applied sciences

MDPI

Article

Lightweight Integrity Preserving Scheme for Secure Data Exchange in Cloud-Based IoT Systems

Zaid Alaa Hussien [1], Husam A. Abdulmalik [2], Mohammed Abdulridha Hussain [2], Vincent Omollo Nyangaresi [3], Junchao Ma [4,*], Zaid Ameen Abduljabbar [2,5,6,*] and Iman Qays Abduljaleel [7]

1 Information Technology Department, Management Technical College, Southern Technical University, Basrah 61004, Iraq
2 Department of Computer Science, College of Education for Pure Sciences, University of Basrah, Basrah 61004, Iraq
3 Faculty of Biological & Physical Sciences, Tom Mboya University, Homabay 40300, Kenya
4 College of Big Data and Internet, Shenzhen Technology University, Shenzhen 518118, China
5 Technical Computer Engineering Department, Al-Kunooze University College, Basrah 61001, Iraq
6 Shenzhen Institute, Huazhong University of Science and Technology, Shenzhen 518118, China
7 Department of Computer Science, College of Computer Science and Information Technology, University of Basrah, Basrah 61004, Iraq
* Correspondence: majunchao@sztu.edu.cn (J.M.); zaid.ameen@uobasrah.edu.iq (Z.A.A.)

Abstract: The information obtained from external sources within the cloud and the resulting computations are not always reliable. This is attributed to the absence of tangible regulations and information management on the part of the information owners. Although numerous techniques for safeguarding and securing external information have been developed, security hazards in the cloud are still problematic. This could potentially pose a significant challenge to the effective adoption and utilization of cloud technology. In terms of performance, many of the existing solutions are affected by high computation costs, particularly in terms of auditing. In order to reduce the auditing expenses, this paper proposes a well-organised, lightweight system for safeguarding information through enhanced integrity checking. The proposed technique implements a cryptographic hash function with low-cost mathematic operations. In addition, this paper explores the role of a semi-trusted server with regard to smart device users. This facilitates the formal management of information prior to distribution through the IoT-cloud system. Essentially, this facilitates the validation of the information stored and exchanged in this environment. The results obtained show that the proposed system is lightweight and offers features such as a safeguarding capability, key management, privacy, decreased costs, sufficient security for smart device users, one-time key provision, and high degree of accuracy. In addition, the proposed method exhibits lower computation complexity and storage expenses compared with those of other techniques such as bilinear map-based systems.

Keywords: data integrity; dynamic integrity checking; lightweight; semi-trust server; one time key; smart device user

Citation: Hussien, Z.A.; Abdulmalik, H.A.; Hussain, M.A.; Nyangaresi, V.O.; Ma, J.; Abduljabbar, Z.A.; Abduljaleel, I.Q. Lightweight Integrity Preserving Scheme for Secure Data Exchange in Cloud-Based IoT Systems. *Appl. Sci.* **2023**, *13*, 691. https://doi.org/10.3390/app13020691

Academic Editor: Dimitris Mourtzis

Received: 14 November 2022
Revised: 21 December 2022
Accepted: 30 December 2022
Published: 4 January 2023

1. Introduction

The inherent adaptability and instantaneity serve to generate a number of advantages for the IoT. These include reductions in hardship in the regulation of storage, general information accessibility regardless of location, and evasion of capital costs for various items such as hardware, software, and individual upkeep [1].

The IoT-cloud system technology is recognised as being the most modern advancement for the structuring of IT corporations. This is because of its numerous unparalleled benefits such as on-demand self-service, pervasive network accessibility, context-independent asset sharing, quick asset flexibility, employment-based costs, and

hazard mitigation [2]. Therefore, the IoT-cloud system has become an innovate system that has exerted crucial effects in many organizations. This has seen a rise in the adoption of certain IT services by corporations. An important aspect of this change is the fact that information on the cloud is obtained from external sources. As such, both corporations and individual users have the opportunity to upload information on to the IoT-cloud system.

Although IoT-cloud-based systems have many attractive features, they generate unique and difficult safeguarding issues for information obtained from external sources. This can be attributed to the fact that IoT-cloud systems are isolated from the regulating companies. Consequently, external information decreases a smart device user's regulation and control of their information. This jeopardises the information stored on these IoT-cloud-based systems in a number of ways. For instance, although this infrastructure has greater influence and dependability compared with individuals' computing technology, individuals continue to encounter various external and internal hazards that might affect the accuracy of the information.

This is exampled by the shortages and violations of the safeguarding methods deployed to protect these IoT-cloud systems. In addition, smart device users are unaware of the location of their external information. For instance, financial constraints may make organizations deploy cloud storage space, especially for information that is not frequently accessed [3–5]. Another challenge is that IoT-cloud systems potentially conceal information misplacement, which may be detrimental to the users. Although outsourcing information to the cloud system has financial benefits in terms of expenses and complications in the lengthy and massive amounts of information storage for smart device users, the honesty and efficient accessibility of this information is not always assured. This negatively affects the effective utilisation of smart device cloud users.

There are many conventional cryptographic primitives that can be deployed to safeguard information stored in the cloud. However, these techniques cannot be immediately employed as smart device users do not have absolute control over the storage of their information. In addition, these cryptographic methods may not be appropriate for the computation and storage-limited user devices. Therefore, techniques for the successful confirmation of the validity of external cloud information in the absence of local replications of information have become a serious issue in storing information safely in IoT-cloud systems [6–8].

The process of downloading information does not represent a feasible method for the confirmation of data integrity. This is due to the high costs associated with I/O and the transfer of files within the network. Additionally, I/O is typically inadequate for identifying information fraud during the process of information retrieval. This is because at this stage, information that has been compromised may no longer be salvaged. Another major issue is the expansive dimensions of external information and smart device users' restrictions regarding asset abilities. As such, confirming the validity of information within the IoT-cloud system can be impractical and cost-ineffective for smart device users [8,9].

The provision of integrity checking for information stored on the IoT-cloud system is essential. This is because it ensures that the information is safeguarded and the smart device users' computation assets are protected. To achieve this, additional semi-trusted servers may be utilized, which have the abilities that the smart device users' lack. These semi-trusted servers could audit the external information as required and in accordance with the audit outcomes. This could be followed by the publication of audit evaluations to help smart device users and cloud systems assess the hazards of the cloud provider and enhance necessary programs, respectively [7]. In essence, creating integrity hazard checking procedures can considerably boost the adoption of IoT-cloud systems. In addition, this offers techniques for smart device users to evaluate hazards and enhance their confidence in these systems.

To address this issue, numerous systems deploying various schemes and safeguarding methods have been suggested in the literature [6,10–15]. Similarly, there have been a number of investigations that have sought to identify solutions that can effectively fulfil these demands and offer a high degree of system organisation. They also seek to offer stateless confirmation, unrestricted usage, and accessibility to information. In this environment, the verifier plays critical roles in that every system has to be classified as having either private or public verifiability.

In order to obtain a high level of competence, private verifiability systems enforce computational consequences on smart device users. Nevertheless, public verifiability relieves smart device users of the burden of completing numerous computations to guarantee the standards of information storage. For instance, smart device users can request another organisation to complete the confirmation processes without committing their computation assets. Within the IoT cloud, smart device users could experience unforeseen crises. They may also be subjected to an excessive number of recurring confirmations of integrity. As such, incorporating public verifiability in confirmation procedures is logical and reasonable. In fact, public verifiability is forecasted to exert a considerable influence on achieving the financial benefits for both cloud and IoT.

The methods of creating alternative organizational integrity checking procedures that are not dependent on information encryption are addressed in this paper. To attain this, informational honesty through integrity checking within the IoT-cloud system with a specific emphasis on information storage is offered. The pervasive nature of the IoT-cloud system could potentially heighten the extent of the checking procedures conducted by various individuals who are serving as semi-trust servers. However, the checking of these increasing duties by smart device users is both monotonous and unmanageable. Consequently, many smart device users tend to specifically request that semi-trusted servers undertake these duties on a simultaneous basis. This paper addresses these issues by using an integrity checking system that is founded on a cryptographic hash function. The system includes low complexity mathematic operations and the key chain method. It assists semi-trusted servers to conduct integrity checking devoid of local replication of the information. This considerably decreases communication and computation overheads. The proposed scheme ensures that the semi-trusted server remains unaware of the subject matter of the information which is stored in the IoT-cloud system for the duration of the integrity checking procedures.

