Blockchain-Enabled Smart Controller for Grid-Connected Microgrids

1st Kiran Rajkumar Shete
Artificial Intelligence and Machine Learning
P.E.S. Modern College of Engineering
Pune, India

2nd Adwait Nitin Deshpande
Artificial Intelligence and Machine Learning
P.E.S. Modern College of Engineering
Pune, India

5th Chandrashekhar Arvind Ghuge
Artificial Intelligence and Machine Learning
P.E.S. Modern College of Engineering
Pune, India

3rd Sharvari Pramod Jape
Artificial Intelligence and Machine Learning
P.E.S. Modern College of Engineering
Pune, India

4th Twarita Panjabrao Pole
Artificial Intelligence and Machine Learning
P.E.S. Modern College of Engineering
Pune, India

[1,23,4,]Student, Dept. of Artificial Intelligence and Machine Learning, PES's Modern College Of Engineering, Pune, Maharashtra, India
[5]Associate Professor, Dept. of Artificial Intelligence and Machine Learning, PES's Modern College Of Engineering, Pune, Maharashtra, India

***

Abstract - The rapid spread of grid-connected microgrids has significantly changed the way we generate, store and distribute energy. These decentralized energy systems have many advantages, such as lower energy costs, greater resilience and reduced environmental impact. However, their integration into the power grid also brings new cybersecurity challenges. This research paper presents the design of a blockchain-enabled smart controller that attempts to address these issues and ensure the secure operation of grid-connected microgrids. This comprehensive solution protectsthe critical infrastructure of microgrids.

***Key Words*: Microgrids, Blockchain, Hyperledger Fabrics, Docker, Web3**

1. Introduction

Grid-connected microgrids enable a paradigm shift in the generation, distribution and consumption of electrical energy and represent a revolutionary development in the energy landscape. These decentralized energy systems, which consist of various energy sources, energy storage options and sophisticated control systems, offer numerous advantages. These benefits include increased resilience, reduced environmental impact and greater energy efficiency. However, the integration of technologies into microgrids presents a number of new difficulties.[1][2]

As grid-connected microgrids become an increasingly important part of today's energy infrastructure, it is essential that they operate safely and continuity. They are an essential part of the energy ecosystem as they are able to adapt to changing energy supply and demand, incorporate renewable resources and provide vital backup power during grid outages.[1]

To address these pressing issues of conventional microgrid controllers in the context of grid-connected microgrids, this paper explores the design

of a blockchain-enabled smart controller. Our investigation into the development of this smart controller emphasizes maintaining the reliability and security of vital energy infrastructure as well as protecting microgrids. In this introduction, we have laid the groundwork for the next few pages, which will delve into the intricacies of developing a reliable and efficient blockchain-enabled smart controller for grid-connected microgrids so that these vital energy systems can survive in an increasingly interconnected and secure world.[1][3]

2. Design Considerations

The process of developing a blockchain-enabled smart controller for grid-connected microgrids is complex and requires careful consideration ofseveral important variables. A comprehensive approach is necessary to effectively protect these vital energy systems from ever-changing threats. This section describes the key design factors that guide the development of a powerful blockchain framework for grid-connected microgrids.[6][1]

2.1 Threat Analysis:

The design process depends on a solid understanding of the potential risks to which microgrids may be exposed. Typical threats include cybersecurity threats such as data integrity, privacy and network attacks, operational threats such as availability, scalability and interoperability, and supply chain risks such as vendor lock-in and software updates. A thorough threat analysis is essential to find vulnerabilities and create remediation measures that successfully mitigate these risks.[8]

2.2 Regulatory Compliance:

Security of grid-connected microgrids must comply with recognized industry regulations and best practices. Compliance with local, national and international standards is essential. Ensure compliance with data protection regulations. Microgrid data, user information and transaction details must be handled securely. Implement features to improve data protection such as pseudonymization and consent management.[1]

Ensure the security and reliability of microgrids by complying with industry and regulatory criteria such as ISO 27001 (information security), ISO 15118 (smart charging for electric vehicles) and NIST SP 800-183 (blockchain technology).[2][4]

2.3 Secure Communication:

Ensuring the security of communication channels is critical to protecting sensitive data and maintaining the integrity of the network infrastructure. It is necessary to implement strong encryption protocols such as TLS/SSL to protect the data transmitted between the microgrid components and the blockchain. The use of secure authentication mechanisms such as digital certificates or API keys helps to verify the identity of participants.[2][4]

Maintaining data integrity ensures that the data exchanged between the microgrid nodes and the blockchain remains tamper-proof.

