Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 03, March-2023

Decentralized IoT Security Gateway System

Dr. Janani A[1], Fabian Ferno[2], Joeshiba K[3] , Shirley Christabel[4]
[1]Associate Professor, Dept. of Information Technology, Loyola ICAM College of Engineering & Technology, Tamil Nadu, India
[2,3,4]B.Tech Information Technology, Loyola ICAM College of Engineering and Technology, Tamil Nadu, India

Abstract:- **The Internet of Things (IoT) has seen widespread adoption in recent years, connecting numerous devices to the internet. However, current IoT networks are vulnerable to various security threats such as data breaches, unauthorized access, and cyber-attacks. To address these security challenges, we propose the Riot protocol - a comprehensive solution for securing IoT networks by providing device authentication, data encryption, decentralized key generation, scalability, and more with cryptographic wallet-based authentication. This protocol ensures the confidentiality, integrity, and authenticity of the data exchanged between IoT devices (publishers) or users (subscribers). The Riot protocol aims to provide a secure and reliable communication between devices, offering an extensive solution to the challenges of IoT security using blockchain technology.**

Key Words: ***Blockchain, CIA triad, Encryption, dApp, Smart Contracts, Cryptographic Salt, Device Signature, Internet of things.***

1.INTRODUCTION

The Internet of Things (IoT) has seen rapid growth in recent years, connecting a multitude of devices to the internet and enabling a range of new applications and services. IoT has the potential to revolutionize many aspects of our lives, from healthcare to smart homes and cities [6]. However, this growth has also brought new security challenges, as these interconnected devices can be vulnerable to various security threats such as data breaches, unauthorized access, and cyber-attacks. In order to unlock the full potential of IoT while ensuring its security, it is imperative to provide secure and reliable communication between IoT devices and its users.

The Riot protocol is the proposed solution for securing IoT networks by leveraging blockchain technology [3]. It offers device authentication, data encryption, data integrity, scalability, and interoperability through cryptographic wallet integration, thereby an end-to-end solution for IoT security [3]. The device authentication process involves using a decentralized key generation system to generate the Riot key [1]. It uses symmetric encryption on the Riot key to encrypt the data exchanged between IoT devices and users, and data integrity is ensured by validating the data signature using the user's private key from their cryptographic wallet [13]. The use of decentralized transaction and data management provides anonymity, safety, data integrity, scalability, interoperability, and an end-to-end solution for IoT security [4]. It aims to provide secure and reliable communication between IoT devices, offering a comprehensive solution to the challenges of IoT security [12].

2. PLATFORM ARCHITECTURE

The Riot protocol is a blockchain-based platform designed to secure IoT networks. Its architecture consists of several components, including a front-end dApp for user interaction, a blockchain node provider for accessing the blockchain, a Device Auth SDK for firmware compatibility, a database layer (ideally using decentralized storage like IPFS), a programmable blockchain layer for smart contract deployment, and a cryptographic wallet for secure user authentication. The platform uses “Riot keys” generated from token ingredients for device authentication and data encryption, ensuring secure and reliable communication between IoT devices and users while providing data integrity and collusion resistance.

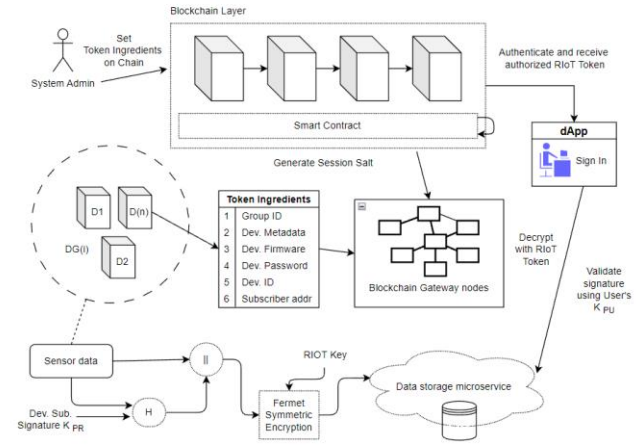

Fig 1: System Architecture

2.1 Security Workflow

1. Each device in the system is previously registered using descriptive parameters like firmware hash, device id, manufacturer metadata, device subscriber address, etc - henceforth called token ingredients.
2. Before streaming sensor data, the riot key is generated by requesting the decentralized key generator using the token ingredients.
3. Riot key is used in the encryption module to encrypt the digitally signed data payload.
4. The device user (subscriber) can request the riot key to the key-generating smart contract as their subscriber address exists in the state of the blockchain.
5. Upon receiving the key, the user can decrypt the payload and hash verify the signature for data integrity validation.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 03, March-2023

