


Article

A Secure Data Sharing Platform Using Blockchain and Interplanetary File System

Muqaddas Naz ¹, Fahad A. Al-zahrani ², Rabiya Khalid ¹, Nadeem Javaid ^{1,*},
Ali Mustafa Qamar ^{3,4}, Muhammad Khalil Afzal ⁵ and Muhammad Shafiq ^{6,*} 

¹ Department of Computer Science, COMSATS University Islamabad, Islamabad 44000, Pakistan; muqaddasawan532@gmail.com (M.N.); rabiyaakhalid672@gmail.com (R.K.)

² Computer Engineering Department, Umm AlQura University, Mecca 24381, Saudi Arabia; fayzahrani@uqu.edu.sa

³ Department of Computer Science, College of Computer, Qassim University, Buraydah 51452, Saudi Arabia; al.khan@qu.edu.sa

⁴ School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad 44000, Pakistan

⁵ Department of Computer Science, COMSATS University Islamabad, Wah Cantonment 47040, Pakistan; khalilafzal@ciitwah.edu.pk

⁶ Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Korea

* Correspondence: nadeemjavaiddqau@gmail.com (N.J.); shafiq.pu@gmail.com (M.S.)

Received: 16 November 2019; Accepted: 3 December 2019; Published: 10 December 2019



Abstract: In a research community, data sharing is an essential step to gain maximum knowledge from the prior work. Existing data sharing platforms depend on trusted third party (TTP). Due to the involvement of TTP, such systems lack trust, transparency, security, and immutability. To overcome these issues, this paper proposed a blockchain-based secure data sharing platform by leveraging the benefits of interplanetary file system (IPFS). A meta data is uploaded to IPFS server by owner and then divided into n secret shares. The proposed scheme achieves security and access control by executing the access roles written in smart contract by owner. Users are first authenticated through RSA signatures and then submit the requested amount as a price of digital content. After the successful delivery of data, the user is encouraged to register the reviews about data. These reviews are validated through Watson analyzer to filter out the fake reviews. The customers registering valid reviews are given incentives. In this way, maximum reviews are submitted against every file. In this scenario, decentralized storage, Ethereum blockchain, encryption, and incentive mechanism are combined. To implement the proposed scenario, smart contracts are written in solidity and deployed on local Ethereum test network. The proposed scheme achieves transparency, security, access control, authenticity of owner, and quality of data. In simulation results, an analysis is performed on gas consumption and actual cost required in terms of USD, so that a good price estimate can be done while deploying the implemented scenario in real set-up. Moreover, computational time for different encryption schemes are plotted to represent the performance of implemented scheme, which is shamir secret sharing (SSS). Results show that SSS shows the least computational time as compared to advanced encryption standard (AES) 128 and 256.

Keywords: blockchain; IPFS; AES; RSA; TTP; SSS; Ethereum; smart contracts

1. Introduction

In the past decade, technological advancements have been made by research consortiums to adapt data sharing approaches. In such a way, research-based activities can improve through collaboration

and intelligent decisions. Data sharing is the fundamental step to gain maximum benefit from research innovations. However, it is very crucial to know the three W's for sharing purpose, such as what, when, and where. These questions need to be very much clear before initiating the data sharing process. There is still some scope to work on, how the data set owner should be given incentives or reward. This research provides secure sharing and selling of data by leveraging the benefits of blockchain [1]. Blockchain; a distributed ledger is a new trend in the world of information technology. A lot of financial and non financial applications have made use of blockchain. The consensus mechanism is considered as the most fundamental and significant invention as declared by one of the specialists of silicon valley. The current paper currency relies on third party, which means that there is a threat to security, trust, and privacy. The significant feature of blockchain is trust, which can be achieved by eliminating third party. A peer to peer currency; bit coin is introduced by Nakamoto in 2008 [2]. This allows the payment to be send directly from one party to another. A paper by Nakamoto was later came up with the implementation protocol of genesis block with 50 coins. Currently, blockchain is used in every domain, like cloud, internet of things (IoT), data trading, information security, health care, and many more because of its striking features [3].

The major underlying problem in research data sharing is the fear of researchers regarding misuse and misinterpretation of data. This is because data sharing approaches are still immature in the context of trust, which is slowly going to be established among research community. To tackle this issue, various solutions are proposed, for example, protection of identities of every individual and controlled access to the data rather than making all the data open access. Still these solutions cannot provide trust, immutability to digital data, and traceability regarding data usage.

Cloud servers store the excessive amount of data, which is a centralized authority. There are various type of risks associated with a central authority, such as single point failure. To avoid such failure, third parties are involved to provide data backups. To eliminate third party for developing a trust based model, a blockchain is introduced to provide trust and transparency. Decentralized storage is a solution, which allows storage of data independently on multiple nodes of the network in the form of a distributed ledger. The problem is the storage and processing limitation of network nodes. For this purpose, interplanetary file system (IPFS) is adapted, which is a peer to peer architecture [4]. There is no risk of single point failure. It is similar to web3, but with different features. It performs content addressing and works in a similar way as bit torrent. Availability of data is ensured by storing it on a decentralized platform; IPFS.

In existing platforms, data owners do not have a complete control over the access and use of data. In most of the cases, the owner himself is not involved in the sharing of data. For example, owner is the passive entity, while escrow is solely responsible for data distribution and payment settlements. At the end, owner gets some percentage of royalty. Without blockchain, it is very difficult to ensure the transparency of funds. It means that a fair share of money cannot be guaranteed. In this scenario, blockchain can provide trust and transparency among the nodes of network for the fair distribution of received payment from requestor of data.

On the other hand, quality and integrity of data cannot be compromised when a customer is paying ethers to get the content. To tackle these issues, a single solution is proposed in response to multiple problems. In the proposed scenario, owner himself is dealing with the sharing of digital data by eliminating any third party. Data owner can set access rules, so that data is not released to unauthorized party. Whenever, a requestor demands the data from owner, the owner sends a request to IPFS server to provide the requested content. When a user registers for data request, it is authenticated through digital signatures. Only after that, the request proceeds, otherwise, transaction would be terminated by blockchain.

Data security is achieved by integrating encryption scheme to the hashes of uploaded data on IPFS. Later on, these hashes are encrypted by owner using shamir secret sharing (SSS) [5] scheme, which divides the hash into n number of encrypted shares. The encrypted shares are stored in smart contract. The customer sends request to access the data, which gets authenticated through digital

signatures. After downloading the data using decrypted shares, the customer is supposed to register reviews about data on review system. To encourage customers for review registration, an incentive is declared, which is the refund of 10% of actual money deposited by the customer.

The rest of the paper is organized as follows: Section 2 presents the state-of-the-art, while motivation and problem statement are given in Sections 3 and 4. System model for proposed scheme and smart contract design are given in Section 5. Section 6 presents implementation details and simulation results. Finally, the paper is concluded in Section 7.

2. Related Work

In recent era, blockchain such as; bitcoin [2], Ethereum [6] and Zcash [7] are considered as hot and fundamental technologies of cryptocurrency. As a result, researchers and industrialists are paying more attention to establish a trust based model in a decentralized manner. In this section, few studies regarding blockchain are presented.

2.1. Digital Content Protection

In order to preserve the privacy for traceable encryption in blockchain, Wu et al. in [8] proposed a system in which authenticity and non-repudiation of digital content is guaranteed. The problem tackled by authors is the secret key of user, which when shared with other entities does not hold the specific information of user. In case the shared key is corrupted or abused, it makes difficult to analyze the source of secret key. Moreover, leakage of confidential information in access control is a bottleneck for existing systems. Therefore, authors have integrated the privacy protection algorithm such as attribute based encryption (ABE) to secure the secret keys. However, decryption mechanism does not show improved efficiency.

Management of digital data rights is a fundamental requirement to achieve protection of digital data. Existing techniques for data rights lack transparency, decentralization, and trust. In response to above mentioned problems, Zhang and Zhao in [9] proposed blockchain-based decentralized solution. Information regarding the use of digital content, such as transaction and license information is transparent to everyone. Smart contract is designed for the automatic assignment of license. In this mechanism, owner can set the prices for selling the license to other customers. However, peers of the network have to possess high computational power to perform key acquisition.

Ma et al. in [10] focus on digital rights management using blockchain to avoid the use of sensitive digital content for illegal purposes. For such concerns, a solution is proposed which is called as DRMchain. This solution ensures the usage of digital content in the right way by authenticated users. Two separate blockchains are designed: one is to store the original content with its cipher summary, and the other stores the cipher summary of protected digital content. DRMchain provides the traceability record of a violation and high level trusted protection. From the proposed solution, protection of digital content, secure authorization of user, and use of multi signatures for usage control is achieved. However, the use of Ethereum coin could be a new research direction for protection of digital content.

Data sharing is a crucial step to gain maximum benefit from the strengths of research. A lot of data sharing mechanisms are proposed and discussed in literature. There is no sufficient work available that focuses on the incentive mechanism to promote data sharing. To cover these limitations, authors in [11] conducted a review on medical and health data to uncover the incentive mechanisms with the pre- and post- results after empirical analysis. According to survey, a single incentive is tested for medical and health data to analyse the rate of data sharing. Therefore, it is concluded that more incentive based researches need to be performed to encourage data sharing.

2.2. Blockchain in Cloud

Efforts are being made by researchers to deploy blockchain on a broader domain due to its striking features. A blockchain provides secure, reliable and trust worthy data sharing environment. It records

any unauthorized access to data, hence provides traceability. However, its distributed nature weakens the controlling capability of networks. Moreover, immutable nature of blockchain can induce the threat of a majority attack on network. To solve these issues, authors in [12] proposed data management system which is controllable in blockchain, named as CBDM, specifically designed for deployment in cloud infrastructures. Trust is achieved by introducing trusted authority in a system which has a higher level of control over the network. In future, a similar prototype can be designed for real time setup.

