J. Adv. Res. in Cloud Computing, Virtualization and Web Applications
Volume 7, Print Issue 1 - 2024, Pg. No. 8-16
Peer Reviewed Journal

Review Article

A Review: Cloud Computing and Blockchain Integration

Nisha Arora[1], Kapil Parashar[2]

[1,2]PCTE Group of Institutes, Baddowal.

I N F O

E-mail Id:
nishaarora@pcte.edu.in
Orchid Id:
https://orcid.org/0009-0002-1603-3387

How to cite this article:
Arora N, Parashar K. A Review: Cloud Computing and Blockchain Integration. *J Adv Res Cloud Comp Virtu Web Appl* 2024; 7(1): 8-16.

Date of Submission: 2024-03-17
Date of Acceptance: 2024-04-20

A B S T R A C T

Blockchain technology is one of the backbone technologies used in crypto-currency that has received a lot of attention in the last decade and act as a necessary technology behind like Bitcoin, which is a popular digital Cryptocurrency. Blockchain technology act as a distributed ledger with records of transactions containing all the data details of the transactions carried out and it will be distributed among the nodes present in the entire network. All the transactions carried out in the system are confirmed by mechanisms, and the data once stored cannot be altered or modified. On the other hand, “Cloud computing is a practice of using a network of remote servers hosted on the internet for the purpose of using to compute, storage, and managing the data, rather than on a local server or a personal computer”. Making a single system by combining both the concepts of cloud computing and blockchain technology that can improve the efficiency of network control, task scheduling, data integrity, resource management, fair pricing, payment, and resource allocation in the day-to-day activities. In this review article, we have mentioned some of the significant opportunities and challenges faced by the cloud and proposed their solutions by integrating it with blockchain technology to enhance the ability. We tried to investigate a brief survey on earlier studies focused on cloud integrating with the blockchain technology. In this, we have also developed architecture integrating blockchain with cloud revealing the communication between blockchain and cloud.

Keywords: Blockchain Technology, Cloud Computing, Cryptocurrency, Homomorphic Encryption

Introduction

Cloud computing clients in various application domains want to be assured that their data is accurate and trustworthy. On the other hand, blockchain is a tamper-proof digital ledger that can be used alongside cloud technology to provide a tamper-proof cloud computing environment. This paper proposes a scheme that combines cloud computing with blockchain that assures data integrity for all homomorphic encryption schemes. Given its widespread accessibility, cloud services are vulnerable to attacks. Data manipulation is a serious threat to data integrity that can occur in cloud computing, a relatively new offering under the umbrella of cloud services. In order to reduce the dangers connected with cloud computing, the Cloud Security Alliance (CSA) has outlined crucial shared obligations for cloud service providers (CSPs) and their clients. Yu-Chi Chen, an associate editor, oversaw the manuscript’s review process and gave the go-ahead for publishing.

Recording, designing, and implementing internal and client security controls are the responsibilities of CSP. The Consensus Assessments Initiative Questionnaire (CAIQ) is a tool used in the design and implementation process. A Cloud Control Matrix (CCM) is a tool used by cloud consumers to record the individuals responsible for putting particular controls into place as well as the methods they employ. In order to account for the considerable differences in the process model that are anticipated to arise during the development of a cloud project, a high-level process model for cloud security management has also been established. The key is to ascertain the prerequisites, organise the architecture, and identify any gaps in relation to the capabilities of the underlying cloud platform.

The top 11 threats were categorised into 14 security domains, which are further divided into the governance and operational domains, by a recent CSA poll that collated the most important security challenges pertaining to cloud computing. While the operational domain is more concerned with tactical security issues, the governance domain concentrates on strategic and policy issues within a cloud computing environment. The majority of security issues give rise to several kinds of threats, such as denial-of-service attacks, data manipulation, information leakage, spoofing, and elevation of privilege. An unauthorised entity releasing, analysing, stealing, or using vital, secure, or secret information is said to have committed a data breach. A data breach may arise due to human error, implementation weaknesses, insufficient security protocols, or as the primary objective of a targeted attack. A data breach occurs when any information that was not intended for public consumption is disclosed.

