Matter Tunnel

Dongwook Kim
College of Engineering
Hanyang University
Dept.of Information Systems
Seoul, Korea
dongwook1214@gmail.com

Jisu Shin
College of Engineering
Hanyang University
Dept.of Information Systems
Seoul, Korea
sjsz0811@hanyang.ac.kr

Giram Park
College of Engineering
Hanyang University
Dept.of Information Systems
Seoul, Korea
kirammida@hanyang.ac.kr

Seoyoon Jung
College of Engineering
Hanyang University
Dept.of Information Systems
Seoul, Korea
yoooonnn@naver.com

***Abstract*—Our team introduces 'Matter Tunnel', which enables the Matter protocol to operate on a blockchain basis. Matter is a protocol that provides interoperability between IoT devices from various manufacturers, allowing control of multiple brands of IoT devices from a single application. However, due to current network constraints such as NAT and firewalls, a dedicated Matter hub is required when using Matter devices. Matter Tunnel resolves the current limitations of Matter by utilizing blockchain technology, operating as if creating a virtual private network between applications and IoT devices. Primarily, it eliminates the mandatory use of Matter hubs, significantly enhancing user experience and flexibility. This innovation also extends the operational range of Matter devices, allowing them to be placed and controlled beyond the confines of a home. Users can easily manage devices in various environments such as home, workspaces, and vehicles through a single application. From an enterprise perspective, all device interactions and transactions are permanently recorded on the blockchain, providing businesses with reliable and immutable data for tracking device usage patterns and extracting valuable operational insights. Furthermore, Matter Tunnel ensures platform independence, freeing users from being locked into specific ecosystems. It strengthens security by enabling complete end-to-end encryption (E2EE) and enhances user privacy. Additionally, by lowering entry barriers, Matter Tunnel opens up opportunities for small development teams to participate in the Matter ecosystem. With these improvements, users can enjoy a more secure, versatile, and unrestricted IoT experience.**

***Index Terms*—Matter Tunnel, Matter, Blockchain, Matter hub, user experience, E2EE**

TABLE I
ROLE ASSIGNMENTS

Roles	Name	Task description and etc.
Development manager Blockchain Developer	Dongwook Kim	The role involves designing and implementing solutions utilizing blockchain technology. This position sets the technical direction for projects and applies the latest blockchain trends and technologies. Responsibilities include developing smart contracts, optimizing blockchain protocols, managing team members' work, and developing their skills.
User, Customer, Front-end Developer	Jisu Shin	From user and customer perspective, considers what features would enhance the IoT experience and how to improve user interaction. From a development perspective, designs and implements user interfaces for IoT applications, focusing on intuitive and responsive front-end solutions that integrate with Matter protocol and blockchain technology.
User, Customer, Embedded Developer	Giram Park	From a user standpoint, evaluates how IoT devices can better serve everyday needs. As an Embedded developer, works on firmware and software for IoT devices compatible with the Matter protocol and proposed blockchain solution. Focuses on implementing user-centric features that enhance device functionality, improve reliability, and simplify setup processes.
User, Customer, Front-end Developer	Seoyoon Jung	From user and customer perspective, considers what features would enhance the IoT experience and how to improve user interaction. From a development perspective, designs and implements user interfaces for IoT applications, focusing on intuitive and responsive front-end solutions that integrate with Matter protocol and blockchain technology. Aims to create user-friendly interfaces that simplify device control and management.

I. INTRODUCTION

A. Motivation

The rapid advancement of Internet of Things (IoT) technology has led to significant growth in the smart home market. However, compatibility issues among IoT devices from various manufacturers have been a persistent challenge. To address this problem, the Matter protocol was developed, offering an approach that enables control of IoT devices from multiple brands through a single application.

Despite the introduction of the Matter protocol, the current implementation still harbors several crucial issues. These problems prevent the full realization of Matter's original goals: true interoperability, security, and protection of user privacy.

Therefore, we believe a new approach is necessary to overcome these limitations and maximize the potential of the Matter protocol.

B. Problem Statement

The current implementation of the Matter protocol presents the following key issues.

1) Mandatory use of Matter hubs

   Network limitations such as NAT and firewalls make direct P2P communication between IoT devices and applications challenging in typical households. This necessitates the use of dedicated Matter hubs.

2) Limited Operational Range

   As a consequence of the mandatory use of Matter hubs, the operational range of Matter is confined to the home. This limitation restricts the potential for broader IoT applications and remote device management beyond the immediate household environment.

3) Device Functionality Constraints

   Matter protocol defines device types with predetermined functionality sets, limiting the execution of device-specific features. As modern IoT devices and appliances increasingly offer diverse and specialized functions, the protocol's standardized approach may prevent the utilization of unique or innovative device capabilities that fall outside the predefined Matter device types.

4) Platform Dependency

   The need for platform-specific solutions, such as Apple's 'HomeKit' and 'HomePod' or Google's 'Google Home' and 'Nest Hub', results in consumers being locked into particular platforms.

5) Limited End-to-End Encryption (E2EE)

   The advantage of E2EE in the Matter protocol is confined to operation within private networks, failing to ensure complete security throughout the entire communication process.

6) Threats to User Privacy

   The communication structure that routes through cloud services potentially threatens user privacy.

7) Centralized Authentication System

   Matter devices must be certified by a centralized root certificate authority (CA), which makes it difficult for small development teams to participate and limits consumer choices.

These issues hinder the original purposes of the Matter protocol: interoperability, security, and openness. Moreover, the existing centralized server-based IoT architecture presents significant challenges for businesses.

1) Data Reliability and Trust Issues

   The centralized server architecture creates vulnerability in data integrity, making it challenging for businesses to maintain reliable records. Without an immutable ledger system, businesses cannot ensure the authenticity and accuracy of device interaction data, which is crucial for operation analysis and service improvement.

2) Data Analysis and Tracking Problems

   Traditional centralized IoT systems lack comprehensive data tracking mechanisms. This limitation makes it difficult for businesses to effectively analyze device usage patterns, maintain accurate maintenance histories, and extract valuable insights for business optimization. The fragmented nature of data across different platforms and environments further complicates the analysis process.

Therefore, we have determined that a new approach is necessary to overcome these limitations and fully realize the potential of the Matter protocol. We propose a solution utilizing blockchain technology to address these challenges.

C. Research on related technical elements

1) Matter

   We use a variety of IoT devices, and several manufacturers develop and sell IoT devices with different names and appearances. It is Matter that enables these various smart home devices to be connected and managed at once.

   Matter is an IP-based smart home interworking standard that is compatible with all devices, designed to overcome the manufacturer-dependent limitations of smart home devices. It was launched in 2019 by four IoT giants Apple, Amazon, Google, Samsung SmartThings, and the global association CSA, formerly the Zigbee Alliance, and renamed Matter in 2021.

   Matter has the following technical features:

   Unlike existing ZigBee and Z-Wave, Matter operates based on IP protocols. Since Matter is based on IP, which is a network layer protocol, the communication protocol below it does not matter, and eventually all processing is done at the application layer. In other words, the transmission method varies depending on what the application is, but as long as IP is used, the method is not important. Therefore, devices with the Matter logo can work together regardless of brands or supported transmission protocols. In addition, the reason why it is important to use IP is that IP protocols are already proven in the market in terms of interoperability and security.

Matter is interoperable between devices. Matter allows each device to interact using the same protocol, even if it is from a different manufacturer. For example, Samsung Electronics' products have been linked to SmartThings, and LG Electronics only to the ThinQ platform, but now Samsung Electronics' products can be connected to ThinQ. Matter is a very desirable standard from a user's point of view because most homes use a mixture of products from multiple brands.

Matter supports both Wi-Fi and Thread, a low-power mesh network protocol, and supports various network protocols, such as using BLE in the device setting process.

In addition, Matter has the characteristics of Multi-Admin, which uses the same device in conjunction with multiple platforms, AES authentication prescribed by NIST in the United States regarding data encryption, and PKI and certificates for device authentication.

An open ecosystem is being created with the introduction of Matter with these characteristics, and the trend of automation and intelligence of residential environments is spreading through integration with Generative AI technology. Korea is also promoting active efforts to build and expand a smart home ecosystem by preparing support plans in line with global trends. The Korean government is expanding policy support by promoting 'AI@Home', a project centered on Matter and Generative AI, to support the creation of a smart home ecosystem.

However, privacy protection, application of smart home technology of existing houses, and high installation costs are challenges that limit the growth of the market, so it is necessary to proactively prepare countermeasures.

2) Network Constraints in P2P Communication

In a Peer-to-Peer (P2P) structure, there is minimal reliance on always-on infrastructure servers. Instead, the application allows pairs of intermittently connected hosts, called peers, to communicate directly with each other. Peers are desktops and laptops controlled by users rather than owned by service providers, and most peers are located in homes, universities, and offices. Since communication occurs directly between peers without passing through a specific server, this structure is referred to as Peer-to-Peer.

Network constraints in P2P communication, particularly due to NAT (Network Address Translation) and firewalls, negatively impact user experience by introducing additional complexity. NAT, which is used to map private IP addresses to public ones, can prevent peers behind it from being directly accessible from external networks, as it often blocks incoming connections. Firewalls similarly restrict inbound traffic, further complicating direct peer-to-peer communication. To address these issues, NAT traversal techniques such as STUN, TURN, and ICE are commonly employed. Alternatively, users may need to manually configure network settings, such as port forwarding, to bypass NAT and firewall restrictions, allowing for direct communication between peers. These manual configurations can be challenging for users, ultimately affecting the overall user experience.

3) Matter Hub

Matter Hub is a central component of the Matter ecosystem, designed to facilitate seamless communication and interoperability between smart home devices from various manufacturers. Matter aims to unify different smart home technologies, allowing devices to work together regardless of brand.

Smart Home Hubs serve as central controllers for smart home devices, enabling communication between Matter-compatible devices from different manufacturers. Samsung SmartThings and Amazon Echo are representative examples.

Matter Hubs connect Matter devices to the internet and other networks, allowing for remote access and control. Notable examples include Google Nest Hub, which integrates with Google services, and Apple HomePod, which utilizes Siri for voice commands.

While Matter Hubs play a crucial role in enhancing interoperability within the smart home ecosystem, it's important to note that using Matter devices typically requires a home hub. Each application may dictate the specific Matter Hub that must be used, which can strictly lock users into particular platforms. This limitation highlights the need for greater flexibility and broader compatibility in the Matter ecosystem to ensure a truly open and user-friendly IoT environment.

4) Blockchain

To effectively integrate blockchain technology into the IoT industry, it is crucial to consider blockchains with high transaction processing speeds (TPS) and enterprise-friendly features.

Several blockchain platforms stand out for their high TPS capabilities, including:

Solana: Solana is an innovative platform designed for mainstream adoption. The core development team, including co-founder Anatoly Yakovenko, focused on scalability and efficiency based on their experience in building telecommunications networks. By implementing Proof of Stake (PoS) and Proof of History (PoH), they achieved a throughput of up to 65,000 TPS and realized very low transaction fees ($0.00025). It's also highly energy efficient, a single Solana transaction uses 0.00051 kWh.

XRP: XRP (Ripple) uses the RPCA consensus algorithm

and provides approximately 1,500 TPS throughput. Specialized in international remittance, it has established partnerships with many banks and financial institutions, featuring fast transaction completion times of 3-5 seconds.

Hyperledger Fabric: Hyperledger Fabric is an enterprise blockchain platform that provides 2,000-20,000 TPS throughput. Through various network configurations, organizations can adjust the throughput and degree of centralization according to their needs. For example, by modifying parameters such as the number of organizations, ordering service nodes, peer nodes, and channels, administrators can find the optimal balance between performance and decentralization.

