Hindawi
Security and Communication Networks
Volume 2020, Article ID 9809874, 13 pages
https://doi.org/10.1155/2020/9809874

Research Article

An Efficient Integrity Verification and Authentication Scheme over the Remote Data in the Public Clouds for Mobile Users

S. Milton Ganesh[1] **and S. P. Manikandan**[2]

[1]*Department of Computer Science and Engineering, University College of Engineering Tindivanam, Tindivanam 604001, Tamilnadu, India*
[2]*Department of Computer Science and Engineering, Jerusalem College of Engineering, Chennai 600100, Tamilnadu, India*

Correspondence should be addressed to S. Milton Ganesh; softengineermilton@gmail.com

Received 23 October 2019; Revised 14 February 2020; Accepted 13 March 2020; Published 5 May 2020

Academic Editor: Cristina Alcaraz

The digitalization of the modern world and its applications seem to be integrated more with the mobile phones than with any other communication devices. Since the mobile phones have become ubiquitous with applications for nearly all users, they have become a preferred choice for uploading the sensitive information to the cloud servers. Though the drive for data storage in cloud servers is implicit due to its pay-per-use policies, the manipulation of the data present in the cloud servers by hackers and hardware failure incidents, as happened in Amazon cloud servers in 2011, necessitates the demand for data verification at regular intervals over the data stored in the remote servers. In this line, modern day researchers have proposed many novel schemes for ensuring the remote data integrity, but they suffer from attacks or overheads due to computation and communication. This research paper provides solutions in three dimensions. Firstly, a novel scheme is introduced to verify the integrity of the data stored in the remote cloud servers in the context of mobile users. The second dimension is that of reducing the computational and communication overheads during the auditing process than the previous works. The third dimension securely authenticates the mobile user during the auditing process and the dynamic data operations such as block modification, insertion, and deletion. Moreover, the proposed protocol is provably secure exhibiting soundness, completeness, and data privacy making it an ideal scheme for implementation in the real-world applications.

1. Introduction

The modern world which is getting more and more digital everyday has mandated the need for data outsourcing to cloud servers through the mobile phones of individual users to corporate offices [1, 2]. Applications like Google Drive, Google App Engine, OneDrive from Microsoft, Google Picasa, Adobe Cloud, Oracle Cloud, Dropbox, Facebook, and other such applications have made an implicitly compelling scenario ranging from layman to the highly resourceful technocrats to make use of the cloud for data storage. Hence, in this digital world, a person who possesses a mobile phone views this vast world as just a small global village where he has access to any information through Internet and cloud storage.

At the same time, cloud storage has its inherent benefits in terms of elasticity, reliability, pay-per-use model, and traffic adjustability during upload, download, and other situations [3]. In such a context, the mobile users who possess powerful processors and random access memory for processing the data need not store the data in their local storage. Once the data is uploaded to the cloud storage, they may be relieved of the burden from maintaining it. But, this flexible data storage comes with its own inherent disadvantages also, as cited in the work [4]. Newer kinds of attacks on the data storage including the cloud servers have become apparent nowadays. If the sensitive data is captured by the hackers, they can use it for mining business patterns, usage patterns which might indeed pave the way for the loss to the

actual data owners. Even the cloud servers may try to hide the fact of data server crashes which will lead to permanent data loss to the data owners.

In order to enable the verification process, first, the user splits the large file to be uploaded into smaller units called file blocks. The user uploads all the blocks to the remote storage area like cloud servers. Later, if the user wants to verify the integrity of the uploaded file, he can do so by making use of some cryptographically verifiable procedure [5–7].

Similarly, to ensure the data verifiability in the cloud servers, multiple schemes have been proposed by authors in their past literatures. In one such scenario, cloud servers were vested with the responsibility of computing the proof of verification based on all of the blocks stored in the cloud storage [8, 9]. In such cases, if the cloud server had to do this computationally intensive work for hundreds of users simultaneously, it may incur a huge computational overhead. On the contrary, authors like Juels and Kalisi in 2007 [10] claimed that the audit task must be done at the cloud user's side. This method will not suit the users who work with computationally constrained battery powered mobile phones.

A pioneering attempt by Ateniese et al. [11] in 2008 claimed that a user who wants to audit the file integrity need not access the entire file stored in the cloud, and also, the user can delegate the audit task to a third party. Some works in this line delegated the verification rights to other parties such as trusted third parties or other such entities. In the works proposed [12–14], a patient who undergoes a treatment from a doctor makes use of electronic health records stored in the cloud storage and also allows the doctor to create the records and store them in the remote storage on behalf of the patient. A recent work proposed by Yu et al. in 2017 [15] seems to be a worthwhile protocol with public auditing capability with efficient computational and storage overheads. Many verifiable schemes have been proposed by Chen et al. in 2019 [16], Peng et al. in 2019 [17], Fujisaki and Okamoto in 1998 [18], Patra et al. in 2015 [19], and others.

Hence, a research work should be able to authenticate a legitimate mobile user during the audit response phase. In this regard, each mobile user should store a unique authentication parameter in the cloud server before sending any audit request. Hence, during audit response, the cloud server is able to successfully authenticate only the legitimate users and able to identify the intruders to abort their audit requests.

One essential fact to be considered during the protocol design is to make it certain that the utmost security with less computational necessity and the robustness of the protocol must be resistant to attackers and hackers. Thus, though there are multiple methods of ascertaining the integrity of the stored file blocks in the cloud servers, each method suffers from one of the problems such as computational burden of cloud servers or data owners or reliability of the verification procedure by third parties done on behalf of the cloud users, lack of authentication procedures for the cloud servers and cloud users. In this research work, a novel method which will avoid the above shortcomings for verification of data stored in the remote servers has been proposed to enable the user who uploads the file to verify the integrity of the same as well.

Unless the data stored in the remote locations are verified thoroughly to the satisfaction of the user with respect to the security assurance, computational, and communicational capabilities, not only these schemes are prone to attacks but also they would be nonoperational for practical use by the mobile user community which constitutes a larger portion of the Internet users. Based on the necessity to address these issues, the contributions of this research work can be highlighted as follows.

1.1. Contributions of This Research Work

(i) A novel scheme to verify the integrity of the remote data and to authenticate the mobile user during the integrity auditing process is introduced

(ii) The presence of the authentication procedure prevents eavesdroppers and hackers from intruding into the system

(iii) The proposed work is resistant to attacks and computationally more efficient than the previous works

(iv) The presence of valid proofs for completeness, soundness, and perfect data privacy makes this a vital contribution for real-world data audits in cloud

1.2. Organization of This Research Work. The organization of this research manuscript is as follows. Section 2 incorporates the much needed recent and old literary works pertaining to the works proposed for remote data integrity verification and showcases the need for the improvements in them. Section 3 provides a quick review of the preliminaries, and a suitable architecture of the protocol proposed in this research work is in Section 4. The subsequent section shows the construction of the proposed protocol with its novel procedures for data integrity verification processes for files and support for data dynamic operations. Section 6 analyses this work in terms of its correctness, soundness, and data privacy during the audit process. In Section 7, the implementation results of the protocol are compared with various schemes, and the results are tabulated. Section 8 concludes this research work.

2. Literature Survey

Latest advancements such as Internet-of-Things (IoT), fog computing, digital transaction with blockchain-based security assurance, smart cities, cloud computing, and other such technologies enable a mobile user to upload sensitive information to the cloud servers for future processing. Many worthwhile schemes have been put forward by researchers and students of various institutions to ensure the correct possession of data stored in the cloud servers.

Wang et al. in 2010 [20] introduced a similar scheme for the proposal of the efficient auditing framework without requiring the user to maintain a local copy of the data

uploaded to the cloud server. This work is resistant to the attacks from the auditor and supports integrity verification for multiple users at the same time. A scheme proposed by Zhu et al. in 2012 [21] enabled the clients to store the files in multiple cloud servers and introduced a scalable integrity verification service with reduced computational and communication complexities based on homomorphic procedures and indexing hierarchies.