There are some important design considerations for implementing secure communication in such a system:

End-to-end encryption: all communication channels within the microgrid system, including between smart controllers, sensors, meters and other devices, should be encrypted using robust cryptographic algorithms. End-to-end encryption ensures that data remains confidential and cannot be intercepted or manipulated by unauthorized parties.[2][4]

Authentication mechanisms: The implementation of strong authentication mechanisms is essential to verify the identities of participants within the blockchain network. This includes device authentication, user authentication and access control mechanisms to prevent unauthorized access to critical system components.[2][4][3]

Secure protocols: Use secure communication protocols such as HTTPS, MQTT with TLS or other industry-standard protocols that provide encryption, authentication and data integrity features. These protocols help prevent eavesdropping, man-in-the-middle attacks and data tampering.[2][4]

2.4 Software Solutions:

The software layer of the blockchain framework is essential for threat identification, access control and system management.

Blockchain: Blockchain enables a decentralized architecture where no single entity has control over the entire system. It serves as a decentralized and immutable ledger that records all transactions and data exchanges within the microgrid network. Each block in the blockchain contains a timestamped record of transactions, ensuring transparency and traceability of energy flows, payments and system operations. It creates trust between participants by ensuring transparency, immutability and consensus. The blockchain ensures secure and tamper-proof transactions between the components of the microgrid. The data stored in the blockchain remains unchangeable and prevents unauthorized changes. Blockchain increases cyber security by preventing unauthorized access and ensuring data protection. Blockchain uses cryptographic techniques to ensure the security and integrity of transactions within the microgrid ecosystem. Transactions are encrypted, timestamped and linked in a tamper-proof manner, making it virtually impossible for malicious actors to alter or tamper with the data stored in the blockchain. This mitigates the risks associated with cyberattacks, data breaches and unauthorized control over microgrid assets.[7][2]

Hyperledger Fabric: Hyperledger Fabric is a permissioned blockchain, which means that only authorized participants can join the network. It ensures that microgrid actors such as utilities, regulators and consumers have controlled access. Fabric supports private channels that enable confidential communication between specific participants. It provides robust identity management through its Certificate Authority (CA). Hyperledger Fabric utilizes Chaincode, a form of smart contract, to automate tasks and enforce business logic within the microgrid. Chaincode can be programmed to manage energy trading rules, implement automated pricing mechanisms and ensure secure and transparent settlements between participants.[7]

The blockchain-powered smart control for grid-connected microgrids can be designed to survive in the dynamic and ever-changing landscape if these aspects are carefully considered in the design phase. With these components in place, the system can ensure the safe and continuous operation of critical energy infrastructures. In an increasingly interconnected world, this foundation provides the necessary safeguards to fully utilize grid-connected microgrids.[2]

3. Blockchain Components

The Hyperledger Fabric binary plays a central role in the development of a blockchain-enabled intelligent controller for grid-connected microgrids. The Fabric binary is the command line tool used to install, configure and manage a Hyperledger Fabric network. It allows you to set up the necessary components, including peers, clients and certificate authorities. It helps to define the network topology, channels and organizations. It also facilitates the lifecycle management of smart contracts, e.g. Chaincode. Channels in Hyperledger Fabric enable private communication between specific participants. The Fabric Binary enables the creation and management of channels and ensures privacy within the microgrid network. The Fabric Binary interacts with the CA to issue and manage digital certificates for network participants.[7][8]

Certificates are crucial for secure communication and identity management. When calling transactions (e.g. energy trading or billing), the fabric binary communicates with peers and clients. It ensures that transactions are correctly confirmed, ordered and acknowledged. The Fabric Binary helps to update chain code versions and network components. If problems occur, the Fabric Binary provides diagnostic tools for troubleshooting. It helps to identify and fix network-related problems.[7]

The Fabric Binary enforces access control policies and ensures that only authorized users can perform network operations. It manages cryptographic materials and authorizations. This binary is responsible for maintaining a copy of the distributed ledger and participating in the consensus process to validate transactions.

