggunaan efektif dari jaringan *private blockchain*.
2. Pada penelitian selanjutnya diharapkan dalam mendistribusikan data IoT dapat menggunakan mekanisme yang lebih baik dari penelitian ini dan mempelajari lebih dalam tentang *networking* yang terjadi pada saat mendistribusikan data tersebut.

# DAFTAR PUSTAKA

Alam S., R., Jain S., and Doriya R. (2021). *Security threats and solutions to IoT using Blockchain: A review.Proccedings – 5th Internasional Conference on Intelligent Computing and Control System, ICICCS 2021*.

Andersson, Karl et al. *LCN Symposium 2019 : 2019 IEEE 44th Local Computer Networks Symposium on Emerging Topics in Networking : Proceedings : 14 October 2019, Osnabrück, Germany*.

­Dorri, A., Kanhere, S.S., Jurdak, R., Gauravaram, P. (2017). *Blockchain for IoT security dan privacy: The case study of a smart home, IEEE International Conference on Pervasive Computing and Communications 15. 2017 Kona, Hawaii et al. 2017 IEEE International Conference on Pervasive Computing and Communications*.

Efendi, Yoyon. 2018. “Internet Of Things (Iot) Sistem Pengendalian Lampu Menggunakan Raspberry Pi Berbasis Mobile.” *Jurnal Ilmiah Ilmu Komputer* 4(2): 21–27.

Fernando, Erick, Meyliana, and Surjandy. 2019. “Blockchain Technology Implementation in Raspberry Pi for Private Network.” *Proceedings of 2019 4th International Conference on Sustainable Information Engineering and Technology, SIET 2019*: 154–58.

Hayes, A., (2023). *Blockchain* Fact: What is it, how it works, and how it can be used. [Online]. Available: https://www.investopedia.com/terms/b/*blockchain*.asp#what-is-*blockchain*

J. Frankenfield, 2019, “Proof of Work”: Investopedia, 28 june 2020. [online]. Available:https://www.investopedia.com/terms/p/proofwork.asp#:~:text=Proof%20of%20work%20describes%20a,launching%20denial%20of%20service%20attacks.

J. Frankenfield, 2019, “ Proof of Stake” Invetopedia, 11 Agustus 2019. [online]. Available:https://www.investopedia.com/terms/p/proof-of-stake-pos.asp.

Khan, M.A., Salah, K., (2018). IoT Security: Review, *blockchain* solution, and open challenges. Future generation computer system.

Mingyu, H,, Tianyu, K., & Li, G., (2020). A *Blockchain* Based Architecture for IoT Data Sharing Systems. Jurnal Beijing University of Posts and Telecommunications Key Lab of Universal Wireless Communications, Ministry of Education.

Na, D E Conduta, and Crise Hipertensiva. “No 主観的健康感を中心とした在宅高齢者における 健康関連指標に関する共分散構造分析Title.”

Nesterenko R., V., dan Maslova M., A. (2022). Menggunakan Teknologi *Blockchain* Untuk Memastikan Keamanan. Jurnal Nasional Teknologi Komputer Vol. 2: No:1 ; Januari 2022. E-ISSN: 2808-4845; P-ISSN: 2808-7801.

Pahlajani, Sunny, Avinash Kshirsagar, and Vinod Pachghare. “Algorithms.” *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*: 1–6.

Panarello A, Tapas N, Merlino G, Longo F, Puliafito A, 2018, *Blockchain* and IoT integration: A systematic survey. Sensors (Switzerland).Andersson, Karl et al. *LCN Symposium 2019 : 2019 IEEE 44th Local Computer Networks Symposium on Emerging Topics in Networking : Proceedings : 14 October 2019, Osnabrück, Germany*.

Premkumar, R., and Priya S. Sathya. 2021. “A Blockchain Based Framework for IoT Security.” In *Proceedings - 5th International Conference on Computing Methodologies and Communication, ICCMC 2021*, Institute of Electrical and Electronics Engineers Inc., 409–13.

Rumah. (2023). [online]. Apa itu MulthiChain di *Blockchain*. Artikel rumah. Available:https://id.tishijie.com/10609/Apa\_itu\_MultiChain\_di\_blockchan

Seth, S., (2022) Explained Crypto in Cryptocurrence. [online]. Available: https://www.investopedia.com/tech/explaining-crypto-cryptocurrency/

Udin, Ikram et al. 2019. “The Internet of Things: A Review of Enabled Technologies and Future Challenges.” *IEEE Access* 7: 7606–40.