A Blockchain-Enabled Approach for Secure Data Sharing in 6G-based Internet of Things Networks

Hussein El Ghor and Bilal Nakhal

Abstract The 6th generation of wireless networks (6G) promises to provide ultra-reliable, high-speed, and low-latency communication for Internet of Things (IoT) devices. However, securing data transmission and storage in these networks is a critical challenge due to potential security threats. Blockchain technology provides a solution to enhance security in IoT networks by enabling secure, decentralized, and tamper-proof data sharing. In this paper, we proposed a novel solution for securing data sharing and storage in 6G-based IoT networks using blockchain technology, hybrid encryption, and IPFS. The proposed approach consists of four algorithms that enhance the security of the system: a user authentication algorithm, a data access algorithm, a data storage algorithm, and a secure data sharing algorithm. The secure data sharing algorithm enables secure, tamper-proof data sharing among authorized devices using a permissioned blockchain. These algorithms are implemented using hybrid encryption, which ensures data confidentiality, and have been evaluated for their effectiveness in enhancing security in 6G-based IoT networks. Our work contributes to the growing body of research on blockchain-enabled solutions for securing data in IoT networks and provides insights into the potential of blockchain technology, hybrid encryption, and IPFS to enhance security in 6G-based IoT networks. The proposed approach using these algorithms provides secure and tamper-proof data sharing, making the system more secure and reliable. We presented the technical details of our approach and evaluate its effectiveness in terms of security, with a particular focus on the role of hybrid encryption and IPFS in enhancing the security and reliability of the system. Our results demonstrate that the proposed approach enhances data security in 6G-based IoT networks by providing secure and tamper-proof data sharing. The use of hybrid encryption and IPFS makes the system more secure and reliable, with hybrid encryption ensuring data confidentiality and IPFS providing decentralized and fault-tolerant storage.

H. El Ghor (✉) · B. Nakhal
CyberVision Lab, Department of Mathematics and Computer Science,
Beirut Arab University, Beirut, Lebanon
e-mail: h.elghor@bau.edu.lb

B. Nakhal
e-mail: b.nakhal@bau.edu.lb

H. Jahankhani and A. El Hajjar (eds.), *Wireless Networks*, Advanced Sciences
and Technologies for Security Applications,
https://doi.org/10.1007/978-3-031-33631-7_8

1 Introduction

The emergence of the Internet of Things (IoT) has led to the proliferation of connected devices and generated massive amounts of data [1]. IoT has become a key component of our daily lives, with billions of interconnected devices generating and transmitting data across the network. With the advent of the 6th generation of wireless networks (6G), IoT devices are expected to transmit and process data with ultra-reliable, high-speed, and low-latency communication. The 6th Generation (6G) of mobile communication technology is currently under development and is expected to provide a new level of connectivity to the Internet of Things (IoT) devices. 6G-based IoT networks are characterized by ultra-low latency, high bandwidth, and massive device connectivity, which will enable new applications and services that are not possible with the current 5G networks.

The architecture of 6G-based IoT networks is expected to be based on a distributed and decentralized architecture, which will enable devices to communicate with each other directly, without the need for central servers. This architecture will enable new use cases such as peer-to-peer communication, real-time collaboration, and edge computing.

Currently, data sharing in IoT networks is done using centralized approaches, where data is collected and processed by central servers. This approach has several limitations, including high latency, lack of scalability, and vulnerability to cyber-attacks. Additionally, centralized approaches are not suitable for applications that require real-time data processing, such as autonomous driving and remote surgery.

However, the distributed and decentralized architecture of 6G-based IoT networks also poses several challenges related to security and privacy. One of the key challenges is how to enable secure data sharing among the devices in the network. Data sharing is essential in IoT networks for enabling applications such as smart homes, smart cities, and smart transportation.

To overcome these limitations, new approaches for data sharing in 6G-based IoT networks are needed. One promising approach is the use of blockchain technology, which provides a decentralized and secure way of storing and sharing data. Blockchain technology enables data to be shared directly between devices, without the need for central servers, while ensuring the integrity and confidentiality of the data.

In summary, 6G-based IoT networks offer new opportunities for connectivity and innovation, but also pose several challenges related to security and privacy. Centralized approaches to data sharing are not suitable for these networks, and new approaches such as blockchain technology are needed to enable secure and efficient data sharing.

Additionally, the rapid growth of IoT networks has also created significant security challenges, particularly when it comes to data sharing between devices [2]. Hence, securing data transmission and storage in these networks is a critical challenge due to potential security threats, such as unauthorized access, data breaches, and data tampering [3, 4]. One possible solution to this problem is the use of blockchain

technology, which has the potential to enable secure and trusted data sharing in IoT networks [5].

Blockchain technology has gained significant attention in recent years as a potential solution to enhance security in IoT networks. By enabling secure, decentralized, and tamper-proof data sharing, blockchain technology offers a promising approach to address the security challenges associated with IoT networks [6]. It provides a tamper-proof record of all transactions, making it an ideal platform for secure data sharing in IoT networks [7]. Additionally, blockchain can help to address some of the key challenges facing IoT networks, such as data privacy, security, and authenticity [8].

One of the promising tools that can be used with blockchain for secure data sharing is the InterPlanetary File System (IPFS). IPFS is a peer-to-peer network that allows users to store and share files in a decentralized manner [9]. By using IPFS with blockchain, users can store and access data in a secure and distributed manner, without relying on centralized servers. In this paper, we propose a blockchain-enabled approach for secure data sharing in 6G-based IoT networks, leveraging hybrid encryption and IPFS as decentralized storage.