2.2 Decentralized Application (dApp)

In the context of the Riot protocol, a front-end dApp, or a decentralized application, is responsible for enabling users to interact with the smart contract on the blockchain [11]. It serves as a user interface for managing the IoT devices that are registered (minted) on the blockchain smart contract or to gain access to the device data decryption. The front-end dApp has an interface that could allow users to request and manage their corresponding Riot keys using their device subscriber address. Once the key is obtained, the user can then use it to decrypt the encrypted data sent by their IoT devices, validate the device's signature using their private key, and view the decrypted sensor data payload. The front-end dApp also enables system administrators to manage devices by updating the token ingredients on-chain, such as device ID, group ID, and firmware hash for each device. In addition, the dApp could display the user's device history and usage statistics for their IoT devices. Overall, the front-end dApp is an essential component of the Riot platform, as it enables users to easily manage and interact with their IoT devices on the blockchain in a secure and decentralized manner [11].

2.3 Blockchain Node Provider

A blockchain node provider offers access to a network of nodes on a blockchain, allowing developers to interact with the blockchain without running their own nodes. This helps integrate blockchain functionality into applications and deploy smart contracts. Popular node providers include Alchemy, Infura, and QuickNode.

2.4 Device Auth SDK

The device firmware SDK (software development kit) is used for developing firmware for IoT devices that are compatible with the Riot protocol. With the device firmware SDK, developers can build firmware that communicates with the Riot protocol and can be registered as a device on the blockchain which helps ensure that the firmware running on a device is legitimate and has not been tampered with. It uses cryptographic techniques such as digital signatures and the delivery of the device token ingredients to the microservices that would communicate with the smart contract. The SDK essentially authenticates and verifies the legitimacy of the firmware on the device.

2.5 Data store

Post encryption, data can be stored in any data store such as MySQL, MongoDB, etc. Ideally, with the intention of making this platform truly decentralized, a database alternative that uses decentralized storage - IPFS [12] - Interplanetary File System - for eg. Orbit DB.

2.6 Programmable Blockchain Layer

In the context of the Riot protocol, a smart contract is used to automate the execution of predefined rules and regulations for the interaction between various IoT devices. It acts as a self-executing contract with the terms of the agreement between the devices written in the code. The smart contract is deployed on the blockchain, ensuring its transparency and immutability [1]. It also eliminates the need for intermediaries, making the transaction process faster, cheaper, and more secure. In summary, smart contracts help to enforce trust, reduce transaction costs, and enhance the overall efficiency of the Riot protocol. It allows programmability on the blockchain. It contains one of the most important modules, the key generator module which resides on a blockchain.

Fig 2: Cryptographic Software & Hardware Wallet

2.7 Cryptographic Wallet

A cryptographic wallet is an important component in the Riot platform as it enables users to securely store and manage their digital assets, such as cryptocurrencies and tokens [2]. It securely holds the private key of the user who holds the account, thereby allowing them to sign the transactions on the smart contract. It seamlessly integrates with the Dapp to let the users authenticate and get access which is a fundamental aspect of the Riot platform's functionality.

3. KEY, TOKEN GENERATION, AND CRYPTOGRAPHIC FUNCTIONS

The key and token generation in the Riot project utilize cryptographic functions to ensure secure and robust authentication. The device token ingredients, including group ID, manufacturer metadata, firmware signature, device ID, and device subscriber address, are used as input to generate a session salt hash with an oracle. This session salt hash, along with the individual hashes of token ingredients, are combined to create a merkle hash, which serves as the Riot key. The use of cryptographic techniques in the key and token generation process ensures the integrity, confidentiality, and authenticity of the generated keys and tokens, contributing to the overall security of the Riot platform.

3.1 Token ingredients

The riot key is generated by the following token ingredients that aim to secure the system on multiple layers of the IoT data lifecycle. They are listed below.

Table -1: Riot Key - Token Ingredients.