The ordering service is responsible for ordering transactions and ensuring their consistency across

the network. It does not maintain a copy of the ledger, but plays a crucial role in ensuring that all peers agree on the order of transactions. Your application would likely interact with an orderer node to submit transactions for validation and inclusion on the blockchain.

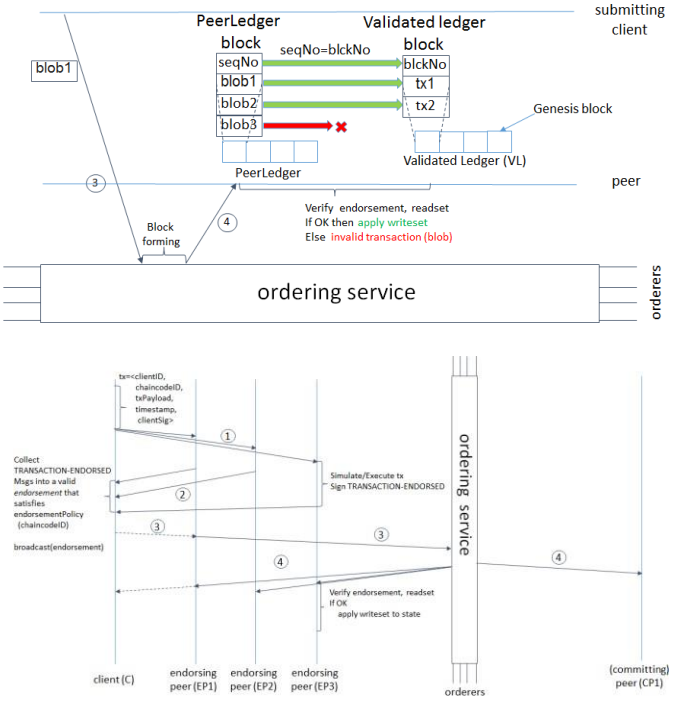

Fig. no. 01. Transactions in ordering service

The command line interface allows developers and administrators to interact with the Fabric network. It provides tools for deploying chaincode (smart contracts), managing channels (private communication subnets) and interacting with peers and orderers.[7][8]

To initialize the network and operate a blockchain-enabled smart controller for grid-connected microgrids, you must first set up the Hyperledger Fabric network with fabric binaries. This includes defining the network topology, creating channels, setting up peers and configuring the ordering service. Next, develop and implement smart contracts (chaincode) to govern interactions within the network, such as energy trading and network management. Once the network is initialized and the smart contracts are implemented, start the network components, including the peers and the ordering service, with fabric binaries. Participants in the microgrid ecosystem, such as energy producers, consumers and grid operators, would then interact with the network via client applications, submitting transactions and querying ledger data via the command line interface (CLI) or other interfaces. Continuous monitoring and maintenance of the network would ensure its smooth operation and integrity.[7]

3.1 Role of Docker

With Docker, you can put your application code, smart contracts and all necessary dependencies into Docker containers. These containers create isolated environments that ensure consistent execution regardless of the underlying hardware or operating system variations. This simplifies deployment on different servers or cloud environments within the microgrid. Docker makes it possible to package the microgrid application, including the blockchain components, in lightweight containers. Containers encapsulate all dependencies such as libraries, binaries and configurations required to run the system. Each container runs in isolation and ensures that the components of the microgrid do not interfere with each other.

Docker ensures consistent behavior in different environments. It helps scale microgrid components horizontally by spinning up multiple containers.
By using Docker, anyone can revert to previous versions or seamlessly update to newer versions. Tools like Docker Compose or Kubernetes help manage multiple containers. Docker offers security features such as user namespaces and resource restrictions. Docker images can be run on any platform that supports Docker (Windows, Linux, macOS).

3.2 Couch DB

CouchDB is a NoSQL database that plays a crucial role in managing data within the microgrid system. While Hyperledger Fabric uses its own statedatabase for storing blockchain ledger data, CouchDB can play a supporting role in your microgrid application design. CouchDB can be usedto store off-chain data related to grid conditions, energy consumption patterns, user profiles and transaction history. Unlike traditional databases, CouchDB works in a distributed manner so that each

node in the network can maintain a replica of the database. This ensures data availability and fault tolerance, even in the event of network partitions or node failures. In addition, CouchDB's support for peer-to-peer replication enables seamless synchronization of data across multiple nodes, increasing data integrity and resiliency. By integrating CouchDB into the blockchain network, the smart controller can efficiently manage and access both on-chain and off-chain data, enabling real-time decision making and optimizing the operation of grid-connected microgrids.