When considering the safeguarding capabilities of the IoT-cloud system and issues of honesty, the results are certainly multifaceted. The developed system tackles every previously identified shortcoming and proposes a lightweight dynamic information honesty public integrity checking service. Here, cryptographic hash function with simple mathematic operations and the key chain method are utilized to minimize the duration of integrity checking. Consequently, this helps semi-trusted servers working as a representative of smart device users to manage information prior to the distribution of such information to the smart device user. In addition, this confirms the integrity of information while avoiding any potentially problematic effects on smart device users. Moreover, this system allows semi-trusted servers to conduct audits in the absence of a local replication of the information. Basically, the IoT-cloud system is unable to create an accurate reaction without the necessary information. A one-time key is implemented to protect against various security threats.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of Related Work, while in Section 3 the Problem Statement is presented. We discuss the Proposed Scheme in Section 4, while the Security Analysis of this scheme is presented in Section 5. The results and analyses are described in Section 6. Finally, Section 7 concludes the paper.

2. Related Works

The provable data possession (PDP) system was initially established by Ateniese et al. [6] and employed an RSA-founded hemimorphic linear authenticators (HLA) to collect a sample of unsystematic file blocks to provide concrete evidence. A different version of this system helped to allow for public integrity checking, but this scheme was disadvantageous as it provided inadequate safeguarding assurance to protect the data from third-party auditors. Curtmola et al. [16] suggested an RSA-founded multiple-replica PDP system that could confirm numerous versions of texts. Curtmola's suggested system could not substantiate information complexities. On the other hand, Ateniese et al. [17,18] suggested a complicated PDP system, but this system could only substantiate a restricted amount of audits. Erway et al. [19,20] suggested a PDP system that encouraged information complexities through a conformation skip catalogue, but this system experienced inadequate safeguarding protection from third-party auditors; it can check the integrity of data blocks of variable sizes, but it is unable to verify the integrity of individual block [21].

The initial sentinel-founded Proof of Retrievability (PoR) system ensured information availability that included a continual relation between information accessibility through codes that rectified mistakes [10]. Shacham and Waters [11] created two compressed PoR systems that were suitable for either private or public integrity checking and were both founded on the characteristics of Boneh–Lynn–Schacham, which confirmed the protection abilities of the PoR system. Nonetheless, these systems provided inadequate safeguarding assurance. Schwarz and Miller [15] created a system that employed distribution erasure to code the information so as to ensure greater accessibility, but this system was inadequate as the server computations and discussion expenses appeared linear within numerous file blocks.

Wang et al. [8,22,23] conducted an examination on the dominant problems within audits on information honesty. One experiment created a third-party auditor system that encouraged information complexities through exploiting the conventional Merkle hash tree. This system [23] included bilinear characteristics that employed public key-founded HLAs that could be unsystematically included into the masking methods for assuring that data was safe-guarded from third-party auditors. Additionally, this system also encouraged batch integrity checking. However, information was not safeguarded within the cloud server. Other researchers such as Chen and Guo [24] created a successful distant information management confirmation system for static information employing RSA-founded modern difficulties techniques. A complex PoR system was created by Zheng and Xu [25] through employing the notion of justness within information dynamics, founded on a genuine information organization known as range-founded 2–3 trees and a consistently increasing characteristic system known as hash-compress-and-sign. Zhu's system [26] encouraged information honesty audits within hybrid clouds, while Hao et al. [27] suggested a system founded on Sebe's procedures [28] for public verifiability that ensured information was safeguarded.

Wei et al. [29,30] considered the issue of outsourced computation security in cloud computing to ensure the IoT-cloud system performed the necessary computations. They proposed a privacy preserving and computational integrity checking method, SecCloud, that uses the commitment-based sampling (CBS) technique [31] and designated verifier signature [32,33] to achieve privacy cheating discouragement, and thus minimize the computational cost in cloud computing. To reduce the communication and computational costs, a batch verification algorithm was suggested for handling various users' requests, concomitantly. They also developed a secure-aware cloud computing environment to implement SecCloud in the real world environment [30]. Li et al. [34] extended this method to support both data integrity and deduplication by using the convergent encryption technique. The existing data integrity checking methods assume that the data owner's secret key is secure, but this is not always true. Yu et al. [35] proposed an integrity checking method to mitigate the damage of the data owner's key exposure

issue, in which the secret key of the data owner is updated by employing the binary tree structure and the pre-order traversal technique.

Based on the principal characteristics of the decentralized blockchain, many authors have studied schema verification of the data integrity based on blockchain. Liu et al. [36] considered that the reliability of the TPA-based framework was not satisfactory and proposed a framework based on blockchain to verify the data integrity. Similarly, Yu et al. [37] suggested a framework to verify the data integrity based on blockchain for cloud storage; the authors used a Merkle tree with a random number for data integrity verification without relying on TPA. Wang et al. [38] suggested a scheme to verify data integrity based on blockchain, but with significant improvement in the security and efficiency of the verification process to avoid excessive reliance on TPA. While existing systems using blockchain to verify data integrity can avoid the issues associated with TPA, they entail extensive communication and processing costs [39].

There are numerous types of systems and some guarantee the validity of remotely stored data within complex safeguarding schemes, but are expensive and thus inappropriate for devices with limited sources. The proposed system has a greater efficiency, is lightweight, and is better protected than the previously discussed systems. It is founded on a cryptographic hash function with the key chain method and can be integrity-checked through a semi-trust server. This scheme [10,11,18,20,22,25] is designed for the data integrity verification of the whole dataset. Unfortunately, it suffers from a very high computational cost resulting from its reliance on bilinear mapping, so it cannot guarantee that the data validation results are entirely correct [39,40]. Thus, this study proposes that the suggested system is more suitable than its predecessor in terms of smart devices with modest sources or capabilities. Additionally, the safeguarding evaluation and investigative outcomes reveal that the recommended system is vigorous, safe, and successful. Table 1 illustrates the contrasts between the safeguarding characteristics of numerous systems.

Table 1. Contrasts of information honesty systems.

Parameter	lightweight	IoT-Cloud System	Public Integrity Check	Batch Integrity Checking	Key Management
Our Proposed	Yes	Yes	Yes	Yes	Yes
[6]	No	No	Yes	No	No
[7]	No	No	Yes	Yes	No
[10]	No	No	No	No	No
[11]	No	No	No	No	No
[18]	No	No	No	No	No
[20]	No	No	No	No	No
[22]	No	No	Yes	Yes	No
[37]	No	Yes	Yes	No	No
[38]	No	Yes	No	No	No
[39]	No	Yes	Yes	No	No

3. Problem Statement

3.1. The Traditional Model

In the integrity checking model, there are three functions—the semi trust server, the smart device user, and the IoT-cloud system. The latter provides services to hold the significant volume of data files owned by the smart device user. The role of the semi-trust server is to validate the integrity of the IoT-cloud system's data, as the smart device user does not have this capability.

In the traditional schemes shown in Figure 1, the smart device user generates the public and private parameters, computes the metadata of their files using the public parameters, and signs the metadata through employing private parameters. The smart device user also transmits file and metadata to the IoT-cloud system and metadata to the semi-trust server. The smart device user sends a request to the semi-trust server to confirm that the data have been stored in the IoT-cloud system. The semi-trust server generates a challenge request and sends it to the semi-trust server.

