IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 5 August 2022, accepted 1 September 2022, date of publication 5 September 2022, date of current version 14 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3204274

RESEARCH ARTICLE

Blockchain and NFTs for Time-Bound Access and Monetization of Private Data

MOHAMMAD MADINE[1], KHALED SALAH[1], (Senior Member, IEEE), RAJA JAYARAMAN[2], AMMAR BATTAH[1], HAYA HASAN[1], AND IBRAR YAQOOB[1]

[1]Department of Electrical Engineering and Computer Science, Khalifa University of Science and Technology, Abu Dhabi 127788, United Arab Emirates
[2]Department of Industrial and Systems Engineering, Khalifa University of Science and Technology, Abu Dhabi 127788, United Arab Emirates

Corresponding author: Ibrar Yaqoob (ibrar.yaqoob@ku.ac.ae)

This work was supported by the Khalifa University of Science and Technology under Award CIRA-2019-001.

ABSTRACT Digital data has enabled organizations to anticipate future threats, opportunities, and trends. However, digital data owners do not know how their data is accessed, shared, and monitized. In this paper, we propose using blockchain technology and non-fungible tokens (NFTs) to enable time-bound access and monitization of private data. Our approach allows users to upload encrypted content and mint it into NFTs. Other users can access the NFTs' content by requesting a purchase or a license. Purchasing content transfers the ownership of the NFTs to the buyer; whereas, licensing them permits accessing the private data for a limited period of time, after which the data gets automatically deleted. Our developed approach uses the decentralized application (DApp), proxy reencryption (PRE), InterPlanetary File System (IPFS), and trusted execution environment (TEE) for managing a fully decentralized and robust system. We implement a proof-of-concept system in an Ethereum-based environment, which is used for testing and vulnerability checks. We present the cost and security analyses and discuss the generalization aspect of the solution. Our smart contracts and testing scripts are publicly available under an open-source license.

INDEX TERMS Blockchain, non-fungible tokens, Ethereum, data sharing, data monetization, decentralized application, trusted execution environment, decentralized storage, proxy reencryption.

I. INTRODUCTION

With the growing amount of data on the internet [1], accompanied by cloud providers gaining dominance over the data storage and delivery networks [2], [3], users no longer have ownership over their data. Presenting a new generation of solutions not only provides users with the ability to control their data but also offers an alternative to the vulnerabilities and lack of availability often found in centralized frameworks [4], [5], [6]. Furthermore, additional features can be integrated as part of the new solution, such as the ability to present data ownership certificates, time-bound data sharing, and secure data monetization. Such a solution can benefit from a decentralization framework, in which no entity can take control over the actions of the data owner [7], [8].

The associate editor coordinating the review of this manuscript and approving it for publication was Yassine Maleh.

A primary use case that embodies all aspects of data ownership, sharing, and monetization is concerned with patient data. Patients are increasingly more cautious about their right to own their data. Ideally, patients should be able to directly receive and verify their medical reports and information, such as x-ray scans and genomic sequences, so no other party can claim ownership over the data or have access to it without permission from the patient. Figure 1 illustrates this use case, which also shows the ability of the patient to grant a paid license of a subset of the data to a medical institution, where the license expires after one week.

The researchers in this space investigate a solution that ideally provides data storage and ownership, time-bound sharing, and monetization. These objectives are defined as follows:

1) **Storage and ownership**: Users can persistently upload and store their public or private data. Using NFTs, the system manages and verifies claims of data ownership.

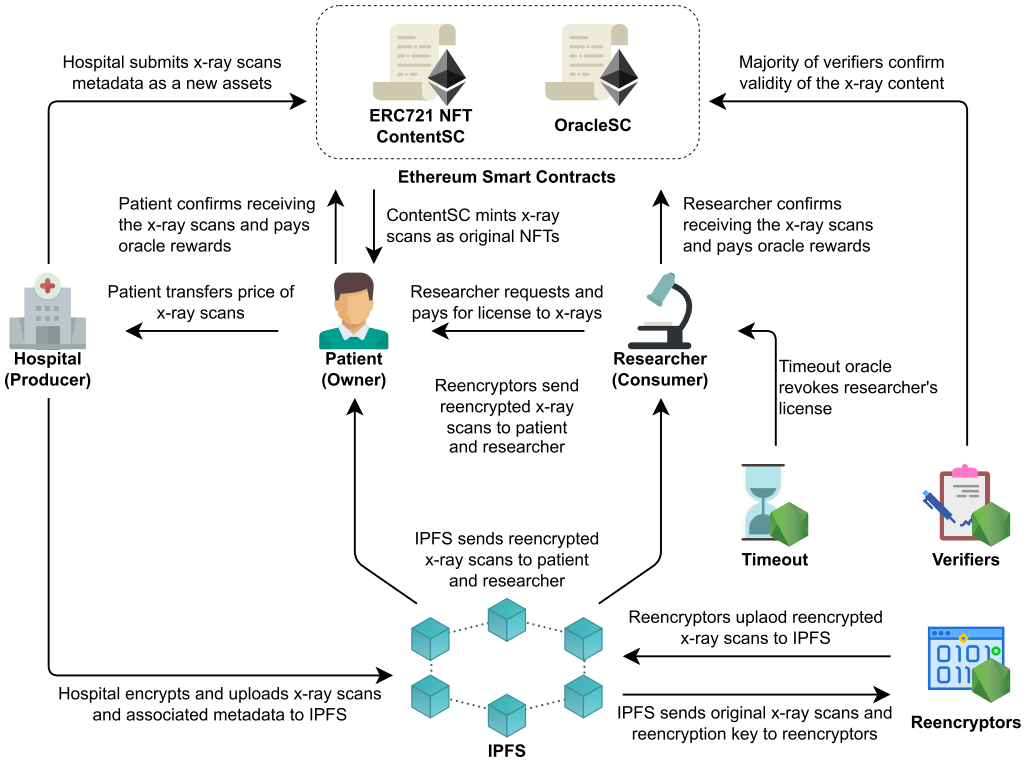

FIGURE 1. Overview diagram of our proposed solution for time-bound data sharing in healthcare using NFTs and blockchain smart contracts.

It also keeps track of who owned the data at different points in time, such as when ownership gets transferred.

2) **Time-bound sharing**: The system preserves the rights of the owners upon sharing by preventing attempts to duplicate the data and enforcing digital rights management (DRM) rules. Owners can specify what subset of data to share, who can access the data, and for how long the data can be obtained and viewed, for example, by setting an expiration policy.
3) **Monetization**: Users can monetize their data while it is shared, accessed, or used by third parties. For example, medical research institutions must pay for patient data as long as it is in use.

Although extensive studies propose and implement the individual components of the problem in question, there is a gap in the literature when it comes to integrating the components into a comprehensive solution. To illuminate this void, we believe taking advantage of decentralization technology generally, and blockchain specifically, can provide scalability and security [9], [10]. More importantly, building the solution with a decentralized mindset further empowers users to take control over their data [11], and to gain confidence in the system.

The introduction of Bitcoin in 2008 has been one of the leading events in shifting the internet from a centralized state to a decentralized one. It is a highly regarded cryptocurrency, mainly because of having blockchain as its backbone technology [12], [13]. The decentralization of the internet gained further traction among developers with the concept of smart contracts in Ethereum [14], which enabled blockchain technology outside the cryptocurrency boundaries. Over the past couple of years, a strong push for bringing decentralization to the mainstream public has been driven by the successful utilization of the technology, such as non-fungible tokens (NFTs) for arts and games, and enabling users to monetize their genomic sequence by sharing it with pharmaceutical companies [15], [16].

Blockchain systems comprise distributed peer-to-peer networks that store ledgers made of blocks of stateful transactions. A consensus mechanism is agreed upon among all peers to decide what transactions are deemed correct. Permissionless public blockchains, in which any entity can join the network and all transactions are kept in plaintext, usually use proof-of-work and proof-of-stake consensus mechanisms [17]. Whereas permissioned private and consortium blockchains in which only authorized entities can join the network, commonly use other consensus mechanisms, such as practical Byzantine fault tolerance [18]. Blockchain networks provide complete provenance of transactions, an immutable ledger, improved security and uptime, and decentralized smart contract software computation.

NFTs recently became one of the fastest-growing uses of blockchain [19]. NFT solutions offer ownership guarantees for every asset minted (i.e., added) to the network. The first official standardization of NFTs was in 2018 with ERC721, in which an Ethereum-based smart contract must keep track of a one-to-one association between a given owner and the

NFT, and each NFT type must have its separate smart contract. More recently, ERC1155 was introduced to extend the functionality and flexibility of ERC721. The new standard allows transferring tokens in batches, multiple token types per smart contract, semi-fungible token behavior, and safe reversible transfers [15], [20].

Although blockchain and NFTs pave the way for developing a data ownership platform, certain hurdles persist. For example, Ethereum nodes can only execute basic smart contracts due to their limited computation performance [21]. Additionally, the nature of stateful and transaction-based blockchain ledgers does not allow storing large-sized data. As a result, a fully decentralized solution must eventually utilize off-chain networks and interactions as part of secondary decentralized solutions [22]. However, the blockchain network must carefully manage and observe such transactions by binding them to the on-chain core using alternative trust models [23]. Otherwise, a flawed off-chain design will result in nodes blindly trusting each other or resolving to centralized entities, defeating the purpose of using blockchain technology.

In this paper, we propose a blockchain-based platform for managing data NFTs to enable ownership, time-bound sharing, and monetization. Our proposed solution is fully decentralized, does not rely on any trusted arbitrators, provides full provenance of actions, and utilizes a variety of trust models to validate off-chain transactions. Our contributions in this paper are summarized in the following:

- We propose an NFT-based platform for verifying and managing ownership of data assets with the provenance of ownership transfers.
- We present a data-sharing mechanism that enables users to select what subset of data to share, whom to share the data with, and the duration before the data sharing expires.
- We outline a monetization system to allow users to receive cryptocurrency-based payments based on how their data is accessed and used.
- We integrate IPFS, off-chain oracle nodes, distributed proxy reencryption (PRE), and decentralized applications (DApps) as part of a secondary off-chain network governed by a robust reputation system.
- We design a complete system architecture with a structured approach and detailed algorithms for all interactions among the blockchain network and entities. Additionally, we develop Ethereum smart contracts to prototype the proposed platform. All code developed is made public on GitHub.[1]
- We perform functional correctness and cost analysis of the proposed and developed systems. Furthermore, we discuss our proposed solution's feasibility, usability, security, and resilience to malicious attacks.

This paper is organized into five sections. Section II reviews the existing solutions for data ownership, sharing, and monetization. Section III presents our proposed architecture and approach details. Section IV discusses the implementation of our algorithms and smart contract functions, in addition to evaluating the implementation in terms of functionality and cost. Finally, we examine our approach in section V and recap our discoveries in section VI.