More specifically, encryption and key-related issues that affect data secrecy and completeness can be caused by inadequate key management systems and inappropriate encryption algorithms. Confidentiality, integrity, and availability are essential components of cloud security, just like they are for any information security management system. Confidentiality and privacy are directly tied to the issue of data breaches. While privacy refers to a client's right to determine how their data is treated, confidentiality demands that sensitive client data not be shared to any unapproved organization. Encryption algorithms are employed to meet both data confidentiality and privacy needs. Several cryptographic approaches have been put forth to maintain the security of processed and/or stored data. On cloud computing systems, several symmetric key encryption techniques have been implemented. By putting out a verification scheme founded on the ideas of BFT and blockchain technology, we solve these current issues. It will be necessary to employ multiple CSPs to store and process client data. Each CSP that wants to store their database on a public blockchain like Ethereum or Bitcoin will need to calculate the master hash value of their database on a regular basis. These CSPs do not need to collaborate or communicate with one another. To find out if there has been data tampering, a client can compare these master hash values. This distributed verification mechanism compares master hash values maintained on the blockchain to ensure integrity and confidentiality (HE will be used for encryption).

Homomorphic Encryption (HE)

The process of converting data into ciphertext so that it can be used for operations on encrypted data without gaining access to the private decryption key; the private key should only be held by the data owner.

In the process of applying arithmetic operations to encrypted data, the same results should be gotten as in the case of unprocessed data. The data owner generates the public-key pair (a public key puk and a private key prk) during the first step of the HE process, known as key generation (KeyGen). The data C = Encpuk (P) is encrypted using the encryption algorithm in the next step, the encryption process Enc, before it is sent to the cloud server. The encrypted data and the puk are kept in a database on the cloud server. The cloud server uses the encrypted data to carry out the specified calculation and then returns the encrypted result to the client upon request. This is referred to as the Eval, or assessment process. The client can process Dec, the decryption function, and retrieve the plaintext by using the matching prk. To sum up, KeyGen, Enc, Eval, and Dec are the four primary processes of HE. Even while homomorphic cryptosystems have advantages, not all designs are IND-CCA2 secure because to their malleability.

This may result in inaccurate calculations that are outsourced. It is noteworthy that same issues may arise in the absence of decryption. CSP still has the ability to undermine data integrity and do it covertly. For instance, without being aware of the contents of the substituted data, the CSP is able to implicitly replace a given ciphertext or the cumulative result with other legitimate ciphertexts. There is no way to get back the original data after integrity is lost. Therefore, data integrity needs to be enforced on such outsourced computations. Furthermore, a CSP is regarded as a third party that a client hires to handle intricate calculations.

Data integrity is further threatened by the fact that these computations are centralized and that anyone has the power to alter data. Because Secure Shell (SSH) lacks authentication functionality, using it to overcome these problems is not possible. In order to create a safe CSP platform, a strong, impenetrable, and verifiable security architecture is required in addition to homomorphic data encryption. These are the characteristics of the BC architecture, which consists of a peer-to-peer network working to validate blocks managing a distributed ledger (or database) of aggregated transactions. The architecture of CSP and BC technologies appears to be at odds with one another—centralization vs decentralization. Nonetheless, CSP and BC can work in concert to provide a single solution that maximizes each of their advantages.

Blockchain Technology (BC)

The use of BC approaches in cloud environments has garnered a lot of interest from academic and industrial sectors. Fundamentally, BC technology is made up of dispersed digital blocks connected to one another by cryptographic rules. Each block has transaction data, a timestamp, and a cryptographic hash of the block before it. By using a peer-to-peer network, BC allows all users to independently authenticate transactions. In order to guarantee that all nodes in the network agree, a consensus mechanism is needed to approve and record transactions in the BC. A block that has been validated cannot be changed later without also changing any blocks that come after it.

Each block has transaction data, a timestamp, and a cryptographic hash of the block before it. By using a peer-to-peer network, BC allows all users to independently authenticate transactions. In order to guarantee that all nodes in the network agree, a consensus mechanism is needed to approve and record transactions in the BC. A block that has been validated cannot be changed later without also changing any blocks that come after it. The distributed digital ledgers that are organised into blocks and contain transactions that are signed cryptographically. Each block is validated and goes through a consensus decision process before being cryptographically linked to the preceding one, making it tamper obvious. Older blocks become harder to change when new ones are introduced, resulting in tamper resistance. Within the network, new blocks are replicated across copies of the ledger, and any conflicts are automatically resolved by applying predefined rules. Many companies are creating cloud-based BCs in response to the growing interest in BC technologies. Based on the Software as a Service (SaaS) model, well-known CSPs have offered Blockchain as a Service (BaaS) to their clients. Using open-source software platforms like Ethereum and Hyperledger Fabric, which enable developers to produce and share information, Amazon Managed Blockchain was introduced. Because BC technology's architecture makes use of well-known computer science procedures, cryptographic primitives, and record-keeping concepts, it is also known as "trust machines." A cryptographic hash function, which is used for address derivation, unique identity creation, block data security, and block header security, is the primary component of a BC network. Three primary characteristics are taken into consideration while designing hash functions: collision resistance, second preimage resistance, and preimage resistance. Different standards for NIST-approved hash functions were set under the Federal Information Processing Standard (FIPS) . CPUs like Intel have specific instruction sets that enable hardware acceleration of the SHA family, which improves computation efficiency.

