applied sciences

MDPI

Article

Lightweight Accountable Privacy-Preserving Protocol Allowing the Cloud Client to Audit the Third-Party Auditor for Malicious Activities

Mohamed Ben Haj Frej [1,*], Julius Dichter [1] and Navarun Gupta [2]

1 Computer Science and Engineering Department, University of Bridgeport, Bridgeport, CT 06604, USA
2 Electrical Engineering Department, University of Bridgeport, Bridgeport, CT 06604, USA
* Correspondence: mbenhaj@bridgeport.edu

Received: 15 July 2019; Accepted: 20 July 2019; Published: 27 July 2019

Featured Application: The potential application of this work is in the field of cloud computing based on a third-party auditor. The idea is to provide the cloud client with tools that allow him to "audit the auditor".

Abstract: Cloud computing is reserving its position in the market as the next disruptive utility paradigm. It is found on the pay-as-you-use model. Cloud computing is changing the way information technology (IT) operates for individuals as well as for companies. Cloud computing comes with different offerings to accommodate diverse applications. It comes with many successful adoption stories and a few unfortunate ones that are related to security breaches. Security concerns are what is making many companies reluctant to fully embrace the cloud realm. To enhance trust and entice adoption between cloud clients (CC) and cloud service providers (CSP), a new paradigm of depending on involving a third-party auditor (TPA) has been introduced. Hence, implementing a solution with a TPA comes with its toll in terms of trust and processing overhead. A lightweight security protocol to give the CC extra control with tools to audit the TPA and the CSP is paramount to the solution. In this paper, we are introducing a novel protocol: the lightweight accountable privacy-preserving (LAPP) protocol. Our proposed protocol is lightweight in terms of processing and communication costs. It is based on a newly introduced mathematical model along with two algorithms. We have conducted simulation experiments to measure the impact of our method. We have compared LAPP to the most eminent privacy-preserving methods in the cloud research field, using the open source cloud computing simulator GreenCloud. Our simulation results showed superiority in performance for LAPP in regard to time complexity, accuracy, and computation time on auditing. The aim of the time complexity and computation time on auditing simulations is to measure the lightweight aspect of our proposed protocol as well as to improve the quality of service.

Keywords: cloud client (CC); cloud service provider (CSP) public auditing; lightweight accountable privacy-preserving (LAPP); third party auditor (TPA)

1. Introduction

Cloud computing comes with different deployment models (private, public, community, and hybrid). Depending on the cloud client's (CC) preferred infrastructure, cloud computing is offered as IaaS (infrastructure as a service), PaaS (platform as a service), and SaaS (software as a service) [1].

The cloud service provider's (CSP) responsibility for security is correlated with the level of infrastructure offered to the CC. In the case of IaaS and PaaS, the CSP offers the necessary tools, but it is up to the CC to adopt them correctly. A service-level agreement allows to manage the relationship between the CSP and the CC [1].

Cybersecurity is a cause for struggle among individuals and companies who want to keep their data safe. For computer networking, the confidentiality, integrity, and availability (CIA) triad and authentication, authorization, and accounting (AAA) security model, has been introduced to provide a concrete model on what should be observed in terms of security [2]. Cloud computing comes with its own security issues, such as colluding attacks, Byzantine attacks, malicious data, and so on [3,4].

To ensure conformity with the current standards and compliance with their security mainframe; nowadays, information technology (IT) auditing is a common practice for midsize and large corporations. In cloud computing, a TPA is introduced to assume the function of an "on-the-fly" audit. This audit encompasses the security of the communication between the cloud client and, as the cloud service provider, as well as the cloud client's data integrity [5,6].

As defined in [7], privacy preserving consists of making sure that the three cloud stakeholders are not involved in any malicious activities coming from insiders at the CSP level, as well as remediating to the TPA vulnerabilities and checking that the CC is not deceitfully affecting other clients.

Cloud storage is a flexible platform that allows the CCs to store their confidential data from their local servers to the cloud platform; nowadays, many CCs have embraced the cloud realm. Nonetheless, the cloud platform introduces new concerns and security trials. To overcome these concerns, a TPA is introduced to safeguard the confidential data and restore CC's confidence [7]. The TPA, as an individual or an organization, has the skills and proficiency necessary for exposing and evaluating the malicious threats encountered by the CSP. On the other hand, the TPA could be dishonest and become an adversary. As a result, the TPA's malicious role could cause significant damage by sharing the CC's confidential and private data with illegitimate parties and altering the CC's privacy, to obtain financial gains as well as other benefits [8]. Therefore, to preserve the CC's privacy, we need to be able to detect whether the TPA has a dishonest role while performing the audit of the CC's confidential information and private data. There are some existing protocols available in the literature to keep the check and balance on the possible malicious activities of the TPA [9–12]. However, computational and communicational overheads are not fully addressed in those existing protocols [13,14], leaving room for possible malicious activities from the TPA. Hence, there is a need for a lightweight protocol to preserve the CC's privacy and detect the TPA's malicious activities, on the occasion that there are any.

The next sections of the paper will be presented as follows: Section 2 evaluates the existing approaches that have tried to solve the same problem. Section 3 presents the proposed lightweight, accountable, privacy-preserving protocol. Section 4 explains the privacy-preserving polynomial model generation, where we demonstrate mathematically how our privacy-preserving polynomes are generated and verified. Section 5 presents the simulation setup and experimental results. In Section 6, we discuss these results; and finally, Section 7 concludes the paper and provides a hint of the planned future work.

2. Related Work

In this section, we will summarize the salient features of existing approaches that are related to our proposed method. Our proposed TPA-based lightweight accountable privacy-preserving scheme (LAPP) is based on homogeneity and includes features such as key update, substantiation, and validation. We are mainly focusing on TPA-based privacy-preserving solutions with homogenous characteristics. The chosen state-of-the-art methods are based on the following algorithms: KeyGen, SigGen, GenProof, and VerifyProof. Below is a recapitulation of the following schemes:

- Security and Privacy for Storage (SPS)
- Panda Public Auditing (PPA)
- Privacy-Preserving Public Auditing (PPPAS)
- Secure and Efficient Privacy-Preserving Public Auditing (SEPPPA) Protocol

2.1. *Security and Privacy for Storage (SPS)*

In this approach, the authors present what they labeled as a secure public auditing cloud storage enabling data dynamics [12]. The proposed approach focused on secure auditing while discouraging privacy-cheating. It bridges secure computation auditing and secure storage in the cloud. This model aimed to achieve a privacy-cheating discouragement using probabilistic sampling techniques, batch verification, and designated verifier signature.

The proposed SPS protocol has been introduced to assure the following objectives:

- Data computation security
- Data storage security
- Privacy-cheating discouragement
- Efficiency

For SPS's testbed, they use four computers with the following: 2.8 GHz i5-760 Intel processors and 4 GB of RAM. Their testing was based on subjecting the system to different traffic loads starting with 100 MB and gradually increasing to 850 MB. They also considered the impact of security overhead's traffic load traffic on both the master and the client sides. Their observations showed that their proposed protocol SecCloud has what they characterized as a small increase of 18% in the best-case scenario and 32% in the worst-case scenario. They have considered this overhead as not significant in regard to reducing the system's performance.

Moreover, based on our study of the protocol, recapitulated in the above section discussion, SPS has its limitations and increased the communication overhead, which has an impact on the QoS parameters.