[1] https://github.com/AnonGitter20220510/nft-content-sharing

II. RELATED WORK

This section presents the existing solutions addressing storage and ownership, time-bound data sharing, and data monetization. Our exploration of the previous literature found no proposal that integrates the three aspects into a single system. Therefore, we discuss each component separately.

A. STORAGE AND OWNERSHIP

The current state of storage systems presents users with an abundance of commercial solutions, such as Google Drive, Microsoft OneDrive, and Dropbox. Although these systems do not offer proof of ownership, they provide users with the means to control who can access the data. However, common aspects of such solutions include their expensive pricing and limitations in feature sets for personal and small-scale use [24]. In addition, centralized solutions fail to defend against attacks and increasingly suffer in terms of availability [25], [26].

Open source solutions offer more flexible and independent categories of storage and ownership systems [27]. For example, open-source software (OSS) guarantees that the user can potentially read and even make changes to the running processes. However, setting up a cheap and reliable OSS data storage system can quickly become a tedious job for the ordinary user. One more problem with OSS solutions is that they expose personal devices to allow data sharing and interoperability across networks.

More recently, an exciting new type of storage system has emerged. Filecoin, Storj, and Arweave are decentralized storage solutions that build peer-to-peer networks and incentivize their nodes to keep the files stored [28], [29], [30]. Filecoin is currently the most popular decentralized storage solution. It uses IPFS as the basis of the storage network and relies on proof-of-storage to ensure incentives are paid to the deserving nodes. An interesting aspect of decentralized solutions is their low pricing, which can be as little as one-tenth the price of Amazon S3 Buckets [24]. However, since decentralized storage systems only provide the means to upload and retrieve data, they do not offer proof of data ownership.

Researchers have proposed blockchain-based ownership solutions for preserving data used in AI training and medical drugs [31], [32], overcoming vulnerabilities found in centralized solutions [33], such as single point of failure. However, the solutions implement non-standard proof of ownership mechanisms, so they lack interoperability and scrutiny.

NFTs, in essence, guard ownership of digital assets by maintaining standardized and indisputable transactions of minting or transferring the tokens. Researchers have studied the various types of digital assets that best suit the use of

NFTs, such as visual artworks, in-game purchases, or even unique arbitrary collectibles [19]. Other works have looked into minting patents and intellectual property as NFTs [34], as well as integrating the standardizations of common image formats such as JPEG with the ERC721 standards [35].

B. TIME-BOUND DATA SHARING

The introduction of blockchain has fueled research to look into rights-preserving data sharing and access control. Researchers from a wide range of industries are investigating solutions that enable owner-first sharing rights [36]. The transparent and trustless design offered by smart contracts has enabled users to know what and with whom their data is shared. For example, in the medical field and personal health records, [37] utilized blockchain and smart contracts to grant a patient-centered experience for managing health record documents. Moreover, the paper discussed opportunities for future systems to allow more restrictive access policies by implementing an expiration date for the shared data.

One approach to the data expiration problem is to set an expiration date for the access token granted by the owner, after which it would not be possible for the receiving party to proceed with the retrieval. This approach is only favorable for problems where the data in question is dynamically produced and streamed to the other party [38].

The research is advancing to a more confidential and time-bound sharing mechanism where the receiving party would not be able to duplicate or copy the data even after obtaining it. Popular techniques to protect against content duplication are implemented in confidential messaging apps like Snapchat. These techniques include using OS-level APIs to disable screen capturing, recording, and sharing of protected media, in addition to requiring the latest software updates with all security patches [39], [40]. To avoid these limitations on the client-side, researchers proposed alternative solutions that assure data deletion and enforce self-expiration, using techniques such as attribute-based encryption and revocation, TEE, threshold secret sharing, and frequently colliding hash tables [41], [42], [43]. The drawback of these techniques is having to reside at a centralized server that cannot be trusted. As far as decentralized solutions, they are limited to adding DRM databases to the blockchain and trusted execution environment (TEE) for a more reliable network and automated issuance [11], [44], which is insufficient to actively prevent users from duplicating the data.

C. DATA MONETIZATION

Traditional centralized monetization solutions claim to split the profit with the owners. However, these solutions suffer from several problems: 1) Inability to guarantee a fair share of the profits to the users due to relying on centralized technologies that hide the transactions. 2) No protection against non-repudiation attacks, where the service provider can claim it never accessed the user data [45]. 3) Relying on traditional payment and money transfer methods requires more processing time and fees, especially for international transfers [16]. 4) Weak protection against data manipulation, leading to a more time-consuming and challenging auditing process [46].

Researchers and commercial solutions have densely explored monetizing genomic data and incorporating blockchain into its process [18], [45]. Early suggestions incentivized patients to share their medical data with researchers using an automated smart contract. For instance, Nebula is a private blockchain platform for sharing genomic data that uses proprietary storage, ledger, and analytics solutions [16]. In addition to genomic data, researchers proposed letting patients sell their encrypted medical records by listing them on a blockchain-based marketplace with mechanisms to assure proof of delivery [47]. The blockchain-based marketplace proposals were also introduced to a broader range of industries, such as real-time IoT data streams and general end-user documents [48], [49].

The research in the monetization field has been limited to special and targeted use cases, such as paying small fees in return for medical information or paying royalties to obtain digital artworks. Additionally, the majority of the marketplace research does not support the proposed design. To the best of our knowledge, we are the first to propose and implement a secure, global, and generalized architecture for encrypted data monetization.

III. NFTs-BASED SOLUTION

In this section, we discuss our proposed approach by describing the different layers in the architecture and presenting the responsibilities of the system's entities and how they interact with it.

A. ARCHITECTURE COMPONENTS

Our proposed architecture comprises four general components: the end-user who interacts through the DApp, the blockchain smart contracts, the off-chain oracle nodes, and the decentralized storage. The types of interactions among the entities are depicted more clearly in Figure 2. Although our design is agnostic of any specific framework, we chose to present the solution with Ethereum and IPFS terminologies because of their common usage in the academic literature.

1) END-USER

An end-user in our system is an entity that interacts with the smart contracts, the oracle nodes, and the decentralized storage through the DApp. We categorize the end-user capabilities into three personas that are not mutually exclusive:

1) Producer: An original creator of data assets eventually owned by an owner persona. It can optionally have the ability to automatically upload assets (plaintext or encrypted) on behalf of a different end-user (owner) and mint the assets as NFTs for the owners.
2) Owner: The owner of the NFT assets, which could be either uploaded by the same end-user or a different one (producer). It can optionally delegate the minting process to be done by the producer. It has the exclusive right to grant sharing of data assets with consumers.

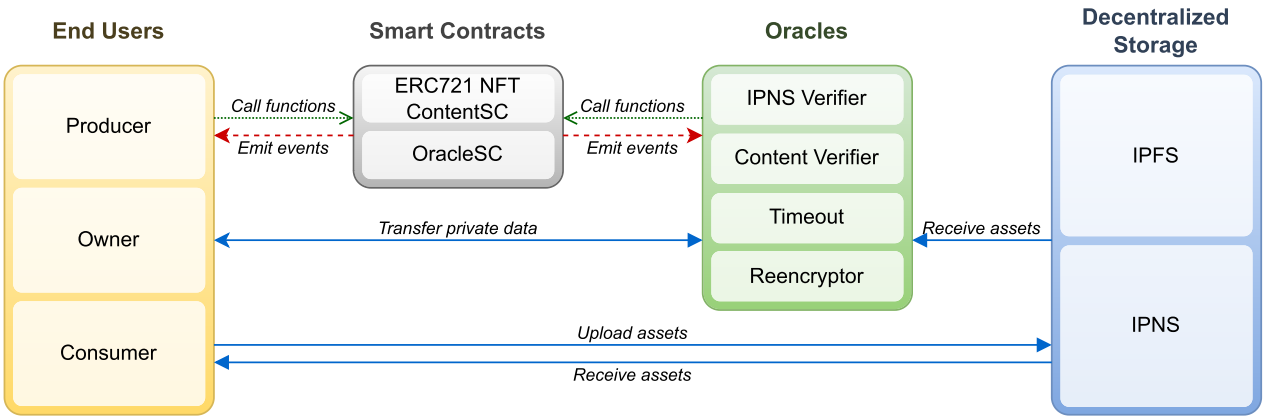

FIGURE 2. High-level perspective of architecture components and their types of interactions.

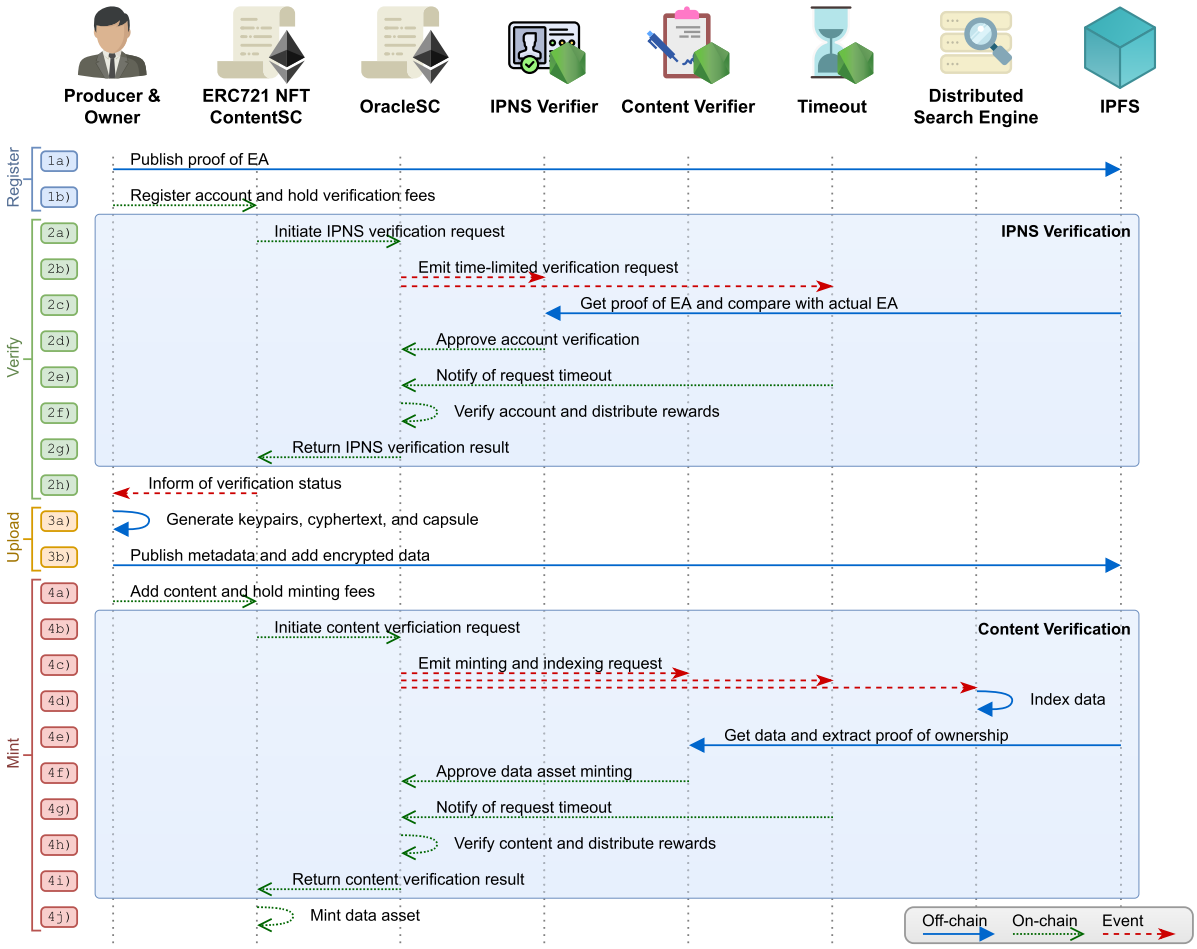

FIGURE 3. Sequence diagram illustrating the transactions of registration of users and minting NFTs.