CouchDB stores important data related to the operation of microgrids, energy transactions and smart contracts. Its decentralized nature is in line with the overall decentralization goal of the blockchain-based system. The data stored in CouchDB remains immutable, making historical records tamper-proof. An accurate and verifiable history of microgrid activities is maintained. CouchDB stores the status of smart contracts (chaincode) deployed on the blockchain. It tracks variables, balances and other relevant information related to energy trading and billing.

CouchDB enables efficient querying and indexing of data. Users can quickly retrieve specific information, improving the responsiveness of the system. CouchDB supports replication across nodes, ensuring fault tolerance. Data redundancy and synchronization increase system reliability. CouchDB is the default state database for Hyperledger Fabric. It integrates seamlessly into the blockchain network and provides a consistent view of the data.

3.3 Blockchain Network

This blockchain network mainly consists of some of the important layers of the OSI model.

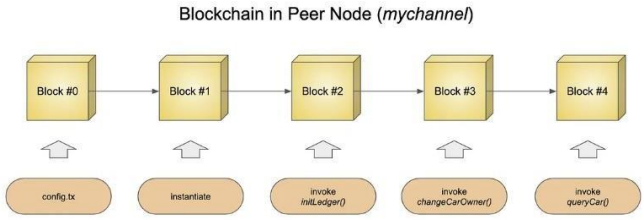

Fig. no. 02. Blocks in peer node

Transport layer: This layer ensures reliable data transmission between applications on different devices within the microgrid. Protocols such as TCP can be used to establish connections, guarantee proper delivery and handle error correction of data packets containing control signals, energy consumption information or smart contract interactions.

Session layer: The session layer establishes, manages and terminates connections between applications. In this context, it manages the communication sessions between the smart controller application and other entities in the microgrid ecosystem, e.g. energy producers, consumers and grid operators. It ensures reliable communication by establishing, synchronizing and terminating sessions.

Presentation layer: The presentation layer is responsible for data translation, encryption and compression. It ensures that the data exchanged between applications is available in a format that is understood by both parties. In the context of the smart controller, this layer can be responsible for data encryption to ensure the confidentiality and integrity of sensitive information such as transaction details and user credentials. Encryption protocols can be implemented here to protect sensitive information such as energy consumption data or financial transactions during transmission over the network.

Application layer: This top layer represents the user applications that interact with the system. In our case, it is the user interface for prosumers who can monitor energy consumption, buy/sell energy or view grid data. The application logic of the smart contract also resides on this layer and interacts with the lower layers to initiate transactions, retrieve data and manage the overall operation of the microgrid.

By working together, these upper layers of the OSI model enable secure and reliable communication

between the various entities within the microgrid ecosystem and ensure a smooth flow of data and control signals for efficient and secure grid-connected microgrid management.

4. Software Solutions

To identify, mitigate and manage cyber risks, the software layer of the blockchain-enabled smart controller is essential for grid-connected microgrids. A full suite of software solutions is required to protect the integrity, availability and confidentiality of the microgrid's vital infrastructure. In this section, we explore the functions of the key software components in relation to the design of the system.

Here you can see what the basic architecture of the Hyperledger Fabric looks like:

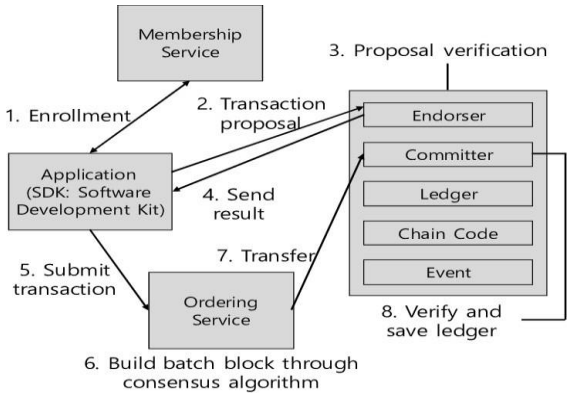

Fig. no. 03. Architecture of Hyperledger

SDK (Software Development Kit): An SDK is a set of tools and libraries that simplifies application development for a specific platform. In this context, the SDK enables applications to interact with the member service, submit transactions and query the general ledger.[7][8]

Transaction proposal: This is a proposed change to the state of the blockchain ledger. It comes from an application or user and is submitted to the member service for confirmation.

