A DAPP Business Data Storage Model Based on Blockchain and IPFS

Xiangyan Tang[1,2], Hao Guo[1,2(✉)], Hui Li[3], Yuming Yuan[3], Jianghao Wang[1,2], and Jieren Cheng[1,2]

[1] School of Computer Science and Cyberspace Security, Hainan University, Haikou 570228, China
[2] Hainan Blockchain Technology Engineering Research Center, Haikou 570228, China
[3] Hainan Huochain Tech Company Limited, Haikou 570100, China

Abstract. Blockchain technology has been applied in various fields, providing strong support for the medical field, supply chain and other industries, followed by the emergence of DApp based on blockchain smart contract technology. DApp as a distributed application model, implements part of the background business functions by the smart contract of blockchain to ensure the transparency, openness and traceability of key businesses, and can effectively resist DDoS attacks. However, blockchain data storage is expensive and not suitable for storing large amounts of data. This is a key barrier to the large amount of user data generated by DApp, and the risk of data loss, tampering and so on if the data is stored in a centralized service organization. In addition, smart contracts cannot be modified once deployed, and DApp often need frequent iterative updates in the later maintenance process, which will be faced with the need to modify the business logic of smart contracts. To solve the above problems, this paper proposes a user's data storage model of DApp based on IPFS to reduce the storage cost of data, and designs a set of hot-swapping smart contract architecture based on contract address to manage DApp user data, so as to meet the functional business replacement in the later stage of smart contract. In addition, for the privacy security of data, this paper introduces the Diffie-Hellman key exchange technology as the encryption scheme to protect user data.

Keywords: Blockchain · Distributed file system · Diffie-Hellman · DApp

1 Introduction

With the rapid development of smart phones in recent years, the types and quantity of APPs in the current application market have increased sharply in recent years, providing people with more convenient and intelligent services, enriching People's Daily life and greatly improving people's quality of life. With the emergence and development of blockchain technology, a kind of APP which uses smart contract technology as the service background begins to appear, which is called distributed application program (DApp). Compared with the traditional App, DApp transfers the main functions undertaken by the centralized server to the blockchain and implements the relevant functions with smart

X. Sun et al. (Eds.): ICAIS 2021, LNCS 12737, pp. 219–230, 2021.
https://doi.org/10.1007/978-3-030-78612-0_18

contracts. The data generated by the user is stored on the blockchain [1–4]. At present, compared with the traditional APP application market, the number and scale of DApp is much smaller. According to the statistics of the website DappTotal in 2020, the total number of DApp on Ethereum is more than 3,000, and the number of DAPP-related smart contracts is more than 16,000 [5]. It can be seen that the number and type of DApp are less than traditional ones, and the development of DApp is still in the early stage compared with traditional ones.

As for the traditional centralized server as the background of App, it faces two prominent problems. (1) The background business logic of APP is realized by the Web background. All front-end requests are sent to the Web background, and the results are returned to the front-end after being processed by the Web. This centralized service mode is vulnerable to malicious attack, typically DDoS leads to service paralysis [6]. (2) Users' data is often stored in the database on the server, and such centralized database data may face the risk of being tampered with and lost [7].

For DAPPs based on blockchain technology, the on-chain data of the blockchain is characterized by traceability, immutability, transparency, etc. and the storage of the business data of the DAPP on the blockchain can effectively maintain data security [8, 9]. In addition, due to the distributed architecture of blockchain itself, such smart contract through blockchain replaces the traditional centralized back-end service, and the model of front-end access to smart contract to meet business needs can effectively resist the influence brought by DDoS [10].

However, DApp also face the following three problems: (1) the cost of on-chain data storage; (2) Security and privacy of on-chain data; (3) Iterative updating of functions of smart contract as a background service [2, 11, 12]. With Ethereum, for example, storing 1KB of data on the blockchain in a smart contract costs about $5, which is expensive to run for DApp that are likely to produce and store large amounts of user data. As a public chain, the data on the chain of Ethereum is open to the outside world, and anyone can view the data in the blockchain. If the data on the chain of DApp users are not protected, they will face the risk of privacy leakage [13, 14]. In addition, smart contracts in Ethereum can no longer modify their code content after they are deployed, only allowed to be destroyed. If the back-end functions need to be changed during the update iteration of the later DApp, the new smart contract can only be redeployed and the old smart contract destroyed, which is extremely tedious for the developers and maintenance personnel.