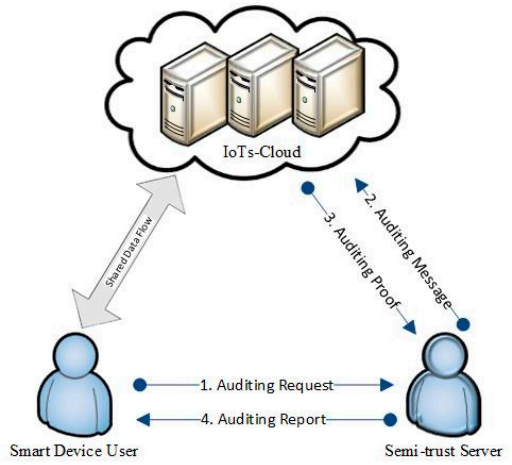

Figure 1. Architecture of the traditional scheme.

The IoT-cloud system receives the challenge request and begins calculating the verification response through employing the parameters of the challenge request for the metadata that is stored in the cloud. The IoT-cloud system then sends the verification response to the semi-trust server. The semi-trust server receives the verification response and confirms the validity of the response by using a bilinear map. Finally, the semi-trust server sends a report to the smart device user regarding the data that are stored in the cloud. The total computation cost of the bilinear map scheme includes numerous modular exponential operations that occur when one file of data is processed. This computation is extremely costly, particularly as substantial information is being audited. However, due to the large number of data tags, their integrity checking protocols will incur a heavy storage overhead on the server. Furthermore, the public-integrity-checking-scheme-based bilinear map incurs a heavy computation cost to the auditor, which makes the integrity checking system inefficient [21].

3.2. Our Proposed Model

The mechanism of our proposed scheme is different from the mechanisms of the traditional scheme. For example, the smart device user generates an owner key that is unique for each smart device user, and sends the encrypted file and owner key to the semi-trust server. Practically, our scheme only requires one round of communication between the smart device user and semi-trust server side, while others require multiple rounds. Thus, our work reduces the communication cost as much as possible.

The upload stage begins once the semi-trust server has received the file and owner key from the smart device user. During this phase, the semi-trust server computes the metadata for the file by generating two keys by using a cryptographic hash function. One of these keys is a challenge key while the other is a verification key. The semi-trust server uses the challenge key with the file to generate the metadata. Finally, the semi-trust server sends the file and metadata to the IoT-cloud system and deletes the file from its local storage.

The semi-trust server employs the integrity checking stage to verify the integrity of the out-sourced data. This process begins with the semi-trust server sending a challenge

key to the IoT-cloud system. The IoT-cloud system computes the audit key using the metadata with the challenge key of the file and sends the audit key to the semi-trust server. The semi-trust server receives the audit key and checks the verification key. If these two keys match, then the file is stored correctly. Figure 2 shows the mechanism of the proposed scheme.

To reduce the integrity checking costs, this paper proposes a scheme based on the cryptographic hash function, key chain method, and low complexity mathematic operation. This scheme would be cost-effective and considerably more efficient than the bilinear map-based schemes.

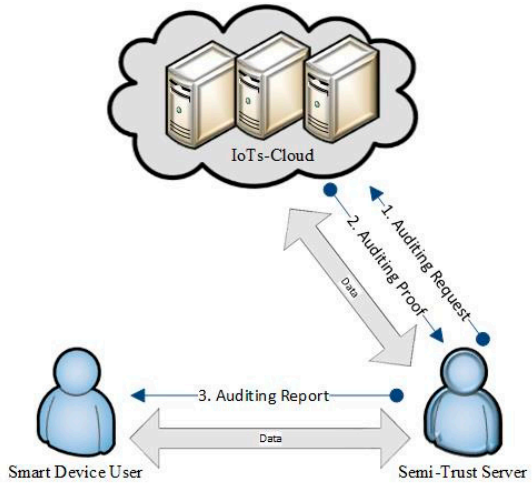

Figure 2. Architecture of the proposed scheme.

3.3. The Threats Model

Instead of being trustworthy, the semi-trust server should be regarded as untrustworthy. It behaves in a trusted way for public integrity checking, but inspects the data it receives. Hence, it may initiate the following attacks [11]:

- Replay attack. The server may generate the proof from the previous proof or other information, without retrieving the actual file.
- Forge attack. The server may forge the integrity checking key of the file and deceive the semi-trust server if the semi-trust server has reused the same challenge key of the same file.
- Replace attack. The server may choose another valid and uncorrupted pair of file and integrity checking parameter to replace the challenged of file, if it has already discarded the current file.

3.4. Design Objectives

In order to address the previously discussed issues, the system employed within this paper attains several safeguarding and execution assurances. For example:

- Public integrity checking enables a semi-trust server to confirm the validity of information stored on the IoT-cloud system as requested, and does not require that they access a replica of the information in its entirety or partially, or have any further online consequences for smart device users.
- Storage suitability guarantees that the IoT-cloud system is not contamination by revealing positive results in the semi-trust server's audit without storing the smart device users' information.
- Additionally, bath integrity checking helps a semi-trust server that has confident and positive integrity checking abilities to handle numerous integrity checking as-

signments at the same time that are likely from numerous and varied smart device users.
- Semi-trust servers are lightweight, which helps them to complete audits with the lowest level of discussion or calculation expenses.

4. Proposed Scheme

This section explains the suggested public integrity checking technique for the IoT-cloud system and includes a thorough information outsourcing resolution that also applies for confirming information integrity methods. This paper explains the primary system and confirms the degree that it encourages semi-trust servers to perform batch integrity checking as a representative and on numerous files. This section also explores the generalization of the public integrity checking system and its encouragement of dynamics information.

4.1. Definitions and Framework

This paper employs explanations that are identical to the explanations presented within formerly suggested systems within the circumstances of remote data integrity confirmation and utilizes guidelines for the public integrity checking system [6,10,11].

The public integrity checking system is composed of the four algorithms of KeyGen, SigGen, GenProof, and VerifyProof. KeyGen can be defined as the primary creation algorithm that smart device users' conduct and that helps semi-trust servers to establish the system, while semi-trust servers employ SigGen to confirm the metadata. The semi-trust server's procedures for confirming the metadata can be composed of digital characteristics. Additionally, the IoT-cloud system utilizes GenProof to confirm accurate information storage and semi-trust servers employ VerifyProof to confirm the audit. The suggested system is composed of three stages: setup, upload, and audit.

- The setup stage involves the smart device user starting the confidential guidelines through conducting the KeyGen and encrypting raw data file F^* to F through the elliptic curve cryptography (ECC).
- The upload stage involves the semi-trust server utilizing the information's confidential guidelines through completing the KeyGen and prior information procedures F through SigGen, which creates metadata. The semi-trust server stores the information F and metadata on the IoT-cloud system and removes the local replica. As part of the pre-processing, the user may alter data file F; in such a case, the user should notify the semi-trust server.
- The integrity check stage involves the semi-trust server submitting an audit appeal to the IoT-cloud system to guarantee that the IoT-cloud system has stored the suitable information file F during the audit. The IoT-cloud system then formulates a response through employing the GenProof and F. Within this stage, the metadata serves as an input. VerifyProof is then employed by the semi-trust server to confirm the reaction.

This paper's regulations presume that the semi-trust server is independent and, as such, does not need to sustain and inform circumstances between audits. Independence is an attractive characteristic for semi-trust servers, especially within a public integrity checking scheme [11]. Broadening the suggested guidelines to include an independent integrity checking scheme is not complicated as the confirmation of metadata only has to be divided in half and kept with the semi-trust server and IoT-cloud system. The plan presented in this paper does not presume further characteristics of the information file.

4.2. Proposed Scheme Details

The details of the proposed design to guarantee that the contents of the data cannot be extracted are outlined below. It is assumed that the smart device user encrypts the raw

data file $F*$ into F using an ECC (ECC is more suitable for IoT) [41] to guarantee that privacy and security are maintained before data are uploaded to the IoT-cloud system.

The new design covers three stages: setting-up, uploading, and integrity checking, and the procedures for doing so are outlined in this document:

4.2.1. Basic Scheme

This section discusses a scenario where a single file is audited.

