IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 21, 2021, accepted April 25, 2021, date of publication May 3, 2021, date of current version May 17, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3077123

An Integrated Architecture for Maintaining Security in Cloud Computing Based on Blockchain

RUBA AWADALLAH, AZMAN SAMSUDIN, JE SEN TEH, AND MISHAL ALMAZROOIE
School of Computer Sciences, Universiti Sains Malaysia, Penang 11800, Malaysia

Corresponding authors: Ruba Awadallah (awadallah.rn@gmail.com) and Azman Samsudin (azman.samsudin@usm.my)

This work was supported in part by the Universiti Sains Malaysia under Grant LRGS/MRUN: 203.PKOMP.6777005.

ABSTRACT Due to its wide accessibility, cloud services are susceptible to attacks. Data manipulation is a serious threat to data integrity which can occur in cloud computing – a relatively new offering under the umbrella of cloud services. Data can be tampered with, and malicious actors could use this to their advantage. Cloud computing clients in various application domains want to be assured that their data is accurate and trustworthy. On another spectrum, 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. To overcome the cloud service provider's (CSP) ultimate authority over the data, the proposed scheme relies on the Byzantine Fault Tolerance consensus to build a distributed network of processing CSPs based on the client requirements. After certain computations performed by all CSPs, they produce a master hash value for their database. To ensure immutable data is produced, master hash values are preserved in Bitcoin or Ethereum blockchain networks. The master hash values can be obtained by tracking the block header address for verification purposes. A theoretical analysis of the overhead costs associated with creating master hash values for each of the cryptocurrencies is presented. We found that Ethereum leads to lower client financial costs and better online performance than Bitcoin. We also specify the data security requirements the proposed scheme provides, the ground-level implementation, and future work. The proposed verification scheme is based on public cryptocurrency as a back-end service and does not require additional setup actions by the client other than a wallet for the chosen cryptocurrency.

INDEX TERMS Blockchain, cloud computing, data integrity, homomorphic encryption.

I. INTRODUCTION

Data security is frequently characterised by data security threats. The area of cloud computing is no different as it is prone to various threats. The primary reason for this is that cloud computing combines many different technologies in its operation. It is paramount to use the process of risk management to balance the benefits of security risks, and cloud computing [1]. Cloud Security Alliance (CSA) [2] is a non-profit organisation set up for the purpose of enforcing general security. CSA has laid out essential shared responsibilities for cloud service providers (CSPs) and the clients to mitigate the risks associated with cloud computing. The job of the CSP is to record, design and implement the client security control and internal security control. The design and implementation are carried out using a tool known as Consensus Assessments Initiative Questionnaire (CAIQ). A cloud consumer uses a Cloud Control Matrix (CCM) to document the people in charge of implementing specific controls and the manners in which they go about it. Also, a high-level process model for cloud security management has been developed to cater to the significant variations of the process model that are likely to occur in building a cloud project, as illustrated in Fig. 1. The essence is to find out the necessary requirements, structure the architecture, and find any missing spaces according to the underlying cloud platform's capabilities.

Despite the CSP's attempts to establish a strong security base, such arrangements are rarely substantive from the data

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Chi Chen.

TABLE 1. Security threats over security domain; Adopted from [3].

	Domain 1	Domain 2	Domain 3	Domain 4	Domain 5	Domain 6	Domain 7	Domain 8	Domain 9	Domain 10	Domain 11	Domain 12	Domain 13	Domain 14	THREAT ANALYSIS					
Security Concern	Cloud Computing Concepts and Architectures	Governance and Enterprise Risk Management	Legal Issues, Contracts & Electronic Discovery	Compliance and Audit Management	Information Governance	Management Plane & Business Continuity	Infrastructure Security	Virtualization and Containers	Incident Response	Application Security	Data Security and Encryption	Identity Entitlement & Access Management	Virtualization	Related Technologies	Spoofing Identity	Tampering with data	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Data Breaches		√	√	√	√	√			√		√	√		√				√		
Misconfiguration and Inadequate Change Control				√	√	√	√	√		√	√	√				√	√	√	√	
Lack of Cloud Security Architecture and Strategy	√					√	√								√	√	√	√	√	√
Insufficient Identity, Credential, Access and Key Management											√	√			√	√	√	√	√	√
Account Hijacking		√				√			√			√			√	√	√	√	√	√
Insider Threat		√			√						√	√			√	√		√		√
Insecure Interfaces and APIs					√	√			√	√	√	√				√	√	√		√
Weak Control Plane	√				√		√	√				√				√		√		√
Metastructure and Applistructure Failures	√	√		√	√	√	√	√	√	√	√	√			√	√	√	√	√	√
Limited Cloud Usage Visibility					√						√				√	√	√	√	√	√
Abuse and Nefarious Use of Cloud Services						√	√		√	√					√	√	√	√	√	√

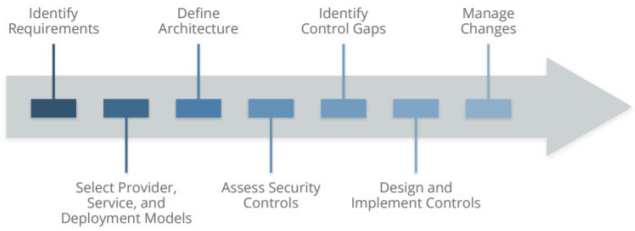

FIGURE 1. CSA Process model for cloud security management [2].

owners' perspective, especially when it comes to trusting the CSP itself. This is compounded by the fact that the growth of cloud computing technology leads to new security vulnerabilities and amplifies existing ones. A recent CSA survey [3] compiled the most significant security issues within cloud computing, classifying the top 11 threats into 14 security domains which can be divided into either the governance or operational domains. The governance domain focuses on strategic and policy issues within a cloud computing environment, whereas the operational domain focuses on tactical security concerns. Table 1 shows how an enterprise opens itself to many commercial, financial, technical, legal, and compliance risks if it adopts a CSP as it is. Most security concerns lead to various threats like identifying spoofing, tampering with data repudiation, information disclosure, denial of service, and elevation of privilege.

Data breaches is at the forefront of threat ranking trend analyses conducted by CSA through 2011, 2016, and 2020 [3], [5], [6]. If such security breaches occur, it would seriously undermine the validity and trust of a cloud-based service. A data breach is an event where an unauthorised entity releases, analyses, steals or uses essential, safe or confidential information. A data breach can either be the primary purpose of a targeted attack or a result of human error, implementation vulnerabilities or inadequate security procedures. The leak of any information not meant for public access is considered a data breach. [7] noted that either encryption and keys vulnerabilities or data storage cryptography vulnerabilities could lead to data breaches. More explicitly, the absence of appropriate encryption algorithms and a poor key management mechanism can contribute to encryption, and key-related failures that directly impact data confidentiality and completeness [8]. Weak key management, defective, insecure, and outdated encryption methods allow data storage to be susceptible to threats [9], [10]. This has been further supported in [11] which highlighted that weak encryption techniques contribute to the most significant risk.

As with any information security management system, fundamental cloud security requirements include confidentiality, integrity, and availability [12]. The problem of data breaches is directly related to confidentiality and privacy. Confidentiality requires that sensitive client data is not disclosed to any

unauthorised entity, whereas privacy refers to the rights of a client to have control over how its data is handled. Encryption algorithms are utilised to achieve both data confidentiality and privacy requirements. There have been various cryptographic schemes proposed to preserve the security of stored data and/or processed data. Various symmetric key encryption algorithms have been used for cloud computing platforms [13]. [14] proposed applying AES to encrypt hard disc data in XEX-based tweaked-codebook mode with ciphertext stealing (XTS). [15] provided data confidentiality, privacy, and availability by employing proxy re-encryption in conjunction with decentralised deletion code. [16] has released a secure storage model for cloud data based on Reed-Solomon (SECRESO) model. They adhere to data security, integrity, availability and error tolerance by improving Reed-Solomon code-based encryption, as well as relying on a log-based backup. [17] established an hourglass protocol to encrypt data files which, at the same time, enables users to check the accuracy of the encrypted file.

Although the use of encryption algorithms like AES can be used to protect data transmitted to the cloud, some findings indicate that traditional cryptographic algorithms are more appropriate for traditional IT infrastructure rather than cloud computing environments [18]. One of the main reasons is that the stored data requires decryption before any calculation can be performed [19]. Despite the merits of the previous proposals in providing adequate security for data-at-rest, it is not feasible for data processing on an external server. Homomorphic encryption (HE) has been proposed to overcome this issue. HE, an asymmetric key encryption scheme, allows calculations to be performed on the encrypted data by a third party without disclosing the corresponding keys. Many HE schemes have been proposed in recent years with the goal of improving cloud computing security [20]–[22].