A worthwhile contribution from Zhu et al. in 2013 [22] attempts to verify the data integrity of the files stored in the cloud servers by making use of fragment structure, hash table indexes, and probabilistic query-based auditing services for frequent verification processes. Though this work is a novel one of its kind, it lacks the proper authentication of the user during the data dynamic operations and incurs relatively significant computational overhead during the integrity verification process. A work on identity-based remote data integrity verification scheme proposed by Yang and Jia [23] in 2013 provides support for both static and dynamic data operations. In this case, the third-party auditor efficiently does the integrity verification of the data stored in the cloud, and provision has been made for doing batch integrity verification operations for multiple data owners and multiple cloud servers at the same time. But, this work does not authenticate the third-party auditor who verifies the integrity of the data.

Huang et al. in 2014 [24] allowed a third-party verification by utilizing the service of semitrusted TPAs. In this scheme, the TPA is assumed to be partially trusted, and a data owner verifies the proof handed to the TPA by the cloud server. Another work proposed by Wang et al. [25] in 2014 was based on the identity-based verification scheme which avoided the complex public key infrastructure based on complex certification process. Similarly, Yu et al. in 2015 [26] proposed a protocol for the public verification using algebraic signatures of data which prevented the replay and deletion attacks.

Another scheme by Liu et al. [27] in 2017 introduced a lattice-based scheme which is free from certificate verification processes and escapes the quantum computer attacks while ensuring data privacy against the third-party auditor. They have successfully verified the integrity of the stored data in the cloud without making use of the costly certification process. The scheme is resistant to the attacks posed by the cloud server. Though this work is a commendable one, it lacks user authentication during the verification procedure.

A recent work by Ren et al. [28] in 2018 makes use of rb23Tree to prevent the cloud servers from manipulating some of the sensitive data and escaping from the integrity verification procedure. Luo et al. in 2018 [29] proposed an efficient scheme using BLS short signatures which preserve the user privacy incurring only less computational and communication complexities. A very useful recent research work which involves the verification of the integrity audit of cloud data was proposed by Yan et al. in 2019 [30]. This efficient scheme preserves user privacy along with data blindness at a much less computational cost. A well-acclaimed work by He et al. in 2015 [31] preserves conditional privacy and ensures authentication in wireless environments. Also, a notable work from Zhang et al. in 2019 [32] preserves the privacy without using bilinear pairings.

2.1. Gaps in the Literature Survey. Some of the gaps identified include lack of authentication, protocols being susceptible to attacks, and more computational complexity, among others. The lack of authentication may help attackers masquerade in the verification process.

Objectives of the proposed research work are as follows:

(1) To invent a novel algorithm for the remote data integrity verification process which is free from attacks

(2) To invent a computationally efficient algorithm for enabling remote data integrity verification over the data stored in the cloud servers

(3) To introduce a secure authentication scheme to authenticate the mobile users during the secret key generation

(4) To enable secure authentication for the challenge response procedure during the integrity verification process

(5) To support dynamic data operations such as modification, deletion, and insertion on blocks of the files stored in the remote cloud storage

(6) To ensure perfect data privacy from the third-party auditor (TPA) and thereby allowing him only to do the verification process without gaining any information of the file stored in the cloud server

3. Preliminaries for the Proposed Work

3.1. Properties of Bilinear Pairing. Let us assume that G_1 and G_2 represent two multiplicative cyclic groups whose order is q and g be the generator of G_1. Now, the bilinear map e: $G_1 \times G_1 \longrightarrow G_2$ represents the bilinear pairing if the map exhibits the following three properties:

(1) The bilinear property of $e(P^x, Q^y) = e(P, Q)^{xy}$ for all $P, Q \epsilon G_1$ and $x, y \epsilon Z_q^*$

(2) The nondegeneracy property of $e(g, g) \neq 1$ where g is a generator of G_1

(3) The bilinear pairing function $e(P, Q)$ is computable using an efficient algorithm

3.2. Notations. The notable notations used in this research work are presented in Table 1.

4. Architecture of the Proposed System

The architecture of the proposed system is shown in Figure 1, and it can be better understood through the following steps. The system manager initializes the system. Then, the third-party auditor registers itself with the system manager. Then, a mobile user who wants to upload data to the cloud

Table 1: Important notations and their meanings.

Sl. no	Notation	Meaning
1.	G_1, G_2	Multiplicative cyclic groups of order q
2.	g	Generator of the cyclic group G_1
3.	$e(.,.)$	The bilinear map
4.	P, G, Y	Points in the group G_1 in which P, Y are public, and G is kept secret by the system manager
5.	α	A random element from Z_q^* kept as a secret by the system manager
6.	ID_i	Identity of the mobile user i
7.	n_i	Public key of the mobile user i
8.	MU_i	Mobile user i
9.	x_i	A random element from Z_q^* kept as a secret by mobile user i
10.	Enc_{n_i}	Asymmetric encryption function with the key n_i
11.	m_i	i^{th} block of file F
12.	σ_i	Tag of the i^{th} block of file F

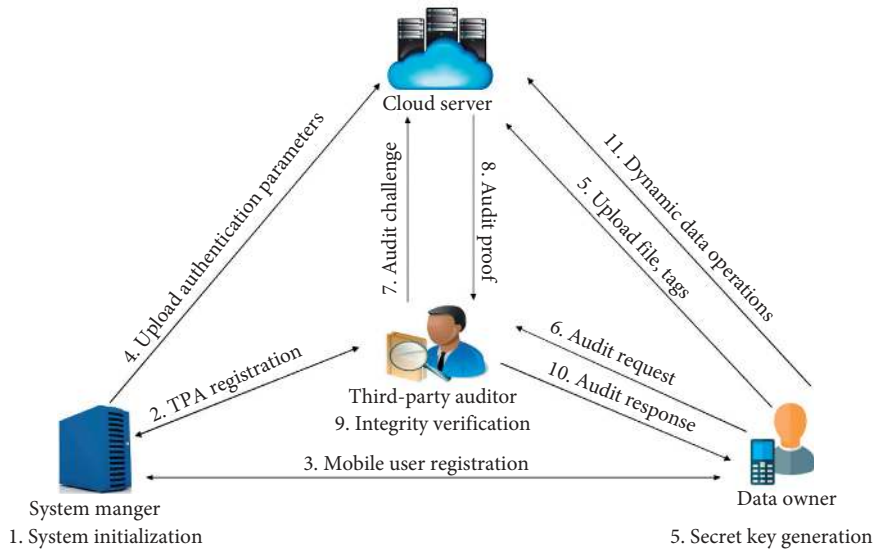

Figure 1: Architecture of the proposed research work.

server registers with the system manager which in turn uploads the unique parameters of the mobile users to the cloud which enables user authentication during audit response. Apart from this, the system manager sends a parameter composed of its master secret which enables the mobile user compute its own secret key. Now, a mobile user who wants to do audit of its data, sends an audit request to the third-party auditor. Accordingly, the third-party auditor creates an audit challenge and sends it to the cloud server. If the user authentication is successful, the cloud server generates an audit response which is verified by the third-party auditor, and the verification status is sent to the mobile user who initiated the audit request. Besides, if a data owner wishes to update, modify, or remove any block of data uploaded to the cloud, it can be achieved as well through dynamic data operations.

The proposed system consists of four major entities such as data owner who is the mobile user, the cloud server (CS), the system manager (SM), and the third-party auditor (TPA).

4.1. Data Owner. They are the mobile users who want to upload the sensitive files to the cloud storage due to lack of local storage and maintenance capabilities. At regular intervals, they will ensure the integrity of the remote data by sending auditing requests to the CS through the TPA. Moreover, the mobile user can modify, delete, or insert blocks of a file in the CS which was previously uploaded by it.

4.2. System Manager. It is the entity which initializes the system and is responsible for generating the secret key to the mobile user and the TPA for verification purposes. This entity also uploads authentication parameters of mobile users to the cloud server to enable secure authentication of the mobile users by the cloud servers.