Transaction endorsement: Endorsers are validating nodes in the network. They check the transaction proposal to verify its validity and compliance with the rules of the network. If a sufficient number of endorsers approve the proposal, it moves to the next phase.

Membership service provider (MSP): The MSP manages the digital identities of the participants in the network and provides authentication and authorization mechanisms. It issues cryptographic certificates and manages the Public Key Infrastructure (PKI) to ensure the security and integrity of network interactions.

Ordering service: The ordering service receives confirmed transactions from peers, arranges them in a block and hands them over to the peers for validation and confirmation. It ensures that the transactions are ordered uniformly for all peers in the network.

Committer: The committer node is responsible for organizing valid transactions and entering them into the blockchain ledger. It checks the confirmed transactions and transfers them to the ledger in a specific order based on the selected consensus mechanism.

Ledger: This is the central data storage of the blockchain network. It contains a permanent and tamper-proof record of all approved transactions.

Lifecycle of Transaction:

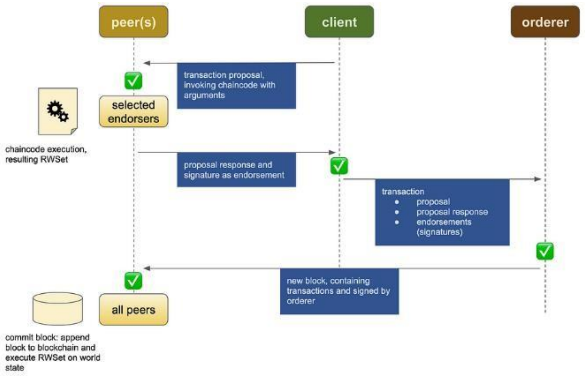

Fig. no. 04. Lifecycle of transactions in Hyperledger

The client sends a transaction proposal to the specified endorsing peer nodes (also called endorsers). The endorsers confirm that the client is authorised to send a transaction proposal and execute the chain code according to the information specified in the proposal and the world state. After execution, they send the response to the proposal back to the client. The response contains the endorser's simulation results in the form of a read-write set (RWSet), which shows what was read from the world state and what was written back after the chain

code was called. As the endorser, the endorser signs the response.[7]

After the client has received the responses, it validates them to ensure that everything is working properly. When the collected responses reach the required number of endorsers (e.g. one peer from each organization is required), the client creates a transaction and sends it to the principal. The transaction includes the transaction proposal, the response to the proposal and the endorsements.

The client receives transactions from the customer and validates them. Then the client creates a block with the validated transactions and sends this newly created block to all peer nodes. Upon receiving this block, the peer node validates the block and executes the transaction within the block, updating the world state according to the contents of the Read Write Set (RWSet). This block is now confirmed as a valid block in each peer node.[7][8]

Cosmos SDK: Cosmos SDK is a popular framework for creating application-specific blockchains. It offers a modular architecture, developer-friendly tools and support for various programming languages and consensus algorithms. Developers can use the Cosmos SDK to create custom blockchain applications for microgrid management, energy trading and decentralized management. The Cosmos SDK facilitates interoperability between different blockchain networks through the Inter-Blockchain Communication (IBC) protocol. This enables seamless communication and transfer of assets between grid-connected microgrid networks and other blockchain ecosystems, increasing flexibility and market connectivity.

4.1 Secure communication protocols:

To enable encrypted and authenticated communication between components of the microgrid, software solutions should include the use of secure communication protocols such as Transport Layer Security (TLS).