In contrast, because CSP is a centralised management approach, HE alone cannot fulfil IND-CCA2 security notions against tampering. Therefore, client data is exposed to administrative risks that may cause data loss or undisclosed manipulation in the database. In order to employ decentralisation and enhance the manipulation transparency of homomorphically encrypted data, blockchain (BC) technology is the one possible solution. Several studies have relied on BC in restructuring the cloud to solve several security issues [23]–[25]. However, they suffer from complicated configuration setup, and the embedding requires a huge effort and budget.

In this paper, we overcome these existing problems by proposing a verification scheme based on the notions of BFT and blockchain technology. More than one CSP will be hired to store and perform computations on client data. Each CSP will have to periodically compute a master hash value of their database to be stored on a public blockchain such as Bitcoin or Ethereum. These CSPs do not need to collaborate or communicate with one another. A client can compare these master hash values to detect if data tampering has occurred. This distributed verification system fulfils the requirements of confidentiality (HE will be used for encryption), and integrity because data modifications by the CSPs can be detected by comparing master hash values stored on the blockchain.

The rest of this paper is structured as follows: In Section II, we provide a detailed explanation of both HE and BC as they play a major role in solving the problem of CSP centralisation in processing client data. Next in Section III, we introduce the proposed scheme. The results, discussion and recommendations for future work are detailed in Section IV. Finally, Section V provides the conclusion of our proposed scheme.

II. PRELIMINARIES

The proposed approach in this paper adopts both HE and BC in a unified approach for maintaining data confidentiality in cloud computing. Important concepts from both HE and BC are detailed in Subsections II-A and II-B respectively.

A. HOMOMORPHIC ENCRYPTION (HE)

Once data is stored in the cloud, a client's sovereignty over its data is lost, leaving the data vulnerable to many security threats. [26] were the first to introduce the concept of ''privacy homomorphism'' in an effort to solve problems with external computations. Over time, other proposals with different execution classifications emerge [27]–[30], and through various attempts, a clear definition of HE has been introduced as follows [31]–[33]:

Definition 1 (Homomorphic Encryption): The conversion of data into ciphertext where the process has the capacity to conduct operations on data that is encrypted without any reach to the private decryption key; the owner of the data should be the only one in possession of the private key. In the process of applying arithmetic operations to encrypted data, the same results should be gotten as in the case of unprocessed data.

The first process of HE is key generation *KeyGen*, where the data owner creates the public-key pair (a public key *puk* and a private key *prk*). The next is the encryption process *Enc* which involves applying the encryption algorithm onto the data $C = Enc_{puk}(P)$ before sending it to the cloud server. At the cloud server, the *puk* and the encrypted data are stored in a database. When instructed by the client, the cloud server performs the requested calculation on the encrypted data before sending the result back to the client in its encrypted form. This is known as the evaluation process, *Eval*. With the corresponding *prk*, the client is able to process the decryption function, *Dec*, to recover the plaintext. To sum up, HE has four main operations, namely: *KeyGen*, *Enc*, *Eval*, *Dec*.

Despite the advantage of a homomorphic cryptosystem, by malleability nature, all designs are not IND-CCA2 secure. This can lead to erroneous outsourced computations. It is important to note that these problems can occur even without decryption [34]. Thus, the use of HE schemes alone does not guarantee full data security. Data integrity can still be compromised by CSP and can go undetected. For example, the CSP can implicitly substitute a given ciphertext or the cumulative result with other valid ciphertexts without

knowing the content of the substituted data. Once integrity is compromised, there is no way to restore the original data. Therefore, data integrity needs to be enforced on such outsourced computations. In addition, a CSP is considered a third party that a client assigns to perform complex computations. The centralised nature of these computations and the authority to modify data are also threats to data integrity. The use of protocols such as Secure Shell (SSH) does not overcome these issues as it has no support for authentication.

To establish a secure CSP platform apart from encrypting data homomorphically, there is a need for a robust, tamper-proof, and verifiable security architecture. These are the properties of the BC architecture that consists of a distributed ledger (or database) of congregated transactions managed by a peer-to-peer network working to validate blocks [35]. CSP and BC technologies look conflicting in architecture, namely, centralisation versus decentralisation as illustrated in Table 2. However, CSP and BC can complement one another in a unified solution to take advantage of both their benefits [36]. We take an in-depth look at BC in the upcoming subsection.

TABLE 2. Comparison between distributed ledger and centralized database; adopted from [37].

Parameter	Distributed Ledger (BC)	Centralized Database (CSP)
DB Ownership	*Distributed Ledgers*: Participants can own and maintain their own copy of the ledger. This design of many backup copies makes ledger destruction very difficult.	*Centrally Owned DB*: The user must trust the DB owner and accept risks (exposure, loss or destruction).
Security	*Secure*: Its distributed nature prevents a one-point attack. If an attack happens to any one of the nodes, it will not affect the overall system.	*Not Secure*: Users must trust the authorised computer system to deal with security issues and implement security countermeasures.
Transactions	*Validation*: BC network prevents all invalid transactions from being propagated throughout the network.	*No Validation*: A user must trust the database owner to validate transactions.
	Complete: Adding new blocks to the BC must depend on hash references to previous blocks otherwise transactions are deemed incomplete.	*No Evidence*: A user must trust the DB owner to include all validated transactions (which might not happen).
	Tamper Resistant: Digital signatures and message digests (from cryptographic hash functions) are added to transactions to achieve tamper resistance.	*Alterable*: A user must trust the DB owner to preserve past transactions.
Network	*Heterogeneous Network*: Software and hardware may be distributed throughout the network. Such an infrastructure has multiple points of attack.	*Homogeneous Network*: Software and hardware are located within the same network infrastructure which increases resilience against attacks.
Location	*Globally Located*: Various nodes distributed around the globe work together in a peer-to-peer manner, making the network fault resistant. The network will still remain functional despite losing a node or entire region of nodes.	*Centrally Located*: Access to the DB can be adversely affected if there are network outages in this specific location.

B. BLOCKCHAIN TECHNOLOGY (BC)

Implementing BC techniques in cloud scenarios have attracted considerable attention in both academia and industry. BC technology, in essence, consists of distributed digital blocks bound to each other based on cryptographic principles. Each block contains a cryptographic hash of the previous block, a timestamp and transaction data. BC grant all participants the ability to authenticate transactions independently on a peer-to-peer network. To approve and record transactions in the BC, a consensus mechanism is required to ensure that the network of nodes is in agreement. Once a block is validated, it cannot be altered retroactively without modification of all subsequent blocks. NIST defines BC technology as follows [37]:

Definition 2 (Blockchain): Distributed digital ledgers of cryptographically signed transactions that are grouped into blocks. Each block is cryptographically linked to the previous one (making it tamper evident) after validation and undergoing a consensus decision. As new blocks are added, older blocks become more difficult to modify (creating tamper resistance). New blocks are replicated across copies of the ledger within the network, and any conflicts are resolved automatically using established rules.

Several businesses catering to the interest in BC technologies by developing cloud-based BCs. Well-known CSPs have provided Blockchain as a Service (BaaS) to their clients based on the Software as a Service (SaaS) model. [38] launched the Amazon Managed Blockchain using open-source software platforms such as Ethereum and Hyperledger Fabric that allows developers to create and share information in a decentralised manner easily. [39] has proposed a model with improved efficiency that allows users to record and track transactions at the audit level. [40] collaborated with ConsenSys to present Ethereum Blockchain as a Service (EBaaS) on Microsoft Azure. This model makes it easier for companies to collaborate and experiment with different business processes before launching them in-house. The R3 consortium with the largest group of financial institutions also proposed a distributed financial ledger known as Corda [41]. Although many entities are actively developing new services and features within the BC space, none of them specifically address the issue of CSPs and their control over client data.

Researchers have also investigated the combination of both technologies. [23] proposed a cloud-based IoT architecture for tracking data stored in the cloud that can be activated by a blockchain-based software-defined network (SDN). In the same vein, a smart contract solution for data allocation has been proposed by [24] to introduce automated resource management using a blockchain network. [25] introduced a blockchain-enabled data tracing method known as ProvChain

to protect data being used during the provisioning process. The main concern of the aforementioned methods is the data origin of distributed computing. Similarly, there is a lack of work dealing with the centralised nature of CSPs and their related security issues.