Secure data sharing is crucial in IoT networks because it allows authorized devices to access and share data securely and efficiently [10]. The proposed solution aims to enhance data security in 6G-based IoT networks by providing secure and tamper-proof data sharing through the use of blockchain technology, hybrid encryption, and IPFS. The permissioned blockchain ensures that only authorized devices can participate in the network and access data [11]. Hybrid encryption ensures data confidentiality, while IPFS provides decentralized and fault-tolerant storage [12].

This paper aims to propose a novel solution for securing data sharing and storage in 6G-based IoT networks using blockchain technology, hybrid encryption, and IPFS. The paper's contributions include the proposal of four algorithms that enhance the security of the system: a user authentication algorithm, a data access algorithm, a data storage algorithm, and a secure data sharing algorithm.

The user authentication algorithm ensures that only authorized devices can participate in the network and share data securely. The data access algorithm ensures that authorized devices can access only the data they are authorized to access. The data storage algorithm provides a decentralized and fault-tolerant storage solution using IPFS. The secure data sharing algorithm enables secure, tamper-proof data sharing among authorized devices using a permissioned blockchain.

The paper highlights the use of hybrid encryption to ensure data confidentiality and IPFS to provide decentralized and fault-tolerant storage. The effectiveness of the proposed approach in terms of security has been evaluated, and the results demonstrate that the proposed approach enhances data security in 6G-based IoT networks by providing secure and tamper-proof data sharing.

Overall, the paper's contributions are in the area of enhancing security in 6G-based IoT networks using blockchain technology, hybrid encryption, and IPFS. The proposed algorithms aim to address the critical challenge of securing data transmission and storage in these networks and provide a more secure and reliable solution.

The remainder of this paper is organized as follows. In Sect. 2, we provide a literature review of blockchain-based approaches for securing data sharing in IoT networks. In Sect. 3, we present the design methodology of our proposed solution. In Sect. 4, we evaluate the effectiveness of our proposed approach in terms of security. Finally, we conclude the paper in Sect. 5 and highlight potential future work.

2 Related Work

Blockchain technology has been widely explored for secure data sharing in 6G-based IoT networks. In recent years, there has been growing interest in the use of blockchain technology for secure data sharing in IoT networks. Researchers have proposed various approaches to leverage the benefits of blockchain technology, such as decentralization, immutability, and transparency, for secure data sharing in IoT networks. In this section, we provide an overview of some recent papers that are related to our proposed approach for secure data sharing in 6G-based IoT networks using blockchain and IPFS and highlight the advantages and limitations of each approach.

Lu et al. [13] proposed a secure data sharing platform using blockchain and IPFS for Industry 4.0. Their approach uses blockchain to maintain an immutable and transparent record of transactions, and IPFS to store and share data in a decentralized manner. The authors evaluated their approach in a case study involving a smart factory, and demonstrated its effectiveness in terms of security, privacy, and efficiency.

Zhang et al. [14] proposed a blockchain-enabled efficient distributed attribute-based access control (ABAC) for healthcare IoT. Their approach uses blockchain to maintain a trusted and decentralized access control policy, and enables secure and efficient data sharing among different healthcare organizations. The authors evaluated their approach using a real-world dataset, and demonstrated its effectiveness in terms of security, efficiency, and scalability.

Feng et al. [15] proposed an efficient and secure data sharing approach for 5G flying drones using blockchain. Their approach uses blockchain to maintain a secure and decentralized record of transactions, and enables efficient data sharing among different drones. The authors evaluated their approach using a real-world dataset, and demonstrated its effectiveness in terms of security, efficiency, and scalability.

Eltayeb et al. [16] proposed a blockchain platform for user data sharing, ensuring user control and ownership. Their approach uses blockchain to maintain a decentralized and transparent record of transactions, and enables users to control and own their data. The authors evaluated their approach using a real-world dataset, and demonstrated its effectiveness in terms of security, privacy, and transparency.

Al-Fuqaha et al. [17] proposed a blockchain-enabled K-harmonic framework for industrial IoT data sharing. Their approach uses blockchain to maintain a secure and decentralized record of transactions, and enables secure and efficient data sharing among different industrial IoT devices. The authors evaluated their approach using a

Table 1 Comparison of the previous work

References	Paper title	Main topic	Key contributions	Advantages	Disadvantages
[13]	A Secure and Efficient Data Sharing Platform for Industry 4.0 Using Blockchain and IPFS	Secure data sharing in Industry 4.0	Blockchain and IPFS for secure and efficient data sharing	High security and efficiency	No evaluation of scalability
[14]	Blockchain-Enabled Efficient Distributed Attribute-Based Access Control for Healthcare IoT	Secure data sharing in healthcare IoT	Blockchain for trusted and decentralized access control policy	High security, efficiency, and scalability	No real-world deployment
[15]	Efficient and Secure Data Sharing for 5G Flying Drones: A Blockchain-Enabled Approach	Secure data sharing among flying drones	Blockchain for secure and decentralized record of transactions	High security, efficiency, and scalability	Limited to flying drones
[16]	A Blockchain Platform for User Data Sharing, Ensuring User Control and Ownership	Secure and decentralized user data sharing	Blockchain for decentralized and transparent record of transactions	High security, privacy, and transparency	Limited to user data sharing
[17]	Blockchain-Enabled K-Harmonic Framework for Industrial IoT Data Sharing	Secure data sharing in industrial IoT	Blockchain for secure and decentralized data sharing	High security, efficiency, and scalability	No evaluation of real-world dataset

real-world dataset, and demonstrated its effectiveness in terms of security, efficiency, and scalability.