In current digital era, size of data is increasing every day with a significant amount. This brings a storage issue at users' terminals. The solution to this could be leveraging these storage limited nodes with cloud services. However, this solution comes with a cost of compromised trust and security, as it is difficult to rely on third party. In consideration to afore mentioned problems, a secure distributed storage architecture is proposed by authors in [13], in which data is divided into multiple encrypted chunks, and randomly stored on peer to peer nodes of network, which voluntarily offer storage services. A genetic algorithm is applied for placement of file blocks among multiple users. However, decentralized storage platform, such as filecoin, storj, and swarm are not considered.

Cloud servers provide services like sharing of resources. Integrating blockchain-based cryptocurrency decentralization requires high computation power. In order to tackle this issue, authors in [14] work on the efficient allocation of resources in cloud servers, such that privacy can also be ensured by making use of blockchain. Multi-level access is granted to users through multi-secret mechanism. This scheme requires no third party which means trust can be achieved. A message is divided into multiple secret and divided independently among multiple users depending upon the level of access. Based upon this mechanism, a decentralized e-voting is designed, where a role is assigned to each sublevel access structure.

The Existing schemes for management of data rights lack fine level of access control. To introduce such fine control in cloud computing, the authors in [15] propose a novel encryption method by making use of attribute based encryption (ABE) with secure key management. The Users which satisfy the attributes in access control policies can get the encrypted content, and can decrypt it with secret key for its personal use. Experimental results show that proposed solution is effective for security, authentication and is reliable than existing schemes.

Improving search results while storing encrypted data over cloud is an important issue to consider. This ensures the privacy of sensitive information when dealing with un-trusted third party in the form of cloud storage. This issue is tackled by authors in [16] by introducing attribute based encryption with user revocation (ABE-UR). Index based search strategies are employed for data outsourcing. Fine grained access control is achieved by ABE and proxy re-encryption. To verify the accuracy of search results, a verification scheme is designed. Furthermore, performance evaluation shows the robustness of designed system. However, computational complexity can further be improved.

Clouds are centralized servers which provide services for data storage and sharing between different stakeholder. In this work [17], sharing of medical data of patients between hospital management is secured using blockchain. End-to-end delivery of medical data needs to be encrypted to handle any misuse by third party. A blockchain-based data sharing platform is designed for multiple services providers on cloud. Smart contracts are written to automate secure sharing and distribution by employing access control strategies by data owners. This scheme also provides transparency and traceability. Performance of proposed scheme is analyzed by making a fair comparison with the existing schemes. However, there is still a scope to improve security and fine grained access control through ABE.

2.3. Blockchain in IoT

IoT plays a vital role in automating our daily lives. Electronic devices connect to each other through internet and tend to share and exchange data. Considering the privacy and security concerns, a mechanism must be adapted, so that data integrity and authentication of digital devices can be

ensured. Authors in [18] proposed a blockchain-based decentralized scenario, named as bubble of trust. In the proposed scheme, devices are first authenticated before initiating a data sharing process, so that any malicious activity can be avoided within the network. A robust identity management and authentication ensures the trust among devices. However, the proposed approach suffers from certain limitations, such as it is not adaptable to real time setup, requires an initialization phase, and evolution of rate for crypto-currencies is not discussed.

Existing data sharing schemes for IoT using blockchain has some loopholes, such as security risks, high maintenance cost, and supervision of big data coming from IoT network. To tackle the afore mentioned challenges, authors in [19] proposed secure fabric based data transmission mechanism. Data is transferred to different trading centers by incorporating secret based mechanism known as Shamir's secret sharing (SSS) algorithm. Storage of data is designed as the dynamic linked based storage along with data consensus protocol. However, this mechanism is only proposed for small scale applications and authenticity of power data security is neglected.

2.4. Access Control Using Blockchain

Storage is a critical issue in blockchain especially when large amount of data needs to be stored on network nodes. Storage capacity of terminal nodes is limited, as it does not allow to store very large size data. This problem gives rise to multiple issues, such as computational power and high cost of processing large amount of data. Considering the aforementioned challenges, Stiechen et al. in [20] proposed a decentralized storage mechanism by using IPFS and access control policies. Files are stored in the form of chunks on each node. However, a file cannot be accessed until unless proper permissions are assigned to users. This is a good approach for preserving the privacy of confidential data. The proposed schemes suffer delay while accessing files from server due to blockchain interaction. Access control and securing sensitive data is the fundamental need of recent era. Blockchain is an emerging technology which provides solution to multiple problems, i.e., security, privacy, access control, trust, traceability and many more. In this perspective, research work [21] presents a blockchain-based solution for securing the patient's data in medical health records. Access control rules are defined to restrict the access of data, so that privacy of patients can be preserved. A secure interoperability among heterogenous blockchains is achieved by exploiting the use of smart contracts. A set of smart contracts are designed to achieve multiple purposes. However, certain measures need to be adapted to increase adaptability and robustness.

Access control reflects the system's security by restricting the access of data to a specific set of people. It is important while dealing with the sensitive data such as patient's health condition and medical record. Existing systems lack the access control policies to ensure data integrity and privacy. For this purpose, the authors of [22] proposed a consortium blockchain with distributed ledger. Each operation performed by smart contract is stored as a transaction record in blockchain. A proof of concept is designed to validate the performance of system as compared to existing approaches. The proposed solution ultimately achieves integrity, scalability, and authenticity. However, privacy of patient's personal information is not considered.

Online data that is stored on a third party centralized server is vulnerable to attacks. For this purpose, Masayuki et al. in [23], proposed secure online storage without involving central server. Data of users can not be attacked by malicious party. Each user's information is divided into parts using secret sharing algorithm. These chunks are stored on distributed nodes of a network. The actual data is converted to metadata by hiding its main features. A user can locate the nodes of network by restoring the actual data. The proposed scheme is reliable because of the fact that with changing of peer nodes, a user can still detect the target data. Moreover, a malicious node can be detected by other nodes of the network using majority rule consensus. However, security evaluation of system is not done in a quantitative manner.

Medical health records of patients are completely controlled by hospital management. Secure sharing of medical record is a need so that patients can consult their health records from multiple

hospitals. The proposed system by Guo et al. [24] provides secure, immutable, and accurate data to patients. In this work, blockchain-based secure attribute encryption scheme is employed in which patients attest their medical data to declare no disclosure of information to un-trusted parties. The Proposed system is compared to existing such schemes to validate the performance. However, a single tone predicate is required for its enriched representation.

3. Motivation

A blockchain-based data sharing framework is proposed by authors in [25]. A framework contributes to the version control of the documents so that the changes can be tracked easily. Multiple users can collaborate on a blockchain platform to negotiate on document changes. Therefore, multiple versions of documents can be stored in a network without the involvement of third party. Moreover, a decentralized storage, IPFS, is used to store the data. Ethereum smart contracts are used to develop an interaction among multiple entities such as owners and users of the document.

Data privacy and confidentiality is ensured by authors in [26]. ABE along with key policy are the important techniques to secure digital content. In this paper, inner product encryption and re-encryption techniques are applied where decryption is only possible if private key has an inner product with a value of zero. A specific set of attributes are specified by the owner of data to grant access. Hence, it achieves fine-grained access control.

Authors in [27] proposed a decentralized storage system with digital content protection. A secret key is distributed to the user by the owner of the data. While releasing the secret key, the owner defines the access policy to have a control over the usage of sensitive information. Attribute encryption scheme is adapted by authors to accomplish their goal.

From the existing literature discussed above, we have gained motivation to work on the digital data sharing using blockchain. Most of the researchers have worked on similar domain, yet there is much scope to enhance and modify the existing work to contribute for the benefit of research community.

4. Problem Statement

Blockchain is one of the leading technologies in the current decade. It brings a solution for challenges like traceability, transparency, trust, and accountability. The authors of [28] proposed a mechanism to ensure digital data rights management. Blockchain is used to store the permanent record of agreements between data owner and user. A user requests the data from the owner by making an agreement according the terms and conditions stored in the blockchain. The proposed solution achieves transparency, traceability and fine grained granularity over research data by using blockchain. Each entity (owner and user) has been assigned a unique Ethereum address to perform transactions. However, decentralized platform to store author's file is not taken under consideration. This leads to a storage problem [29] because network nodes have memory limitation and blockchain can only store the record of transaction and logs, not a complete file. Moreover, protection of digital content hashes using encryption are not considered. A trusted relation between buyer and seller is not build in this paper; therefore, authenticity of owner and data is not guaranteed for a user to initiate a trading process.

A frame work for fair share of author's royalty is proposed by authors in [30]. Online publishing and distribution of digital content is ensured by using blockchain to add transparency and trust in the model. Owner of the data shares its content with the publisher's server. The server distributes the content to the users according to received request. When the user successfully receives the file, the participating entities get their fair share according to defined criteria. A publisher's server is a centralized server, which stores digital content of the owner. Instead of centralized server, the authors of this paper did not consider the decentralized storage. This issue is tackled by authors of [31]. Moreover, access control for the academics research data, protection of digital content through encryption scheme and owner's reputation system are not considered for this scenario.

The authors of [31] proposed a proof of delivery protocol using smart contracts and blockchain. Digital content provider and file server are responsible entities to deliver the content to customer. After successful download of content by the customer, the participating entities (owner and file server) are given incentives as a reward. To solve downloading disputes between customer and file server, arbitrator acts as an unbiased agent to resolve the conflicts. The limitation of this research work is that encryption of digital content to avoid leakage of data to be sold by owner, access rights over data and a review system for owner's authenticity and quality of data are not considered.