2.2. *Panda Public Auditing (PPA)*

PPA [9] is introduced by the authors as assuring public auditing for secure data storage in the cloud through a third-party auditor. They propose an auditing protocol that uses cipher cryptography while communicating with the TPA. This protocol is focused on data storage and its integrity.

In this scheme, the third-party auditor executes the edits without needing copies of the CC's cloud data. The auditing process is based on four algorithms comparable to the ones used in [14].

- "KeyGen," allows generating keys for the TPA and the CC.
- "SigGen," will be run by the TPA for metadata verification.
- "GenProof," the CSP uses this protocol to check on how the data is stored.
- "VerifyProof" used to test the confirmation given by the CSP and verify it.

In their simulation experiments, the authors have measured the performance of the user revocation versus the number of re-signed blocks with or without verification. They are using a computer with 4GB of RAM, Intel processor i5 (2.5 GHz). They have presented their findings showing better results when using their method. They have also measured the auditing time (ms) versus the number of existing users (0 to 10, increasing by two users) and showed a linear increase. Their last simulation was on measuring the communication cost (KB) versus the number of existing users (0 to 10, increasing by two users) and showed a slight increase.

In this approach, the third-party auditor completes the auditing without having a copy of the CC's data. The concept of the intermediary server is used by the TPA to send the data. As a result, there is a possibility of hidden server failures that could affect the overall throughput performance.

2.3. *Privacy-Preserving Public Auditing (PPPAS)*

For the "Privacy-Preserving Public Auditing for Secure Cloud Storage" method, the users can have resort to a third-party auditor (TPA) to check the integrity of the outsourced data and increase confidence between the stakeholders. They are proposing a secure cloud system that supports public auditing while preserving privacy [15]. This approach further extends the results to empower the TPA to audit multiple users simultaneously. It utilizes a public key technique using HLA (homomorphic linear authenticator), which allows the TPA to do the auditing without having recourse to access the data.

The proposed scheme is based on the following algorithms: KeyGen, SigGen, GenProof, and VerifyProof. [16–18] In this scheme, the authors have compared their results to Compact Proofs of Retrievability and found that their scheme's response is close to the method they have compared to, even though their scheme supports privacy preserving while the Compact Proofs of Retrievability does not support it [10].

For their testbed, they are using a Linux-based computer with 2 GB of Memory, a 250 GB HDD, and an Intel Core 2 processor. They have compared, between batch auditing (c = 300, and c = 460) and individual auditing, the auditing time per task (ms) versus the number of auditing tasks (varying from 0 till 200) as well as the auditing time per task (ms) versus the fraction of invalid responses; from 0 till 18 (increasing by 2), using the same setup. They have compared their results to Compact Proofs of Retrievability and found that their scheme's response is close to the method they have compared to, even though their scheme supports privacy preserving while Compact Proofs of Retrievability does not. However, in this proposed approach, they have based their method on the homomorphic nonlinear authenticator algorithm, which is known for leading to additional overhead. The communication overhead is inherent in how the algorithm works, as it starts by generating a random real number. Then, based on the generated random number, binary random sequences are generated.

2.4. Secure and Efficient Privacy-Preserving Public Auditing (SEPPPA) Protocol

This privacy-preserving public auditing system is based on a mathematical model, and there were no simulations. It is based on algorithms; KeyGen, SigGen, ProofGen, and VerifyProof. The proposed scheme was designed with the purpose to use a blinding technique to improve security. For better efficiency, their proposed scheme stated to be able to provide the TPA with the tools to conduct auditing of multiple CCs simultaneously. They claim that the efficiency of their scheme is improved by minimizing expensive operations, in terms of processing, such as bilinear mapping, and by minimizing such computations from the user side [11]. This proposed approach was introduced to improve the efficiency by minimizing expensive operations such as bilinear mapping [19] but failed to improve the efficiency and increased the operations. On the other hand, this approach increased the computation overhead at the user's side.

3. Proposed Solution

This paper's contributions could be summarized as follows:

- Ensuring minimum overhead while successfully issuing the secret key to the three stakeholders (CSP, CC, and TPA).
- Determining and avoiding the malicious role of the TPA, if any, with a simple and lightweight algorithm.
- Enforcing the trust between the TPA and the CC by introducing a malicious-detection algorithm that enables both parties to keep check and balance each other.
- Assuring a more secure communication at a minimum communication cost.
- Accurately detecting malicious activities.
- Improving the quality-of-service (QoS) provision, our time complexity simulation results were of a paramount significance.

3.1. System Model

In this section, we describe our model's security requirements. The proposed protocol is constructed in a cloud computing environment. To describe the actual requirements, we start with discussing the system model that audits for the CC and makes the TPA accountable in front of the CC. As depicted in Figure 1, the system model supports three entities (TPA, CSP, and CC).

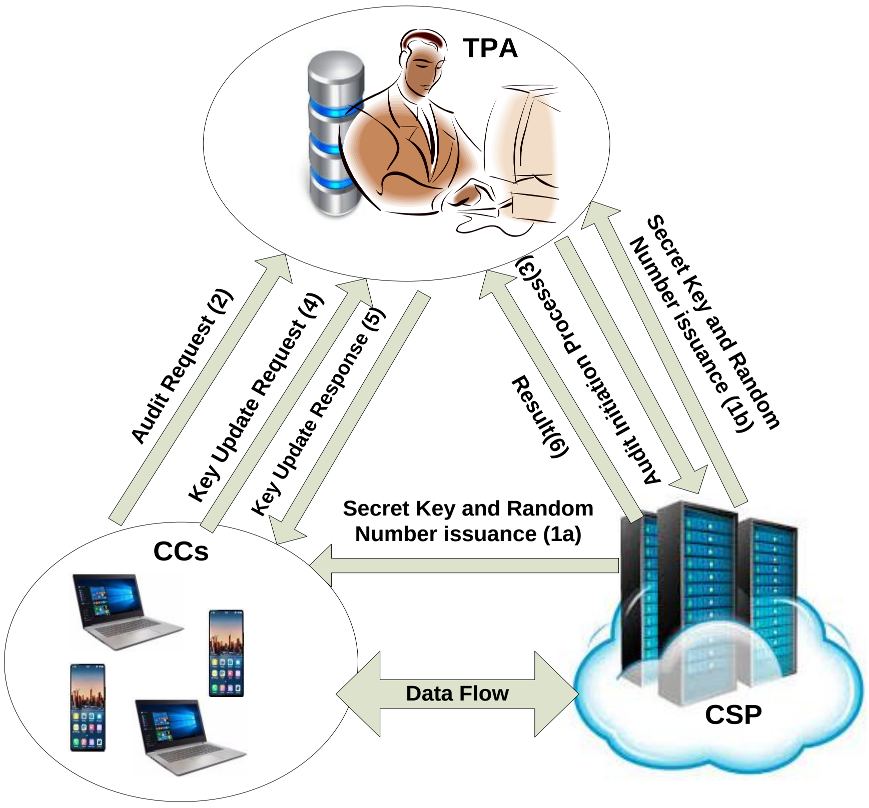
TPA
CCs
CSP
Audit Request (2)
Key Update Request (4)
Key Update Response (5)
Result(6)
Audit Initiation Process(3)
Secret Key and Random Number issuance (1b)
Secret Key and Random Number issuance (1a)
Data Flow

Figure 1. System model of the lightweight accountable privacy-preserving (LAPP) protocol.