To solve the problems mentioned above, this paper designs and implements a DAPP user data storage model based on distributed storage system. The system uses IPFS, a distributed storage system, to store user data to reduce the data storage cost of blockchain, and a smart contract system is designed to manage each user's stored data. In order to solve the problem of iterative update of smart contract, this paper divides the background service function into several sub-functions, and realizes the sub-functions through the smart contract respectively. Smart contracts call each other through the contract address, so as to build the hot swapping architecture of the background service function to reduce the iterative update and maintenance cost of the background. To solve the privacy problem of user data, this paper uses Diffie-Hellman key exchange technology to generate corresponding keys for different data to encrypt the data.

2 Background Knowledge

2.1 Blockchain

Blockchain is a peer-to-peer distributed ledger. Each node in the system will back up a copy of all the transaction data. When a new transaction occurs, each node verifies the transaction through a consensus mechanism, and if it passes the verification, the validity of its transaction is recognized and recorded in the ledger. This process does not require the participation of a third party, so it has the characteristics of decentralization. Blockchain can be divided into public chain, alliance chain and private chain. Public chain is a kind of block chain open to everyone. Anyone can participate in transaction and verification and obtain all transaction data in the public chain. Public chain is the most widely used block chain at present. Alliance chain is a group of bookkeepers who are elected in some way within a group. These bookkeepers maintain the ledger of transactions, while other common nodes only participate in the generation of transactions and are not responsible for the verification of transactions. A private chain is a block chain that is not open to the outside world. The block chain is only owned by an organization or individual, and outsiders cannot access the data within the block chain.

The structure of the data in the blockchain is chain-like. Each data block is related to the hash value of the data in the previous block. Changing the data in any block will change the hash value of the data in that block. Each block stores the data hash value of the previous block, so if there is data tampering in the blockchain ledger data, illegal data can be easily verified according to the hash value of the data [15–17] (see Fig. 1).

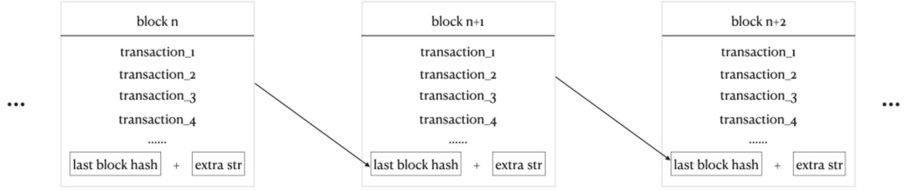

Fig. 1. Block structure

2.2 Distribute File System

Distributed storage system consists of peer-to-peer nodes, which is a content-addressable file system. Currently, such as IPFS, Swarm, etc. adopt this model. The distributed system is open to the outside world and any device can be added as a storage node. There is no need for trust between nodes, and nodes can store and download data with the help of clients. For each stored data in the distributed storage system will be cut into smaller blocks of data, from different nodes are stored, when a node to request a copy of data to the other nodes through P2P networks when they initiate the request, will send the data to request the rest of the node, it will effectively improve data throughput and save broadband [7, 12]. Since each piece of data is stored in various nodes in a distributed way, this can effectively avoid the network attacks faced by traditional centralized mode, such as DDoS (see Fig. 2).