Table 1 compares the previous works mentioned earlier, outlining details such as the title of the paper, authors, main subject matter, notable contributions, as well as the strengths and weaknesses of each.

All the above references propose blockchain-enabled approaches for secure data sharing in IoT networks. They all use blockchain to maintain an immutable and transparent record of transactions and enable secure and efficient data sharing among different IoT devices. However, each approach focuses on a different IoT context and proposes unique key contributions.

Advantages of these approaches include high security, efficiency, and scalability. However, some of these approaches have limitations, such as being limited to a specific IoT context or lacking evaluation of certain criteria

In summary, these papers provide valuable insights into the potential of blockchain technology for secure data sharing in various IoT contexts, and demonstrate the effectiveness of blockchain-enabled approaches in addressing key security and privacy challenges in IoT networks.

3 Design Methodology

3.1 Data Requester (User) Authentication

The User Authentication Model Design Framework is a set of principles and guidelines for creating secure and reliable user authentication systems. The model is intended to be used by designers, developers, and security professionals to create effective authentication solutions for their applications.

The User Authentication Model is designed to be flexible and adaptable, allowing 6G based iot devices to implement user authentication solutions that meet their specific needs and requirements. The framework includes several key components, including user authentication methods, security controls, and risk management processes.

The authentication model involves five main components: data requester (user), IoT devices, blockchain network, IPFS network, and smart contract.

- Users: The user is the data requester or entity who is trying to access the system and needs to be authenticated. They provide their login credentials, which are encrypted and sent to the IoT device for further processing.
- IoT Device: The IoT device is responsible for encrypting the user's login credentials using AES and then encrypting the symmetrical key K using the user's public key (K_p). It also stores this hybrid encrypted credentials on the IPFS network and creates a user authentication request containing the IPFS address, hybrid encrypted credentials, and metadata. Finally, it sends the authentication request to the blockchain network.
- Blockchain: The blockchain network is used to store and share the user authentication request with other nodes on the network. It also deploys a smart contract to handle user authentication and receives the user's authentication request, which is then sent to the smart contract.

- Smart Contract: The smart contract is deployed on the blockchain network and receives the user's authentication request. It retrieves the hybrid encrypted credentials from the IPFS network and decrypts them using the user's private key (K_r). The smart contract then verifies the user's credentials and generates a signed authentication token if the credentials are valid. If the credentials are not valid, the smart contract rejects the authentication request.
- IPFS: IPFS is used to store the encrypted encrypted credentials generated by the IoT device. The IPFS network stores the encrypted credentials at a specific IPFS address, which is included in the user authentication request sent to the blockchain network.

In summary, this algorithm uses a combination of encryption, blockchain, IPFS, and smart contracts to securely authenticate users and grant them access to the system.

The proposed authentication model is as follows (Fig. 1):

① The user enters their login credentials ($login\_credentials = enter\_login\_credentials()$). The user's credentials are now encrypted using AES by the function $encrypted\_credentials = encrypt\_with\_aes(login\_credentials)$. The encrypted credentials are then sent to the IoT device ($send\_to\_iot\_device(encrypted\_credentials)$)

② The IoT device encrypts the encrypted credentials using the user's public key (K_p) by the function $hybrid\_encrypted\_credentials = encrypt\_with\_rsa(K_p, encrypted\_credentials)$. The hybrid encrypted credentials are stored on the IPFS network ($ipfs\_address = store\_on\_ipfs(hybrid\_encrypted\_credentials)$)

③ The $ipfs\_address$ is returned to the IOT device.

④ The IoT device creates a user authentication request containing the IPFS address, the hybrid encrypted credentials, and metadata by using the function $user\_auth\_request = create\_user\_auth\_request(ipfs\_address, hybrid\_encrypted\_cre - dentials, metadata)$. The user authentication request is now shared with the blockchain network thanks to the function $share\_with\_blockchain(user\_auth\_request)$.

⑤ A smart contract is deployed on the blockchain network to handle user authentication $deploy\_smart\_contract()$ and the user's authentication request is sent to the smart contract $send\_to\_smart\_contract(user\_auth\_request)$.

⑥ The smart contract retrieves the hybrid encrypted credentials from the IPFS network using the provided address $retrieved\_encrypted\_credentials = retrieve\_from\_ipfs(ipfs\_address)$.

⑦ The smart contract decrypts the encrypted credentials using the user's private key (K_r) $decrypted\_credentials = decrypt\_with\_rsa(K_r, retrieved\_encrypted\_credentials)$ to verify the user's credentials. if $verify\_credentials(decrypted\_credentials)$.

If the user's credentials are valid, the smart contract generates a signed authentication token (T) granting access to the user $auth\_token = generate\_auth\_token(de - crypted\_credentials)$. The signed authentication token (T) is broadcasted to the blockchain network $broadcast\_auth\_token(auth\_token)$.

