cryptography

Article

A Novel Method of Secured Data Distribution Using Sharding Zkp and Zero Trust Architecture in Blockchain Multi Cloud Environment

Komala Rangappa [1,2,†], Arun Kumar Banavara Ramaswamy [1,3,*,†], Mahadeshwara Prasad [3,†] and Shreyas Arun Kumar [4,†]

1 VTU Research Center, Department of Master of Computer Applications (MCA), BMS Institute of Technology & Management, Bengaluru 560064, India; komal.uday@gmail.com
2 Department of Computer Applications, M.S. Ramaiah Institute of Technology, Bengaluru 560054, India
3 Department of Computer Science & Engineering, BMS Institute of Technology & Management, Bengaluru 560064, India; mahadeshwara.prasad07@gmail.com
4 Department of Computer Science & Engineering, Sai Vidya Institute of Technology, Bengaluru 560064, India; shreyasvasista05@gmail.com
* Correspondence: arunkumarbr@bmsit.in; Tel.: +91-98-8600-8210
† These authors contributed equally to this work.

Abstract: In the era of cloud computing, guaranteeing the safety and effectiveness of data management is of utmost importance. This investigation presents a novel approach that amalgamates the sharding concept, encryption, zero-knowledge proofs (zkp), and blockchain technology for secure data retrieval and data access control to improve data security, efficiency in cloud storage and migration. Further, we utilize user-specific digital wallets for secure encryption keys in order to encrypt the file before storing into the cloud. As Large files (greater than 50 MB) or Big data files (greater than 1 TB) require greater computational complexity, we leverage the sharding concept to enhance both space and time complexity in cloud storage. Hence, the large files are divided into shards and stored in different database servers. We also employ a blockchain smart contract to enhance secure retrieval of the file and also a secure access method, which ensures the privacy of the user. The zk-snark protocol is utilized to ensure the safe transfer of data between different cloud services. By utilizing this approach, data privacy is preserved, as only the proof of the data's authenticity is shared with the verifier at the destination cloud, rather than the actual data themselves. The suggested method tackles important concerns related to data protection, privacy, and efficient resource utilization in cloud computing settings by ensuring it meets all the cloud policies required to store data. Since the environment maintains the privacy of the user data and the raw data of the user is not stored anywhere, the entire environment is set up as a Zero trust model.

Keywords: cloud computing; big data; sharding concept; blockchain; smart contract; digital wallets; zero knowledge proof; zk-snarks; zero trust model; data privacy

check for updates

Citation: Rangappa, K.; Ramaswamy, A.K.B.; Prasad, M.; Kumar, S.A. A Novel Method of Secured Data Distribution Using Sharding Zkp and Zero Trust Architecture in Blockchain Multi Cloud Environment. *Cryptography* **2024**, *8*, 39. https://doi.org/10.3390/cryptography8030039

Academic Editor: Josef Pieprzyk

Received: 29 July 2024
Revised: 20 August 2024
Accepted: 26 August 2024
Published: 2 September 2024

1. Introduction

In the era of digital transformation, the protection and confidentiality of user data are of utmost significance. As the amount of data being stored and managed on cloud platforms continues to grow rapidly, it is of utmost importance to guarantee the confidentiality and security of sensitive information, preventing unauthorized access. Users expect strong security measures to protect their data from breaches and leaks, ensuring the confidentiality, integrity, and availability of their information. Consequently, establishing reliable methods for safeguarding data and transferring it between cloud environments is a crucial area of study and advancement.

Prior to storing files in the cloud, it is crucial to encrypt them as a fundamental measure to safeguard data security. It is also necessary to employ the use of digital wallets, which can be used for user authentication, using the wallet's keys for encryption. By employing encryption, sensitive data are converted into an unreadable format that can only be accessed by authorized individuals. Nevertheless, when it comes to handling extensive files, encryption alone may not be enough to tackle concerns regarding efficient storage and easy access. This is where the concept of dividing data into smaller parts comes into play. Sharding entails breaking down a substantial file into smaller, more manageable fragments known as shards. Each piece of the puzzle is encrypted independently, which not only strengthens security but also enhances the speed and ease of data storage and retrieval. By spreading these encrypted fragments across various storage nodes, the system can attain improved performance, scalability, and fault tolerance. To ensure the security and privacy of user data, it is recommended to utilize blockchain smart contracts for secure data retrieval, access control, and a transparent system for managing data access, thus ensuring that file metadata cannot be tampered with. In order to add another layer of security, the metadata are encrypted before sending to the smart contract so that the information of the file is kept hidden in the blockchain. This guarantees that only authorized individuals can access the file, safeguarding the confidentiality and integrity of the data. Moreover, smart contracts streamline access control procedures, minimizing the chances of human mistakes and unauthorized entry. This approach aligns with cloud security policies, providing a comprehensive solution that satisfies stringent privacy requirements. The incorporation of smart contracts not only strengthens the zero-trust model but also capitalizes on the inherent security features of blockchain technology to protect user data at every stage of its existence.

Zero-knowledge proofs, particularly zero-knowledge succinct non-interactive arguments of knowledge (zk-snarks), provide an effective approach for securely transferring data between cloud services. Zkps enable one party to demonstrate to another that they possess specific knowledge without disclosing the knowledge itself. Zkps are a form of zero knowledge proof. In the realm of cloud data migration, this implies that a user can produce a validation that their data have been correctly and securely decrypted without revealing the actual data to the verifier. Both the verifier and the user can interact with the smart contract to ensure the data integrity of the file. The verifier can only have access to the metadata of the file to check the integrity but he does not have access to the raw data. By adopting this method, the data are safeguarded and kept confidential throughout the migration process. The verification process at the destination cloud can validate the accuracy of the data migration solely based on the proof, ensuring the privacy and security of the user's information. By employing zk-snarks, this approach not only improves security but also simplifies the verification process, making it an efficient and reliable solution for securely moving data to the cloud.

The significant contributions in this paper are as follows:

- We suggest a new file sharding technique that is combined with blockchain smart contracts to improve privacy and security of data.
- To guarantee data integrity and secrecy during storage and retrieval processes, we create a zero-knowledge proof technique.
- We carry out a thorough performance study, showing how effective our strategy is in comparison to other approaches.

2. Related Work

The landscape of cloud storage and information migration has been appreciably explored in recent years, with a strong emphasis on improving security, performance, and consumer privacy [1–32]. A substantial body of studies has focused on the implementation of advanced cryptographic strategies, including encryption and sharding, to protect sensitive data and optimize storage processes. Moreover, the adoption of blockchain-era technology and clever contracts has won traction as a means of offering transparent, tamper-proof control of records as well as obtaining admission to control mechanisms. These innovations aim to cope with the developing worries concerning breaches of facts, unauthorized access, and compliance with stringent regulatory frameworks. In this section the below Table 1 provides the critiques key contributions and findings in the field, highlighting the combination of encryption, sharding, and blockchain technology, and positions our work within the broader context of secure and green cloud storage solutions.

Table 1. Related reference findings in our investigation.