From the literature review, it is concluded that most of the researchers have worked on digital data trading with and without blockchain. Yet there are certain limitations that need to be catered for further enhancements in existing systems. In this paper, decentralized storage, IPFS, Ethereum, and encryption mechanism are used for secure and trusted consolidated platform for sharing of digital data between buyer and seller. The problem of storage is catered by introducing IPFS [32], which stores data and return hashes to the owner or seller. To achieve data security, owner encrypts the hashes to avoid risk of data leakage which is to be sold. So that data on IPFS can only be accessed by verified customers, who belong to research community and submit the required deposit for digital content. In this way, owner can run his business with reasonable profit. Moreover, owner's reputation based system is designed to ensure quality of data, motivated by [33]. A user or buyer can first check the reviews of seller and its data before submitting the deposit for content. However, in paper [33], buyer incentives are not declared to motivate the customer for registering their feedback. Through rating or reviews, a future customer can easily judge the quality and authenticity of data.

Online review system is a guarantee for customers regarding the quality of any particular product. The review or reputation system is also applicable to digital content. For example, a movie or season downloading web sites ask from the users for their reviews or experience after downloading the content. At the end, the user submits the reviews in any format, either stars or textual form. These reviews help the upcoming users to check the quality of content. However, the underlying problem is an online review system that needs to be considered is the registration of fake reviews. The authors of [33] presented an idea of reputation system, but they did not consider the registration of fake reviews. In most of the cases, users do not realize the importance of reviews and therefore, they do not provide their honest opinion regarding the product. In other case, most of the customers are being paid by owner to register good reviews, no matter, what is the quality of content.

Buying and selling of digital content using blockchain can guarantee security, trust and decentralization. When a customer buys any content from owner, a single time payment is made in form of virtual currency. For example, in case of any digital book related to academia, a customer pays the amount of digital book and can download it for its use from IPFS. Now, the problem here is, what if the content of book are later modified by owner. More likely, the owner release their books with minor changes after certain amount of time, as different editions of the book. Now, the owner already paid the amount. So a document version control system needs to be adapted for old customers, so that they get notified whenever a new version of that book comes in market. Document version control is not tackled by authors in work [28].

In online trading of data, disputes may occur between seller and buyer. There needs to be some unbiased entity that can resolve the downloading disputes, for example, if customer claims that the data downloaded from the server is incomplete. Then the customer cannot be trusted, because he may utter the false statement just for the sake of refund. In order to confirm or verify the claims of customer, an unbiased entity needs to be introduced into the system which authors did not tackle in [28]. The main contributions of this paper are given below.

- *Identity management* Users are first authenticated using RSA signatures before giving them access to data.
- *Security of digital assets:* Adding encryption to the data hashes ensures the security and avoids data leakage.
- *Authenticity of owner:* Seller can trust the owner by checking the reviews about owner and its data.

- *Quality of data*: Rating or reviews depict the quality of data; therefore, by checking the reviews, buyer will only get high quality data.
- *Management of fake reviews*: Fake reviews are being identified using Watson analyzer.
- *Dispute handling*: Arbitrator is introduced to resolve downloading disputes.
- *User incentive*: By giving incentives, sellers are being motivated to give feedback on used data to aid future customers. The comparison of existing work with proposed scenario is given in Table 1.

Table 1. Comparison with existing work.

References	Blockchain	Data Encryption	IPFS	Reputation System	Access Control	Incentive	Review Validation	Dispute Handling
Base paper 1 [28]	✓	✗	✗	✗	✓	✗	✗	✗
Base paper 2 [30]	✓	✗	✓	✗	✗	✓	✗	✓
Base paper 3 [31]	✓	✗	✓	✗	✗	✓	✗	✓
Base paper 4 [33]	✓	✗	✗	✓	✗	✗	✗	✗
Proposed scenario	✓	✓	✓	✓	✓	✓	✓	✓

5. System Model

The proposed system model is motivated from [30,32,33]. The system model has the following entities.

Owner: It may be a person or organization which owns the data to be shared among customers. It can also control the query and access of data by filtering out the requestors.

Customer: Makes a Request for buying the data from the system. After receiving the desired content, the customer downloads the file from IPFS server by reconstructing the hash and registers his reviews about data on the review system.

Workers: It is the passive entity of system, provides decryption services on behalf of the customers. Authenticates the new customers through signatures and query the smart contract for requested data by customers.

Arbitrator: A trusted entity is responsible for solving disputes between the buyer and seller regarding the downloading of requested content. Based on the decision of arbitrator, a customer may be refunded the deposited amount.

5.1. Data Sharing

At first, owner initiates the digital data sharing by generating the meta data of original file. Meta data would include the information such as name of file, type, description, and size. Once the meta data is ready, it is uploaded to the IPFS along with complete file of data. The snippet of file uploading to IPFS is given next:

```
//upload the plain file meta
ipfs.files.add(buf, function (err, meta_result) {
  if(err) {
    console.log(err);
    return res.sendStatus(500);
  }
  console.log(meta_result);
  res.json({ "meta_hash": meta_result[0].hash,
    "file_hash": fileMeta.hash,
    "address": recipient_addr,
    "email": recipient_email });
});
```


Once the file is uploaded to IPFS, hashes of that data are generated by IPFS and returned back to the owner. In our proposed scenario, worker nodes are generated and their public–private key pair is stored in smart contracts. When the owner gets the hash by IPFS, it searches in the smart contract for verified worker nodes, who are responsible for providing decryption services to customers. Only that worker can provide the services, who is selected by the owner. When the owner receives the hash, SSS algorithm is used to split the IPFS hash of file into k number of shares. In response to these shares, owner decides the n number of random keys to be used for encryption. Once all the shares are encrypted, they are stored in the blockchain along with other important information such as; authorized recipient for the file. Data security is ensured by encrypting the hashes. The reason behind this is that, hash itself is not secure. It just represents a unique finger print for some data. If any unauthorized customer, who has not submitted deposit for digital content gets access to hash, then the complete file of data from IPFS can be fetched. In such a way, owner would be in complete loss of business. In the proposed scenario, only those customers are able to decrypt the hash, who have deposited the fund and have seen authorized by the worker nodes. The overall flow of file uploading on IPFS by owner is shown in Figure 1.

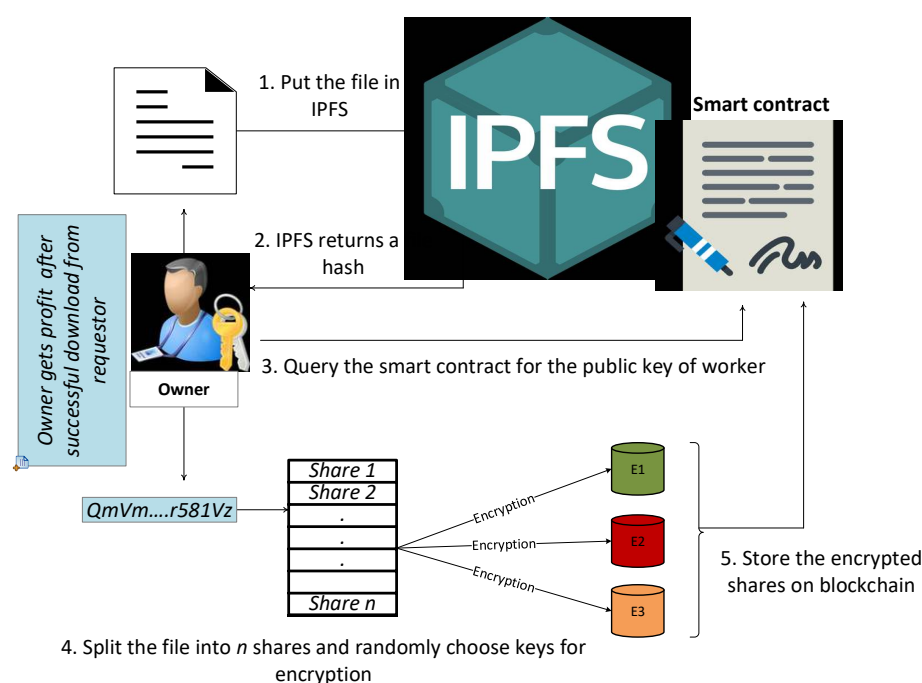


Figure 1. Data sharing on IPFS by owner.

SSS

Consider a secret, S , which is to be divided into n pieces so that each piece of data can be represented as S_1, \dots, S_n .

1. Combination of k or greater number of S_i makes it easy to compute original file
2. In other case, if there is $k - 1$ or less number of S_i then it would be difficult to reconstruct the actual data, therefore S becomes undetermined. It is known to be (k, n) threshold scheme, where $k = n$ means that all the involved entities need to reconstruct the actual secret S [5]. In our case, split size and number of shares are taken as 5 and 3, respectively. When shares are generated using SSS, these shares are encrypted and stored into smart contract. These shares can only be decrypted by verified workers whose public key is searched by the owner while uploading the file to IPFS.

5.2. Data Retrieval

A detailed system model for customer requesting the digital content is shown in Figure 2. A customer first checks the existing review of data by previous customers, so that quality of data can be properly verified before depositing the ethers. The details of review system are going to be discussed in Section 5.2.1.