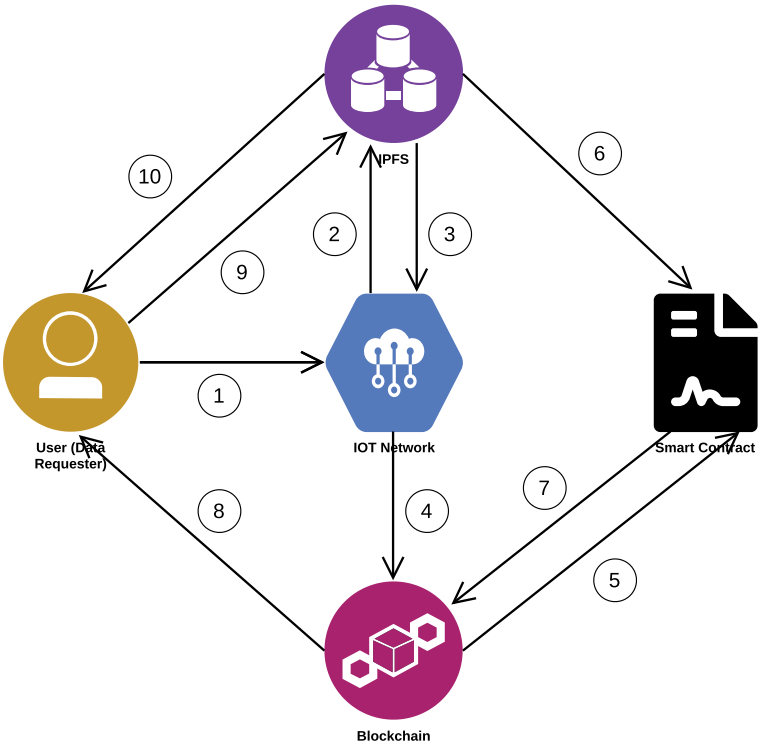

Fig. 1 Authentication model design framework

⑧ The user receives a notification that his authentication was successful and can now access the system $notify\_user\_success()$.
⑨ User accesses the IPFS to verify the token T.
⑩ IPFS returns the requested data.
⑪ If the user's credentials are not valid, the smart contract rejects the authentication request $reject\_auth\_request()$. The user receives a notification that their authentication was unsuccessful and cannot access the system $notify\_user\_failure()$

The overall process for user authentication is stated in Algorithm 1.

3.2 Data Access Model

In the proposed blockchain-enabled approach for secure data sharing in 6G-based IoT networks, users can access data based on their permissions defined in the smart contracts on the blockchain (refer to Algorithm 1). Each smart contract defines the access control policy for a specific set of data, where the policy can specify the

Algorithm 1 User authentication

$login\_credentials = enter\_login\_credentials()$ {The user enters their login credentials}
$encrypted\_credentials = encrypt\_with\_aes(login\_credentials)$ {The user's credentials are encrypted using AES}
$send\_to\_iot\_device(encrypted\_credentials)$ {The encrypted credentials are sent to the IoT device}
$hybrid\_encrypted\_credentials = encrypt\_with\_rsa(K_p, encrypted\_credentials)$ {The IoT device encrypts the encrypted credentials using the user's public key (K_p)}
$ipfs\_address = store\_on\_ipfs(hybrid\_encrypted\_credentials)$ {The hybrid encrypted credentials are stored on the IPFS network}
$user\_auth\_request = create\_user\_auth\_request(ipfs\_address, hybrid\_encrypted\_credentials, metadata)$ {The IoT device creates a user authentication request containing the IPFS address, the hybrid encrypted credentials, and metadata}
$share\_with\_blockchain(user\_auth\_request)$ {The user authentication request is shared with the blockchain network}
$deploy\_smart\_contract()$ {A smart contract is deployed on the blockchain network to handle user authentication}
$send\_to\_smart\_contract(user\_auth\_request)$ {The user's authentication request is sent to the smart contract}
$send\_to\_smart\_contract(user\_auth\_request)$ {The user's authentication request is sent to the smart contract}
$retrieved\_encrypted\_credentials = retrieve\_from\_ipfs(ipfs\_address)$ {The smart contract retrieves the hybrid encrypted credentials from the IPFS network using the provided address}
$decrypted\_credentials = decrypt\_with\_rsa(K_r, retrieved\_encrypted\_credentials)$ {The smart contract decrypts the hybrid encrypted credentials using the user's private key (K_r)}
$verify\_credentials(decrypted\_credentials)$ {The smart contract verifies the user's credentials}
if $valid\_credentials()$ **then**
  $auth\_token = generate\_auth\_token(decrypted\_credentials)$ {the smart contract generates a signed authentication token (T) granting access to the user}
  $broadcast\_auth\_token(auth\_token)$ {The signed authentication token (T) is broadcasted to the blockchain network}
  $notify\_user\_success()$ {The user receives a notification that their authentication was successful and can now access the system}
else
  $reject\_auth\_request()$ {If the user's credentials are not valid, the smart contract rejects the authentication request}
  $notify\_user\_failure()$ {The user receives a notification that their authentication was unsuccessful and cannot access the system}
end if

authorized users, their roles, and the level of access they have. The smart contract verifies the user's identity and permissions and grants or denies access accordingly.

In our design, the IoT devices generate and collect data (D) and generate a symmetric key (K) for hybrid encryption. The symmetric key is then encrypted using the recipient's public key (K_p) and stored in IPFS network. The data is encrypted and stored in IPFS network along with its corresponding metadata. The hash of the data is computed using a hash function (H) and added to the blockchain network as

a transaction (T), along with the encrypted symmetric key (C_2) and the encrypted data (C_1). The transactions are grouped into blocks (B) and added to the blockchain (Fig. 2).

