Blockchain Security Encryption to Preserve Data Privacy and Integrity in Cloud Environment

Blockchain Security Encryption to Preserve Data Privacy and Integrity in Cloud Environment

1st Abel Yeboah-Ofori
School of Computing and Eng
University of West London
United Kingdom
Abel.yeboah-ofori@uwl.ac.uk

1st Sayed Kashif Sadat
School of Computing and Eng
University of West London
United Kingdom
21524360@student.uwl.ac.uk

2nd Iman Darvishi
School of Computing and Eng
University of West London
United Kingdom
Iman.darvishi@uwl.ac.uk

***Abstract*--Blockchain security issues in relation to encryption for data privacy and integrity in cloud computing have become challenging due to the decentralized and peer-to-peer systems for securing data storage and transfer in smart contracts. Further, Blockchain technology continues revolutionizing how we handle data, from improving transparency to enhancing security. However, various instances of data breaches, piracy, and hacking attacks have compromised the safety measures employed by these providers. The paper aims to explore Blockchain technology and how encryption algorithms are used to leverage security properties to uphold data privacy and integrity in a cloud environment to enhance security. The novelty contribution of the paper is threefold. First, we explore existing blockchain attacks, vulnerabilities, and their impact on the cloud computing environment supported by numerous cloud services that enable clients to store and share data online. Secondly, we used an encryption approach to detect data security by combining AES encryption, cloud storage, and Ethereum smart contracts in cloud AWS S3. Finally, we recommend control mechanisms to improve blockchain security in the cloud environment. The paper results show that AES algorithms can be used in blockchain smart contracts to enhance security, privacy, and integrity to ensure secure data in transit and at rest.**

Keywords—Blockchain Security, Encryption, Data Privacy, Integrity, Cloud, AWS

I.INTRODUCTION

The blockchain technology and security issues for a decentralized and peer-to-peer system for secure data storage and transfer have gained significant attention since Satoshi Nakamoto's white paper on Bitcoin [1]. Data is becoming an increasingly important asset as we navigate the digital era. Due to the ubiquitous nature of the internet, data is stored and shared via cloud services, which have become the primary medium for exchanging and storing information. This evolution, however, has resulted in an increasing concern over data security and privacy. Security measures currently employed by cloud service providers are vulnerable to data breaches, piracy, and cyberattacks caused by various security issues. Moreover, Traditional security solutions, which rely on centralization, are no longer sufficient to protect against cyber-attacks and breaches. Innovative solutions are needed to bolster their security to ensure complete protection of sensitive information in these systems. Blockchain technology, with its decentralized and secure nature, has the potential to address these challenges by providing a new approach to data privacy and integrity in the cloud environment. The following proposal will analyze the use of blockchain-based security encryption to provide data integrity, ensuring that data has not been tampered with or altered. Additionally, due to the secure nature of this technology, many different applications for blockchain technology are being utilized today, including cloud computing, banking, the Internet of Things, big data, healthcare, etc. One of the key features of the blockchain is that it is structured as a digital log file that is held in the form of linked groups called blocks. There is a cryptographic link between each block and its predecessor. Due to the cryptographic nature of blockchain systems, many security experts speculate that they are resilient enough to withstand continuous hacking and security threats [2]. Furthermore, Our paper explores an innovative approach to encryption and cloud storage that integrates AES encryption with Ethereum smart contracts and cloud storage. Combining these two technologies makes it possible to secure data during transit and storage and provide a transparent record of data access and modification that is tamper-resistant and transparent. According to our analysis, a cloud-based data management model that creates an immutable log of activities prevents unauthorized access to the data and enhances trust in cloud-based data storage. Furthermore, some of the key challenges associated with blockchain security in the cloud environment cannot be ignored, including endpoints, scalability, criminal activities, third parties, and a variety of attacks on blockchain systems, including the 51% attack, a vulnerability in blockchain technology where attackers or groups can gain control of over 50% of the blockchain of a network. We aim to create a system where data is safe and transparently verifiable, increasing user confidence in cloud services. The objectives of the paper are as follows:

- To demonstrate the capabilities of blockchain technology in enhancing the security and privacy of cloud computing services
- to provide a method that securely stores data before storing it in the cloud using AES encryption. This encryption will assist in safeguarding the data while it is in use and at rest, reducing the possibility of unauthorized access or data breaches.
- the creation of an Ethereum smart contract that can be used to keep track of data exchange activity logs. The smart contract will store crucial metadata such as data access, alterations, and the users involved to provide an unchangeable and transparent record of events.
- Describe the optimal approach for implementing blockchain-based data privacy using smart contracts and encryption techniques.
- To show the viability and usefulness of this strategy by a real-world implementation that involves storing encrypted data on a cloud platform like AWS S3 and keeping track of relevant logs on the Ethereum blockchain.

By achieving these objectives, we aim to pave the way for new security standards that leverage the power of blockchain technology and robust cryptographic methods to create a more secure and trustworthy environment for cloud storage

and data exchange by leveraging the power of blockchain technology.

i. How do blockchain technology's trustless system and cryptographic security contribute to its effectiveness in protecting against security and privacy breaches?
ii. What are the vulnerabilities and countermeasures associated with the blockchain environment?
iii. inefficiency and vulnerability of centralized security solutions in distributed applications.