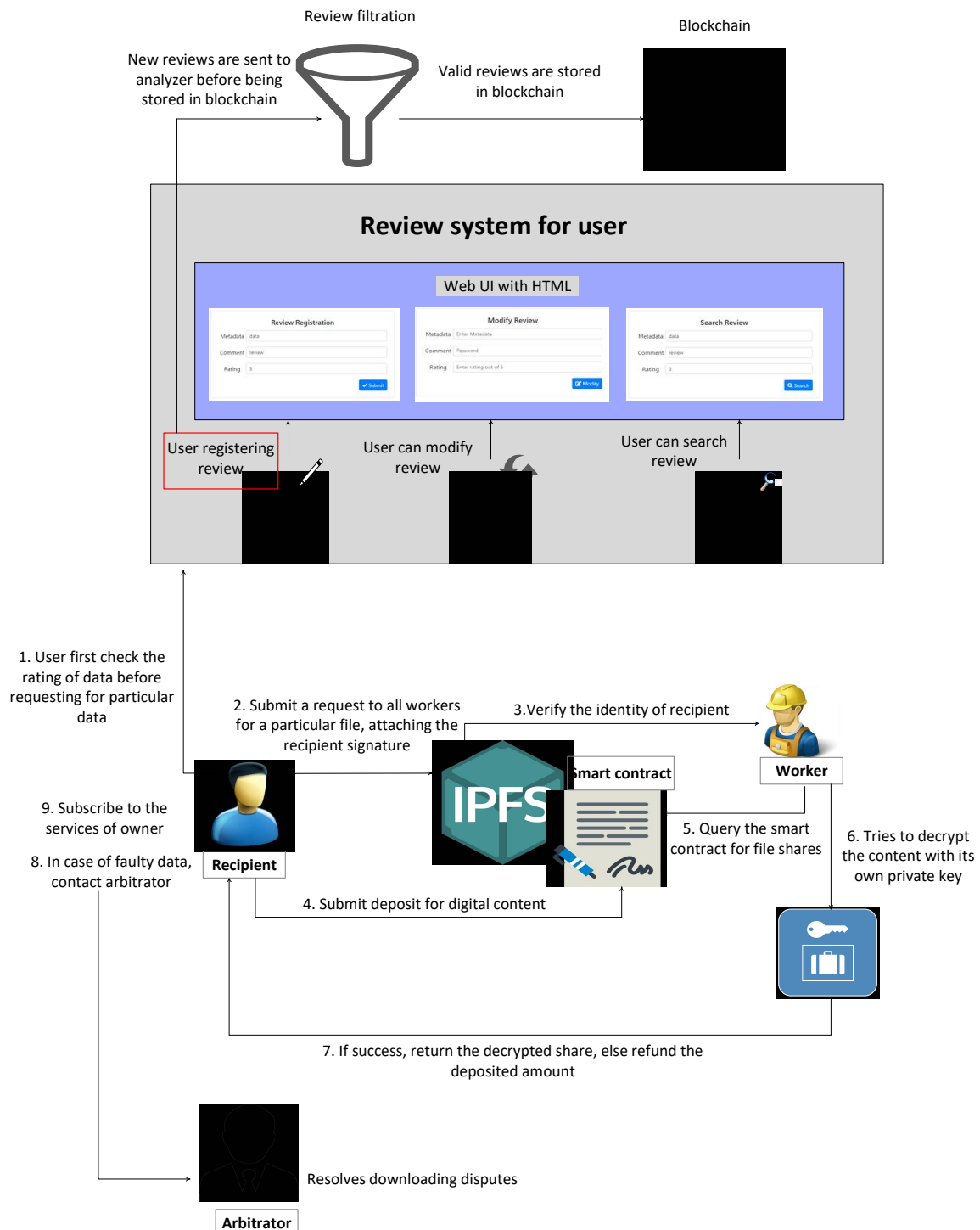


Figure 2. Data request by recipient.

5.2.1. Data Review System

For data trading between owner and customer, a reputation system is available to review the data being traded. Note that review, rating and reputation are used alternatively in this work. First of all, review system provides the ratings or scores provided by customers, who have already used the data. Based upon these reviews, new buyers can check the quality, reliability, and integrity of data. The immutability of reviews is ensured by using blockchain because data is tamper proof in blockchain. After checking the ratings, the new customer decides, whether the data is worth buying or not. After that, a request is submitted to workers for particular data based upon the reviews. Once the customer is satisfied from the reviews, a request is submitted to workers to access desired digital content. For this purpose, a smart contract is deployed with functions; *set reviews*, *get reviews* and *get ratings*. When a customer gets the data, he can submit his own reviews using the *set reviews* function. These reviews are being stored in blockchain as a result of each transaction carried out while submitting the review. *Isreviewexist* function allows the new customers to verify if there is any available review for particular data? If not, then the integrity and authenticity of data cannot be guaranteed at user's end.

5.2.2. Detection of Fake Reviews

To detect fake online reviews, a system is presented in Figure 3. At first step, a customer submits the review to the detection app, known as Watson analyzer. The tool first filters out the fake reviews based upon the emotional state. The emotion can be analyzed by the way review has been written. This task is performed using artificial intelligence techniques. After that, these reviews are cross-checked by the company or owner of the product. The cross-checking of reviews by owner is important because most of the time, market competitors register negative reviews about the product to degrade the market value. The owner can reject or accept the reviews after verification. In case, reviews get rejected by filtration tool and company, then the rejection reasons are delivered to customers to ensure transparency. The accepted reviews are stored in blockchain and cannot be altered because of immutability. Different cases of fake reviews registration is given as

- **Vote Manipulation:**
A situation where owner asks his friends or employees to post fake reviews regarding the quality of product.
- **Vote Buying:**
When companies hire special employees to post reviews using fake emails and user names by paying them handsome amount of money.
- **Vote on wrong thing:**
This category has nothing to do with fake reviews, but most of the times, customers do not actually give review on the product they are supposed to register review on. For example, a customer has booked a room in a hotel and may not like the beach closer to the hotel. Then, the customer is most likely to give negative reviews.

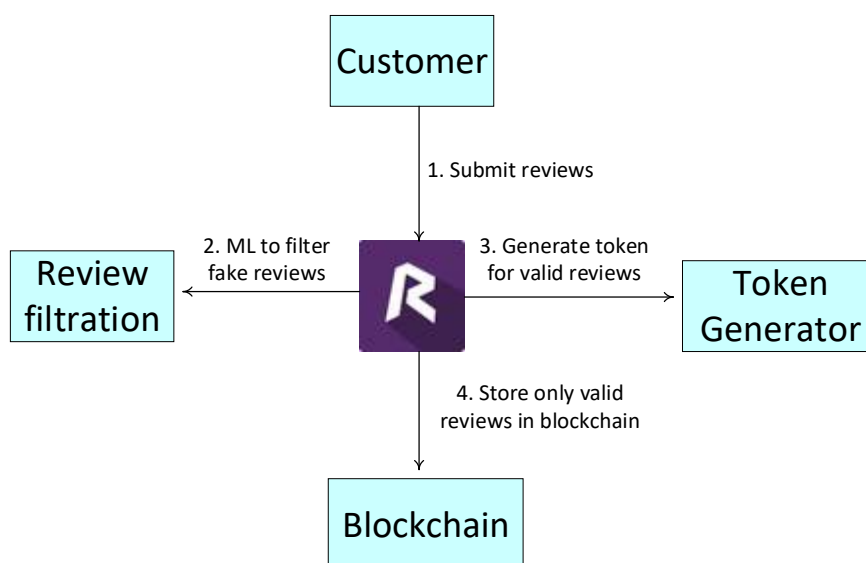


Figure 3. Fake review.

5.2.3. Data Distribution to Customers

Whenever a customer acquires a data, it sends a request to all the workers. A request is accompanied by the digital signatures of customers. New requestor is firstly identified by workers using RSA signatures. RSA is asymmetric system, where public and private key pairs are generated, where private is a secure one and public is shared among others.

```
//RSA key pair generation
key0.generateKeyPair(2048, 65537);\\
fs.writeFileSync("privateKey_0.pem", key0.exportKey('pkcs8-private-pem'));
fs.writeFileSync("publicKey_0.pem", key0.exportKey('pkcs8-public-pem'));
console.log("0 - RSA Key Pairs saved.");
```

After the identity of recipient is verified by workers, the recipient is supposed to submit the ether as a price of digital content. Once the payment is done, the workers proceed to search the encrypted shares in smart contract as requested. If recipient is unauthorized or identity is not confirmed as a valid requestor, then the smart contract terminates the transaction and workers ignore the received request. On the other hand, workers retrieve n encrypted shares from blockchain and decrypt them using their private keys. The important thing to notice here is that worker can only decrypt the shares on behalf of recipient if their public keys are selected by owner while sharing the content on IPFS. After decryption, the recipient is able to get the original IPFS file hash by reconstructing it. Once the data has been downloaded by customer, data rating is expected to be registered from that customer. Data rating depicts the quality of data and authenticity of owner. The future customers can easily verify the quality by looking at the reviews of particular data.

To ensure that the customer keeps on receiving the updated version of digital content, a smart contract with subscription function is deployed. This function allows the customer to subscribe the services of owner. Whenever owner of the data modifies the existing content or releases the new editions of file, all the old customers get notified about the release of new version. The advantage of version control is that existing customers who have already bought the content from same owner and paid the amount n form of digital currency, they do not have to pay the full amount again. Either they pay some percentage of the actual price or no price at all. In this way, customers get the updated content without paying the full amount again. These subscription services help the existing customers to get the latest edition of documents.

To motivate the customer for their authentic feedback registration, an incentive mechanism is introduced. There is an offer for customers that 10% of actual money deposited is refunded to them who submit valid reviews. When the smart contract gets notified about the review registration by particular customer, then the reviews are validated as shown in Figure 3. The smart contracts refunds the 10% of actual amount submitted if the review is not fake. If there is fake review registration, no amount is refunded to customer. In a similar way, new customer can search the reviews of data from the review based system. The reviews are being stored in blockchain and can be fetched upon customer request. The incentive assigned to customer is shown in Figure 4. The sequence diagram for all the participating entities in the network and their interaction is shown in Figure 5.