4.3. Cloud Server. It represents the computer farms with vast potential for data storage sold in pay-per-use cost models. It is where the huge files which are divided into individual blocks of the mobile users are stored with provisions for integrity verification. The files along with their corresponding tags are stored to enable the auditing of the outsourced data of mobile users and authentication of mobile users. During the audit process by the TPA, the CS

receives a challenge from the TPA and accordingly sends a response back to the TPA to enable integrity verification.

4.4. Third-Party Auditor. It is the entity which does the auditing work on behalf of the mobile user. The TPA receives an auditing request from the mobile user and creates an auditing challenge based on some secret parameters and sends it to the CS for the authentication of the mobile user and the data integrity verification. The CS creates the corresponding audit response and sends it to the TPA. Now, the TPA verifies whether the received response is a genuine one or not. If successfully verified, this entity sends the auditing response to the mobile user. At regular intervals, they will assure the integrity of the remote data through auditing requests.

5. The Proposed Scheme

5.1. Initialization of the System. The SM initializes the system by selecting two multiplicative cyclic groups whose order is q, and the bilinear map is defined by

$$e\colon G_1 \times G_1 \longrightarrow G_2. \tag{1}$$

It selects a hash function H to produce the message digest and randomly selects an integer $\alpha \in Z_q^*$ and the points $P, G \in G_1$. It computes

$$Y = G^{\alpha}. \tag{2}$$

Now, the SM publishes the parameters of the system such as G_1, G_2, q, e, P, Y, and H. The parameters α and G are kept as a secret by the SM.

5.2. Mobile User Registration in the System. This phase consists of the following steps between the mobile user MU_i and the system manager SM. The MU_i sends ID_i, n_i to SM. Here, ID_i refers to the identity of MU_i, and n_i refers to the public key to MU_i. The SM in turn computes

$$X = e\left(P^{\alpha}, G^{n_i \cdot H(\mathrm{ID}_i)}\right), \tag{3}$$

$$A = n_i \cdot X. \tag{4}$$

It sends the computed values X and A to the mobile user MU_i. Now, the MU_i computes

$$D = n_i \cdot e(P, Y)^{n_i \cdot H(\mathrm{ID}_i)} \tag{5}$$

and verifies whether $D = A$ as follows:

$$\begin{aligned} D &= n_i \cdot e(P, Y)^{n_i \cdot H(\mathrm{ID}_i)} \\ &= n_i \cdot e\left(P, G^{\alpha}\right)^{n_i \cdot H(\mathrm{ID}_i)} \\ &= n_i \cdot e(P, G)^{\alpha \cdot n_i \cdot H(\mathrm{ID}_i)} \\ &= n_i \cdot e\left(P^{\alpha}, G^{n_i \cdot H(\mathrm{ID}_i)}\right) \\ &= n_i \cdot X \\ &= A. \end{aligned} \tag{6}$$

If successfully verified, MU_i ascertains that it has finished its registration with SM while sharing its public key with SM. Also, SM stores n_i, A of the user in its local storage. By now, MU_i and the SM have identified each other. This phase avoids any attacks posed by the attackers during the key generation, file upload, and integrity verification processes. Similarly, the TPA registers itself with the SM by sending its identity $\mathrm{ID}_{\mathrm{tpa}}$ and its public key n_{tpa}. Thus, the TPA and the SM identify each other as well.

5.3. System Manager Generating the Secret Key for the Mobile User. In this phase, the mobile user MU_i, after successful registration, makes a conversation with the SM in order to generate an exclusive secret key pertaining to this mobile user.

(1) The MU_i selects x_i at random from Z_q^* and computes P^{x_i}. It sends $n_i, n_i \cdot A, \mathrm{ID}_i, P^{x_i}$ to SM as depicted in Figure 2.

(2) The SM on receiving the parameters, retrieves the value of A based on the identity ID_i from its local storage. It computes $n_i \cdot A$ and checks whether it is the same as the received value. If not verified, $Z = e(P, G)$, and then the operation aborts. If verified, it selects β at random from Z_q^* and computes

$$K_1 = e\left(P^{x_i}, G^{(\beta/(\alpha+\beta))}\right), \tag{7}$$

$$K_2 = P^{(1/\alpha+\beta)}, \tag{8}$$

$$Z = e(P, G). \tag{9}$$

Now, the SM sends $\mathrm{Enc}_{n_i}(\mathrm{ID}_i, K_1, K_2, Z)$ to the mobile user MU_i. Besides, the SM updates the authentication table as cited in Table 2 with the identity ID_i and the corresponding parameter P^{x_i} of the MU_i.

The system manager SM, at the end of each day, uploads this authentication table along with the signature for the authentication details to the cloud server CS. The system manager computes the signature as $e(P^{\alpha \cdot h_1}, G^{H(\mathrm{ID}_{cs})})$, where h_1 refers to the hash of the details in the authentication table and ID_{cs} refers to the identity of the cloud server. The cloud server verifies the received data by checking whether

$$e\left(P^{\alpha \cdot h_1}, G^{H(\mathrm{ID}_{cs})}\right) \stackrel{?}{=} e\left(P^{h_2}, Y^{H(\mathrm{ID}_{cs})}\right). \tag{10}$$

In equation (10), h_2 refers to the hash value pertaining to the authentication table as received by the cloud server given in Table 2. Hence, the proof for this equation is

$$\begin{aligned} e\left(P^{h_2}, Y^{H(\mathrm{ID}_{cs})}\right) &= e\left(P^{h_2}, G^{\alpha \cdot H(\mathrm{ID}_{cs})}\right) \\ &= e\left(P^{h_2}, G^{H(\mathrm{ID}_{cs})}\right)^{\alpha} \\ &= e\left(P^{\alpha \cdot h_2}, G^{H(\mathrm{ID}_{cs})}\right). \end{aligned} \tag{11}$$

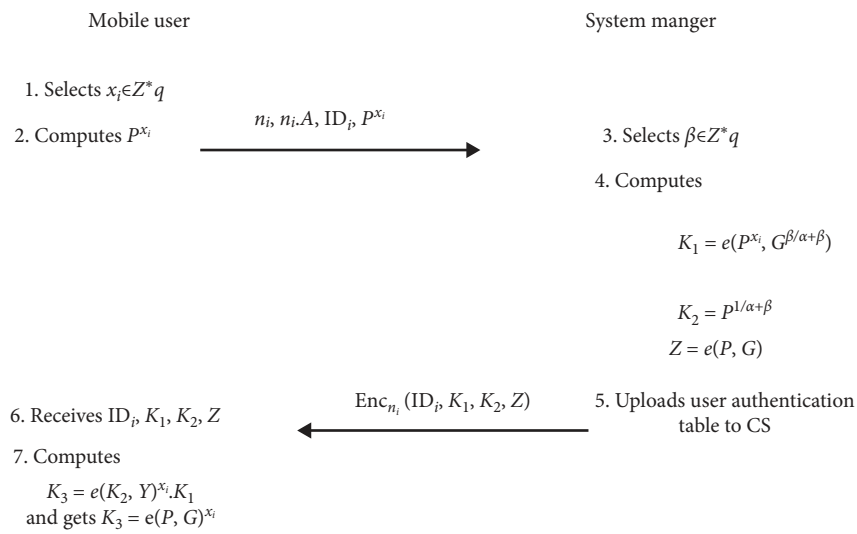

Figure 2: System manager generating the secret key for the mobile user.

Table 2: Mobile user authentication table.

Identity of the mobile user	Parameter
ID_1	P^{x_1}
ID_2	P^{x_2}
.	.
.	.
.	.
ID_i	P^{x_i}

Therefore, if the received hash value h_1 and the computed hash value h_2 are the same, the equation $e(P^{\alpha \cdot h_2}, G^{H(\mathrm{ID}_{cs})}) = e(P^{\alpha \cdot h_1}, G^{H(\mathrm{ID}_{cs})})$ becomes valid which ascertains that the received authentication table is completely intact.