1. Setup: An owner key Wk is derived by the smart device user:

$$Wk = H(UID||TPAID||CSID) \quad (1)$$

where:

- UID identifies the smart device user and is usually a biometric value. It is also the secret key for the smart device users.
- TPAID identifies the semi-trust server. Using a secure channel, (Wk, F) is sent to the semi-trust server by the smart device user.
- $CSID$ identifies the IoT-cloud system.

2. Upload: The smart device user sends (Wk, F) to the semi-trust server, which starts calculating the metadata of file F. The challenge value of file c, which has a random value, is chosen. Thereafter, Wk and c generate the challenge key Ck, as highlighted:

$$Ck = H(Wk||c) \quad (2)$$

The following step is for the semi-trust server to calculate the challenge hash value Ch, which is derived from F and Ck as follows:

$$Ch = H(F||Ck) \quad (3)$$

The verification key Vk is then generated by the semi-trust server, which selects the random verification value v and then deploys this value along with Wk:

$$Vk = H(Wk||v) \quad (4)$$

The verification key Vk is used to compute the metadata σ using XOR and the challenge hash value Ch as follows:

$$\sigma = Ch\ XOR\ \ Vk \quad (5)$$

After σ is calculated, the IoT-cloud system receives the file and metadata {σ, F} from the semi-trust server. Alternatively, the smart device user receives σ as a receipt from the semi-trust server, which thereafter retains the keys, and removes the file F from its local storage position.

3. Auditing: Using a secure channel, the IoT-cloud system receives a request that includes a challenge key Ck from the semi-trust server. This process validates the integrity of the data that has been outsourced. When it receives the key the IoT-cloud system it initially calculates Cf from F and Ck by employing the cryptographic hash function, as per the following:

$$Cf = H(F||Ck) \quad (6)$$

Lastly, the audit key Ak is generated by the IoT-cloud system, from the metadata σ and Cf as follows:

$$Ak = \sigma\ XOR\ \ Cf \quad (7)$$

The semi-trust server then receives the audit key Ak from the IoT-cloud system, and upon receipt, the semi-trust server compares it to the verification key Vk. When these two keys match each other, it is certain that the file has not been modified and has been stored correctly. The verification equation's validity can thus be proven.

$$Vk = ? Ak \quad (8)$$

The validity of the preceding verification equation can be proven as follows:
Proof of Equation (8).

$$\begin{aligned} Ak &= \sigma \; XOR \;\; Cf \\ &= \sigma \; XOR \;\; H(F||Ck) \\ &= \sigma \; XOR \;\; Ch \\ &= Ch \; XOR \;\; Vk \; XOR \;\; Ch \\ &= 1 \; XOR \;\; Vk \\ Ak &= Vk \end{aligned}$$

4.2.2. Support Batch Integrity Checking

The creation of a public integrity checking system within the IoT-cloud system calculations allows the semi-trust server to simultaneously juggle numerous integrity checking assignments for various file appeals. Individual integrity checking duties can be cumbersome and unsuccessful for the semi-trust server. The individual auditing of these tasks for the semi-trust server can be tedious and inefficient. With y auditing delegations on y distinct data files. The semi-trust server can then compute two keys, such as Vk and Ck, for numerous files, which might cause the IoT-cloud system to try remove every file except for the one file required for confirmation. In order to address this issue, the semi-trust server creates various confirmation keys Vk for different files through incorporating the file identity value $FILEID$ to the following formula:

$$VK_i = H(Wk||v||FILEID) \quad (9)$$

To confirm the files' integrity, the semi-trust server submits a single key Ck appeal and i where $\{0 \le i \le y\}$ to identify the file throughout the integrity checking process. Furthermore, the semi-trust server can submit $\{i\}$ with an identical key Ck_i in a single appeal for numerous files that are to be audited together. The IoT-cloud system then returns Ak_i for another file.

4.2.3. Dynamic File Support

Within the IoT-cloud system calculations, smart device users can regularly retrieve and modernize external information for an array of reasons. Therefore, encouraging information dynamics within public integrity checking systems is essential [17,42,43]. An encouraging information dynamic system should involve the following stages. To start the process:

- The smart device user submits an appeal for the file to the semi-trust server. The semi-trust server then obtains the appeal and passes it on to the IoT-cloud system.
- The IoT-cloud system then transmits the file to the semi-trust server and the semi-trust server passes it on to the smart device user.
- The smart device user then alters the file, returns the file to IoT-cloud system, and transmits the altered file to the semi-trust server including the information version parameter ver. Within this context, ver has the value of 0 or 1:
  - If the ver has a value of 0, this indicates that the file has not been altered.
  - If the ver has a value of 1, this indicates that it has been altered.
- Upon receiving the file, the semi-trust server considers the ver value to identify the next steps. When the ver has a value of 0, then the file has not been altered and no further steps are required. However, when the ver has a value of 1:
  - The semi-trust server must create new keys (Ck, Vk) for the altered file and then re-calculates Ch and σ.

- Once this has been completed, the semi-trust server submits a file appeal to the IoT-cloud system to substitute the new file for the old file and the new σ for the old σ.

4.2.4. Support for Continuous Integrity Checking

The prior section only explored one repetition of an audit. Once an audit has occurred, the IoT-cloud system possesses two keys (*Ck* and *Ak*) and, most importantly, *Ak*. The IoT-cloud system then removes a file to retain *Ak*. Then, when the semi-trust server submits a different appeal for an audit, the IoT-cloud system sends *Ak* and does complete new calculations. A key chain method is incorporated into the suggested system to address this issue. Our scheme generates fewer parameters, unlike traditional schemes. Within the key chain, the keys *Ck* and *Vk* are transfigured through the below equation.

The semi-trust server, for a single file, broadens both keys *Ck* and *Vk* to transform them into n keys ($0 \le j \le n$) through the following equations:

$$\text{If } j = 0, \quad Ck_0 = H(W\,k \,||\, c)$$

$$V\,k_{i0} = H(W\,k \,||\, v \,||\, F\ ILEID)$$

$$\text{If } j > 0, \quad Ck_j = H(W\,k \,||\, c \,||\, Ck_{j-1})$$

$$V\,k_{ij} = H(W\,k \,||\, v \,||\, F\ ILEID \,||\, Vk_{i(j-1)}) \qquad 0 \le j \le n$$

The semi-trust server repeatedly executes the upload phase by employing the below key chain method. Ch_j is identified from *F* and Ck_j through the following equation:

$$Chj = H(F \,||\, Ckj) \tag{10}$$

The metadata files are then calculated through the *XOR* function:

$$\sigma_j = Ch_j \quad XOR \quad V\,k_{ij} \tag{11}$$

By this stage, the semi-trust server has obtained the *n* metadata $\varphi = \{\sigma j\}$, $0 \le j \le n$ for one file at one-time and transmits φ and *F* to the IoT-cloud system.

Throughout the integrity checking process, the semi-trust server unsystematically selects a single Ck_j and transmits this value to the IoT-cloud system, which the IoT-cloud system uses to calculate Cf_j through the below equation:

$$Cf_j = H(F \,||\, Ck_j) \tag{12}$$

The IoT-cloud system then creates the audit key Ak_j with the below equation:

$$Ak_j = \sigma_j \quad XOR \quad Cf_j \tag{13}$$

The IoT-cloud system then transmits the audit key Ak_j to the semi-trust server, and the semi-trust server keeps this key and contrasts it to Vk_{ij} to guarantee that the file has been stored accurately. Theorem 3 explicates how this is accomplished while protracting for multi integrity checking.

5. Security Analysis

This section initially establishes a security scheme for the proposed system and then validates the system's security.

5.1. Security Modelling

There are four algorithms within the suggested system: *KeyGen* (), *SigGen* (), *GenProof* (), and *VerifyProof* (). This section will provide an explanation of these algorithms.

- *KeyGen* (*Wk*) produces both *Vk* and *Ck*. Additionally, *KeyGen* creates a key chain for one-time use.

- *SigGen* (*Vk*, *Ck*, *F*) generates the group of metadata φ.
- *GenProof* (*Ck*, σ, *F*) generates the poof of data storage *P*. Practically, *P* is the auditing key *Ak*.
- *VerifyProof* (*Ak*, *Vk*) is a comparison algorithm. If *Ak* = *Vk*, then the data are correctly stored.