The output of SHA-2 is 32 bytes (1 byte equals 8 bits, 32 bytes equals 256 bits), and it is typically shown as a 64-character hexadecimal string. Certain proof-of-work (PoW) consensus algorithms also employ hash functions like SHA-2. A BC's core component is its consensus process or mechanism, which is used to choose which nodes will publish new blocks.

BC technology can save businesses time and money while also fostering better justice and transparency. The market is home to a wide range of BC-based applications involving several industries, including supply chain, business, healthcare, cybersecurity, cryptocurrency, and the Internet of Things.[46] Our suggested project makes use of cryptocurrencies, which are among the first and undoubtedly the most well-known applications of BC technology. We will be employing Ethereum and Bitcoin, two of the most well-known cryptocurrencies available right now.

Bitcoin

The first and most well-known cryptocurrency on the market was Bitcoin (BTC), which was introduced by Nakamoto in 2008.[48] Bitcoins (BTCs) are earned as rewards for solving the PoW puzzle through mining, and they can be moved between Bitcoin accounts. Every transfer is documented as a transaction that is kept on the BC in a block. The identical copy of the Bitcoin BC is stored on every node that is involved. Block leaders are chosen from among the nodes that successfully compute the PoW to build, announce, and append a new block to the BC. Other nodes will accept the new block and add it to their own copies of the BC if all of the transactions in it are legitimate.

Digital wallets are necessary for cryptocurrencies to handle key pairs and enable transactions. The primary functions of the Bitcoin wallet are to compute public addresses and hold the private key needed to redeem Bitcoin. BTCs are not physically kept in the wallet from a technological standpoint. Rather, they are located on the BC and are only accessible by those who with the appropriate private keys. Transactions can also be "signed" with the use of private keys. All of the main operating systems and apps are compatible with free Bitcoin wallets, which are made to meet a range of user needs. Numerous systems provide a wide range of wallet options. Although they all have some comparable qualities, each wallet has its own unique features. A shared wallet referred to as the multi-signature wallet, or multisig wallet, is one of the most beneficial features. A minimum of one key is necessary to approve a Bitcoin transaction in a multisig wallet, which can be accessed by two or more keys. Apart from typical transactions, which are signed by a single private key owner, several signatures are needed before the money is transmitted. Since it limits what may be done with Bitcoin, it is more secure. To be more precise, in the event that one of the wallets is compromised, the hacker will be unable to spend Bitcoin from the shared wallet without permission from the other owners. Additionally, it prevents other parties from controlling purchases in a community and tracks involved parties by accessing a single wallet's transaction history.

Ethereum

Ethereum is a network of separate computers that work as a single supercomputer rather than merely a cryptocurrency network. It is adaptable, enabling transactions to be established over networks with or without permission. Since it's a BC-based platform for smart contract execution, it offers support for more than simply bitcoin transactions.

The Ethereum Virtual Machine is the name of this platform (EVM). To run on the EVM, each smart contract is compiled into a unique bytecode. A smart contract can define any kind of rule or functionality because the Ethereum platform is Turing-complete. Externally owned accounts (EOA) and contract accounts (CA) are the two primary account kinds in Ethereum. Unlike CAs, which contain accompanying code, EOAs are managed by private keys and lack it. Although in distinct ways, these accounts are able to speak with those who are identical to them as well as with each other. Ether is the name of the currency that peers in the Ethereum network exchange among themselves.

An Ethereum wallet can be either a standard wallet (like a Bitcoin wallet) or a smart contract wallet that can use the Solidity programming language to create, execute, or trigger smart contracts in a CA, depending on the kind of account. Simple wallets and multisig wallets are the two types of wallets that contracts can implement. A multisig wallet contains many owner accounts, including the creator's account, in contrast to a simple wallet, which typically only has one account that manages and owns the wallet.