Authors	Citation	Title	Objectives	Findings
M. A. Alshammari, H. Hamdi, M. A. Mahmood, and A. A. A. El-Aziz (2024)	[1]	Cloud Computing Access Control Using Blockchain.	Secure solution for access control in cloud computing environments using blockchain.	By using blockchain technology efficiently, a more secure, scalable, and Transparent access control framework can be implemented.
Dalila Ressi, Riccardo Romanello, Carla Piazza, Sabina Rossi, (2024)	[2]	AI-enhanced blockchain technology: A review of advancements and opportunities.	Integrating AI into Blockchain applications to improve performance like security, consensus, scalability, and interoperability.	The research work highlights that AI-based blockchain provides a better solution for scalability, thereby reducing gas fee.
Ayush Thakur, Sanskar Chauhan, and Ilisha Tomar (2024)	[3]	Self-Healing Nodes with Adaptive Data-Sharding.	Improve Data Storage Efficiency In Cloud Systems.	It is shown that breaking down large datasets into smaller components enhances storage efficiency, scalability, and overall performance.
M. Almasian, A Shafieinejad (2024)	[4]	Secure cloud file sharing scheme using blockchain and attribute-based encryption.	Leveraging blockchain technology for secure access control of the user data.	Using blockchain to implement access control as smart contract, wherein user can request to access his file by logging a transaction in the blockchain.
G. Sucharitha, V. Sitharamulu, S. N. Mohanty, A. Matta, and D Jose (2023)	[10]	Enhancing Secure Communication in the cloud Through Blockchain-Assisted CP-DABE.	Use of encryption to protect sensitive data.	Usage of Blockchain technology for secure key generation, and for access control while the immutability of the blockchain ensures the confidentiality of ciphertext.
D. Dhinakaran, D. Selvaraj, and N. Dharini. (2023)	[11]	Towards A Novel Privacy-Preserving Distributed Multiparty Data Outsourcing Scheme For Cloud Computing With Quantum Key Distribution.	Encryption And Distribution Techniques In Enhancing Data Privacy And Access Speed.	Leveraging encryption technique to store and migrate data from one source to another.
F. Stodt and C. Reich (2023)	[15]	A Review of Digital Wallets and Federated Service for Future of Cloud Services Identity Management.	Utilizing Digital Wallets For Encryption And Key Management.	Digital wallets can play a key role in both identity of the user as well as security of user's data.
W. Alsuwat and H. Alsuwat (2022)	[17]	A Survey on Cloud Storage System Security via Encryption Mechanisms.	Choosing efficient encryption algorithm that is suitable for cloud environment.	Searchable encryption, attribute-based, identity-based encryption, homomorphic encryption and cloud DES algorithms. Each of the above methods has some limitations and disadvantages.
G. C. Jadhav, K. I. Awale, A. A. Patil, and K. N. Rode (2022)	[18]	Cloud Cryptography.	Use of cryptographic algorithm for secure data.	When users upload or store data during cloud service, the data owner does not seem to understand the path of data transfer. Users do not know whether the data is collected, analyzed and accessed by third parties.

Table 1. *Cont.*

Authors	Citation	Title	Objectives	Findings
E. Avstein (2021)	[19]	Zero-Knowledge Cloud Storage: What is it and Why You Need it Now.	Utilization of zkcs, users can securely store data on a remote server without disclosing the actual information.	Employ zk-snarks to create zkps for secure data transmission, ensuring that encrypted information remains hidden.
R. Ragul and R. Arokia Paul Rajan (2020)	[23]	Efficient Horizontal Scaling of Databases Using Data Sharding Technique.	Enhancing cloud data protection using aes and rsa encryption.	Combining data privacy and integrity measures in cloud storage approach aligns with the project's current focus on encryption and secure data migration using zero-knowledge proofs (zkps).
F. Zhang, X. Fan, P. Zhou, and W. Zhou (2020)	[24]	Zero Knowledge Proofs for Cloud Storage Integrity Checking.	Efficient and secure storage for decentralized systems provides valuable insights into zero knowledge proofs for cloud storage integrity checking.	Examine proof-of-replication (porep) to guarantee that storage providers store data in multiple locations, improving security and efficiency.
G. S. Mahmood, D. J. Huang, and B. A. Jaleel (2019)	[25]	A Secure Cloud Computing System by Using Encryption and Access Control Model.	Access control model that can safeguard data in cloud computing.	Employing encryption and access control to guarantee the confidentiality, integrity, and appropriate control of access to sensitive data.
E. K. K. Edris and M. Aiash (2018)	[26]	ZKPVM: A Zero-Knowledge Authentication Protocol for VMs' Live Migration in Mobile Cloud Computing.	Application of ZKPs in both contexts demonstrates their versatility and effectiveness in enhancing security protocols for cloud-based operations.	Employing ZK-SNARKs to generate Zero-Knowledge Proofs, allowing secure and privacy-preserving data migration between cloud services.
C. H. Costa, J. V. B. Moreira Filho, P. H. M. Maia, and F. C. M. B. Oliveira (2015)	[30]	Sharding By Hash Partitioning—A Database Scalability Pattern To Achieve Evenly Sharded Database Clusters	Hashing partition method to increase the scalability of the database.	Efficient scalable and data management in cloud storage and data migration, underscores the significance of sharding in improving performance and reliability in distributed computing applications.
M. P. Patel, M. I. Hasan, and H. D. Vasava (2014)	[31]	Survey Study On Issues In Mongodb In Cloud Environment.	Security enhancement by using encryption, data fragmentation, and distributed storage methods.	Adopting robust encryption techniques and effective data fragmentation methods, which can emphasize the use of the sharding concept and the encryption technique for secure user data storage and sharing.

Extensive analysis of the literature has led us to identify several research gaps. While providing an efficient and secure cloud environment is just as important as user's security and privacy, our architecture provides an efficient storage system using the sharding concept, which plays a huge role in maintaining large files and big data while simultaneously providing security for the data. Using smart contract not only provides secure access control but also stores and secures the metadata of the file, which ensures immutability. Used for secure data transmission from one cloud to another cloud to verify the user data without revealing his actual data, Zk-SNARKS provides an extra layer of privacy and security for this architecture. The servers are built using zero trust architecture where the server handler has no authority over the user's data, thus providing a Zero trust environment.

3. Novelty of the Work

The novelty of this investigation lies in its comprehensive integration of cutting-edge cryptographic techniques and innovative data management strategies to tackle the significant challenges of secure and efficient cloud storage and migration. The design of high-level architecture offers several distinctive contributions to the field. Figure 1 represents the high-level architecture of the secure cloud environment.