II. RELATED WORKS

This section discusses related works and state of the art in Blockchain, a proposed decentralized and trusted cloud data provenance architecture using blockchain technology. Regarding architecture using blockchain technology, [4] suggests that using blockchain for data origin management can result in records that cannot be altered, increase the transparency of data accountability in the cloud, and improve the privacy and accessibility of the origin data. The authors proposed blockchain-based data provenance architecture (ProvChain) for cloud storage applications that provides data operations assurance while enhancing privacy and availability. ProvChain records operation history as provenance data and anchors it to a blockchain transaction. However, state-of-the-art cloud-based provenance services are vulnerable to accidental corruption or malicious forgery of provenance data.[4]. Further, [5] proposed a solution called "cloud@blockchain" that utilizes blockchain technology to enhance the security and privacy of cloud computing services. The authors design two functions: anonymous file sharing and inspections for illegally uploaded files to demonstrate the capabilities of their solution by comparing the performance of their hybrid blockchain with cache, a pure blockchain, and a traditional database in accessing data, and find that the hybrid blockchain significantly outperforms the other two options. The results show that using blockchain as a platform for secure cloud computing services can provide improved privacy and protection against cyber-attacks. However, the anonymous file-sharing mechanism and inspection feature for illegally uploaded files also address specific concerns related to data sharing and infringement on cloud computing platforms [5],. Furthermore, [6] proposed blockchain-based approaches for several security services, including authentication, confidentiality, privacy, access control list, data and resource provenance, and integrity assurance, which are critical for the current distributed applications. A certificate authority was considered considering the large amount of data being processed over the network in cloud computing. However, the services are prone to attacks on the centralized controller due to scalability, privacy, anonymity, and time-consuming [6]. Moreover, [8] proposed a Homomorphic encryption scheme and Byzantine Fault Tolerance consensus to ensure data integrity and decrease the cloud service provider's absolute control over data. The authors proposed a master hash value of databases instead of AES, generated by service providers after computations, that are preserved on Bitcoin or Ethereum blockchain networks for immutability. However, the paper needs more empirical evidence from real-world implementation and testing to verify the theoretical framework [8]. Regarding blockchain-based data protection system for medical records is explored. [7] proposed a reliable storage method that ensures stored data's primitiveness and verifiability while preserving user privacy. The technique is based on the blockchain framework that uses cryptography algorithms to preserve data during data submission, manipulation, query, and verification. However, the framework proposed is conceptual and needs to provide more information on practical implementation [7].

Further, [19] proposed the application of blockchain, which integrate blockchain technology and stores data in encrypted chunks on hospital websites to enhance data security, trustworthiness, efficiency, and real-time system for storing patient reports in a cloud environment. The proposed system divides patient reports into chunks, encrypts these chunks using the AES, and stores them in secure AWS S3 buckets. The authors highlight that their proposed system aligns with the Health Insurance Portability and Accountability Act (HIPAA) rules and can enhance trust between doctors and patients. However, the paper did not address user acceptance issues, an important factor when introducing new technology. Not exploring potential barriers and strategies to increase user adoption could enrich future studies [19]. Furthermore,[9] published a comprehensive analysis of the vulnerabilities and countermeasures associated with blockchain technology and threats that can be deliberate or accidental [9]. Moreover, [12] proposed preserving data security in a cloud environment using an adaptive homomorphic blockchain technique that focuses on the challenges of securing data in cloud storage and maintaining efficiency in the encryption and decryption process. The authors propose a novel Ring Character Hash (RCH) with Ring Elliptical Curve Cryptography (RECC) Homomorphic model for simultaneously encrypting and decrypting large files. Related to cloud computing security, blockchain technology, cryptographic methods, and homomorphic encryption, highlighting the significance of blockchain-based encryption in securing cloud computing and the role of blockchain technology in enhancing security levels [19].

III. APPROACH

This section discusses the approach used for our implementations by implementing a blockchain, AES encryption, and cloud storage environment, including AWS and Ethereum.

We implemented the AES algorithm using an open-source cryptographic library and PyCharm IDE to build and test our encryption and decryption methods; we utilized PyCharm, a professional Python-integrated programming environment. PyCharm IDE was chosen for its excellent coding assistance, debugging features, and comprehensive Python programming tools.

- *Amazon Web Services (AWS):* The encrypted data was then uploaded to Amazon S3, a scalable object storage service offered by AWS. This platform was chosen for its high durability, availability, and comprehensive security and compliance capabilities.
- *Ethereum Blockchain:* We used Ethereum as our blockchain platform. Ethereum offers a decentralized platform that runs smart contracts, applications that run exactly as programmed without any possibility of downtime, censorship, fraud, or third-party interference.

- Remix *IDE:* To write and deploy our smart contract, we used Remix IDE, a powerful open-source tool that allows developing, testing, deploying, and interacting with smart contracts.

A. Our procedures included the following steps:

- Data Encryption: We first encrypted our data using AES encryption, converting plaintext data into cipher text, thus ensuring it was unreadable without the correct decryption key.
- Data Upload: The encrypted data was manually uploaded to AWS S3, securely stored, and made accessible to authorized users.
- Smart Contract Deployment: We wrote and deployed a smart contract on the Ethereum blockchain using the Remix IDE. This contract facilitated the logging of data-related activities onto the blockchain.
- Metadata and Logging: Every time an operation was performed on the data, metadata about the operation and activity logs were written onto the blockchain via the smart contract, providing a transparent and tamper-resistant record of data access and modifications.
- Testing and Verification: We conducted rigorous testing throughout the process to ensure the functionality and security of our system. We also demonstrated the feasibility and advantages of our proposed system.