This paper evaluates the proposed system's security as an alternative to Shacham and Waters' scheme [11]. Shacham and Waters' scheme encourages public verifiability and its essential goal is to attain evidence confirming the system's accessibility. However, its characteristics indirectly guarantee that rivals are unable to produce confirmed evidence of the integrity of file *F* to any degree, which suggests errors in the counter arguments.

The proposed system's security varies from Shacham and Waters [11] proposed security scheme for numerous reasons. For example:

- Within Shacham and Walters' security scheme, *F* is not included in the *GenProof* and, thus the IoT-cloud system is able to erase files and storage metadata σ to confirm its validity. In order to correct this error, this paper's proposed system incorporates F into *GenProof*.
- Within Shacham and Waters' proposed security scheme, the challenge is incorporated into the *GenProof* and *VerifyProof*, which allows the IoT-cloud system to erase *F* and uses σ to maintain *proof* (*P*) once the audit has been effectively completed. Therefore, when the semi-trust server requests conducting another audit on the file, the IoT-cloud system only provides the semi-trust server with *P*. In order to correct this error, in our proposed scheme, *Ck* is changed every time; consequently, *P* is also changed every time. The scheme of Shacham and Waters [11] was not designed for IoT, especially for smart devices with a minimal size; meanwhile, the proposed system was customized to be compatible with the IoT-cloud system and smart devices. The experimental results' proficiency evaluation section proves the compatibility of our work for smart devices in terms of consuming time and communication cost.

5.2. Security Proofs

The underlying principle in the scheme that we are proposing is that in order for the IoT-cloud system to generate a correct response, which is then received by the semi-trust server, the data must be stored correctly.

Definition 1. *It is argued that offence algorithms, such as GenProof*(), should not be held by the IoT-cloud system. This is because these algorithms might allow smart device users to gain accurate evidence of validation and to compute the outcome employed in VerifyProof().*

Hence, in Definition 1, we show that the IoT-cloud system will not create a correct proven result by using *GenProof* (σ, *Ck*). Next, we demonstrate that the verification algorithm *VerifyProof* (*Ak*, *Vk*) will fail in the absence of the correctly proven result.

Theorem 1. *One argues that the IoT-cloud system is unable to create an accurate reaction in the absence of a file.*

Proof of Theorem 1. Confirms that the IoT-cloud system performs sufficiently with a consistent and correct methodology without deviating from the dictated procedures. Nonetheless, the IoT-cloud system might ignore or intentionally remove infrequently retrieved information files that are recovered and owned by typical smart device users. Furthermore, the IoT-cloud system can conceal information, deceptions, or manipulation created by hackers or Byzantine malfunctions in order to sustain its status [16]. For example, the IoT-cloud system possesses file σ without possessing file *F*. As a result, the

IoT-cloud system has σ. However, the IoT-cloud system cannot create *Cf* without *F*. The offense algorithm *GenProof**(σ, *Ck*) cannot create *Ak* as *Ak* must be calculated using the equation $Ak = \sigma$ *XOR Cf*. *Cf* can only be calculated through $Cf = H(F||Ck)$, which means that the IoT-cloud system is unable to create an accurate reaction if it does not have the right file. □

Theorem 2. *Argues that in the absence of an accurate Ak, the verification algorithm VerifyProof (P, sk) will not be able to generate the proof correctly.*

Proof of Theorem 2. Suggests that the verification algorithm *VerifyProof* (*P*, *sk*) contrasts the audit key *Ak* and the *Vk* key. The semi-trust server currently stores *Vk*, but the IoT-cloud system transmits *Ak*. When the IoT-cloud system is unable to transmit an audit key *Ak*, the algorithm outcomes will be incorrect. When an attacker acquires *Ak* during transmission, the key *Ak* can no longer be employed as it is a one-time key. □

Theorem 3. *Argues that the suggested encouragement system is a one-time key.*

Proof of Theorem 3. Argues that once the audit is complete, the IoT-cloud system possesses two keys, *Ck* and *Ak*, but especially *Ak*. In order to maintain *Ak*, the IoT-cloud system can erase a file. Therefore, when the semi-trusted server sends the audit of this specific file again, the IoT-cloud system will send the audit key *Ak* file. Thus, the proposed system uses a one-time key generated by the key chain method. The semi-trust server for a single file alters both of the keys created through the key chain method (*Ck* and *Ak*) to create n keys through the following equations:

$$\text{If } j = 0, \quad Ck_0 = H(W\,k||c)$$

$$V\,ki_0 = H(W\,k||v||F\ ILEID)$$

$$\text{If } j > 0, \quad Ck_j = H(W\,k||c||Ck_{j-1}),$$

$$V\,k_{ij} = H(W\,k||v||F\ ILEID||V\,k_{i(j-1)}) \qquad \text{when } (0 \le j \le n)$$

The semi-trust server repeatedly executes the upload phase through the key chain method. Within this stage, the semi-trust server possesses the n^{th} metadata ($\varphi = \{\sigma j\}$, $0 \le j \le n$) for a single file and transmits φ and *F* to the IoT-cloud system. Within the integrity checking process, the semi-trust server unsystematically selects a single *Ckj* to transfer to the IoT-cloud system. The IoT-cloud system then calculates *Cfj* and uses σj to create the audit key *Akj*, which is transmitted to the semi-trust server. The semi-trust server then puts *Akj* aside to contrast it to *Vkij* to guarantee that the file has been accurately stored. □

Theorem 4. *Argues that the suggested system can supply key management.*

Proof of Theorem 4. The suggested system argues that the smart device user obtains their key *Wk* ($W\,k = H(UID||T\ P\ AID||CSID)$) and transmits it to the semi-trust server. The semi-trust server then employs the smart device user key *Wk* to create two keys, *Ck* and *Vk*. The opposition key *Ck* is calculated by employing the cryptographic hash function ($Ck = H(W\,k||c)$) and the verification key (*Vk*) *or* ($Vk = H(W\,k||v)$). The semi-trust server then transmits the opposition key (*Ck*) to the IoT-cloud system throughout the integrity checking procedures. The IoT-cloud system then employs the opposition key (*Ck*) to calculate *Cf* ($Cf = H(F||Ck)$). Once *Cf* has been determined, it can be employed to create the audit key *Ak* ($Ak = \sigma$ *XOR Cf*), which is transmitted back to the semi-trust server. Thus, this system can encourage key management. □

Theorem 5. *Argues that the suggested system encourages privacy and confidentiality.*

Proof of Theorem 5. A semi-trusted server can concurrently handle several integrity check tasks within multiple file restore requests using general integrity check procedures. Therefore, a semi-trusted server must keep the information private and confidential for every smart device user. Within the suggested system, the semi-trust server incorporates the file specification value *FILEID* throughout the creation of the verification key *Vk* (*Vki* = *H*(*Wk*||*v*||*F ILEID*)) for every individual file of metadata. □

Theorem 6. *Argues that the proposed system can protect against a replay attack.*

Proof of Theorem 6. The hackers conduct repeated attacks by eavesdropping on the Ak (integrity checking key) transmission between the IoT-cloud system and the semi-trust server. When the transmission is complete, the hacker employs the *Ak* (integrity checking key) to replicate the original copy of the IoT-cloud system when the semi-trust server re-requests to audit the file. This flaw within the proposed system can be solved by altering the integrity key *Ak* after every integrity checking procedure. In addition, *Ak* is a one-time key; thus, the hacker is unable to repeatedly circulate the integrity checking key. As a result, the hacker will not be able to complete this type of attack due to system protection measures. □

6. Proficiency Evaluation

Public auditing is a very demanding resource in terms of communication costs, computational resources, and memory space. This paper assesses the suggested system's proficiency, durability, and validity by evaluating the system based on the intended goals. Data integrity checking for limited source devices is a resource-demanding service in terms of computational and communication costs. This study compared the computation and communication costs of the proposed system and the existing audit system proposed by Yang and Jia [7].