BC technology is also referred to as trust machines [42] because their architecture utilises well-known mechanisms of computer science and cryptographic primitives together with record-keeping principles. The main component within a BC network is a cryptographic hash function which is utilised for address derivation, creating unique identifiers, securing the block data, and securing the block header [37]. Hash functions are designed based on three main properties: preimage resistance, second preimage resistance, and collision resistance [43]. Federal Information Processing Standard (FIPS) 180-4 determined different specifications for NIST-approved hash functions [44]. The proposed work utilises Secure Hash Algorithm 2 (SHA-2) with an output size of 256-bits to calculate its master hash value. There are specialised instruction sets for CPUs such as Intel that supports hardware acceleration of the SHA family, making it efficient to compute. SHA-2 has an output of 32-bytes (1-byte = 8-bits, 32-bytes = 256-bits), and is generally displayed as a 64-character hexadecimal string [45]. Hash functions such as SHA-2 are also used in some consensus mechanisms such as proof-of-work (PoW). A consensus protocol or mechanism is a fundamental building block of a BC as it is a means of selecting nodes to publish new blocks.

BC technology has the potential to create greater transparency and fairness while also saving companies time and resources. Numerous BC-based applications are involved in the marketplace with different industries such as cryptocurrencies, cybersecurity, the Internet of Things, healthcare, business, logistics, supply chain, and many other real-world applications [46]. Our proposed work relies on cryptocurrencies, one of the earliest and still arguably the most well-known use case of BC technology [47]. We will be utilising Bitcoin and Ethereum, two of the most popular cryptocurrencies in the market today.

1) BITCOIN

Bitcoin (BTC) was the first and most popular cryptocurrency in the market, proposed by Nakamoto in 2008 [48]. BTCs are obtained as rewards via mining (calculating the PoW puzzle) and can be transferred between Bitcoin accounts. Each transfer is recorded in a transaction that will be stored in a block on the BC. All participating nodes store the same copy of the Bitcoin BC. Nodes who are successful in computing the PoW are selected as block leaders to create, broadcast and append a new block to the BC. If all transactions within the block are valid, other nodes will accept the new block and include it in their own copy of the BC.

Cryptocurrencies rely on digital wallets to facilitate transactions and manage key pairs for receiving and sending cryptocurrency [49]. The Bitcoin wallet's main role is to store the private key used to redeem BTC and to calculate public addresses. From a technological point of view, BTCs themselves are not physically *stored* in the wallet. Instead, they exist on the BC, accessible only by those with the correct private keys. Private keys are also used to "sign" transactions [50]. Free Bitcoin wallets are designed to address a variety of client needs across all major operating systems and apps. There are a variety of different wallets offered by various platforms. While they all share some similar functionality, features differ from wallet to wallet. One of the most valuable features is a shared wallet known as the multi-signature wallet, or multisig wallet [51].

A multisig wallet is accessible by two or more keys and requires at least one to authorise a BTC transaction. Other than standard transactions signed by a single owner of a private key, multiple people are required to sign before the funds can be transferred. It is more secure as it restricts actions on BTC. More specifically, if an attack occurs to one of the wallets, the hacker will not be able to spend BTC from the shared wallet without authorisation from its other owners. Moreover, in a community, it strips off the purchasing control from third party hands as well as tracking participating parties by accessing the transaction history of a single wallet.

2) ETHEREUM

Ethereum is not just a cryptocurrency network but rather a network of independent computers that function together as one supercomputer [52]. It is flexible in allowing the establishment of transactions over either permissioned or permissionless network. It supports more than just cryptocurrency transactions as it is a BC-based platform for executing smart contracts. This platform is known as the Ethereum Virtual Machine (EVM). All smart contracts are compiled into specific bytecode to deploy on the EVM. As the Ethereum platform is Turing-complete, any type of rule or functionality can be described by a smart contract. Ethereum has two main types of accounts: externally owned accounts (EOA) and contract accounts (CA). EOAs are controlled by private keys and have no associated bytecode, whereas CAs have associated code (known as smart contracts) and data storage. Both accounts can communicate with those identical to them and also with one another albeit in different ways [53]. The circulated currency between peers in the Ethereum network is known as Ether.

Depending on the type of account, an Ethereum wallet is either a normal wallet (similar to a BTC wallet) for EOAs or a smart contract wallet that can additionally write, deploy or trigger smart contracts in a CA using the Solidity programming language [54]. Contracts can implement two types of wallets, simple wallets and multisig wallets. In general, the simple wallet only has one account which controls and owns the wallet, whereas a multisig wallet has more than one owner account, one of which is the creator's account [55].

Although Ethereum transactions can be used for multiple purposes, its transaction structure is the same [56]. Any transactions must include at least the following data types:

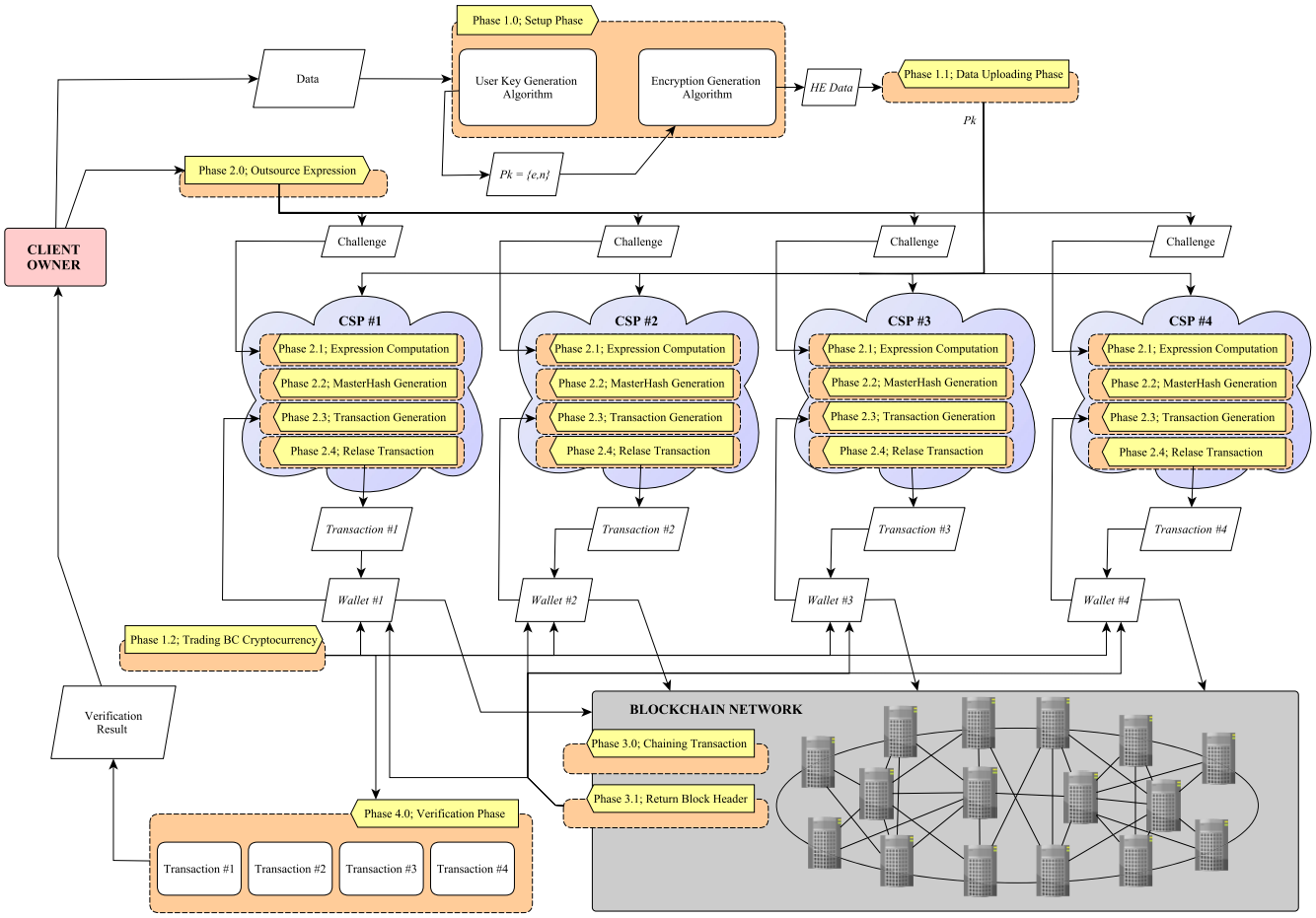

FIGURE 2. Flow diagram.