(3) MU_i on receiving the encrypted message $\mathrm{Enc}_{n_i}(\mathrm{ID}_i, K_1, K_2, Z)$, decrypts it using its corresponding private key (using a suitable algorithm like RSA) and gets the parameters such as $\mathrm{ID}_i, K_1, K_2, Z$. Since this message is confidential, it avoids any man-in-the-middle attack or other such attacks during its transit from the SM to MU_i.

(4) Now, the mobile user MU_i computes the secret key K_3 as follows:

$$\begin{aligned} K_3 &= e\left(K_2, Y\right)^{x_i} \cdot K_1 \\ &= e\left(P^{(1/(\alpha+\beta))}, G^{\alpha}\right)^{x_i} \cdot e\left(P^{x_i}, G^{(\beta/(\alpha+\beta))}\right) \\ &= e\left(P^{(1/(\alpha+\beta))}, G^{\alpha}\right)^{x_i} \cdot e(P, G)^{(x_i \cdot \beta/(\alpha+\beta))} \\ &= e(P, G)^{(1/(\alpha+\beta)) \cdot \alpha \cdot x_i} \cdot e(P, G)^{(x_i \cdot \beta/(\alpha+\beta))} \\ &= e(P, G)^{(\alpha \cdot x_i/(\alpha+\beta))} \cdot e(P, G)^{(x_i \cdot \beta/(\alpha+\beta))} \\ &= e(P, G)^{((\alpha \cdot x_i)/(\alpha+\beta)) + ((x_i \cdot \beta)/(\alpha+\beta))} \\ &= e(P, G)^{x_i((\alpha+\beta)/(\alpha+\beta))} \\ &= e(P, G)^{x_i}. \end{aligned} \tag{12}$$

(5) Besides, the mobile user MU_i authenticates the system manager SM by verifying whether

$$K_3 = Z^{x_i}. \tag{13}$$

Since the value of Z can be known only to the SM, this verification procedure successfully authenticates the SM to the mobile user MU_i.

5.4. Tag Generation and File Upload by the Mobile User. Let us assume that the mobile user MU_i wants to upload a sensitive file F to the public cloud server CS. To store the file without any integrity breach in the middle and in order to be able to ascertain the genuineness of the file later, the mobile user MU_i performs the following steps:

(1) MU_i divides the file F into n blocks. Let us assume that $m_1, m_2, m_3, \ldots, m_n$ refer to the individual blocks of that file.

(2) It randomly selects $\gamma \epsilon Z_q^*$ to be used in the remote integrity verification process.

(3) It takes each block m_i and computes the corresponding block tag σ_i as

$$\begin{aligned} \sigma_i &= Z^{\gamma \cdot x_i} \cdot Z^{\gamma \cdot m_i} \\ &= e(P, G)^{\gamma \cdot x_i} \cdot e(P, G)^{\gamma \cdot m_i} \\ &= e(P, G)^{\gamma \cdot x_i + \gamma \cdot m_i} \\ &= e(P, G)^{(x_i + m_i) \cdot \gamma}. \end{aligned} \tag{14}$$

(4) The mobile user MU_i stores all the blocks of the file F along with the corresponding tags $\{\sigma_i\}_{i \epsilon n}$ of those blocks in the cloud server, where σ_i refers to the block tag of the corresponding block m_i.

(5) Finally, MU_i deletes its local copy of the file F from its local storage.

5.5. RDIC Challenge by the Third-Party Auditor. After a certain interval, the mobile user MU_i wants to verify the integrity of the file F which is stored in the public cloud. In order to verify the integrity, MU_i requests SM to send the public key to the TPA and gets it. Now, MU_i sends the parameters $ID_i, Z, x_i, n_i, \gamma$ to the TPA as $Enc_{n_{tpa}}(ID_i, Z, x_i, n_i, \gamma)$, where n_{tpa} is the public key of the TPA. Subsequently, TPA creates a challenge as follows based on the few randomly selected file blocks:

(1) The TPA selects a random integer $v_i \epsilon Z_q^*$ for each of the random selected block to be verified.

(2) In order to identify the mobile user MU_i to the cloud server CS and to avoid the man-in-the-middle attack and other possible attacks, the TPA computes the parameter E_1 as follows:

$$\begin{aligned} E_1 &= Z^{x_i \cdot n_i \cdot H(ID_i)} \cdot e(P,Y)^{x_i} \\ &= e(P,G)^{x_i \cdot n_i \cdot H(ID_i)} \cdot e(P,Y)^{x_i} \\ &= e(P,G)^{x_i \cdot n_i \cdot H(ID_i)} \cdot e\left(P,G^{\alpha}\right)^{x_i} \\ &= e(P,G)^{x_i \cdot n_i \cdot H(ID_i)} \cdot e(P,G)^{\alpha \cdot x_i} \\ &= e(P,G)^{x_i \cdot n_i \cdot H(ID_i) + \alpha \cdot x_i}. \end{aligned} \tag{15}$$

(3) Also, the TPA computes $E_2 = [e(P,G)^{x_i}]^{n_i}$ in which n_i is the public key of MU_i and x_i refers to the secret parameter of MU_i.

(4) Now, the TPA creates the challenge based on the corresponding block numbers and the randomly generated integers for those blocks as $chal = \{(i, v_i)\}_{i \epsilon I}$. For example, consider the case of $(5, v_5)$, where 5 refers to block number 5 and v_5 refers to the corresponding integer which was randomly chosen by the TPA.

(5) At last, the TPA sends E_1, E_2, ID_i, chal to the cloud server CS.

5.6. RDIC Response by the Cloud Server. Upon receiving the challenge from the TPA, the cloud server creates the response as follows:

(1) Firstly, the CS authenticates the MU_i by checking whether the equation

$$E_2^{H(ID_i)} \cdot e\left(P^{x_i}, Y\right) \stackrel{?}{=} E_1, \tag{16}$$

holds true or not. Here, P^{x_i} refers to the parameter corresponding to the mobile user MU_i which is present in the user authentication table. The proof for the equation can be understood as follows:

$$\begin{aligned} & E_2^{H(ID_i)} \cdot e\left(P^{x_i}, Y\right) \\ &= e(P,G)^{x_i \cdot n_i \cdot H(ID_i)} \cdot e\left(P^{x_i}, G^{\alpha}\right) \\ &= e(P,G)^{x_i \cdot n_i \cdot H(ID_i)} \cdot e(P,G)^{x_i \cdot \alpha} \\ &= e(P,G)^{x_i \cdot n_i \cdot H(ID_i) + x_i \cdot \alpha} \\ &= E_1. \end{aligned} \tag{17}$$

That is, the verification of

$$E_2^{H(ID_i)} \cdot e\left(P^{x_i}, Y\right) = E_1, \tag{18}$$

enables the cloud server to authenticate the mobile user. If the authentication is not successful, the CS aborts the integrity verification process.

(2) It computes the parameter

$$\mu = \sum_{i \epsilon I} m_i \cdot v_i. \tag{19}$$

(3) It also computes

$$\sigma = \prod_{i \epsilon I} \sigma_i^{v_i}. \tag{20}$$

(4) Sends μ, σ to the TPA who is waiting for the response.

5.7. Integrity Verification by the Third-Party Auditor. To verify whether the file uploaded long back was kept intact by the cloud server, the TPA does the integrity verification as

$$\sigma \stackrel{?}{=} \prod_{i \epsilon I} Z^{x_i \cdot v_i \cdot \gamma} \cdot Z^{\gamma \cdot \mu}. \tag{21}$$

This proof for the above equation ascertains the fact that the individual blocks of the file F which is stored in the remote server is kept intact by the cloud server. The overall interaction between the TPA and the CS during the auditing process is depicted in Figure 3.

5.8. Data Dynamic Operations. Under some circumstances, the information stored in a sensitive file may need to be modified or inserted or deleted. This research work strives to ensure the same with secure authentication procedure as follows.

For the file block modification operation, the mobile user wants to replace the block m_i of the file F with m_i^*.