The system's initial abilities were quantified by calculating the uploading duration and integrity checking stages. The discussion expenses were quantified based on the overall extent of the integrity checking appeal in contrast with the bilinear map-founded system [7].

6.1. Computation Cost of the Uploading Stages

According to Section 4, the semi-trust server needs to generate a challenge key *Ck*, verification key *Vk*, and metadata σ for each smart device user separately. This study evaluated the generation of challenge key *Ck*, verification key *Vk*, and metadata σ for the system. Figure 3 indicates that the parameter generation cost is proportional to the number of smart device users. Note that the system upload cost is one-time and will not influence the real-time performance of the integrity checking.

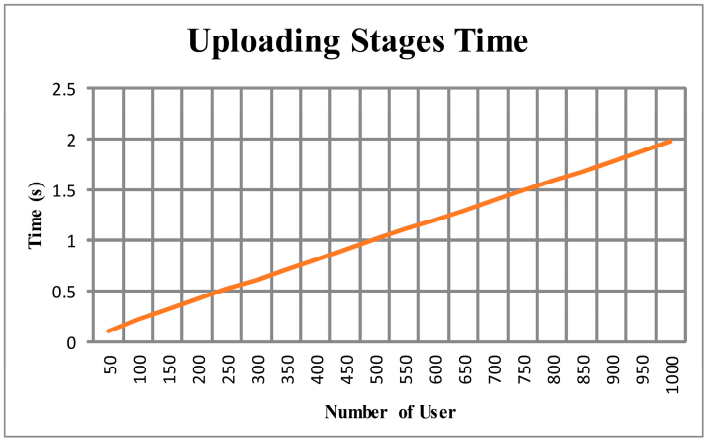

Figure 3. Computation cost of the uploading stages in the semi-trust server.

6.2. Computation Cost of the Integrity Checking Stages

6.2.1. Computation Cost of the Semi-Trust Server

This stage contrasts the number of challenges, the number of smart device users, and the number of clouds with the semi-trust server's computation time. In the first instance, the semi-trust server's computation time was assessed depending on the volume of challenged files in the single smart device user and the single IoT-cloud system. The size of the challenge data was calculated to be equal to 1000 kilobytes. If the number of challenges reached 1000, then the size of the challenges data was equal to 1000 kilobytes. It can demonstrate that there is a linear relationship between the challenge number and the computation cost of the semi-trust server. In Figure 4, it is clear that when handling large volumes of challenged data, the computational cost is higher in Yang's solution [7] than in the proposed solution; this is because the challenge size key in the proposed system is (*Ck* = 128 *bit*) and is independent of the challenged data size. In contrast with Yang [7], which depends on the size of the data, when the challenge data increases, so does the challenge key. Consequently, a highly efficient scheme has been improved in the proposed work. It can also further reduce the computation complexity.

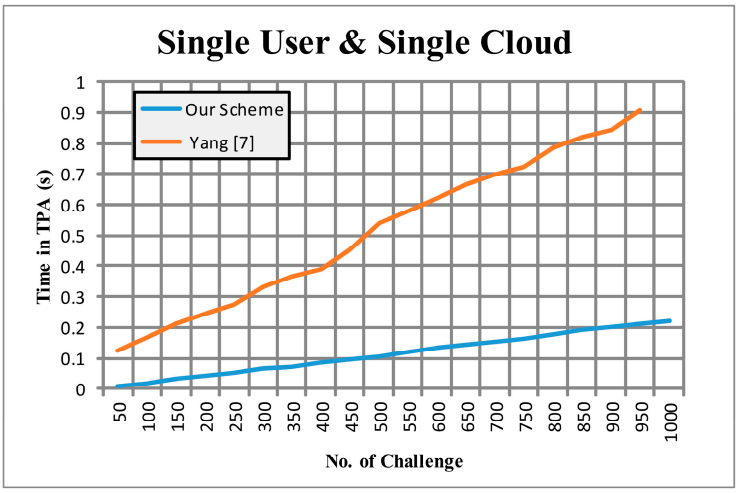

Figure 4. Computation cost on the semi-trust server with the single smart device user and single IoT-cloud system.

In the proposed system, the integrity of vast amounts of data (e.g., petabytes) is protected by applying the sampling integrity checking method; this is in contrast with the function of an IoT-cloud system. Service level agreements set the frequency and sample sizes. During the tests, this study concluded that about 0.223 s are required to audit 1000 challenges. However, the computational time required was likely to be small because the PC used in this study lacked the computational ability of the semi-trust server and the IoT-cloud system. This result confirms that in large-scale IoT-cloud systems, the proposed scheme for checking the system integrity is practical.

Next, this review examined the computation cost of the semi-trust server of the multi-cloud batch integrity checking scheme based on the number of challenged clouds. Figure 5 shows that Yang's solution [7] incurred a higher computational cost for the semi-trust server than the suggested system. This is particularly true when the large-scale IoT-cloud system contains a large number of the IoT-cloud system and results because of complex algorithms. Yang used the bilinear map, which is a complicated method and takes time to obtain results, unlike the proposed system, which uses lightweight operations, namely the cryptographic hash function, XOR, and concatenation, and calculates the results quickly. These were used in Yang's scheme [7] to support batch integrity

checking for multi-cloud. In the proposed scheme, where one smart device user was selected for every ten clouds, the dependency was in the number of challenges.

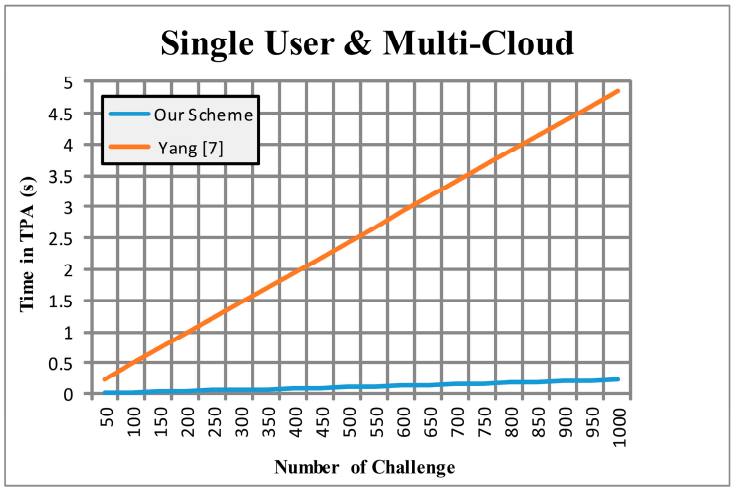

Figure 5. Computation cost on the semi-trust server with a single smart device user and multi-cloud.

Lastly, the general integrity checking protocol used to support the multi-owner batch integrity checking in Yang's scheme [7] was compared (from the perspective of the computation cost of the semi-trust server) with this study's multi-smart device user batch integrity checking scheme. As can be observed in Figure 6, the computational costs could be significantly reduced when batch integrity checking for multiple owners was undertaken. Therefore, this study's solution was considerably more efficient and cost-effective than Yang's scheme [7]. This is due to using the lightweight cryptographic hash function, XOR, and concatenation operations that do not require significant time or resources to obtain results. In addition, the scheme proposed in Section 4 is short and does not contain complex operations. Conversely, Yang used the Bilinear Map method, a complex algorithm that takes a long time to obtain results in a large-scale IoT-cloud system where there may be millions or billions of smart device data users. This delay is a factor even when the number of smart device data users is 1000 or less.

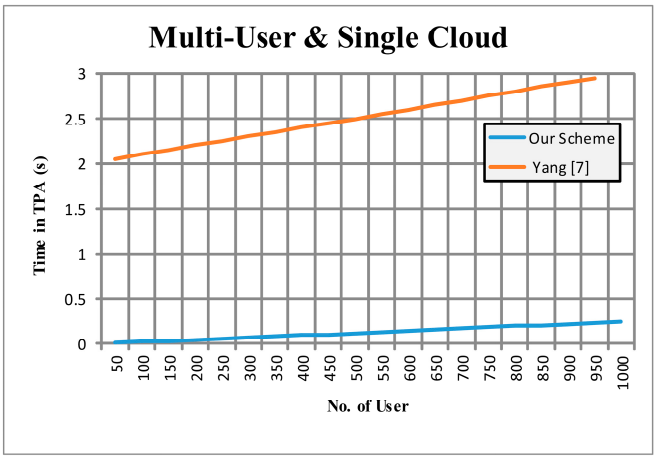

Figure 6. Computation cost on semi-trust server with multi smart device user and single cloud.