Proposal Design

The technologies that underpin the verified computation design are BC and CSP, both of which are extremely important. Before introducing our suggested design, we first go over the client verification procedure of operations applied to the requested data. To accomplish immutability, we start by using several paths for computations and then store the results on the BC. The multi-CSPs, BC-application, and client—whose responsibilities are described below—will make up the three primary parts of the verification process, which align with the three main stages of the suggested verification scheme:-

Multi-CSPs: A customer may use multiple CSPs.

Although each formed CSP and the customer have separate contracts, all are bound by the same conditions. The hired CSPs will carry out the calculations, and after they are finished, they will create a master hash for their database and send the outcome to the BC-based application.

Application based on BC: generates new blocks including the master hashes as a transaction and sends the block header back to CSP.

Client: Using the block header data they have received, clients can verify by comparing the master hash values from each CSP.

The frequency of computing master hash values (defined by a frequency variable, t) and the associated cryptocurrency wallet are the two primary factors that govern the design workflow, and the client should ascertain these before going through a full explanation of each step. The number of calculations a client requests before several CSPs are needed to calculate the master hash of their respective databases is determined by t. T's value is determined by two primary components. The client's data growth % is the first, and his capacity to pay the BC transaction costs is the second.

CSP - Computation Phase

We take some of BC's BFT consensus features and apply them to verification in the CSP environment. The distributed ledger's characteristics and hash algorithms will both be used in the proposed study. The proposal adopts the idea of using numerous nodes to create a new block, allowing the client to obtain support from various CSPs rather than just one. The BFT scenario is used to calculate the number of CSPs that are hired. When there are two f + 1 CSPs in a system and f of them are Byzantine (or malicious), the malicious CSPs work together to say random things to the other f + 1 nodes. As an illustration, a system is trying to agree on the result of the calculation (x). $N = 2f + 1 = 3$ can be used to determine the number of CSPs if $f = 1$. CSP-A, CSP-B, and CSP-C are the designations for the three CSPs, respectively. Figure 3 shows how CSP-C can stop all three CSPs from reaching a consensus, assuming that CSP-C is Byzantine. While notifying CSP-B that its outcome is x, CSP-C notifies CSP-A that its outcome is y. Because this is the conclusion with the most votes, CSP-A and CSP-B both accept the outcomes y and x, respectively, because the results from CSP-C correspond to both of their own results. Consequently, to tolerate one Byzantine node, $N = 4$ CSPs must be met as a minimum. The analysis of the suggested work will be done assuming that at least $N = 4$ CSPs exist. The points that follow summarize the tasks t at a single CSP will complete are:

1. Determine the database's master hash value using

SHA-2 after t times of specified calculations.

2. The master hash is stored by the CSP in a transaction log, after which it is sent to the mining pool for storage inside the BC network.

The suggested work depends on the absence of direct communication between the various CSPs in order to prevent 51% assaults. This will actually be the case because these CSPs can come from various businesses or organizations. In the event that the CSPs are able to establish direct communication with one another, the presence of three malevolent CSPs may result in peer consensus over inaccurate data and transaction documentation that uses the incorrect master hash. Rather, the BC assists in achieving authentication.

Blockchain - Master Hash Phase

We take into consideration two well-known cryptocurrencies, Ethereum and Bitcoin, to store the master hash values. The transfer of money from the CSP's wallet to the network is the identical for both Bitcoin and Ethereum, despite their distinct structures and features. Detailed explanations of each of these processes will be given.

Bitcoin

Step I - Setting up BTC wallet: One multi-signature wallet will be provided to the client for each of the four CSPs who were hired. Each wallet will generate three signature keys, of which at least two thirds must be used in order to conduct Bitcoin transactions. The client has access to two keys, whereas CSP is in possession of the third key. The client's keys must be kept in two separate places: one is in the client's wallet, and the other is kept as a backup recovery key in a secure location. As a result, the client possesses the bulk of the keys, and the second key is still functional even if the first key is misplaced. For the suggested method to be used, wallets must contain a minimum of 546 Satoshis, or 0.00000546 BTC, after they have been enabled.

Step 2 - Prepare raw transaction and embed master hash: Assuming that the shared wallets have already been created and the master hash is prepared for storing in the Bitcoin BC, this step can be carried out. The following are the specific actions to prepare the raw transaction and incorporate the master hash value:

1. **Create multisig transaction address:** Every CSP and client must generate a distinct pair of cryptographic keys, comprising a public key and matching private key. The two public keys will be used to create a multisig address.