3) Consumer: Pays to view or access NFTs-based data assets. It can perform search operations to find relevant data, then pay for obtaining the data without modifying the NFTs' ownership.

Considering that all personas require the end-user to interact with the decentralized storage, our system requires the end-user to have a verified registration linking the smart contract with their storage address. Our developed mechanism requires the user to publish a `verify.txt` file containing their EA on the InterPlanetary Name Service (IPNS), and the smart contracts can later verify the validity of the proof. IPNS is a service that lets you make addresses that point to IPFS content that can be read by humans and changed.

The DApp is where users can send requests to and receive updates from the smart contracts. From the perspective of the

network, it is the DApp that gets connected to the blockchain gateways (e.g., Infura) and nodes [50]. Therefore, the DApp must be built on top of a secure development framework to prevent unwanted actors from easily impersonating the end-users. Moreover, in order to combat attempts to leak content from the DApp, the app can incorporate mechanisms such as disabling all views if it is not connected to the network, preventing screen capturing of confidential content, performing encryption operations, and displaying data viewer modules using secure TEE sessions [39], [51].

2) SMART CONTRACT

Blockchain plays a pivotal role in our proposed system. The decentralized network stores ownership data on the ledger and executes transaction calls implemented in the smart contract. Our architecture uses the Ethereum blockchain framework for its significant adoption and public nature. Ethereum-based smart contracts are executed in the Ethereum virtual machine (EVM) and mined in exchange for a monetary reward paid in Ether cryptocurrency. Smart contract function fees are measured in gas units based on their logical and arithmetic operations, and gas is converted to Wei depending on the market gas price, where $1\,\text{wei} = 10^{-18}$ Ether. Even though Ethereum smart contracts do not encrypt the ledger, our system fully supports confidential content by allowing minting, transferring, sharing of, and searching for encrypted data.

Our design splits the blockchain functions into two smart contracts. The first smart contract is called **ContentSC** and it expands on the ERC721 NFT standard by incorporating additional unique attributes for each NFT, account-based access controls, and Ether transfer management. The second smart contract is called **OracleSC**, it organizes the off-chain oracles, can be used for on-chain and off-chain services, and maintains the reputations and participation of the oracle nodes.

3) OFF-CHAIN ORACLE

The Oracle nodes in our proposed architecture play a vital role in expanding the capabilities of the smart contract. Oracles have a variety of tasks that they can subscribe to; by doing so, they declare themselves capable of performing such tasks. To ensure oracles correctly perform their tasks, we incorporate trust models as a logical interface in **OracleSC**. The trust models functionality is three-fold: 1) Aggregate the responses of multiple oracles into a single accepted final result. 2) Assign a calculated reputation score for each participating oracle based on its performance. 3) Split the agreed-upon monetary incentive among participating oracles, considering their performance. Table 1 lists the types of oracle tasks and their corresponding trust model. Details of the oracle types and the trust model mechanisms are discussed later in this section.

The trust model ratings are combined into a reputation score using dynamic averaging as part of a reputation system that keeps track of the general performance of each task of each node. The reputation scores eliminate unsatisfactory nodes and calculate the trust model ratings. Rather than directly using the reputation scores, we incorporate Laplace's rule of succession to produce a probabilistic reputation score [52], [53]. The probabilistic score is capable of fairly comparing nodes with a high reputation score and few participations against ones with a slightly lower reputation score and many participations.

TABLE 1. Types of oracles and their corresponding trust model mechanism.

Oracle	Trust Model	Performance Factors
IPNS Verifier	Majority Voting	Response time, correctness
Content Verifier	Majority Voting	Response time, correctness
Timeout	First Responder	Response time
Reencryptor	Reputation Threshold	User rating

Interactions between end-users and oracle nodes are not supervised by the trust models but instead managed by other means of incentivizing correct actions, such as paying collateral. Additionally, oracles are prohibited by **OracleSC** from participating in multiple requests. Locking oracles keeps them from getting involved in a lot of requests when they don't want to or can't answer all of them.

4) DECENTRALIZED STORAGE

The decentralized storage component can be considered an unmanaged entity, albeit crucial to the system. The two required capabilities in a decentralized storage solution are the ability to add and publish content assets. Adding a content asset will result in a content-addressable URL and, therefore, an inability to update the asset on the same URL, whereas publishing an asset will result in a fixed URL that points to the latest version of it. These fundamental capabilities are offered in the IPFS network. However, one downside of IPFS is the lack of monetary incentives, and as a result, it lacks guarantees to store content for long periods. Filecoin solves that problem by embedding monetary incentives based on proof-of-storage on IPFS.

Uploading an asset to the network may optionally be accompanied by encrypting the asset data beforehand and injecting it with an identifier that can be used to trace back the content to its uploader or supposed owner without relying on blockchain to reveal the identity. As a result, regardless of the storage or blockchain networks, or if an asset gets minted, there will always be a claimed identity associated with uploaded data, which remains unverified if the minting does not carry out [54].

The content structure of the IPNS storage is not restricted to a fixed format, except for the `verify.txt` file, which must be at the root. Figure 7 captures an example of the IPNS content structure, and Figure 8 showcases an instance of an ERC721-compliant JSON metadata file. For any of the content assets published on IPNS, the user will get an IPNS path such as `IPNS/conference/meeting.mp3.json` and a content identifier (CID) to retrieve the asset from the

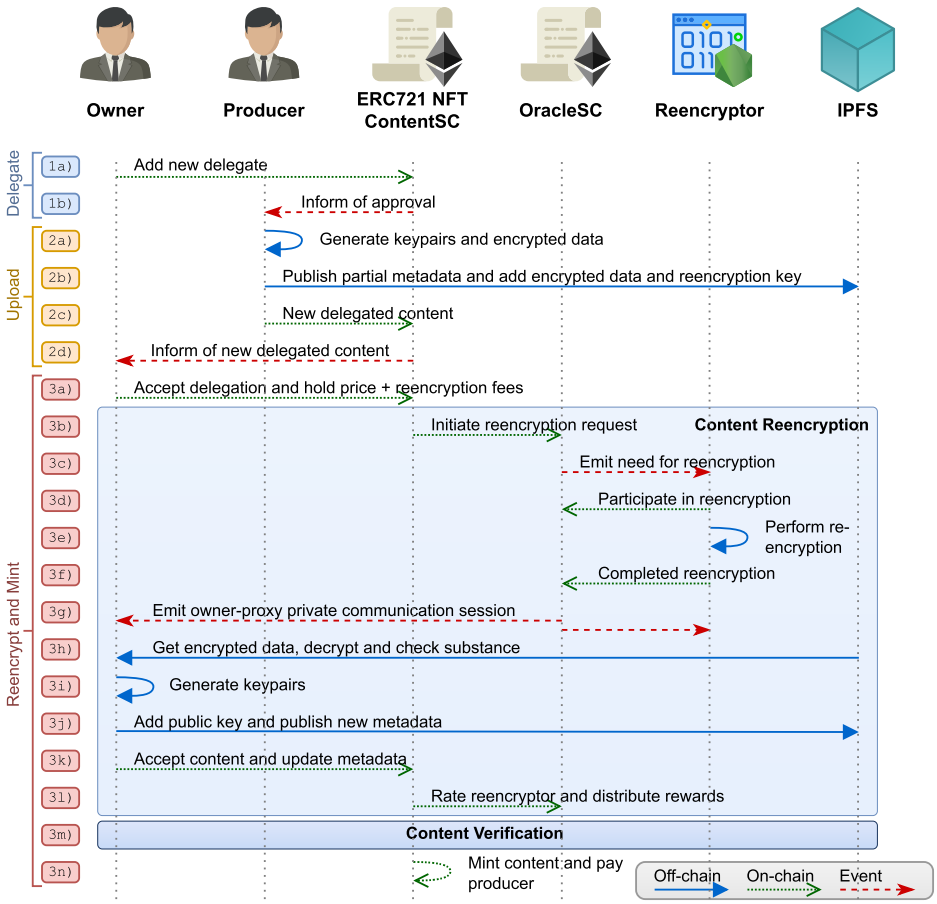

FIGURE 4. Sequence diagram illustrating the transactions of minting NFTs on behalf of the owner.

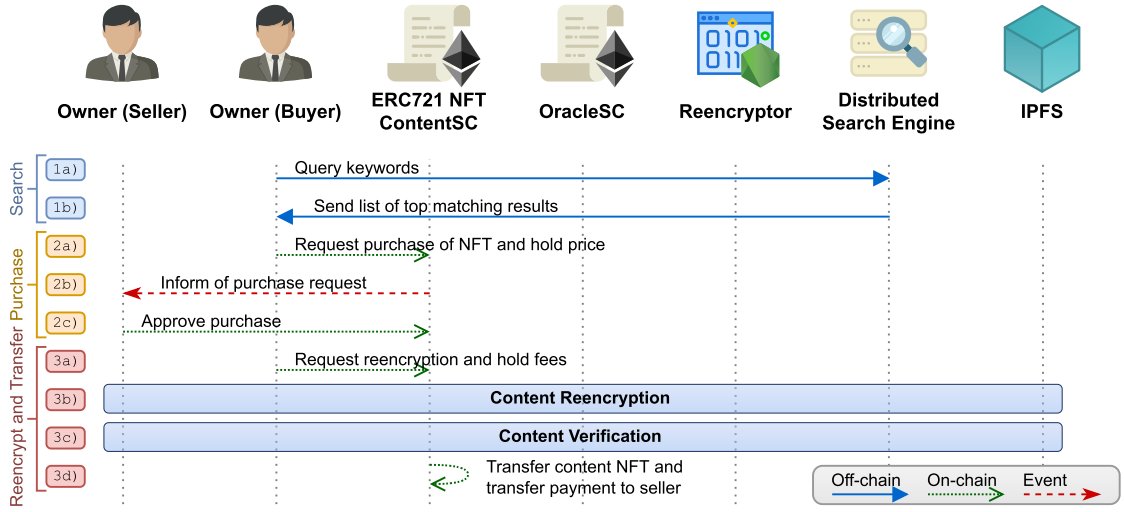

FIGURE 5. Sequence diagram depicting the transactions of purchasing NFTs.