(1) The mobile user MU_i finds that the i^{th} block m_i of the file F needs to be replaced by m_i^* in the cloud server. Hence, it computes the corresponding block tag for the block m_i^* as

$$\sigma_i^* = e(P,G)^{(x_i + m_i^*) \cdot \gamma}. \tag{22}$$

(2) Now, the mobile user MU_i computes $E_1 = e(P,G)^{x_i \cdot n_i \cdot H(ID_i) + \alpha \cdot x_i}$ and $E_2 = [e(P,G)^{x_i}]^{n_i}$ as in the challenge generation process.

(3) MU_i sends the parameters $(DO, ID_i, i, m_i^*, \sigma_i^*, E_1, E_2)$ to the cloud server in which DO refers to the file block modification operation, i refers to the i^{th} block which is to be updated, m_i^* refers to the new block which is going to replace the old block m_i and σ_i^* refers to the block tag of the new block m_i^*.

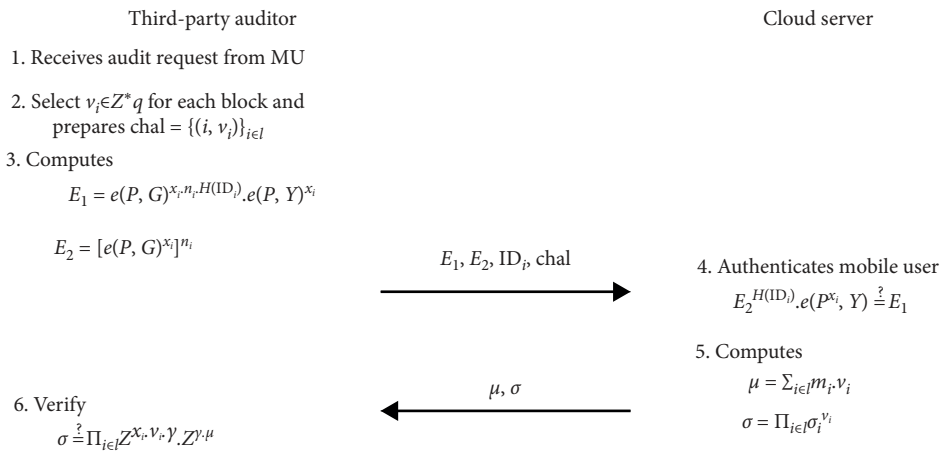

Figure 3: Auditing challenge, proof generation, and verification processes.

(4) The CS, upon receiving the parameters $(\mathrm{DO}, \mathrm{ID}_i, i, m_i^*, \sigma_i^*, E_1, E_2)$from MU_i, tries to authenticate MU_i as $E_2^{H(\mathrm{ID}_i)} \cdot e(P^{x_i}, Y) \stackrel{?}{=} E_1$.
That is,

$$\begin{aligned} & E_2^{H(\mathrm{ID}_i)} \cdot e\left(P^{x_i}, Y\right) \\ &= e(P, G)^{x_i \cdot n_i \cdot H(\mathrm{ID}_i)} \cdot e\left(P^{x_i}, G^{\alpha}\right) \\ &= e(P, G)^{x_i \cdot n_i \cdot H(ID_i)} \cdot e(P, G)^{x_i \cdot \alpha} \\ &= e(P, G)^{x_i \cdot n_i \cdot H(\mathrm{ID}_i) + x_i \cdot \alpha} \\ &= E_1. \end{aligned} \tag{23}$$

(5) If the authentication of MU_i is successful, CS replaces the old block m_i with m_i^*, and hence, the file F becomes $m_1 \ldots m_{i-1} m_i^* m_{i+1} \ldots m_n$. Also, it updates the corresponding tags of the file as $\sigma_1 \ldots \sigma_{i-1} \sigma_i^* \sigma_{i+1} \ldots \sigma_n$.

To delete a block m_i of the file F, the mobile user performs the following steps:

(1) As in block modification operation, MU_i sends the parameters $(\mathrm{DO}, \mathrm{ID}_i, i, m_i, \sigma_i^*, E_1, E_2)$ to the cloud server where DO refers to the block deletion operation

(2) The CS, upon receiving the parameters $(\mathrm{ID}_i, i, m_i, \sigma_i^*, E_1, E_2)$ from MU_i, tries to authenticate MU_i as $E_2^{H(\mathrm{ID}_i)} \cdot e(P^{x_i}, Y) \stackrel{?}{=} E_1$and verifies whether the received signature σ_i^* matches with σ_i of the block m_i which is to be deleted

(3) If both authentication of MU_i and tag verification are successful, CS deletes the block m_i from its storage space, and hence, the file F shall exist as $m_1 \ldots m_{i-1}$ $m_{i+1} \ldots m_n$, and the corresponding signatures are $\sigma_1 \ldots \sigma_{i-1} \sigma_{i+1} \ldots \sigma_n$

To insert a new file block m_{i+1} for the file F, MU_i performs the following steps:

(1) MU_i sends the parameters $(\mathrm{DO}, \mathrm{ID}_i, i, m_{i+1}, \sigma_{i+1}, E_1, E_2)$ to the CS in which DO refers to the block insertion operation

(2) The CS, upon receiving the parameters $(\mathrm{DO}, \mathrm{ID}_i, i, m_{i+1}, \sigma_{i+1}, E_1, E_2)$from MU_i, tries to authenticateMU_i as $E_2^{H(\mathrm{ID}_i)} \cdot e(P^{x_i}, Y) \stackrel{?}{=} E_1$

(3) If the authentication is successful, CS inserts the block m_{i+1} into its storage space, and hence, the file F shall exist as $m_1 \ldots m_{i-1} m_i m_{i+1} \ldots m_n$, and the corresponding signatures are $\sigma_1 \ldots \sigma_{i-1} \sigma_i \sigma_{i+1} \ldots \sigma_n$

6. Security Analysis of the Proposed Work

A remote integrity protocol is assumed to be secure if it exhibits the properties such as completeness, soundness, and data privacy. This section analyses the proposed protocol with regard to these essential properties.

Theorem 1 (completeness). *The integrity verification done by the TPA after receiving the audit response is based on a valid proof.*

Proof. In this research work, $\prod_{i \in I} Z^{x_i \cdot v_i \cdot \gamma} \cdot Z^{\gamma \cdot \mu} = \sigma$, as shown during the integrity verification process by the TPA, ensures the completeness of the proposed RDIC protocol. The proof for this equation can be given as follows:

$$\begin{aligned} & \prod_{i \in I} Z^{x_i \cdot v_i \cdot \gamma} \cdot Z^{\gamma \cdot \mu} \\ &= \prod_{i \in I} e(P, G)^{x_i \cdot v_i \cdot \gamma} \cdot e(P, G)^{\gamma \cdot \Sigma_{i \in I} m_i \cdot v_i} \\ &= \prod_{i \in I} e(P, G)^{x_i \cdot v_i \cdot \gamma} \cdot \prod_{i \in I} e(P, G)^{\gamma \cdot m_i \cdot v_i} \\ &= \prod_{i \in I} \left(e(P, G)^{x_i \cdot v_i \cdot \gamma} \cdot e(P, G)^{\gamma \cdot m_i \cdot v_i}\right) \\ &= \prod_{i \in I} \left(e(P, G)^{x_i \cdot v_i \cdot \gamma + \gamma \cdot m_i \cdot v_i}\right) \\ &= \prod_{i \in I} \left(e(P, G)^{(x_i + m_i) v_i \cdot \gamma}\right) \\ &= \prod_{i \in I} \left(\left(e(P, G)^{(x_i + m_i)\gamma}\right)^{v_i}\right) \\ &= \prod_{i \in I} \sigma_i^{v_i} \\ &= \sigma. \end{aligned} \tag{24}$$

If the mobile user and the cloud server are truthful and free from deceit, then the equation $\sigma \stackrel{?}{=} \prod_{i\epsilon I} Z^{x_i \cdot v_i \cdot \gamma} \cdot Z^{\gamma \cdot \mu}$ should hold true. □

Theorem 2 (user authentication). *The authentication verification of the cloud server during audit response is done based on a valid proof.*