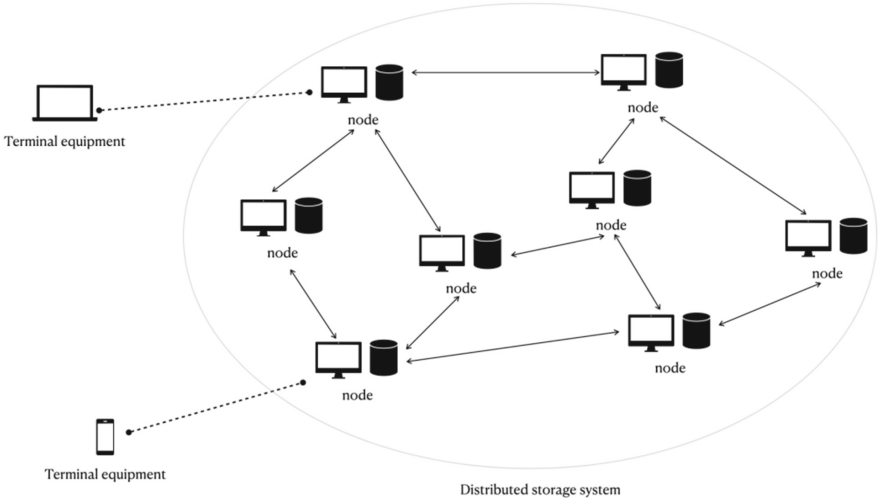

Fig. 2. Distribute file system

2.3 Diffie-Hellman Key Exchange

Diffie-Hellman key exchange is a key exchange algorithm, which was proposed by Whitfield Diffie and Martin Hellman in 1976. This algorithm can generate the same encryption key for two roles in a fully public environment. It is widely used in various data transmission protocols. According to the algorithm, two users only need to generate public content and private content respectively, and the same key can be generated by exchanging the public content of the other party [19]. However, Diffie-Hellman key exchange cannot prevent man-in-the-middle attack. Due to the data traceability and tamper-proof features of blockchain, this problem can be effectively avoided if the Diffie-Hellman key exchange process is implemented in the blockchain environment [20].

3 System Architecture Design

3.1 The Overall Architecture

The system model proposed in this paper includes the following four roles: (1) DApp front-end; (2) Blockchain smart contract system; (3) Distributed storage system; (4) Management Organization. Its detailed introduction is as follows:

- The front end of DApp is the program used by users, which provides corresponding services for users. The front-end processes user data by interacting with smart contracts and distributed storage systems.
- Blockchain smart contracts provide back-end services for DApp, which manage users' data hashes and generate data encryption keys.

- Distributed storage systems divide data into several small data blocks and store them on multiple devices, which will synchronize relevant data between devices. After users upload data to the system, they will get the encrypted hash value of the data, through which they can query data in the system.
- The management body is responsible for the late iteration and update of the DApp, which is responsible for the iteration of the front end of the DApp and the management and update of the smart contract.

3.2 Smart Contract System Design

By default, each DApp user has a separate set of public and private keys as an Ethereum account, and each user interacts with the smart contract through the Ethereum account. Smart contract is divided into user data storage contract, data management contract, key contract and directory contract. The following sections describe the content and functionality of each of these contracts.

DataStorage Contract. DataStorage contract are responsible for storing two types of data. One is to store relevant data of users, which maintains a user data table, and its structure is shown in the following table. Each user is uniquely identified by the account address. Data with a large amount of data is first uploaded to IPFS and then the hash value is stored in the smart contract. Data with a small amount of data can be directly stored in the smart contract. This contract only accepts the data in the DataManage contract to modify the data inside the contract. When the DApp requests to update the data of the user in the DataStorage contract, it will send a request to the DataManage contract first. The DataManage contract verifies the account permissions of the requestor, and after the permissions are approved, the DataManage contract modifies the hash value in the DataStore contract. The other is the system data that can be obtained by all users, which is publicly available without encryption. For the data with a large amount of data, it is stored in IPFS, and only the hash value is stored in the contract, while for the data with a small amount, it can be stored directly in the contract. This type of data can also only be manipulated by a DataManage contract, and only the specified account address (usually the service manager) has the right to modify the data (Tables 1 and 2).

Table 1. The structure of user data stored in the DataStorage contract.