6.2.2. Computation Cost of the Cloud

This study looked at the number of files sent to one smart device user (Figure 7) and the overall amount of data for smart device users (Figure 8) and compared the computa-

tion cost of the IoT-cloud system. Compared with the proposed system, it is clear that Yang's scheme [7] incurred a higher computation cost for the IoT-cloud system.

Furthermore, in Yang's solution [7], a high computation cost time was observed on the IoT-cloud system side; this occurred due to an attempt to reduce the computation cost of the semi-trust server by moving the computing loads of the integrity-checking function to the IoT-cloud system.

In contrast, the proposed solution led to a lower computation cost time in the IoT-cloud system side. This was due to its adoption of low-cost operations to compute the proof of data in the IoT-cloud system.

Consequently, compared with Yang's scheme [7] which used the Bilinear Map method, the proposed solution using lightweight and secure operations offered a low computation time and cost for both the semi-trusted server and IoT-cloud side.

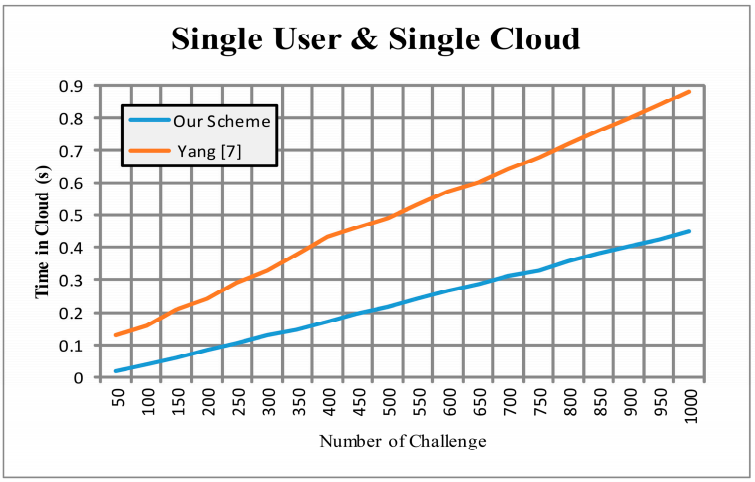

Figure 7. Computation cost for the IoT-cloud system with a single smart device user and single cloud.

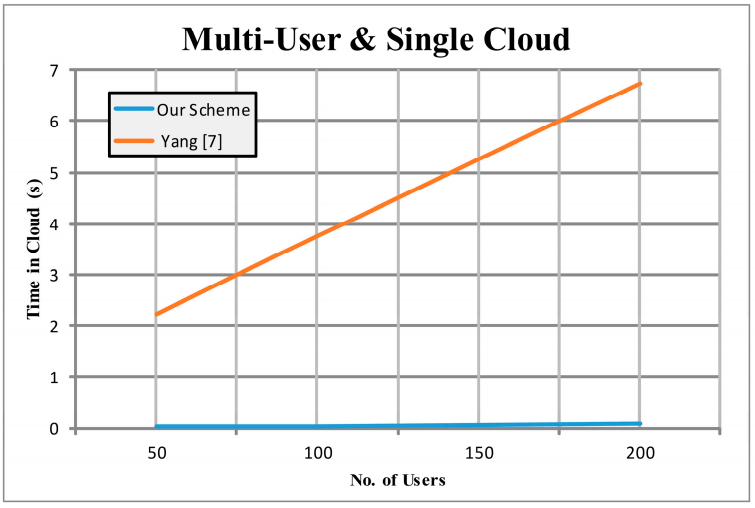

Figure 8. Computation cost for the IoT-cloud system with a multi smart device user and single cloud.

6.3. Cost of Communication

The communication cost between the semi-trust server and the IoT-cloud system server, which consists of the challenge and the proof, is further examined in this section. This study reviewed the batch integrity checking between smart device users and IoT-cloud systems. For the sake of comparison, it was assumed that the number of challenged files from each smart device user using different IoT-cloud systems was the same.

When reviewing Yang's scheme [7] on each of the IoT-cloud system servers, it was clear that there was a linear relationship between the communication cost of the challenge data and the number of challenged data blocks from each smart device user. There can be many data blocks in a large-scale IoT-cloud system, which means that Yang's system [7] may result in a high communication cost. Alternatively, in the proposed solution the communication cost has a linear relationship only to the number of challenge requests that the semi-trust server sends to the IoT-cloud system at CK = 128 bits.

Therefore, the results prove that the proposed solution is more cost-effective than Yang's scheme [7].

7. Conclusions

This paper presents a rigorous system for ensuring the integrity, confidentiality, and validity of information that is stored in the cloud. The system uses mathematic operations and a cryptographic hash function to protect information and avoid security breaches in the cloud. A semi-trust server cannot uncover or retrieve the subject matter of the information contained within the file. XOR and the cryptographic hash function are employed to evaluate the integrity and accuracy of the information in the cloud.

The semi-trust server and IoT-cloud system can execute every function in the suggested system. The semi-trust server transmits records relating to the file's conditions only to the smart device user. Furthermore, the suggested system encourages dynamic data, batch integrity checking, and integrity checking via the key chain technique. Public integrity checking can be accomplished in the suggested system through the utilisation of a semi-trust server. The semi-trust server possesses the skills and abilities that the smart device user does not. The semi-trust server is in charge of checking the integrity of the information in the IoT-cloud system as a representative of the smart device user. It cannot remove information that is stored in the IoT-cloud system throughout the integrity checking procedure, which reduces smart device users' cumbersome and costly auditing tasks and eliminates their concerns regarding breaches in the IoT-cloud system resulting from external information.

Author Contributions: Conceptualization, V.O.N.; Methodology, Z.A.H. and M.A.H.; Software, H.A.A., M.A.H. and I.Q.A.; Investigation, V.O.N.; Resources, J.M.; Data curation, H.A.A.; Writing—original draft, Z.A.H.; Writing—review & editing, I.Q.A.; Supervision, Z.A.A.; Project administration, Z.A.A.; Funding acquisition, J.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by university-enterprise cooperative R&D project of SZTU (grant no.20221061030001).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: All individuals included in this section have consented to the acknowledgement.

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

References

1. Li, Y.; Li, Y.; Zhang, K.; Ding, Y. Public integrity auditing for dynamic group cooperation files with efficient user revocation. *Comput. Stand. Interfaces* **2022**, *83*, 103641.
2. Ogonji, M.M.; Okeyo, G.; Wafula, J.M. A survey on privacy and security of Internet of Things. *Comput. Sci. Rev.* **2020**, *38*, 100312.
3. Hussien, Z.A.; Abduljabbar, Z.A.; Hussain, M.A.; Al Sibahee, M.A.; Lu, S.; AL-Asadi, H.A. An efficient and secure scheme for dynamic shared data in cloud. In Proceedings of the 3rd International Conference on Computer Science and Application Engineering, Sanya, China, 22–24 October 2019.
4. Kuldeep, G.; Zhang, Q. Multi-class privacy-preserving cloud computing based on compressive sensing for IoT. *J. Inf. Secur. Appl.* **2022**, *66*, 38–60.