B. Data analysis

This study aimed to analyze the proposed system's effectiveness, with particular attention to the AES encryption process, the Ethereum blockchain smart contract, and the use of the AWS S3 cloud platform. We evaluated the encryption and decryption times to understand how quickly the AES algorithm can encrypt and decrypt data for varying data sizes. Additionally, we tested the efficiency and feasibility of our blockchain-based approach by analyzing the gas cost of the smart contract per transaction, the transaction time, and the accuracy of the logged data based on the Smart Contracts. Furthermore, Our analysis of the cloud platform consisted of verifying that encrypted data had been uploaded successfully, evaluating access controls, and analyzing retrieval times for various data types on the platform.

IV. IMPLEMENTATION AND PROCESS

We have comprehensively summarized the steps to execute the thesis topic, "Blockchain-based security encryption to preserve data privacy and integrity in Cloud Environment". With the help of this implementation, it will be possible to keep the information stored securely in the cloud environment, free from unauthorized access or modification, and protected against data tampering. This has been achieved by combining the robustness of blockchain technology with the strength of AES encryption to provide an effective solution and use the Amazon S3 bucket. It is important to note that the implementation process comprises several interconnected steps, each of which plays an important role in realizing the project's overall objective. As a result of carefully following the steps outlined below, a comprehensive system can be created that not only address the concerns related to the privacy and integrity of sensitive data in the cloud environment but also maintains proof of transparency and a tamper-proof record of encryption operations in the cloud. After the encryption process, the encrypted file is saved on the local file system. Below is the sequence diagram, which shows the implementation process of our paper.

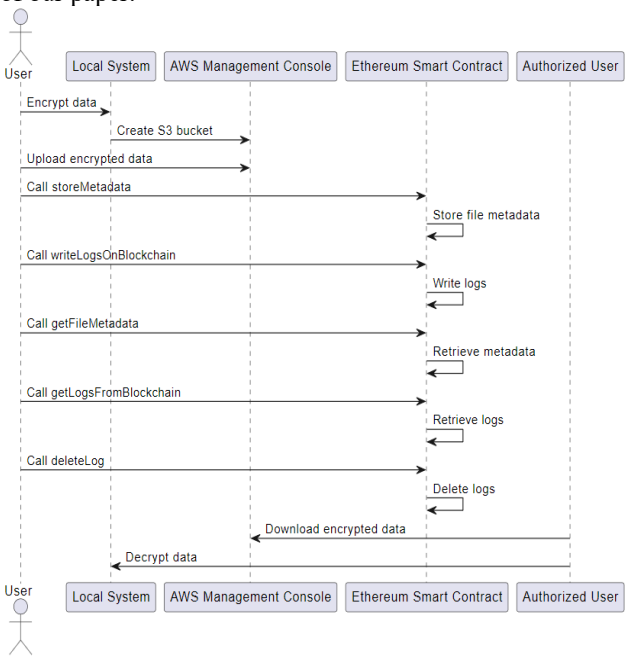


Fig. 1. Overall Implementation Process for Secure Data Storage and Log Management.

In the sequence diagram, we can see interactions between the user, the local system, the AWS S3 bucket, and the Ethereum smart contract as secure data storage and log management are implemented. The sequence of events depicted in the "Fig. 1" may be outlined using the following steps:

- The user initiates the encryption process on the local system by running the AES encryption script. The data file is encrypted, and the key is securely stored.
- The user manually uploads the encrypted data file to the AWS S3 bucket, ensuring appropriate access control policies are in place to prevent unauthorized access.
- Next, the user calls the **storeMetadata** function in the Ethereum smart contract to store the file metadata, including data hash, encryption key hash, storage location, data owner, and file name.
- To store log details, the user calls the **writeLogsOnBlockchain** function, providing information such as the user ID, log data, log time, document name, access user ID, and log permission.
- When users need to retrieve file metadata, they call the **getFileMetadata** function, which returns the data hash, encryption key hash, storage location, data owner, and file name associated with the provided data ID.
- To access logs, the user calls the **getLogsFromBlockchain** function using their user ID. The smart contract returns a list of logs associated with the user.
- If users need to delete a log entry, they call the **delete Log** function with their user ID and the document name. The smart contract removes the log entry from the list of logs.

To access the encrypted data, the user must download the encrypted file from the AWS S3 bucket and run the AES decryption script on their local machine, providing the

encryption key. Understanding how the various system parts interact and contribute to the system's management of log files and safe data storage improves as we follow the sequence of events in the diagram. This diagram's objective is to make it easier for readers to comprehend the project's implementation process quickly.

A. AES Encryption and Decryption

This project involved the development of a Python script that encrypts and decrypts files using the Advanced Encryption Standard (AES) algorithm. Using this script, a random 32-byte encryption key will be generated and used to encrypt the input file in Cipher Block Chaining (CBC) mode using the random encryption key. Furthermore, It is also possible to decrypt the encrypted file using the encryption key stored in the script. In addition, the Python script performs AES encryption and decryption using the **PyCryptodome** library in conjunction with **hashlib**, which generates data hashes and encryption keys.

Fig. 2. AES encryption/decryption algorithm in Python.