User	Data1 hash	Data2 hash	…
User1	mbsPceqURmvYefhH jo4PfmGbRgEcbBVc6BonjSHF4Rru	QmeqYYxh6HbfwhSKwaB5DzZ77A yPV9VfKm1pNNvwuyrhyF	…
User2	QmbTqKdRi7H7CkvyghuB6eilztdlh 8HhnwytADf1gox9Xd	QmdgmHHitPahhd2k85bXZ5VtrwtCL8my VYZUgwXv1CTTNP	…
User3	QmUAT8T24ESpMvvkTkRYYep bPdu9tmGFcWtHhBiJLfXDyi	QmdmJW3XCcbTDQCZQTGB5Bu W7ZfXmN1L89mB39N3pJrxKe	…
…	…	…	…

Table 2. The structure of public data stored in the DataStorage contract.

Data1	Data2	Data3	…
QmVd8EbQYj9zA 8Ya67mzboPYR1 KPb9kwP6BiLVrgdDRZrE	QmXJv98swdb WHJ9JwmWCA28Tgqc 9JgGXvkBy2eDYPw35oP	QmejkAXfh32SZWn QJp9BRP2R7yaft EMXQKI2dgX8YhUNHD	…

DataStorage contract should have the following functions to manage the internal data.

- *initUserData()*: This method is used to initialize the data information of a user account, which usually occurs during the DApp user registration phase.
- *updateUserData()*. This method is used to modify or add data to the specified user.
- *addPublicData()*. This method is used to add new public system data.
- *updatePublicData()*. This method is used to modify or add information that specifies the name of the exposed system data.
- *deleteContract()*. This method is used to delete this contract, after which the contract cannot be called again.

DataManage Contract. The DataManage contract is used to manage the contents of the DataStorage contract. Its internal functions need to design corresponding functions according to the specific DApp type, and call the functions of the DataStorage contract through these functions. The DataManage contract should contain the following functions.

- *userAuthorityVerification()*. This method is used to determine whether the caller has the right to modify the data in the data storage contract. If the caller has the right (for example, the data that the caller applies to modify belongs to the user address itself), he or she can choose to conduct subsequent related operations; if not, he or she will refuse other subsequent requests.
- *manageUserData()*. This method is used to manage user data. For example, to modify or add account data, this method calls functions in the data management contract for user data.
- *managePublickData()*. This method is used to manage system exposed data, such as adding new data or modifying existing data. This method calls functions in the data management contract for system data.
- *deleteContract()*. This method is used to delete this contract, after which the contract cannot be called again. This function can only be called by the operator.

SecretKey Contract. From the perspective of data privacy security, this paper divides data into the following two types.

- *Public data*. This type of data is visible to everyone, and its data itself or hash values are stored directly in smart contracts, so encryption is not required.

- *Data visible only to the user and operator*. This type of data is generated by DApp users, and its information is visible to both the operators and the users themselves. In order to generate encryption key of data, Diffie-Hellman key exchange technology is adopted in this paper. The manager and the user respectively generate public content and secret content for the corresponding stored data. The secret content is not public as private data, while the public content can be obtained by anyone. Users and managers respectively drop their public content and store it in the SecretKey contract, through which both parties can obtain the other party's public content and calculate the same data encryption and decryption key by mathematical formula.

The SecretKey contract maintains a Diffie-Hellman disclosure table internally, as shown in the following table. The contract provides the corresponding public key for different types of data, and the front-end user obtains the public key by accessing the SecretKey contract to encrypt the data. The Diffie-Hellman Open Information Table maintains multiple data tables for different types of data. Each data table is provided by the operator with one public information. DApp users who want to store such data must also provide one public information and store it in the data table. Operator and user can respectively generate the same key to encrypt and decrypt data through the public information provided by the other party (Table 3).

Table 3. A Diffie-Hellman public information table for a class of data.

Manager public content	User1 public content	User2 public content	…
QmNU32urwsz VZULo4tBoczwiqjk 41TyBxWc1hwVcAUuaBt	QmVjdMtgPE4bpalck PrmJs4xi1aob DkJoS59Emg8J7dHQQ	Qmbzr59EzXAos xBqTgo4o3Vshsrgj VQE97cwTQitUi1kHr	…

The SecretKey contract contains the following functions internally.