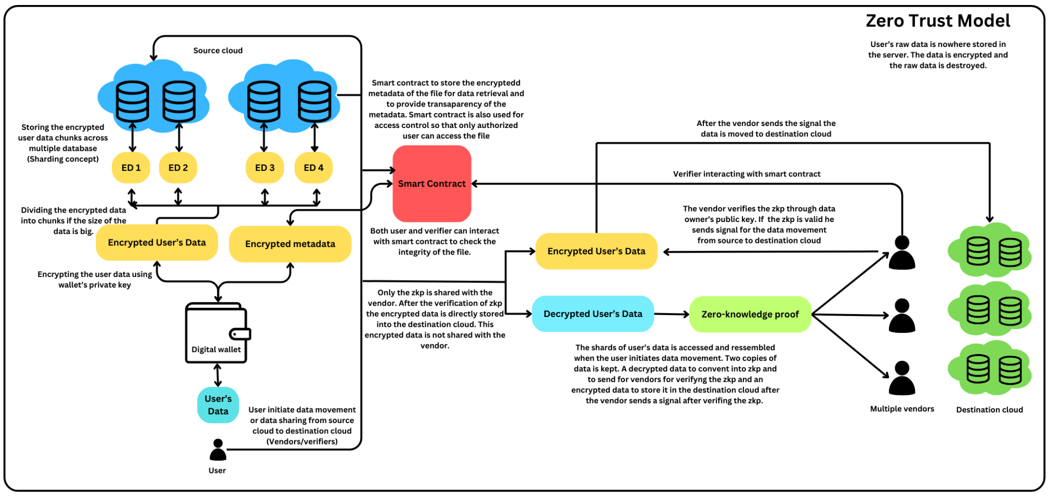
Zero Trust Model
User's raw data is nowhere stored in the server. The data is encrypted and the raw data is destroyed.
Source cloud
Storing the encrypted user data chunks across multiple database (Sharding concept)
ED 1
ED 2
ED 3
ED 4
Dividing the encrypted data into chunks if the size of the data is big.
Encrypted User's Data
Encrypted metadata
Encrypting the user data using wallet's private key
Digital wallet
User's Data
User initiate data movement or data sharing from source cloud to destination cloud (Vendors/verifiers)
User
Smart contract to store the encryptedd metadata of the file for data retrieval and to provide transaparency of the metadata. Smart contract is also used for access control so that only authorized user can access the file
Smart Contract
Both user and verifier can interact with smart contract to check the integrity of the file.
Only the zkp is shared with the vendor. After the verification of zkp the encrypted data is directly stored into the destination cloud. This encrypted data is not shared with the vendor.
After the vendor sends the signal the data is moved to destination cloud
Verifier interacting with smart contract
The vendor verifies the zkp through data owner's public key. If the zkp is valid he sends signal for the data movement from source to destination cloud
Encrypted User's Data
Decrypted User's Data
Zero-knowledge proof
The shards of user's data is accessed and ressembled when the user initiates data movement. Two copies of data is kept. A decrypted data to convent into zkp and to send for vendors for verifyng the zkp and an encrypted data to store it in the destination cloud after the vendor sends a signal after verifing the zkp.
Multiple vendors
Destination cloud

Figure 1. Novel architecture of secure cloud environment.

Encryption and digital wallet integration play a significant role in safeguarding user data privacy, as each file is encrypted using cryptographic keys derived from user-specific digital wallets. This guarantees that data remain protected and accessible exclusively to authorized users, offering a personalized and comprehensive security layer. By utilizing digital wallets for encryption, the system guarantees that each user possesses a distinct encryption key, thereby minimizing the chances of unauthorized access and bolstering overall data security.

4. Secure Cloud Environment for Data Storage and Data Transmission

We provide a comprehensive methodology that combines cutting-edge cryptographic methods, sharding, zero-knowledge proofs (zkps) and smart contracts to guarantee secure and efficient cloud storage and data transfer. The key components of the methodology involve encrypting files using personalized digital wallets, breaking down the encrypted files into shard pieces for effective storage and Improved upload speed, and utilizing zkps for secure data verification in the destination cloud. The system's architecture guarantees the protection of sensitive user data at all stages of its existence, employing a zero-trust approach to prevent any unauthorized access by server handlers.

4.1. Sharding and Encryption Modelling

As soon as a user uploads a file through the client-server interface, the file is instantly encrypted using cryptography keys stored in the user's digital wallet, as illustrated in Figure 2. The encrypted shards are then uploaded to the available servers simultaneously, which ensures faster uploading time for large files or big data, along with the hash of the entire file and additional metadata such as the original file name and format are recorded.

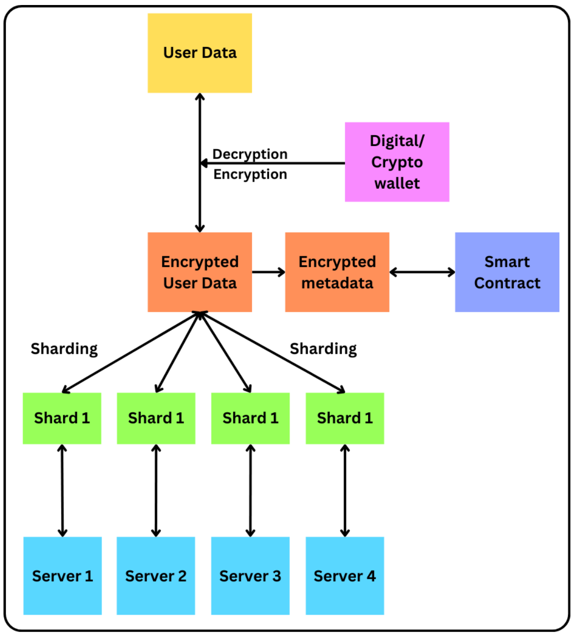

Figure 2. Flow control diagram of sharding and encryption.

4.1.1. Variables and Definitions of the Model

$$\begin{aligned}
F &: \text{The original file as a binary data sequence}\\
|F| &: \text{The size of the original file in bytes}\\
n &: \text{The number of shards (pieces) the file is divided into}\\
P_i &: \text{The } i\text{-th piece (shard) of the file, where } i \in \{1,2,\ldots,n\}\\
|P_i| &: \text{The size of the } i\text{-th piece}\\
K &: \text{The encryption key}\\
IV &: \text{The initialization vector used for encryption}\\
E_{K,IV}(P_i) &: \text{The encrypted version of the i-th piece using the}\\
&\quad\ \text{encryption algorithm with key K and IV}\\
H(x) &: \text{A cryptographic hash function applied to data } x
\end{aligned}$$

4.1.2. Sharding Model

The file F is divided into n pieces, where the size of each piece P_i is ideally equal, but may vary slightly due to file size not being perfectly divisible by n. The size of each piece can be represented as follows:

$$|P_i| = \begin{cases} \left\lfloor \frac{|F|}{n} \right\rfloor + 1 & \text{if } i \leq |F| \mod n \\ \left\lfloor \frac{|F|}{n} \right\rfloor & \text{if } i > |F| \mod n \end{cases}$$

where

- $|F| \mod n$ is the remainder when the file size $|F|$ is divided by the number of pieces n.
- $\lfloor x \rfloor$ denotes the floor function, which rounds down to the nearest integer.

4.1.3. Encryption of Shards

Each shard P_i is encrypted using a symmetric encryption algorithm with a key K and an IV. The encrypted piece E_i is given by

$$E_i = E_{K,IV}(P_i)$$

4.1.4. Hashing and Metadata

The unique report, each shard, and the encrypted shards may be hashed to offer a unique identifier and make certain integrity. The hashes may be denoted as follows:

$$\begin{aligned} H(F) &: \text{Hash of the original file} \\ H(P_i) &: \text{Hash of the } i\text{-th piece before encryption} \\ H(E_i) &: \text{Hash of the } i\text{-th encrypted piece} \end{aligned}$$

4.1.5. Storing Metadata

The metadata stored for the document and its shards include the original document hash, the hashes of the shards, and the encryption information. These metadata ensure that the report can be confirmed and reassembled successfully and these metadata are encrypted and stored in blockchain through a smart contract, thus enhancing the immutable state of the metadata:

$$\text{Metadata} = \{H(F), \text{file\_type}, [H(E_i), i]_{i=1}^{n}, \text{encryption\_algorithm}, \text{K}, \text{IV}\}$$

where

- $[H(E_i), i]_{i=1}^{n}$ represents the list of hashes for all encrypted shards along with their respective indices.
- file_type is the type or format of the original file.
- encryption_algorithm specifies the encryption algorithm used

4.2. Data Migration Using ZK-SNARK

For data migration from the source cloud to the destination cloud, the system utilizes zero-knowledge proofs (zkps) to guarantee secure and private data movement as per Figure 3. The zk-SNARKs construction process includes defining the circuit, which represents the computation verifying the correctness of the reassembly and integrity of the shards. This circuit is then used for generating proving and verifying keys, which are required for the generation of zk-SNARKs. The operations involved in this model contain two phases in order. They start with the proving phase, where the proof of the encrypted file is generated after the decryption of reassembled shards. This stage is followed by the verification phase where the verifier in the destination cloud verifies the zkp data and sends the verification results to proceed to further operations.