Ingredient	Source	Description
Group ID	Session / Region	A unique identifier assigned to a group/region/zone. Eg. Devices in a particular room, zone, etc.
Session Salt	Smart Contract	A random hash generated using oracle networks assigned for a

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 03, March-2023

		device by on-chain smart contract methods.
Manufacturer metadata	Physical	JSON data with device specifications like serial_no, model number, manufacturer details, etc.
Firmware signature	Application	The firmware signature is used to detect the integrity of the official vendor who issued it.
Device ID	Physical	A unique identifier that is used to identify the device itself, issued by the smart contract.
Device subscriber address	Session	the public wallet address of the user who will be consuming the data.

Upon a device is done produced and deployed, a Merkle hash of these ingredients is stored in Blockchain smart contract [3]. The ingredients are securely used to create the device entities on the smart contract to the collection of devices. Each new device added shall be mapped to a non-fungible entity on-chain.

3.2 Salt Generator

A cryptographic salt can be defined as random data that is used as an additional input to a one-way function that hashes data, a password, or a passphrase [13] . In the context of the Riot platform, a session salt is a random string of characters that is generated by the network for each session of the devices. It is used in combination with the other token ingredients (device ID, group ID, firmware hash, device password, and device subscriber address) to create the Merkle root hash, which is referred to as the 'Riot key' [9].

The use of a group salt in this way adds an additional layer of security to the Riot protocol by making it more difficult for attackers to precompute hashes and launch attacks against the system. Given h = H(t) where t is the token, H() is a cryptographic hashing function, and h is the resulting hash output, the following characteristics exist for well-accepted and proven hash functions:

• Computing h from s should be computational simple while deriving any s from a given h should be computationally difficult.

• There should be an extremely low probability of h1 = h2 given s1≠ s2.

• Any identical inputs s1 = s2 should generate identical outputs h1 = h2 [13].

This function generates a random number using the current timestamp, sender address, and block number. It then converts the number into a bytes32 data type, which is used as the session salt. The function returns the salt to be used in combination with the other token ingredients to create the Merkle root hash [9].

3.3 RIOT key generator

The Riot protocol uses Fernet symmetric encryption algorithm to encrypt the sensor data along with its digital signature using the Riot key as the key [12]. Riot Key Generator is responsible for generating symmetric keys that are used to encrypt and decrypt data in the Riot protocol. Programmatically, the key generator function takes in the Riot device token ingredients and the corresponding session salt [13], as inputs and generates a symmetric key using the PBKDF2 algorithm with the SHA-256 hash function. The symmetric key is then used to initialize a Fernet object, which can be used to encrypt and decrypt data using the generated key.

By using symmetric encryption, the Riot protocol eliminates the need for a complex key management system and reduces the computational overhead of encrypting and decrypting data [1]. Additionally, Fernet encryption provides a high level of security and confidentiality for the data exchanged between devices and the system.

Overall, the combination of device authentication through Riot keys and secure data exchange through Riot keys provides a robust security framework for IoT devices in the Riot protocol.

Code Snippet: Smart Contract Generator method

```
function generateRiotKey(... tokenIngredients) public view returns (bytes32) {
  // Verify that the token ingredients match the token data
…
  // Generate session salt hash
…
  // Compute the individual hashes of all the token ingredients
…
  // Create a Merkle hash using the group salt hash and the individual hashes of all the token ingredients
  bytes32[] memory merkleHashElements = new bytes32[](6);
  merkleHashElements[0] = sessionSaltHash;
  merkleHashElements[1] = tokenIdHash;
  merkleHashElements[2] = groupIdHash;
  merkleHashElements[3] = manufacturerMetadataHash;
  merkleHashElements[4] = firmwareSignatureHash;
  merkleHashElements[5] = deviceIdHash;
  bytes32 riotKey = merkleize(merkleHashElements);
  return riotKey;
}
```

3.4 Collusion resistance

IoT data security is critical in ensuring that the data collected from devices is protected from unauthorized access, modification, and theft. A significant approach to IoT data security is collusion resistance - a security measure that aims to prevent multiple parties from collaborating to compromise the system. This is achieved by ensuring that the security of the system does not depend on the trustworthiness of any single party. In the context of our project, collusion resistance is achieved through the use of individual device keys generated through the smart contract method based on the device's unique attributes ensuring that each device has its own unique key, making it difficult for colluding devices to collectively compromise the security of the platform.