After executing the function related to the encryption/decryption algorithm using Python, illustrated in Figure 2, we can obtain the hashes of the data and the encryption keys used to encrypt the file. In our paper, these values play a crucial role, as they are subsequently stored on the blockchain through a smart contract which is a crucial part of our research. Through this approach, sensitive information related to encrypted files and the encryption keys associated with the encrypted files is securely and transparently managed.

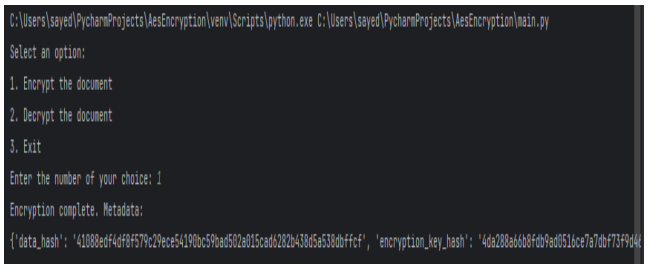


Fig. 3. Encrypting a file

When a user enters '1' into our program, the AES encryption protocol kicks into action, morphing the readable file into an encrypted format. This process also yields a unique data hash and an encryption key. The data hash is like a fingerprint for the encrypted data, while the encryption key is a vital tool that unlocks the original content from its encrypted form. It is crucial to store this encryption key for future decryption safely. With it, readable data is available. Thus, the dual presence of encryption and decryption in our AES process ensures data privacy and integrity.

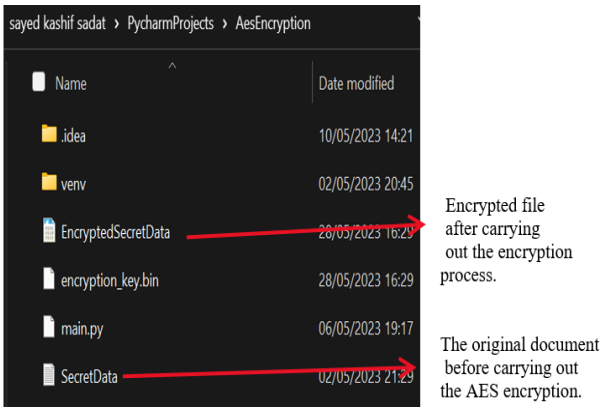


Fig. 4. After encrypting the document.

For the decryption process, we retrieve the data hash and encryption key hash from the blockchain, which were previously stored using our smart contract. By obtaining these crucial pieces of information, we can securely access and decrypt the encrypted files, ensuring the integrity and confidentiality of the data. This approach highlights the seamless integration of blockchain technology and AES encryption in managing and protecting sensitive information.

Fig. 5. Decryption of the encrypted fil

1) Uploading Encrypted Data to AWS S3: Implementing Secure Cloud Storage

Once the data is encrypted using the AES algorithm. The next step in putting the findings into practice is to encrypt the data and then transfer it securely to a cloud storage

provider like Amazon Web Services' Simple Storage provider (S3). For storing and retrieving data via the Internet, AWS offers S3, a scalable, secure, and highly accessible storage option. As a result, it is the best option for online encryption file storage.

The following steps need to be followed to upload the encrypted data to AWS S3:

2) *Set up an AWS S3 bucket*: First, create a new S3 bucket through the AWS Management Console. Further, keep data storage requirements in mind and Pick a unique name and an appropriate region for your bucket. Finally, configure the bucket with appropriate access control restrictions to prevent unauthorized access to the encrypted data.

3) *Upload encrypted data:* Once the S3 bucket is set up, you must navigate to the AWS Management Console to access the S3 service, enabling you to upload encrypted data. Upon opening the bucket you created, click on the "Upload" button to start uploading your files. In the file picker dialog, select the encrypted data file stored locally on your computer. Ensure the correct metadata (e.g., content type) is set during uploading. Once the file is uploaded, you can view it in the S3 bucket.

4) *Fine-tune access control:* Properly configuring the access control policies for the S3 bucket containing the encrypted data is of utmost importance. Make sure only authorized users can read and write to the bucket. Consider using AWS Identity and Access Management (IAM) to create custom roles and permissions for users requiring data access.

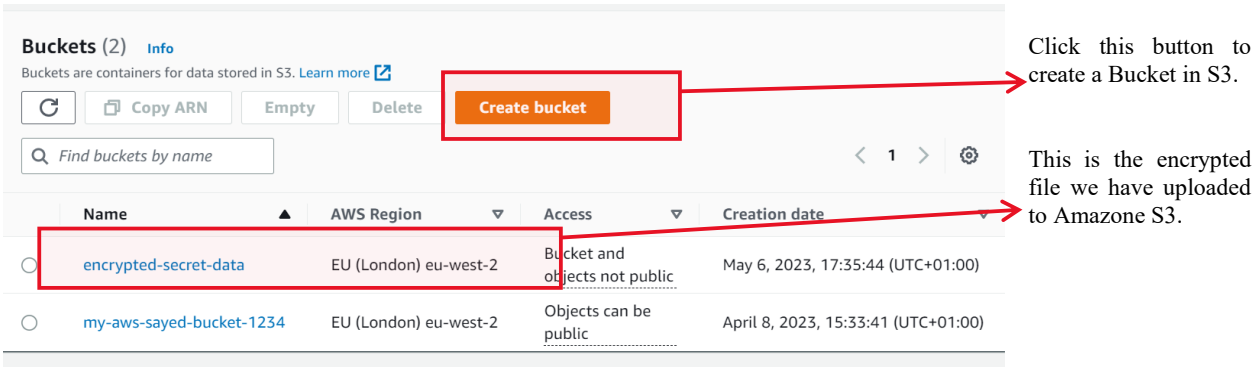


Fig. 6. (*Amazone S3 Bucket creation*)