The blockchain-enabled smart controller for grid-connected microgrids ensures secure and uninterrupted operation of the microgrid by detecting and mitigating the threats through the integration of various software solutions. These software elements form a strong blockchain framework that protects the vital energy infrastructure of grid-connected microgrids and provides a thorough defense against ever-changing threats.[7]

5. Implementation and Testing

To confirm that a blockchain-enabled smart controller for grid-connected microgrids effectively protects vital energy infrastructure from threats, it must be successfully implemented and undergo a rigorous testing process. This section examines the important stages of testing and implementation to ensure that the controller functions properly and provides strong security measures.[6][5]

Network setup: In this step, we need to install and configure the necessary components, including order service nodes, peer nodes and certificate authorities (CAs), using Hyperledger Fabric binaries or tools such as Docker Compose.

Orderer Genesis Block: This is the first configuration block for the Orderer service. It defines important parameters such as the consensus mechanism used for transaction validation (e.g. PBFT) and the guidelines for creating channels.

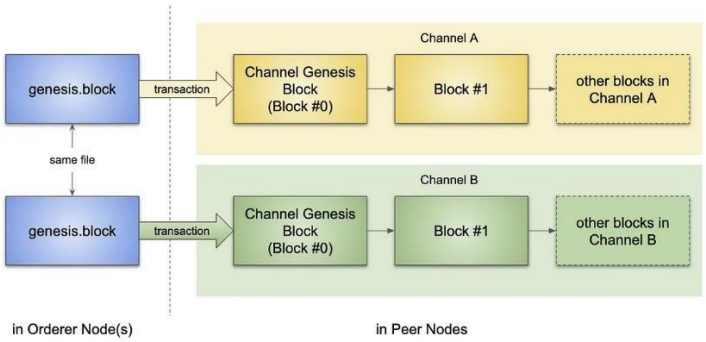

Fig. no. 05. Generic blocks of orderer

Channel configuration transaction: A channel represents a private communication subnetwork within the overall blockchain network. This transaction defines the consortium of organizations that can participate in a particular channel. In this case, it would define the utility company and the authorized prosumers that can interact on the microgrid channel. This transaction defines the channel properties and policies.[6][7]

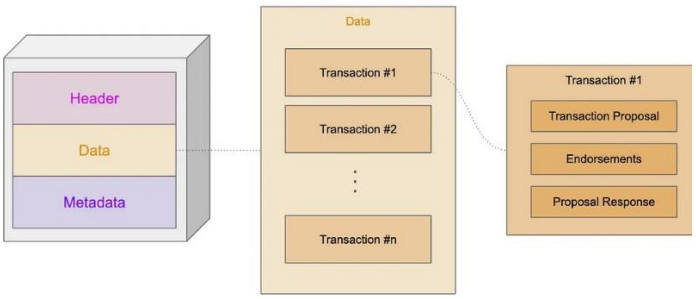

Fig. no. 06. Last anchor peer transaction

Anchor Peer Transactions: Each peer organization submits an anchor peer transaction to establish its identity and network address within the channel. This transaction sets up communication channels between peers in different organizations.[7]

Creating the channel: Once you have the Orderer Genesis block, the channel configuration transaction and the anchor peer transactions, you can submit them to the Orderer service to create a new channel on the blockchain network. This initializes the channel and makes it ready for peer participation.

Peer joins the channel: Each peer node joins the channel by sending a join request to the ordering service. Once the request has been approved, the peer node receives the channel configuration and becomes a participant in the channel.[7]

Use of chaincode: Chaincode, essentially smart contracts written in a language like Go or Chaincode (specifically for Hyperledger Fabric), defines the business logic for your microgrid application. This includes rules for trading energy, managing control signals for DERs (Distributed Energy Resources) and ensuring secure financial transactions. The chaincode is deployed on each peer node that participates in the microgrid channel.[7]

5.1 Implementation on Docker

To implement a blockchain-enabled smart controller for grid-connected microgrids using Docker, first create Docker containers for each component of the system, including the blockchain network nodes (peers, ordering service), databases (e.g. CouchDB) and any client applications or tools required. Docker Compose can be used to define and orchestrate the multi-container setup by specifying the configuration of each component and their interactions.

Once the Docker Compose file is defined, create the Docker images for each component using Dockerfiles and ensure that all requireddependencies and configurations are included. Once the Docker images are ready, you can use Docker Compose to start the containers, creating an isolatedand portable environment for running the blockchain-enabled smart controller.[6][7]

Docker's containerization technology ensures consistency across environments, simplifies deployment and scaling, and facilitates efficient management of resources. In addition, Docker's networking capabilities enable communication between containers within the same network, enabling seamless interaction between the blockchain network nodes and other system components.