- **From**: 20-byte address of the transaction initiator's account
- **To**: 20-byte address of the recipient's account
- **Value**: The amount of fund in Wei (1 Ether = 10^{18} Weis).
- **Input**: Data input of various types and size
- **Gas Price**: Determines the cost of executing each operation in a smart contract
- **Gas Limit**: The maximum amount of gas that can be consumed before execution halts

III. PROPOSAL DESIGN

The verified computation design is based on CSP and BC technologies which both play an equally crucial role. To introduce our proposed design, we first discuss the client verification process of operations applied to requested data, beginning with employing more than one path for performing computations before storing the results on the BC to achieve immutability (see Fig. 2). The verification process (as illustrated in Table 3) will consist of three main components (which correspond to the three main phases of the proposed verification scheme) which are the multi-CSPs, BC-application and client, whose roles are detailed below:

1) **Multi-CSPs**: A client will hire more than one CSP. Each established CSP has its own contract with the client but all are subjected to the same terms [57]. The n number of hired CSPs will perform computations, upon completion of which will produce a master hash for their database and forward the result to the BC-based application.
2) **BC-based application**: Creates new blocks that contain the master hashes as a transaction, then returns the block header to CSP.
3) **Client**: Clients can perform verification by comparing master hash values from each CSP based on the received block header information.

Before going through a detailed description of each stage, the client should determine two main aspects that define the design workflow: the frequency of computing master hash values (determined by a frequency variable, t) and the corresponding cryptocurrency wallet. t determines the number of computations requested by a client before the multi-CSPs are required to compute the master hash of their corresponding databases. The value of t depends on two main factors. The first is the client's data growth percentage, and the second is his financial ability to pay the BC transaction fees. Also,

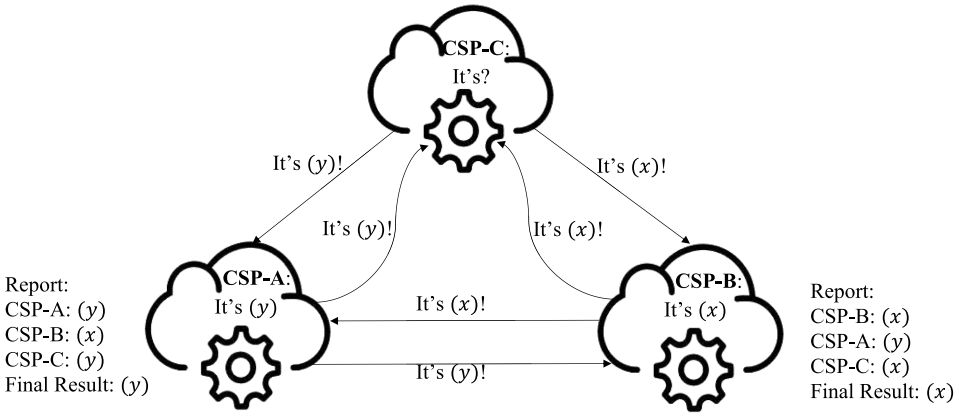

FIGURE 3. BFT Concept.

TABLE 3. Proposed verification scheme.

1.	Client	: Homomorphically Encrypt Data $c_i = Enc_{puk}(m_i)$.
2.	Client → CSP No.1	: Store [encrypted data $(c_1, c_2, c_3, \ldots, c_n)$].
	Client → CSP No.2	: Store [encrypted data $(c_1, c_2, c_3, \ldots, c_n)$].
	Client → CSP No.3	: Store [encrypted data $(c_1, c_2, c_3, \ldots, c_n)$].
	Client → CSP No.4	: Store [encrypted data $(c_1, c_2, c_3, \ldots, c_n)$].
3.	Client → CSP No.1	: query$_1$ $[C_r = c_x \diamond c_y]$.
	Client → CSP No.2	: query$_2$ $[C_r = c_x \diamond c_y]$.
	Client → CSP No.3	: query$_3$ $[C_r = c_x \diamond c_y]$.
	Client → CSP No.4	: query$_4$ $[C_r = c_x \diamond c_y]$.
4.	Client ↔ CSP No.1 → BC	: Transaction Log Hash$_1$ [homomorphically encrypted (DB)].
	Client ↔ CSP No.2 → BC	: Transaction Log Hash$_2$ [homomorphically encrypted (DB)].
	Client ↔ CSP No.3 → BC	: Transaction Log Hash$_3$ [homomorphically encrypted (DB)].
	Client ↔ CSP No.4 → BC	: Transaction Log Hash$_4$ [homomorphically encrypted (DB)].
5.	BC → CSP No.1 ↔ Client	: Block Header [Transaction Log (Hash$_1$)].
	BC → CSP No.2 ↔ Client	: Block Header [Transaction Log (Hash$_2$)].
	BC → CSP No.3 ↔ Client	: Block Header [Transaction Log (Hash$_3$)].
	BC → CSP No.4 ↔ Client	: Block Header [Transaction Log (Hash$_4$)].
6.	Client	: $Hash_{ver} = Hash_1 \oplus Hash_2 \oplus Hash_3 \oplus Hash_4$ $:= \begin{cases} if Hash_{ver} = 0;\ true \\ Hash_{ver} = 1;\ otherwise \end{cases}$

the transaction fee amount and the client's budget are essential factors that will be discussed in Section IV-A. Clients must also prepare their cryptocurrency wallet and share its address with the CSP in order to monitor or verify master hash values. The proposed work will rely on either Bitcoin or Ethereum cryptocurrency. Despite their differences, the steps involved in setting up a wallet are the same [58].

A. CSP - COMPUTATION PHASE

To perform verification in the CSP environment, we adopt certain BC's BFT consensus features and put them into practice. The proposed work will also rely on hash functions as well as the properties of the distributed ledger. The concept of utilising many different nodes in creating a new block is adopted by the proposal, such that the client receives support via multiple CSPs as opposed to just one. The number of hired CSPs is determined based on the BFT scenario. If f CSPs are Byzantine (or malicious), and the system consists of $2f + 1$ CSPs, the malicious CSPs coordinate to say arbitrary things to the other $f + 1$ nodes.

For example, a system is attempting to reach a consensus on the outcome of the calculation (x). Let $f = 1$ and the number of CSPs can be calculated as $N = 2f + 1 = 3$. The three CSPs are denoted as CSP-A, CSP-B, and CSP-C, respectively. Assuming that CSP-C is Byzantine, Fig. 3 illustrates how CSP-C can prevent a consensus from being reached by all three CSPs. CSP-C informs CSP-A that its outcome is y while informing CSP-B that its outcome is x. As the results obtained from CSP-C correspond to each of their own results, CSP-A and CSP-B both accept the results y and x respectively because this is the outcome with the most votes. Thus, the upper limit of f for Byzantine faults should be set to $f < \frac{N}{3}$ [59]. Therefore, in order to tolerate one Byzantine node, the minimum requirement is $N = 4$ CSPs. The proposed work will be analysed under the assumption that there are at least $N = 4$ CSPs. The following points summarise the operations that will be performed by a single CSP:

1) Calculate the master hash value of its database by applying SHA-2 after t times of requested computations.
2) The CSP saves the master hash in a transaction log and forwards it to the mining pool to be stored in the BC network.

To avert 51% attacks, the proposed work relies on the fact that there is no direct communication between the multiple

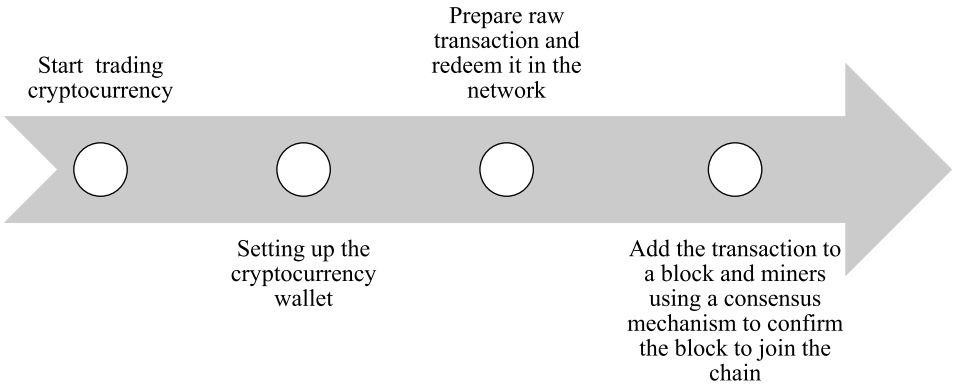

FIGURE 4. Currency flow.