- **Cloud Client:** It is the main entity in our proposed approach, which is composed of cloud users, devices, and customers. The privacy of the client is of a paramount significance along with the utilization of the limited storage and computing resources efficiently. The CC can request the TPA to audit its data while having a lack of trust in the TPA. Hence, the CSP issues a secret key and a random number to the CC and the TPA explained in the arrows 1a and 1b, respectively. Thus, the CC sends a key update request to the TPA for auditing process referred in (2). Furthermore, CC wants to confirm whether the TPA possesses a similar secret key and a random number, as CC sends a message to TPA presented as an audit request message as shown in (4). If the similar key and random number are found, then the TPA is considered as reliable; otherwise, the TPA is considered illegitimate and sharing the private information of the CC with someone else for personal gains.
- **Third Party Auditor:** It is considered as a highly trusted party. The TPA has a better capability of storage and computation. The performance of the TPAs is examined to safeguard their impartiality and fairness. Moreover, TPAs use their expertise to audit the data and provide the auditing result as the confirmation or the negation of the deceitful role of the CSP. However, the role of the TPA could be malicious, and the CCs will not have enough trust in him. Thus, the CC wants to ensure that the TPA is not playing a malicious role during the auditing process, as explained in this system model. Once the TPA receives the auditing request from the CC, the TPA establishes the audit initialization process with the CSP referred in message (3) to complete the auditing process. Furthermore, the TPA is also responsible for updating its key with the CC. In response, the TPA sends the key update response message as referred in (5).
- **Cloud Service Provider:** It involves several distributed servers that offer various services to CCs. The CSP has unlimited resources of computation and storage. The CC enjoys cloud services delivered via the Internet. According to our proposed approach, the CSP is assumed as trustworthy and could be considered deceitful. Thus, the CSP has been given the responsibility to issue the secret key and random number for both the TPA and the CC. Besides, this method of issuing the secret key and random number helps the CC to check the role of the TPA. The CSP is

responsible for forwarding the auditing results to the TPA, as referred in (6). The CSP interacts with the TPA and CCs and initially distributes its identity 'CSP'_{id} and the TPA and CCs record this information. In response, the CC and the TPA can use CSP_{id} to calculate the secret key as follows: For $i \in \{0, 1\}$, it calculates two hash values in which $T_i = L_1(CSP_{id}, i)$, $T_i \in O_1$. The output gives a secret key as: $S_i = vT_i$.

3.2. Proposed Lightweight Accountable Privacy-Preserving (LAPP) Protocol

There are several security threats to the cloud computing realm that cause damage to the privacy and confidentiality of data. Hence, complete satisfaction of CC is of a paramount significance. In this section, we present our proposed LAPP model, which provides the mechanisms for preserving the data privacy of the CC. This mechanism aims to ensure that the TPA should not expose or steal the CC's private data. Our proposed solution ensures that the TPA does not expose the CC's data contents while auditing the out-sourced data on the cloud servers. This concept is significant as it prevents issues resulting from involving the TPA in the process, whether they are intentional (TPA bribed by other sources) or unintentional (TPA compromised by attackers). Cloud storage is an easy and flexible platform that allows the CC to store their confidential data from their local computers to the servers on the cloud. Nowadays, many CC's store their confidential data in the cloud. However, this platform introduces new concerns and security trials. To overcome these concerns, the TPA is introduced to safeguard the confidential data and restore the confidence of the CC's, but it is also expected that the TPA could be dishonest. The TPA could share confidential information to illegitimate parties for the sake of gaining financial benefits. Therefore, after a comprehensive survey on cloud computing, based on a TPA homogeneity, dynamicity, and privacy-preserving, we concluded the need for a lightweight (in terms of processing and communication cost) solution, that allows the CC to have more control on the auditing process when a TPA is involved.

We assume that the CC transmits the data file 'D_f' with data contents 'D_c' given by:

$$D_f = \{D_{c1}, D_{c2}, D_{c3}, \ldots, D_{cn}\} \tag{1}$$

Each data content has physical significance and can be updated by the CC anytime. The CC can decrypt the private data contents.

The CC encrypts the data using its private key:

$$D_f = P_R < D_{c1}, D_{c2}, D_{c3}, \ldots, D_{cn} > \tag{2}$$

Therefore, private data contents are encrypted by

$$D_f = \sum_{i=0}^{T_{pr}} E_{key}\{P_R(D_{ci})\} \tag{3}$$

where, E_{key}: Encryption key; $P_R(Dci)$: Private data; T_{pr}: complete testing process.

In Equation (4), the CC uses its private key to encrypt the data contents. The data contents are further divided into subdata contents '$\mathrm{Sd_c}$' which are denoted as:

$$\mathrm{D_c} = \{\mathrm{Sd_{c1}}, \mathrm{Sd_{c2}}, \ldots, \mathrm{Sd_{cn}}\} \tag{4}$$

By adopting the same concept and for the sake of making the data more secure, the subdata contents $\mathrm{Sd_c}$ are further fragmented into smaller chunks $\mathrm{C(Sd_c)}$, explained as:

$$(\mathrm{Sd_c}) = \mathrm{C(Sd_{c1})}, \mathrm{C(Sd_{c2})}, \ldots, \mathrm{C(Sd_{cn})} \tag{5}$$

For simplicity, we only focus on single data content in our construct, and a constant number of small chunks for each subdata content. We chose the maximum number of chunks '$\mathrm{M[C(Sd_c)]}$'from the list of small chunks as '$\mathrm{C(Sd_c)n}$.'

Thus, $C(Sd_c)n < M[C(Sd_c)]$. Therefore, each subdata content Sd_c has a maximum of $M[C(Sd_c)]$ data chunks, by setting $(Sd_c)k = 0$ for $(Sd_c) < k < M[C(Sd_c)]$.

In our case, we are assuming that the size of each data chunk is persistent and equivalent to the security metric 'γ.' Therefore, the total of subdata contents '$T(Sd_c)$' can be calculated by Equation (6):

$$T(Sd_c) = S * D_c/\{(Sd_c) \log \gamma\} \tag{6}$$

where k: compatible data chunks and S: data content size.

Hence, the encrypted data contents are given by:

$$Dc = \{[(Sd_c)k\}\ k \in |1, T(Sd_c)\ |, k \in |1, C(Sd_c)\ | \tag{7}$$

The multiple encryption is used that leads to higher security. The significance of using multiple encryption is to address those problems which mostly doesn't exist visibly but can be resolved using multiple encryptions.

Let a bilinear map be used so that Z_1, Z_2, and Z_x are the multiplicative groups with identical prime order e and p. The bilinear map creates the relationship between cryptographic groups.

Thus, the bilinear map can be expressed as $Z_x \leftarrow Z_1 \times Z_2$ and their generators are z_1 and $z_2 \in Z_p$.

$$z1 \in Z1.$$

$$z2 \in Z2.$$

$$e, p \in Zp.$$

$$f: Z1 \times Z2 \rightarrow Zx$$

where $f\ (z_1{}^e, z_2{}^p) = f\ (z_1, z_2)^{\ ep}$

Let $f(h)$: $\{0,1)\ * \rightarrow Z_1$ be the secure hash function that is mapped with data contents D_c for the point Z_1.

We are using a hash function with a Boolean output. The purpose is to map the data with the purpose of further securing our system, as by definition hash functions takes in its input data with different sizes and will have an output data with a fixed size.