Among enterprise-friendly blockchain platforms, the following are noteworthy:

Hyperledger Fabric: Hyperledger Fabric features a modular architecture and permissioned blockchain characteristics, supporting channel-based data partitioning. It has various enterprise use cases including supply chain management, asset tracking, identity management, and healthcare data management.

Quorum: Quorum, developed by JP Morgan, is an enterprise blockchain based on Ethereum, featuring enhanced privacy features and high throughput. It supports private transactions, voting-based consensus mechanisms, role-based access control, and multi-signature contracts.

Hyperledger Besu: Hyperledger Besu is a Java-based blockchain platform compatible with Ethereum. It supports both public and private networks and provides enterprise-grade governance. It is being utilized in various fields including financial services, supply chain management, digital asset management, and inter-enterprise collaboration platforms.

After careful consideration of these options, Hyperledger Fabric is judged as the most suitable blockchain platform for the IoT industry. It offers a combination of high TPS and enterprise-grade features that are essential for large-scale IoT implementations. Furthermore, Hyperledger Fabric is compatible with Monachain, a blockchain platform developed by LG CNS based on Hyperledger Fabric. This compatibility allows for seamless integration and immediate application in existing systems, potentially accelerating adoption and reducing implementation barriers.

5) Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. In the context of Matter IoT, Arduino plays a significant role due to its flexibility, ease of use, and strong community support. When considering Arduino for Matter IoT applications, the following aspects are crucial:

Processing power: Matter protocol requires sufficient computational resources to handle encryption and network communication.

Connectivity options: Wi-Fi or Ethernet capability is essential for Matter, as it's IP-based.

Memory capacity: Adequate RAM and flash memory to run Matter stack and application code.

Power efficiency: For battery-operated IoT devices, low power consumption is critical.

Compatibility with Matter SDK: The board should be capable of running the Matter SDK.

Some Arduino boards suitable for Matter IoT projects include:

Arduino Nano 33 IoT: This compact board features Wi-Fi connectivity and a powerful SAMD21 microcontroller, making it suitable for small Matter devices.

Arduino MKR WiFi 1010: With its low power consumption and robust Wi-Fi capabilities, it's excellent for battery-operated Matter devices.

Arduino Portenta H7: This high-performance board with dual-core processor and multiple connectivity options is ideal for more complex Matter applications.

ESP32-S3: While not an official Arduino board, the ESP32-S3 is widely used in the Arduino ecosystem and offers powerful processing capabilities, Wi-Fi and Bluetooth connectivity, and ample memory.

These boards offer various combinations of processing power, connectivity, and memory, allowing developers to choose the most suitable option for their specific Matter IoT application.

6) Web Assembly

Web Assembly (Wasm) is a binary instruction format designed for efficient execution in web browsers. It serves as a portable target for compilation of high-level languages like C, C++, and Rust, enabling deployment on the web for client and server applications.

Key features and benefits of Web Assembly in the context of IoT and Matter include:

Language Versatility: Web Assembly allows developers to use languages like C, C++, or Rust in web environments. This is particularly beneficial for IoT applications, as these languages are commonly used in embedded systems development.

Performance: Wasm provides near-native performance, making it suitable for computationally intensive tasks often required in IoT applications.

Code Reusability: It enables the use of the same codebase across different platforms - from embedded devices to web interfaces. This is especially valuable for functions

like encryption and decryption, where consistent implementation across platforms is crucial.

Frontend Capabilities: Web Assembly empowers frontend applications to perform complex operations typically associated with backend or embedded environments. This can include data processing, encryption, and other intensive computations directly in the browser.

In the context of Matter Tunnel, Web Assembly can play a significant role in creating consistent, high-performance interfaces for controlling and managing IoT devices across different platforms. It allows developers to implement complex Matter protocol operations in web applications, maintaining consistency with the implementations on the devices themselves. This consistency across platforms is particularly valuable for ensuring that security measures, device interactions, and data handling are uniform across the entire Matter ecosystem.

7) gRPC

gRPC is a modern open source high performance Remote Procedure Call (RPC) framework that can run in any environment. It can efficiently connect services in and across data centers with pluggable support for load balancing, tracing, health checking and authentication.

High Performance and Efficiency: gRPC is designed based on HTTP/2, allowing it to support features like multiplexing, server push, and streaming. These characteristics enable efficient management of connections between multiple clients and servers, minimizing latency and optimizing bandwidth. This ensures that high performance is maintained even in large-scale distributed systems.

Protocol Buffers: gRPC uses Protocol Buffers for service definition. Protocol Buffers is a powerful binary serialization tool developed by Google, allowing the definition of data structures and their conversion into various programming languages. This increases the efficiency of data transmission and maintains consistency in APIs.

Easy Installation and Scalability: gRPC offers simple installation, allowing developers to set up runtime and development environments with just a single command. It also provides scalability that can handle millions of RPCs per second, making it suitable for large-scale applications. This allows developers to quickly build and operate services without complex infrastructure setups.

Cross-Language and Platform Support: gRPC works across multiple programming languages and platforms. It can automatically generate idiomatic client and server stubs for various languages such as Java, C++, Python, Go, and Ruby. This facilitates collaboration between teams using different tech stacks and enhances code reusability.

Bi-Directional Streaming: gRPC supports bi-directional streaming between clients and servers. This allows clients to send data to the server while simultaneously receiving streamed data from the server. This feature is particularly useful in applications that require real-time data transmission.

Integrated Authentication and Security: gRPC enhances security by integrating full pluggable authentication features at the HTTP/2 transport layer. This simplifies the implementation of user authentication and authorization, ensuring safe transmission of sensitive data.

Due to these features, gRPC is widely used in various fields, including microservices architecture, IoT applications, and mobile backend services. By using gRPC, developers can build efficient and scalable systems, making it easier to manage communication between services.

D. Research on related study

1) Benefits of Blockchain for Data Mining

The integration of blockchain technology and data mining techniques for anomaly detection provides efficient methods through their combined application. Data stored on the blockchain can be treated as big data, and data mining techniques enable the extraction of hidden patterns and insights.

This paper explores analytical approaches to blockchain data and practical applications, demonstrating how these technologies enhance anomaly detection and fraud prevention. Blockchain's transparent recording system facilitates data tracking and monitoring. It serves as an effective tool for corporate data analysis, with a key advantage being rapid detection of operational changes or fraudulent activities.

Key applications from the literature include:

a) Analysis of Bitcoin Transaction Networks

Zola et al. (2019) analyzed changes in Bitcoin transaction patterns by utilizing the time-series data of the blockchain. They used data from WalletExplorer and the Bitcoin mainnet over the past three years, calculating F1 scores through k-fold cross-testing. By analyzing the transaction data linked in chronological order on the blockchain, it is possible to detect cybersecurity threats and identify changes in behavioral patterns.

b) Blockchain Data Collection and Analysis

Brinckman et al. (2019) presented techniques for crawling, collecting, and analyzing blockchain data. They demonstrated a method for clustering transactions and extracting account characteristics to identify fraudulent accounts, which serves as a good example of understanding the data structure of the blockchain and effectively analyzing it.

c) Time-Series Transaction Data Analysis

Zhao et al. (2021) analyzed the entire dataset of the Ethereum blockchain from a temporal perspective. They utilized the ethereum blockchain dataset from the Bigquery Public Data Repository to examine changes in transaction patterns over time, comparing the accuracy of Random Forest and Logistic Regression, and visualizing the temporal evaluation of the collected data.

These application examples demonstrate that effective analysis is possible by leveraging the connected data structure and temporal characteristics of the blockchain. In particular, the data structure of the blockchain can be effectively utilized to identify patterns in sensitive transaction data and detect anomalous behaviors, which can be considered a significant advantage of blockchain-based data analysis.

2) Benefits of Blockchain for Data Integrity and Accessibility

Blockchain technology has emerged as an innovative solution in healthcare data management, addressing important challenges in data security, integrity, and interoperability. Here are three representative implementations that demonstrate the applications of blockchain in healthcare:

a) MedRec

MedRec is a blockchain project in healthcare that enables comprehensive management of medical data, including data provision by medical institutions, patient licensing, and data utilization by research institutes. MedRec 2.0 is implemented using Go-ethereum and Solidity languages, utilizing smart contracts on the Ethereum blockchain for intermediary-free data exchange. Like other blockchains, MedRec ensures security through its blockchain nature. Due to decentralization, data is maintained across all network nodes and stored in nodes of patients and their service providers. The blockchain's consensus mechanism prevents security issues from single-point vulnerabilities. Additionally, if one node attempts to modify a specific transaction in a block, that modified node becomes inconsistent with others and is excluded from consensus, maintaining record integrity.

b) Estonian e-Health

The Estonian e-Health Foundation and Guardtime have strengthened security and patient monitoring by implementing KSI blockchain technology. The healthcare system integrates medical service data through X-Road, a data exchange platform, enabling comprehensive medical service data analysis. This system provides personal healthcare data integrity verification, with healthcare providers transmitting data integrity to the KSI server for permanent blockchain and offline recording. Through the "e-prescription service," doctors and pharmacies can verify prescription integrity. Additionally, the "electronic health registration service" allows doctors to access medical images like X-rays using only the patient's ID code. These systems enable secure storage, efficient sharing, and integrity verification of medical data.

c) Mediblock

Mediblock is a blockchain-based integrated management platform for personal medical data that aims to integrate distributed patient medical data. Mediblock secures data by granting patients access to medical data and allowing only them to decrypt the entire data, while recording data hash values on the blockchain for integrity. It improves data reliability by restricting medical record creation to certified medical personnel, and ensures transparency by recording data access information and rights on the blockchain. Additionally, it enables safe data sharing through relay services and secondary backup storage, while facilitating secure data transactions through an encrypted data trading market within the platform.

These implementations demonstrate how blockchain technology enhances healthcare data management through improved integrity, security, accessibility, and transparency. These advantages can be applied to blockchain-based Matter tunnels, potentially replacing Matter hubs to enhance the reliability and efficiency of the Matter protocol.

3) Hyperledger Fabric Performance

The performance of Hyperledger Fabric is often questioned, particularly regarding transaction processing speeds (TPS) in comparison to other blockchain platforms. The performance of a Fabric network is complex due to the involvement of multiple organizations with varying hardware and networking infrastructures, as well as factors such as the number of channels and chaincode implementation.

Hyperledger Fabric 2.x has introduced performance improvements over version1.4, which is no longer in long-term support (LTS). Fabric 2.5 is the latest LTS version and includes a new peer gateway service. This service, along with the new gateway SDK, is expected to enhance the performance of applications.

a) Hardware and Topology

In Hyperledger Fabric, the topology used consists of two peer organizations (PeerOrg0 and PeerOrg1), each with one peer node, along with a single ordering service organization (OrdererOrg0) that utilizes Raft consensus with one ordering service node. TLS is

enabled on each node.

All nodes utilized the same hardware configuration:

- Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz

- 40 Cores made up of 2 CPUs. Each CPU has 10 physical cores supporting 20 Threads in total

- 64Gb Samsung 2933Mhx Memory

- MegaRAID Tri-Mode SAS3516 (MR9461-16i) disk controller

- Intel 730 and DC S35x0/3610/3700 Series SSD attached to disk controller

- Ethernet Controller X710/X557-AT 10GBASE-T

- Ubuntu 20.04

All machines were connected to the same switch. Hyperledger Fabric was deployed natively across three physical machines, meaning that the native binaries were installed and executed without using container technologies such as Docker or Kubernetes.

b) Hyperledger Fabric Application Configuration

State Database: LevelDB was used as the state database.

Gateway Service: The concurrency limit was set to 20,000.

Application Channel: A single application channel was created with two peers and an orderer, configured to use V1_4 capabilities for lifecycle deployment while other capabilities were set to V2_0.