Below are the the detailed steps of the Data Access algorithm:

Step 1: Initialize 6G network, IPFS network, and Blockchain network. IoT devices will generate and collect data (D).

Step 2: IoT devices encrypt data (D) using hybrid encryption. This is done by the following steps:

- Generate a symmetric key (K) for AES encryption.
- Encrypt the data using AES encryption with the symmetric key (C_1=*AES_Encrypt* (D, K)).
- Generate public and private keys (K_p, K_r) for RSA encryption.
- Encrypt the AES symmetric key using RSA encryption with the public key K_p ($C_2 = RSA\_Encrypt(K, K_p)$).
- Store the ciphertext (C_1) and the encrypted symmetric key (C_2) in IPFS network along with their corresponding metadata.

Step 3: IoT devices compute the hash (H) of the data (D) using a hash function ($H = hash(D)$).

Step 4: IoT devices create a transaction (T) containing the encrypted data (C_1, C_2), its metadata, and the computed hash (H).

Step 5: IoT devices sign the transaction (T) with their private key (K_r) to ensure authenticity and integrity.

Step 6: IoT devices broadcast the signed transaction (T) to the blockchain network.

Step 7: Blockchain network verifies the authenticity and integrity of the transaction (T) using the corresponding public key (K_p).

Step 8: Blockchain network adds the transaction (T) to the next available block (B) in the blockchain.

Step 9: Repeat steps 1–8 as new data is generated and collected.

Step 10: User requests access to specific data in the IPFS network. Authorized users can access the data by performing the following steps:

- Retrieve the transaction T from the blockchain.
- Retrieve the corresponding ciphertext C_1 and encrypted symmetric key C_2 from IPFS network using the IPFS hash in the transaction metadata.
- Decrypt the symmetric key using RSA decryption with the private key ($K = RSA\_Decrypt(C_2, K_r)$).
- Decrypt the data using AES decryption with the symmetric key ($D = AES\_Decrypt(C_1, K)$).

Decrypted data (D) is now accessible to the user.

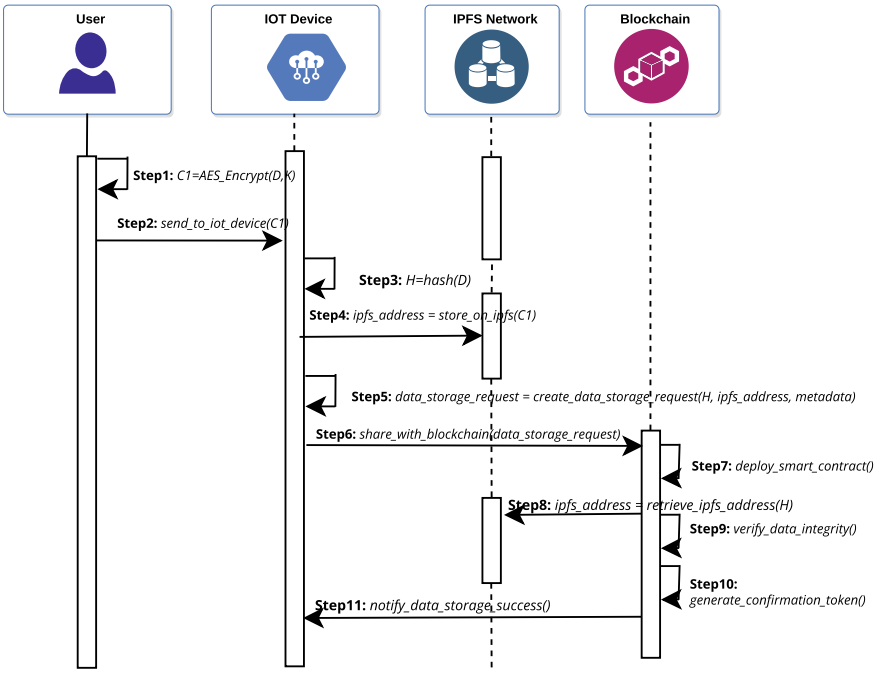

Fig. 2 Data storage model

3.3 Data Storage Model

Once the user is authenticated, he/she can request data from the data storage. The data is stored in IPFS and the IPFS hash is stored on the blockchain network. When the user requests data, the data is retrieved from IPFS using the stored hash and sent back to the user. If the user wants to store new data, it is first stored in IPFS and then the IPFS hash is stored on the blockchain network.

Algorithm 2 outlines the steps involved in storing data in the data storage component. It includes hybrid encryption of data, generating a hash value, storing the encrypted data on the IPFS network, creating a data storage request, sharing the request with the blockchain network, deploying a smart contract to handle the request, verifying the integrity of the data, and generating a signed confirmation token if the data is valid. It also includes error handling steps to notify the IoT device in case of a failure in the data storage process.