Our LAPP protocol involves the following procedures:

- Key Generation
- Key Update
- Label Generation
- Testing process
- Substantiation process
- Validation process

3.2.1. Key Generation

Since we have subdivided the data into multiple subchunks, to reassemble the data correctly, we are issuing labels for every subchunk created. The key generation process involves the security parameter 'β' that is based on two random numbers γ_1 and γ_2; for the secret hash key (for the data) and secret label key (for the labels), respectively. It can be expressed as $\gamma_{1,}\ \gamma_2 \in Z_p$. Thus, the secret label key output 'Sk_o' can be determined as:

$$Sk_o = z_2{}^{\gamma 2} \in Z_2 \tag{8}$$

Each generated key requires a signature that should be calculated while the end user gets the sent file. File X can be split into m chunks: $A \rightarrow \{c_1, c_2, \ldots, c_{n-1}, c_n\},\ n \in K^*$. The TPA can compute the pair of signatures $\{S_k, T_k\}$ for every file chunk c_k:

- Calculating two hash values as: $T_{\vartheta} = L_2(filename)$, $x_k = L_3(q_k,\ uuid,\ filename)$, q_k—is the pointer of the chunk c_k in the file X, $1 \leq k \leq n$.
- Starting an item of Shamir's *(r, d)* secret sharing $\mathcal{JNS}_{(r,\ d)}$ with $y(x) = T_{\vartheta} + q_1x + \ldots + q_{r-1}x^{r-1}$ and calculates $r - 1$ point $up = \ \{(e_1, y(e_1)),\ (e_2, y(e_2)),\ \ldots,\ (e_{r-1}, y(e_{r-1}))\big| e_l \in \{0,1\}^*\},\ \ up$ is a universal parameter. After that algorithm calculates $e' = L_4(uuid),\ o = y(e'), enc = \frac{e'||y(e')}{S_0||S_1}$.
- Computing n random variables as: $v_k \in K_t,\ 1 \leq k \leq n$, then calculates $R_k = v_kT$.

The key-generation process is performed by the CSP; And the authentication is initiated by the CC, as described in Algorithm 1.

Algorithm 1. Key-Generation and authentication process.

```
1.  **Initialization:** ($CS_p$: Cloud service provider; $S_k$: Secret key; $C_c$: Cloud Client; $Rn$: Random number; $T_{pa}$:
    Third-party auditor; $C_s$: Cloud Server; $K_{req}$: Key Request; $Vc$: Valid Client)
2.  **Input:** ($K_{req}$)
3.  **Output:** ($S_k$; $Rn$)
4.  Set $S_k$ & $Rn \in C_s$
5.  $C_c$ makes $K_{req}$
6.  Check $CS_p$ into $C_s$
7.  **If** $C_c = V_c$ then
8.  $CS_p$ releases $S_k$& $Rn$ to $C_c$ & $T_{pa}$
9.  ***Else if*** Denied $K_{req}$ to $C_c$
10. *Endif*
11. End-Elseif
```

Algorithm 1 describes the key-generation process when the CC requires an audit. In this algorithm, the CSP is considered as a reliable and trustworthy entity. In step 1, the initialization of variables is given. In steps 2 and 3, the input and the output processes are defined, respectively. In step 4, the secret key and random numbers are stored for each client on the cloud server that is assigned to the CC based on the given request. Step 5 shows the key request process done by the CC when intending to audit. Once the request is received by the CSP; it starts checking the validity of the CC into the cloud server shown in steps 6-7. If the CC is found as a valid client, the CSP releases the secret key and a random number to the CC and the TPA as described in step 8. If the CC is not registered with a given cloud server, then the request for obtaining the keys is denied.

3.2.2. Key Update

Each generated key requires updates. Without the key update, there is a chance of a compromise of the generated key.

Theorem 1. *The proposed agreement is evident for CCs.*

Proof. During the KeyGen process, CCs can examine the legitimacy of the key updates with the help of open keys and parameters from the CSP. In other words, after the CSP makes the clients' secret keys up to date, using open keys and parameters, the clients can check if the CSP's secured keys have been refreshed. Based on this, the proposed plan supports the key updates' obviousness for customers. □

Theorem 2. *The Proposed agreement is responsible for the period of key updates.*

Proof. During the key updates, the hash estimation of the encoded key is built by a certain counter estimation Ti. Hence, from the above presentation of key updates, the actual counter is a calculation of the prior encoded key and counter. Therefore, in the process of true key generation, customers can check the present counter against the past counter. If the present counter is equal or less than the prior counter, it can be inferred that the prior key has not been refreshed by the TPA. □

Hypothesis 1. *The proposed convention is correct if all parameters in the proposed convention are created, processed, and transmitted accurately.*

Proof. During the key updates, clients can check whether their secret keys have been refreshed by the CSP. As referenced in the KeyGen period, if the condition meets $\beta(\mathrm{F},\ \lambda) = \beta(rq,\ w)$, then the client can be sure that the secret key has been refreshed. Based on the bilinearity of the bilinear pairing, it is possible to elaborate the right side of the equation as $\beta(rq,\ w) = \beta\left(\mathrm{F}^t,\ w\right) = \beta\left(F,\ w^t\right) = \beta(F,\ \lambda)$. In other words, the outcomes are equivalent on the two sides of the equation. □

In the reviewing procedure of the proposed agreement, we can evaluate the rightness of the data storage in the cloud through registration, whether the produced evidence by the cloud is solid. From the definition of the ProofGen, the verification equation is built by the label confirmation, the data evidence, and the secret key sets. The left side of the mathematical checking statement can be expressed as:

$$\beta(r_a,\ rw_a) = \beta\left(\left(f^p \cdot \prod {v_j}^{\mu_j}\right)^{\frac{1}{d_j}},\ F^{d_j}\right) = \beta\left(f^p \cdot \prod {v_j}^{\mu_j},\ F\right) = \beta\left(f^p \cdot {\mu_j}^{\sum v_j},\ F\right) \tag{9}$$

Correspondingly, the right-hand side of the mathematical checking statement can be expounded as follows:

$$\begin{gathered}\beta(lw,\ rw) \cdot T_U = \beta(f,\ F^n) \cdot \prod \beta\left(U_j^{\mu_j},\ F^{\mu_j}\right) = \beta(f,\ F^n) \cdot \prod \beta\left(v_j,\ F^{\mu_j}\right) \\ \beta(lw,\ rw) \cdot T_U = \beta(f,\ F^n) \cdot \beta\left(v_j,\ F\right)^{\sum \mu_j} = \beta(f,\ F^n) \cdot \beta\left({v_j}^{\sum \mu_j},\ F\right) = \beta\left(f^m \cdot {v_j}^{\sum \mu_j},\ F\right)\end{gathered} \tag{10}$$

Subsequently, the accuracy of the evaluating stage can be verified. From the above verification process, it can be considered that key updates and the capacity inspecting are correct. Based on this, the proposed agreement is considered as correct.