IPFS network. Both the path and the CID of each content asset are stored alongside its NFT.

B. FUNCTIONS AND INTERACTIONS

To demonstrate the various functionalities of the proposed approach, we divide the interactions into scenarios. The scenarios do not build upon each other in the mentioned order for the system to behave correctly.

1) STORAGE AND OWNERSHIP

All entities that interact with the smart contracts must first register themselves as one of the predefined entity types, such

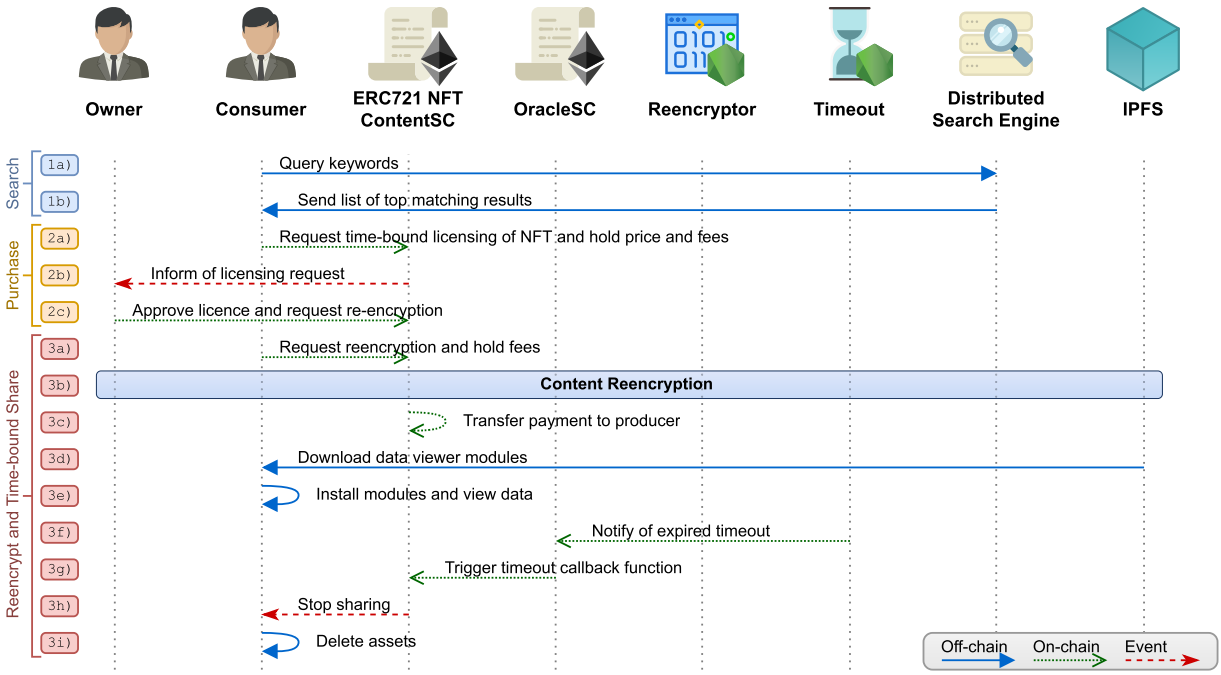

FIGURE 6. Sequence diagram illustrating the transactions of sharing content assets in a time-bound manner.

as an end-user or an oracle node. End-users must also verify their account registration by providing their IPNS address to **ContentSC**, which gets checked by IPNS verification oracles through **OracleSC**. One of the functionalities available to the end-user after their IPNS verification is to upload and mint new content assets. Figure 3 depicts the sequence diagram of a scenario where the end-user registers and verifies the account, then proceeds to upload and mint an asset. Since it is unsustainable for the owner to encrypt the content multiple times and share the encrypted content with the buyers or consumers, our mechanism uses a hybrid cryptosystem and PRE to share encrypted content without ever having the owner decrypt and encrypt it for each receiver.

1a - 1b) The end-user prepares a `verify.txt` file containing their EA, and publishes the file at the root of the IPNS address. Once the file is published, the end-user requests to register their account on **ContentSC**.

2a) **ContentSC** initiates a new verification request on **OracleSC** and transfers the verification fees to the latter smart contract.

2b - 2d) **OracleSC** emits a verification request that lasts for one hour, during which IPNS verifiers download the `verify.txt` file and compare its content against the end-user's EA, the responses are sent back to the smart contract.

2e - 2g) The timeout oracle notifies **OracleSC** to begin finalizing the request. algorithm 6 delineates how the smart contract computes a score upon each oracle's verification response, and algorithm 7 details how to update the reputation of oracles and award correct ones. The symbols used in the algorithms are defined in Table 2.

2h) The resulting verification result is emitted by **ContentSC** to inform the end-user of their updated status.

3a - 3b) The end-user uses hybrid cryptosystem to encrypt content asset d with symmetric key m and public data key p [55], and injects the encrypted content with EA a, to receive an injected file $I(E(d, m), a)$ and encapsulated key $E(m, p)$. Both of which are then added to IPFS alongside an ERC721-compliant JSON metadata file describing the uploaded data published to IPNS.

4a - 4b) The end-user submits the IPNS path and IPFS CID to **ContentSC**, which in return starts a new content verification request on **OracleSC**.

4c - 4i) In a similar fashion to the IPNS verification steps, the request is processed by **OracleSC**'s oracles and is then returned as a final result to **ContentSC**. The processing for content verification depends on finding the uploader's EA injected into the claimed content, and if so, the content gets approved for minting. Additionally, the smart contract emits an indexing request to the distributed search engine, so that end-users can query the assets later on.

4g) **ContentSC** receives the content verification result, which is assumed to be valid, and proceeds to mint the content asset as an original NFT.

2) DELEGATION OF CONTENT UPLOAD

The producer and the content owner are two separate entities in certain use cases. Instead of requiring the producer to

pay the minting fees, our approach lets the producer only upload the encrypted assets, and it is depicted in Figure 4. The uploaded assets are sent directly to the owner's address and re-encrypted so that only the owner can access them.

1a - 1b) The owner adds the producer as a delegate, and the smart contract informs the producer of the transaction.

2a - 2d) The producer uses hybrid cryptosystem to encrypt content d with symmetric key m and public data key p_P, and injects the encrypted content with the owner's EA a_O, resulting in injected content $I(E(d, m), a_O)$ and encapsulated key $E(m, p_P)$. The content and key are then added to IPFS, and the partial metadata is published to IPNS. Then the uploading details are submitted as a delegated content on **ContentSC**, which emits an event to inform the owner.
Additionally, the producer downloads the owner's public key p_O and combines it with their private key s_P to produce a reencryption key r_{PO}, which is then stored on IPFS without making the CID public.

3a - 3c) The owner accepts the content and transfers its price and reencryption fees to the smart contract. Upon receiving the payments, **ContentSC** initiates a reencryption request and sends the fees to **OracleSC**.

3d - 3f) A reencryptor with a satisfactory reputation score announces participation in the reencryption request, then proceeds with downloading the content key $E(m, p_P)$ and reencryption key r_{PO} and completes the reencryption to end up with $E(m, p_O)$.

3g - 3m) The reencryptor sends the content keys $E(m, p_O)$ to the owner over a private communication session, allowing the owner to decrypt and confirm the substance of the content to **ContentSC**, which passes the confirmation to **OracleSC** and initiates a new content verification request.

3n) **ContentSC** mints the content asset as an NFT of the owner, and sends the payment to the producer.

3) MARKETPLACE SEARCH AND PURCHASE

A different use case is for another owner to buy the content NFTs and fully access the material. The steps in Figure 5 assume the content is already uploaded, minted, and owned by the seller.

1a - 1b) The buyer queries the distributed search engine, which returns a list of most matching buyable content NFTs [51].

2a - 2b) The buyer requests through **ContentSC** to buy the NFT. The smart contract holds the offered price and transfers fees and emits an event to inform the seller of the request.

2c - 3a) The seller approves the purchase request, then the buyer completes the payment by transferring the reencryption and content verification fees to **ContentSC**, which sends them to **OracleSC**.

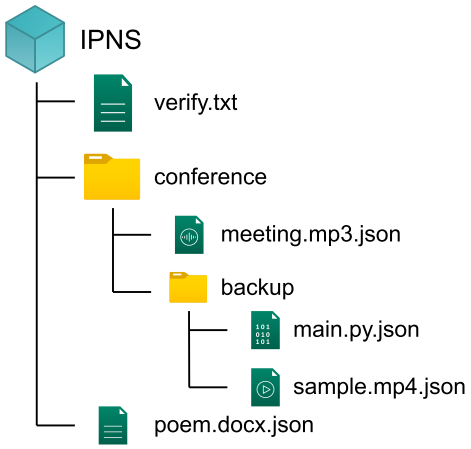

FIGURE 7. File structure of content published by the owner.

3b - 3d) **OracleSC** completes the content reencryption and verification requests. Once the content is validated, **ContentSC** transfers the content NFT from the seller to the buyer, and sends the payment to the seller.

4) TIME-BOUND DATA SHARING AND MONETIZATION

When an end-user only wants to access the material of encrypted content NFTs, our approach offers the means to request licensing for using the content. The owner shares the data with the consumer in this use case, similar to the purchasing scenario. However, since the NFT ownership is not transferred to the consumer, we propose a mechanism to allow access to the data in a time-bound manner. The time-bound restrictions are enforced by a data viewer DApp running in a TEE. Figure 6 presents the interactions for this use case.

1a - 3a) As in the purchasing use case, the consumer queries the distributed search engine and requests licensing for it. The owner approves and proceeds with the reencryption process.

3b - 3c) Upon successful reencryption determined by the consumer confirmation, **ContentSC** transfers the licensing price to the owner.

3d - 3e) The DApp determines the received content type and requests to download the required modules to read the content properly.

3f - 3i) Once the time-bound sharing expires, a timeout oracle notifies the DApp - via the smart contracts - to stop sharing; upon such request, the DApp deletes the content assets.

IV. IMPLEMENTATION AND EVALUATION DETAILS

We implement the necessary smart contract functions that the various types of entities can call to evaluate our proposed approach. This section explains how we chose to build a prototype of the system and put it into use.

meeting.mp3.json *ERC721 JSON Metadata*

```
{
  title: "Meeting Recordings Asset Metadata",
  type: "object",
  properties: {
    name: {
      type: "string",
      description: "Conference Meeting Recording"
    },
    description: ... ,
    dataPath: ... ,
    dataCapsulePath: ... ,
    tags: ... ,
    options: {
      type: "object",
      description: {
        encryptedData: true,
        injectedData: true,
        searchable: false,
      }
    }
  }
}
```

FIGURE 8. Example of an ERC721-compatible JSON metdata file.