Configuration: Only the application channel existed, no private data was used, and default policies applied. No range or JSON queries were conducted, and the network was TLS-enabled without mutual TLS.

Chaincode: Go chaincode (fixed-asset-base) was deployed without the Contract API, and an endorsement policy of "1 Of Any" was specified.

c) Load Generator

Hyperledger Caliper 0.5.0 served as both the load generator and for generating report outputs. It was configured to work with Fabric 2.4, utilizing the peer Gateway Service to initiate and assess transactions, with all transactions originating from a single organization directed to its gateway peer. The load was defined based on the fixed-asset benchmark from Hyperledger Caliper-Benchmarks.

Four bare-metal machines were employed to host remote Caliper workers and a single Caliper manager, which was responsible for generating the load on the Hyperledger Fabric network. To create sufficient workload on the Fabric network, multiple Caliper workers were necessary, corresponding to the number of clients currently connected to the network. The results section includes details on the number of Caliper workers utilized.

d) Key Points

The results presented here were generated using the latest builds of Hyperledger Fabric 2.5, employing the default node and channel configuration values. The block cutting parameters used were as follows:

Block Cut Time: 2 seconds

Block Size: 500

Preferred Maximum Bytes: 2 MB

To avoid hitting concurrency limits while pushing enough workload through, the gateway concurrency limit was set to 20,000.

Only a single channel was utilized, meaning the peer did not leverage its full resource potential.

The chaincode was optimized for these tests, and real-world chaincode is expected to perform less efficiently.

The Caliper workload generator was also optimized for transaction throughput, whereas real-world applications would involve client implementations that may introduce latency.

Transactions were sent to a single gateway peer from the same organization; real-world scenarios would likely involve multiple organizations sending transactions concurrently, potentially leading to higher TPS results.

Using the gateway service allowed for better performance as blocks were not received via the delivery service to confirm transaction completion, enhancing both client and network performance compared to the legacy node SDK.

e) Result

TABLE II
HYPERLEDGER FABRIC BENCHMARK RESULT

Name	Max Latency	Average Latency	Throughput
create asset 100	2.13	0.33	2946.7
create asset 1000	3.21	1.52	2938.9
read write assets previously read 100	0.21	0.06	2544.3
read write assets previously read 1000	0.26	0.11	1527.0

i) Blind Write of a Single Key 100 Byte Asset Size

Caliper test configuration:

- workers: 200

- fixed-tps : 3000

The TPS value represents the peak performance achieved during the tests. Attempts to exceed this throughput resulted in unexpected failures and a decrease in overall throughput.

ii) Blind Write of a Single key 1000 Byte Asset Size

Caliper test configuration:

- workers: 200

- fixed-tps : 3000

The throughput remains the same as that observed with the 100-byte blind write benchmark; however, latency increases.

iii) Read Write of a Single Key 100 Byte Asset Size

Caliper test configuration:

- workers: 200

- fixed-tps, tps: 2550

The above results were achieved under the expectation of no failures. The latency remained very low, indicating that the Fabric network was reaching its capacity during this test.

iv) Read Write of a Single Key 1000 Byte Asset Size

Caliper test configuration:

- workers: 200

- fixed-tps, tps: 1530

The above results were achieved under the expectation of no failures. The latency remained very low, indicating that the Fabric network was reaching its capacity during this test.

II. REQUIREMENTS

A. Core Requirements

The solution proposed in this study must meet the following key requirements

1) Eliminate the mandatory use of Matter Hubs

It must eliminate the mandatory use of Matter Hubs, enabling direct and secure device-to-application communication without relying on intermediary hardware. This removal of hub dependency not only reduces system complexity and cost but also enhances system reliability by eliminating single points of failure.

2) Extended Operational Range

The solution should overcome the limitation of traditional Matter Hub-based systems, which confine device operation to a home network. It must enable secure and efficient management and control of IoT devices from remote locations, expanding the utility of Matter-compatible devices beyond the immediate household environment. This extended range should not compromise security or user privacy.

3) Enhanced Device Functionality Support

The solution must provide mechanisms to support diverse and specialized device functionalities beyond Matter's predefined device types. It should implement an expandable approach that allows manufacturers to define and integrate custom device capabilities. This enhancement enables the full utilization of modern IoT devices' innovative features without being constrained by standardized device type limitations, fostering technological advancement and product differentiation in the IoT ecosystem.

4) Enhanced data tracking mechanisms

The solution must incorporate enhanced data tracking mechanisms that provide comprehensive visibility into device operations, interactions. This tracking system should maintain tamper-proof records of all device activities, enabling businesses to analyze usage patterns, monitor performance metrics, and optimize their operations effectively. The implementation should support both real-time monitoring and historical data analysis while maintaining user privacy and data security.

5) Improved system reliability

To ensure system reliability and trust, all data interactions and transactions must be recorded on an immutable ledger, creating a verifiable and transparent history of

device operations. This trustworthy data foundation is crucial for both operational intelligence and regulatory compliance, allowing businesses to make data-driven decisions with confidence. The system should provide mechanisms for data verification and validation while maintaining appropriate access controls and privacy measures.

6) Decentralization

The solution should embrace decentralization by removing dependencies on centralized certificate authorities (CAs) and platform-specific ecosystems. This decentralization should establish a more democratic and open IoT ecosystem where small developers and manufacturers can participate freely, fostering innovation and competition. Furthermore, the solution allow IoT devices from various manufacturers to interact seamlessly.

7) Enhanced End-to-End Encryption (E2EE)

E2EE should be guaranteed even outside private networks, maintaining data confidentiality throughout the entire communication process.

8) User Privacy Protection

It should reduce reliance on centralized cloud services for communication and minimize the collection and use of user data while managing it transparently.

9) Real-time Performance

The solution must support real-time communication and responsiveness, even when utilizing blockchain technology. It should ensure that blockchain integration does not introduce significant latency or delays in device interactions. The system should maintain quick response times for user commands and device state updates, while leveraging the benefits of blockchain for enhanced security and decentralization.

By meeting these requirements, the proposed solution is expected to overcome the limitations of the current Matter hub-based Matter protocol and provide a better user experience, security, and privacy.

B. Development Requirements

Client

1) User Authentication

The login system should prioritize security, simplicity, and compatibility with the Matter protocol. To achieve this, we propose implementing a login mechanism based on asymmetric cryptography, specifically using the secp256k1 elliptic curve algorithm, which is also employed by Matter. Users can easily log in by entering their secp256k1 private key instead of using social login methods.

a) Sign Up

Users can initiate the registration process by clicking the Sign Up button on the login page of the application. During the sign-up process, users will create their private key, which will serve as their unique identifier for logging in. Generated private key will be securely stored in local storage. If desired, users can retrieve their private key at any time from the application's account management section. This feature allows users to back up or transfer their private key to another device if needed.

b) Login

Users can log in by entering their private key. Upon successful login, a public key is derived from the private key, and users are directed to a page where they can register Matter devices. If the entered key doesn't match the required format, an error message starting "Your key is incorrectly formatted" will be displayed.

2) Add Device

To register a Matter-compatible device, the user clicks the '+' button to add a new device. User can scan the QR code or enter the setup code provided by the device to proceed with the registration. The device information will be stored in the local storage. Once the device is successfully registered, the user gets registration confirmation message and will be directed to a screen displaying the device status and features.

3) Remove Device

To remove an unnecessary device, the user select the device and click the 'Remove' button. The user will be prompted to confirm choice before the device is removed from the system. Upon confirming the removal, the system will delete the device from the local storage and the user will receive a message confirming the successful deregistration. If an error occurs during the process, an error message will be provided.

4) Device Control

A user-friendly interface will be designed for controlling each device, focusing on intuitive navigation and clear functionality. The interface will display available control options. When the user issues a command to control a device, the command will be executed through communication with the blockchain and the device. Feedback will be provided to the user upon successsful command execution. User can set devices to operate automatically based on specific time or conditions. The application will allow users to configure and manage their automations.

5) Data Display

Matter devices transmit a variety of signals to the application through the Matter Tunnel. These signals are structured in various formats to accommodate diverse data types and use cases. The formats include, but are not limited to, JSON, binary data. Each format serves a specific purpose, allowing for flexible and efficient data transmission.

The application is responsible for processing these diverse signals and presenting them to users in a manner that aligns with their respective data formats. Upon receiving the signals, the application will parse and interpret the data to ensure it is accurately represented. The transformed data will then be displayed in a user-friendly and intuitive interface that enhances the overall user experience.

The application will be designed to automatically update the user interface in real-time, reflecting any changes in the device status or incoming data. This ensures that users have access to the most current information available, allowing for informed decision-making and timely responses.

By supporting various signal formats and providing a clear, interactive display, the application aims to enhance the usability and effectiveness of Matter devices within the connected ecosystem.

Dashboard

1) Integrated Dashboard

The Integrated Dashboard provides a unified view of the status of all devices and the network. Through visual representations of real-time device statuses, transaction logs, and key metrics, users can monitor and understand system performance at a glance, enabling quick situation assessment.

2) Natural Language Query System

The Natural Language Query System supports intuitive natural language input for querying blockchain data. The AI model converts user questions into precise data query commands, allowing users without programming knowledge to access and analyze blockchain data. This interface enhances accessibility, making data analysis easy for both technical and non-technical users.

3) Data Visualization and Analysis Tools

Data Visualization and Analysis Tools improve the platform's usability by presenting blockchain data results in accessible visual formats. Charts, graphs, and other visuals provide clear insights, while advanced analytics functionalities let users analyze transaction logs, and real-time performance indicators (KPIs) to generate meaningful insights for decision-making.

4) Insight and Report Service

The Insight and Report Service features automatically generates daily, weekly, and monthly operational reports. These reports allow decision-makers to track performance trends, identify areas needing attention, and access actionable insights and recommendations, supporting effective planning and strategic problem-solving.

III. Development environment

A. *software development platform*

1) JavaScript

Fig. 1. JavaScript

JavaScript is a programming language used to make web pages dynamic, allowing for content changes in response to user interactions. It evolved from historically static web pages and is now utilized in both client-side and server-side development, with various libraries and frameworks expanding its functionality. JavaScript is interpreted by the browser, modifying the DOM in response to user events on the client side and generating dynamic content by interacting with databases on the server side. Additionally, when combined with HTML and CSS, it enhances the UI of web applications and allows for efficient task execution. As a client-side scripting language, JavaScript is one of the core technologies of the World Wide Web. Its features can improve the user experience of websites, from refreshing social media feeds to animations and interactive map displays. For example, when browsing the internet, if you encounter an image slideshow, a dropdown menu that appears upon clicking, or dynamic color changes of objects on a webpage, you are witnessing the effects of JavaScript in action. Its selection for this project is driven by the team's familiarity with it, which enhances efficiency and productivity.

2) React

Fig. 2. React

React is a JavaScript library developed by Facebook, primarily used for building user interfaces (UI). It has a component-based structure, allowing developers to create reusable components for UI construction. React uses a Virtual DOM to efficiently handle updates and optimize performance, making it widely used in the front-end development of web applications. The decision to use React for this project was influenced by the fact that the team conducted a study on React together during their vacation, enhancing their familiarity and readiness to implement it effectively.

3) Electron

Fig. 3. Electron

Electron is a framework for building desktop applications using JavaScript, HTML, and CSS. By embedding Chromium and Node.js into its binary, Electron enables developers to maintain a single JavaScript codebase and create cross-platform applications that work on Windows, macOS, and Linux. Popular desktop applications like Slack and Visual Studio Code are developed using Electron. Implementing the dashboard as a desktop application is advantageous for local use, particularly when want to use AI function in local, which makes Electron a suitable choice for this project. This decision reflects the need for a robust and efficient local application to meet the project's requirements.