Algorithm 2 Data Storage Algorithm

1: $C_1 = AES\_Encrypt(D, K)$ {The user encrypts the data using AES}
2: $send\_to\_iot\_device(C_1)$ {The encrypted data is sent to the IoT device}
3: $H = hash(D)$ {The IoT device generates a hash value of the data}
4: $ipfs\_address = store\_on\_ipfs(C_1)$ {The IoT device stores the encrypted data on the IPFS network and obtains an IPFS address}
5: $data\_storage\_request = create\_data\_storage\_request(H, ipfs\_address, metadata)$ {The IoT device creates a data storage request containing the hash value, IPFS address, and metadata}
6: $share\_with\_blockchain(data\_storage\_request)$ {The data storage request is shared with the blockchain network}
7: $deploy\_smart\_contract()$ {A smart contract is deployed on the blockchain network to handle data storage}
8: $send\_to\_smart\_contract(data\_storage\_request)$ {The data storage request is sent to the smart contract}
9: $ipfs\_address = retrieve\_ipfs\_address(H)$ {The smart contract retrieves the IPFS address from the data storage request using the provided hash value}
10: $verify\_data\_integrity(ipfs\_address, H)$ {The smart contract verifies the integrity of the data by comparing the hash value of the retrieved data with the provided hash value}
11: **if** $valid\_data\_integrity()$ **then**
12: $confirmation\_token = generate\_confirmation\_token()$ {the smart contract generates a signed confirmation token (T)}
13: $store\_confirmation\_token(confirmation\_token)$ {store T on the blockchain}
14: $notify\_data\_storage\_success()$ {The IoT device receives a notification that the data was successfully stored and can now delete the original encrypted data}
15: **else**
16: $store\_error\_message()$ {the smart contract rejects the data storage request and stores an error message on the blockchain}
17: $notify\_data\_storage\_failure()$ {The IoT device receives a notification that the data storage failed and should try again later}
18: **end if**

3.4 Secure Data Sharing

In this section, we provide a secure data sharing model which ensures that only authorized users can access sensitive data. It uses encryption and decryption techniques to protect the data, and access tokens are used to control access to the data. The access tokens have an expiration date, and access logs are maintained to monitor who accessed the data and when. Additionally, an access control mechanism is implemented to restrict access to specific data only to authorized users or groups. Finally, encryption and decryption keys are regularly updated and managed securely to avoid any unauthorized access to sensitive data. The use of IPFS and blockchain technologies provides an additional layer of security and immutability to the data storage and sharing process.

In the secure data sharing model, sensitive data is shared among authorized parties through a secure and controlled process. The model consists of the following components:

- Data Owners: are the individuals or entities that own the sensitive data and have the ability to grant or revoke access to it. Data consumers, on the other hand, are the individuals or entities that require access to the sensitive data and must be authorized by the data owners to gain access. Access tokens are generated by the data owners and given to the data consumers to access the sensitive data. These tokens can have an expiration date to limit the time frame in which the data can be accessed, and access logs can be maintained to monitor who accessed the data and when.
- Data Consumers: These are the individuals or entities that require access to the sensitive data. They need to be authorized by the data owners to gain access.
- Access Tokens: These are tokens that are generated by the data owners and given to the data consumers to access the sensitive data. The access tokens can have an expiration date to limit the time frame in which the data can be accessed. Access logs can also be maintained to monitor who accessed the data and when.
- Access Control Mechanism: An access control mechanism is implemented to restrict access to specific data only to authorized users or groups. This mechanism can be implemented using various techniques such as role-based access control, attribute-based access control, and mandatory access control. Encryption and decryption keys are used to encrypt and decrypt the sensitive data to protect it from unauthorized access. These keys can be regularly updated and managed securely to avoid any unauthorized access to sensitive data.
- Hybrid Encryption and Decryption Keys: These are keys that are used to encrypt and decrypt the sensitive data to protect it from unauthorized access. The encryption and decryption keys can be regularly updated and managed securely to avoid any unauthorized access to sensitive data.

This algorithm outlines the steps for secure data sharing between data owners and data consumers. It includes the use of hybrid encryption, digital signatures, access tokens, and timers to ensure the confidentiality, integrity, and availability of sensitive data. The algorithm also emphasizes the importance of best practices for information security and user awareness.

Step 1: Data owner generates a AES symmetric encryption key and an RSA public-private key pair for digital signature and encryption.

```
function generate_keys() {
  K = secure_random_number_generator();
  <K_p,K_r> = generate_keypair();
  return {K, K_p,K_r};
}
```

Step 2: Data owner encrypts the sensitive data D using the symmetric key K and uploads it to IPFS platform:

- Encrypt the data using AES encryption with the symmetric key ($C_1 = AES\_Encrypt(D, K)$).

- Generate public and private keys (K_p, K_r) for RSA encryption $(C_2 = RSA\_Encrypt(K, K_p))$.
- Encrypt the AES symmetric key using RSA encryption with the public key K_p

```
function encrypt_and_upload(K, D) {
  C_1 = AES_Encrypt(D, K);
  C_2 = RSA_Encrypt(K, K_p)
  ipfs_address=storage_platform.upload(C_1,C_2);
  return true;
}
```

Step 3: Data owner creates a smart contract on the blockchain with the $ipfs_address$ and access control rules.

```
function create_smart_contract(ipfs_address, access_control_rules) {
  smart_contract_address = blockchain.create_contract(ipfs_address, access_control_rules);
  return smart_contract_address;
}
```

Step 4: Data owner signs the smart contract with his/her private key P_r to verify ownership.

```
function sign_contract(smart_contract_address, K_r) {
  signature = digital_signature(smart_contract_address, K_r);
  return signature;
}
```