Let us assume that a user i with identity ID_i has uploaded the parameter P^{x_i} to the cloud before uploading any file. Based on the request from that user, the TPA computes $E_1 = e(P,G)^{x_i \cdot n_i \cdot H(\mathrm{ID}_i) + \alpha \cdot x_i}$ and $E_2 = [e(P,G)^{x_i}]^{n_i}$ and sends $E_1, E_2, \mathrm{ID}_i, \mathrm{chal}$ to the cloud server. Here, the validity of $E_2^{H(\mathrm{ID}_i)} \cdot e(P^{x_i}, Y) = E_1$ done by the cloud server during the integrity verification ensures that only a valid user is part of the verification process and not an intruder. In this case, P^{x_i} is taken by the cloud server from the authentication table, and E_1, E_2 are received from the TPA. Assume that an attacker with a random value x_i' and masquerading as a legitimate user MU_i sends an audit request $E_1', E_2', \mathrm{ID}_i, \mathrm{chal}$ with $E_1' = e(P,G)^{x_i' \cdot n_i \cdot H(\mathrm{ID}_i) + \alpha \cdot x_i'}$ and $E_2' = [e(P,G)^{x_i'}]^{n_i}$. The CS, after receiving the audit request $E_1', E_2', \mathrm{ID}_i, \mathrm{chal}$ from the attacker, retrieves P^{x_i} from the authentication table and tries to verify the authentication as follows:

$$\begin{aligned} & E_2'^{H(\mathrm{ID}_i)} \cdot e\left(P^{x_i}, Y\right) \\ &= e(P,G)^{x_i' n_i \cdot H(\mathrm{ID}_i)} \cdot e\left(P^{x_i}, G^{\alpha}\right) \\ &= e(P,G)^{x_i' n_i \cdot H(\mathrm{ID}_i)} . e(P,G)^{x_i \cdot \alpha} \\ &= e(P,G)^{x_i' n_i \cdot H(\mathrm{ID}_i) + x_i \cdot \alpha} \\ &\neq E_1'. \end{aligned} \tag{25}$$

Thus, the authentication fails. Hence, the proposed system gives a valid proof only to a legitimate user during the authentication process.

Theorem 3 (soundness). *In our scheme, the attacker by possessing $e(P,G), v_i, \gamma$ cannot break the system and involve in audit cheat.*

The parameters $e(P,G)^{x_i}, v_i, \gamma$ are vital and are shared only between MU_i and the TPA. An attacker may be an existing member of the system and would like to masquerade as user MU_i by using its $Z = e(P,G), v_i, \gamma$ and tries construct the equation $\prod_{i\epsilon I} Z^{x_i \cdot v_i \cdot \gamma} \cdot Z^{\gamma \cdot \mu}$ but will only fail to do so since the value of x_i is known only to the legitimate user and the TPA. The equation in the proposed work $\prod_{i\epsilon I} Z^{x_i \cdot v_i \cdot \gamma} \cdot Z^{\gamma \cdot \mu} = \prod_{i\epsilon I} e(P,G)^{x_i \cdot v_i \cdot \gamma} \cdot e(P,G)^{\gamma \cdot \sum_{i\epsilon I} m_i \cdot v_i}$ shows that, during integrity verification, an attacker who wants to break the system should possess $e(P,G)^{x_i}, v_i, \gamma$ which are kept confidential and being shared only between MU_i and the TPA. Unless the TPA or the mobile user divulges this information, an attacker cannot guess the values of γ and $e(P,G)^{x_i}$ by knowing only the pairing operation $e(.)$ and P. Moreover, since the integrity verification is done based on all the blocks under consideration, neither the CS nor the TPA can involve in fraud as the values of $\mu = \sum_{i\epsilon I} m_i \cdot v_i$ and $\sigma = \prod_{i\epsilon I} \sigma_i^{v_i}$ cannot be easily computed if the data is corrupt in the cloud server or fiddled with by an attacker.

Table 3: Comparison of the security features.

Protocol	Integrity verification	Dynamic data operations	User authentication during integrity verification
Yu et al. [15]	Yes	No	No
Wang et al. [20]	Yes	No	No
Zhu et al. [22]	Yes	Yes	No
Yang et al. [23]	Yes	Yes	No
Proposed protocol	Yes	Yes	Yes

Theorem 4 (perfect data privacy). *In our scheme, the TPA is unable to learn any information regarding the data.*

A user sends $\mathrm{Enc}_{n_{\mathrm{tpa}}}(\mathrm{ID}_i, Z, x_i, n_i, \gamma)$ to the TPA during an audit request. During the whole process of challenge generation and integrity verification, the TPA never accesses the file blocks $m_1, m_2, m_3, \ldots, m_n$ or their signatures $\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_n$. During the challenge generation process, the TPA works with $\mathrm{ID}_i, n_i, E_1, E_2, \mathrm{chal}$ which do not divulge the details of any of the file block. Besides, during the verification process, the TPA verifies $\sigma \stackrel{?}{=} \prod_{i\epsilon I} Z^{x_i \cdot v_i \cdot \gamma} \cdot Z^{\gamma \cdot \mu}$ which does not reveal any information regarding the signatures or the file blocks. Though the value of μ is based on the file blocks and the value of σ is based on the signatures of those blocks, they are computed by the cloud server and not by the TPA. The TPA plays only the verification role. Hence, the TPA cannot learn any file information from these parameters leaving the system preserving the privacy of data during the file integrity challenge and the verification processes.

7. Results and Discussion

The proposed protocol is implemented in a machine with Windows operating system, Intel Core i5-4460 processor running at 3.20 GHz and 3.20 GHz with a primary memory of 4 GB. The experiments were conducted using pairing-based cryptography library pbc-0.5.14, and C programming language is used for the implementation purposes. Table 3 shows the security features provided by the proposed work and some notable similar works in the literature.

Let us assume that T_E refers to the cost of an exponentiation operation, T_H refers to the cost of a hash operation, T_P refers to the cost of a pairing operation, T_{PA} refers to the cost incurred during one point addition, T_{PM} refers to the cost of one point multiplication, and T_M and T_A refer to the cost of one integer multiplication and integer addition, respectively. During the tag generation phase, tag for each of the file block is generated, and n refers to the number of blocks in the file. The proposed research work is compared with the significant works proposed by Yu et al.

Table 4: Comparison of computational cost.

Sl. no.	Protocol	Tag generation (ms)	Proof generation (ms)	Proof verification (ms)
1.	Yu et al. [15]	$n*(1T_E+1T_H+1T_E+1T_{PA})$	$2T_H+2T_P+T_E+T_{PA}+c*(T_M+T_E)+(c-1)*(T_A+T_{PA})$	$c*(T_H+T_E+T_P)+1T_E+(c-1)*T_{PA}+1T_H$
2.	Wang et al. [20]	$c*(T_H+T_E+T_{PA})+T_E$	$1T_E+c*T_M+(c-1)*T_A+T_M+T_A+T_H+c*T_E+(c-1)*T_{PA}$	$1T_P+T_{PA}+T_E+T_H+c*(T_H+T_E+T_{PA})+(c-1)*T_{PA}$
3.	Zhu et al. [22]	$(n*T_E+n*T_A+1T_H)+(n*3T_M+(n-1)*T_A+1T_E)+(n*(T_H+T_E)+n*T_{PM})$	$c*T_M+c*T_E+(c-1)*T_{PA}+c*(1T_A+1T_M+c*T_M+(c-1)T_A)$	$T_P+T_{PA}+(c-1)T_A+c*(T_H+T_E)+T_P+c*T_E+(c-1)T_A+T_P+T_P$
4.	Yang et al. [23]	$n*(T_H+2T_{PM}+2T_E)$	$c*(1T_E+1T_M+1T_P+1T_E)+(c-1)T_{PA}+(c-1)T_A+(c-1)T_{PA}$	$c*(T_H+T_E)+(c-1)T_{PA}+T_{PM}+T_P+T_{PA}+T_P$
5.	Proposed protocol	$n*(T_P+T_A+1T_M+1T_E)$	$(1T_H+1T_E+1T_P+1T_{PA})+c*T_M+(c-1)*T_A+c*T_E+(c-1)*T_{PA}$	$T_H+c*(T_H+T_E)+(c-1)*T_{PA}+T_M+1T_E+T_{PA}$

[15], Wang et al. [20], Zhu et al. [22], and Yang et al. [23]. Though the previous works strive to ensure the integrity of the documents, they lack the user authentication factor during this process. This novel research work does both integrity verification and authentication processes. During the document integrity verification process, the major overhead incurred is due to the tag generation, proof generation, and proof verification processes. Hence, as part of this research work, Table 4 compares the computational cost of the proposed protocol with the recent protocols in the literature. The order of the group is setup as 160 bits, and the tests were conducted with a file size of 1 MB with approximately 50,000 data blocks. In Table 4, n refers to the number of blocks uploaded to the cloud server and c refers to the number of challenged blocks during the integrity verification process.