5.2 Chaincode

It implements a Hyperledger Fabric Chaincode for the management of energy records in a microgrid. It defines structures for storing energy consumption/production data for each house at a given time. The chaincode provides functions to:
add new energy records (appendRecord), retrieve a specific record by its key (getRecord), retrieve all records within a date-time range categorized by bids (positive for supply, negative for demand) (getBidsByRange), and optionally retrieve all records within a date-time range or all records if no range is specified (getRecordsByRange). It uses JSON for data serialization and uses the Hyperledger Fabric APIs to interact with the state ledger. It demonstrates the key functions for managing energy data in a permissioned blockchain network designed for a grid-connected microgrid system.[7]

It also defines unit tests for the chain code (EnergyRecords) written for the microgrid energy management system. The tests use a mock stub (shim.NewMockStub) to simulate interactions with the Hyperledger Fabric Ledger. It implements unit tests for a Hyperledger Fabric chaincode designed to manage energy records in a microgrid system. It uses a mock stub (shim.NewMockStub) to simulate interaction with the ledger.

It allows agents to submit bids (price and quantity to buy or sell energy) and the chain code calculates the Market Clearing Price (MCP) and Bus Clearing Price (BCP) based on these bids. The code defines structures for bids and uses ledger queries to retrieve bids within a time window. It also contains functions

to initialize the chain code and to handle various functions called by external applications.

5.3 Threat Modelling:

Threat modeling is the process of identifying and assessing potential security threats to the system. This process usually involves the analysis of various attack vectors, such as unauthorized access to the blockchain network, manipulation of transaction data, denial of service attacks and the exploitation of vulnerabilities in smart controllers. By identifying these threats, security measures can be implemented to mitigate the risks, e.g. encryption techniques, access controls, consensus mechanisms and monitoring systems. In addition, threat modeling helps develop robust security protocols to protect the integrity, confidentiality and availability of the microgrid system and its blockchain infrastructure.[8]

5.4 Real-World Testing:

To ensure the system's suitability for practical use after the initial test phases have been completed, a practical test in an operational context is of crucial importance. In this phase, the intelligent controller is integrated into a functional microgrid connected to the power grid so that it can function in the complicated processes of a production environment. Its performance, security and resilience to potential blockchain threats are continuously monitored.[3][6]

It simulates two houses bidding for a smart contract on a blockchain network. The script defines the connection details and then uses a loop to generate random bids four times for each house. It creates unique keys for each transaction and calls the smart contract to append these records. Finally, it queries the smart contract to retrieve all records, records within a specified time period, and bids within a specified time period for both houses.

It also simulates a microgrid market with agents submitting bids (price and quantity to buy or sell energy). It defines agents of different types (PV, EV, HVAC) and generates random bids for them. The code interacts with a smart contract called "mcp", which is deployed on a Hyperledger Fabric blockchain network. In the first part, random bids are submitted for different agents over four rounds, in the second part, predefined test cases are executed with specific bids and the results are checked against the expected market-clearing price (MCP) and bus-clearing price (BCP), and finally, more bids are submitted and the contract is queried again to obtain the MCP.[6]

Timestamp	Price	Quantity	Agent	Agent Type	Unit	Result
2024-04-12 18:18:45.763 UTC	96	-49	House01	PV	Wh	Chaincode invoke success
2024-04-12 18:18:48.077 UTC	73	13	House02	EV	Wh	Chaincode invoke success
2024-04-12 18:18:50.055 UTC	48	5	House02	BESS	Wh	Chaincode invoke success
2024-04-12 18:19:21.805 UTC	14	21	House03	HVAC	Wh	Chaincode invoke success
2024-04-12 18:19:22.001 UTC	35	42	House03	EV	Wh	Chaincode invoke success
2024-04-12 18:19:22.165 UTC	55	-2	House04	PV	Wh	Chaincode invoke success
2024-04-12 18:19:22.355 UTC	73	23	House04	HVAC	Wh	Chaincode invoke success
2024-04-12 18:19:53.281 UTC	98	2	House05	BESS	Wh	Chaincode invoke success
2024-04-12 18:19:53.441 UTC	54	-46	House05	PV	Wh	Chaincode invoke success
2024-04-12 18:20:23.636 UTC	17	4	House05	EV	Wh	Chaincode invoke success
2024-04-12 18:20:24.016 UTC	25	48	House06	HVAC	Wh	Chaincode invoke success
2024-04-12 18:20:24.322 UTC	91	29	House06	BESS	Wh	Chaincode invoke success

Table no. 01
The Table captures the details of the various bid transactions invoked within the system.