3.2.3. Label Generation

Since we have issued labels for all the data subchunks, the label generation is implemented in a way that prevents injection of invalid data in the system and to mitigate any vulnerability of the system that could be introduced through the label generation. This process involves data content 'D_c' that encloses the secret hash key 'γ_1' and the secret label key 'γ_2'. 'γ_1' and 'γ_2' are used as inputs. The small chunks '$C(Sd_c)$' require random values, as illustrated in Equation (11):

$$C(Sd_c) = \{\tau_1,\ \tau_2,\ \tau_3,\ \ldots,\ \tau_{C(Sdc)}\} \in Z_p \tag{11}$$

Once the random values are chosen, the computing process 'C_k' is initiated as $C_k = {z_1}^{\tau k} \in Z_1$ for all k $\in$ [1, $C(Sd_c)$]. Thus, for each data contents $(Sd_c)k$ (k $\in$ [1, $C(Sd_c)$], the data label dl_j is determined by:

$$dl_j = \{f(h)(\gamma_1, d_{id} \| j)\} * \prod_{k=1}^{C(Sd_c)} C_k(Sd_c)k\}^{\gamma_2} \tag{12}$$

Labels are a vital part and data labels must be updated automatically. Therefore, the complete set of data labels can be obtained from the data contends by substituting the dl_j.

$$S(dtj) = [(dlj)j\ \in \{1,\ C(Sdc)\}]$$

$$S(dt_j) = \left\{ (\mathrm{f(h)}(\gamma_1,\ d_{id}\ \| j) * \prod_{k=1}^{C(Sd_c)} C_k{}^{(Sd_c)k})^{\gamma_2} \right\} j\ \in\ [1, C(Sd_c)] \tag{13}$$

where j is the label number for each data content, d_{id} is the data identifier, and $S(dt_j)$ is the complete set of data labels.

3.2.4. Testing Process

This section describes the first step in the overall validation of our method. The idea is to work on a sample of the data contents Dci, then generalize the method to the overall data. Hence, few samples of data contents are chosen to build the testing process '$\mathrm{T_{pr}}$' and generate random number 'Rn.' Thus, it can be written as Rn ϵ $\mathrm{Z_p}$* for selected data contents ($\mathrm{D_c}$) k {j ϵ $\mathrm{T_{pr}}$}. We calculate the testing process sample '$\mathrm{T_{ps}}$'as $\mathrm{Tp_s}$ = $\{\mathrm{T_{pr}}\}^{\mathrm{Rn}}$ by using a random number. Therefore, the complete testing process $\mathrm{T_{pr}}^*$ can be obtained as:

$$\mathrm{T_{pr}}^* = \{(\mathrm{j, Rn}), \mathrm{j} \epsilon\ \mathrm{T_{pr},\ Tp_s}\} \tag{14}$$

3.2.5. Substantiation Process

This phase is called "the substantiation process" as it gives more details of the testing with specifics to the data contents 'Ʈ' and the generated labels 'Ψ.' This process gets the data contents as inputs and then applies to them the complete testing process $T_{pr}{}^*$. Hence the process encompasses the label substantiation 'Ψ' and the data content substantiation 'Ʈ.'

Thus, the label substantiation can be generated by:

$$\Psi = \prod_{k \in T_{pr}} dl_j^{Rn} \tag{15}$$

The small data chunks of all tested data contents $\{T(D_c)\}$ are first calculated for generating the tested data content substantiation for each k ϵ [1, $C(Sd_c)$] given by:

$$T(D_c) = \sum_{T(D_c)} Rn * (Sd_c)k \tag{16}$$

The data content substantiation process can be obtained as

$$Ʈ = T(D_c) \times \Psi$$

By substituting the values of all tested data contents and label substantiation, the data content substantiation process is obtained by Equation (17)

$$Ʈ = \left\{ \sum_{T(D_c)} Rn * (Sd_c)k \right\} \times \prod_{k \in T_{pr}} dl_j^{Rn} \tag{17}$$

3.2.6. Validation Process

After applying our testing methodology to the labels and the data contents (substantiation phase), we implement a method to check whether or not the data contents have been tampered with by the TPA. We proceed by applying the complete testing process $\mathrm{T_{pr}}^*$, data content substantiation process Ʈ, secret hash key γ_1, the secret label key $\mathrm{Sk_o}$, using a sample of the data contents D_{ci}, as inputs. The identifier hash function f(h_i) is calculated first as

$$\mathrm{f}(h_i) = \{\mathrm{Tpr}^* + D_{ci} + Ʈ + \gamma 1(SK_o)\} \tag{18}$$

Based on given identifier hash function, we need to determine the tested hash function f(h_t) for all tested data contents. The calculation of the tested hash f(h_t) is given by:

$$\mathrm{f}(h_t) = T(D_c) + \mathrm{f}(h_i)$$

Substitution of tested data contents $T(D_c)$ & the identifier hash function $\mathrm{f}(h_i)$

$$\begin{aligned}\mathrm{f}(h_t) = \Bigg[\Bigg\{\Bigg\{\sum_{T(D_c)} Rn * (Sd_c)k\Bigg\} + \{(\mathrm{j, Rn}),\ \mathrm{j} \in \mathrm{Tpr, Tps}\} + \Bigg\{\sum_{T(D_c)} Rn * (Sd_c)k\Bigg\} \\ + \Bigg\{\sum_{T(D_c)} Rn * (Sd_c)k\Bigg\} \times \prod_{k \in T_{pr}} dl_j^{Rn} + \gamma 1(\mathrm{z}2\gamma 2\ \in\ \mathrm{Z}2)\Bigg]\end{aligned} \tag{19}$$

Since the hash function, defined earlier, f(h) has a Boolean output, the validation process can be determined by

{Data contents not tampered with by TPA if $\text{Ʈ}.\ \varepsilon\{\mathrm{f}(h_t), Sk_o\} = 1\}$

{Data contents tampered with by TPA if $\text{Ʈ}.\ \varepsilon\{\mathrm{f}(h_t), Sk_o\} = 0\}$

After determining the tampering process done by TPA, the possible validation process can be obtained by

$$\begin{aligned}\text{Ʈ}.\ \varepsilon\{\mathrm{f}(h_t), Sk_o\} &= \Bigg\{\sum_{T(D_c)} Rn * (Sd_c)k\Bigg\} \\ &\times \prod_{k \in T_{pr}} dl_j^{Rn} \Bigg\{\Bigg[\Bigg\{\sum_{T(D_c)} Rn * (Sd_c)k\Bigg\} + \{(\mathrm{j, Rn}),\ \mathrm{j} \in \mathrm{Tpr, Tps}\} \\ &+ \Bigg\{\sum_{T(D_c)} Rn * (Sd_c)k\Bigg\} + \Bigg\{\sum_{T(D_c)} Rn * (Sd_c)k\Bigg\} \times \prod_{k \in T_{pr}} dl_j^{Rn} \\ &+ \gamma 1(\mathrm{z}2\gamma 2\ \in\ \mathrm{Z}2)\Big],\ \mathrm{z}2\gamma 2\ \in\ \mathrm{Z}2\ \Big\}\end{aligned} \tag{20}$$

where ε is a small positive value.

In this section, the validation of the malicious role of the TPA is described in Algorithm 2.

Algorithm 2. Validation of the malicious role of TPA.

```
1.  Initialization: {$CS_p$: Cloud service provider; $S_k$: Secret key; $C_c$: Cloud Client; $Rn$: Random number; $A_{req}$: Audit Request; $T_{pa}$: Third party auditor; $C_s$: Cloud Server; $K_{ureq}$: Key Update Request; $K_{ures}$: Key Update Response; $TP_l$: Third party auditor legitimate; $TP_i$: Third party auditor illegitimate} Input: {$A_{req}$}
2.  Output: { $TP_l$; $TP_i$}
3.  $C_c \rightarrow A_{req}$ to $T_{pa}$
4.  Initiate $T_{pa} \rightarrow CS_p$
5.  Set $K_{ureq} \rightarrow T_{pa}$
6.  $T_{pa} \rightarrow K_{ures}$ to $C_c$
7.  $C_c$ Checks if $K_{ures} = S_k$ & $Rn$ then
8.  $T_{pa} = TP_l$
9.  Else if $T_{pa} = TP_i$
10. End if
11. End Elseif
```

Algorithm 2 determines the malicious activity of TPA. In step 1, the initialization process is described. Steps 2–3 show the input and output processes, respectively. In Steps 4–5, the CC initiates the request for the auditing process to the TPA. Once the TPA receives the request of auditing, it starts the auditing process. On the other hand, the CC does not have trust in the TPA, so it wants to send a key update request to ensure that the TPA has similar keys shown in Step 6. Once the TPA receives the key update request, if it is a legitimate TPA, then it sends the key update response to the CC as described in Step 7. When the CC receives the key update response from the TPA, it checks whether the response with a secret key and a random number matches, and the TPA is declared as legitimate; otherwise, the TPA is considered as illegitimate (refer to Steps 8–10).