4.2.1. Variables and Definitions

$$\begin{aligned} F &: \text{Original file as a binary data sequence} \\ P_i &: \text{The } i\text{-th shard (piece) of the file} \\ E_i &: \text{The } i\text{-th encrypted shard} \\ H(E_i) &: \text{Hash of the } i\text{-th encrypted shard} \\ r_i &: \text{Random value used for the proof} \\ C_i &: \text{Commitment to the } i\text{-th encrypted shard} \\ \text{PK} &: \text{Proving key} \\ \text{VK} &: \text{Verifying key} \\ \pi &: \text{zk-SNARK proof} \end{aligned}$$

4.2.2. zk-SNARKs Construction

The setup phase generates public parameters for proof generation and verification.

- Circuit Definition: Define a circuit C that represents the computation verifying the correctness of the reassembly and integrity of the encrypted shards.
- Public Parameters: Generate proving key (PK) and verifying key (VK):

$$(\text{PK}, \text{VK}) \leftarrow \text{Setup}(C)$$

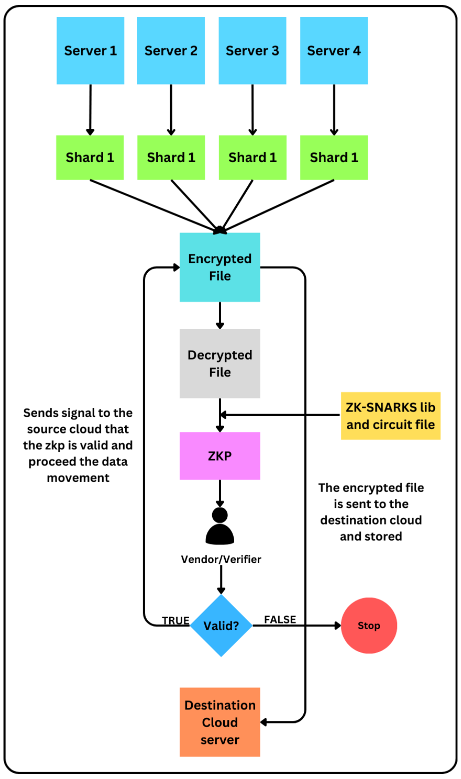
Server 1
Server 2
Server 3
Server 4
Shard 1
Shard 1
Shard 1
Shard 1
Encrypted File
Decrypted File
ZK-SNARKS lib and circuit file
ZKP
Vendor/Verifier
Sends signal to the source cloud that the zkp is valid and proceed the data movement
The encrypted file is sent to the destination cloud and stored
TRUE
Valid?
FALSE
Stop
Destination Cloud server

Figure 3. Flow control diagram of transmission of user data to destination cloud.

4.2.3. Proving Phase

- Inputs and Witness:The public input x includes the commitments C_i and the hash $H(F)$. The witness w consists of E_i and r_i.

$$x = (C_1, C_2, \ldots, C_n, H(F))$$

$$w = (E_1, E_2, \ldots, E_n, r_1, r_2, \ldots, r_n)$$

- Proof Generation: The proof π is generated as follows:

$$\pi \leftarrow \text{Prove}(\text{PK}, x, w)$$

4.2.4. Verification Phase

The verifier checks the proof using the verifying key.

- Verification: Verify the proof against the public input:

$$\text{Verify}(\text{VK}, x, \pi) \rightarrow \text{True}/\text{False}$$

4.2.5. Summary

$$\begin{aligned}(\text{PK}, \text{VK}) &\leftarrow \text{Setup}(C)\\ \pi &\leftarrow \text{Prove}(\text{PK}, x, w)\\ \text{Verify}(\text{VK}, x, \pi) &\rightarrow \text{True}/\text{False}\end{aligned}$$

4.3. Smart Contract for Access Control and Metadata Storage

Storing the encrypted metadata on the blockchain, the system guarantees the integrity and authenticity of the metadata, creating an unalterable record of the data. Furthermore, smart contracts are employed to regulate access control, guaranteeing that only authorized users can access and decrypt the files. By adopting this approach, organizations not only can enhance data security but also adhere to strict cloud policies that prioritize user privacy and data protection.

4.3.1. Variables and Definitions

$$\begin{aligned} \text{Address} &: \text{Unique identifier for a user or entity in the blockchain network} \\ file_i d &: \text{Unique identifier for each file} \\ \text{Metadata} &: \text{Information related to the file, including the hash of the original file} \\ &\quad \text{shard hashes, encryption details, and file type} \\ \text{Access Control} &: \text{Permissions granted to users for accessing or interacting with the file} \end{aligned}$$

4.3.2. Smart Contract Model for Access Control

- **Storing Metadata function:** Refer to Section 4.1.5 to know about Metadata function.

$$\text{StoreMetadata}(file_i d, H(F), \text{file\_type}, \{H(E_i)\}_{i=1}^{n}, \text{encryption\_algorithm}, \text{K}, \text{IV})$$

- **Access Function:** The smart contract function that controls the access of user data:

$$[\text{Access}_{file_i d}](u) = \begin{cases} \text{True} & \text{if user } u \text{ has access} \\ \text{False} & \text{otherwise} \end{cases}$$

  where
  - $file_i d$ is the identifier of the file.
  - u is the user address.
- **Grant Access Function:** The smart contract function for granting access to a user:

$$GrantAccess(file_i d, u) \rightarrow \text{True/False}$$

- **Revoke Access Function:** The smart contract function for revoking access from a user:

$$\text{RevokeAccess}(file_i d, u) \rightarrow \text{True/False}$$

- **Check Access Function:** The smart contract function for checking if a user has access:

$$\text{CheckAccess}(file_i d, u) \rightarrow \text{True/False}$$

4.4. Challenges and Solutions

- **Protecting Data Privacy and Privacy-Aware Method in Blockchain Data Management**
  As discussed earlier, We encrypt metadata prior to its storage on the blockchain as part of our strategy to safeguard data privacy in blockchain systems. Since the user's crypto wallet's private keys are used for this encryption, only the wallet owner is able to decode and view the metadata that has been stored. We improve the privacy of data on the blockchain by integrating this encryption process, which prevents unauthorized parties from viewing or changing the metadata. By protecting sensitive data and utilizing the transparency and immutability of the blockchain, this approach not only preserves the data but also conforms to privacy-preserving strategies and Privacy-Aware Methods such as SymmeProof, BlockShare and VQL.
- **Practical Implementation Challenges**
  A certain hardware and software infrastructure is needed for our system to be implemented in practice, especially for the smooth operation of cryptographic and

blockchain transactions. However, by utilizing cloud-based systems that provide scalable and affordable solutions, these requirements can be lessened. For instance, the demand for specialized hardware can be decreased by leveraging cloud services like AWS or Azure for blockchain nodes and sharding operations. Furthermore, our approach works with open-source blockchain frameworks like Ethereum and Hyperledger, which provide a wealth of tools for developers to create and implement the system.