Furthermore, the decentralized nature of the platform ensures that there is no single point of failure or control, making it difficult for malicious actors to collude and compromise the security of the platform [8].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 03, March-2023

3.5 Encryption/Decryption Module

The encryption/decryption module is responsible for securing the IoT sensor data before it is stored in the data storage microservice. Fernet symmetric encryption is used for encryption, which provides a high level of security and speed. It uses a shared secret key to both encrypt and decrypt data [13].

Version 1 byte	Timestamp 8 bytes	IV 16 bytes	Ciphertext n*16 bytes	HMAC 32 bytes

Table -2: Fernet Encryption Segmentation

Encryption: The signature and the sensor data payload are combined to form a hash. The module takes the sensor data payload and the generated Riot key as the input. The hash and the original data are then encrypted using Fernet symmetric encryption. The encrypted data is stored in the data storage microservice or could be sent to the communication channels that need this data.
Decryption: The module takes the encrypted data and the Riot key as input. The Riot key is used as the decryption key using Fernet symmetric decryption. The decrypted data is then unconcatenate into the signature and original data. The signature is verified using the device subscriber's key using the cryptographic wallet [13]. If the signature is valid, the original data is decrypted.
This encryption/decryption module provides a high level of security for the IoT sensor data by encrypting the data with the Riot key. Additionally, digital signatures are used to verify the integrity of the sensor data.

3.6 Signature validation

In our project, we use digital signatures to ensure data integrity. They are created using the private key of the signer and can only be verified using the corresponding public key. To validate the integrity of data, we first create a hash of the original data and then sign the hash using the private key of the device subscriber. This creates a unique digital signature for the data that can be used to verify using the provided public key to ensure that the data has not been tampered with. This process helps to prevent masquerade attacks and ensures that the data being accessed is authentic and has not been modified by an unauthorized party.

3.7 Key invalidation cases

There are several scenarios where the RIOT key may be invalidated or rendered unusable. These include
Device tampering: If the device is tampered with, the RIOT key may become compromised or invalidated. This can occur if the device is physically compromised or if its firmware is modified or corrupted. The protocol immediately identifies this case because of the token ingredient that includes the firmware hash and the device's manufacturer metadata.
Riot key compromised: If the RIOT key is compromised by an attacker, it can no longer be used to secure data. This can occur if an attacker gains access to the key, either through a vulnerability in the device or through a data breach. This is also avoided by using cryptographic wallets and smart contracts to generate the Riot keys making it impossible to generate the keys on their own.
Token expiration: Riot keys are typically designed to have a limited lifespan for each session. If the token expires, it can no longer be used to secure data.
User revocation: If a system admin revokes access to a device or their data, the Riot key associated with that user may become invalidated.
System upgrades: If the device firmware is upgraded or changed, existing Riot keys may no longer be valid or compatible with the new system due to the change in the firmware hash.

4. IMPLEMENTATION

To realize the proposed Riot protocol, several key components were developed and integrated to form a cohesive platform. This section outlines the primary steps involved in the implementation of the Riot protocol for the IoT security platform.

4.1 Smart Contract Deployment: The smart contract was developed using Solidity and deployed on the Polygon network. It manages device registration, validation, and key generation, with extensive testing conducted to ensure security and robustness.

4.2 Device Firmware Integration: An SDK was created to streamline the incorporation of the Riot protocol into IoT devices' firmware. The SDK facilitated the integration of cryptographic functions, communication protocols, and device registration processes, ensuring compatibility with a wide range of devices and development platforms.

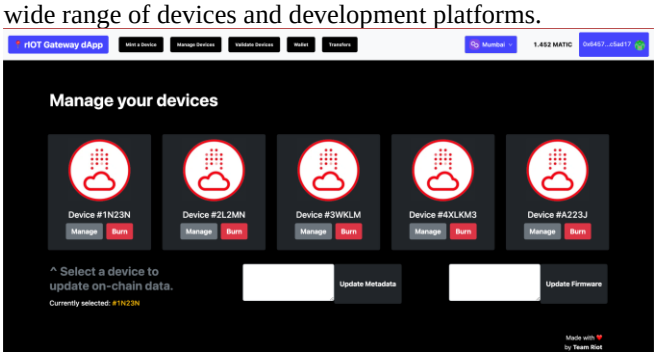

Fig 3: Dapp Front End

4.3 User Interface Development: A user-friendly decentralized application (dApp) was designed for end-users to manage their IoT devices, request Riot keys, and access decrypted data. The dApp integrated seamlessly with cryptographic wallets for secure user authentication and key management.