4. Privacy-Preserving Polynomial Model Generation

In this section, we use the Chebyshev polynomials to demonstrate the effectiveness of our introduced method (LAPP) mathematically.

Consider a dataset $\{(x_i, y_i)\big| \leq i \leq n\}$, and let

$$\hat{f}(x) = a_1\varphi_1(x) + a_2\varphi_2(x) + \cdots + a_m\varphi_m(x) \tag{21}$$

where, $a_1, a_2, \ldots, a_m$ are coefficients and $\varphi_1(x), \varphi_2(x), \ldots, \varphi_m(x)$ are Chebyshev polynomials of the first kind,

$$\varphi_1(x) = T_0(x) = 1$$

$$\varphi_2(x) = T_1(x) = x$$

$$\varphi_n(x) = T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

Assume that the data $\{x_i\}$, are chosen from an interval $[\alpha, \beta]$. The Chebyshev polynomials can be modified as

$$\varphi_k(x) = T_{k-1}\left(\frac{2x - \alpha - \beta}{\beta - \alpha}\right) \tag{22}$$

The approximated function $\hat{f}$ of degree $(m-1)$ can be given by Equation (22), where the degree of (φ_k) is $k-1$. We will assume the interval $[\alpha, \beta] = [0, 1]$ and construct the model accordingly. According to Equation (22) when $[\alpha, \beta] = [0, 1]$, we get Equation (23)

$$\varphi_k(x) = T_{k-1}\left(\frac{2x - \alpha - \beta}{\beta - \alpha}\right) = T_{k-1}(2x - 1) \tag{23}$$

We can obtain the following equations for $m = 4$.

$$\varphi_1(x) = T_0(2x - 1) = 1$$

$$\varphi_2(x) = T_1(2x - 1) = 2x - 1$$

$$\varphi_3(x) = T_2(2x - 1) = 8x^2 - 8x + 1$$

$$\varphi_4(x) = T_3(2x - 1) = 32x^3 - 48x^2 + 18x - 1$$

To determine the $\hat{f}(x)$ when $m - 4$.

$$\hat{f}(x) = a_1\varphi_1(x) + a_2\varphi_2(x) + a_3\varphi_3(x) + a_4\varphi_4(x) \tag{24}$$

$$\hat{f}(x) = a_1(1) + a_2(2x - 1) + a_3\left(8x^2 - 8x + 1\right) + a_4\left(32x^3 - 48x^2 + 18x - 1\right)$$

Let the actual input be y_i, where $i = 1$ to n. The error of the approximated input can be determined by Equation (25)

$$e_i = \hat{f}(x_i) - y_i \tag{25}$$

We need to determine the values of a_1, a_2, a_3, and a_4, we use the root mean square error.

$$E = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\hat{f}(x_i) - y_i\right]^2}$$

Take the least squares fitting of $\hat{f}(x_i)$ of the class of functions C which minimizes E as $\hat{f}*(x)$

We can obtain $\hat{f}*(x)$ by minimizing E. Thus, we seek to minimize $M(a_1, a_2, a_3, a_4)$ which is given in Equation (26)

$$\begin{aligned} M(a_1,a_2,a_3,a_4) &= \sum_{i=1}^{n}\Big[a_1 + a_2(2x-1) + a_3\Big(8x^2 - 8x + 1\Big) \\ &+ a_4\Big(32x^3 - 48x^2 + 18x - 1\Big) - y_i\Big]^2 \end{aligned} \tag{26}$$

The values of a_1, a_2, a_3, and a_4 that minimize $M(a_1, a_2, a_3, a_4)$ will satisfy the expressions given from Equations (27)–(30).

$$\frac{\partial M(a_1,a_2,a_3,a_4)}{\partial a_1} = \frac{\partial\Big(\sum_{i=1}^{n}\big[a_1+a_2(2x-1)+a_3\big(8x^2-8x+1\big)+a_4\big(32x^3-48x^2+18x-1\big)-y_i\big]^2\Big)}{\partial a_1} = 0 \tag{27}$$

$$\frac{\partial M(a_1,a_2,a_3,a_4)}{\partial a_2} = \frac{\partial\Big(\sum_{i=1}^{n}\big[a_1+a_2(2x-1)+a_3\big(8x^2-8x+1\big)+a_4\big(32x^3-48x^2+18x-1\big)-y_i\big]^2\Big)}{\partial a_2} = 0 \tag{28}$$

$$\frac{\partial M(a_1,a_2,a_3,a_4)}{\partial a_3} = \frac{\partial\Big(\sum_{i=1}^{n}\big[a_1+a_2(2x-1)+a_3\big(8x^2-8x+1\big)+a_4\big(32x^3-48x^2+18x-1\big)-y_i\big]^2\Big)}{\partial a_3} = 0 \tag{29}$$

$$\frac{\partial M(a_1,a_2,a_3,a_4)}{\partial a_4} = \frac{\partial\Big(\sum_{i=1}^{n}\big[a_1+a_2(2x-1)+a_3\big(8x^2-8x+1\big)+a_4\big(32x^3-48x^2+18x-1\big)-y_i\big]^2\Big)}{\partial a_4} = 0 \tag{30}$$

A summary of the used notation is given in Table 1.

Table 1. Naming conventions.

Symbol/Function	Details
Df	data file
Dci	data contents
Sdci	subdata contents
C(Sdci)	smaller chunks of subdata contents
M[C(Sdc)]	the maximum of data chunks
T(Sdc)	total sub data contents
γ	security metric
k	compatible data chunks
S	data content size
f(h)	secure hash function
z1 and z2	key generators
β	security parameter consisting of two random numbers γ1 and γ2
γ1	secret hash key
γ2	secret label key
Sk0	secret label key output
τ1	random number
did	data identifier
dlj	data label
S(dtj)	complete set of data labels
Ck	computing process
A(Dc)	data content abstract information
Tpr*	complete testing process
Ψ	the label substantiation
Rn	random number
Ʈ	data content substantiation
ε	small positive value

5. Simulations Setup and Results

We use the same simulation parameters as in our previous publication [20]. To measure the performance of our newly introduced protocol, we have programmed the LAPP protocol in C++ to test it using the GreenCloud simulator. The open source simulator was loaded on an Ubuntu-based computer. The simulations were performed on a computer with 6 GB of RAM and an Intel Pentium Multi-Core CPU (4 GHz). For the testing system, we have used Windows 10 (64 bit). The data center scenarios were designed and simulated using the GreenCloud simulator. A highly scalable network was designed and consisted of 1600 × 1600 square meters. 2003-line cards, 2600 chassis switches, and 49 ports in the core layer (C1). The proposed LAPP protocol was installed on layer (C2) that consisted of 145-line cards, 234 chassis switches, and 53 ports. The data center covered 15,000 cloud clients, and each CC performs a maximum of 20 tasks/second. Each task consisted of 12,000 Bytes. The network with supported data center used 6 L3, 128 L2, and 106 hosts in each rack. Each host used eight processors with 32 GB memory, 240 GB virtual disk, and 180 GB storage. The switches' positions were set securely to avoid any possible malicious threat. The L3 is set at the center of the network with degrees of 60 at X-axis, Y-axis is at 250, and Z-axis at 0.