A. DEVELOPMENT OF SMART CONTRACT ALGORITHMS

For the Ethereum-based smart contracts development, we used the Solidity language to write the contracts logic. **ContentSC** extends the NFTs standard implemented by OpenZeppelin with no direct modification to the interface, and **OracleSC** is implemented as a base contract that does not derive from other contracts.

All Ethereum accounts interacting with the smart contract must have a registration type, which prior to calling the `registerAs` function, would be initialized to the default `UNREGISTERED` state. The deployer of the smart contract is automatically set as the `DEPLOYER` registration type upon deploying the smart contract, whereas other accounts must set their registration type to either an `END_USER` or an oracle, more specifically, `IPNS_VERIFIER`, `CONTENT_VERIFIER`, `REENCRYPTOR`, or a `TIMEOUT`.

The smart contracts manage three categories of attributes:

- *NFT* attributes are solely accessible to **ContentSC** and they associate the content NFT identifiers with the uploader, the IPNS path, the IPFS CID, a purchaser, and their offered amount.
- *Entity* attributes are accessible to **ContentSC** to maintain `END_USER` and **OracleSC** to maintain oracles. An entity of the former type has an IPNS address, a list of accepted delegates, and a lock status for making purchases, whereas an entity of the latter type has a reputation score, a participations count, and a lock status for participating in new tasks.
- *Action* attributes are only accessible to **OracleSC**. One type is for verification, where each verification action has a timeout, a reward amount, a positive score sum, a negative score sum, participating oracles, and their responses and scores; whereas, the other type is for reencryption actions, which are associated with a reencryptor's reputation threshold, a reward amount, a completion state, and a chosen reencryptor.

Algorithm 1 `SetIPNS`: Sets the IPNS Address. Callable by Owners and Producers on **ContentSC**

Data: `ipns`
1 Require `ipns` is not set;
2 Require oracle rewards are paid;
3 Store `ipns` as under verification;
4 Call `StartVerify` function, callback `SetIPNSState` on completion;

Algorithm 2 `AddContentFor`: Adds New Content. Callable by Owners or Producers on **ContentSC**

Data: `benef`, `path`, `cid`, `price`
1 Require caller and `benef` have valid IPNSs;
2 Generate new `assetId`;
3 Associate `assetId` with caller, `benef`, `path`, `cid` and `price`;
4 Emit an event to update `benef`;

Algorithm 3 `AcceptAsset`: Accepts Successful Reencryption. Callable by Owners on **ContentSC**

Data: `assetId`, `path`, `cid`
1 Require caller is correct beneficiary;
2 Require oracle rewards are paid;
3 Update `assetId`'s `path` and `cid`;
4 Send Ether from **ContentSC** to delegate;
5 Call `RateReencrypt` function;
6 Call `StartVerify` function, callback `MintAsset` on completion;

A primary step for `END_USER` registration is to specify their IPNS. First, the user calls algorithm 1 which updates the IPNS verification attributes, resets the verification state, holds the verification fees, and initiates a verification task with a time limit of 1 hour. During this period, IPNS verifiers submit their responses to algorithm 6, which updates the attributes and locks the verifier from any further participation. Once the task timeout expires, the verifiers can call algorithm 7 to unlock their accounts, update their reputations, and receive their rewards, and a timeout calls `ReturnVerify` function to submit the new verification state to **ContentSC**, which triggers a callback function `SetIPNSState`.

Uploading a new content asset and minting it begins with the user calling algorithm 2, which updates the content attributes and calls `StartVerify` to initiate a new content verification task with a time limit of 1 hour. Content verifiers can call algorithm 6 during the hour, which updates the verification attributes and locks the oracles. After the timeout, the verifiers and timeout call algorithm 7 and `ReturnVerify`, respectively. Upon receiving the verification, **ContentSC** calls `MintAsset` to mint the content.

Algorithm 4 `FinalizeSharing`: Confirms Receiving Shared Content. Callable by Consumers on **ContentSC**

Data: `assetId`, `path`, `cid`
Require correct consumer;
if purchase **then**
    Update `assetId`'s `path` and `cid`;
    Call `StartVerify` function, callback `TransferAsset` on completion;
end
Send Ether from **ContentSC** to seller;

Algorithm 5 `TransferAsset`: Transfers Valid Content. Callable by **OracleSC** on **ContentSC**

Data: `assetId`, `state`
Update `assetId` state to `state`;
Emit an event to update the consumer;
if `state` = `VALID` **then**
    Call **ERC721**'s `safeTransfer` function;
end

TABLE 2. Symbols definitions used in algorithm 6 and algorithm 7.

Symbol	Definition	Type
t_{current}	Current time	Implicit value
$r_{\max}$	Maximum reputation score	Global constant
r	Oracle reputation	Per-oracle global attribute
p	Oracle participations count	Per-oracle global attribute
f	Offered reward	Per-task global attribute
t_{timeout}	Verification timeout	Per-task global attribute
Σs^+	Sum of positive scores	Per-task global attribute
Σs^-	Sum of negative scores	Per-task global attribute
$\Sigma s_{\max}$	Maximum of scores	Local computed variable
s	Score of oracle	Local computed variable
$\hat{s}$	Normalized score of oracle	Local computed variable
Δr	Participation reputation	Local computed variable
a	Reward assigned to oracle	Local computed variable

After the global attributes are updated, the score of the participating oracle is computed based on two factors: the oracle's reputation and response time. The higher the reputation and the lower the response time, the higher the score. Additionally, the reputation of the oracle is not directly represented in the score, but rather after using Laplace's rule of succession formula, as given by Equation 1. The summation attributes are changed based on whether the oracle response was positive or negative, and the oracle index is returned.

$$r_{\text{rep}} = \frac{rp + r_{\max}}{p + 2} \tag{1}$$

The function described in algorithm 7 is called for reputation updates and reward distribution. The function combines and normalizes the score and correctness of the participating oracles into a single continuous and non-negative value, such that the lowest correct score is higher than the highest incorrect score, and the lowest score is equal to or greater than zero. The score is then converted to a change in reputation by mapping it to the reputation range, which in our implementation is $[0, 2^{16})$. Solidity language does not offer native support for floating points. Therefore, our code uses the larger range to accommodate the reputations. The oracle's reputation and participation count are updated based on a simple dynamic averaging formula, and the award is calculated based on the ratio of the oracle's score to the summation of correct scores.

For delegated uploading and minting, the owner needs to approve the delegate by calling `AddDelegate`, which gives privileges for another user to reencrypt content from their keys to the owner's keys. Then, the producer and the owner call algorithm 2 and `PayDelegate` respectively, which automatically invokes the `StartReencrypt` function in **OracleSC**. The smart contract at this point is waiting for a reencryptor oracle with a satisfactory reputation score to handle the processing, which, if available, should call algorithm 8, and upon completing the reencryption process, the oracle calls algorithm 9 to unlock their account and assign an initial score based on their response time. Once the reencrypted content is uploaded and verified by the content verifiers, or once the owner confirms the delivery of the content asset by calling algorithm 3, the smart contract transfers the reward to the reencryptor, updates their reputation, and calls back `MintAsset` to mint the asset for the owner.

Purchasing and licensing content takes a similar route to delegated upload. The interested user makes a purchase or a licensing request by calling `RequestSharing`, which locks the entity from making additional requests, and locks the content from taking other purchases. If the owner approves, they call `AcceptSharing`. After paying for the content with `PayForSharing`, `StartReencrypt` is triggered, making **OracleSC** wait for a satisfactory reencryptor to complete, calling algorithm 8 and algorithm 9. Once the delivery to the new owner or the consumer is confirmed by algorithm 4, the sharing process is considered complete. If the sharing is a purchase, then the buyer can update the metadata, which calls `StartVerify` and ends with triggering algorithm 5 if the content is valid. If the sharing is a time-bound licensing, the timeout oracle notifies the smart contract when the expiration is over, which forces the TEE-based data viewer DApp to drop the asset data.

B. DEPLOYMENT AND TESTING OF SOLIDITY CODE

We used the Truffle suite[2] as a development framework and deployment environment. Our code was written using Solidity language version 0.8.0 and compiled using the Solc compiler[3] of the same version, with optimization settings enabled at 200 runs. Truffle also includes Ganache,[4] a local Ethereum testing network (testnet), used to deploy the compiled contracts. Lastly, Truffle tests, which were built on the Mocha JavaScript test framework,[5] were used to run asynchronous and automated assertion tests. These tests could

[2]https://trufflesuite.com/truffle/.
[3]https://soliditylang.org/.
[4]https://trufflesuite.com/ganache/.
[5]https://mochajs.org/.

Algorithm 6 `RespondVerify`: Receives Verification Response. Callable by Verifiers on **OracleSC**

Data: `verifyId`, `response`

Require request timeout not reached;
Require verifier is not locked;
Lock the verifier;
Calculate `score` as $s \leftarrow \frac{rp+r_{\max}}{p+2}(t_{\text{timeout}} - t_{\text{current}})$;
Store verifier, `response` and `score`;
if `response`=`VALID` **then**
  Calculate $\Sigma s^{+} \leftarrow \Sigma s^{+} + s$
else
  Calculate $\Sigma s^{-} \leftarrow \Sigma s^{-} + s$
end

Algorithm 7 `FinalizeVerify`: Receives Rewards. Callable by Verifiers on **OracleSC**

Data: `verifyId`

Require request timeout reached;
Require verifier participated and did not finalize;
Unlock the verifier;
Calculate `maxScore` as $\Sigma s_{\max} = \max(\Sigma s^{+}, \Sigma s^{-})$;
if `response` does not match majority **then**
  Normalize `score` as $s = s + \Sigma s_{\max}$;
end
Calculate $\Delta r \leftarrow \frac{\hat{s}r_{\max}}{2\max(\Sigma s^{+}, \Sigma s^{-})}$;
Update reputation as $r \leftarrow \frac{rp+\Delta r}{p+1}$;
Update participations as $p \leftarrow p + 1$;
if `response` matches majority **then**
  Calculate reward as $a \leftarrow \frac{fS_i}{\max(\Sigma s^{+}, \Sigma s^{-})}$;
  Send reward Ether from `OracleSC` to verifier;
end

go back in time to make sure that time-sensitive functions worked correctly.

1) ASSERTION TESTING

Our testing path follows the interactions among entities found in the sequence diagrams, by which we ensure all the important functions of the smart contract execute as expected. The testing path incorporates five scenarios, 15 assertion tests, and 91 transactions made to the **ContentSC** and **OracleSC**. The tests use 17 accounts as detailed in Table 3.