CSPs. In reality, this will be the case as these CSPs may be from different organisations or companies. Assuming that the CSPs can directly communicate with one another, having three malicious CSPs can lead to peer consensus on faulty data and documentation of transactions containing wrong master hashes. Instead, authentication is achieved with the help of the BC.

B. BLOCKCHAIN - MASTER HASH PHASE

To store the master hash values, we consider two popular cryptocurrencies, Bitcoin and Ethereum. Despite having different architectures and features, the flow of currency from the CSP's wallet to the network is the same for both, as illustrated in Figure 4. In the following subsections, detailed explanations on each of these steps will be provided with respect to Bitcoin and Ethereum.

1) BITCOIN

Step 1 - Setting up BTC wallet:

The client will have four multi-signature wallets, one for each of the hired CSPs. On each wallet, there will be three signature keys created, whereby at least two-thirds of those keys are required to perform BTC transactions. Two keys are at the client's disposal, and the third key is held by CSP. The client's keys need to be stored in two different locations: one in the client's wallet and the second is in a safe repository (such as a hardware wallet) as a backup recovery key. Thus, the majority of keys is in the client's hands, and if the first key has somehow been lost, the second key can still be used. After enabling the wallet, wallets should have at least 546 Satoshis (equivalent to 0.00000546 BTC) amount of Bitcoin for the proposed scheme to be used.

Step 2 - Prepare raw transaction and embed master hash: This step can be performed assuming that the shared wallets have already been established and the master hash is ready to be stored in the Bitcoin BC. The individual steps to prepare the raw transaction and embed the master hash value are as follows:

1) **Create multisig transaction address:** Each CSP and the client have to create their own cryptographic key pair consisting of a unique public key and a corresponding private key. From the two public keys, a multisig address will be generated. This process is repeated twice to create two different addresses: one represents the input, and the second represents the output. This allows BTC exchange between two different addresses inside the same wallet [60].
2) **Creating raw transaction and writing master hash in transaction data:** A Bitcoin transaction is a collection of data that describes a BTC transaction. The proposed work will comply with the standard transaction data structure [61] while making modifications only to the output `ScriptPubKey` data.
Bitcoin network is a financial transaction record not meant to store arbitrary data [62]. However, developers have come up with numerous ways to encode data in transactions based on different standard scripts. We are concerned with two types of scripts: the Pay To Pubkey Hash (P2PKH) script and the NULL DATA script.
In the first scheme (P2PKH script), the programmer can store arbitrary data where the hashed public key should be [63]. This means that only 160 bits are available to encode data. The CSP could apply this scheme if SHA-1 was used instead of SHA-2 to calculate the master hash. However, this approach has both security concerns (shorter hash values can lead to attacks) and efficiency concerns as it has a detrimental impact on the users' memory. The efficiency problems stem from the fact that the output is not easily distinguishable from the standard locking script. Hence they have to be kept in the unspent transaction output (UTXO) set, a waste of RAM.
The second scheme depends on the NULL DATA script [64], a standard script that allows pushing metadata onto the BC. The idea works by inserting an additional output and placing a NULL DATA lock script on it. This script overcomes the security and efficiency concerns of the first scheme. Bitcoin core version 0.12.0 allows storing a maximum of 83-bytes of metadata. Also, it contains the *OP_RETURN* script

opcode, which makes it provably unspendable. Hence, it is not necessary to preserve it in the database of the UXTO, thus saving RAM space. A transaction can, however, only contain one NULL DATA locking script in order for it to be certified as a standard transaction. Since we are adopting SHA-2 in the proposed scheme to generate master hash values, we will be utilising this approach.

Step 3 - Signing transaction and broadcasting to the Bitcoin network: When spending from a multisig address, both the CSP and client need to sign the encoded transaction with their private key. Then, the CSP broadcasts the encoded transaction to the network, which will be collected and included into blocks by miners. These blocks will later be appended to the Bitcoin BC upon producing the PoW.

2) ETHEREUM

In the following subsections, we look at two possible master hash storage scenarios for each Ethereum account type based on the currency flow in Figure 4.

C. EXTERNALLY OWNED ACCOUNTS

Step 1 - Setting up ETH wallet: Normal Ethereum wallets store private keys and offer a public ETH address for user accounts [65]. To perform an ETH transaction, each of the hired CSPs has to get a normal wallet. In other words, the CSPs need to be light nodes. However, EOA does not support multi-signatures.

Step 2 - Prepare raw transaction and embed master hash: The following illustrates the transaction layout (in JSON format) that CSPs will implement:

```
{
"from":"CSP EOA address",
"to":"CSP EOA address",
"value":"fund",
"input":"master hash ",
"gas limit":"larger enough",
"gas price":"determined by initiator"
}
```

Step 3 - Signing transaction and broadcasting to Ethereum network: To release a transaction on the Ethereum network, the transaction should be signed by the private key of the initiator account. The signed transaction is submitted to the local Ethereum node, which validates the signed transaction to ensure that it was really signed by this account's address. In the final stage, the signed transaction is broadcast to all peers in the network.

D. SMART CONTRACT ACCOUNTS

Step 1 - Setting up ETH wallet: A multisig wallet in Ethereum is a smart contract deployed for storing ETH that belongs to multiple owners. Each transaction must be approved by a specified number of owners [55]. With respect to the proposed scheme, the client will deploy four shared smart wallets, one for each CSP. There will be three owner accounts on each shared smart wallet: the CSP and two client accounts. To approve a transaction, the two-thirds rule is applied. After launching the shared smart wallet between the client and CSPs, like any other Ethereum address, ETH can be sent to this wallet.

Step2 - Prepare raw transaction and embed master hash: Smart contract deployment in the Ethereum network is performed via transactions. The transaction structure is the same as an EOA, but the data included in the transaction differs [56]. The input data should include the bytecode plus any encoded arguments if required by a constructor. To deploy a multisig contract to the Ethereum network, the CSP will execute the submit_Transaction function to make an ETH exchange order. This can be performed if the wallet has sufficient ETH. A transaction_Id or hash code is returned to CSP as a response. The CSP will disclose this transaction_Id to clients so that the clients can validate it. When the client gets the transaction_Id from CSP, the client can use the transaction_Id to check the transaction data. By invoking the confirm_Transaction function, the client can approve the transaction. The pseudocode of multisig contract can be presented as follows:

Algorithm 1 Multisig Contract in Ethereum Network

```
set maximumOwners = 3
if maximumOwners is occupied then
    for each owner do
        call submit_Transaction( )
        return transId
        call confirm_Transaction( )
    end for
end if
procedure confirm_Transaction( )
    if msg.sender & transId = true then
        call execute_Transaction( )
    end if
end procedure
```

While the multisig contract alone does not have the functionality to store arbitrary data, the CA can deal with data storage differently. Data could be saved as a variable or log event, both of which are suitable for smart contracts. On the other hand, storing data as a variable facilitates the efficient search for data and state changes. Both of these approaches are detailed below:

- The first option is to save arbitrary data as a variable [66]. Each smart contract has data storage of its own, in the form of an array of 2^{256} byte values. We can store in this array, and every non-light (full) node will have a copy of this state. In other words, every smart contract is able to store data in its own database as a vector, i.e. 32-byte values referenced by 32-byte keys. Thus,

the CSP will write the master hash data into the smart contract storage space via the *SSTORE* opcode with a hexadecimal counterpart of $'0 \times 55'$. The master hash data is considered as a fixed-size array with 32-bytes, so it takes one slot of storage as shown in the following pseudocode:

Algorithm 2 CA Variable Data Storage

```
if uint256 [1] Master_Hash then
    set struct entry = uint256 value
end if
```

- The second option is preserving arbitrary data as a log event [67]. Once the contract implements a transfer operation, a log record is created to describe the event emitted. The log record contains event-related information, followed by some additional data. Record data can include large or complicated data types like arrays or strings. While this data storage is not indexed by design (not searchable), a CSP can customise this space to store the master hash data as following:

Algorithm 3 CA Log Event Data Storage

```
set event Storage = uint256 value
```

In the proposed scheme, we will embed the value of the master hash as both options.

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.

E. CLIENT PHASE - VERIFICATION

After storing the master hash in the BC, it is now the client's role to verify that the values sent by all CSPs are identical. Verifications differ according to the different platforms used in this research, so that this process will be discussed separately with respect to the Bitcoin and Ethereum models.