4) Go

Fig. 4. Go

Go (or Golang) is an open-source programming language developed by Google, designed to be fast and concise while supporting concurrency, making it ideal for network applications and server-side programming. Its straightforward syntax and ease of error handling contribute to its popularity in projects that require high performance and efficiency. Additionally, Go boasts a large ecosystem of partners, communities, and tools, making it easy to learn and fostering effective team collaboration. The decision to use Go for this project is influenced by the requirement that Hyperledger Fabric must be developed using Go, ensuring compatibility and optimal performance within the blockchain framework.

5) gRPC

Fig. 5. gRPC

gRPC is a high-performance, open-source Remote Procedure Call (RPC) framework initially developed by Google. It uses HTTP/2 for transport and Protocol Buffers as the interface description language, enabling efficient communication between distributed systems across different languages and platforms. gRPC excels in scenarios requiring high-throughput and low-latency communication, making it particularly suitable for microservices architectures. The framework's strong typing system, bidirectional streaming capabilities, and built-in support for authentication enhance the reliability and security of service-to-service communication. The decision to use gRPC for this project was influenced by its essential role in communicating with the Hyperledger Fabric gateway. Since the Electron-based frontend needs to interact with the Hyperledger Fabric network through its gateway, gRPC provides the necessary protocol and tools to establish this communication efficiently and securely.

6) C++

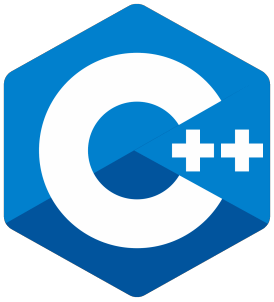

Fig. 6. C++

C++ is a high-performance, object-oriented programming language developed by Bjarne Stroustrup as an extension of the C language. It is widely used in various applications, including game engines, system software, IoT, embedded systems, and graphics processing. C++ provides a clear program structure and enables code reuse, which helps reduce development costs, while also offering portability for creating applications that can adapt to multiple platforms. Furthermore, C++ offers a high level of control over system resources and memory. This project leverages C++ for building Arduino and WebAssembly applications, capitalizing on its efficiency and versatility in these domains.

7) Arduino

Fig. 7. Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software, primarily used in IoT and embedded systems projects. Arduino boards can read inputs—such as light from a sensor, a finger press on a button, or a Twitter message—and turn them into outputs, like activating a motor, lighting an LED, or publishing data online. Programmed using C/C++, Arduino enables rapid prototyping by connecting various sensors and actuators, and is also popular for educational and DIY projects. Users can control their boards by sending instructions to the microcontroller, allowing for flexible and dynamic applications. Arduino was chosen for this project because it offers a simple way to develop embedded systems, streamlining the development process and enhancing efficiency.

8) Python

Fig. 8. Python

Python is a high-level programming language with concise and easy-to-read syntax. It is used in various fields, including data science, artificial intelligence, web development, automation, and scripting. Thanks to its rich libraries and community support, Python enables rapid prototyping and highly productive development. In this project, Python is chosen specifically for AI development, leveraging its capabilities to create efficient and effective AI solutions.

9) Visual Studio Code

Fig. 9. VS Code

Visual Studio Code is a code editor redefined and optimized for building and debugging modern web and cloud applications. Developed by Microsoft, it is a free and open-source editor that supports a wide range of programming languages, including JavaScript, Python, and C++. With features like extensive extensibility through a marketplace of plugins, built-in debugging tools, and seamless integration with version control systems like Git, VS Code provides a user-friendly interface that enhances productivity. Additionally, it is available on multiple platforms, including Windows, macOS, and Linux, making it accessible to developers regardless of their operating system. Visual Studio Code was chosen for this project because it is the most commonly used code editor, offering familiarity and reliability for efficient development

10) GoLand

Fig. 10. GoLand

GoLand is an integrated development environment (IDE) specifically designed for the Go programming language, developed by JetBrains. It offers smart code assistance with advanced code completion, navigation, and refactoring tools, making it easier to write clean and efficient code. GoLand features a powerful integrated debugger for setting breakpoints and inspecting variables, as well as built-in support for unit testing and code coverage analysis. Additionally, it integrates seamlessly with version control systems like Git, allowing developers to manage their repositories directly within the IDE. With a customizable interface and cross-platform compatibility for Windows, macOS, and Linux, GoLand enhances the development experience, enabling developers to write, test, and deploy Go applications more effectively. GoLand was chosen for this project to facilitate the use of the Go programming language, providing the necessary tools and environment for optimal development.

11) LaTeX

Fig. 11. LaTeX

LaTeX is a high-quality typesetting system. It includes features designed for the production of technical and scientific documentation. LaTeX is the de facto standard for the communication and publication of scientific documents.

12) GitHub

Fig. 12. GitHub

GitHub is a web-based platform that uses Git version control for managing and sharing code repositories. It enables developers to collaborate on projects by allowing them to track changes, manage branches, and resolve conflicts seamlessly. With features like pull requests, code reviews, and issue tracking, GitHub facilitates efficient team collaboration and project management. Additionally, it hosts a vast repository of open-source projects, providing developers with resources to learn from and contribute to. GitHub's integration with various CI/CD tools and support for GitHub Actions enhances its capabilities, making it an essential tool for modern software development.

13) Notion

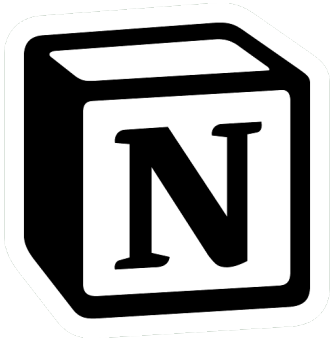

Fig. 13. Notion

Notion is an all-in-one workspace that combines note-taking, task management, databases, and collaboration tools, allowing teams to organize and share information effectively. With its flexible structure, users can create custom templates and pages tailored to their specific needs, promoting productivity and collaboration. Notion's rich formatting options, including tables, kanban boards, and calendars, enable users to visualize and manage their work dynamically. Additionally, its real-time collaboration features allow multiple users to edit and comment simultaneously, making it a powerful tool for project management and team communication.

14) macOS

Fig. 14. macOS

macOS is a widely used operating system for software development, known for its user-friendly interface and exceptional versatility. It equips developers with essential tools and integrated development environments (IDEs) for creating a variety of applications, including web, desktop, mobile, and gaming software. The platform supports multiple programming languages and frameworks, offering the flexibility to adapt to specific project requirements. Its intuitive design simplifies the setup of development environments and project management. Additionally, an active macOS developer community fosters collaboration and knowledge sharing. With continuous updates, developers have access to the latest technologies and tools, enabling them to modernize their applications effectively. Overall, macOS is recognized as a crucial platform for software development, playing a significant role in turning innovative ideas into reality.

B. Computer resources

TABLE III
COMPUTER RESOURCES

Name	Computer Resource	Version of OS, SW
Dongwook Kim	Apple M3 Pro Chip 18GB RAM memory	macOS Sequoia 15.0.1
Jisu Shin	Apple M1 Chip 16GB RAM memory	macOS Sequoia 15.0.1
Giram park	Apple M2 Chip 16GB RAM memory	macOS Sequoia 15.0.1
Seoyoon Jung	Apple M2 Chip 8GB RAM memory	macOS Sequoia 15.0.1

C. Cost Estimation

Although it is different from the actual application in the industry, we envision a test network operating two Hyperledger Fabric peers and four orderers on a single computer. For this configuration, we plan to use an AWS EC2 t2.medium instance (2vCPU, 4GB RAM). This t2.medium instance is deemed suitable for a test network of this scale, as it meets the minimum specifications required for running peers and orderer while providing a cost-effective option for development and testing purposes. According to the AWS pricing calculator, operating a t2.medium instance in the Seoul region with a long-term commitment would cost approximately $18.47 per month. Adding the cost of a required 50GB EBS volume at approximately $4.56, the total estimated monthly cost would be $23.03. This represents the minimum cost for establishing a test environment, and an actual production environment would likely require higher-specification instances and additional infrastructure configuration. However, if we can utilize the company's existing underutilized server resources, we expect to significantly reduce these cloud cost.

D. Software in use

1) ADEPT

ADEPT, which stands for Autonomous Decentralized Peer-to-Peer Telemetry, is a blockchain platform designed for IoT devices that operates on a peer-to-peer basis. The idea behind ADEPT was introduced in 2015 as a result of collaboration between Samsung and IBM. It incorporates technologies like BitTorrent, Telehash, and Ethereum. ADEPT organizes IoT devices based on their capacities, allowing them to independently manage, analyze, and share their data. This platform is being implemented in wearable technology and household appliances. For instance, Samsung's smart washing machine employs ADEPT technology to automatically order essential supplies, such as detergent, when they are running low.

2) IoT Chain

IoT Chain operates on blockchain technology and incorporates various mechanisms like PBFT (Practical Byzantine Fault Tolerance), DAG (Directed Acyclic Graph), SPV (Simple Payment Verification), and CPS (Cyber Physical System). Its primary goal is to improve security within the IoT ecosystem while utilizing ICT (IoT Chain Token) for accessing IoT products. By leveraging the decentralized security of conventional blockchains, IoT Chain overcomes challenges related to transaction speed and scalability through PBFT and DAG. The architecture consists of a main chain and a side chain; the side chain executes smart contracts using coins generated from the main chain. The main chain employs PBFT for rapid transaction validation, while the side chain utilizes DAG for efficient transaction processing. SPV allows payment verification by checking only the headers of blocks, rather than all their components, which reduces verification fees and decreases user overhead. IoT Chain finds applications in shared economies and smart home technologies. In November 2018, initiatives were launched to create a developer ecosystem, with plans to publicly release IoT Chain in December.

3) SLOCK.IT

SLOCK.IT, a startup based in Germany, focuses on creating a sharing economy infrastructure utilizing Ethereum

technology. They are in the process of developing the Universal Sharing Network, which integrates an automated payment system with Ethereum. This platform allows individuals to share and trade unused resources like homes or cars via blockchain technology, ensuring trust between parties. SLOCK.IT provides a smart lock feature, allowing users to unlock their assets for others by paying with tokens to execute Ethereum smart contracts. Additionally, users can control the keys required for transactions through a mobile application.

4) JD.COM

JD.com offers blockchain gateway services, blockchain node services, and blockchain consensus network services. The platform utilizes a BFT-like consensus algorithm and employs an authentication protocol to manage the number of accesses to the blockchain network. The system consists of three types of peers: consensus peers, gateway peers, and IoT devices. Gateway peers operate within the middleware layer to integrate inputs and protocols from the lower layers.

E. Task distribution

TABLE IV
TASK DISTRIBUTION

Name	Task
Dongwook Kim	Blockchain Development
Jisu Shin	Front-end Development
Giram Park	Embedded Development
Seoyoon Jung	Front-end Development

IV. SPECIFICATIONS

A. Core Requirements Specifications

1) Hub Elimination and Range Extension

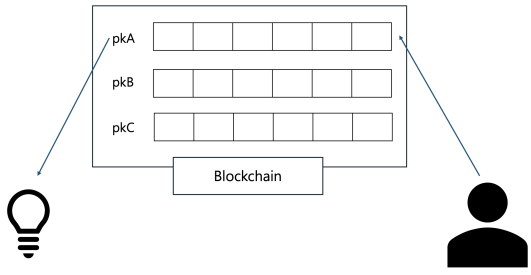

Fig. 15. Blockchain Queue

To eliminate the mandatory use of Matter Hubs and extend operational range, we propose replacing traditional Matter hubs and cloud services with Hyperledger Fabric's chaincode functionality. This transformation fundamentally changes how Matter devices communicate and operate, freeing them from the physical constraints of home networks.