B. *Smart Contract Development:*

Developing the smart contract is one of the most important parts of the thesis implementation process. With the Solidity programming language, the smart contract acts as a decentralized ledger, storing file metadata and logs on the Ethereum blockchain and thus serves as a decentralized ledger. The Solidity contract contains several elements, such as data structures, mappings, events, and functions, all written in Solidity. Data structures include UserDetails, Logs, and FileMetadata, while mappings associate user and data IDs with their corresponding details. The several components that make up the smart contract are described below.

C. *Smart contract for secure log management and file metadata storage on the blockchain*

The LogsStorage smart contract is a decentralized system for securely storing and managing file metadata and logs. It comprises three primary data structures: UserDetails, Log, and FileMetadata.

- UserDetails contains logs for a specific user.
- Log contains information about documents, such as their names, contents, timestamps, permissions, and user access information.
- FileMetadata stores details about an encrypted file, including the hash, the encryption key hash, the storage location, the owner, and the file name.

Figure 7 shows how we deploy the smart contract and the next step after writing and compiling it in Remix IDE.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract LogsStorage {
    struct UserDetails {
        Log[] logs;
    }

    struct Log {
        string documentName;
        string logContent;
        string logTime;
        string accessUserId;
        string logPermission;
    }

    struct DataMetadata {
        string dataHash;
        string encryptionKeyHash;
        string storageLocation;
    }

    mapping(string => UserDetails) User;
    mapping(string => DataMetadata) dataMetadataMapping;

    event Actual(Log[]);

    function storeMetadata(string memory dataID, string memory dataHash,
     string memory encryptionKeyHash, string memory storageLocation) public {
        DataMetadata memory metadata;
        metadata.dataHash = dataHash;
        metadata.encryptionKeyHash = encryptionKeyHash;
        metadata.storageLocation = storageLocation;

        dataMetadataMapping[dataID] = metadata;
    }
  function writeLogsOnBlockchain(
        string memory userID,
        string memory logData,
        string memory logTime,
        string memory documentName,
        string memory accessUserId,
        string memory logPermission
    ) public {
        Log memory l;
        l.logContent = logData;
        l.logTime = logTime;
        l.documentName = documentName;
        l.accessUserId = accessUserId;
        l.logPermission = logPermission;