1) BITCOIN VERIFICATION

The verification of master hash values is simple if the client selects the Bitcoin platform. That is due to the multisig wallets with each hired CSP, which means the client must first agree before BTC transactions can be made. When the client is required to agree on a BTC transaction, the client has the ability to read the stored data value inside the transaction. Nevertheless, it is not logical for the client to continue verifying the data this way, especially if t is small. Therefore, at any time that the client wants to verify the hash values, the corresponding block headers of blocks that contain these transactions can be referred to. The block header contains a 4-byte long timestamp that indicates when the block has been included in the BC. The client has to select the transactions that happen approximately within the same time period and compare their shared values.

2) ETHEREUM VERIFICATION

Verification using Ethereum will vary depending on the type of account being used. For EOAs, each CSP is required to send the block header to the client for each transaction. This allows the client to track all transactions and perform verification. As for CAs, the client can obtain the block headers associated with the CSPs' transactions from the multisig wallets. Thus, the verification process can be performed based on the timestamp data in each block header. The approach based on CA outperforms EOA due to its ability to set up the shared wallet, which facilitates the data verification process.

IV. RESULT AND DISCUSSION

In this section, we theoretically evaluate the implementation costs and performance of the proposed scheme based on each BC-related model. The purpose of these calculations is to determine which model is the most financially feasible option and has the best online performance. We exclude the cost of hiring the CSPs and the computational time of the hash functions in our calculations.

A. COST ANALYSIS

1) BITCOIN-BASED COST ANALYSIS

At the time of writing this paper, BTC is currently trading at $33,792.00$ USD[1] [68]. Since the price of BTC is very high, the proposed scheme will only rely on the smallest BTC trading amount possible, taking into consideration network requirements and transaction fees. [69] reported in early 2020 that the transaction fee is, on average, 0.00001 BTC, which is equivalent to approximately \$0.09. Generally, the minimum BTC trading amount is based on how the client obtains BTC (e.g. a cryptocurrency exchange). Suppose, for the sake of argument, the client chooses a Luno [70] which allows users to buy BTC from 0.75 \$. Thus, the client runs into two problems: transaction fees and dust rules [71]. The transaction fees are very close to the holding BTC amount, in which after some time, the fee required to redeem a new transaction will be more than the money that the client is sending. Dust rules mean that transactions with outputs below a specific size will simply be dropped. If one of these problems occurs, it will negatively affect the performance of our proposal. More clearly, it delays the process of chaining the transaction in a block, and consequently, it can cause significant time differences in established blocks for each CSP or even the failure to block a transaction for one of the CSPs.

2) ETHEREUM-BASED COST ANALYSIS

The trading value of Ethereum's cryptocurrency is 97% lower than Bitcoin. 1 ETH costs approximately $\$1,022.43$ USD[1] [72]. If the CSP opens an EOA, the master hash value

[1] As of 13[13] January 2021

will be a part of the transaction. The cost of every zero-byte transaction is at least 21, 000 gas [73]. Each additional byte of transaction data incurs a cost of 68 gas. Thus, the total cost of storing the master hash value generated from SHA-2 is about \$0.006[18] USD. The estimated cost of transaction fees if transactions are performed continuously every half an hour is around \$105 USD per year. We now analyse the additional costs associated with the two data storage approaches for Ethereum.

The first option is to store arbitrary data as a variable in the smart contract's data storage. The number of *SSTORE* operations determines the expenses involved in saving data. In order to store a master hash value of 32-bytes, one *SSTORE* transaction is required to convert data from zero to non-zero, at the cost of 20, 000 gas. As previously mentioned, each operation costs a minimum of 21,000 gas as a carrier. Extra gas is incurred due to the data payload of the operation which comprises the actual data and function signature. There is also an extra cost for the creation of the smart contract itself. The total cost is estimated to be over 0.010 $USD(20,000 + 21,000 + 32 \times 68$ gas).

The second option is to save data as a log event. To compute cost, many aspects need to be taken into consideration. Firstly, logged data is saved in log topics at the cost of 375 gas, while every byte of data in the log topic incurs an additional 8 gas. A rough estimate of utilising a log event in storing 32-bytes of data is 0.005 $USD(21,000 + 375 + 32 \times 8$ gas). The fixed cost of the carrier operation and the data payload is included. Storing and modifying data as a variable in a CA is more efficient but less adaptable due to the restrictions of the Solidity language on the types of value and length.

Table 4 provides a quantitative comparison of verification overhead costs for embedding data in Bitcoin transactions, EOA transactions, or as a variable data and log event in CAs. Fig 5 illustrates the average verification overhead cost for all these options.

B. PERFORMANCE ANALYSIS

The BC network performance depends on the transaction price (the miner's reward for including the transaction inside a block) and network congestion. If the client wants to reduce the waiting time for the transaction, the profit percentage must be increased to attract miners, which increases the overall cost. Inversely, reducing the cost will increase waiting time. This can lead to synchronisation problems, whereby the master hash values from different CSPs take too long to be included in a block or are included in different blocks at different times. This will make it difficult for a client to verify the master hash values as the client needs to locate master hash values from within the same timeframe (based on timestamps). Thus, this is a trade-off between overhead costs borne by the client and performance.

Furthermore, performance also depends on additional tasks specific to a particular approach. In the case of Bitcoin, CSPs embed master hash values using the NULL DATA script whereas embedding master hash values in an Ethereum transaction will not require opcodes. In contrast, CA options will require many function calls to store the master hash. Table 5 illustrates the comparison between cost and performance for all options.

TABLE 4. Overhead costs for verification.

(a) Bitcoin Costs

Transaction fee total cost in	Master hash every			
	30 min	One hour	Half day	One day
One day	\$4.32	\$2.16	\$0.18	\$0.09
One week	\$30.24	\$15.12	\$1.26	\$0.63
One month	\$131.4	\$65.7	\$5.475	\$2.738
One year	\$1576.8	\$788.4	\$65.7	\$32.85

(b) EOA Costs

Transaction fee total cost in	Master hash every			
	30 min	One hour	Half day	One day
One day	\$0.288	\$0.144	\$0.012	\$0.006
One week	\$2.016	\$1.008	\$0.084	\$0.042
One month	\$8.76	\$4.38	\$0.365	\$0.1825
One year	\$105.1	\$52.56	\$4.38	\$2.19

(c) CA (Variable Data) Costs

Transaction fee total cost in	Master hash every			
	30 min	One hour	Half day	One day
One day	\$0.48	\$0.24	\$0.02	\$0.01
One week	\$3.36	\$1.68	\$0.14	\$0.07
One month	\$14.6	\$7.3	\$0.608	\$0.304
One year	\$175.2	\$87.6	\$7.3	\$3.65

(d) CA (Log Event) Costs

Transaction fee total cost in	Master hash every			
	30 min	One hour	Half day	One day
One day	\$0.24	\$0.12	\$0.01	\$0.005
One week	\$1.68	\$0.84	\$0.07	\$0.035
One month	\$7.3	\$3.65	\$0.3042	\$0.152
One year	\$87.6	\$43.8	\$3.65	\$1.825

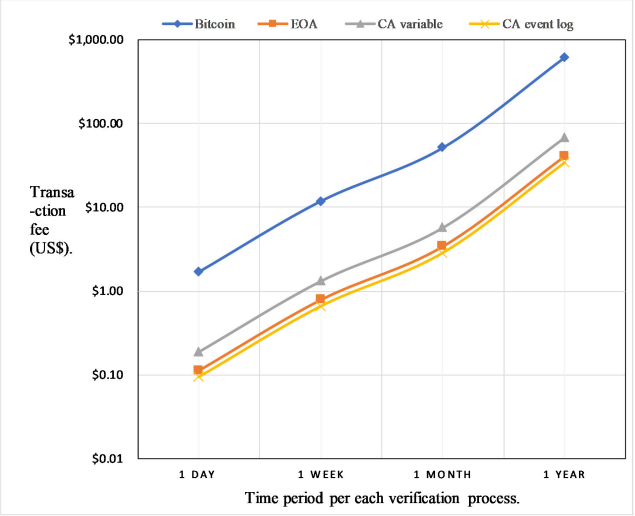

FIGURE 5. Average of verification overhead cost for all options.

TABLE 5. Blockchain overhead cost vs performance comparison.