The core of this solution lies in implementing a message queue system within the blockchain. Instead of relying on a physical hub for communication, each Matter device interacts with a dedicated queue in the blockchain. This queue serves as a virtual communication channel, enabling devices to operate beyond the traditional home network boundaries.

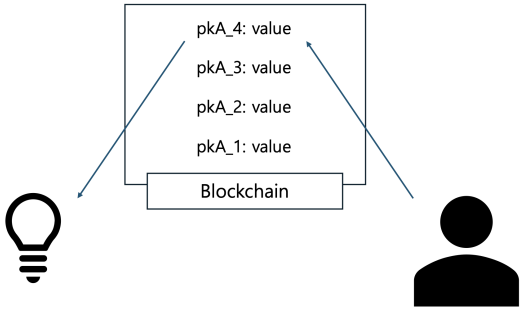

Fig. 16. KeyValue based Blockchain Queue

However, implementing a traditional queue structure in Hyperledger Fabric would be inefficient due to the complexity of transaction operations. Reading and writing to an array-based queue would require reading the entire array for each push operation, creating unnecessary overhead. To optimize this process, we propose using a Key-Value Store structure where the key is formatted as *devicePK-index* (device public key combined with a index) and the value contains the message payload.

This architecture provides several advantages:

a) Eliminates the need for physical Matter hubs by virtualizing their functionality through blockchain

b) Enables device operation from any location with internet connectivity

c) Maintains secure and reliable communication through blockchain's inherent security features

d) Optimizes performance through efficient key-value based message handling

2) Protocol Flexibility and QR Code Innovation

To support diverse device functionalities and provide greater flexibility, we innovate the protocol and QR code structure. Instead of relying on predefined function clusters in Matter protocol, our solution implements function definitions within QR codes, enabling dynamic functionality support.

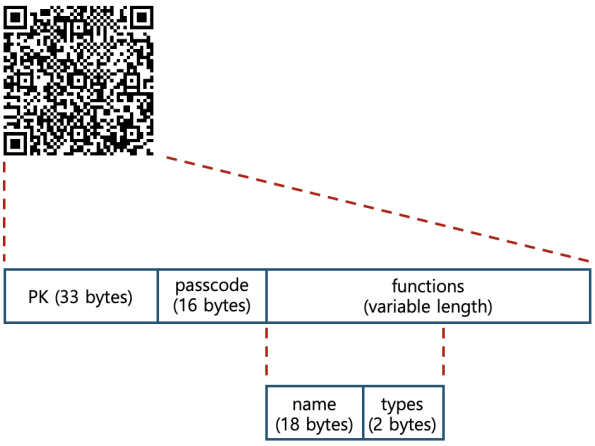

Fig. 17. Matter Tunnel QR Code

The QR code structure consists of three main components: a 33-byte public key (PK), a 16-byte passcode, and function definitions. Each function definition comprises an 18-byte function name and 2 bytes representing parameter and return value types. Matter Tunnel supports four basic types - void, string, number, and boolean - each represented by 2 bits. This efficient encoding allows for up to seven parameters and one return value within the 2-byte type definition.

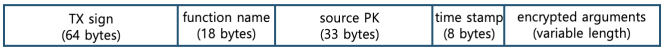

Fig. 18. Matter Tunnel TX Format

The Matter Tunnel transaction format similarly reflects this flexibility, consisting of a 64-byte signature, 18-byte function name, 33-byte public key, 8-byte timestamp, and encrypted arguments. By including function names directly in the transaction format rather than using opcodes, the solution simplifies data analysis while maintaining extensibility. This approach effectively eliminates the constraints of Matter's predefined device types, allowing manufacturers to implement custom functionalities while ensuring seamless integration with the Matter ecosystem.

3) Data Analytics and System Reliability

For enhanced data tracking and improved system reliability, we propose a revolutionary approach to data analytics and visualization that leverages blockchain's inherent advantages in data integrity and accessibility. Traditional IoT data analysis systems typically involve multiple intermediaries - data analysts processing raw data and relay servers transmitting processed information to decision-makers. This multi-step process introduces potential points of data corruption and creates opportunities for malicious administrators to compromise data integrity.

Our solution implements a direct data access approach through a desktop application built with Electron, which connects directly to Hyperledger Fabric via gRPC. Additionally, the solution incorporates generative AI to transform natural language inputs into executable queries, enabling administrators without programming expertise to interact with and analyze raw data directly. This AI-powered query generation system allows non-technical users to extract meaningful insights from the blockchain data through intuitive language-based interactions.

This approach significantly improves system reliability by ensuring that decision-makers work with authentic, unaltered data. The combination of blockchain's immutable data storage, direct access through gRPC, and AI-assisted query generation creates a powerful platform for data-driven decision making that maintains data integrity while being accessible to users regardless of their technical expertise.

4) Decentralization

Our approach to decentralization fundamentally reimagines the Matter protocol's architecture while maintaining its core benefits. Unlike the traditional Matter protocol that relies on a centralized Certificate Authority (CA) for device certification, Matter Tunnel eliminates this requirement while preserving protocol compatibility. By removing the centralized authentication system, we lower barriers to entry for device manufacturers and smaller development teams, fostering innovation and competition in the IoT ecosystem.

This decentralized approach maintains the Matter protocol's ability to control multiple vendors' devices through a single application, ensuring that the key benefit of interoperability remains intact. The elimination of platform dependencies further enhances true decentralization, freeing users from vendor lock-in and creating a more open IoT environment.

5) Enhanced Security and Privacy Protection

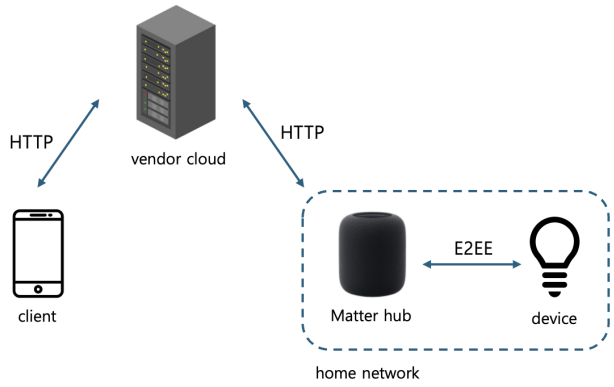

Fig. 19. Traditional Matter

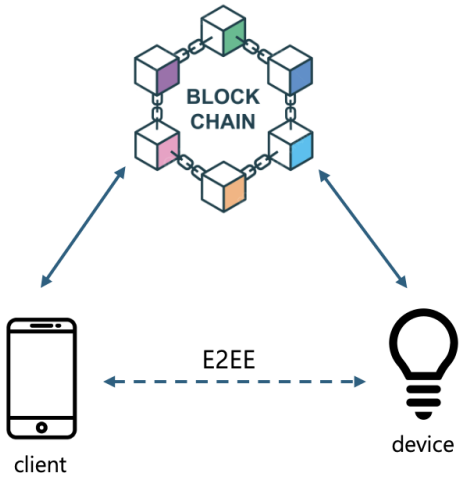

Fig. 20. Matter Tunnel E2EE

Our solution significantly enhances End-to-End Encryption (E2EE) and user privacy protection by fundamentally restructuring the communication architecture of Matter devices. Traditional Matter implementations rely on a cloud-based communication model where applications communicate with cloud services, and Matter hubs poll these services for updates. This structure inherently compromises both E2EE and privacy. Matter Tunnel addresses these limitations through a blockchain-based approach.

Key Security and Privacy Enhancements:

a) Direct Blockchain Communication

By eliminating cloud service intermediaries, this system enables direct and encrypted communication between applications and devices. The removal of vulnerable points in the communication chain enhances security, while ensuring true end-to-end encryption throughout the entire process.

b) Enhanced Privacy Protection

This system maintains privacy by only exposing device public keys on the blockchain, while keeping sensitive information like IP addresses and location data completely private. This approach enables anonymous device operation, significantly reducing the attack surface for potential privacy breaches.

c) Secure Device Registration and Authentication

The system implements a robust security mechanism through a predefined 'regist' function on the blockchain where users must verify device ownership by providing the correct 128-bit passcode. This establishes a secure transaction relationship between the user and device. With the passcode's substantial bit length, the probability of an adversary successfully guessing it is negligible $\left(\frac{1}{2}\right)^{128}$, effectively preventing unauthorized access at the fundamental level. This approach creates an inherently secure system that prevents attacks by design rather than through reactive measures.

By leveraging blockchain technology and existing Matter security features, our solution creates a more robust and private IoT ecosystem. The combination of anonymous operation, secure message counting, and direct blockchain communication ensures that both E2EE and user privacy are maintained at the highest possible level.

6) Real-Time Performance with Hyperledger Fabric

To ensure real-time performance in Matter Tunnel, we carefully selected Hyperledger Fabric as our blockchain platform after extensive evaluation of various options. This choice was driven by Hyperledger Fabric's unique characteristics that align with the real-time requirements of IoT device communication.

Hyperledger Fabric's high transaction processing capability stands out as a crucial feature for our implementation, supporting 2,000-20,000 transactions per second (TPS). This throughput is achieved through multiple channels that enable parallel transaction processing, effectively handling concurrent device communications while providing near-instantaneous transaction finality. The platform's configurable architecture can enhance performance by allowing optimization of network parameters, enabling adjustment of block creation time, supporting custom channel configurations for different device groups, and permitting fine-tuning of consensus mechanisms to match specific use case requirements.

By leveraging Hyperledger Fabric's comprehensive capabilities, Matter Tunnel achieves the real-time performance necessary for effective IoT device control and monitoring. The platform's ability to handle high transaction volumes while maintaining low latency ensures that device interactions remain responsive and reliable, meeting the

demanding requirements of modern IoT applications.

B. Development Requirements Specifications

Client

1) Entry & Tutorial

ID	Name	Description
001	MatterTunnel-Entry	When the application is launched, this component displays the initial entry page with the Matter Tunnel logo. It serves as the first touchpoint for users and should maintain visibility until the application completes loading its core data and services.

ID	Name	Description
002	MatterTunnel-Tutorial	A comprehensive tutorial screen that appears after the initial entry, remaining active until the user completes all tutorial steps or chooses to skip. The component guides users through multiple slides introducing Matter Tunnel's core functionalities and features, helping new users understand the platform's key advantages before they begin using the service.

ID	Name	Description
003	MatterTunnel-Tutorial-Skip	A component that allows users to bypass the entire tutorial sequence and directly proceed to the login page. This feature is designed for experienced users who are already familiar with the system and wish to access their accounts immediately.
004	MatterTunnel-Tutorial-Next	A navigation component that enables users to progress through multiple tutorial pages sequentially. It maintains its functionality until the user reaches the final tutorial page, guiding them through each step of the introduction process.
005	MatterTunnel-Tutorial-NavigateToLogin	A component that appears on the final tutorial page, directing users to transition from the tutorial completion to the login page. This represents the end of the tutorial flow and the beginning of the actual application usage.

2) Authentication

a) Login

ID	Name	Description
006	MatterTunnel-Login-Page	The authentication page that allows users to access the Matter Tunnel service. This component contains two primary functions: a login form for existing users to enter their credentials and a sign-up option for new users who want to create an account. The page features a clean, minimalist design with a text input field and a primary action button for login, along with a secondary option to navigate to the sign-up process.

ID	Name	Description
007	MatterTunnel-Login-Error	The system must validate user input during login. Both when the input field is empty and when the key does not match the required format (exactly 64 hexadecimal characters using 0-9 and A-F, pattern: `/^[0-9a-fA-F]{64}$/`), display the error message "Your key is incorrectly formatted" below the input field.

b) SignUp

ID	Name	Description
008	MatterTunnel-SignUp-Initial	Upon accessing the Sign-Up box under the Login section, users are presented with an initial prompt encouraging them to register if they are not yet members. The "Sign Up" button is displayed, inviting users to start the sign-up process. This component serves to inform users that they can proceed to registration to enjoy the features of the Matter Tunnel platform.