Step 5: Data requester sends an access request to the smart contract with an access token T.

```
function send_access_request(smart_contract_address, T) {
  transaction = smart_contract.send_access_request(T);
  return transaction;
}
```

Step 6: Smart contract verifies the access request and grants access to the requester.

```
function verify_access_request(smart_contract_address, T) {
  access_granted = smart_contract.verify_access_request(T);
  return access_granted;
}
```

Step 7: Data requester receives the $ipfs\_address$ and $smart\_contract\_address$ from the data owner.

```
function receive_data(K, ipfs_address, encrypted_contract_address, K_p,K_r) {
```

```
contract_address = hybrid_decrypt(K_r,encrypted_contract_address);
is_verified = verify_signature(contract_address, K_p,signature);
if (is_verified) {
return {ipfs_address, contract_address};
} else {
throw new Error('Invalid signature');
}
}
```

Step 8: Data requester sends the access token to the smart contract to gain access to the data on the IPFS platform.

```
function request_access(smart_contract_address, access_token) {
transaction = smart_contract.request_access(access_token);
return transaction;
}
```

Step 9: Smart contract verifies the access token and grants access to the data on the IPFS platform.

```
function verify_access_token(smart_contract_address, access_token) {
access_granted = smart_contract.verify_access_token(access_token);
return access_granted;
}
```

Step 10: Data requester downloads the encrypted data from the secure storage platform and decrypts it using the symmetric key.

```
function download_and_decrypt(K, ipfs_address) {
  encrypted_data = storage_platform.download(ipfs_address);
  decrypted_data = hybrid_decrypt(K, C_1);
  return decrypted_data;
}
```

Step 11: Data requester performs operations on the decrypted data and then uploads any changes to the secure storage platform.

```
function encrypt_and_upload_changes(K, changes) {
  encrypted_changes = hybrid_encrypt(K, changes);
  storage_platform.upload(encrypted_changes);
  return true;
}
```

Step 12: Data requester sends the encrypted symmetric key to the device owner with their public key so the device owner can decrypt the changes.

```
function send_encrypted_key(K, K_p) {
  encrypted_key = hybrid_encrypt(K, K_p);
  send_secure_message(C_2 K_p);
}
```

Step 13: Device owner decrypts the symmetric key with their private key and then decrypts the changes made by the data requester.

```
function receive_encrypted_key(C_2,K_p,K_r) {
  symmetric_key = hybrid_decrypt(K_r, C_2);
  return symmetric_key;
}


function decrypt_changes(K,C_2) {
  changes = hybrid_decrypt(K, encrypted_changes);
  return changes;
}
```

Step 14: Device owner can revoke access to the data by removing the data from the secure storage platform and destroying the smart contract.

```
function revoke_access(ipfs_address, smart_contract_address) {
  storage_platform.remove(ipfs_address);
  blockchain.destroy_contract(smart_contract_address);
}
```

The secure data sharing algorithm has the ability to prevent and detect different types of cyber attacks, such as hacking, phishing, and social engineering, due to several security measures that are implemented within the algorithm.

Firstly, the algorithm can prevent hacking attacks by implementing strong encryption techniques that protect the data from unauthorized access. This makes it difficult for hackers to gain access to the data even if they manage to breach the system.

Secondly, the algorithm can detect phishing attacks by implementing access control mechanisms that verify the identity of the user before granting access to the data. This prevents unauthorized users from accessing the data and reduces the risk of phishing attacks.

Thirdly, the algorithm can detect and prevent social engineering attacks by implementing user awareness training programs that educate users on how to recognize and respond to social engineering attacks. This reduces the chances of users falling for social engineering attacks and providing access to the data.

In addition to these measures, the algorithm can also implement other security mechanisms such as firewalls, intrusion detection systems, and antivirus software that can detect and prevent different types of cyber attacks.

Overall, the secure data sharing algorithm has the ability to prevent and detect different types of cyber attacks due to the multiple layers of security measures that are implemented within the algorithm. By implementing these measures, the algorithm ensures that the data is protected from unauthorized access and that the users can securely share the data without any security risks.

4 Performance Evaluation

In order to evaluate the effectiveness of our proposed approach for securing data sharing and storage in 6G-based IoT networks using blockchain technology, hybrid encryption, and IPFS, we conducted a simulation study. The simulation study allowed us to assess the performance and security of our proposed approach under a variety of conditions and scenarios. In this section, we describe the simulation methodology and parameters used in our study.

4.1 Simulation Methodology

We conducted a discrete-event simulation using the NS-3 network simulator, which is a widely used open-source network simulation tool. Our simulation model was based on a realistic 6G-based IoT network architecture. We used a variety of parameters and scenarios to test the performance and security of our proposed approach.

Specifically, we simulated the sharing and storage of data in the network, as well as the use of blockchain technology, hybrid encryption, and IPFS for securing the data. We varied parameters such as the number of nodes in the network, the size of the data being transmitted, the network topology, and the type of attacks being launched against the network.

4.2 Network Topology

In this part, we studied the impact of different network topologies on the performance and security of our proposed approach.

Based on the simulation study (Fig. 2), we found that the ring topology is the best option for our approach, as it offers the lowest latency and highest throughput, as well as high security. The star topology is a reasonable alternative, with slightly higher latency and lower throughput, but still providing moderate security. The mesh

Table 2 Network topology simulation results

Network topology	Latency (ms)	Throughput (Gbps)	Security
Ring	5	10	High
Star	10	5	Medium
Mesh	20	2	Low

Table 3 Data size simulation results

Data size	Latency (ms)	Throughput (Gbps)	Security
Small (10 kB)	3	12	High
Medium (1 MB)	8	8	Medium
Large (100 MB)	15	3	Low

topology should be avoided, as it has the highest latency and lowest throughput, as well as low security.