        User[userID].logs.push(l);
    }
```

Fig. 7. Secure Log Management and File Metadata Storage on Blockchain

Sending a contract to the blockchain is the process of "deploying" it, shown in Figure 8 below, thereby turning it into a blockchain-based instance. As this procedure includes completing network transactions, gas (a tiny quantity of the cryptocurrency Ether used on the Ethereum blockchain) is needed.

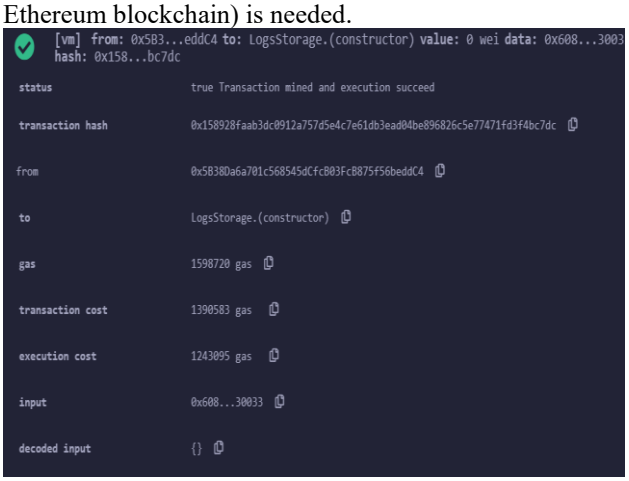


Fig. 8. Results After Deploying a Smart Contract

Furthermore, the smart contract uses two mappings - User IDs and Data IDs - to associate user IDs and data IDs with the respective information. Additionally, the smart contract incorporates various functions to facilitate its operations:

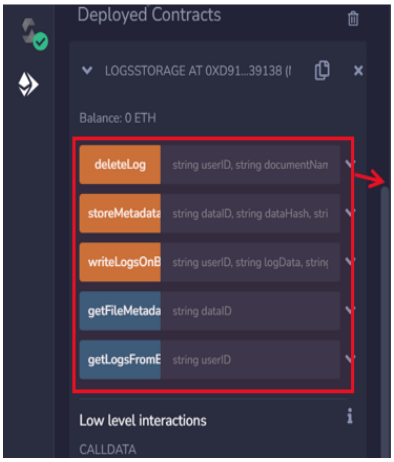
i. storeMetadata function: Enables users to save file metadata to the blockchain.

ii. getFileMetadata function: Retrieves the stored metadata.

iii. writeLogsOnBlockchain function: Writes logs to the blockchain, accepting parameters like userID, logData, logTime, documentName, accessUserId, and logPermission. UserDetails data structure stores the logs for each user.

iv. getLogsFromBlockchain function: Facilitates the retrieval of logs. It returns a formatted string containing all logs for the given user ID or a "Log not found" message if no logs are found.

v. deleteLog function: Allows users to remove a log associated with a specific user ID and document name. It iterates through the logs, finds the log with the matching document name, removes it from the array, and returns a message indicating whether the deletion was successful or if the log was not found.



So after we deployed the **LogsStorage** smart contract, we can interact with it by providing the necessary inputs for its various functions respectively.

Fig. 9. After deploying the LogStrorage smart contract

D. *Storing and retrieving the metadata*

In our contract, the **StoreMetadata** function writes file metadata into the blockchain. Metadata encompasses the data hash, encryption key hash, storage location, data owner, and file name. It is saved on the blockchain to provide a tamper-proof data record. We input this metadata using the Remix IDE, shown in Figure 10 below. This function will then store the metadata onto the Ethereum blockchain, offering a strong, immutable record of your file's details.

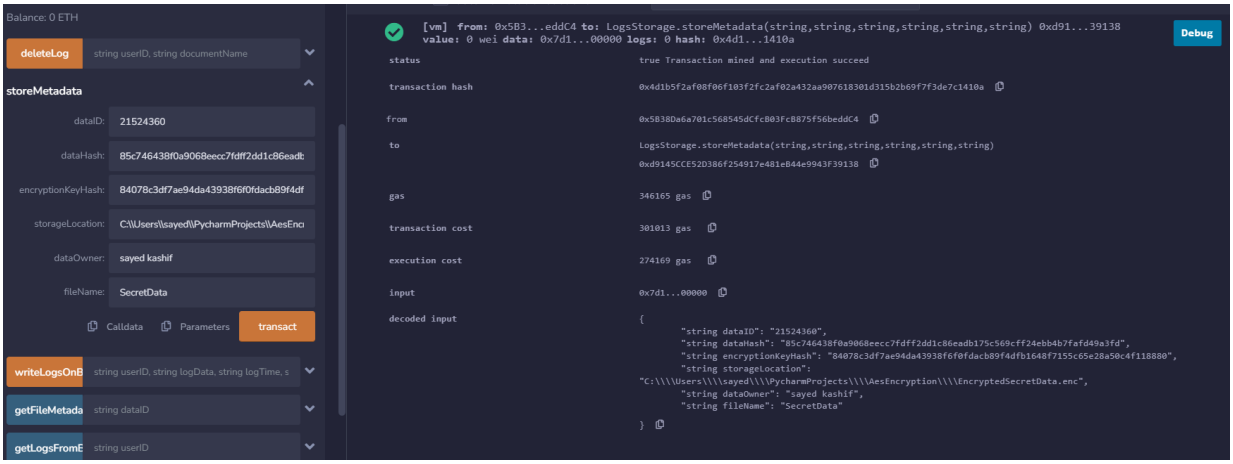


Fig. 10. Storing the data into the smart contract

The next step involves utilizing the getFileMetadata function in our LogsStorage contract to fetch file metadata from the blockchain. By entering a unique dataID, as shown in Figure 11, this function retrieves corresponding metadata details such as dataHash, encryptionKeyHash, storageLocation, dataOwner, and fileName. This step helps ensure data integrity by confirming that metadata aligns with originally stored information.

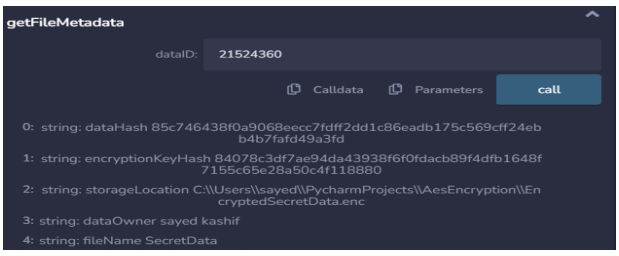


Fig 11: Retrieving the metadata

E. Result analysis of the implementation

After implementing the proposed system, we analysed the results to evaluate our solution's effectiveness in maintaining data privacy and integrity within a cloud environment. During the analysis, the following aspects were considered:

- Data encryption and decryption: Data is successfully encrypted using the AES encryption algorithm, making it unreadable without the correct decryption key. The data was returned to its original, readable form by applying the appropriate key to the decryption process. This showed that our encryption and decryption methods effectively safeguarded the data and protected its confidentiality.
- Cloud Storage Security: The encrypted data was stored securely on the AWS S3 bucket, which provided multiple layers of protection to maintain data privacy. Individuals with permission could read and download the encrypted files, limiting access to the kept material. This proves our solution's cloud storage aspect effectively prevented unauthorized data access.
- *Smart Contract Functionality:* Our Ethereum smart contract, deployed using Remix IDE, performed as intended. It recorded metadata and logged all data-related activities on the blockchain, creating a transparent and tamper-resistant data access and modifications record. This not only facilitated auditing and tracking of data usage but also enhanced overall data integrity.
- *Blockchain Transparency and Immutability:* The Ethereum blockchain provides a transparent and immutable ledger of all data-related activities. This aspect of our system played a crucial role in ensuring data integrity, as it made it possible to verify the authenticity of the data and trace any unauthorized access or modifications.
- *Performance and Scalability:* Our proposed system demonstrated acceptable performance levels, with data encryption, decryption, and logging processes being executed efficiently. Additionally, the system showed potential for scalability, as the underlying technologies (AES, AWS S3, and Ethereum smart contract) are designed to accommodate a growing number of users and increasing amounts of data.

Moreover, a summary of the threats and vulnerabilities associated with Blockchain and their countermeasures can be found in the following Table.

TABLE 1. Threat and Vulnerabilities with Countermeasures

Threats and Vulnerabilities:	**Countermeasures:**