ID	Name	Description
009	MatterTunnel-SignUp-PrivateKey	After clicking the "Sign Up" button, users are directed to a screen where a private key is generated for them. This key is shown in a 64-character hexadecimal format, ensuring its precision and security. The private key is crucial for the user's future access to Matter Tunnel, and users are encouraged to store it securely.

ID	Name	Description
010	MatterTunnel-SignUp-Completion	Once the "Sign Up" button is pressed, the private key is generated and displayed on the screen. Simultaneously, a "Sign Up Completed" message appears, indicating that the registration process has been successfully completed. This stage reassures the user that they are now fully registered and ready to proceed to the next steps in the Matter Tunnel platform.

ID	Name	Description
011	MatterTunnel-SignUp-Copy	This component allows users to copy their generated private key by clicking the "Copy" button. A confirmation message is displayed once the key is successfully copied to the clipboard, ensuring that users can easily save their credentials for future use and proceed with logging into the Matter Tunnel platform.

c) Account Management

ID	Name	Description
012	MatterTunnel-Account-PrivateKey	This component displays the user's private key. It is a critical part of the user's identity within Matter Tunnel, and users can view it to manage their credentials. The key is displayed securely, ensuring it is visible only to the authorized user.

ID	Name	Description
013	MatterTunnel-Account-CopyButton	A "Copy Private Key" button is provided below the private key. When users click the button, the private key is copied to their clipboard. After the action is completed, an alert message "Private key copied" appears at the top of the screen, confirming that the key has been successfully copied and can be securely stored for later use.

ID	Name	Description
014	MatterTunnel-Account-LogoutButton	The "Logout" button allows users to exit their current session. When clicked, the system logs the user out and redirects them to the Login page. Upon successful logout, a "Logout Completed" message appears below the "Login" button, confirming that the user has been successfully logged out.

3) Main Interface

a) Device list

ID	Name	Description
015	MatterTunnel-Device-List	This component displays a list of devices registered by the user within the Matter Tunnel platform. If no devices are added, the screen will prompt the user to add a device in order to experience various services provided by Matter Tunnel. This list serves as the main interface for device management. The list is interactive, allowing users to view details of their devices and navigate to further actions like adding or deleting devices.

ID	Name	Description
016	MatterTunnel-NavigateTo DeviceAddition Button(+)	This button is located on the Device List screen and allows users to add new devices to their Matter Tunnel account. When clicked, the user is redirected to the QR code scanning screen, where they can scan a QR code to register a new device. This action enhances the user experience by expanding the available devices within the Matter Tunnel.

ID	Name	Description
017	MatterTunnel-NavigateTo DeviceDeletion Button(-)	This button is located on the Device List screen, allowing users to delete one or more devices from their Matter Tunnel account. When clicked, the user can select devices to delete by using checkboxes displayed on the Device List. This action provides users with an easy and efficient way to manage their devices by removing unnecessary ones, helping maintain a clutter-free account.

ID	Name	Description
018	MatterTunnel-DeviceComponent	Each device in the list is represented by a Device Component. This component includes the device icon, device name, and the public key associated with the device, represented by a QR code. Additionally, there is a NavigateToDeviceControl-Button, which allows users to navigate to the Device Control screen for managing the device. The device component provides an overview of each device and serves as an interactive element for device management.
019	MatterTunnel-NavigateTo DeviceControl Button(:)	This button is embedded within the Device Component and allows users to navigate to the Device Control screen. By clicking this button, users can control and configure specific settings for the selected device. The button provides quick access to device management functions.

ID	Name	Description
020	MatterTunnel-NavigationBar	The Navigation Bar is a persistent component at the bottom of the screen, providing users with easy access to different sections of the app. It includes buttons for navigating to the Device List and Account Management pages. Clicking the Home button redirects users back to the Device List screen, while the Account button takes them to the Account Management screen. This navigation ensures that users can seamlessly switch between managing their devices and adjusting their account settings. However, the Navigation Bar is not visible during the Login and Sign-Up processes.

b) Device addition

ID	Name	Description
021	MatterTunnel-DeviceAdditionScreen	This component displays the Device Addition screen, which allows users to add a new device to their account. When the ”+” button on the Device List screen is clicked, the user is navigated to this screen, where they can scan a QR code and enter a name for the device to complete the registration.

ID	Name	Description
022	MatterTunnel-DeviceQRScanScreen	This screen is displayed for users to scan a QR code for adding a device. Once the QR code is successfully scanned, the screen changes to black, and a message “QR Code Scan Complete!” appears. “Scan again” button is also available at the bottom of the screen, allowing users to scan a new QR code if needed. If the QR code has already been scanned and the device is already registered, an alert message appears saying ”Already Registered Device!”

ID	Name	Description
023	MatterTunnel-DeviceNameInput	After successfully scanning a QR code, this input field allows users to name the device. The input field becomes active, enabling users to assign a unique name to the device they are registering. Once the name is entered, the ”Register” button will be enabled.

ID	Name	Description
024	MatterTunnel-DeviceRegister Button	Once a device name is entered, the ”Register” button becomes active. Clicking this button triggers the addition of the device to the system. When the device is successfully registered, an alert message appears saying ”Device Successfully Registered.”

ID	Name	Description
025	MatterTunnel-DeviceAddition CancelButton(X)	The ”Cancel” button allows users to exit the Device Addition screen without adding a new device. Clicking this button returns users to the Device List screen, canceling the addition process at any point.

c) Device deletion

ID	Name	Description
026	MatterTunnel-DeviceDeletionScreen	This screen is displayed when the user clicks the “-” button to initiate the device deletion process. The checkboxes appear at the top-right corner of each device component, and the “Cancel” and “Delete” buttons replace the previous button. The user can select one or more devices for deletion. After selecting the devices, the “Delete” button becomes active. The user can confirm the deletion, and a confirmation message is shown once the devices are successfully deleted. If the user chooses to cancel, the selection is cleared, and no changes are made to the device list.

ID	Name	Description
027	MatterTunnel-DeviceDeletionCheckbox	This checkbox appears at the top-right corner of each device component on the Device Deletion screen. Users can select one or more devices for deletion by checking the corresponding checkboxes. The checkboxes become active once the device list is displayed in the deletion mode.

ID	Name	Description
028	MatterTunnel-DeviceDeletion CancelButton	This button allows users to cancel the device deletion process. It is displayed on the Device Deletion screen alongside the ”Delete” button. If clicked, the user’s selection is cleared, and no changes are made to the device list.

ID	Name	Description
029	MatterTunnel-DeviceDeletio nButton	This button becomes active after the user selects one or more devices for deletion. When clicked, it initiates the deletion of the selected devices, and a confirmation message appears notifying the user of the successful deletion.

d) Device control

ID	Name	Description
030	MatterTunnel-DeviceControl Screen	This screen provides users with the option to control device functions within the Matter Tunnel system. It features two main components: Feedback and Device Functions. The Feedback section displays the feedback received from the server after executing a function. The Device Functions section lists the available actions, which are derived from the device's QR code. When a function is executed, a transaction is sent to the server, and the corresponding feedback is displayed, allowing users to interact with the device and manage settings.
031	MatterTunnel-DeviceFeedback Section	This section displays the feedback received from the server after executing a function. It includes details such as the device icon, device name, and the feedback content. While the function is being executed, the feedback section shows the message "Processing Transaction...". Once the function is completed, the appropriate feedback is displayed based on the function's execution.
032	MatterTunnel-DeviceFunction Name	This component displays the name of the function retrieved from the device's QR code. Each device has different functions, and this field shows the specific function available for that device. The function name serves as an indicator of what action will be executed once the user interacts with it.
033	MatterTunnel-DeviceFunction Arguments	This section shows the arguments required for the function, which may vary based on the function itself. If the function requires inputs, users can provide them here. If the required number of arguments is not entered, the message "Please Enter All Parameters" will appear. If the arguments do not match the expected format, an error message "Please Enter Valid Format" will be displayed.
034	MatterTunnel-DeviceFunction ExecuteButton	This button allows users to execute the function after providing the necessary arguments. Once clicked, a transaction is sent to the server to perform the requested function, and feedback is shown in the Feedback section. The button is only enabled once the required inputs are correctly filled out.

Dashboard

1) Dashboard Screen

ID	Name	Description
001	Dashboard-Screen	The main full-screen interface of the dashboard, consisting of a navigation bar, four key components, and a refresh button. These elements allow users to interact with and monitor blockchain transactions effectively.

2) Dashboard Components

a) Transactions Per Second Chart

ID	Name	Description
002	Dashboard-TransactionsPerSec	A real-time line chart that displays the number of blockchain transactions processed per second. The chart dynamically updates every second to show the latest transaction rates. When the user hovers the mouse over any point on the chart, the number of transactions processed at that specific time is displayed, offering insights into the transaction rate for the selected moment.

b) Transactions by PK Chart

ID	Name	Description
003	Dashboard-TransactionsByPK	A bar chart that visualizes the transaction distribution based on Public Keys. When hovering over each bar, the exact number of transactions for the corresponding PK is shown, allowing users to analyze the transaction count for each key.

c) Total Transactions and Users

ID	Name	Description
004	Dashboard-Total	Displays a comprehensive summary of the total blockchain transactions and the overall number of registered users. This component provides a quick, easy-to-understand overview of key metrics related to transaction volume and user activity.

d) Transaction Management and Natural Language Processing

ID	Name	Description
005	Dashboard-NLQueryTool	A tool that allows users to manage and query blockchain transaction details using natural language. Users can input queries such as ”Show all transactions for PK X in the last week,” and results are displayed visually for easy interpretation. This tool enables the management of blockchain transactions across the IoT industry and allows users to query transaction data through natural language processing.

e) Refresh Button

ID	Name	Description
006	Dashboard-RefreshButton	A button designed to refresh all charts and data components on the dashboard. Upon clicking, all visualizations are updated to reflect the most recent blockchain information, ensuring users have access to real-time data.

3) Dashboard Navigation Menu

ID	Name	Description
007	Dashboard-Navigation	Navigation bar provides access to key system functions including Dashboard, Products, Customers, Orders, Analytics, Marketing, Discounts, Payouts, Statements, Calendar, and Storefront. Dashboard is the primary view, displaying essential metrics and visualizations for monitoring blockchain transactions.

V. Architecture Design & Implementation

A. Overall Architecture

Matter Tunnel's architecture consists of several interconnected modules designed to provide a secure, and efficient IoT management system. Each module serves a specific purpose while maintaining robust communication with other components.