- *changePublicContent ()*. This function modifies the public key of a piece of data in a key contract. This function can only be called by an operator.
- *createPublicContent ()*. This function creates a new Diffie-Hellman disclosure table for a new piece of data in the key contract, along with the operator's key disclosure information. This function can only be called by the operator.
- *addUserPublicContent ()*. This function is called by the DApp user to add the user's public key information to a certain piece of data in the key contract.
- *deleteContract ()*. This method is used to delete this contract, after which the contract cannot be called again. This function can only be called by the operator.

Directory Contract. The directory contract is used as a smart contract system subdirectory, which contains the address index of other sub-function modules and the ABI of other contracts stored in IPFS. DAPP can access the directory contract to search for other sub-function contracts. The catalog contract maintains a table of data about the subfunction contract internally, as shown below.

The directory contract implements the following functions.

- *addChildContract ().* This function adds a new subcontract information inside the directory contract, including the contract name, the contract address, and the contract ABI.
- *changeChildContract ().* This function modifies the information of one of the subcontracts in the catalog contract, such as the contract address and the contract ABI.
- *deleteChildContract ().* This function invalidates a contract information in a directory contract.
- *deleteContract ().* This function is used to delete this contract.

Note that the call authority of the directory contract should be strictly controlled. Generally, it can only be operated by the account address of the manager, and ordinary DApp users have no right to call (Table 4).

Table 4. Subfunction contract data table.

Contract name	Contract address	Contract ABI hash
Data storage contract	0x26AAA1C996354a56Gi 755726cAC15FEaA618FD21	QmYyKJUJiFiZLSgAKXAg7F63 PWeoaxEfVimmTo88HfRe5B
Secret key contract	0xCA0D34C770835go8340 CC7731b484c0Ae1e9A2d6	QmPjuKrBWQHDQJSygjmXUNKa E3GV1QNshzD8u23wZGxFuq
Data manage contract	0x52B49a8DFoF9A048ka35 BC5c8Db3e7888B60E400	QmcbvD7y5uyF8HRdzpJ459s2 AqNspvr6Meue8xtaHnSsT5
…	…	…

3.3 The Front End Designs

The front end of the DApp uses Android system as the platform to develop the corresponding App to interact with the Ethereum smart contract. For convenience of testing, the front-end program is responsible for two parts: (1) uploading data to IPFS and calling the corresponding smart contract. (2) Call the smart contract to obtain the corresponding data and hash value, and access the IPFS system to download the data to the local area according to the hash value.

Upload Data. The front-end user data is uploaded using HTTPS protocol to interact with the smart contract with the help of Infura. The front-end device needs to access the directory contract to get the address of the data management contract to call the smart contract. For the data to be encrypted, the SecretKey contract is called to get the public key of the manager of the corresponding data to generate the key, and its hash value will be used as the key of AES encryption. User data is encrypted by the AES of the key and then uploaded to IPFS. IPFS will return the hash value of the data. Similarly, the

hash value also needs to be encrypted by the AES of the key to ensure privacy. Finally, DataManage Contract is called to store the encrypted hash value in the blockchain. For the public data without encryption, you can directly store the data hash returned by uploading to the IPFS system in DataStorage Contract (see Fig. 3).

Download Data. Data downloading is similar to uploading. The front-end device first accesses the directory contract to get the address of the data management contract and then accesses the function of the data management contract to get the hash of the data. For the unencrypted hash value, the user data of the hash value can be obtained directly by accessing the IPFS system. For the encrypted hash value, the front-end device needs to access the SecretKey Contract to obtain the Manager Publick Key of the data, and then generate the corresponding Key according to the Private Keys stored by the front-end device itself to decrypt the encrypted hash value. Finally, the decrypted hash value is submitted to the IPFS system to obtain the data, and the obtained data is decrypted.