- **Cost-Effectiveness and Gas Fees for Blockchain Transactions.**
  There are extra expenses associated with using smart contracts and blockchain technology, such as gas prices. But by cutting out pointless calculations and storage activities, we have optimized the design of our smart contracts to use the least amount of gas possible. In addition, implementing the system on layer-2 scaling platforms such as Polygon can considerably reduce gas expenses while preserving the blockchain's advantages in terms of security and decentralization. Although these technologies have initial costs, improved data security, privacy, and verifiability can have long-term benefits that exceed these costs, making the system cost-effective in situations where data integrity is crucial.

5. Experimental Results and Discussion

This investigation aimed to analyze the performance of a cloud storage system by comparing the time it takes to upload files when using sharding versus a system without sharding. The main goal was to assess whether the sharding technique provides a substantial improvement in terms of uploading speed. The experiment entailed transferring substantial files using a client-server interface in two situations: with sharding, where the files were divided into smaller encrypted fragments, and without sharding, where the entire file was uploaded as a single unit.

The performance analysis was conducted in the local environment using Nodejs to calculate the uploading time with and without using the sharding concept and also the time required to generate and verify zkp. The last experiment was conducted in order to calculate the time taken for the migration of encrypted files to the destination server. The method of implementation was conducted by deploying the server in four different ports and one port for the destination cloud (assumed to be cloud storage). The performance was conducted on a laptop with the following specifications: AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz with 16 GB RAM of 64-bit operating system, x64-based processor of Windows 11 Operation system.

The findings of the performance evaluation indicated that the uploading time was noticeably faster when employing the sharding technique. By dividing the encrypted file into smaller fragments, each piece could be processed and uploaded independently and concurrently on multiple servers. The ability to perform multiple tasks simultaneously resulted in a significant reduction in the time it took to upload files. In contrast, the non-sharded approach necessitated the upload of the entire file as a single entity, leading to extended upload times due to the sequential processing and increased resource utilization on a single server.

The below Figure 4 represents the analysis of the uploading time vs file size on a varying number of servers. As we can see, the uploading time gradually decreases as the number of servers increases, and we observe a significant reduction in the time it took to upload files. As the number of servers increases, not only does the efficiency increase, but so does the system's tolerance for faulty nodes or servers.

Through the comparison, we can see the time function gradually decreases on an increasing number of servers n. From the above analysis, we calculated the mathematical equation that can approximately calculate the time taken on n number of servers which was calculated by using the data provided in Table 2.

When $T(n)$ was first formulated, it was thought that the execution time and the number of servers may be inversely related. This assumption, however, ignores the effects of overheads and intrinsic latencies that are not proportional to the number of servers. We add a delay term, L, to the formula in order to handle this. With this change, the execution time $T(n)$ will approach the latency limit L, rather than becoming negligible, as n rises. This gives a more accurate and realistic representation of the system's performance, since it takes into account the fact that some parameters, including network latency and processing overhead, do not change depending on the number of servers.

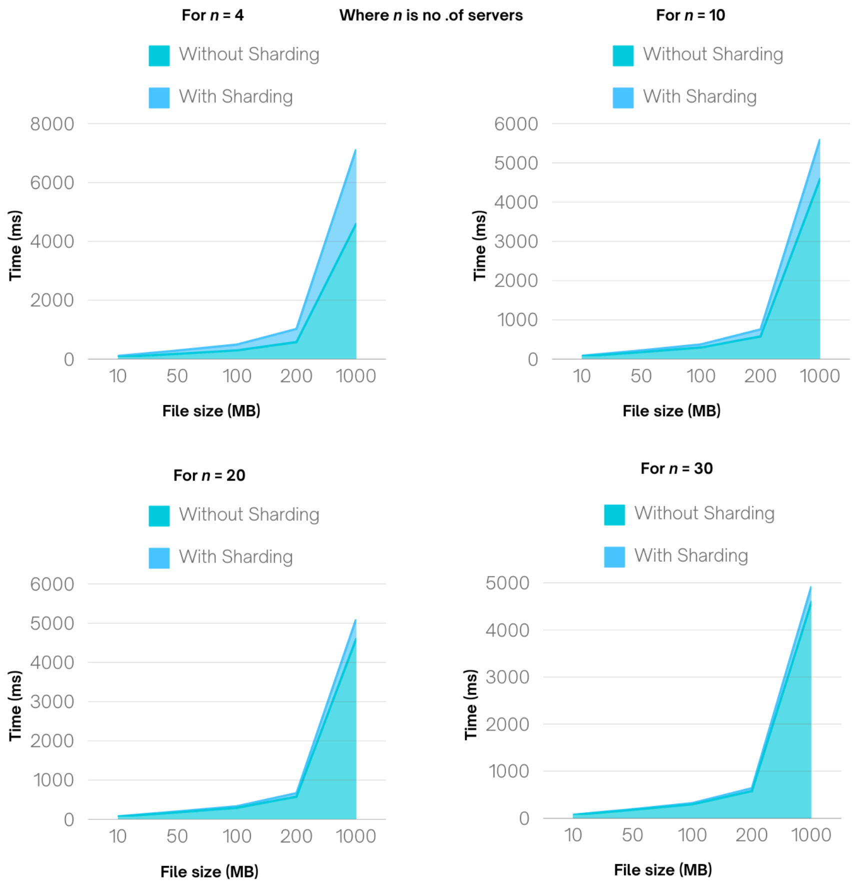

Figure 4. Analysis of time complexity for n no. of servers.

$$T(n) = \frac{T(4) \times 4}{n} + L, \quad \text{where } n_{\min} \leq n \leq n_{\max}, \text{ and } L \geq L_{\min} > 0 \tag{1}$$

where:

- $T(n)$ is the sharding time with n servers.
- $T(4)$ is the sharding time with 4 servers.
- n is the number of servers.
- $n_{\min}$ represents the minimum number of servers required, and $n_{\max}$ is constrained by the system's resources, typically equal to the number of shards

We aimed to apprehend the useful resource performance and value implications of various operations inside the settlement. By categorizing transactions into sorts that include storing metadata, granting and revoking access, checking get admission to permission, and updating metadata, we quantified the fuel intake associated with each. Through our findings, we discovered that operations involving data storage, along with storing and updating metadata, typically consumed the maximum amount of gas because of the higher computational and storage requirements. Conversely, operations like checking get admission to, which by and large involves study operations, consumed substantially less gas. A detailed view is provided in Figure 5. This analysis affords valuable insights into optimizing clever contract design to reduce gas costs, especially for frequent transactions, thereby enhancing the general fee-effectiveness and performance of the system deployed at the Polygon blockchain.

Table 2. Results of Uploading time with and without sharding of data.