It is worth noting that these results are specific to the simulation conditions and parameters used in our study, and may not apply to all scenarios. However, they provide a useful starting point for evaluating the impact of different network topologies on the performance and security of our proposed approach.

We also measured various performance metrics such as throughput, packet loss, and delay, as well as security metrics such as the number of successful attacks on the network. By analyzing the results of our simulations, we were able to assess the effectiveness of our proposed approach for securing data transmission and storage in 6G-based IoT networks.

4.3 Data Size

Table 3 summarizes the results about the impact of different data sizes on the performance and security of your proposed approach.

These results found that smaller data sizes offer the best overall performance and security for your proposed approach, while larger data sizes have a negative impact on performance and security. Specifically, the small data size had the lowest latency and highest throughput, while the large data size had the highest latency and lowest throughput. The medium data size had intermediate performance and security characteristics.

5 Conclusion

this paper proposed a novel solution for enhancing security in 6G-based IoT networks using blockchain technology, hybrid encryption, and IPFS. The proposed approach consists of four algorithms that enhance the security of the system: a user authentication algorithm, a data access algorithm, a data storage algorithm, and a secure data sharing algorithm. The paper's contributions include the use of hybrid encryption to ensure data confidentiality and IPFS to provide decentralized and fault-tolerant storage.

Through evaluation, the proposed approach was found effective in enhancing data security in 6G-based IoT networks by providing secure and tamper-proof data sharing. The results demonstrate the potential of blockchain technology, hybrid encryption, and IPFS to enhance security in 6G-based IoT networks.

Future work could focus on the scalability of the proposed approach to larger networks and the use of other technologies to further enhance security in 6G-based IoT networks. Additionally, the proposed approach could be extended to address other security challenges in these networks, such as protecting against denial of service attacks or ensuring privacy.

References

1. Srivastava A, Das DK (2022) A comprehensive review on the application of Internet of Thing (IoT) in smart agriculture. Wireless Pers Commun 122:1807–1837
2. Xu H, Klaine PV, Onireti O, Cao B, Imran M, Zhang L (2020) Blockchain-enabled resource management and sharing for 6G communications. Digital Commun Netw 6(3):261–269
3. Bodkhe U, Tanwar S (2021) Secure data dissemination techniques for IoT applications: research challenges and opportunities. Softw Pract Exper 51:2469–2491
4. Dahiya P, Kumar V (2023) IOT security: recent trends and challenges, emerging technologies in data mining and information security, pp 3–10
5. Deshmukh A, Sreenath N, Tyagi AK, Eswara Abhichandan UV (2022) Blockchain enabled cyber security: a comprehensive survey. In: 2022 international conference on computer communication and informatics (ICCCI), Coimbatore, India, pp 1–6
6. Rathod T, Jadav NK, Tanwar S, Sharma R, Tolba A, Raboaca MS, Marina V, Said W (2023) Blockchain-driven intelligent scheme for IoT-based public safety system beyond 5G networks. Sensors 23(2):969
7. Wang J, Ling X, Le Y, Huang Y, You X (2021) Blockchain-enabled wireless communications: a new paradigm towards 6G. Natl Sci Rev 8(9) (2021)
8. Li W, Su Z, Li R, Zhang K, Wang Y (2020) Blockchain-based data security for artificial intelligence applications in 6G networks. IEEE Netw 34(6):31–37
9. Dwivedi SK, Amin R, Vollala S (2022) Smart contract and IPFS-based trustworthy secure data storage and device authentication scheme in fog computing environment. In: Peer-to-peer network and applications
10. Deep S, Zheng X, Jolfaei A, Yu D, Ostovari P, Kashif Bashir A (2022) A survey of security and privacy issues in the Internet of Things from the layered context. Trans Emerging Tel Tech 33
11. Yeasmin A, Baig A (2020) Permissioned blockchain-based security for IIoT. In: 2020 IEEE international IOT, electronics and mechatronics conference (IEMTRONICS), pp 1–7

12. Moraes Rossetto AG, Sega C, Leithardt VRQ (2022) An architecture for managing data privacy in healthcare with blockchain. Sensors 22
13. Naz M, Al-zahrani FA, Khalid R, Javaid N, Qamar AM, Afzal MK, Shafiq M (2019) A secure data sharing platform using blockchain and interplanetary file system. Sustainability 11(24)
14. Ye Z, Leyou Z, Wu Q, Mu Y (2022) Blockchain-enabled efficient distributed attribute-based access control framework with privacy-preserving in IoV. J King Saud Univ—Comput Inf Sci 34(10):9216–9227
15. Feng C, Yu K, Bashir A, AI-Otaibi Y, Lu Y, Chen S, Zhang Di (2020) Efficient and secure data sharing for 5G flying drones: a blockchain-enabled approach. IEEE Netw 35
16. Shrestha AK, Vassileva J, Deters R (2020) A blockchain platform for user data sharing ensuring user control and incentives. Front Blockchain 3
17. Baalamurugan KM, Bacanin N et al (2023) Blockchain-enabled K-harmonic framework for industrial IoT-based systems. Sci Rep 13:1004