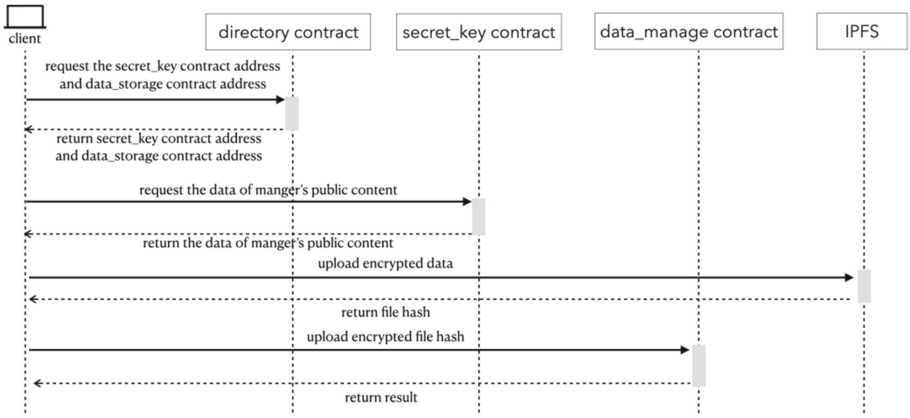

Fig. 3. Upload encrypted data

4 Experimental Evaluation

In this paper, related experiments are carried out to evaluate the system. Smart contracts are deployed in the Ethereum test chain. The DApp manager uses the Python-based Web3 library to interact with the smart contract, while the DApp client uses the Java-based Web3 library to interact with the smart contract. An Intel Core i5 quad-core 2.3 GHz desktop is used as the IPFS client.

The deployment cost of each sub-function smart contract is as follows (Fig. 4):

Fig. 4. The gas used of contract deploy

Gas consumption of function call related to operation data in the contract is as follows (Fig. 5):

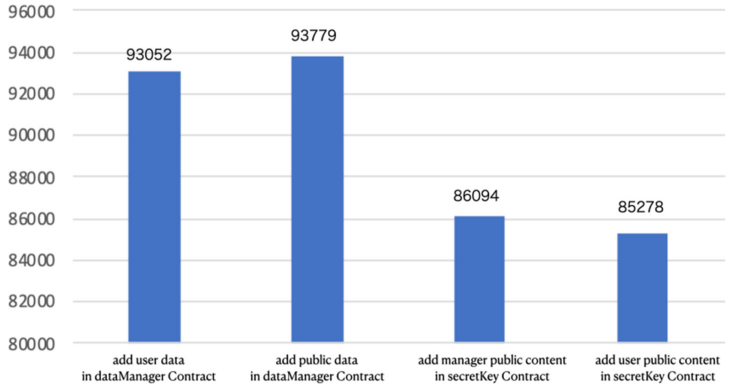

Fig. 5. The gas used of contract function

IPFS client local data upload efficiency is as follows (Fig. 6):

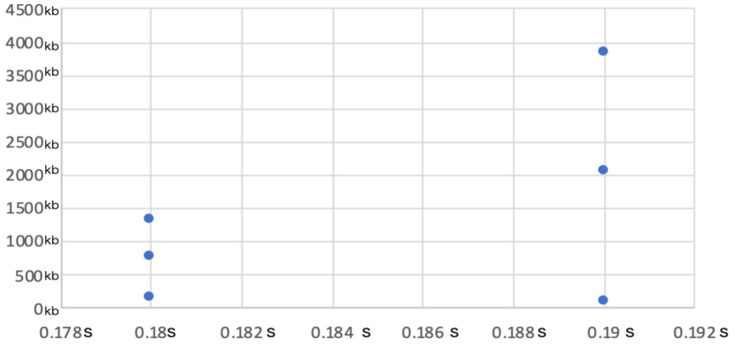

Fig. 6. The time coast of upload file in local IPFS node

5 Conclusion

This paper designs a business data storage model for DApp, which stores business data with distributed storage system, maintains the hash value of business data with blockchain smart contract, and implements the data encryption key with Diffie-Hellman key exchange technology. This model can guarantee the high throughput of data storage, and has the characteristics of traceability and tamper-proof. And this kind of distributed architecture can resist DDoS well.