1) **Registration**: OracleAdmin and ContentAdmin deploy the **OracleSC** and **ContentSC**, respectively, allowing the remaining accounts to register in the appropriate smart contract based on their roles. Users call the `Register` function with no inputs, whereas oracles call the `RegisterAs` function and declare the desired role by specifying the input value corresponding to their role. For example, the input value for a verifier is 2, for a timeout is 3, and for a reencryptor is 4. The rest of the functions have strict access controls based on roles, so they can't be used by accounts that aren't supposed to.

Algorithm 8 `ParticipateReencrypt`: Receives Reencryption Response. Callable by Reencryptors on **OracleSC**

Data: `reencryptId`

Require `reencryptId` is available;
Require reencryptor reputation is satisfactory based on Equation 1;
Lock reencryptor and mark request as unavailable;

Algorithm 9 `DoneReencrypt`: Completes Reencryption Phase. Callable by Reencryptors on **OracleSC**

Data: `reencryptId`

Require caller matches `reencryptId` reencryptor;
Unlock reencryptor;
Establish `sessionId` as hash of reencryptor, receiver and `reencryptId`;
Emit `sessionId` to reencryptor and receiver;

TABLE 3. Types of Ethereum accounts used in functional testing.

Type	Account
Admins	OracleAdmin ContentAdmin
Users	Owner Producer Consumer
Oracles	Verifier (6 oracles) Timeout (3 oracles) Reencryptor (3 oracles)

2) **Minting**: Users intending to add content must have a valid IPNS address. Therefore, the owner and producer call the `SetIPNS` function and provide the IPNS addresses 0×123 and 0×234, along with 0.15 Ether and 0.05 Ether as rewards for oracles to verify their addresses for them. The verifiers call `RespondVerify` within an hour and set the inputs to the verification request identifier of type uint256 (256-bit unsigned integer) and validation response of type Boolean. After an hour elapses, the oracles call `FinalizeVerify` and specify the verification request identifier to receive their rewards. The first timeout oracle then calls `ReturnVerify` with the request identifier as an input to update the states of the IPNS addresses. Our test varies the responses of the verifier oracles to confirm the robustness of the approach. Verifiers and timeout oracles who did their jobs right were rewarded with more reputation and money, while those who got it wrong lost reputation and didn't get back the transaction fees they paid.
The owner proceeds with adding a content asset by calling `AddContentFor` and providing the inputs starting with own Ethereum address as beneficiary,

`img1.png` as IPNS path and `baf123` as IPFS CID, along with 0.3 Ether and 0.1 Ether as rewards for the oracles. The verifiers respond to the request with varying correctness (3 invalidations, and 3 validations) with the `RespondVerify` function and then call `FinalizeVerify` function while providing the asset identifier of type uint256 as a request identifier and their response index of type uint256. As a final step, a timeout oracle updates the content validity by calling `ReturnVerify` function and specify the asset identifier. Our test concludes that the content is valid, which leads to minting the content as a new ERC721 NFT, as Figure 9 shows.

3) **Delegated Minting**: The owner adds a producer as an accepted delegate by calling `AddDelegate` and specifying the Ethereum address of the producer. The producer then calls `AddContentFor` and supplies the owner address as beneficiary, `doc2_1.pdf` as the path, $0 \times 987\_1$ as CID, and the price in Ether, which leads to creating a new content asset intended to be registered under the owner's account. Considering the owner did not grant full delegation privileges to the producer, the function `PayDelegate` with inputs specifying the new asset identifier, and the reencryption rewards, must be called by the owner, which also initiates the reencryption process. In our testing the owner invokes the function with asset identifier of 2, and rewards of 0.2 Ether for the reencryptors.
The **OracleSC** waits for a satisfactory reencryptor to call `ParticipateReencrypt` with request identifier of 2 as input. The reencryptor then calls `DoneReencrypt` with the same input, before initiating a new private communication session between the owner and the reencryptor using the identifier in Figure 10. The owner then calls `AcceptAsset` and specifies inputs 2 for asset identifier, `doc2_0.pdf` as new path, $0 \times 987\_0$ as new CID, and rewards of 0.2 Ether and 0.15 Ether to the verifiers. This transaction leads to initiating a content verification process that ends up validating the asset and minting it as a new NFT, with inputs matching those in the first verification.
4) **Purchase**: The third user, as a buyer and an owner, calls `RequestSharing` with asset identifier 1 and payment of 3 Ether to purchase the first content NFT from the first owner (seller). The seller responds with acceptance by calling `AcceptSharing` with input 1 for the identifier, then the buyer initiates the reencryption and content verification process by calling `PayForSharing` with the same input and payment of 0.2 Ether as rewards. After successful reencryption, the buyer calls `FinalizeSharing` with inputs 1 as asset identifier, `img1_2.png` as new path, $0 \times 123\_2$ as new CID, and rewards of 0.2 Ether and 0.15 Ether for oracles to verify the new addresses. The aggregated response of the verifiers marks the bought asset as valid, resulting in transferring the NFT from the original owner to the buyer.
5) **License**: Similar to the purchase test case, the third user, as a consumer, calls `RequestSharing` with input 2 and payment 0.7 Ether to request a license of the second content asset from its owner. The owner accepts the request with the `AcceptSharing` function given input 2, which emits the event depicted in Figure 11. Finally, the consumer calls `PayForSharing`, and provides inputs 2 for the asset identifier, and payment of 0.2 Ether for the reencrypters.

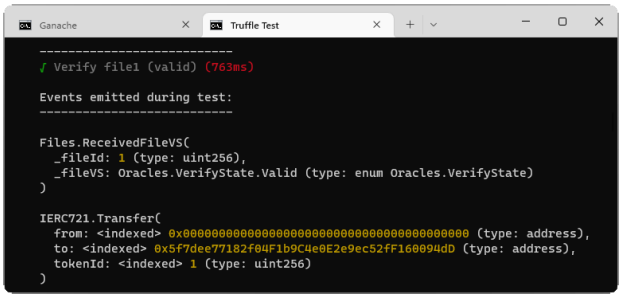

FIGURE 9. Emitted event to announce minting a new content NFT.

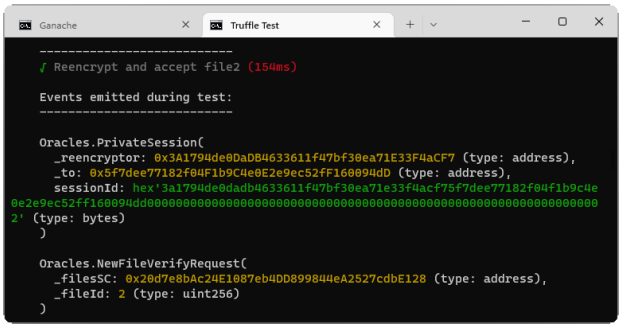

FIGURE 10. Emitted event to establish private communication session.

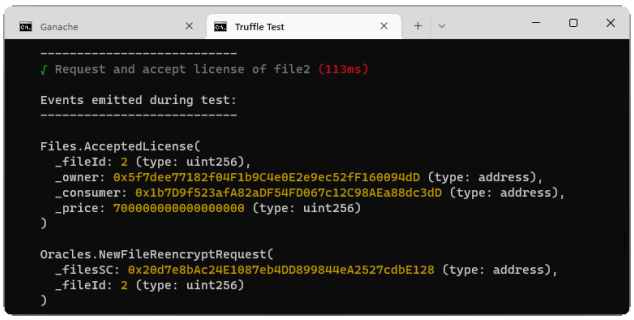

FIGURE 11. Emitted event to confirm acceptance of license request.

2) COST ANALYSIS

The testing of our implementation extends to measuring the average cost of calling the various functions in the smart contracts. The cost analysis is based on the inputs and monetary rewards provided in the assertion testing. Table 4 shows the cost of deploying and calling functions of the **OracleSC**. The deployment of the contract is the most expensive operation, compared to the registration functions. The costs of the remaining oracle functions should indicate the corresponding

TABLE 4. Transaction costs in gas and USD for **OracleSC** functions.

Caller	Function Name	Cost [Gas]	Ethereum Cost [USD]	Polygon Cost [USD]
Oracles admin	Deploy **OracleSC**	2,161,880	259.43	0.102
Oracles admin	`RegisterContentSC`	46,218	5.55	0.002
Oracle	`RegisterAs`	66,342	7.96	0.003
Verifier	`RespondVerify`	139,377	16.73	0.007
Verifier	`FinalizeVerify`	77,930	9.35	0.003
Timeout	`ReturnVerify`	76,533	9.18	0.006
Reencryptor	`ParticipateReencrypt`	68,609	8.23	0.003
Reencryptor	`DoneReencrypt`	32,156	3.86	0.002

TABLE 5. Transaction costs in gas and USD for **ContentSC** functions.

Caller	Function Name	Cost [Gas]	Ethereum Cost [USD]	Polygon Cost [USD]
Content admin	Deploy **ContentSC**	2,980,717	357.69	0.140
Content admin	`RegisterOracleSC`	46,258	5.55	0.002
All Users	`Register`	43,696	5.24	0.002
All Users	`SetIPNS`	196,801	23.62	0.009
Owner	`AddDelegate`	46,294	5.56	0.002
Producer	`AddContentFor`	205,468	24.65	0.010
Owner	`PayDelegate`	161,989	19.44	0.008
Owner	`AcceptAsset + Mint`	227,257	27.27	0.011
Consumer	`RequestSharing`	75,148	9.02	0.004
Owner	`AcceptSharing`	52,138	6.26	0.002
Consumer	`PayForSharing`	150,370	18.04	0.007
Consumer	`FinalizeSharing`	33,141	3.98	0.002
Consumer	`FinalizeSharing + Transfer`	187,166	22.46	0.009

minimum reward values. For example, if the user wishes to verify their IPNS address using the collective response of 5 oracles, they must pay in rewards at least $(16.73+9.35)\times 5+9.18 = 139.58$ USD only to cover the oracle's transactions.

Functions in the **ContentSC** are generally more expensive than the **OracleSC** functions due to the full implementation of ERC721 inside the contract. Table 5 depicts all the costs for the publicly callable functions in the **ContentSC**. Besides the oracle rewards, adding a content asset for the first time after registration costs the owner $23.62 + 31.99 = 55.61$ USD. Delegated minting and buying an asset cost about the same, but licensing is much cheaper.

Our cost analysis computes the costs in USD by converting the transaction cost from gas to Gwei, based on the average gas price of 40 Gwei, and the Ether price of 3,000 USD.[6] Considering the increasing gas and Ether prices, we also compute the costs of the functions for deploying the smart contracts on the Polygon network, a proof-of-stake layer two blockchain. Our calculations use a gas price of 30 Gwei and the MATIC price of 1.57 USD.[7] MATIC is the cryptocurrency used in the Polygon network.

An important finding in our testing is the constant execution time and cost as the inputs and the number of transactions increase. We optimized our algorithms to have a time complexity of $O(1)$, which is apparent considering we have no iterative behavior in the design. The efficiency of the algorithms makes our solution highly scalable, as the number of interactions scales linearly with the growth of users, assets, and oracles.