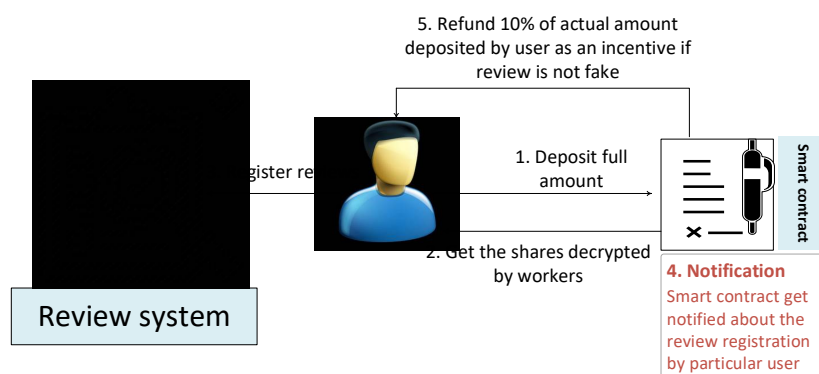


Figure 4. Incentive for user registering reviews.

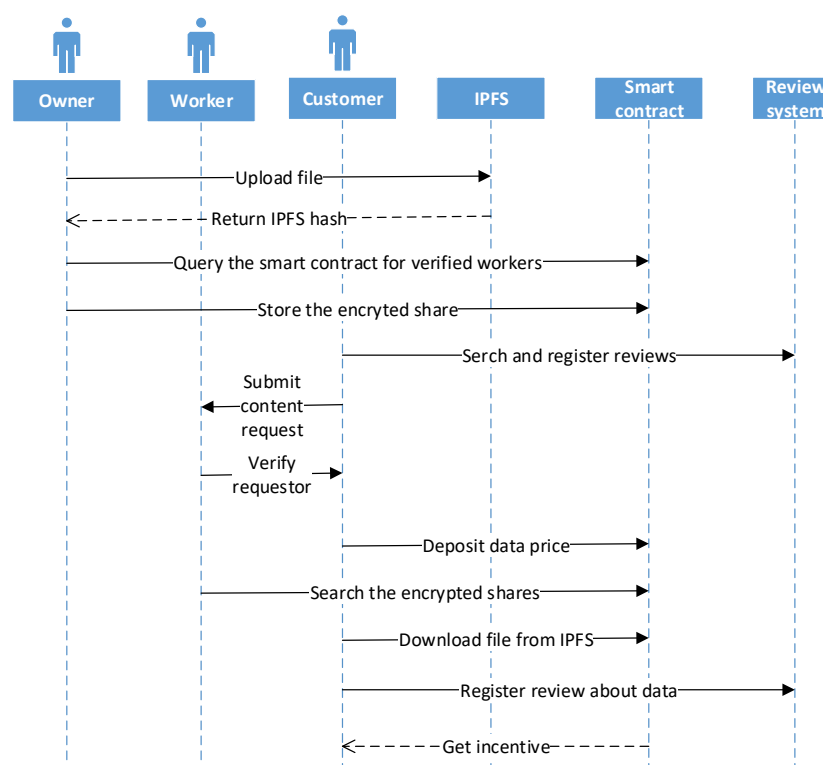


Figure 5. Sequence diagram for data sharing scenario.

5.2.4. Dispute Handling

When a customer downloads the digital content from IPFS server using hashes, there could be a possibility that customer may get faulty or incomplete data. Data is stored on IPFS nodes, so it may get altered while being stored on network nodes. In Figure 6, which is motivated from [31], three different cases are discussed, where arbitrator solves the disputes for customers. Arbitrator acts as an unbiased entity. The decisions like payment settlement or refund to the customer is merely dependent on the decision of arbitrator.

- *Case 1:* In this scenario, a customer submits the price for digital content, gets access to the hashes and gets the digital data on IPFS. There is no dispute in this case, customer successfully downloads the accurate and complete data from the server. The scenario for satisfied customer is shown in green box in Figure 6.
- *Case 2:* After downloading the file from server, customer gets to know that data is either incomplete or faulty. To verify the claim of customer, arbitrator decrypts the same hash and access the data from IPFS. When data is cross-checked by arbitrator, it finds the data as complete and accurate. As a result, it is concluded that customer's claim is wrong and there is no refund policy in this case.
- *Case 3:* When arbitrator tries to download the file from server, and same error occurs as highlighted by customer, then a refund is done to customer through smart contract by paying back the full deposited amount. In case, the arbitrator makes decision, that customer's claim is right, and there is some fault or error, while downloading the file from IPFS.

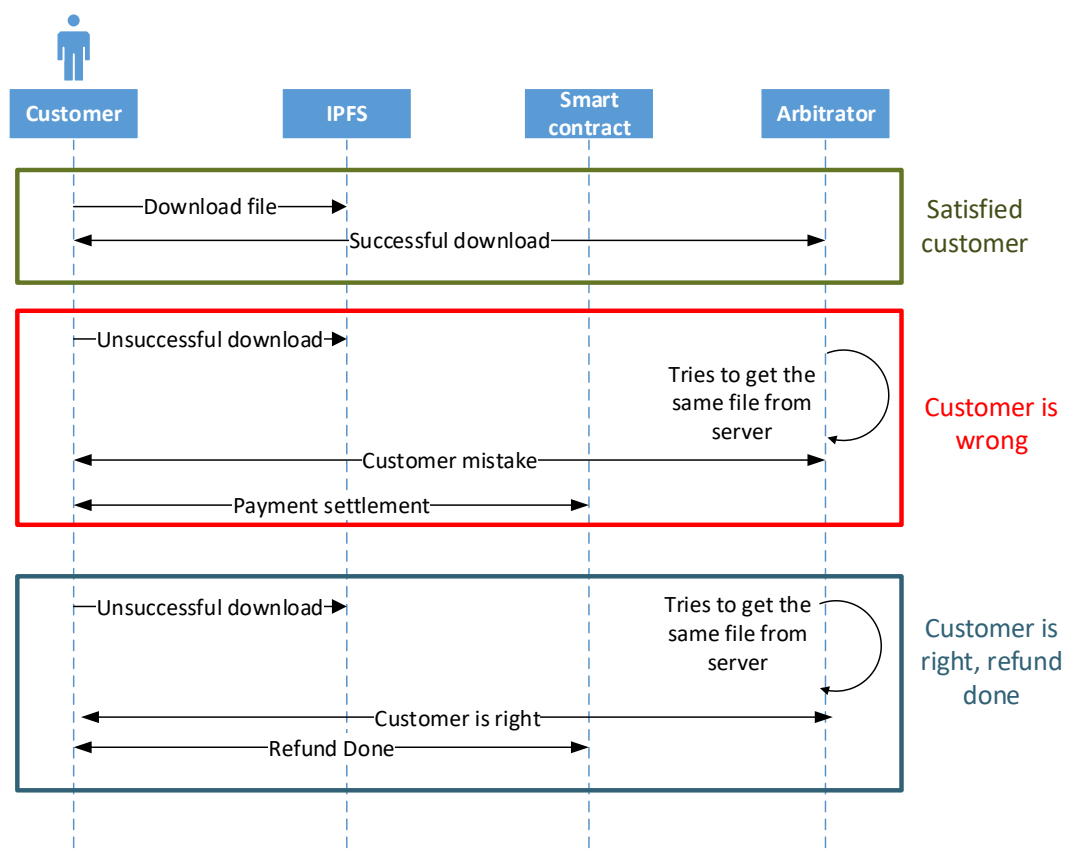


Figure 6. Sequence diagram for dispute handling.

5.3. Smart Contract Design

Three special variables are mainly used in this paper:

msg.sender: Message or transaction of the sender (current call). A contract address associated with the content creator, whenever a contract is deployed.

msg.value: Number of wei associated with the message, it is the cost in dollar (\$) when the number of wei are fixed. One ether equals to 10^{18} wei.

tx.origin: Accounts that call the smart contract.

5.3.1. IPFS Storage Contract

This contract provides the following primary functions:

addFile: Owner or creator of the contract can execute this function. Event is triggered when a new file is uploaded to IPFS server, which returns IPFS hash to owner and it gets stored in the smart contract. The uploaded content is only requested by valid requestor except those, who are blacklisted by the owner.

```
//Add file
function addFile(bytes32 id, bool isVisible) public {
    require(! blacklist[msg.sender] && FileMap[id].timestamp == 0);
    .
    .
    .
    emit confirmFileIndexID( FileList.length-1, id );
}
```

deleteFile: This function is only executed when owner desires to delete specific file from the server. When file deletion is required, index and address arguments are passed to this function.

```
// File deletion
function deleteFile(uint index, bytes32 id, address[] memory addr) public {
    .
    .
    .
    FileList[index] = 0;
    emit confirmFileDeletion( index, id );
}
```

setRecipient When a data is requested by user, it is searched and mapped to the request. If the desired data is found, then a list of recipients is maintained by smart contract to assign the digital content.

```
//Search content
function setRecipients(bytes32 id, bytes32[] memory newShares) public {
    .
    .
    .
    for (uint j = 0; j < addr.length; j++) {
        FileMap[id].recipients[ addr[j] ] = value;
    }
}
```

5.3.2. Data Recipient Contract

This contract is specifically designed for data recipient. Whenever a recipient needs to buy a digital content, he/she has to deposit some ethers after getting identity verification from workers.

```
// Deposit money
function DepositEtherToBuycontent() public payable {
    .
    .
    .
    emit DepositMoneyDone(msg.sender, "Money Deposited ,
    Customer Waiting for decrypted share"); //trigger event
}
```

Customer download result: This function verifies that the file is successfully downloaded from IPFS against decrypted hash. It ensures that the hash was decrypted successfully and correctly. After downloading the file from IPFS, the customer returns true to the bool function.

File server results: After the successful download confirmation from the customer, file server also confirms that the customer was able to download the requested data from the system. The Confirmation from customer and file server is recorded to eliminate any conflict between server and the customer. In case, the file server returns true to the successful downloading of file and customer claims that no file was downloaded or data was unavailable on the server, a third unbiased entity is introduced to resolve the conflicts by authors in [31]. The third party; arbitrator downloads the file using the same token provided to customer. If it gets successful, it means that the claim of customer is false, and no refund is given to customer. Otherwise, the deposited fund is given back to customer in case of incorrect data.