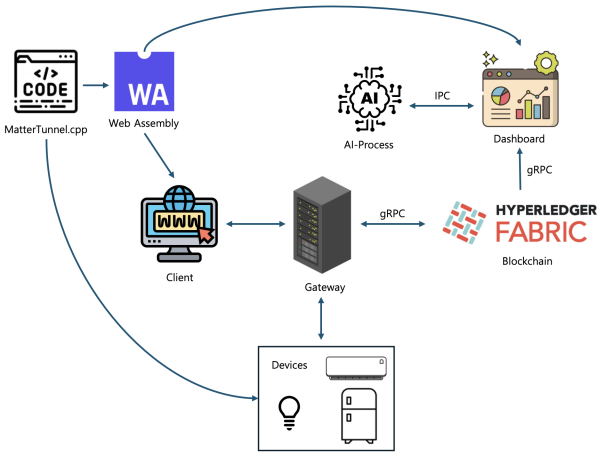

Fig. 21. Overall Architecture

1) Core Modules

a) Client Application

- Developed using React for cross-platform compatibility
- Provides intuitive user interface for device management
- Integrates WebAssembly-compiled Matter Tunnel utilities
- Communicates with the gateway for blockchain interaction

b) Matter Tunnel Utility

- Core functionality implemented in C++
- Dual deployment approach:
  - Direct C++ implementation for IoT devices
  - WebAssembly compilation for web-based clients
- Ensures consistent behavior across different platforms
- Handles device communication and protocol management

c) Gateway

- Serves as intermediary between clients/devices and blockchain
- Simplifies development process for client and device implementations
- Provides standardized API for blockchain interaction
- Manages connection pooling and request routing

d) Hyperledger Fabric Network

- Implements decentralized device management
- Maintains immutable record of device interactions
- Provides high-performance transaction processing
- Supports multiple channels for scalability

e) Dashboard Application

- Built with Electron for desktop performance
- Connects directly to blockchain via gRPC
- Provides comprehensive monitoring and analytics interface
- Integrates with AI component through IPC

f) AI Analysis Component

- Based on fine-tuned T5 model
- Processes natural language queries
- Communicates with dashboard through simple IPC
- Generates blockchain queries from natural language

2) Key Technical Decisions

The architecture reflects several key technical decisions made to optimize system performance and usability:

a) Development Approach

- React for cross-platform client development
- Electron for high-performance dashboard
- C++ for core Matter Tunnel functionality
- WebAssembly for web integration

b) Communication Protocols

- gRPC for blockchain communication
- IPC for AI-Dashboard interaction
- Gateway API for client-blockchain abstraction

c) AI Implementation

- T5 model fine-tuning for specific use case
- Natural language processing for query generation
- Integration with blockchain data analysis

This architecture enables Matter Tunnel to provide a robust, scalable, and user-friendly IoT management system while maintaining high security and performance standards.

B. Blockchain Architecture

Matter Tunnel's blockchain infrastructure is built on Hyperledger Fabric, implementing a robust and scalable architecture designed for enterprise-grade IoT device management. The network consists of multiple components organized to ensure high availability, fault tolerance, and efficient transaction processing.

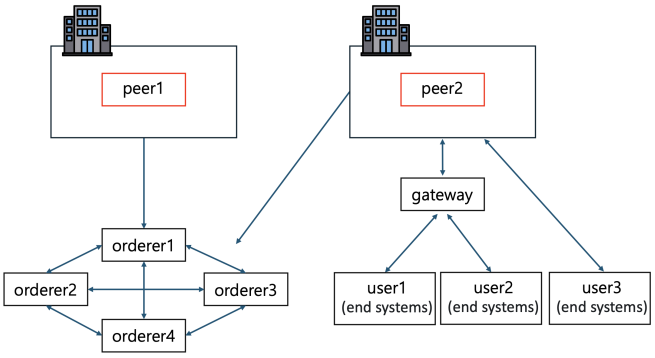

Fig. 22. Blockchain Architecture

1) Network Components

a) Peer Organizations

- Two peer nodes (peer1, peer2) maintained by separate organizations
- Each peer maintains a copy of the ledger
- Responsible for endorsing transactions and maintaining state
- Hosts chaincode for device management and communication

b) Ordering Service

- Four ordering nodes (orderer1-4) in Byzantine Fault Tolerance configuration
- Implements crash fault tolerance and Byzantine fault tolerance
- Ensures consensus among network participants
- Can continue operation even if f nodes fail, where f = (n-1)/3

c) Gateway Service

- Provides simplified access point for end systems
- Manages connection pooling and load balancing
- Handles authentication and authorization
- Optimizes network communication

d) End Systems

- Multiple user applications and devices
- Can connect either through gateway or direct gRPC
- Maintains flexibility in connection methods
- Supports various client implementations

2) Communication Patterns

a) Gateway-mediated Communication

- Simplified access through gateway API
- Reduced connection overhead
- Centralized authentication management
- Optimal for standard client applications

b) Direct gRPC Communication

- High-performance direct peer connection
- Suitable for advanced applications
- Full access to Fabric SDK capabilities
- Used by dashboard and specialized clients

3) Fault Tolerance

a) Byzantine Fault Tolerance

- Four-node ordering service ensures BFT
- Tolerates up to f = 1 Byzantine failure
- Maintains consensus under malicious behavior
- Provides strong consistency guarantees

b) High Availability

- Multiple peers ensure continuous operation
- Redundant ordering service nodes
- No single point of failure
- Automatic failover capabilities

This blockchain architecture provides the foundation for secure, scalable, and reliable IoT device management while maintaining flexibility in how clients connect to and interact with the network.

C. Directory Organization

Matter Tunnel consists of six GitHub repositories: blockchain, IoT_embedded, AI_dashboard, matter_tunnel_utils, client, Documents.

1) The blockchain repository handles blockchain-related implementation built on Hyperledger Fabric. This repository contains smart contracts and protocols designed to enable secure device-user communication between applications and IoT devices, ensuring complete end-to-end encryption(E2EE) and enhanced security.

2) The IoT_embedded repository is dedicated to embedded systems development for IoT devices. It contains software implementations for devices compatible with the Matter Tunnel protocol and the proposed blockchain solution. This repository focuses on implementing features that enhance device functionality.

3) The AI_dashboard repository creates and manages dashboard interfaces for monitoring and controlling IoT devices connected through the Matter Tunnel System. This repository implements both AI services and frontend visualization components to provide a comprehensive view of IoT device operations and blockchain network status. It integrates data processing capabilities with user-friendly interfaces, enabling real-time monitoring, visualization of transactions.

4) The matter_tunnel_utils repository provides essential utilities and tools for handling Matter tunnel protocol operations. It implements the core functionality that enables direct P2P communication between applications and IoT devices on the blockchain.

5) The client repository focuses on developing the front-end application that provides user interfaces for IoT device control. This repository contains code for implementing intuitive and responsive front-end solutions that integrate with both the Matter protocol and blockchain technology, ensuring a smooth user experience without requiring a Matter hub.

6) The Documents repository maintains comprehensive project documentation, including technical specifications, architecture designs, and implementation details of the Matter Tunnel system.

TABLE V
DIRECTORY ORGANIZATION - BLOCKCHAIN

Directory	File Name
blockchain/matter_tunnel /application_gateway	go.mod go.sum main.go
blockchain/matter_tunnel/chaincode	go.mod go.sum main.go

TABLE VI
DIRECTORY ORGANIZATION - IOT EMBEDDED

Directory	File Name
IoT_embedded/IoT_embedded	main.cpp matter_tunnel.cpp

TABLE VII
DIRECTORY ORGANIZATION - AI_DASHBOARD

Directory	File Name
AI_dashboard/ai/src/config	init.py model_config.py
AI_dashboard/ai/src/utils	init.py data_loader.py filter_data.py preprocessing.py query_augmenter_nl_pa ug.py query_augmenter.py inference.py train.py
AI_dashboard/frontend/components	BarChartComponent.js ChatBox.js Sidebar.js TPSChart.js chartDimensions.js MetricCards.js
AI_dashboard/frontend/dataset	getTransactions.js

TABLE VIII
DIRECTORY ORGANIZATION - MATTER_TUNNEL_UTILS

Directory	File Name
matter_tunnel_utils/IoT_crypto	main.cpp matter_tunnel.cpp wasm_matter_tunnel.cpp
Matter_tunnel_utils/qr_maker	qr_maker.py

TABLE IX
DIRECTORY ORGANIZATION - CLIENT

Directory	File Name
client/src/common	matter_tunnel.js
client/src/modules/auth/components	Login.js SignUp.js auth.module.css user.js
client/src/modules/auth/hooks	useLogin.js useSignUp.js
client/src/modules/auth/states	authSlice.js
client/src/modules/core/states	store.js
client/src/modules/device/components	AddDevice.js BottomNav.js DeviceList.js RemoveDevice.js UpdateDevice.js
client/src/modules/device/hooks	useRemoveDevice.js
client/src/modules/device/states	deviceSlice.js

D. Module 1: Blockchain

1) Purpose: For implementing Matter Tunnel, we utilized Hyperledger Fabric. The blockchain module serves as a crucial component in resolving network constraints in P2P communication and ensuring secure communication between Matter devices and applications. This enables IoT devices to communicate with each other without requiring a Matter hub.

2) Functionality: This module supports bypass of NAT constraints by acting as a virtual private network. It enables end-to-end encrypted communication between IoT devices and applications. The module provides a platform-independent communication structure and implements a decentralized authentication system for user privacy protection.

3) Location of source code: https://github.com/Winter-Zzzz/blockchain/matter_tunnel

4) Class component:

- application_gateway folder: This folder contains Go files that act as a gateway to convert the gRPC protocol to an HTTP/REST API, allowing client applications to easily communicate with the blockchain network.
- chaincode folder: This is a folder containing smart contract implementation files that define communication rules in Hyperledger Fabric and managing transactions.
- go.mod: This is a file for managing dependencies in GoLang. All modules used in GoLang are maintained in the go.mod file.
- go.sum: This file lists down the checksum of direct and indirect dependency requires along with the version. It is to be mentioned that the go.mod file is enough for a successful build. The checksum present in go.sum file is used to validate the checksum of each of direct and indirect dependency to confirm that none of them has been modified.

E. Module 2: IoT_embedded

1) Purpose: We developed a virtual IoT device implementation. This approach allowed us to simulate MatterTunnel-compatible devices without requiring physical hardware. The software-based virtual devices implement the same communication protocols and functionalities as real IoT devices would in the Matter Tunnel ecosystem, enabling thorough testing and validation of the system's capabilities in a controlled environment.

2) Functionality: This module manages the IoT device's connection to the blockchain network by gateway, handles device-specific operations, and implements the Matter protocol specifications. It enables secure P2P communication and integrates with the blockchain module for data transfer.

3) Location of source code: https://github.com/Winter-Zzzz/IoT_embedded/IoT_embedded

4) Class component:

- main.cpp: This is the main entry point file containing device initialization and core control logic.
- matter_tunnel.cpp: This is a file implementing Matter Tunnel protocol specifications.

F. Module 3: AI_dashboard

1) Purpose: For visualizing and analyzing blockchain metrics and transaction data, we developed a dashboard using Electron. The AI dashboard module provides an intuitive interface for monitoring system performance, displaying statistical data, and managing blockchain transactions. This module helps understanding the system's behavior and performance through various visual components.

2) Functionality: This module visualizes transaction statistics, provides real-time monitoring capabilities, and offers interactive data visualization components. It includes features for displaying blockchain transactions, and key metrics through various chart types.

3) Location of source code: https://github.com/Winter-Zzzz/AI_dashboard

4) Class component:

- ai folder: A folder containing AI implementations and models for data analysis.
  - config folder: Configuration files for AI models.
    - init.py: Initialization file for the config module.
    - model_config.py: Configuration file for AI model parameters.
  - utils folder: Utility functions for data processing.
    - data_loader.py: Loading and managing data.
    - filter_data.py: Data filtering operations.
    - preprocessing.py: Data preprocessing functions.
    - query_augmenter_nlpaug.py: NLP-based query augmentation.
    - query_augmenter.py: General query augmentation.
    - inference.py: Model inference operations.
    - train.py: Model training functions.
- components folder: Reusable UI components for a dashboard.

  - BarChartComponent.js: Bar chart visualizations.
  - ChatBox.js: Communication interface component.
  - Sidebar.js: Sidebar component.
  - TPSChart.js: TPS Chart visualizations.
  - chartDimensions.js: Utility file for chart sizing.
  - MetricCards.js: Displaying total transactions and users in card style.
- dataset folder: File for handling blockchain transaction data processing and analytics.
  - getTransactions.js: Fetches and processes blockchain transaction data.