File Size (MB)	Without Sharding (ms)	With Sharding, for $n = 4$ (ms)	With Sharding, for $n = 10$ (ms)	With Sharding, for $n = 20$ (ms)	With Sharding, for $n = 30$ (ms)
10	69.5	40.158	16.06	8.03	5.35
50	117.57	112.25	44.9	22.45	14.97
100	295.65	197.109	78.84	39.42	26.28
200	577.76	450.16	180.60	90.03	60.02
1000	4595	2540	1016	508	338.67

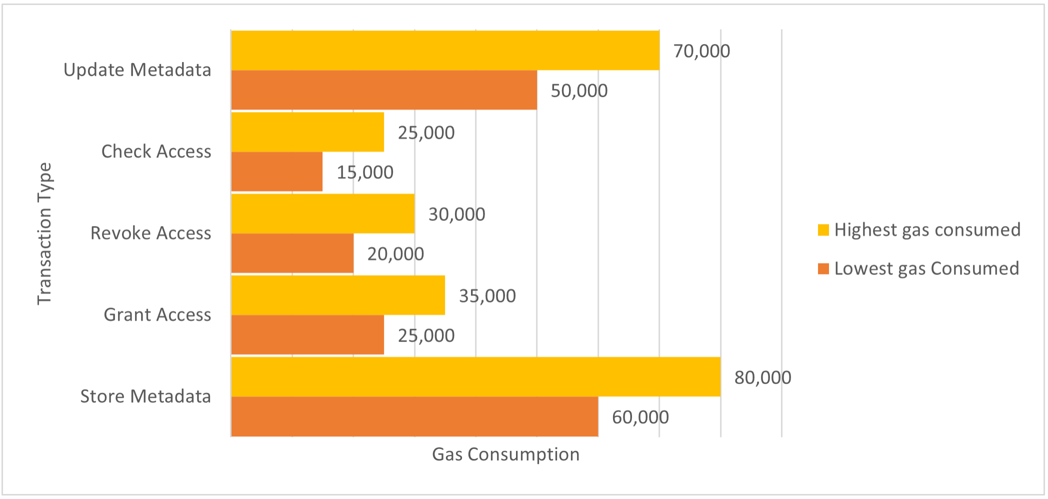

Figure 5. Gas consumption for each transaction type involved in the smart contract.

The gas consumption analysis was conducted on Remix editor, where we can deploy and test our smart contracts. We deployed our smart contract on the polygon network as it provides higher scalability and a lower gas fee. The transaction functions are designed to consume less gas while incorporating secure methods that protect against smart contract vulnerabilities. The yellow line indicates the highest gas consumed for the particular function mentioned and the orange line indicates the lowest amount of gas consumed.

The evaluation of our project's performance included assessing the time required to produce a zero-knowledge proof (zkp) and transmit it to the verifier in the cloud-based destination. The process starts with putting together and decrypting the encrypted file pieces, then creating the zkp using zksnarks. This phase is crucial as it guarantees the accuracy and reliability of the data being transferred without revealing the actual data. The study showed that the zkp generation and transmission to the verifier were executed effectively, requiring an appropriate amount of time. Upon receiving the zkp, the verifier promptly validated it, ensuring a quick and efficient validation process with minimal latency. This rapid validation process is crucial for ensuring a seamless and secure data migration flow, validating that the data integrity checks are thorough yet efficient.

The subsequent performance analysis centered around the time it took for the generation of zkp, as well as the time taken for the actual migration of the file to the destination cloud, provided in the Figure 6. This phase included piecing together the encrypted fragments, ensuring their accuracy, and safely moving the encrypted file to the designated cloud server upon successful verification of Zkp from the destination cloud.

Our analysis revealed that the file uploading process with the sharding concept for big data and large files, file migration process, as well as the zkp validation, were all carried out with remarkable efficiency. We also carried out the efficiency calculating metric using the below formula, which resulted in an increased efficiency of approximately 40.94% for 4 servers, 76.37% for 10 servers, 88.18% for 20 servers and 92.12% for 30 servers when using the sharding concept compared to using a single cloud server. As we can see, as the number of servers increases, the efficiency rate also increases. Thus, the more servers we use, the more file storage and management efficiency increases for big data or large files.

$$\text{Efficiency (\%)} = \left(1 - \frac{\text{Sharding Time}}{\text{Without Sharding Time}}\right) \times 100 \tag{2}$$

By adopting this streamlined approach, the encrypted file can be delivered to the destination promptly and securely, safeguarding the privacy and security of user data throughout the transmission. The integration of efficient zkp validation and optimized file migration highlights the effectiveness of our system in delivering secure and swift data migration services.

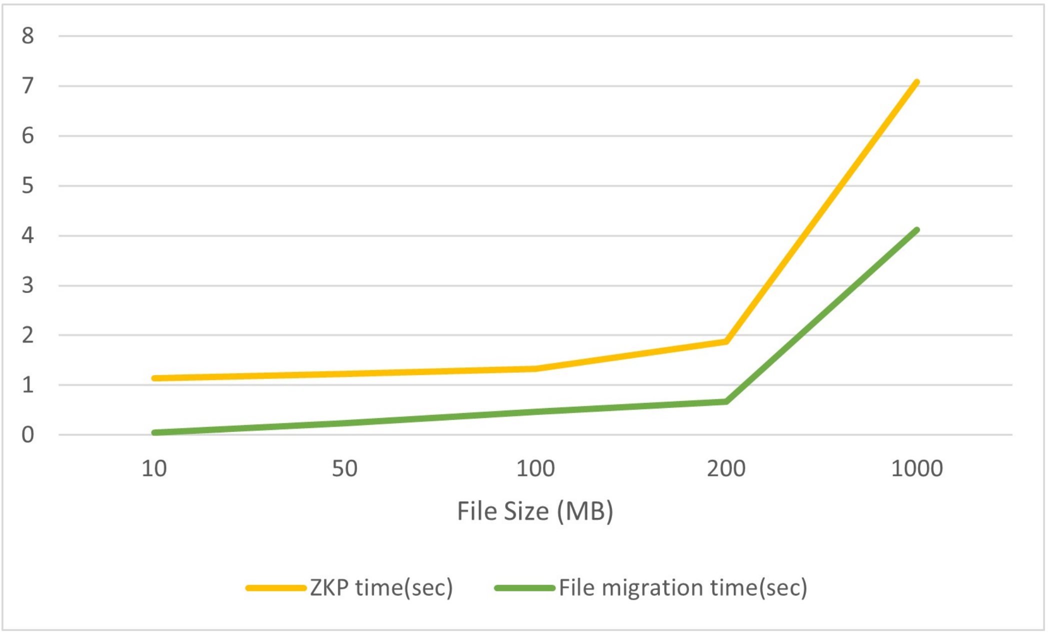

Figure 6. Zkp generation time and file migration time for different file sizes.

We also conducted a detailed analysis by comparing our system with existing systems like Amazon S3, Azure Blob Storage, and Google Cloud Storage. We will need to construct a fictitious dataset based on the features of several cloud storage systems in order to compare the upload performance using sharding versus other current systems. Below is a table that contrasts the upload times for various file sizes using sharding (from our system) with more conventional systems such as Microsoft Azure Blob Storage, Amazon S3, and Google Cloud Storage. An explanation of the analysis conducted in our work is provided in the following Table 3.

Table 3. Comparison of Upload Times Across Different Systems

File Size (MB)	Our System (Sharding, for *n* = 4) (ms)	Amazon S3 (ms)	Google Cloud Storage (ms)	Microsoft Azure Blob Storage (ms)
100 MB	797.109	950	899	912
200 MB	1050.16	1525	1311	1482
1 GB	3140	4231	4018	4211

We have included a latency term, represented as L, in our research to ensure a fair comparison of the upload times of our proposed sharding-based system and current cloud storage options. This delay term takes into consideration a number of variables, including network traffic, processor overhead, and fault tolerance methods, that affect the total upload performance but may vary between systems. A detailed view of this comparsion is provided in Figure 7. By adding L, we make sure that our comparison takes into account both the actual operating conditions of these systems as well as the raw upload speeds.