5.3.3. Review System Contract

The smart contract of review system has mainly four functions.

Set review: This function allows the customer to upload the reviews regarding the used data from particular owner. The size of reviews vary, depending upon the description added by customer. Therefore it shows different values of gas consumption for different customers. A review interface is provided to customers containing fields, such as, meta data of file against which reviews need to be submitted, comments and rating of data. After that, a submit button is clicked to store the reviews in blockchain.

Get review: This function in smart contract allows new users to search for the reviews of available data. A new customer can search the submitted reviews by using the same query words against which prior customers have submitted the review. These reviews include the description about the quality of product. Once the new customer is satisfied from the comments and reviews, it can process further to buy the data from owner.

Get rating: The customers need to rate the product from 1 to 5, starting from lowest to highest ranking. A review from 3.5 to 4 is considered satisfactory, whereas 5 shows the highest ranked product among the customers. The rating of any product is fetched from the blockchain per the customer's request.

Is review exist: It is possible that for any product or data, a review may not exist. Therefore, a customer first need to check the availability of review. In case, there is no available review for a file, the system returns no results. Thus, a customer cannot rely on the quality of that product. For this purpose, an incentive for the customers is introduced, so that they are encouraged to register their review for every downloaded file from IPFS.

6. Implementation Details

In this section, the implementation details are provided. The proposed system is a private network of Ethereum blockchain. Ethereum is open source distributed platform that makes efficient use of solidity. A programmable language that allows to write smart contracts (scripting language).

6.1. Simulation Setup

The specifications for implementation setup are: Intel(R) core(TM) m3-7Y30 CPU@1.61 GHz, 8 GB RAM, 64 bit operating system and X64-based processor. The programming language is solidity to write smart contract. Front-end web GUI is developed using Bootstrap 4.0 and Javascript for user interactive forms. The primary tools used to develop this system is given below:

Visual Studio Code

A lightweight cross-platform code editor designed by Microsoft for multiple operating systems. It allows powerful debugging with a variety of tools, highlighted syntax, intelligent completion of code, build in Git control and code refactoring [34].

Ganache

A blockchain-based emulator used to execute several tests and commands. It controls the blockchain operation by inspecting the states of system. Formerly, its name was Test RPC, which was later renamed to ganache. It provides information such as; Visual MNEMONIC (key phrase for ganache accounts) and account addresses [35].

Metamask

It is a browser extension that allows a connectivity to distributed web. Instead of running the full Ethereum node, it run Ethereum decentralized applications in browser [36].

6.2. Simulation Results

To run the corresponding smart contracts, the gas price is adjusted to 2 Gwei. 1Gwei is equivalent to 10^9 wei, which is approximately 10^{-9} ethers. The cost measured for smart contracts are given in Tables 2 and 3.

Table 2 refers to the gas consumption values for multiple functions executed in smart contract. After one time deployment of IPFS storage smart contract, the transaction and execution gas is recorded as 1,808,235 and 1,338,219, respectively, which are noticed to be unchanged after multiple transactions. Here, the transaction cost is the amount of gas required to send data on blockchain network and execution cost is the computational fee required to execute the function. The actual cost is measured as \$0.55 USD for the contract creation. When an owner uploads a file to server, *addFile* function performs the operation costing \$0.016 USD as the actual cost. Similarly, when any file is deleted from the system, *deleteFile* function is invoked. As a result, the *ConfirmFileDeletion* event is triggered. The actual cost for the execution of this function is recorded as \$0.0088 USD. The recipient list maintains the record of authorized requestor who are validated by workers. Whenever a new requestor gets authenticated, it becomes a part of this list and *setReceipient* function is executed. Execution of this function costs \$0.0091 USD. Requestors which are not authorized and authenticated by workers are stored in blacklist for future reference. So that, whenever, any person tries to send a data request to owner, its request will be immediately discarded by the system. As a result, *setBlacklist* function is executed which has a recorded cost of \$0.0043 USD. Gas consumption for all aforementioned functions is given in Figure 7.

For data recipient contract, Table 3 includes the gas consumption values for the functions that are executed in this contract. For the contract creation, the amount of transaction and execution gas consumed is 1,723,531 and 1,277,355 respectively and the actual cost is \$0.52 USD. When the file is requested from the worker, the identity of requestor is first verified. After successful verification,

the user is allowed to deposit the price of digital content which is requested from the server. In response to this action, *CustomerPayment* operation is performed with the cost of \$0.021 USD. The transaction and execution gas consumed for this operation is recorded as 70,190 and 47,510 respectively. Smart contract is tested with two requestor with Ethereum addresses as *0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db* and *0x583031d1113ad414f02576bd6afabfb302140225*. Figure 8 shows that the ether equivalent to the price of digital content are deducted from these two Ethereum accounts. The other two functions of this contract are *DownloadResults* and *confirmFileserverResults*. Once the file is downloaded from the server, the download results are later verified by the recipient; whether the downloaded file is the requested one or not. The actual cost for these functions are \$0.0092 USD and \$0.016 USD, respectively. The gas consumption graph for this contract is given in Figure 9.

Table 2. Smart contract cost test (IPFS storage: gas price = 2 Gwei), 1 ether = 150 USD.

Functions	Transaction Gas	Execution Gas	Actual Cost (ether)
Contract creation	1,808,235	1,338,219	0.00361647
Add file	67,512	53,948	0.000107896
Delete file	29,206	19,034	0.000058412
Set recipient	34,098	30,231	0.000060462
Set blacklist	31,290	14,203	0.000028402

Table 3. Smart contract cost test (Proof of delivery: gas price = 2 Gwei), 1 ether = 150 USD.

Functions	Transaction Gas	Execution Gas	Actual Cost (ether)
Contract Creation	1,723,531	1,277,355	0.003447062
Customer Payment	70,190	47,510	0.00014038
Download Results	30,373	30,412	0.000060746
Confirm file Server Results	54,428	76,930	0.000108856

To add a review system in a data sharing scenario, a separate smart contract is deployed. A user needs to enter the account, which become the identity of the user to view the reviews about the data. To register new reviews, a Register button must be clicked by user to fill out the fields like Metadata, comments, and ratings. In the same way, reviews can be searched by new user by clicking the search button and adding the query.

The review system smart contract is deployed on Ethereum platform. To deploy any contract, specific amount of gas is required. The maximum limit of gas is pre-decided by creator, which is 3,000,000. Two types of gas consumed is observed; transaction gas and execution gas. In Figure 10, the highest gas is consumed by *Add file* function, as uploading review file on the system requires time and cost. The amount of gas varies depending upon the size of file to be uploaded on the review system. The rest of the operations, such as, *get review*, *get rating*, *is review exist* show the minimum cost, as they does not require some additional uploading. These functions just search the already existing review and their rating from the system. Execution cost depends upon the number of lines of codes and the logical operation being performed within the function. Ethereum yellow paper [6] is referred to check the gas consumed for different types of operations.

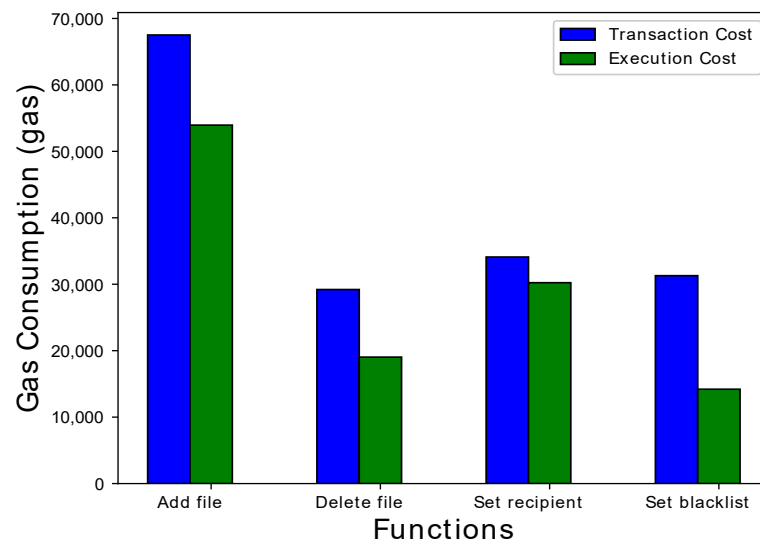


Figure 7. Gas consumption for owner side contract.

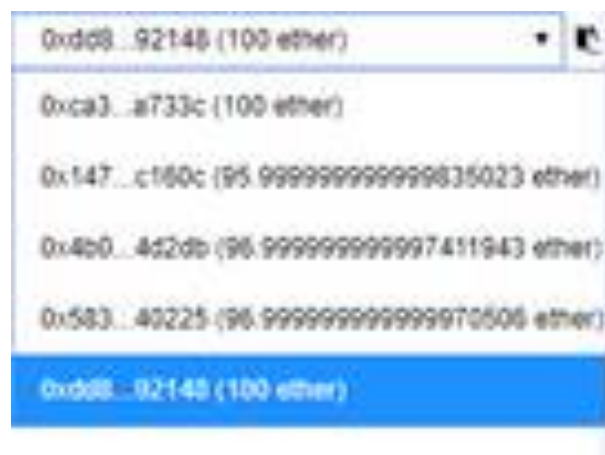


Figure 8. Money deposit from recipient.