We have used different bandwidths at each layer; 32 for L1, 128 for L2, 264 for L3, and 512 for L4. The bandwidths elapsed and changed after half of the simulation time and became approximately half of the initial bandwidth. Real-time traffic was used so that the processing delay was ignored and only considered propagation delay that was 0.024 milliseconds, 0.005 milliseconds, and 0.0004 for L1, L2, and L3, respectively. The queue delay, burst time, and idle time were set as 0.005 seconds, 0.0056 seconds and 0.0032 seconds, respectively. The priority queue was used for buffering of the packets. We have used TCP Westwood for communications, and the packet size was 1260 bytes. The total experimental simulation time was set to 30 min. The output task size was not extended to more than 250,000 packets. The remaining parameters are summarized in Table 2.

Table 2. Simulation parameters.

Parameters	Details
Number of chassis switches at L4	1920
Line cards at L4	1630
Ports at L4	72
Number of racks at L4	16
Number of chassis switches at L3	432
Line cards at L3	164
Ports at L3	48
Number of racks at L3	128
Used virtual machines	1800
Number of servers	64
Maximum number of cloud service users	18,000
Hosts in each rack	132
Network size	1600 × 600 square meters
Each host supports	16 processors
Memory with each processor	256 GB
Storage memory	512 GB
Virtual disk memory	430 GB
Bandwidth for L4	256 GB/s
Bandwidth for L3	128 GB/s
Bandwidth for L2	64 GB/s
Bandwidth for L1	16 GB/s
Queue delay	0.005 s
Burst time	0.0056 s
Idle time	0.0032 s
Packet Size	1260

5.1. Simulation Parameters

Our simulation parameters were closer to a real cloud environment. In contrast with the methods we have compared our results to, the most developed testbed was the one used by SPS (it consisted of four computers with 2.8 GHz i5-760 Intel processors and 4 GB of RAM). These platforms are considered very limited compared to what the cloud can offer nowadays and to our testing platform. To give an example, we have used 1800 Virtual Machines, 64 servers, 132 hosts (every host: 16 cores, 256 GB). Thus, our testbed is far more resourceful than the one used by the other methods.

5.2. Simulation Results

We have conducted our simulations using the following rates of TPA malicious activities: 0%, 1%, 2%, and 5%. We have only included the graphs of malicious attempts at 2% to keep the paper short.

The following measurements have been conducted:

- Computation time of auditing (number of challenged data blocks).
- Accuracy (number of malicious attempts).
- Time complexity (input files).

5.2.1. Computation Time of Auditing versus the Number of Challenged Data Blocks

In this experiment, we measure the computation time of auditing in seconds versus the number of challenged blocks (please refer to Figure 2). We proceed from measurement to measurement by increasing the number of challenged blocks by 500. Our results show that at 4500 challenged blocks, the computation time on auditing for LAPP is about 2.25 s. For the other methods, it varies between 2.9 and 3.25 s.

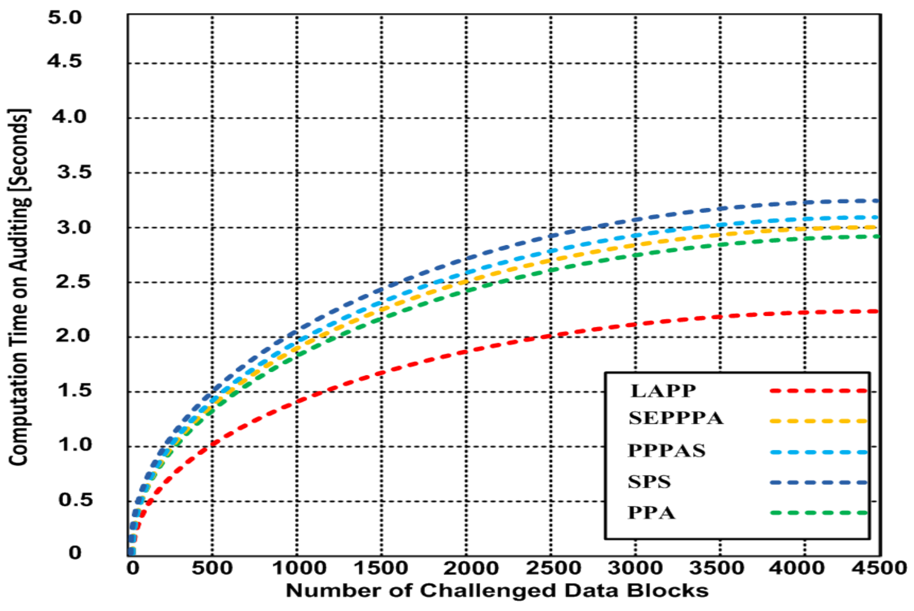

Figure 2. Computation time of auditing (number of challenged data blocks).

5.2.2. Accuracy versus the Number of Malicious Attempts

In this simulation, we measured the accuracy in percentage versus the number of malicious attempts, by increasing the number of malicious attempts by three for each measurement (please refer to Figure 3). Our results show that LAPP has an accuracy of 99.98% at 27 malicious attempts, while the other methods are between 99.46% and 99.58%.

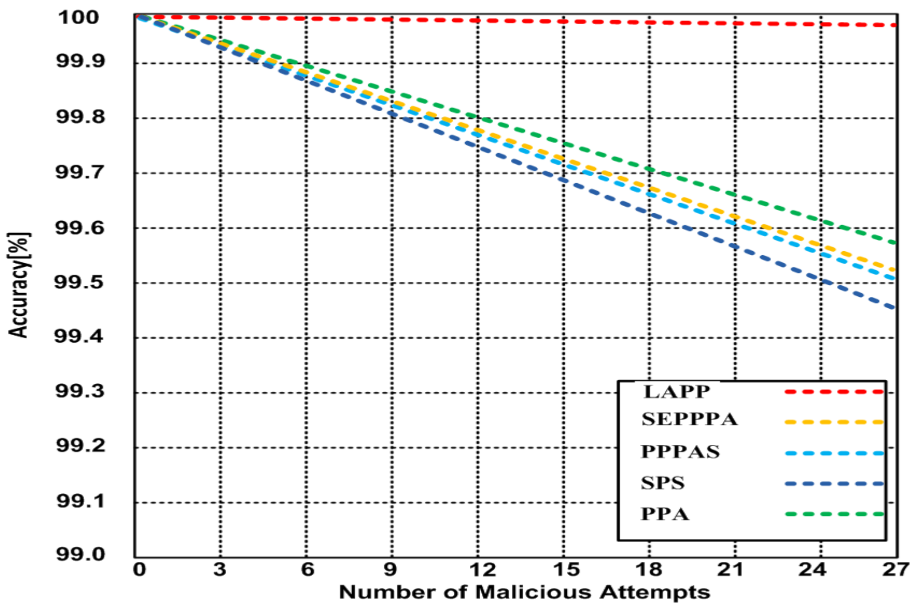

Figure 3. Accuracy (number of malicious attempts).