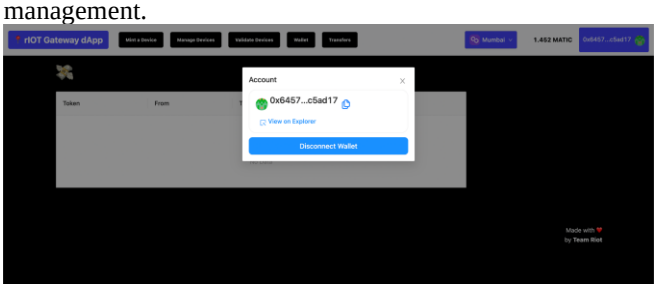

Fig 4: Dapp + Wallet Connection

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 03, March-2023

4.4 Data Storage Selection and Implementation: A suitable database technology was selected and implemented for storing encrypted sensor data. In addition, a decentralized storage solution such as IPFS was explored to enhance the platform's decentralization and resilience against data tampering or loss. The encryption/decryption module was implemented using Fernet symmetric encryption to secure IoT sensor data. This module was designed to work seamlessly with the generated Riot keys, providing a high level of security and performance.

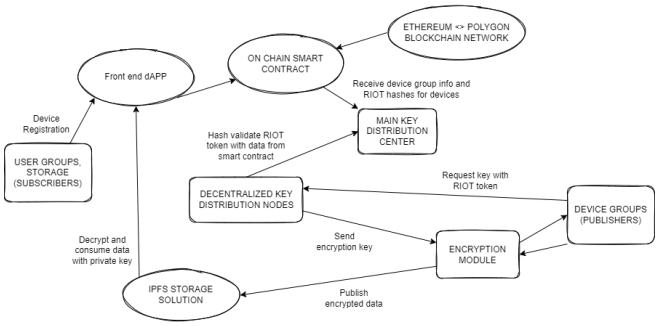

Fig 5: Data Flow - Level 2 Diagram

4.5 Platform Testing and Evaluation: Comprehensive testing and evaluation were conducted to ensure the platform's functionality, security, and performance. Rigorous unit testing, integration testing, and stress testing were performed, and the platform was benchmarked against alternative IoT security solutions.

5. CONCLUSIONS

The Riot protocol is a novel approach to securing Internet of Things (IoT) devices through the use of blockchain technology. By leveraging smart contract functionality and cryptography, the Riot protocol provides a secure and verifiable mechanism for transmitting and storing data from IoT devices. This project demonstrates the potential of the Riot protocol to address the security challenges facing IoT devices, while also highlighting the important role of blockchain technology in creating secure and trustworthy systems.

Following the progress of the paper, the future scopes of the project is extensive. Addressing emerging security threats by incorporating advanced cryptographic techniques and privacy-preserving technologies to ensure user data privacy without compromising the platform's performance.

Scalability: Further improving the scalability of the Riot protocol to accommodate the exponential growth of IoT devices and handling their data communication efficiently.

Energy efficiency: Optimizing the system for energy efficiency and resource-constrained environments, allowing the Riot protocol to be more suitable for low-power IoT devices.

Interoperability: Investigating ways to facilitate seamless integration of the Riot protocol across different IoT platforms and standards, thus promoting widespread adoption.

Collaboration: Fostering collaboration between various stakeholders in the IoT ecosystem, including manufacturers, developers, and regulators, to ensure a unified approach to IoT security.

However the Riot platform comes with its own challenges to be applied in mainstream industries. They are:

Adoption: Encouraging IoT device manufacturers and developers to adopt the Riot protocol and integrate it into their products, which may require overcoming resistance to change and demonstrating the benefits of the platform.

Performance trade-offs: Balancing security and privacy enhancements with the platform's performance, ensuring that new features do not negatively impact system efficiency and speed.

Regulatory compliance: Ensuring that the Riot protocol adheres to evolving data privacy and security regulations across different jurisdictions, which may require continuous updates and modifications to the platform.

User education: Educating end-users about the benefits of the Riot protocol and guiding them through the process of using the platform effectively, which may require the development of user-friendly documentation and training materials.

Advanced attack vectors: Preparing for and addressing sophisticated attacks that may target the Riot protocol, including the potential exploitation of vulnerabilities in the underlying blockchain and cryptographic algorithms.