TABLE 6. Maximum throughput of actions on Ethereum and Polygon networks.

Action	Total Cost [Gas]	Ethereum Throughput	Polygon Throughput
Register	55,019	41.94	272.63
Set IPNS	1,369,657	1.68	10.95
Add content	1,439,460	1.60	10.42
Add delegate	46,294	49.85	324.02
Add delegated content	1,729,130	1.33	8.67
Purchase content	1,700,465	1.36	8.82
License content	527,568	4.37	28.43

3) THROUGHPUT AND LATENCY ESTIMATION

The main factors that affect the throughput and latency of the implemented system are the deployment network and the number of transactions per task. At the time of writing, the mainnet Ethereum and Polygon networks set a block size limit of 30 million, whereas their block time durations are 13 seconds and 2 seconds, respectively.[8] Therefore, assuming the network is not congested and the entities pay fair fees to the miners, a transaction's latency averages half the block time duration. As for the number of transactions (txn) per task, they vary depending on the action and the desired number of verifiers. Table 6 highlights the throughput as actions per second, and assumes 5 verifications whenever required.

[6]Etherscan. As of 24th of March, 2022.
[7]Polygonscan. As of 24th of March, 2022.
[8]Etherscan Statistics and Polygonscan Statistics. As of 7th of July, 2022.

V. DISCUSSIONS

Herein, we discuss the assessment of our proposed approach, the strengths and weaknesses concerning security performance, and the ability to generalize the solution.

A. SECURITY ANALYSIS

The goal of this security evaluation is to make sure that the different parts of the proposed architecture and implementation can withstand relevant cyber attacks.

- **Availability**: Blockchain is the core technology in our architecture, acting as the core that other components attach to. As a result, the system will not suffer from downtime due to distributed denial-of-service (DDoS) attacks. Furthermore, other networks such as IPFS are also decentralized, and with the added layer of Filecoin as a mechanism to ensure storage of content, users of our proposed system can rest assured that their uploaded assets and their NFTs are accessible at any time. On the other hand, it might be possible for some transactions to stall due to having no oracle responding to the user's requests. However, our approach provides monetary incentives for oracles to respond correctly, minimizing such possibilities.
- **Compliance**: Monetization of data is a significant aspect of our paper, and therefore, we ensure payers cannot maliciously withdraw from proceeding with paying the seller or the oracles by requiring holding the summation of fees whenever the user is participating in a payable interaction. If the transaction fails, the smart contract can be trusted to distribute the money and rewards or even return them to the buyer or consumer.Time-bound data sharing is another important part of this paper. To do this, we use TEE in the data viewer DApp and other mechanisms that enforce requirements, such as requiring the app to be connected to the blockchain network.
- **Data Privacy**: A combination of factors we incorporate in our approach leads to the elimination of all significant risks of privacy invasion. For example, all keys are generated by the users and not a central server, and therefore, as long as users keep their private keys out of public access, there would be no third party with unannounced access to the data. Additionally, atomic reencryption ensures that even when content is shared with a new owner or a consumer, the reencryptor oracles can never read the plaintext data. One issue that may arise is the current state of TEE security, because if content is leaked from the consumer's DApp, it will be possible to decrypt it or even claim ownership as a new NFT.
- **Authenticity and Integrity**: Our approach and implementation integrate a robust access control that allows one EA holder to register as one type of account, and all public functions in the smart contract require that account to be called by the proper account registration type. This is done in addition to further requirements to ensure unintended entities do not modify ledger information. Moreover, unlike traditional NFT solutions that place blind trust in the user's claims of ownership, our approach requires all users who wish to mint NFTs to verify their IPNS address and verify that they (or their delegates) uploaded the content onto the IPFS network. These requirements minimize attacks of stealing content and claiming ownership over it. Finally, it is worth mentioning that although the uploader of the content can be traced back to their EA, the substance is not verified and accepted as claimed by the user. This is where the manual rating reputation system comes into play, since upon a successful content purchase or licensing; the receiver can submit a rating for the original owner and the NFTs.

Our security evaluation extends to quantitatively investigating and assessing the vulnerabilities hidden in the implementation. Specifically, we utilize the Slither smart contract analysis framework [56] to audit the developed Solidity smart contracts: **ContentSC** and **OracleSC**. Our test uses Slither version 0.8.2, and does not exclude any vulnerability detector from running. The tool resulted in unmasking 152 vulnerabilities, 136 of which are related to formatting and optimization improvements. Among the remaining 16 vulnerabilities, we manually investigated the ones marked as 'High' and 'Medium' impact, to realize they are false positives. However, we noticed 10 'Low' impact vulnerabilities have been overlooked, which can be addressed in security patches to the smart contracts.

B. GENERALIZATION

Our paper intends to maintain a generalizable approach that can be taken advantage of in various systems. A wide range of industries can adopt the proposed approach where encrypted data ownership is essential. For example, in the medical field, the hospital is considered the producer of the data and can provide the encrypted electronic health records of the patient (the owner) described in the delegated upload and minting process. Additionally, the patient may allow other parties, such as a medical research institution, to have license-based access to the encrypted data without sharing the encryption key with the institution and guarantees never to exceed the time-bound rules set for the shared content. A different scenario can be embodied in the digital art medium, where artists can mint their work in the form of encrypted content (image, video, or 3D model) and let other collectors propose a price. What sets our approach apart from existing solutions is the automation of the data transfer process while maintaining complete confidentiality when the data is in transit.

The modules we made for our proposed architecture could be used in other systems as well, such as: 1) Hybrid cryptosystem reencryption and a fully decentralized reputation system. 2) Mechanisms to upload encrypted content once and share it multiple times. 3) Trace content-addressable assets back to their original uploader even on networks such as IPFS. Our

TABLE 7. Comparison of our proposed solution with the state-of-the-art solutions C: creator, D: delegate, NS: non-standard, L: license, O: ownership.

	[32]	[37]	[47]	[33]	[16]	[49]	Our Solution
Decentralized storage	No	No	No	Yes	Yes	Yes	Yes
Large data support	No	Yes	Yes	No	No	Yes	Yes
Data provider	C	C+D	C	C	C	C	C+D
Proof of ownership	NS	NS	-	NS	NS	NS	NFT
Encrypted data	Yes	No	No	No	Yes	Yes	Yes
Offloaded processing	No	No	No	No	Yes	No	Yes
Access control expiration	No	No	No	No	No	Yes	Yes
Data self-destruction	No	No	No	No	No	No	Yes
Monetization model	-	-	L	O	L	L	O+L
Decentralized payment	No	No	Yes	Yes	Yes	Yes	Yes

proposed modules and mechanisms can be valuable in distant applications like confidential data sharing protocols and e-commerce platforms.

C. COMPARISON WITH PRIOR STUDIES

We compare our solution with the existing solutions. We choose cutting-edge works that propose and implement decentralized solutions for the comparison. Table 7 shows how our proposed solution compares against the referenced literature. The comparison covers the three main components: data ownership; time-bound data sharing; and data monetization. In line with our preliminary findings, this comparison concludes that none of the previous works manages to integrate the important aspects of a data ownership, sharing, or monetization system.

Our solution accomplishes the data ownership component by utilizing IPFS for decentralized storage, adopting a hybrid cryptosystem for large data support, allowing creators and delegates to upload data, and implementing the ERC-721 standard to issue NFTs. As for time-bound data sharing, this proposal encrypts the data at rest and in transit, offloads the sharing process from the end-user, forces strict access control policies with timeouts, and ensures data self-destruction using TEE. Finally, we provide the means for users to be paid in cryptocurrencies under two monetization models.

VI. CONCLUSION

In this paper, we have leveraged NFTs to let users claim ownership over their data, share the data with others without compromising digital rights, and earn money upon granting access rights to the data. Our proposed architecture comprises four building blocks: the end-users with the ability to produce, own, and consume; oracle nodes with verification and reencryption capabilities; distributed storage for permanent storage of large content data; and the NFTs-compliant smart contracts as a primary hub to call transactions and store verified information. We investigated various use case scenarios of our proposed approach and summed them into four typical sequences of actions, such as uploading and minting content assets, delegated minting of assets, purchase of content NFTs, and licensing the content for a limited period of time. We designed algorithms for decentralized reputation systems that can self-assess the performance of oracles and update their reputations with mechanisms to prevent skewed results because of a low number of participants. The core functions of the decentralized system were developed using the Solidity programming language and deployed on an Ethereum-based testnet for extensive testing that concluded in a fully functional system with efficient gas costs and a lack of common vulnerabilities. Lastly, we discussed how safe our system is against attacks and how much our approach can be used in other areas, such as medicine and digital art.

REFERENCES

[1] D. R.-J. G.-J. Rydning, J. Reinsel, and J. Gantz, "The digitization of the world from edge to core," *Framingham, Int. Data Corp.*, vol. 16, pp. 1–28, Nov. 2018.

[2] (Apr. 2022). *Huge Cloud Market Still Growing at 34% Per Year; Amazon, Microsoft & Google Now Account for 65% of the Total*. [Online]. Available: https://www.srgresearch.com/articles/huge-cloud-market-is-still-growing-at-34-per-year-amazon-microsoft-and-google-now-account-for-65-of-all-cloud-revenues

[3] A. Tilley, "Amazon, microsoft, Google strengthen grip on cloud," *Wall Street J.*, [Online]. Available: https://www.wsj.com/articles/amazon-microsoft-google-strengthen-grip-on-cloud-11657018980

[4] H. Tabrizchi and M. K. Rafsanjani, "A survey on security challenges in cloud computing: Issues, threats, and solutions," *J. Supercomput.*, vol. 76, no. 12, pp. 9493–9532, 2020, doi: 10.1007/s11227-020-03213-1.

[5] A. Singh and K. Chatterjee, "Cloud security issues and challenges: A survey," *J. Netw. Comput. Appl.*, vol. 79, pp. 88–115, Feb. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804516302983

[6] S. Basu, A. Bardhan, K. Gupta, P. Saha, M. Pal, M. Bose, K. Basu, S. Chaudhury, and P. Sarkar, "Cloud computing security challenges & solutions—A survey," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2018, pp. 347–356.

[7] P. Kumar, G. Shrivastava, and P. Tanwar, *Ethereum Technology: Application and Benefits of Decentralization* (Forensic Investigations and Risk Management in Mobile and Wireless Communications). Hershey, PA, USA: IGI Global, 2020, pp. 242–256. [Online]. Available: https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-5225-9554-0.ch010

[8] K. Yu, L. Tan, M. Aloqaily, H. Yang, and Y. Jararweh, "Blockchain-enhanced data sharing with traceable and direct revocation in IIoT," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7669–7678, Nov. 2021.

[9] J. Al-Jaroodi and N. Mohamed, "Blockchain in industries: A survey," *IEEE Access*, vol. 7, pp. 36500–36515, 2019.