G. Module 4: matter_tunnel_utils

1) Purpose: For supporting Matter Tunnel's core functionality, we developed utility modules. It provides essential tools and utilities, including IoT device cryptography handling and QR code generation capabilities. This module serves as a supporting layer for the main Matter Tunnel implementation.

2) Functionality: This module handles cryptographic operations for IoT devices, manages WebAssembly implementations of Matter Tunnel, and provides tools for generating QR codes used in device pairing and configuration processes.

3) Location of source code: https://github.com/Winter-Zzzz/matter_tunnel_utils

4) Class component:

- IoT_crypto folder: This is a folder containing cryptographic & Matter tunnel implementations.
  - main.cpp: This is the main entry point file that contains many tests.
  - matter_tunnel.cpp: This is a file implementing Matter Tunnel cryptographic functions.
  - wasm_matter_tunnel.cpp: This is a WebAssembly implementation of Matter Tunnel functions.
- qr_maker folder: This is a folder for QR code generation utilities.
  - qr_maker.py: This is a file for generating QR codes for device configuration.

H. Module 5: client

1) Purpose: The client module serves as the frontend interface for Matter Tunnel, providing user authentication, device management, and interaction capabilities. It enables users to control their IoT devices through a web-based interface without requiring a traditional Matter hub, leveraging the blockchain infrastructure for secure communication.

2) Functionality: This module implements user authentication and registration, device management (adding, removing, and updating devices), and provides a seamless interface for controlling Matter-compatible devices. It utilizes React hooks for state management and implements end-to-end encryption for secure communication with the blockchain network.

3) Location of source code: https://github.com/Winter-Zzzz/client/src

4) Class component:

- common folder: This is a folder containing shared utilities and components used across the application.
  - matter_tunnel.js: This is a file for core functionality implementation for Matter protocol integration.
- modules folder: This is a folder containing the core functional units of the application.
  - auth folder: This is a folder containing all authentication-related components, hooks, and state management.
    - components folder: This is a folder containing React components related to authentication features.
      - Login.js: This is a file that renders the login form and handles user authentication logic.
      - SignUp.js: This is a file implementing user registration interface and validation.
      - auth.module.css: This is a CSS file containing styles specific to authentication components.
      - user.js: This is a file that manages user authentication state.
    - hooks folder: This is a folder containing files for authentication-related functionality.
      - useLogin.js: This is a file that manages login state and provides login-related functions.
      - useSignUp.js: This is a file that handles registration logic.
    - states folder: This is a folder containing files for managing authentication-related state using Redux.
      - authSlice.js: This is a file that manages authentication state.
  - device folder: This is a folder managing all device-related functionality and user interface components.
    - components folder: This is a folder containing components for device management.
      - AddDevice.js: This is a file that provides interface for adding new devices.
      - BottomNav.js: This is a navigation component file.
      - DeviceList.js: This is a file displaying the list of connected devices.
      - RemoveDevice.js: This is a file handling device removal functionality.
      - UpdateDevice.js: This is a file managing device update operations.
    - hooks folder: This is a folder containing custom hooks for device operations.
      - useRemoveDevice.js: This is a file managing device removal logic.
    - states folder: This is a folder managing device-related state using Redux.
      - deviceSlice.js: This is a file managing the state of devices.
  - core folder: This is a folder containing core application

state management.

- states folder: This is a folder managing global application state.
  - store.js: This is a file for the Redux store.

VI. Use Cases

A. Use case 1: Client

1) Initial Experience

When users first launch the application, they are greeted with the Matter Tunnel logo on the entry screen while the system loads. Following this, new users are presented with a comprehensive tutorial that introduces core functionalities through multiple slides. Users can navigate through the tutorial using the "Next" button or choose to skip it entirely. The tutorial concludes by directing users to the login page to begin using the application.

2) Login

The Login page enables users to access the system by entering their private key for authentication. Users must input a 64-character hexadecimal key (using characters 0-9 and A-F) for validation. If the key format is incorrect, the system displays an error message "Your key is incorrectly formatted". This secure authentication method ensures that only authorized users can access their device management interface.

3) Sign Up

a) Sign Up button: The user needs its own private key to use program. They can create new private key by clicking "Sign up" button on the login page. The system then generates and displays a new private key.

b) Copy button: Users can copy using the "Copy" button. After creating their own key, they can proceed to the login screen and attempt to log in with the newly created private key.

c) A "Sign Up Completed" message confirms successful registration.

4) Main Page

a) QR Code recognition: On the main page, users can add new devices to their system by clicking '+' button. After clicking the button, users enter the scanning interface where they can capture the device's QR code. Upon successful scanning, the screen displays "QR Code Scan Complete!" A "Rescan" option is available if needed. If the scanned device is already registered, the system shows "Already Registered Device!" warning.

b) Device addition: After QR scan, users can input a custom name for the device and the "Register" button becomes active. The system provides immediate feedback through notifications, either confirming successful registration with "Device Registration Complete" or warning if the device is already registered. Users can cancel the addition process at any time using the X button.

c) Device List Management: The main screen displays all registered devices in a square card format, with each card showing the device icon, name, and public key obtained from the QR code. A three-dot menu button ($\cdots$) on each card provides access to device controls. When accessed, users can view device feedback from the server and execute available device. The function section includes input fields for required arguments and displays "Processing Transaction..." during execution. Users can easily monitor and manage their connected devices from this central interface.

d) Device Deletion: Users can initiate device removal by clicking the '-' button, which activates deletion mode. This displays checkboxes on each device card, allowing users to select multiple devices for simultaneous deletion. The interface shows "Delete" and "Cancel" buttons during deletion mode. The deletion process requires at least one device selection and confirms the removal through a notification message.

5) Navigation

The application features a bottom navigation bar containing Home and Account buttons. The Home button returns users to the Device List screen. The Account Button directs account management page. This navigation bar remains visible throughout the application except during login and signup processes.

6) Account Management

When users click the Account button, they are directed to the account management page where they can view their public key information and manage their private key settings. Users can copy their private key using the "Copy Private Key" button. It displays a "Private key copied" alert upon successful copying. The page also includes a "Logout" button that, when clicked, logs the user out and redirects them to the login page with a "Logout Completed" confirmation message.

B. Use case 2: Enterprise

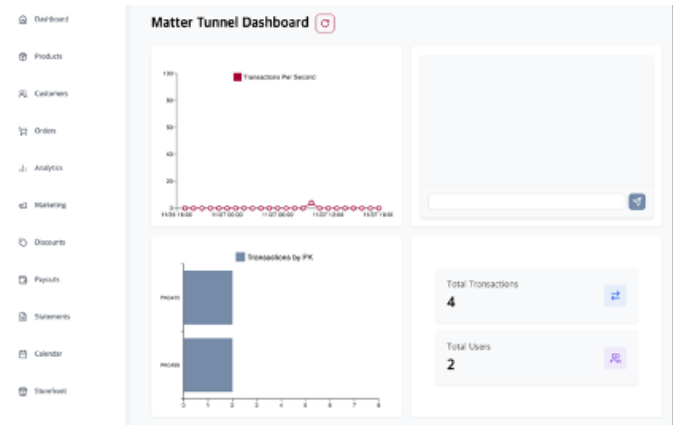

Fig. 23. Dashboard Full Screen

The Matter Tunnel Dashboard is a comprehensive monitoring system designed to provide real-time visibility into transaction processing and system performance. The primary users of this dashboard are system administrators and performance analysts who need to track and analyze transaction throughput across different time periods and processing nodes.

1) TPS chart

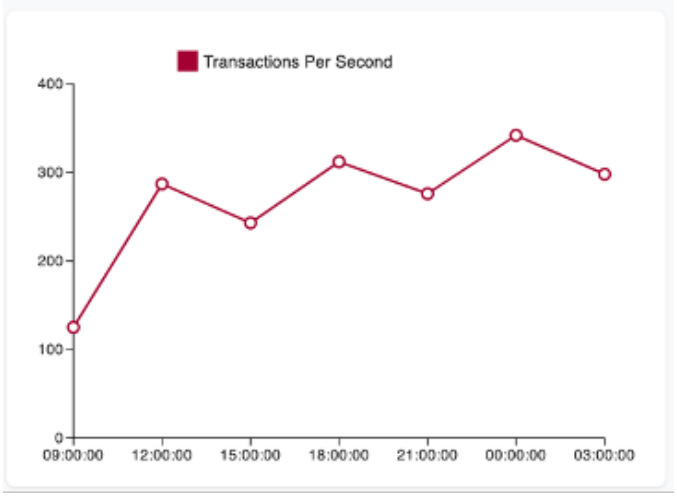

Fig. 24. TPS Chart

CopyThis is a real-time Transactions Per Second (TPS) line graph that refresh automatically to show the most recent 18-hour-period. The system plots actual transaction throughput as it occurs, allowing operators to observe performance trends and respond to any anomalies immediately.

2) PK transactions chart

This horizontal bar chart displays transaction distribution across distinct PKs (Primary Keys). This visualization updates in real-time to reflect the current processing load across different nodes. The bars will dynamically adjust as transaction volumes shift between PKs, enabling immediate detection of load balancing issues or processing bottlenecks.

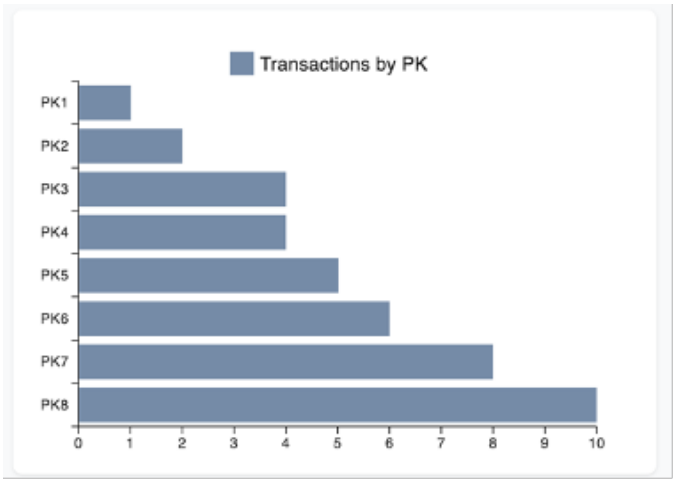

Fig. 25. PK Transactions Chart

3) Total Transaction&User Card

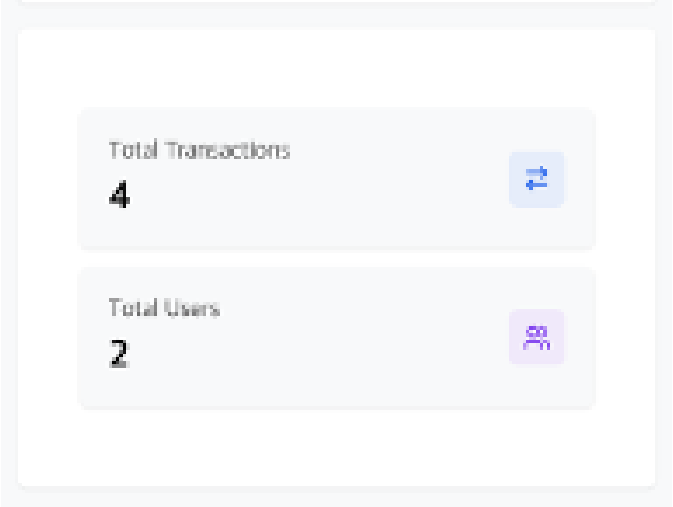

Fig. 26. Total Transaction & User Card

This part of dashboard displays key transaction statistics through metric cards. The first card shows total transactions, and the second card shows total users. These cards provide administrators and operators with immediate visibility into the system's current usage and activity levels, enabling quick assessment of blockchain network participation and transaction volume.

VII. Discussion

...