5. Belal, M.M.; Sundaram, D.M. Comprehensive review on intelligent security defences in cloud: Taxonomy, security issues, ML/DL techniques, challenges and future trends. *J. King Saud Univ. -Comput. Inf. Sci.* 2022, *in press*.
6. Ateniese, G.; Burns, R.; Curtmola, R.; Herring, J.; Kissner, L.; Peterson, Z.; Song, D. Provable data possession at untrusted stores. In Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07), New York, NY, USA, 28 October 2007.
7. Yang, K.; Jia, X. An efficient and secure dynamic auditing protocol for data storage in cloud computing. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *24*, 1717–1726.
8. Sengupta, B.; Dixit, A.; Ruj, S. Secure cloud storage with data dynamics using secure network coding techniques. *IEEE Trans. Cloud Comput.* **2022**, *10*, 2090–2101.
9. Majumdar, S.; Chawla, G.S.; Alimohammadifar, A.; Madi, T.; Jarraya, Y.; Pourzandi, M.; Wang, L.; Debbabi, M. ProSAS: Proactive security auditing system for clouds. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 2517–2534.
10. Juels, A.; Kaliski, B.S. Pors: Proofs of retrievability for large files. In Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07), New York, NY, USA, 28 October 2007.
11. Shacham, H.; Waters, B. Compact proofs of retrievability. *J. Cryptol.* **2012**, *26*, 442–483.
12. Naor, M.; Rothblum, G.N. The complexity of online memory checking. *J. ACM* **2009**, *56*, 1–46.
13. Feng, Y.; Mu, Y.; Yang, G.; Liu, J.K. A new public remote integrity checking scheme with user privacy. In Proceedings of the 20th Australasian Conference on Information Security and Privacy (ACISP '15), Brisbane, QLD, Australia, 1 January 2015.
14. Garg, N.; Bawa, S. Comparative analysis of cloud data integrity auditing protocols. *J. Netw. Comput. Appl.* **2016**, *66*, 17–32.
15. Schwarz, T.S.J.; Miller, E.L. Store, forget, and check: Using algebraic signatures to check remotely administered storage. In Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS '06), Lisboa, Portugal, 4–7 July 2006.
16. Curtmola, R.; Khan, O.; Burns, R.; Ateniese, G. MR-PDP: Multiple-replica provable data possession. In Proceedings of the 28th International Conference on Distributed Computing Systems (ICDCS '08), Beijing, China, 17–20 June 2008.
17. Ateniese, G.; Kamara, S.; Katz, J. Proofs of storage from homomorphic identification protocols. In Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 09), Tokyo, Japan, 6–10 December 2009.
18. Ateniese, G.; Di Pietro, R.; Mancini, L.V.; Tsudik, G. Scalable and efficient provable data possession. In Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks (SecureComm '08), New York, NY, USA, 22 September 2008.
19. Erway, C.; Küpçü, C.A.; Papamanthou, C.; Tamassia, R. Dynamic provable data possession. In Proceedings of the 16th ACM Conference on Computer and Communications Security, New York, NY, USA, 9–13 November 2009.
20. Erway, C.C.; Küpçü, A.; Papamanthou, C.; Tamassia, R. Dynamic provable data possession. *ACM Trans. Inf. Syst. Secur.* **2015**, 17, 1–29.
21. Sookhak, M.; Gani, A.; Khan, M.K.; Buyya, R. WITHDRAWN: Dynamic remote data auditing for securing big data storage in cloud computing. *Inf. Sci.* **2017**, *380*, 101–116.
22. Wang, C.; Wang, Q.; Ren, K.; Lou, W. Privacy-preserving public auditing for data storage security in cloud computing. In Proceedings of IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010.
23. Wang, C.; Wang, Q.; Ren, K.; Lou, W. Ensuring data storage security in cloud computing. In Proceedings of the 17th International Workshop on Quality of Service (IWQoS), Charleston, SC, USA, 13–15 July 2009.
24. Chen, L.; Guo, G. Comparative analysis of cloud data integrity auditing protocols. *Int. J. Digit. Content Technol. Its Appl.* **2011**, *5*, 43–50.
25. Zheng, Q.; Xu, S. Fair and dynamic proofs of retrievability. In Proceedings of the First ACM Conference on Data and Application Security and Privacy (CODASPY '11), New York, NY, USA, 21 February 2011.
26. Zhu, Y.; Wang, H.; Hu, Z.; Ahn, G.J.; Hu, H.; Yau, S.S. Efficient provable data possession for hybrid clouds. In Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10), New York, NY, USA, 4 October 2010.
27. Hao, Z.; Zhong, S.; Yu, N. A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. *IEEE Trans. Knowl. Data Eng.* **2011**, *23*, 1432–1437.
28. Sebé, F.; Domingo-Ferrer, J.; Martinez-Balleste, A.; Deswarte, Y.; Quisquater, J.J. Efficient remote data possession checking in critical information infrastructures. *IEEE Trans. Knowl. Data Eng.* **2008**, *20*, 1034–1038.
29. Wei, L.; Zhu, H.; Cao, Z.; Jia, W.; Vasilakos, A.V. SecCloud: Bridging secure storage and computation in cloud. In Proceedings of the IEEE 30th International Conference on Distributed Computing Systems Workshops, Genova, Italy, 21–25 June 2010.
30. Wei, L.; Zhu, H.; Cao, Z.; Dong, X.; Jia, W.; Chen, Y.; Vasilakos, A.V. Security and privacy for storage and computation in cloud computing. *Inf. Sci.* **2014**, *258*, 371–386.
31. Du, W.; Jia, J.; Mangal, M.; Murugesan, M. Uncheatable grid computing. In Proceedings of the 24th International Conference on Distributed Computing Systems, Tokyo, Japan, 24–26 March 2004.
32. Huang, Q.; Yang, G.; Wong, D.S.; Susilo, W. Efficient strong designated verifier signature schemes without random oracle or with nondelegatability. *Int. J. Inf. Secur.* **2011**, *10*, 373.
33. Zhang, J.; Mao, J. A novel id-based designated verifier signature scheme. *Inf. Sci.* **2008**, *178*, 766–773.
34. Li, J.; Li, J.; Xie, D.; Cai, Z. Secure auditing and deduplicating data in cloud. *IEEE Trans. Comput.* **2016**, *65*, 2386–2396.

35. Yu, J.; Ren, K.; Wang, C.; Varadharajan, V. Enabling cloud storage auditing with key-exposure resistance. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 1167–1179.
36. Liu, B.; Yu, X.L.; Chen, S.; Xu, X.; Zhu, L. Blockchain based data integrity service framework for IoT data. in Proceedings of the 2017 IEEE International Conference on Web Services (ICWS), Honolulu, HI, USA, 25–30 June 2017; pp. 468–475
37. Yue, D.; Li, Y.; Zhang, Y.; Tian, W.; Huang, Y. Blockchainbased verification framework for data integrity in edge-cloud storage. *J. Parallel Distrib. Comput.* **2020**, *146*, 1–14, .
38. Wang, H.; He, D.; Yu, J.; Xiong, N.N.; Wu, B. RDIC: A blockchain-based remote data integrity checking scheme for IoT in 5G networks. *J. Parallel Distrib. Comput.* **2021**, *152*, 1–10.
39. Wang, H.; Lin, Y.; Xiao, F. A lightweight data integrity verification with data dynamics for mobile edge computing. *Secur. Commun. Netw.* **2022**, *2022*, 1870779.
40. Yang, A.; Xu, J.; Weng, J.; Zhou, J.; Wong, D.S. Lightweight and Privacy-Preserving Delegatable Proofs of Storage with Data Dynamics in Cloud Storage. *IEEE Trans. Cloud Comput.* **2021**, *9*, 212–225.
41. Abduljabbar, Z.A.; Jin, H.; Ibrahim, A.; Hussien, Z.A.; Abbdal, M.A.; Zou, D. Privacy preserving image retrieval in IoT-cloud. In Proceedings of the 15th International Conference on Trust, Security and Privacy in Computing and Communications, Tianjin, China, 23–26 August 2016.
42. Chen, B.; Curtmola, R. Robust dynamic remote data checking for public clouds. In Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12), New York, NY, USA, 16 October 2012.
43. Wang, C.; Wang, Q.; Ren, K.; Cao, N.; Lou, W. Toward secure and dependable storage services in cloud computing. *IEEE Trans. Serv. Comput.* **2012**, *5*, 220–232.