[10] S. Pavithra, S. Ramya, and S. Prathibha, "A survey on cloud security issues and blockchain," in *Proc. 3rd Int. Conf. Comput. Commun. Technol. (ICCCT)*, Feb. 2019, pp. 136–140.

[11] Z. Ma, M. Jiang, H. Gao, and Z. Wang, "Blockchain for digital rights management," *Future Gener. Comput. Syst.*, vol. 89, pp. 746–764, Dec. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18301614

[12] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, vol. 10, p. 21260, Oct. 2008.

[13] I. Makarov and A. Schoar, "Blockchain analysis of the Bitcoin market," *Nat. Bur. Econ. Res., Work*. Paper 29396, Oct. 2021. [Online]. Available: http://www.nber.org/papers/w29396

[14] V. Buterin, "A next-generation smart contract and decentralized application platform," Tech. Rep., 2014.

[15] G. Wang and M. Nixon, "SoK: Tokenization on blockchain," in *Proc. 14th IEEE/ACM Int. Conf. Utility Cloud Comput. Companion*, New York, NY, USA, Dec. 2021, pp. 1–9, doi: 10.1145/3492323.3495577.

[16] D. Grishin, K. Obbad, P. Estep, K. Quinn, S. W. Zaranek, A. W. Zaranek, W. Vandewege, T. Clegg, N. César, M. Cifric, and G. Church, "Accelerating genomic data generation and facilitating genomic data access using decentralization, privacy-preserving technologies and equitable compensation," *Blockchain Healthcare Today*, vol. 1, pp. 1–23, Dec. 2018. [Online]. Available: https://blockchainhealthcaretoday.com/index.php/journal/article/view/34

[17] C. Lepore, M. Ceria, A. Visconti, U. P. Rao, K. A. Shah, and L. Zanolini, "A survey on blockchain consensus with a performance comparison of PoW, PoS and pure PoS," *Mathematics*, vol. 8, no. 10, p. 1782, Oct. 2020. [Online]. Available: https://www.mdpi.com/2227-7390/8/10/1782
[18] T. K. Mackey, T.-T. Kuo, B. Gummadi, K. A. Clauson, G. Church, D. Grishin, K. Obbad, R. Barkovich, and M. Palombini, "'Fit-for-purpose?'—Challenges and opportunities for applications of blockchain technology in the future of healthcare," *BMC Med.*, vol. 17, no. 1, p. 68, Mar. 2019, doi: 10.1186/s12916-019-1296-7.
[19] U. W. Chohan, "Non-fungible tokens: Blockchains, scarcity, and value," in *Critical Blockchain Research Initiative (CBRI) Working Papers*. 2021, doi: 10.2139/ssrn.3822743.
[20] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (NFT): Overview, evaluation, opportunities and challenges," 2021, *arXiv:2105.07447*.
[21] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018, doi: 10.1504/IJWGS.2018.095647.
[22] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17315765
[23] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic, "Trustworthy blockchain oracles: Review, comparison, and open research challenges," *IEEE Access*, vol. 8, pp. 85675–85685, 2020.
[24] M. Legault, "A Practitioner's view on distributed storage systems: Overview, challenges and potential solutions," *Technol. Innov. Manage. Rev.*, vol. 11, no. 6, pp. 32–41, Jul. 2021. [Online]. Available: https://timreview.ca/article/1448.
[25] A. M. Abdelrahman, J. J. P. C. Rodrigues, M. M. E. Mahmoud, K. Saleem, A. K. Das, V. Korotaev, and S. A. Kozlov, "Software-defined networking security for private data center networks and clouds: Vulnerabilities, attacks, countermeasures, and solutions," *Int. J. Commun. Syst.*, vol. 34, no. 4, p. e4706, Mar. 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4706
[26] H. K. Bella and S. Vasundra, "A study of security threats and attacks in cloud computing," in *Proc. 4th Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Jan. 2022, pp. 658–666.
[27] R. K. Bunkar, B. Mishra, and P. K. Rai, "Secure data storage analysis in cloud computing: An advanced approach," *Recent Adv. Math. Res. Comput. Sci.*, vol. 2, pp. 147–155, Oct. 2021. [Online]. Available: https://stm.bookpi.org/RAMRCS-V2/article/view/4467
[28] J. Benet and N. Greco, "Filecoin: A decentralized storage network," *Protoc. Labs*, pp. 1–36, 2018.
[29] O. Oren, "ICO's, DAO's, and the SEC: A partnership solution," *Colum. Bus. L. Rev.*, p. 617, 2018.
[30] S. Williams, V. Diordiiev, L. Berman, and I. Uemlianin, "Arweave: A protocol for economically sustainable information permanence," Arweave Yellow Paper, 2019. [Online]. Available: https://www.arweave.org/yellow-paper.pdf
[31] S. Mishra, S. Singh, and S. T. Ali, "MPoWS: Merged proof of ownership and storage for block level deduplication in cloud storage," in *Proc. 9th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2018, pp. 1–7.
[32] N. B. Somy, K. Kannan, V. Arya, S. Hans, A. Singh, P. Lohia, and S. Mehta, "Ownership preserving AI market places using blockchain," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 156–165.
[33] R. Raj, N. Rai, and S. Agarwal, "Anticounterfeiting in pharmaceutical supply chain by establishing proof of ownership," in *Proc. IEEE Region 10 Conf. (TENCON)*, Oct. 2019, pp. 1572–1577.
[34] S. M. H. Bamakan, N. Nezhadsistani, O. Bodaghi, and Q. Qu, "Patents and intellectual property assets as non-fungible tokens; key technologies and challenges," *Sci. Rep.*, vol. 12, no. 1, p. 2178, Feb. 2022, doi: 10.1038/s41598-022-05920-6.
[35] N. Martinod, K. Homayounfar, D. Lazzarotto, E. Upenik, and T. Ebrahimi, "Towards a secure and trustworthy imaging with non-fungible tokens," in *Applications of Digital Image Processing XLIV*, A. G. Tescher and T. Ebrahimi, Eds., vol. 11842. Bellingham, WA, USA: SPIE, 2021, pp. 401–411, doi: 10.1117/12.2598436.
[36] H.-S. Huang, T.-S. Chang, and J.-Y. Wu, "A secure file sharing system based on IPFS and blockchain," in *Proc. 2nd Int. Electron. Commun. Conf.*, New York, NY, USA, 2020, pp. 96–100, doi: 10.1145/3409934.3409948.
[37] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *Proc. 2nd Int. Conf. Open Big Data (OBD)*, Aug. 2016, pp. 25–30.
[38] H. R. Hasan, K. Salah, I. Yaqoob, R. Jayaraman, S. Pesic, and M. Omar, "Trustworthy IoT data streaming using blockchain and IPFS," *IEEE Access*, vol. 10, pp. 17707–17721, 2022.
[39] C. Lao, C. Mao, and A. Sy, "Security analysis on Snapchat," Tech. Rep., 2018.
[40] C. Collberg, "Code obfuscation: Why is this still a thing?" in *Proc. 8th ACM Conf. Data Appl. Secur. Privacy*, New York, NY, USA, Mar. 2018, pp. 173–174, doi: 10.1145/3176258.3176342.
[41] D. Zheng, L. Xue, C. Yu, Y. Li, and Y. Yu, "Toward assured data deletion in cloud storage," *IEEE Netw.*, vol. 34, no. 3, pp. 101–107, May 2020.
[42] M. Gao, H. Dang, and E.-C. Chang, "TEEKAP: Self-expiring data capsule using trusted execution environment," in *Proc. Annu. Comput. Secur. Appl. Conf.*, New York, NY, USA, Dec. 2021, pp. 235–247, doi: 10.1145/3485832.3485919.
[43] Y. Zhu, S. Yang, G. Gan, and X. He, "An efficient retrograde storage for self-destructing messages on frequently colliding hash table," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2018, pp. 914–919.
[44] V. Costan and S. Devadas, "Intel SGX explained," Cryptology ePrint Archive, Tech. Rep. 2016/086, 2016, [Online]. Available: https://ia.cr/2016/086
[45] D. Grishin, K. Obbad, and G. M. Church, "Data privacy in the age of personal genomics," *Nature Biotechnol.*, vol. 37, no. 10, pp. 1115–1117, Oct. 2019, doi: 10.1038/s41587-019-0271-3.
[46] A. Z. Faroukhi, I. El Alaoui, Y. Gahi, and A. Amine, "A novel approach for big data monetization as a service," in *Advances on Smart and Soft Computing* (Advances in Intelligent Systems and Computing), F. Saeed, T. Al-Hadhrami, F. Mohammed, and E. Mohammed, Eds. Singapore: Springer, 2021, pp. 153–165.
[47] W. Badreddine, K. Zhang, and C. Talhi, "Monetization using blockchains for IoT data marketplace," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2020, pp. 1–9.
[48] M. Travizano, C. Sarraute, G. Ajzenman, and M. Minnoni, "Wibson: A decentralized data marketplace," 2018, *arXiv:1812.09966*.
[49] A. Alsharif and M. Nabil, "A blockchain-based medical data marketplace with trustless fair exchange and access control," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.
[50] G. Zheng, L. Gao, L. Huang, and J. Guan, *Decentralized Application*. Singapore: Springer, 2021, pp. 253–280, doi: 10.1007/978-981-15-6218-1_9.
[51] M. Li, J. Zhu, T. Zhang, C. Tan, Y. Xia, S. Angel, and H. Chen, "Bringing decentralized search to decentralized services," in *Proc. Symp. Operating Syst. Design Implement. (OSDI)*, 2021, pp. 331–347. [Online]. Available: https://par.nsf.gov/biblio/10312008
[52] T. Kyriacos, Z. Christina, and S. George, "Ranking domain names using various rating methods," in *Proc. 9th Int. Multi-Conf. Comput. Global Inf. Technol. (ICCGI)*, A. Leist and T. Pankowski, Eds. Spain, 2014, pp. 107–114.
[53] H. A. Al-Hussaini and H. Al-Dossari, "A lexicon-based approach to build reputation from social media," *Int. Res. J. Electron. Comput. Eng.*, vol. 2, no. 2, pp. 14–19, 2016. [Online]. Available: https://researchplusjournals.com/index.php/IRJECE/article/view/163
[54] N. Nizamuddin, H. R. Hasan, and K. Salah, "IPFS-blockchain-based authenticity of online publications," in *Blockchain—ICBC* (Lecture Notes in Computer Science), S. Chen, H. Wang, and L.-J. Zhang, Eds. Cham, Switzerland: Springer, 2018, pp. 199–212.
[55] L. Kumar and N. Badal, "A review on hybrid encryption in cloud computing," in *Proc. 4th Int. Conf. Internet Things, Smart Innov. Usages (IoT-SIU)*, Apr. 2019, pp. 1–6.
[56] J. Feist, G. Grieco, and A. Groce, "Slither: A static analysis framework for smart contracts," 2019, *arXiv:1908.09878*.