Option	Cost	Performance
Embedded in transaction (Bitcoin)	⊕	⊕⊕⊕
Embedded in transaction (Ethereum)	⊕⊕⊕	⊕⊕⊕⊕
CA variable (Ethereum)	⊕⊕	⊕⊕
CA log event (Ethereum)	⊕⊕⊕⊕	⊕

⊕: Least favorable, ⊕⊕: Less favorable, ⊕⊕⊕: More favorable, ⊕⊕⊕⊕: Most favorable

C. SECURITY ANALYSIS

In this subsection, we analyse the security of the proposed verification scheme from the perspectives of confidentiality, privacy and data integrity.

Confidentiality. Confidentiality is fulfilled by default because the client should already be using HE when storing data in the cloud. The data is encrypted with the public key and can only be decrypted with the corresponding private key in the client's possession. Therefore, this cryptosystem allows access to client data without disclosing it to unauthorised individuals and entities, including the CSP.

Privacy. The client can authorise a CSP to perform data processing via the HE scheme. This is performed by providing the public key of the encrypted data to the CSP. Thus, client data is immune to false or unauthorised collect, use, and/or disclose attacks.

Integrity. We assume that a malicious CSP can perform computation attacks (computations not requested by the client). Based on the principle of BFT consensus, there are at least 4 CSPs. All of these CSPs have to create the master hash for their database, which is then stored on the BC. The block header will be provided to clients for verification purposes. When the client verifies the master hash values of each CSP, the malicious CSP will be revealed as its hash value will differ from the rest. Thus, client data is not modified or deleted in an unauthorised and undetected manner.

D. IMPLEMENTATION ANALYSIS AND FUTURE WORK

The proposal is easy to establish because it does not require writing codes to reengineer the cloud structure or change the architecture of CSP processing. It just requires that the CSP perform additional operations. The first operation calculates the hash value of the CSP's database, then embedding the hash within a BC transaction. The proposed scheme is also scalable because a client can hire any number of CSPs as long as it is no less than 4. Furthermore, the proposed scheme does not require communication or consensus between the CSPs themselves. Thus, the proposed scheme does not suffer from unpredictable data and challenges arising from CSP desynchronising with the client.

Despite the various advantages, the proposed scheme cannot provide information about which data records have been attacked or tampered with. In future work, we aim to implement a feature that can pinpoint the erroneous data records in the CSP's database.

V. CONCLUSION

This paper focuses on addressing data breaches in cloud computing and the all-encompassing cloud service provider authority over client data operations. We propose an approach that improves the client's ability to protect data. The proposed scheme relies on homomorphic encryption to provide data confidentiality and privacy during outsourced computations. In order to ensure data integrity and detect data tampering from the cloud service provider itself, a novel approach based on a distributed network of cloud service providers and Byzantine Fault Tolerance consensus is introduced. In the proposed scheme, there is no need for direct communication between the multiple cloud service providers. To provide the client with immutable verification data, the cloud service providers are required to calculate master hash values of their databases and store them in blockchain networks, namely Bitcoin or Ethereum. We provided a quantitative analysis of overhead costs based on several time options to suit different client requirements. Embedding the master hash value as a log event in the Ethereum network has shown to be the least expensive of all options (around $88 USD annually) when consistently generating master hash verification values every 30 minutes. On the other hand, the model with the most efficient online performance is where the master hash values are embedded as a variable in an Ethereum transaction. We also analysed the security requirements and explained the ease of implementation of the proposed scheme.

REFERENCES

[1] V. Agarwal, A. K. Kaushal, and L. Chouhan, "A survey on cloud computing security issues and cryptographic techniques," in *Social Networking and Computational Intelligence*. Singapore: Springer, 2020, pp. 119–134, doi: 10.1007/978-981-15-2071-6_10.

[2] Cloud Security Alliance. (2017). *Security Guidance V4.0*. [Online]. Available: https://cloudsecurityalliance.org/download/security-guidance-v4/

[3] CSA. (2020). *Top Threats to Cloud Computing: Egregious Eleven*. [Online]. Available: https://cloudsecurityalliance.org/artifacts/top-threats-to-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. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7298r2.pdf

[5] CSA. (2013). *Practices for Secure Development of Cloud Applications*. [Online]. Available: https://safecode.org/practices-for-secure-development-of-cloud-applications/

[6] Cloud Security Alliance. (2016). *Top Threats Research*. [Online]. Available: https://cloudsecurityalliance.org/group/top-threats/

[7] R. Kumar and R. Goyal, "On cloud security requirements, threats, vulnerabilities and countermeasures: A survey," *Comput. Sci. Rev.*, vol. 33, pp. 1–48, Aug. 2019.

[8] N. Phaphoom, X. Wang, and P. Abrahamsson, "Foundations and technological landscape of cloud computing," *ISRN Softw. Eng.*, vol. 2013, pp. 1–31, Feb. 2013, doi: 10.1155/2013/782174.

[9] B. Grobauer, T. Walloschek, and E. Stocker, "Understanding cloud computing vulnerabilities," *IEEE Secur. Privacy Mag.*, vol. 9, no. 2, pp. 50–57, Mar. 2011, doi: 10.1109/MSP.2010.115.

[10] D. A. B. Fernandes, L. F. B. Soares, J. V. Gomes, M. M. Freire, and P. R. M. Inácio, "Security issues in cloud environments: A survey," *Int. J. Inf. Secur.*, vol. 13, no. 2, pp. 113–170, Apr. 2014, doi: 10.1007/s10207-013-0208-7.

[11] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, "A survey on security issues and solutions at different layers of cloud computing," *J. Supercomput.*, vol. 63, no. 2, pp. 561–592, Feb. 2013, doi: 10.1007/s11227-012-0831-5.

[12] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, "NIST cloud computing reference architecture," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. SP 500-292, 2011. [Online]. Available: https://ws680.nist.gov/publication/get_pdf.cfm_pub_id=909505

[13] R. P. Madhubala, "Survey on security concerns in cloud computing," in *Proc. Int. Conf. Green Comput. Internet Things (ICGCIoT)*, Oct. 2015, pp. 1458–1462.

[14] L. Martin, "XTS: A mode of AES for encrypting hard disks," *IEEE Security Privacy Mag.*, vol. 8, no. 3, pp. 68–69, May 2010, doi: 10.1109/MSP.2010.111.

[15] H.-Y. Lin and W.-G. Tzeng, "A secure erasure code-based cloud storage system with secure data forwarding," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 6, pp. 995–1003, Jun. 2012, doi: 10.1109/TPDS.2011.252.

[16] M. Ahmed, Q. H. Vu, R. Asal, H. Al Muhairi, and C. Y. Yeun, "Lightweight secure storage model with fault-tolerance in cloud environment," *Electron. Commerce Res.*, vol. 14, no. 3, pp. 271–291, Nov. 2014, doi: 10.1007/s10660-014-9140-9.

[17] M. van Dijk, A. Juels, A. Oprea, R. L. Rivest, E. Stefanov, and N. Triandopoulos, "Hourglass schemes: How to prove that cloud files are encrypted," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2012, pp. 265–280, doi: 10.1145/2382196.2382227.

[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] I. 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] A. 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] D. 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.

[31] E. Orsini, N. P. Smart, and F. Vercauteren, "Overdrive2k: Efficient secure MPC over Z_2^k from somewhat homomorphic encryption," in *Proc. Cryptographers' Track RSA Conf.* Cham, Switzerland: Springer, 2019, pp. 254–283.

[32] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 1–35, Sep. 2018.

[33] T. S. Fun and A. Samsudin, "A survey of homomorphic encryption for outsourced big data computation," *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 8, pp. 3826–3851, 2016.

[34] N. Döttling, J. Müller-Quade, and A. C. Nascimento, "IND-CCA secure cryptography based on a variant of the LPN problem," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2012, pp. 485–503.

[35] M. Kandeeban and T. Nivetha, "Blockchain: A tool for a secure, safe and transparent way of food and agricultural supply chain," *Int. J. Farm Sci.*, vol. 9, no. 1, pp. 97–100, 2019.

[36] K. Gai, K.-K. R. Choo, and L. Zhu, "Blockchain-enabled reengineering of cloud datacenters," *IEEE Cloud Comput.*, vol. 5, no. 6, pp. 21–25, Nov./Dec. 2018.