To establish two distinct addresses—one for the input and the other for the output— this process is repeated twice. This enables the exchange of Bitcoins between two distinct addresses within a single wallet.

2. **Creating raw transaction and writing master hash in transaction data:** A set of information describing a Bitcoin transaction is called a transaction. The suggested work will only modify the ScriptPubKey data output and adhere to the standard transaction data structure.

The purpose of the Bitcoin network is to record financial transactions, not to hold random data. On the other hand, programmers have devised a variety of methods based on several standard scripts for encoding data in transactions. The two kinds of scripts that we are interested in are the NULL DATA script and the Pay To Pubkey Hash (P2PKH) script. The programmer can store any kind of data where the hashed public key should be in the first scheme (the P2PKH script). This indicates that there are just 160 bits available for data encoding. If the master hash was computed using SHA-1 rather than SHA-2, the CSP could implement this scheme. However, because it negatively affects users' RAM, this strategy raises issues with performance as well as security (shorter hash values can result in attacks). The inability to quickly separate the output from the typical locking script is the root cause of the efficiency issues. The standard script NULL DATA, which enables the pushing of metadata onto the BC, is the foundation of the second approach. The concept operates by appending a NULL DATA lock script to an extra output. The security and efficiency issues of the first scheme are resolved by this script. With Bitcoin Core version 0.12.0, 83 bytes of metadata can be stored at most. Therefore, there is no need to save information in the UXTO database, freeing up RAM. For a transaction to be approved as a regular transaction, it can only have one NULL DATA locking script.

We will be using this method since the suggested strategy uses SHA-2 to produce master hash values.

Step 3 - Signing transaction and broadcasting to the Bitcoin network: Spending from a multisig address requires the client and CSP to sign the encrypted transaction using their respective private keys. The encoded transaction is then broadcast to the network by the CSP, where miners gather it and incorporate it into blocks, which are then added to the Bitcoin BC once the PoW is generated.

Ethereum

Externally Owned Accounts

Step 1: Configuring an Ethereum wallet Standard Ethereum wallets provide a public Ethereum address for user accounts in addition to storing private keys. To perform an ETH transaction, each of the hired CSPs has to get a normal wallet. Stated differently, the CSPs must be light nodes. However, EOA does not support multi-signatures.

Step 2: Get the transaction ready and embed the master hash.

Step 3 - Signing transaction and broadcasting to Ethereum network: A transaction must be signed by the initiator account's private key in order to be published on the Ethereum network. The executed deal is delivered to the local Ethereum node, which verifies that the signed transaction was actually signed by the address associated with this account. The signed transaction is broadcast to every peer in the network at the end of the process.

Smart Contract Accounts

Step 1 - Setting up ETH wallet: An Ethereum smart contract that is used to store ETH that is owned by numerous parties is called a multisig wallet. A certain number of shareholders must authorise each transaction. The client will roll out four shared smart wallets—one for each CSP—as part of the proposed plan. Each shared smart wallet will have three owner accounts: the CSP and two client accounts.

Applying the two-thirds rule allows a transaction to be approved. ETH can be transmitted to this wallet, just like any other Ethereum address, once the shared smart wallet between the client and CSPs has been launched.

Step2 - Prepare raw transaction and embed master hash: Transactions are used in the Ethereum network to install smart contracts. Although the data included in the transaction is different from an EOA, the transaction structure is the same.[56] The bytecode and any encoded arguments that a constructor may require should be included in the input data. The CSP will use submit_Transaction to place an ETH exchange order in order to deploy a multisig contract to the Ethereum network. If there is enough ETH in the wallet, this can be done. In response, CSP receives a transaction_Id or hash code. This transaction_Id will be made available to clients by the CSP so they can verify it. The client can use the transaction_Id to verify the transaction data after receiving it from CSP.

Step 3 - Signing transaction and broadcasting to Ethereum network: All accounts in an Ethereum network follow the same procedure in sending out the transactions: signing the transactions with the private key and broadcasting transaction to local nodes which are responsible for validating and redistributing the transaction to their own peers.

Client Phase – Verification

It is now the client's responsibility to confirm that the values supplied by each CSP are the same after saving the master hash in the BC. Since the platforms utilised in this research vary in their verification requirements, this procedure will be covered individually for the Bitcoin and Ethereum models.