In conclusion, Riot is a groundbreaking project that uses blockchain, smart contracts, and cryptography to enhance IoT security. With its innovative approach, Riot offers a decentralized and secure infrastructure for IoT devices, holding potential for revolutionizing IoT authentication and security.

REFERENCES

[1] Maissa Dammak, Sidi-Mohammed Senouci, Mohamed Ayoub Messous, Mohamed Houcine Elhdhili, Christophe Gransart, 2020. “Decentralized Lightweight Group Key Management for Dynamic Access Control in IoT Environments” IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, VOL. 17, NO. 3, SEPTEMBER 2020. ieeexplore.ieee.org/document/9119178

[2] M. Dammak, O. R. M. Boudia, M. A. Messous, S. M. Senouci and C. Gransart, "Token-Based Lightweight Authentication to Secure IoT Networks" 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2019, pp. 1-4. doi: 10.1109/CCNC.2019.8651825.

[3] Jawad Ali, Toqeer Ali &, Yazed Alsaawy, Ahmad Shahrafidz Khalid and Shahrulniza Musa. 2019. “Blockchain-based Smart-IoT Trust Zone Measurement Architecture. INTERNATIONAL CONFERENCE ON OMNI-LAYER INTELLIGENT SYSTEMS (COINS)”, May 5–7, 2019, Crete, Greece. ACM, NewYork, NY, USA, 6 pages. doi.org/10.1145/3312614.3312646

[4] Jollen Chen. 2017. “Devify: Decentralized Internet of Things Software Frame-work for a Peer-to-Peer and Interoperable IoT Device”. In Proceedings of Advances in IoT Architecture and Systems, Toronto, Canada, June 2017 (AIo-TAS’17), 6 pages. sigbed.seas.upenn.edu/archives/2018-03/paper4.pdf

[5] Kazım Rıfat Özyılmaz and Arda Yurdakul. 2017. “Work-in-Progress: Integrating Low-Power IoT devices to a Blockchain-Based Infrastructure”. In Proceedings of EMSOFT’ 17 Companion, Seoul, Republic of Korea, October 15–20, 2017, 2 pages. doi.org/10.1145/3125503.3125628

[6] Nada Alasbali, Saaidal Razalli Azzuhri, Rosli Salleh. 2020. “A Blockchain-Based Smart Network for IoT-Driven Smart Cities”. IECC 2020, July 8-10, 2020, Singapore. doi.org/10.1145/3409934.3409957

[7] Ali Dorri; Salil S. Kanhere; Raja Jurdak; Praveen Gauravaram. 2017. “Design and Development of Blockchain-Based Decentralized IoT Platform. IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)”. 13-17 March 2017.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 03, March-2023

[8] S. Nakamoto. Bitcoin: a peer-to-peer electronic cash system https://bitcoin.org/bitcoin.pdf (2008), Accessed 23rd Dec 2021 Google Scholar
[9] Yin-qing Fang; Jian-bin Liao; Lian-you Lai, 2020. Verifiable Secret Sharing Scheme Using Merkle Tree.2020 International Symposium on Computer Engineering and Intelligent Communications (ISCEIC). doi.org/10.1109/ISCEIC51027.2020.00008
[10] W. Yang, E. Aghasian, S. Garg, D. Herbert, L. Disiuta and B. Kang, "A Survey on Blockchain-Based Internet Service Architecture: Requirements, Challenges, Trends, and Future," in IEEE Access, vol. 7, pp. 75845-75872, 2019, doi: 10.1109/ACCESS.2019.2917562.
[11] Lodovica Marchesi, Michele Marchesi, Roberto Tonelli, Maria Ilaria Lunesu. 2022, p "A blockchain architecture for industrial applications. Blockchain: Research and Applications 3 (2022)," doi.org/10.1016/j.bcra.2022.100088
[12] J. Sun, X. Yao, S. Wang and Y. Wu, "Blockchain-Based Secure Storage and Access Scheme For Electronic Medical Records in IPFS," in IEEE Access, vol. 8, pp. 59389-59401, 2020. doi: 10.1109/ACCESS.2020.2982964
[13] A. D. Kent and L. M. Liebrock, "Secure Communication via Shared Knowledge and a Salted Hash in Ad-Hoc Environments," 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops, Munich, Germany, 2011, pp. 122-127. doi: 10.1109/COMPSACW.2011.30