[37] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," 2019, *arXiv:1906.11078*. [Online]. Available: http://arxiv.org/abs/1906.11078

[38] Amazon Web Services. (2020). *Announcing General Availability of Amazon Managed Blockchain*. [Online]. Available: https://aws.amazon.com/ar/about-aws/whats-new=2019=04/introducing-amazon-managed-blockchain/#::text=Amazon%20Web%20Services%20%28AWS%29%20announces,Hyperledger%20Fabric%20is%20available%20today

[39] *IBM Blockchain Platform*. (2020). [Online]. Available: https://www.ibm.com/cloud/blockchainplatform

[40] Microsoft Azure. (2020). *Ethereum Blockchain as a Service Now on Azure*. [Online]. Available: https://azure.microsoft.com/en-us/blog/ethereum-blockchain-as-a-service-now-on-azure/

[41] R3. (2020). *About R3*. [Online]. Available: https://www.r3.com/about/?_bt=449955842636&_bk=%2Br3%20%2Bdistributed%20%2Bledger&_bm=b&_bn=g&_bg=77836212690&gclid=CjwKCAjwzIH7BRAbEiwAoDxxTjVhaZJeeQhH5y4c1vvGsOF8CC5xoLFlti68xpr5OrA1kjUb5dlfaRoCQ3cQAvD_BwE

[42] M. Jun, "Blockchain government—A next form of infrastructure for the twenty-first century," *J. Open Innov., Technol., Market, Complex.*, vol. 4, no. 1, p. 7, Dec. 2018, doi: 10.1186/s40852-018-0086-3.

[43] S. Indesteege, F. Mendel, B. Preneel, and C. Rechberger, "Collisions and other non-random properties for step-reduced SHA-256," in *Proc. Int. Workshop Sel. Areas Cryptogr.* Berlin, Germany: Springer, 2008, pp. 276–293, doi: 10.1007/978-3-642-04159-4_18.

[44] M. J. Dworkin, *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, Federal Information Processing Standards NIST FIPS-202, 2015. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/202/final

[45] C. Easttom, "Information assurance/encryption," in *The NICE Cyber Security Framework*. Cham, Switzerland: Springer, 2020, pp. 1–30.

[46] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics Informat.*, vol. 36, pp. 55–81, Mar. 2019.

[47] M. H. Miraz and M. Ali, "Applications of blockchain technology beyond cryptocurrency," 2018, *arXiv:1801.03528*. [Online]. Available: http://arxiv.org/abs/1801.03528

[48] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[49] D. Yermack, "Is Bitcoin a real currency? An economic appraisal," in *Handbook of Digital Currency*. New York, NY, USA: Academic, 2015, pp. 31–43.

[50] R. Zhang, R. Xue, and L. Liu, "Security and privacy on blockchain," *ACM Comput. Surv.*, vol. 52, no. 3, pp. 1–34, 2019.

[51] E. Elrom, "Bitcoin wallets and transactions," in *The Blockchain Developer*. Berkeley, CA, USA: Apress, 2019, pp. 121–171.

[52] W. W. Hargrove, F. M. Hoffman, and T. Sterling, "The do-it-yourself supercomputer," *Sci. Amer.*, vol. 285, no. 2, pp. 72–79, Aug. 2001.

[53] N. He, L. Wu, H. Wang, Y. Guo, and X. Jiang, "Characterizing code clones in the Ethereum smart contract ecosystem," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2020, pp. 654–675.

[54] C. Dannen, "Solidity programming," in *Introducing Ethereum and Solidity*. Berkeley, CA, USA: Apress, 2017, pp. 69–88.

[55] M. di Angelo and G. Salzer, "Wallet contracts on Ethereum—Identification, types, usage, and profiles," 2020, *arXiv:2001.06909*. [Online]. Available: http://arxiv.org/abs/2001.06909

[56] S. Rouhani and R. Deters, "Performance analysis of ethereum transactions in private blockchain," in *Proc. 8th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Nov. 2017, pp. 70–74.

[57] R. McLelland, G. Hurey, Y. Hackett, and D. Collins, "Agreements between cloud service providers and their clients: A review of contract terms," in *Proc. Arxius i Industries Culturals*, Girona, Spain, 2014, p. 11.
[58] B. Badr, R. Horrocks, and X. B. Wu, *Blockchain by Example: A Developer's Guide to Creating Decentralized Applications Using Bitcoin, Ethereum, and Hyperledger*. Birmingham, U.K.: Packt Publishing, 2018.
[59] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," Ph.D. dissertation, Dept. Eng. Syst. Comput., Univ. Guelph, Guelph, ON, Canada, 2016.
[60] D. Carboni, "Feedback based reputation on top of the bitcoin blockchain," 2015, *arXiv:1502.01504*. [Online]. Available: http://arxiv.org/abs/1502.01504
[61] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. Sebastopol, CA, USA: O'Reilly Media, 2014.
[62] R. Matzutt, J. Hiller, M. Henze, J. H. Ziegeldorf, D. Müllmann, O. Hohlfeld, and K. Wehrle, "A quantitative analysis of the impact of arbitrary blockchain content on bitcoin," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2018, pp. 420–438.
[63] K. Baqer, D. Y. Huang, D. McCoy, and N. Weaver, "Stressing out: Bitcoin 'stress testing,'" in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2016, pp. 3–18, doi: 10.1007/978-3-662-53357-4_1.
[64] M. Bartoletti and L. Pompianu, "An analysis of Bitcoin OP_RETURN metadata," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2017, pp. 218–230.
[65] *Ethereum Wallets*. Accessed: Sep. 19, 2020. [Online]. Available: https://Ethereum.org/en/wallets/
[66] A.-T. Pănescu and V. Manta, "Smart contracts for research data rights management over the Ethereum blockchain network," *Sci. Technol. Libraries*, vol. 37, no. 3, pp. 235–245, Jul. 2018.
[67] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, Apr. 2017, pp. 243–252.
[68] *Bitcoin Price Index*. Accessed: Jan. 13, 2021. [Online]. Available: https://cointelegraph.com/bitcoin-price-index
[69] *Billfol Report*. [Online]. Available: https://privacypros.io/tools/bitcoin-fee-estimator/
[70] *Luno Website*. Accessed: Jun. 29, 2020. [Online]. Available: https://www.luno.com
[71] C. Pérez-Solà, S. Delgado-Segura, G. Navarro-Arribas, and J. Herrera-Joancomartí, "Another coin bites the dust: An analysis of dust in UTXO-based cryptocurrencies," *Roy. Soc. Open Sci.*, vol. 6, no. 1, Jan. 2019, Art. no. 180817.
[72] *Ethereum Price Index*. Accessed: Jan. 13, 2021. [Online]. Available: https://www.coindesk.com/price/Ethereum
[73] DRG WOOD. (2014). *Ethereum: A Secure Decentralised Generalised Transaction Ledger—Byzantium Version*. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

RUBA AWADALLAH received the B.Sc. degree (Hons.)in network and communication engineering from Al Ain University, United Arab Emirates, in 2012, and the M.Sc. degree in engineering management from Abu Dhabi University, United Arab Emirates, in 2014. She is currently pursuing the Ph.D. degree with the School of Computer Sciences, Universiti Sains Malaysia (USM). She is also a Research Assistant with USM. Her research interests include data security, cryptography, blockchain, and parallel computing.

AZMAN SAMSUDIN received the B.Sc. degree in computer science from the University of Rochester, NY, USA, in 1989, and the M.Sc. and Ph.D. degrees in computer science from the University of Denver, CO, USA, in 1993 and 1998, respectively. He is currently a Professor with the School of Computer Science, Universiti Sains Malaysia (USM). He has published more than 100 articles, over a series of books, professional journals, and conferences. His research interests include cryptography, switching networks, and parallel computing.

JE SEN TEH received the B.Eng. degree (Hons.)in electronics from Multimedia University, Malaysia, in 2011, the M.Sc. degree in computer science from Universiti Sains Malaysia, in 2013, and the Ph.D. degree from the School of Computer Sciences, Universiti Sains Malaysia, in 2017. He is currently a Senior Lecturer with the School of Computer Sciences, Universiti Sains Malaysia. His research interests include cryptography, cryptanalysis, random number generation, machine learning, and chaos theory.

MISHAL ALMAZROOIE received the M.Sc. degree in computer science and the Ph.D. degree in cryptology from Universiti Sains Malaysia (USM), in 2014 and 2018, respectively. He involved in Research and Development (R&D) of information and communication security. He is currently an Independent Researcher. His research interests include symmetric and asymmetric cryptography, postquantum cryptography, quantum computing, quantum cryptography, QKD, quantum reversible circuits, high performance computing, GPGPU, image convolution filters, image segmentation, and clustering.