The computation involved in the system initialization is done only during the initialization of the system. For each user, the computation during registration is done only once during the user registration. A user may try to upload multiple files, and hence, tag generation is done only during file upload to the cloud server, and the computations for audit works will be done at regular intervals. From Table 4, it is evident that the proposed work incurs relatively less overhead than the other recent works in the literature.

For tag generation during the file upload, the mobile user makes one pairing operation and one exponentiation operation as $e(P,G)^{(x_i+m_i)\cdot\gamma}$ for each block. For n blocks, the user has to make n pairing and n exponentiation operations which are very less compared to the similar works in the literature. The results as shown in Figure 4 indicate the fact that the proposed work shows an improved efficiency with regard to the computational overhead of the previous protocols in the literature. The graph shows the comparison for the cost for a minimum of 100 blocks and a maximum of 1,000 blocks.

The mobile user incurs a computational cost of only one exponentiation and one pairing operation for the tag generation for each block. All the other recent works under comparison incur greater costs compared to the proposed work. For instance, for the tag generation of one block, the work proposed by Yu et al. involves two exponentiation operations per block, one hash operation per block, and one point addition operation which is more costly than the proposed work. From the graph, it can be inferred that, for the generation of tags for a total of 300 blocks, the proposed protocol incurs a computational overhead of 691 ms which is 599 ms less than Yu et al.'s scheme, 390 ms less than Wang et al.'s scheme, 724 ms less than Zhu et al.'s scheme, and 270 ms less than Yang et al.'s scheme.

In Figure 5, the performance of the proposed protocol for proof generation is compared with the works proposed by Yu et al., Wang et al., Zhu et al., and Yang et al., respectively. For instance, let us consider the cost for the proof generation for 700 blocks. In this case, the proposed protocol attains only a minor performance improvement like 10 ms and 41 ms than the works proposed by Yu et al. and Wang et al., respectively. But, it shows a major performance improvement such as 1,053 ms and 1,656 ms less than the works proposed by Zhu et al. and Yang et al., respectively. Thus, from the figure, we can infer that, as the number of blocks increases, the performance of the proposed protocol increases as well.

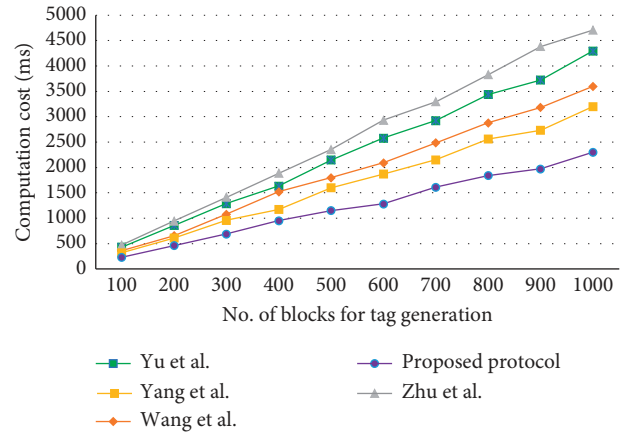

Figure 4: Computation cost for tag generation.

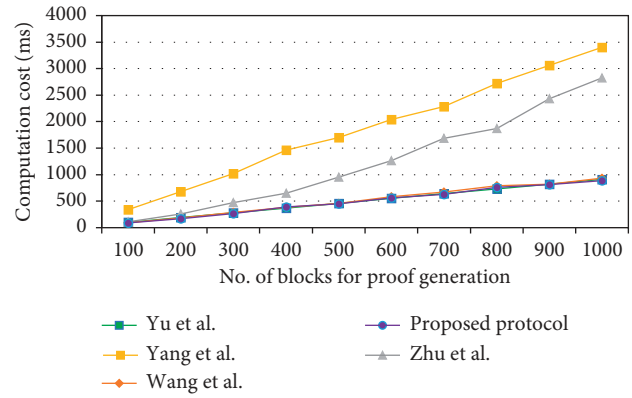

Figure 5: Computation overhead during proof generation.

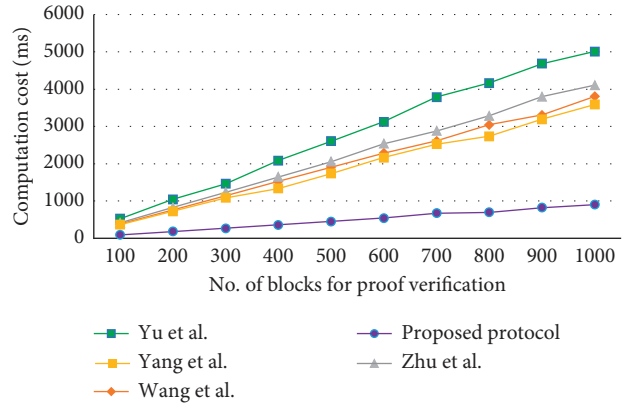

Figure 6: Computation overhead due to proof verification.

Similarly, Figure 6 shows the comparison of the computation cost for the proof verification process by the third-party auditor. The proof verification makes use of the blocks which are randomly selected by the third-party auditor

Table 5: Comparison of communication overhead.

Protocol	Challenge message (bits)	Response message (bits)
Yu et al. [15]	$c(\|p\| + \|p\| + \|n\| + \|q\| + \|p\|)$	$\|p\| + \|p\| + 2\|p\|$
Wang et al. [20]	$c(\|n\| + \|q\|)$	$\|q\| + 2\|p\|$
Zhu et al. [22]	$c(\|n\| + \|q\| + \|p\|)$	$4\|p\|$
Yang and Jia [23]	$c(\|n\| + \|q\|)$	$2\|p\|$
Proposed protocol	$c(\|n\| + \|q\|)$	$2\|p\|$

during audit challenge. During the computation of $\prod_{i\in I} Z^{x_i \cdot v_i \cdot \gamma}$, the TPA can compute the value of $x_i \cdot \gamma$ only once and use it for the subsequent operations as $(x_i \cdot \gamma) \cdot v_i$. The tests have been conducted by increasing the number of blocks from 100 to 1,000. The results clearly indicate that, for 600 blocks, the proposed protocol incurs 541 ms only which is 2,582 ms, 1,744 ms, 1,993 ms, and 1,622 ms less when compared to the works proposed by Yu et al., Wang et al., Zhu et al., and Yang et al., respectively. Thus, the proposed protocol shows improved performance than the previously proposed protocols in terms of the computational overhead.

Table 5 provides the comparison of the communication cost incurred by various protocols. In the table, $|p|$ refers to the size of an element in G, $|q|$ refers to the size of an element in Z_q^*, $|n|$ refers to the size of an element with regard to the block number, and c refers to the number of challenged blocks.