Bitcoin Verification

If the client chooses the Bitcoin platform, verifying the master hash values is straightforward. This is because each hired CSP has multisig wallets; in order to transact BTC, the client must first consent. The client can view the stored data value inside the transaction when they are required to consent to a BTC transaction.

However, especially if the data is little, it makes no sense for the client to keep checking it in this manner. Therefore, the relevant block headers of the blocks containing these transactions can be resorted to at any moment if the client wishes to confirm the hash values. A four-byte long timestamp in the block header shows when the block was added to the BC.

Ethereum Verification

The Ethereum verification process varies based on the kind of account being utilised. Every CSP for EOAs must transmit the block header to the client for every transaction. This enables the customer to monitor every transaction and carry out validation. Regarding CAs, the multisig wallets contain the block headers linked to the CSPs' transactions that the client can access. Therefore, the timestamp information in each block header can be used to conduct the verification procedure. The ability of the CA- based method to set up a shared wallet, which speeds up the data verification procedure, makes it superior than EOA.

Result and Discussion

This section presents a theoretical assessment of the suggested scheme's performance and implementation costs using each model connected to BC. These computations are meant to identify the model with the best online performance and the most viable financial solution. In our computations, we do not account for the cost of hiring the CSPs or the hash function computation time.

Cost Analysis

Bitcoin-Based Cost Analysis

The suggested plan will only rely on the smallest BTC trading quantity feasible, taking into account network needs and transaction fees, due to the extreme high price of BTC.

Early in 2020, information was released indicating that the average transaction cost is 0.00001 BTC, or roughly $0.09.

Generally speaking, the minimum BTC trading amount depends on the client's method of obtaining BTC (such as through a cryptocurrency exchange).

Assume for the purposes of argument that the client selects a Luna that enables users to purchase Bitcoin for 0.75 dollars. The transaction costs are quite similar to the amount of Bitcoin that is being held; eventually, the fee for redeeming a new transaction will exceed the amount that the client is transferring. Transactions with outputs less than a predetermined size will automatically be deleted due to dust regulations. The effectiveness of our approach will suffer if any of these issues arise. To put it more succinctly, it slows down the chaining of the transaction in a block, which can lead to notable delays in the time intervals between established blocks for each CSP or even the inability of one of the CSPs to block a transaction.

Ethereum-Based Cost Analysis

One way to store data in the data storage of a smart contract is to store random data as a variable. The cost of storing data depends on how many SSTORE operations are performed. One SSTORE transaction—costing 20,000 gas—is needed to transform data from zero to non-zero in order to store a master hash value of 32 bytes. As was already established, a transporter must pay at least 21,000 petrol for each operation. The operation's data payload, which consists of the actual data and function signature, results in additional gas usage. The process of creating the smart contract itself also has an additional expense.

An estimated 0.010 USD will be spent overall (20, 000 + 21, 000 + 32 68 petrol). Preserving data as a log event is the second choice. Numerous factors must be taken into account in order to calculate cost.

First, each byte of data in a log topic costs an extra 8 gas, and logged data is preserved in log topics at a cost of 375 gas. Using a log event to store 32 bytes of data is roughly estimated to cost 0.005 USD (21, 000 + 375 + 32 8 gas).

Included are the fixed costs for both the data payload and the carrier operation. Although more effective, storing and altering data as a variable in a CA is less flexible because of the Solidity language's limitations on the kinds of values and lengths.

Performance Analysis

The transaction price—the miner's compensation for adding the transaction in a block—and network congestion determine how well the BC network performs. The profit percentage must be raised to draw miners, which raises the total cost, if the customer want to shorten the transaction's wait time. On the other hand, cutting costs will result in longer wait times. When master hash values from various CSPs take too long to appear in a block or appear in different blocks at different times, it might cause synchronization issues. Therefore, there is a trade-off between performance and client- borne overhead costs. Moreover, extra tasks unique to a certain strategy can influence performance. While master hash values in an Ethereum transaction can be embedded without the need for opcodes, CSPs embed them in Bitcoin transactions utilizing the NULL DATA script. On the other hand, storing the master hash with CA options will necessitate numerous function calls.

Security Analysis

Confidentiality: Since the client should already be using HE while storing data in the cloud, confidentiality is guaranteed by default. The client must possess the associated private key in order to decrypt the data, which is encrypted using the public key. Consequently, client data can be accessed using this cryptosystem without being revealed to unauthorized parties, such as the CSP.

Privacy: A CSP may be authorized by the customer to handle data processing under the HE scheme. This is accomplished by giving the CSP access to the encrypted data's public key. Client data is thereby protected against unauthorized or false collection, usage, and/or disclosure assaults.