Key	Amount	Bid	House	Time
240511_182512_H01	8	-13	House01	2024-05-11 18:25:12
240511_182512_H02	88	-7	House02	2024-05-11 18:25:12
240511_182513_H01	75	-11	House01	2024-05-11 18:25:13
240511_182513_H02	79	49	House02	2024-05-11 18:25:13
240511_182514_H01	77	-34	House01	2024-05-11 18:25:14
240511_182514_H02	50	46	House02	2024-05-11 18:25:14

Table no. 02
Result of Deployment of chaincode

6. Conclusion

Strong cybersecurity is more important than ever at a time when our energy system depends more and more on smoothly functioning grid-connected microgrids. As our work highlights, it is critical to address the cybersecurity challenges of these critical energy systems. It details the complex elements,software solutions and design considerationsrequired to ensure the safe, reliable and continuous operation of grid-connected microgrids in an interconnected world.

Grid-connected microgrids, which offer resilience, sustainability and adaptability, are a creative answerto many of the energy industry's problems. But as their function grows and changes, the threats are becoming more sophisticated and complex. The exploration of a blockchain-enabled smart controller in our publication recognizes the close relationship between the security and performance of these systems. A solid foundation for blockchain comes from the fundamental design considerations, which include hyperledger fabric, fabric binary, secure communications, redundancy, threat intelligence and regulatory compliance. By providing a layered defense against a variety of cyber threats, these components ensure the availability, confidentiality and integrity of vital energy infrastructure.

7. Future Directions

Performance evaluation: The review could providea comprehensive performance evaluation of the proposed cybersecurity-enabled intelligent controller for grid-connected microgrids based on various criteria such as security, reliability, efficiency and scalability. The evaluation could compare the proposed solution with existing approaches and highlight its advantages and limitations.

Hardware solutions: Hardware security modules (HSM) can improve key management and encryption and protect the integrity andconfidentiality of data in transit. This includes amicrocontroller unit (MCU) as the brain of the system, which collects data, executes control algorithms and communicates with other devices. Depending on the configuration, the actuators can control current transformers or battery systems or even manage controllable loads.

REFERENCES

[1] Maria Luisa Di Silvestre , Pierluigi Gallo , Mariano Giuseppe Ippolito , Eleonora Riva Sanseverino , Gaetano Zizzo, "A Technical Approach to the Energy Blockchain in Microgrids" , IEEE Transactions on Industrial Informatics , November 2018

[2] Andrija Goranović , Marcus Meisel , Stefan Wilker, "Blockchain applications in microgrids an overview of current projects and concepts" , IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society , December 2017

[3] Gijs vanLeeuwen, Tarek AlSkaif, Madeleine Gibes cu, Wilfried van Sark, "An integrated blockchain-based energy management platform with bilateral trading for microgrid communities" , Applied Energy , April 2020,

[4] Zhuoli Zhao, Juntao Guo, Xi Luo, "Energy Transaction for Multi-Microgrids and Internal Microgrid Based on Blockchain" , IEEE Access, August 2020

[5] Jiawei Yang, Amrit Paudel, Hoay Beng Gooi, "Compensation for Power Loss by a Proof-of-Stake Consortium Blockchain Microgrid" , IEEE Transactions on Industrial Informatics, July 2020

[6] Yu-Chung Tsao, Vo-Van Thanh, "Toward sustainable microgrids with blockchain technology-based peer-to-peer energy trading mechanism: A fuzzy meta-heuristic approach" , Renewable and Sustainable Energy Reviews , February 2021

[7] KC Tam, "Transactions in Hyperledger Fabric", Medium , July 2019

[8] Antonio Banda, Matthew Hamilton, Eileen Lowry, John Widdifield, "An introduction to building a blockchain solution" , IBM Blockchain, June 2020