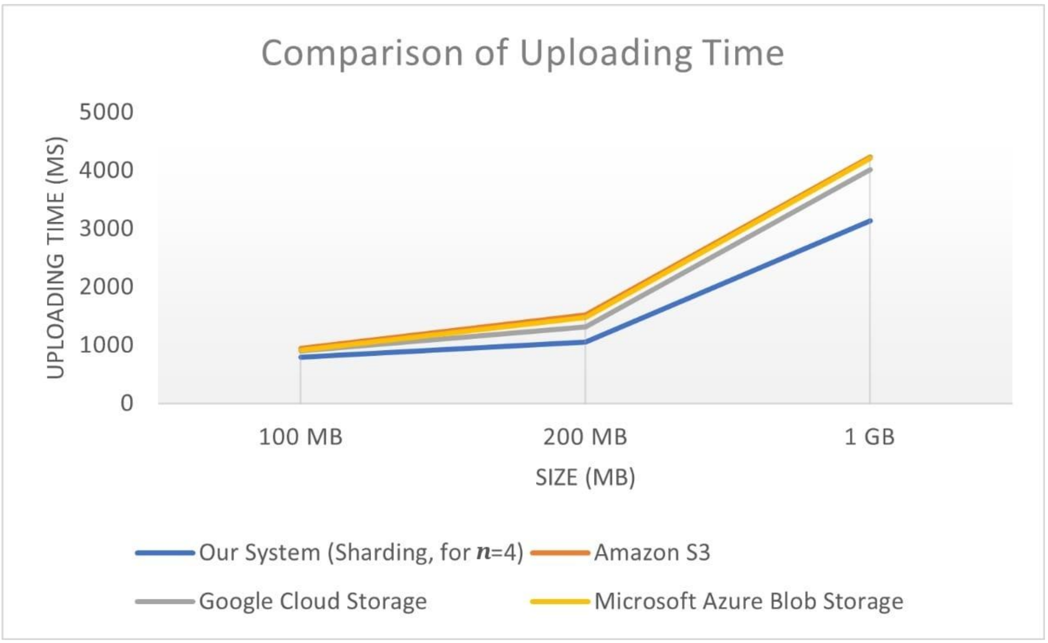

Figure 7. Comparison of our system with existing system.

When discussing the intrinsic architectural and operational differences between our sharding-based system and more conventional cloud storage services like Amazon S3, Google Cloud Storage, and Microsoft Azure Blob Storage, the latency term L is especially crucial. Upload durations can be greatly impacted by higher processing overhead in older systems, for example, from centralized data handling or increased network traffic during peak usage periods. Similar to this, fault tolerance techniques like consistency checks and data replication impose extra delays that are frequently overlooked in straightforward performance comparisons.

Our analysis's L = 600 ms was determined after a thorough review of current systems. This number was selected to provide a reasonable approximation of the cumulative delay impacts seen in most cloud systems. Under modest network traffic and processing loads, studies and benchmarks of Amazon S3, Google Cloud Storage, and Microsoft Azure Blob Storage show that these systems frequently have latencies in the 500–700 ms range. By accounting for comparable network and operating conditions, setting L = 600 ms guarantees that our sharding-based solution is compared on an equal footing with these existing services. This method offers a more realistic depiction of the useful performance advantages of our sharding technology, particularly in situations where user experience is greatly impacted by latency.

This modification demonstrates how well our system can manage the ancillary difficulties caused by network and processing overheads in addition to the main tasks of data sharding and upload. The presence of L indicates that our system still has an advantage over other solutions, especially in circumstances involving large-scale data transfer where upload speed and dependability are critical.

6. Conclusions

In culmination of this novel work, it presents a comprehensive framework that ensures the security and efficiency of cloud storage and data migration, utilizing cutting-edge cryptographic techniques, sharding, blockchain smart contracts and zero-knowledge proofs (zkps). By encrypting files using user-specific digital wallets and dividing the encrypted data across multiple local database servers, the system guarantees that sensitive user information remains secure and inaccessible to unauthorized individuals at all stages of its existence. By implementing a zero-trust model, where encrypted data are stored and server handlers have no direct access to raw data, data security and privacy are greatly improved.

Looking into the future, there are plans to introduce additional improvements to the system, aiming to enhance its capabilities and tackle new challenges in managing cloud data. Initially, the implementation will progress to generate zkps by utilizing cryptographic keys directly obtained from the user's digital wallet. This improvement not only strengthens the security of the proof generation process but also guarantees a personalized and comprehensive verification procedure that is specifically designed to meet the unique encryption key requirements of each individual user.

Further, deploying the smart contracts on a zero-knowledge proof (zkp) blockchain will be examined. This strategy intends to enhance the security and privacy protocols of the cloud storage system. By utilizing zkp blockchain technology, the verification of transactions and data access can be conducted without disclosing any sensitive information, ensuring the privacy of users. Utilizing smart contracts on a zkp blockchain will also improve compliance with privacy regulations and cloud policies, guaranteeing that user data is safeguarded to the highest possible standards. This future direction will play a crucial role in the creation of a more robust, secure, and privacy-focused cloud storage solution.

These advancements are designed to enhance the system's ability to maintain data accuracy, protect privacy, and optimize operational efficiency in cloud environments. Through the incorporation of state-of-the-art technologies and methodologies, this investigation establishes a solid groundwork for future advancements in secure cloud data storage and migration, propelling the development of reliable and scalable cloud computing infrastructures.

Apart from the main contributions of this research, we acknowledge that handling critical user data prior to their storage or transfer in cloud environments is important. As a continuation of this work, we suggest a pre-processing architecture to improve security and privacy in cloud computing. Before data are sent to the cloud, they should be encrypted, anonymized, or secured in some other way to make sure they stay safe, even in the event that the cloud environment is compromised. Future studies could focus on creating more sophisticated encryption methods, using AI to classify data, and creating safe transfer protocols that work well with blockchain and cloud storage systems. Our goal is to develop a complete solution for safe and private data management by tackling these issues in the cloud environment.

Author Contributions: Conceptualization, K.R.; A.K.B.R.; M.P.; S.A.K.; methodology, K.R.; A.K.B.R.; M.P.; S.A.K.; software, K.R.; A.K.B.R.; M.P.; S.A.K.; validation, K.R.; A.K.B.R.; M.P.; S.A.K.; formal analysis, K.R.; A.K.B.R.; M.P.; S.A.K.; investigation, K.R.; A.K.B.R.; M.P.; S.A.K.; resources, K.R.; A.K.B.R.; M.P.; S.A.K.; data curation, K.R.; A.K.B.R.; M.P.; S.A.K.; writing—original draft preparation, K.R.; A.K.B.R.; M.P.; S.A.K.; writing—review and editing, A.K.B.R.; visualization, A.K.B.R.; supervision, A.K.B.R.; project administration, A.K.B.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding authors.