1). Losing a private key in the event of a Blockchain transaction: The private key gives you sole control over your transaction, thus the prime target for cyber attackers	Incorporates salt values in each operational profile as a standard sequence. The private key must spread over multiple assets. It must be synchronized during the regeneration of the private key to execute a Bitcoin transaction (Rehman et al., 2018, cited in [21]).
2). Weakened cryptographic primitives: Bitcoin and other cryptocurrencies rely on cryptographic primitives to ensure security and proper operation. Such primitives often become less effective over time because of advancements in cryptanalysis and the processing capacity of attackers. Therefore, over time, the cryptographic foundations of Bitcoin will gradually become largely, if not entirely, compromised (Giechaskiel, 2016, cited in Hassan et al., 2020) [9].	Using the Coinbase transaction to launch a pre-image attack on the mining header target will be more challenging. It is advised that users avoid recycling their Bitcoin addresses. To migrate from outdated addresses, new address types should be established utilizing improved hashing and signature techniques. Instead of employing layered hashes, users can strengthen defense-in-depth by combining the primitive's Address Hash and Main Hash. A hard fork would be an excellent way to fix a weak primary hash primitive since it would restructure the headers and transactions without relying on outdated primitives (Hassan et al., 2020 cited in [9]).
***3). Double spending**:* Digital cash systems may be prone to the issue of double-spending, which occurs when a single digital token can be used to make multiple purchases. - If a beneficiary conspires with the sender to return the deposit if he decides to equivocate and double spend, there is the possibility of a collision attack against the non-equivocation contract proposed by Hassan et al. (Ruffing, 2015, cited in Hassan et al., 2020, [9]).	Use an accountable assertion algorithm. Using the time-locked Bitcoin deposit the sender creates, any sender who equivocates or double spends will be penalized Apply timestamp server as a reliable solution to detect kind of attack. The Timestamp server takes a hash of a block of items to be time-stamped and widely publishes it. A timestamp proves that data existed at a certain point (Kaushik et al., 2017, cited in [20]).
4). 51% attack: A 51% attack, also known as a majority attack, occurs when an individual or group controls more than half of the computing power on a blockchain network. This attack allows the attacker to manipulate the network by controlling most of the mining power, which can be used to double spending, block transactions, and more.	The technique of randomly selecting mining groups, put forth [2], aims to decrease the computational power and protect against 51% attacks. Studies have shown that by increasing the number of groups to two or more, the chances of an attacker successfully uncovering the next block are greatly re-duced [2].
***5). Crypto-jacking or drive-by mining**:* This type of attack involves the unauthorized use of an individual's device to mine for cryptocurrency without their knowledge or permis-sion. This can occur on various devices, including comput-ers, smartphones, tablets, and servers."	As a solution to detect this type of attack, the author suggests using a tech-nique called Minesweeper which keeps track of the CPU cache by examining the cryptographic elements present in the code used for crypto-jacking.
6). Sybil attack: This type of attack aims to compromise the credibility of a peer-to-peer network by creating false identities. Attackers create fake identities and use them to manipulate the network's reputation system.	Timing-based inference attacks, DoS attacks, and the Sybil attack can be defended against simultaneously with a two-party decentralized mixing protocol. Various consensus algorithms can prevent this attack in blockchain environments, including Proof of Work (POW) and Proof of Stack (PWS).

V. CONCLUSION

This paper's journey has shown how blockchain technology can bolster cloud security. We explored how decentralization, immutability, and transparency principles can enhance data privacy and integrity in the cloud. Our study demonstrated that integrating AES encryption, cloud storage, and Ethereum smart contracts could preserve the confidentiality and integrity of data. The encouraging results highlight that AES encryption and decryption procedures efficiently secure the data, while AWS S3 provides a robust cloud storage environment. Simultaneously, our Ethereum smart contract provides a transparent and unchangeable record of data access and updates. However, some limitations need acknowledgment. The seamless integration of AWS, Ethereum, and AES encryption poses inherent challenges, and vulnerabilities in these platforms may compromise the system's security. The system's effectiveness also heavily depends on users' proper key management.

And external factors such as network conditions and jurisdictional data privacy laws.