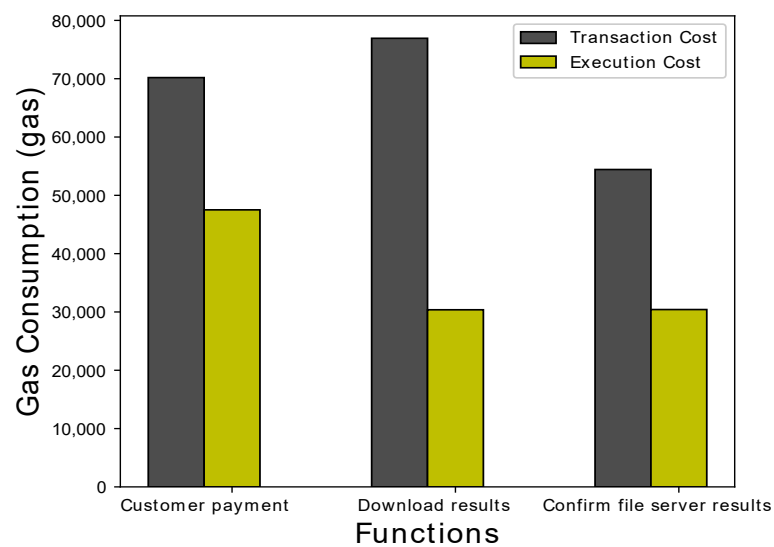


Figure 9. Gas consumption for recipient side contract.

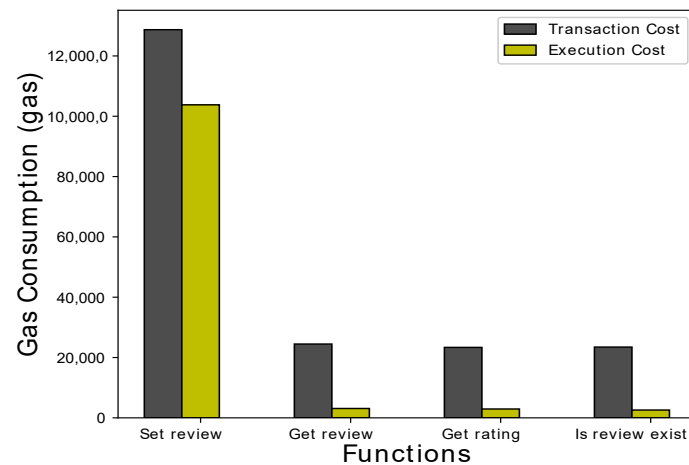


Figure 10. Gas consumption for review system.

Figure 11 represents the average computation time of CPU against the number of shares generated in SSS. It is evident from the graph that as the number of shares increases, the average CPU time increases exponentially. In this scheme, when $k = n$, the shares are generated with polynomial whose degree increase with every additional k in the system. The highest computation time is recorded as 0.66 ms when k is equal to 11 shares and 0.32 ms is the lowest computational time when number k is 3. Higher the number of shares, more is the increase in computational complexity because greater the number of secrets, the more time is taken to encrypt each share. Less number of shares are encrypted in shorter amount of time. In our case, the number of shares is 3 with split size 5 for the simplicity of scenario. In this case, the average computation time is least which is 0.32 ms.

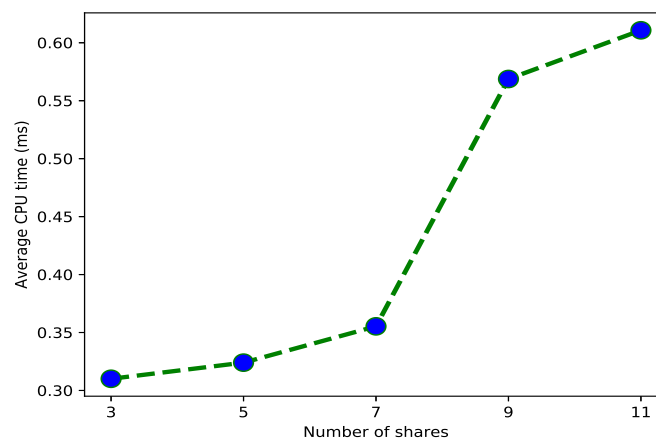


Figure 11. Computational time for number of shares.

SSS is used in our scenario to split and then encrypt the hashes of file uploaded to IPFS. A comparison between two encryption schemes, i.e., advanced encryption standard (AES) 128 and AES 256. The encryption computational time of both schemes is recorded as 6.64 and 8.62 ms respectively, whereas the encryption time for SSS is 0.37. The significance difference in the computational time of these schemes is because of search space size. AES 128 offers 2^{128} combinations of keys, while AES 256 offer 2^{256} keys. The larger the combination of keys, greater the computational time as shown in Figure 12. In contrast, SSS is showing least computational time because there are no combination of keys. It works as a polynomial function of (k, n) combination which means the only the defined number of splits are to be encrypted; this is why the computational time of SSS is least as compared to other scheme.

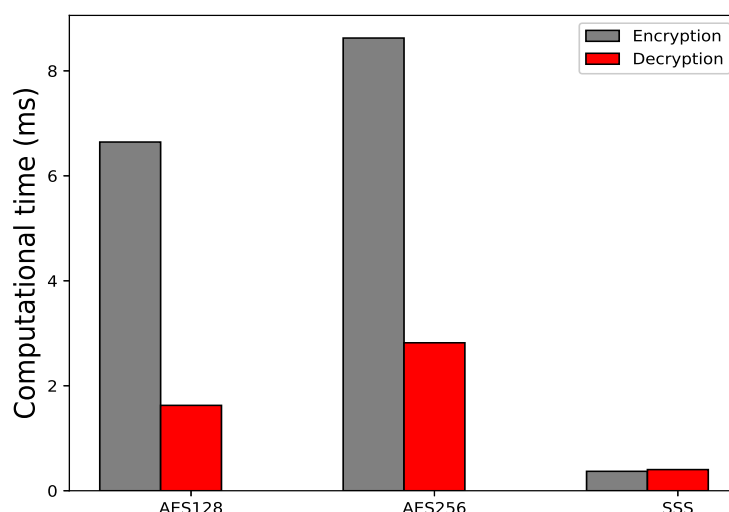


Figure 12. Computational time for different encryption schemes.

The total transaction gas, execution gas, and actual cost for multiple functions executed in IPFS are 1,970,341, 1,455,635 and 0.003871642 (ether), respectively. For proof of delivery, total transaction gas is 1,878,522, overall execution gas is 1,432,207 and total actual cost is 0.003757044 (ethers) for all of the functions. The smart contract cost test for the review system shows total 848,856 transaction gas, 577,785 overall execution gas, and 0.001695 total actual cost in terms of ethers. These results show that overall high execution gas, transaction gas and actual cost are recorded for IPFS smart contract test and minimum are recorded for review system. Moreover, overall actual cost of the system for all three smart contracts is calculated as 0.009323686 (ethers). Whereas, total transaction gas and execution gas of the complete system are 4,697,719 and 346,5627, respectively.

In Figure 13, different values of gas consumption are shown. Each value varies depending upon the length of review added by user. The amount of gas consumed changes as the length of input string changes. Multiple gas consumption bars are shown in Figure 13. The x axis shows the independent variable, whereas, y axis represent dependent variable which is gas consumption. Each bar shows higher value of gas consumed than the previous one. It means that the string length is increased from bar 1 to bar 4. Greater the length on input string, higher the gas consumed [33].

Similarly Figure 14 shows a relationship between input string length and mining time of transactions. Mining time varies according to length of string entered by user as it is a dependent variable. However, it is temporary and may not necessarily increase with greater string length. It also does not depend upon the lines of codes in a system. The condition of network or the choice of miners is the major factor that influence the mining time of transaction. Sometimes, for greater input length, the mining time may not remain the same due to performance of miners. Table 4 shows the functions of review system with transaction and execution gas along with actual cost in terms of ethers [33].

Table 4. Smart contract cost test (Review system: gas price = 2 Gwei), 1 ether = 150 USD.

Functions	Transaction Gas	Execution Gas	Actual Cost (ether)
Contract creation	673,317	468,305	0.001346
Set review	128,705	103,785	0.000257
Get review	23,364	3106	0.000046
Get rating	23,470	2589	0.000046
Does review exist	-	-	-

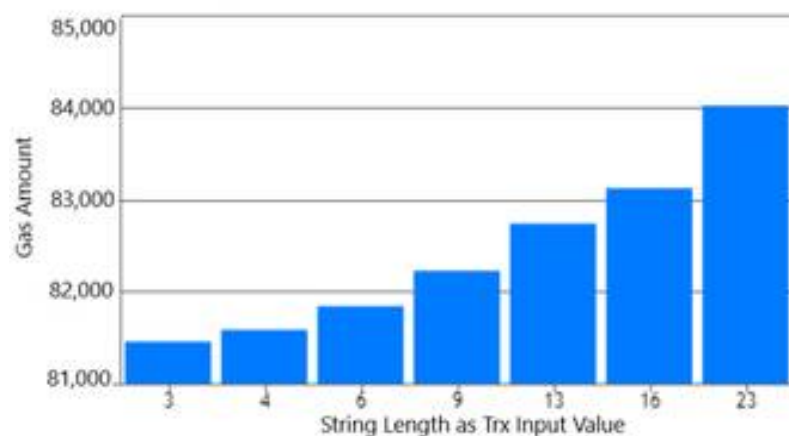


Figure 13. Change of amount of gas consumed with different input lengths of transactions.

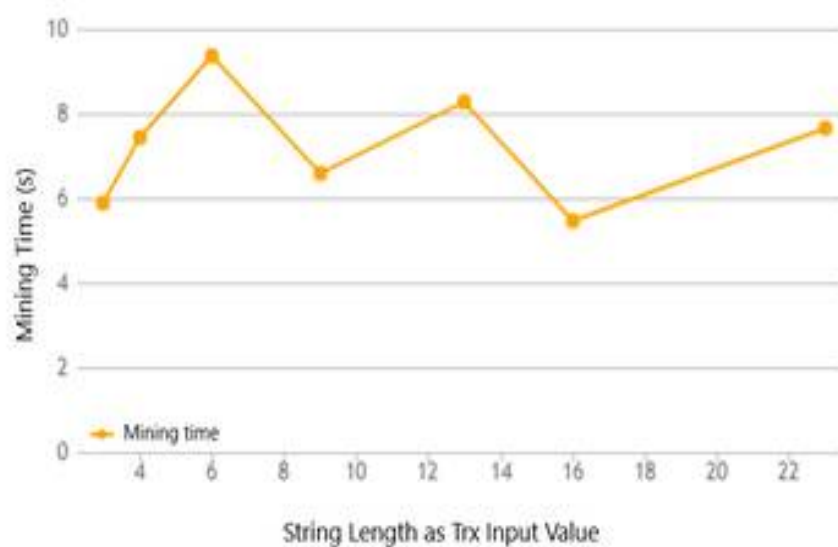


Figure 14. Change of mining time with different input lengths of transactions.