Acknowledgement. This work was supported by the Hainan Provincial Natural Science Foundation of China (Grant No. 2019RC041 and 2019RC098), Research and Application Project of Key Technologies for Blockchain Cross-chain Collaborative Monitoring and Traceability for Large-scale Distributed Denial of Service Attacks, National Natural Science Foundation of China (Grant No. 61762033), Opening Project of Shanghai Trusted Industrial Control Platform (Grant No. TICPSH202003005-ZC), and Education and Teaching Reform Research Project of Hainan University (Grant No. hdjy1970).

References

1. Tsung-Ting, K., Hyeon-Eui, K., Lucila, O.M.: Blockchain distributed ledger technologies for biomedical and health care applications. J. Am. Med. Inform. Assoc. **24**(6), 1211–1220
2. Dwivedi, A., Srivastava, G., Dhar, S., et al.: A decentralized privacy-preserving healthcare blockchain for IoT. Sensors **19**(2), 326 (2019)
3. Li, M., Weng, J., Yang, A., et al.: CrowdBC: a blockchain-based decentralized framework for crowdsourcing. IEEE Trans. Parall. Distrib. Syst. **30**(6), 1251–1266 (2018)
4. Decentralized Applications – dApps. https://blockchainhub.net/decentralized-applications-dapps/. Accessed 20 Sept 2020
5. dapptotal Homepage. https://dapptotal.com/. Accessed 12 Oct 2020
6. Peng, T., Leckie, C., Ramamohanarao, K.: Survey of network-based defense mechanisms countering the DoS and DDoS problems. ACM Comput. Surv. **39**(1), 31–342 (2007)
7. IPFS-Content Addressed, Versioned, P2P File System. https://arxiv.org/pdf/1407.3561.pdf. Accessed 21 Nov 2020
8. Wang, S., Ouyang, L., Yuan, Y., et al.: Blockchain-enabled smart contracts: architecture, applications, and future trends. IEEE Trans. Syst. Man Cybernet. Syst. **49**(11), 2266–2277 (2019)
9. Treiblmaier H.: The impact of the blockchain on the supply chain: a theory-based research framework and a call for action. Supply Chain Manage. (2018)
10. Dai, H.N., Zheng, Z., Zhang, Y.: Blockchain for Internet of Things: a survey. IEEE Internet Things J. **6**(5), 8076–8094 (2019)
11. Nizamuddin, N., Salah, K., Ajmal Azad, M., et al.: Decentralized document version control using Ethereum blockchain and IPFS. Comput. Electr. Eng. **76**, 183–197 (2019)
12. Patsakis, C., Casino, F.: Hydras and IPFS: a decentralised playground for malware. Int. J. Inf. Secur. **18**(6), 787–799 (2019)
13. Zhumabekuly Aitzhan, N., Svetinovic, D.: Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. IEEE Trans. Dependable Secure Comput. **5**, 840–852 (2016)
14. Shi, P., Wang, H., Yang S., et al.: Blockchain-based trusted data sharing among trusted stakeholders in IoT. Software Pract. Exper. (15) (2019)

15. Li, Y., Yang, W., He, P., et al.: Design and management of a distributed hybrid energy system through smart contract and blockchain. Appl. Energy **248**, 390–405 (2019)
16. Treiblmaier H. The impact of the blockchain on the supply chain: a theory-based research framework and a call for action. SSRN Electron. J. (2018)
17. Blockchain-for-beginners-what-is-blockchain-just-7-step. https://ethfans.org/posts/blockchain-for-beginners-what-is-blockchain-just-7-step. Accessed 12 Sept 2020
18. Ali, M.S., Dolui, K., Antonelli, F.: IoT data privacy via blockchains and IPFS. In: International Conference on the Internet of Things, pp. 1–7. ACM (2017)
19. Joux, A.: A one round protocol for tripartite Diffie–Hellman. J. Cryptol. **17**(4), 263–276 (2004)
20. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_9