Our findings have significant implications, contributing to the ongoing conversation about data privacy and protection and emphasizing how blockchain technology can enhance data security in cloud environments. While timely and relevant in our increasing reliance on cloud services, this study is but a stepping stone towards a more secure data handling approach in cloud environments. Notwithstanding the promising outcomes, the journey is far from over. Future work should focus on mitigating the current limitations, optimizing the system's speed, enhancing scalability, and bolstering security. This study acts as a springboard, opening up new research directions at the intersection of blockchain technology and cloud security. Therefore, this project should be considered an important initial step towards a more robust and user-friendly system for data privacy and integrity in the cloud.

REFERENCES

[1] Taylor, P.J., Dargahi, T., Dehghantanha, A., Parizi, R.M. and Choo, K.-K.R. (2019). A systematic literature review of blockchain cyber security. *Digital Communications and Networks*, 6(2). doi:10.1016/j.dcan.2019.01.005.

[2] Hasanova, H., Baek, U.J., Shin, M.G., Cho, K. and Kim, M.S., 2019. A survey on blockchain cybersecurity vulnerabilities and possible countermeasures. *International Journal of Network Management*, *29*(2), p.e2060.

[3] Varshney, T., Sharma, N., Kaushik, I. and Bhushan, B. (2019). *Authentication amp; Encryption Based Security Services in Blockchain Technology*. [online] IEEE Xplore. doi:10.1109/ICCCIS48478.2019.8974500.

[4] Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K. and Njilla, L. (2017). ProvChain: A Blockchain-Based Data Provenance Architecture in Cloud Environment with Enhanced Privacy and Availability. *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. doi:10.1109/ccgrid.2017.8.

[5] Tsai, W.-Y., Chou, T.-C., Chen, J.-L., Ma, Y.-W. and Huang, C.-J. (2020). *Blockchain as a Platform for Secure Cloud Computing Services*. [online] IEEE Xplore. doi:10.23919/ICACT48636.2020.9061435.

[6] Salman, T., Zolanvari, M., Erbad, A., Jain, R. and Samaka, M. (2019). Security Services Using Blockchains: A State of the Art Survey. *IEEE Communications Surveys & Tutorials*, 21(1), pp.858–880. doi:10.1109/comst.2018.2863956.

[7] Li, H., Zhu, L., Shen, M., Gao, F., Tao, X. and Liu, S. (2018). Blockchain-Based Data Preservation System for Medical Data. *Journal of Medical Systems*, 42(8). doi:10.1007/s10916-018-0997-3.

[8] Awadallah, R., Samsudin, A., Teh, J.S. and Almazrooie, M. (2021). An Integrated Architecture for Maintaining Security in Cloud Computing Based on Blockchain. *IEEE Access*, 9, pp.69513–69526. Doi https://doi.org/10.1109/access.2021.3077123.

[9] Hassan, A., Mohd, Z., Mas'ud, Shah, W., Faisal, S., Ahmad, R., Ariffin, A. and Yunos, Z. (2020). A Systematic Literature Review on the Security and Privacy of the Blockchain and Cryptocurrency. *Journal of Cyber Security*, [online] 2(1), pp.1–17. Available at: https://www.oiccert.org/en/journal/pdf/2/1/211.pdf#:~:text=In%20general%2C%20the%20main%20objective%20of%20this%20systematic [Accessed 29 Nov. 2022].

[10] Creswell, J.W. (2014). Research design: qualitative, quantitative, and mixed methods approaches. *Sage publications*. doi:10.4135/9781483375663. (Accessed: 01-20-2023)

[11] McLeod, S., 2019. Qualitative vs. Quantitative Research: Methods & Data Analysis. Available at: https://www.simplypsychology.org/qualitative-quantitative.html.(Accessed: 01-25-2023)

[12] Sharma, P., Jindal, R. and Borah, M.D. (2022). Blockchain-based cloud storage system with CP-ABE-based access control and revocation process. *The Journal of Supercomputing*. Doi https://doi.org/10.1007/s11227-021-04179-4.

[13] Jyoti, A. and Chauhan, R.K. (2022). A blockchain and smart contract-based data provenance collection and storing in a cloud environment. *Wireless Networks*. Doi https://doi.org/10.1007/s11276-022-02924-y.

[14] Wang, S., Wang, X., and Zhang, Y. (2019). A Secure Cloud Storage Framework With Access Control Based on Blockchain. *IEEE Access*, 7, pp.112713–112725. Doi https://doi.org/10.1109/access.2019.2929205.

[15] Kumar, M. and Singh, A.K. (2020). *Distributed Intrusion Detection System using Blockchain and Cloud Computing Infrastructure*. [online] IEEE Xplore. Doi:https://doi.org/10.1109/ICOEI48184.2020.9142954.

[16] Devmane, V., Lande, B.K., Joglekar, J. and Hiran, D. (2022). Preserving Data Security in Cloud Environment Using an Adaptive Homomorphic Blockchain Technique. *Arabian Journal for Science and Engineering*, 47(8), pp.10381–10394. Doi https://doi.org/10.1007/s13369-021-06347-3.

[17] Ravishankar, B., Kulkarni, P. and Vishnudas, M.V. (2020). *Blockchain-based Database to Ensure Data Integrity in Cloud Computing Environments*. [online] IEEE Xplore. Doi https://doi.org/10.23919/ICOMBI48604.2020.9203500.

[18] S. Joseph Gabriel and P. Sengottuvelan (2021). An Enhanced Blockchain Technology with AES Encryption Security System for Healthcare System. Doi:https://doi.org/10.1109/icosec51865.2021.9591956.

[19] Yadav, D., Shinde, A., Nair, A., Patil, Y. and Kanchan, S. (2020). *Enhancing Data Security in Cloud Using Blockchain*. [online] IEEE Xplore. Doi:https://doi.org/10.1109/ICICCS48265.2020.9121109.

[20] Kaushik, A., Choudhary, A., Ektare, C., Thomas, D. and Akram, S. (2017). Blockchain — Literature survey. [online] IEEE Xplore. doi:10.1109/RTEICT.2017.8256979.

[21] Rehman, H. ur, Khan, U.A., Nazir, M. and Mustafa, K. (2018). Strengthening the bitcoin safety: a graded span based key partitioning mechanism. International Journal of Information Technology. doi:10.1007/s41870-018-0252-7.