The communication cost is mainly due to the frequent audit challenges sent by the third-party auditor to the cloud server and the audit responses from the cloud server to the third-party auditor. The communication overhead occurs due to the messages exchanged as part of the registration process, challenge generation, and response processes. The communication overhead during the registration is not accounted in this work as it happens only once for a cloud user. For the phase of audit challenge sent by the TPA to the CS, TPA sends E_1, E_2, ID_i, chal to the cloud. Moreover, the size of chal = (i, v_i) is based on the number of challenged blocks, and hence, the size of the audit challenge is in $O(\text{chal})$. TPA sends E_1, E_2 to the CS only for the authentication purpose. Thus, the size of the audit challenge is $O(\text{chal})$ which is better than Yu et al. [15] and Zhu et al. [22] schemes and incurs similar overhead as that of Wang et al. [20] and Yang and Jia [23] schemes, respectively.

During the audit response phase of the proposed protocol, the cloud server sends μ, σ to the TPA. The size of both μ and σ is based on the number of challenged blocks. The proposed scheme is better than Yu et al.'s scheme [15] which incurs a communication overhead of $l + \log_2 r + 320$ bits and is more efficient than Wang et al.'s scheme [20] which involves μ, σ, R where R incurs an additional overhead of $\log_2 q$ bits compared to the proposed work. Moreover, the proposed work incurs half of the communication overhead as that of Zhu et al. [22] and is identical compared to Yang and Jia [23] scheme. Thus, the proposed protocol shows that it incurs a minimal communication overhead while providing support for efficient authentication feature and minimal computational complexity.

8. Conclusions

To sum up, a novel attack resistant and an efficient protocol for the verification of the remotely stored data in the cloud servers has been introduced in this research work. This work is the first of its kind in enabling authenticated verification procedure over the remote data stored in the cloud servers for the mobile users involved in the verification process. Enough security analysis has been provided to ascertain the genuineness of this work with regard to its resistance to the attacks in all aspects. The implementation results clearly show that this work provides auditing service with less computational complexity compared to the similar works in the recent literature. In this era of mobile computing, this work shall strive to prove its importance for sensitive files stored by the mobile users in the cloud servers. In future, this work can be further extended to support verification for the users of a corporate office or other communities who store data in multiple cloud servers.

Data Availability

No data were used to support this research work.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] Y. Zhang, Y. Li, and Y. Wang, "Secure and efficient searchable public key encryption for resource constrained environment based on pairings under prime order group," *Security and Communication Networks*, vol. 2019, Article ID 5280806, 14 pages, 2019.

[2] Y. Song, H. Wang, X. Wei, and L. Wu, "Efficient attribute-based encryption with privacy-preserving key generation and its application in industrial cloud," *Security and Communication Networks*, vol. 2019, Article ID 3249726, 9 pages, 2019.

[3] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," *Mobile Networks and Applications*, vol. 16, no. 3, pp. 270–284, 2011.

[4] E. Munivel and A. Kannammal, "New authentication scheme to secure against the phishing attack in the mobile cloud computing," *Security and Communication Networks*, vol. 2019, Article ID 5141395, 11 pages, 2019.

[5] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[6] B. Wang, B. Li, and H. Li, "Oruta: privacy-preserving public auditing for shared data in the cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 43–56, 2014.

[7] F. Sebe, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE*

Transactions on Knowledge and Data Engineering, vol. 20, no. 8, pp. 1034–1038, 2008.

[8] G. Ateniese, R. Burns, R. Curtmola et al., "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM conference on Computer and Communications Security*, pp. 598–609, New York, USA, October 2007.

[9] Y. Deswarte, J. J. Quisquater, and A. Saidane, "Remote integrity checking," in *Proceedings of the 5th Working Conference on Integrity and Intl control in Information Systems*, pp. 1–11, Lisbon, Portugal, October 2004.

[10] A. Juels and B. S. Kalisi Jr., "Pors: proofs of retrievability for large files," in *Proceedings of the 14th ACM conference on Computer and Communication Security (CCS07)*, pp. 584–597, Alexandria, VA, USA, October 2007.

[11] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proceedings of the 4th International Conference on Security and Privacy in Communication Networks*, September 2008.

[12] J. Sun and Y. Fang, "Cross-domain data sharing in distributed electronic health record systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 6, pp. 754–764, 2010.

[13] J. Sun, X. Zhu, C. Zhang, and Y. Fang, "HCPP: cryptography based secure HER system for patient privacy and emergency healthcare," in *Proceedings of the 31st IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 373–382, Washington, DC, USA, June 2011.

[14] L. Guo, C. Zhang, J. Sun, and Y. Fang, "PAAS: a privacy-preserving attributed-based authentication system for eHealth networks," in *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 224–233, Macau, China, June 2012.

[15] Y. Yu, M. H. Au, G. Ateniese et al., "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 767–778, 2017.

[16] X. Chen, T. Shang, F. Zhang, J. Liu, and Z. Guan, "Dynamic data auditing scheme for big data storage," *Frontiers of Computer Science*, vol. 14, no. 1, pp. 219–229, 2019.

[17] S. Peng, F. Zhou, J. Li, Q. Wang, and Z. Xu, "Efficient, dynamic and identity-based remote data integrity checking for multiple replicas," *Journal of Network and Computer Applications*, vol. 134, pp. 72–88, 2019.

[18] E. Fujisaki and T. Okamoto, "A practical and provably secure scheme for publicly verifiable secret sharing and its applications," *Advances in Cryptology EUROCRYPT in Lecture Notes in Computer Science*, Springer, vol. 1403, pp. 32–46, Berlin, Germany, 1998.

[19] A. Patra, A. Choudhury, and C. Pandu Rangan, "Efficient asynchronous verifiable secret sharing and multiparty computation," *Journal of Cryptology*, vol. 28, no. 1, pp. 49–109, 2015.

[20] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of the IEEE Infocom*, pp. 525–533, IEEE, San Diego, CA, USA, March 2010.

[21] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 12, pp. 2231–2244, 2012.

[22] Y. Zhu, G. J. Ahn, H. Hu, S. S. Yau, H. G. An, and S. Chen, "Dynamic audit services for outsourced storages in clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 227–238, 2013.

[23] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 9, pp. 1717–1726, 2013.

[24] K. Huang, J. Liu, M. Xian, and S. Fu, "Securing the cloud storage audit service: defending against frame and collude attacks of third party auditor," *IET Communications*, vol. 8, no. 12, pp. 2106–2113, 2014.

[25] H. Wang, J. Domingo-Ferrer, B. Qin, and Q. Wu, "Identity-based remote data possession checking in public clouds," *IET Information Security*, vol. 8, no. 2, pp. 114–121, 2014.

[26] Y. Yu, Y. Zhang, J. Ni, M. H. Au, L. Chen, and H. Liu, "Remote data possession checking with enhanced security for cloud storage," *Future Generation Computer Systems*, vol. 52, pp. 77–85, 2015.

[27] Z. Liu, Y. Liao, X. Yang, Y. He, and K. Zhao, "Identity-based remote data integrity checking of cloud storage from lattices," in *Proceedings of the 3rd International Conference on Big Data Computing and Communications (BIGCOM)*, pp. 128–135, Chengdu, Sichuan, China, August 2017.

[28] Z. Ren, L. Wang, Q. Wang, and M. Xu, "Dynamic proofs of retrievability for coded cloud storage systems," *IEEE Transactions on Services Computing*, vol. 11, no. 4, pp. 685–698, 2018.

[29] X. Luo, Z. Zhou, L. Zhong, J. Mao, and C. Chen, "An effective integrity verification scheme of cloud data based on BLS signature," *Security and Communication Networks*, vol. 2018, Article ID 2615249, 11 pages, 2018.

[30] X. Y. Yan, L. Wu, W. Y. Xu, H. Wang, and Z. Man, "Integrity audit of shared cloud data with identity tracking," *Security and Communication Networks*, vol. 2019, Article ID 1354346, 11 pages, 2019.

[31] D. He, S. Zeadally, B. Xu, and X. Huang, "An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2681–2691, 2015.

[32] J. Zhang, J. Cui, H. Zhong, Z. Chen, and L. Liu, "PA-CRT: Chinese remainder theorem based conditional privacy-preserving authentication scheme in vehicular ad-hoc networks," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2019.