Integrity: The author postulates that computation attacks—that is, computations not requested by the client—can be carried out by a malevolent CSP.

There are a minimum of four CSPs according to the BFT consensus principle. The master hash for each of these CSPs' databases must be generated and saved on the BC. Clients will receive the block header for the purpose of verification.

The malicious CSP will be identified when the client compares the master hash values of all the CSPs, since its hash value will be different from everyone else's. Client data is therefore not altered or removed in an illegal or covert way.

Implementation Analysis and Future Work

Because it does not necessitate creating scripts to reengineer the cloud structure or alter the architecture of CSP processing, the concept is simple to implement. All that needs to happen is for the CSP to carry out extra tasks. The CSP database's hash value is first calculated, and it is then embedded within a BC transaction. Because a client can hire as many CSPs as they want as long as there are at least four, the suggested approach is also scalable. Moreover, there is no need for consensus-building or communication amongst the CSPs themselves under the suggested plan. As a result, the suggested plan is free from issues caused by erratic data and CSP desynchronization with the client. The suggested system, in spite of its many benefits, is unable to reveal which data records have been compromised or altered. Our goal for future work is to put in place a feature that can identify the inaccurate data records in the CSP database.

Conclusion

This article addresses cloud computing data breaches and the cloud service provider's all-encompassing jurisdiction over client data activities. We suggest a method that strengthens the client's capacity to safeguard info. The suggested method uses homomorphic encryption to protect the privacy and confidentiality of data while it is being computed by third parties.

A novel solution based on a distributed network of cloud service providers and Byzantine Fault Tolerance consensus is developed to guarantee data integrity and identify data tampering from the cloud service provider itself. The various cloud service providers do not need to communicate directly with one another under the suggested arrangement. Cloud service providers must compute the master hash values of their databases and store them on blockchain networks, such as Ethereum or Bitcoin, in order to give the client immutable verification data. To accommodate various client needs, we offered a quantitative study of overhead expenses based on a number of time possibilities. When routinely producing master hash verification values every 30 minutes, embedding the master hash value as a log event in the Ethereum network has proven to be the least expensive of all the options (around $88 USD yearly). However, the architecture where the master hash values are included as a variable in an Ethereum transaction offers the best online performance. Additionally, we examined the security specifications and clarified the simplicity of proposed work.