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

References

1. Alshammari, M.A.; Hamdi, H.; Mahmood, M.A.; El-Aziz, A.A.A. Cloud Computing Access Control Using Blockchain. *Int. J. Intell. Syst. Appl. Eng.* **2024**, *12*, 380–390.
2. Ressi, D.; Romanello, R.; Piazza, C.; Rossi, S. AI-enhanced blockchain technology: A review of advancements and opportunities. *J. Netw. Comput. Appl.* **2024**, *225*, 103858. [CrossRef]
3. Thakur, A.; Chauhan, S.; Tomar, I. Self-Healing Nodes with Adaptive Data-Sharding. *arXiv* **2024**, arXiv:2405.00004.
4. Almasian, M.; Shafieinejad, A. Secure cloud file sharing scheme using blockchain and attribute-based encryption. *Comput. Stand. Interface* **2024**, *87*, 103745. [CrossRef]
5. Hamid, I.; Frikha, M. Blockchain-Enhanced Cybersecurity and Privacy in Cloud Computing: A Systematic Literature Review. *J. Theor. Appl. Inf. Technol.* **2024**, *102*, 514–531.
6. Behera, S.; Prathuri, J.R. FPGA-Based Acceleration of K-Nearest Neighbor Algorithm on Fully Homomorphic Encrypted Data. *Cryptography* **2024**, *8*, 8. [CrossRef]
7. Chen, C.; Yang, G.; Li, Z.; Xiao, F.; Chen, Q.; Li, J. Privacy-Preserving Multi-Party Cross-Chain Transaction Protocols. *Cryptography* **2024**, *8*, 6. [CrossRef]
8. Jiang, Y.; Baee, M.A.R.; Simpson, L.R.; Gauravaram, P.; Pieprzyk, J.; Zia, T.; Zhao, Z.; Le, Z. Pervasive User Data Collection from Cyberspace: Privacy Concerns and Countermeasures. *Cryptography* **2024**, *8*, 5. [CrossRef]
9. Bespalov, Y.; Kovalchuk, L.; Nelasa, H.; Oliynykov, R.; Viglione, R. Models for Generation of Proof Forest in zk-SNARK Based Sidechains. *Cryptography* **2023**, *7*, 14. [CrossRef]
10. Sucharitha, G.; Sitharamulu, V.; Mohanty, S.N.; Matta, A.; Jose, D. Enhancing Secure Communication in the Cloud Through Blockchain Assisted-CP-DABE. *IEEE Xplore* **2023**, *11*, 99005–99015. [CrossRef]
11. Dhinakaran, D.; Selvaraj, D.; Dharini, N. Towards A Novel Privacy-Preserving Distributed Multiparty Data Outsourcing Scheme For Cloud Computing With Quantum Key Distribution. *Int. J. Intell. Syst. Appl. Eng.* **2023**, *12*, 286–300.
12. Dubey, H.; Roy, K. Secure Access Control in Cloud Computing Environments: Smart Contract Blockchain. *Vidhyayana* **2023**, *8*, 392–404.
13. Prasad, S.N.; Rekha, C. Block chain based IAS protocol to enhance security and privacy in cloud computing. *Meas. Sens.* **2023**, *28*, 100813. [CrossRef]
14. Jansirani, E.; Kowsalya, R.N. Analysis of ECC and ZKP Based Security Algorithms in Cloud Data. *J. Theor. Appl. Inf. Technol. (JATIT)* **2023**, *101*, 6354–6368.
15. Stodt, F.; Reich, C. A Review on Digital Wallets and Federated Service for Future of Cloud Services Identity Management. In Proceedings of the 15th International Conference on Advanced Service Computing (SERVICE COMPUTATION 2023), Nice, France, 26–30 June 2023; pp. 16–20.
16. Rajguru, S.N.; Choubey, S.K. Blockchain in Cloud Computing for Securing Documents. *Int. Res. J. Mod. Eng. Technol. Sci. (IRJMETS)* **2023**, *5*, 123–130. [CrossRef]
17. Alsuwat, W.; Alsuwat, H. A Survey on Cloud Storage System Security via Encryption Mechanisms. *Int. J. Comput. Sci. Netw. Secur.* **2022**, *22*, 52–61.
18. Jadhav, G.C.; Awale, K.I.; Patil, A.A.; Rode, K.N. Cloud Cryptography. *Int. J. Res. Publ. Rev. (IJRPR)* **2022**, *3*, 2200–2202.
19. Avstein, E. Zero-Knowledge Cloud Storage: What is it and Why You Need it Now. *Codemot. Mag.* **2021**. [CrossRef]
20. Mandal, S.; Khan, D.A.; Jain, S. Cloud-Based Zero Trust Access Control Policy: An Approach to Support Work-from-Home Driven by COVID-19 Pandemic. *New Gener. Comput.* **2021**, *39*, 599–622. [CrossRef]
21. Ghosh, P. The State-of-the-Art in Zero-Knowledge Authentication Proof for Cloud. In *Machine Learning Techniques and Analytics for Cloud Security*, 1st ed.; Wiley: Hoboken, NJ, USA, 2021; pp. 149–170. [CrossRef]
22. Hamid, I.; Frikha, M. A Review on Cryptography in Cloud Computing. *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.* **2024**, *6*, 225–230.
23. Ragul, R.; Rajan, R.A.P. Efficient Horizontal Scaling of Databases Using Data Sharding Technique. *Int. J. Innov. Technol. Explor. Eng. (IJITEE)* **2020**, *9*, 590–593. [CrossRef]
24. Zhang, F.; Fan, X.; Zhou, P.; Zhou, W. Zero Knowledge Proofs for Cloud Storage Integrity Checking. *arXiv* **2019**, arXiv:1912.00446.
25. Mahmood, G.S.; Huang, D.J.; Jaleel, B.A. A Secure Cloud Computing System by Using Encryption and Access Control Model. *J. Inf. Process. Syst.* **2019**, *15*, 538–549. [CrossRef]
26. Edris, E.K.K.; Aiash, M. ZKPVM: A Zero-Knowledge Authentication Protocol for VMs' Live Migration in Mobile Cloud Computing. In Proceedings of the 13th International Conference on Software Technologies (ICSOFT), Porto, Portugal, 26–28 July 2018; pp. 858–864.
27. Shaik, A.; Madhurima, B.; Neelakantappa, M. An Approach To Zero Knowledge Proof For Secure Data Sharing in Cloud Storage: New Direction. *Int. J. Innov. Technol. Explor. Eng. (IJITEE)* **2018**, *8*, 195–201.
28. Jain, T.; Khan, J.A. Secure Big Data Access Control Policies for Cloud Computing Environment. *Int. J. Innov. Res. Comput. Sci. Technol. (IJIRCST)* **2017**, *5*, 254–256. [CrossRef]
29. Bagui, S.; Nguyen, L.T. Database Sharding: To Provide Fault Tolerance and Scalability of Big Data on the Cloud. *Int. J. Cloud Appl. Comput. (IJCAC)* **2015**, *5*, 36–52. [CrossRef]

30. Costa, C.H.; Filho, J.V.B.M.; Maia, P.H.M.; Oliveira, F.C.M.B. Sharding By Hash Partitioning—A Database Scalability Pattern To Achieve Evenly Sharded Database Clusters. In Proceedings of the 17th International Conference on Enterprise Information Systems (ICEIS), Barcelona, Spain, 27–30 April 2015; Volume 2, pp. 313–320. [CrossRef]
31. Patel, M.P.; Hasan, M.I.; Vasava, H.D. Survey Study On Issues In Mongodb In Cloud Environment. *Int. J. Adv. Innov. Res.* **2014**, *3*, 18–21.
32. Balasubramaniam, S.; Kavitha, V. A survey on data encryption tecniques in cloud computing. *Asian J. Inf. Technol.* **2014**, *13*, 494–505.