7. Conclusions

In this paper, a blockchain-based secure data sharing and delivery of digital assets framework is presented. The main aim of this proposed scenario is to provide data authenticity and quality of data to customer as well as a stable business platform for owner. A decentralized storage IPFS provides the solution for bloating problem at owner's end. Data hashes returned by the IPFS are encrypted using SSS, so that a customer who has not deposited digital content price, cannot access the data. In this way, owner is satisfied from any type of hash leakage to unauthorized customer. Data authenticity is ensured by adding a review based system, where customers can record their comments and rating about the data. In this way, new customers can judge the quality of data, therefore money can be saved at customer's end. Different smart contracts are designed for various purposes; such as, owner smart contract for uploading the file on IPFS and encrypting the hashes. The other smart contract is for user side. A user can access the file hashes from smart contract after authentication from worker nodes. Finally, the review system smart contract can help new and old customers to search and register reviews. Simulation results are performed for gas consumption and cost analysis of these smart contracts. Each function consumes different gas value depending upon the logics and complexity of operations being performed in each function. Moreover, computational complexity of

other encryption schemes are compared with the one used in this paper. Results have shown that the encryption scheme used in this paper for encrypting the shares has least computational time.

In future, our focus will be on the trading and monetization of data generated by IoT devices using blockchain.

Author Contributions: All authors contributed the main conceptual and scientific ideas. F.A.A.-z., R.K. and A.M.Q. did the literature review and performed the simulations. M.N., N.J. and M.K.A. did the mathematical modeling and write the manuscript. M.S. did the revision and contributed to the final version of the manuscript along with M.N. and N.J.

Funding: This work was supported by the National Research Foundation of Korea (NRF) through the Brain Korea 21 Plus Program under Grant 22A20130012814.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Shrestha, A.K.; Vassileva, J. Blockchain-Based Research Data Sharing Framework for Incentivizing the Data Owners. In *International Conference on Blockchain*; Springer: Cham, Switzerland, 2018; pp. 259–266.
- Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: <https://bitco.in/pdf/bitcoin.pdf> (accessed on 6 April 2019).
- Crosby, M.; Pattanayak, P.; Verma, S.; Kalyanaraman, V. Blockchain technology: Beyond bitcoin. *Appl. Innov.* **2016**, *2*, 71.
- Benet, J. Ipfs-content addressed, versioned, p2p file system. *arXiv* **2014**, arXiv:1407.3561.
- Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [[CrossRef](#)]
- Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
- Hopwood, D.; Bowe, S.; Hornby, T.; Wilcox, N. *Zcash Protocol Specification*; GitHub: San Francisco, CA, USA, 2016.
- Wu, A.; Zhang, Y.; Zheng, X.; Guo, R.; Zhao, Q.; Zheng, D. Efficient and privacy-preserving traceable attribute-based encryption in blockchain. *Ann. Telecommun.* **2019**, *74*, 401–411. [[CrossRef](#)]
- Zhang, Z.; Zhao, L. A Design of Digital Rights Management Mechanism Based on Blockchain Technology. In *International Conference on Blockchain*; Springer: Cham, Switzerland, 2018; pp. 32–46.
- Ma, Z.; Jiang, M.; Gao, H.; Wang, Z. Blockchain for digital rights management. *Future Gener. Comput. Syst.* **2018**, *89*, 746–764. [[CrossRef](#)]
- Rowhani-Farid, A.; Allen, M.; Barnett, A.G. What incentives increase data sharing in health and medical research? A systematic review. *Res. Integr. Peer Rev.* **2017**, *2*, 4. [[CrossRef](#)]
- Zhu, L.; Wu, Y.; Gai, K.; Choo, K.K.R. Controllable and trustworthy blockchain-based cloud data management. *Future Gener. Comput. Syst.* **2019**, *91*, 527–535. [[CrossRef](#)]
- Li, J.; Wu, J.; Chen, L. Block-secure: Blockchain based scheme for secure P2P cloud storage. *Inf. Sci.* **2018**, *465*, 219–231. [[CrossRef](#)]
- Li, J.; Wang, X.; Huang, Z.; Wang, L.; Xiang, Y. Multi-level multi-secret sharing scheme for decentralized e-voting in cloud computing. *J. Parallel Distrib. Comput.* **2019**, *130*, 91–97. [[CrossRef](#)]
- Huang, Q.; Ma, Z.; Yang, Y.; Niu, X.; Fu, J. Attribute based DRM scheme with dynamic usage control in cloud computing. *China Commun.* **2014**, *11*, 50–63. [[CrossRef](#)]
- Sun, W.; Yu, S.; Lou, W.; Hou, Y.T.; Li, H. Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 1187–1198. [[CrossRef](#)]
- Xia, Q.I.; Sifah, E.B.; Asamoah, K.O.; Gao, J.; Du, X.; Guizani, M. MeDShare: Trust-less medical data sharing among cloud service providers via blockchain. *IEEE Access* **2017**, *5*, 14757–14767. [[CrossRef](#)]
- Hammi, M.T.; Hammi, B.; Bellot, P.; Serhrouchni, A. Bubbles of Trust: A decentralized blockchain-based authentication system for IoT. *Comput. Secur.* **2018**, *78*, 126–142. [[CrossRef](#)]
- Liang, W.; Tang, M.; Long, J.; Peng, X.; Xu, J.; Li, K.C. A Secure Fabric Blockchain-based Data Transmission Technique for Industrial Internet-of-Things. *IEEE Trans. Ind. Inform.* **2019**, *15*, 358–3592. [[CrossRef](#)]

20. Steichen, M.; Fiz Pontiveros, B.; Norvill, R.; Shbair, W. Blockchain-Based, Decentralized Access Control for IPFS. In Proceedings of the 2018 IEEE International Conference on Blockchain (Blockchain-2018), Halifax, NS, Canada, 30 July 2018–3 August 2018; pp. 1499–1506.
21. Gaby, G.; Chandra, L.; Enderson, T. Towards Secure Interoperability between Heterogeneous Blockchains using Smart Contracts. In Proceedings of the Future Technologies Conference (FTC), Vancouver, BC, Canada, 15–16 November 2017; pp. 73–81.
22. Dias, J.P.; Reis, L.; Ferreira, H.S.; Martins, Â. Blockchain for access control in e-health scenarios. *arXiv* **2018**, arXiv:1805.12267.
23. Fukumitsu, M.; Hasegawa, S.; Iwazaki, J.; Sakai, M.; Takahashi, D. A proposal of a secure P2P-type storage scheme by using the secret sharing and the blockchain. In Proceedings of the 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), Taipei, Taiwan, 27–29 March 2017; pp. 803–810.
24. Guo, R.; Shi, H.; Zhao, Q.; Zheng, D. Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems. *IEEE Access* **2018**, *6*, 11676–11686. [[CrossRef](#)]
25. Nizamuddin, N.; Salah, K.; Azad, M.A.; Arshad, J.; Rehman, M.H. Decentralized document version control using ethereum blockchain and IPFS. *Comput. Electr. Eng.* **2019**, *76*, 183–197. [[CrossRef](#)]
26. Agyekum, O.; Opuni-Boachie, K.; Xia, Q.; Sifah, E.B.; Gao, J.; Xia, H.; Du, X.; Guizani, M. A Secured Proxy-Based Data Sharing Module in IoT Environments Using Blockchain. *Sensors* **2019**, *19*, 1235. [[CrossRef](#)]
27. Wang, S.; Zhang, Y.; Zhang, Y. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *IEEE Access* **2018**, *6*, 38437–38450. [[CrossRef](#)]
28. Panescu, A.T.; Manta, V. Smart Contracts for Research Data Rights Management over the Ethereum Blockchain Network. *Sci. Technol. Libr.* **2018**, *37*, 235–245. [[CrossRef](#)]
29. Dai, M.; Zhang, S.; Wang, H.; Jin, S. A low storage room requirement framework for distributed ledger in blockchain. *IEEE Access* **2018**, *6*, 22970–22975. [[CrossRef](#)]
30. Nizamuddin, N.; Hasan, H.; Salah, K.; Iqbal, R. Blockchain-Based Framework for Protecting Author Royalty of Digital Assets. *Arab. J. Sci. Eng.* **2019**, *44*, 3849–3866. [[CrossRef](#)]
31. Hasan, H.R.; Salah, K. Proof of delivery of digital assets using blockchain and smart contracts. *IEEE Access* **2018**, *6*, 65439–65448. [[CrossRef](#)]
32. Chen, Y.; Li, H.; Li, K.; Zhang, J. An improved P2P file system scheme based on IPFS and Blockchain. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; pp. 2652–2657.
33. Park, J.S.; Youn, T.Y.; Kim, H.B.; Rhee, K.H.; Shin, S.U. Smart contract-based review system for an IoT data marketplace. *Sensors* **2018**, *18*, 3577. [[CrossRef](#)]
34. Truffle Suite. Available online: <https://truffleframework.com/tutorials/configuring-visual-studio-code> (accessed on 23 April 2019).
35. Truffle Suite. Available online: <https://truffleframework.com/docs/ganache/overview> (accessed on 23 April 2019).
36. MetaMask. Available online: <https://metamask.io/> (accessed on 23 April 2019).



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).