References

1. Agarwal V, Kaushal AK, Chouhan L. A survey on cloud computing security issues and cryptographic techniques. InSocial Networking and Computational Intelligence: Proceedings of SCI-2018 2020 (pp. 119-134). Springer Singapore.
2. Cloud Security Alliance. (2017). Security Guidance V4.0. Available: https://cloudsecurityalliance.org/download/security-guidance-v4/
3. CSA. (2020). Top Threats to Cloud Computing: Egregious Eleven. Available: https://cloudsecurityalliance.org/artifacts/top-threatsto-cloud-computing- egregious-eleven/
4. R. Kissel, ''Glossary of key information security terms,'' Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NISTIR 7298, 2013, Revision 2. Available: http://nvlpubs.nist.gov/ nistpubs/ir/2013/NIST.IR.7298r2.pdf
5. Sullivan B, Tabet S, Bonver E, Furlong J, Orrin S, Uhley P. Practices for secure development of cloud applications. SAFECode & Cloud Security Alliance. 2013 Dec.
6. Cloud Security Alliance. (2016). Top Threats Research. [Online]. Available: https://cloudsecurityalliance.org/group/top-threats/
7. Kumar R, Goyal R. On cloud security requirements, threats, vulnerabilities and countermeasures: A survey. Computer Science Review. 2019 Aug 1;33:1-48.
8. Phaphoom N, Wang X, Abrahamsson P. Foundations and technological landscape of cloud computing. International Scholarly Research Notices. 2013;2013.
9. Grobauer B, Walloschek T, Stocker E. Understanding cloud computing vulnerabilities. IEEE Security & privacy. 2010 Jun 17;9(2):50-7.
10. Fernandes DA, Soares LF, Gomes JV, Freire MM, Inácio PR. Security issues in cloud environments: a survey. International journal of information security. 2014 Apr;13:113-70.
11. Modi C, Patel D, Borisaniya B, Patel A, Rajarajan M. A survey on security issues and solutions at different layers of Cloud computing. The journal of supercomputing. 2013 Feb;63:561-92.
12. Liu F, Tong J, Mao J, Bohn R, Messina J, Badger L, Leaf D. NIST cloud computing reference architecture. NIST special publication. 2011 Sep;500(2011):292.
13. Patil Madhubala R. Survey on security concerns in Cloud computing. In2015 International Conference on Green Computing and Internet Of Things (ICGCIoT) 2015 Oct 8 (pp. 1458-1462). IEEE.
14. Martin L. XTS: A mode of AES for encrypting hard disks. IEEE Security & Privacy. 2010 May 24;8(3):68-9.
15. Lin HY, Tzeng WG. A secure erasure code-based cloud storage system with secure data forwarding. IEEE transactions on parallel and distributed systems. 2011 Oct 6;23(6):995-1003.
16. Ahmed M, Vu QH, Asal R, Al Muhairi H, Yeun CY. Lightweight secure storage model with fault-tolerance in cloud environment. Electronic Commerce Research. 2014 Nov;14:271-91.
17. Van Dijk M, Juels A, Oprea A, Rivest RL, Stefanov E, Triandopoulos N. Hourglass schemes: how to prove that cloud files are encrypted. InProceedings of the 2012 ACM conference on Computer and communications security 2012 Oct 16 (pp. 265-280).
18. S. Eletriby, E. M. Mohamed, and H. S. Abdelkader, ''Modern encryption techniques for cloud computing randomness and performance testing,'' in Proc. 3rd Int. Conf. Commun. Inf. Technol. (ICCIT), 2012, pp. 800–805.
19. S. Zaineldeen and A. Ate, ''Review of cryptography in cloud computing,'' Int. J. Comput. Sci. Mobile Comput., vol. 9, no. 3, pp. 211–220, Mar. 2020.
20. Mouhib, D. Ouadghiri, and N. Hassan, ''Homomorphic encryption as a service for outsourced images in mobile cloud computing environment,'' in Cryptography: Breakthroughs in Research and Practice. Hershey, PA, USA: IGI Global, 2020, pp. 316–330, doi: 10.4018/978-1-7998-1763- 5.ch019.
21. P. Awasthi, S. Mittal, S. Mukherjee, and T. Limbasiya, ''A protected cloud computation algorithm using homomorphic encryption for preserving data integrity,'' in Recent Findings in Intelligent Computing Techniques (Advances in Intelligent Systems and Computing). Singapore: Springer, 2019, p. 707, doi: 10.1007/978-981-10- 8639-7_53.
22. Alanwar, Y. Shoukry, S. Chakraborty, P. Martin, P. Tabuada, and M. Srivastava, ''PrOLoc: Resilient localization with private observers using partial homomorphic

encryption,'' in Proc. 16th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN), Apr. 2017, pp. 41–52, doi: 10.1145/3055031.3055080.

23. P. K. Sharma, M.-Y. Chen, and J. H. Park, ''A software defined fog node based distributed blockchain cloud architecture for IoT,'' IEEE Access, vol. 6, pp. 115–124, 2018, doi: 10.1109/ACCESS.2017.2757955.
24. K. Gai, Y. Wu, L. Zhu, L. Xu, and Y. Zhang, ''Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks,'' IEEE Internet Things J., vol. 6, no. 5, pp. 7992–8004, Oct. 2019, doi: 10.1109/JIOT.2019.2904303.
25. X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, ''ProvChain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability,'' in Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID), May 2017, pp. 468–477, doi: 10.1109/CCGRID.2017.8.
26. R. L. Rivest, L. Adleman, and M. L. Dertouzos, ''On data banks and privacy homomorphisms,'' Found. Secure Comput., vol. 4, no. 11, pp. 169–180, 1978.
27. S. Goldwasser and S. Micali, ''Probabilistic encryption & amp; how to play mental poker keeping secret all partial information,'' in Proc. 14th Annu. ACM Symp. Theory Comput., 1982, pp. 365–377.
28. P. Paillier, ''Public-key cryptosystems based on composite degree residuosity classes,'' in Proc. Int. Conf. Theory Appl. Cryptograph. Techn. Berlin, Germany: Springer, 1999, pp. 223–238.
29. Boneh, E. Goh, and K. Nissim, ''Evaluating 2-DNF formulas on ciphertexts,'' in Theory Cryptography. Berlin, Germany: Springer, 2005, pp. 325–341, doi: 10.1007/978-3-540-30576-7_18.
30. C. Gentry, ''A fully homomorphic encryption scheme,'' Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2009.