5.2.3. Time Complexity versus Input Files

In this experiment, we measure the time complexity in seconds versus the size of input files in KB (please refer to Figure 4). We proceed from measurement to measurement by increasing the size of the input files by 4 KB. The order of complexity is depicted in terms of the big-O notation. The simulation results show a linear increase for LAPP, a logarithmic increase for SEPPA and PPA, and quadratic time complexity for PPPAS and SPS.

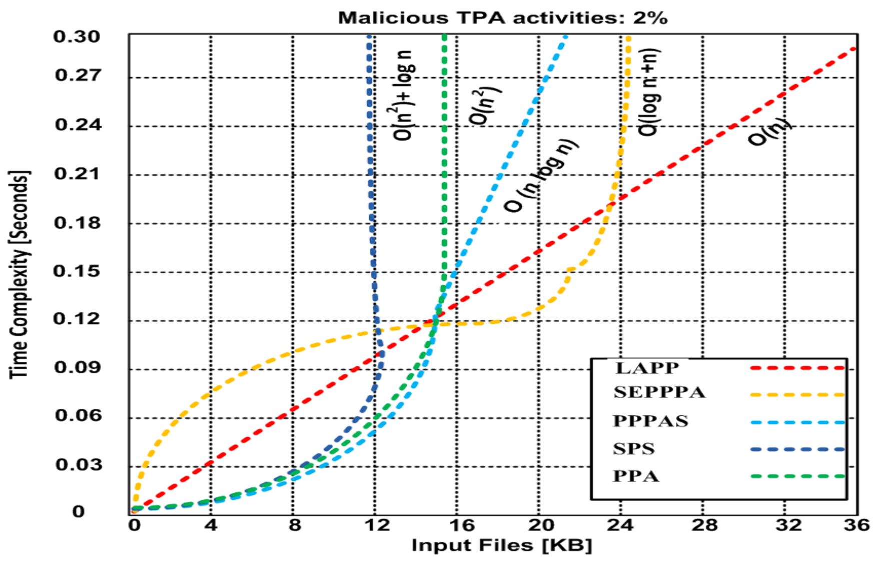

Figure 4. Time complexity (input files).

6. Discussions

Our newly introduced protocol is a lightweight privacy-preserving protocol allowing secure validations between the cloud's three main players, namely: the CC, CSP, and the TPA; with the aim to detect, on the fly, dishonest activities of the TPA as well as auditing the CSP's procedures. We start by presenting the methods that we compare LAPP to, namely SPS, PPA, PPPAS, and SEPPPA (Table 3).

Table 3. Comparison of LAPP and competing approaches.

Security Models	Security Requirements	Threats	Advantages
SPS	■ Third-party auditing ■ Supports data dynamics ■ Support privacy-preserving public auditing ■ Use of private channels to relay public information ■ KeyGen, SigGen, GenProof, and VerifyProof features	■ TPA somehow trusted ■ Communication overhead	■ Practical for cloud system on a large scale. ■ Considers the vulnerability of dynamic data
PPA	■ Third-party auditing ■ Supports data dynamics ■ Double block transportation ■ Supports privacy-preserving public auditing ■ KeyGen, SigGen, GenProof, and VerifyProof features	■ TPA is used as an intermediary to send encrypted data ■ Hidden server failure.	■ Practical for cloud system on a large scale. ■ TPA doesn't need a local copy of Data
PPPAS	■ Third party auditing. ■ Supports batch auditing. ■ Supports privacy-preserving public auditing. ■ KeyGen, SigGen, GenProof, and VerifyProof features	■ TPA is somehow trusted	■ TPA does not need a local copy of data. ■ Identification of invalid Response. ■ Supports for dynamic data
SEPPPA	■ Third party auditing. ■ Supports batch auditing. ■ Supports privacy preserving public auditing. ■ KeyGen, SigGen, GenProof, and VerifyProof features	■ TPA is somehow trusted	■ TPA does not need a local copy of data. ■ To date, a pioneer in privacy-preserving schemes for cloud
LAPP	■ Third-party auditing ■ Determining malicious TPA activity ■ Supports data dynamics ■ Support privacy-preserving ■ Testing process ■ Substantiation process ■ Validation process	■ TPA is somehow trusted	■ Practical for cloud system on a large scale. ■ Supports for dynamic data ■ Detecting malicious activity of the TPA. ■ Less communication overhead ■ Minimum time complexity ■ Better accuracy

In our first experiment, we measured the computation time of auditing in seconds versus the number of challenged blocks, increasing the number of blocks by 500 for each measurement. The results presented a better response for LAPP by 28% compared to PPA and 44.45% compared to SPS.

In our second simulation, we measured the accuracy in percentage compared to the number of malicious attempts, and our results showed a steady accuracy for LAPP, whereas it started degrading considerably for the other methods.

In our last simulation, we measured the time complexity in seconds versus the size of input files in KB by increasing the size of the input files by 4 KB each time. Our results showed a constant proportional increase in time to run LAPP after increasing the number of input files, while the other methods showed a logarithmic or quadratic increase.

Our simulation results demonstrate the efficiency of our newly introduced protocol (LAPP) while proving its lightweight nature when it comes to its communication and processing overheads.

7. Conclusions

In the information technology field, it is becoming more common to hire third-party auditors to verify if mid-size companies and corporations are adhering to their security frameworks. This same concept has been introduced to cloud computing, in the form of on-the-spot auditing, by implementing the TPA in the solution. Hence, a TPA comes with its issues, mainly trust and an extra cost regarding processing and communication.

In order to further secure cloud computing and enhance the trust between its main stakeholders (CSP, CC, and TPA), in this paper, we have proposed a novel protocol: the lightweight accountable privacy-preserving (LAPP) protocol. LAPP aims to provide tools to the CC to "audit the auditor" [12,13]. It has been implemented through three algorithms and a mathematical model.

To verify LAPP's performance, we have conducted simulations with the introduction of TPA's malicious attempts at the rates of 0%, 1%, 2%, and 5% as follows:

- Computation time of auditing (number of challenged data blocks).
- Accuracy (number of malicious attempts).
- Time complexity (input files).

Based on our simulation results, we have demonstrated LAPP's superiority to the existing privacy-preserving protocols. This has led us to conclude that our newly introduced scheme allows, in an optimal fashion, to "audit the auditor" to detect dishonest activity.

In our future work, we are working on simulating our protocol along with the other methods on different size networks (small and medium sizes) to see if it maintains the same performance.

Author Contributions: M.B.H.F. has elaborated this work under the supervision of J.D. and N.G.

Funding: This research received no external funding.

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

References

1. Singh, S.; Jeong, Y.S.; Park, J.H. A survey on cloud computing security: Issues, threats, and solutions. *J. Netw. Comput. Appl.* **2016**, *75*, 200–222. [CrossRef]
2. Lins, S.; Schneider, S.; Sunyaev, A. Trust is good, control is better: Creating secure clouds by continuous auditing. *IEEE Trans. Cloud Comput.* **2016**, *6*, 890–903. [CrossRef]
3. Xu, J. Auditing the Auditor: Secure Delegation of Auditing Operation over Cloud Storage. *IACR Cryptol. EPrint Arch.* **2011**, *2011*, 304.
4. Huang, K.; Xian, M.; Fu, S.J.; Liu, J. Securing the cloud storage audit service: Defending against frame and collude attacks of third-party auditor. *IET Commun.* **2014**, *8*, 2106–2113. [CrossRef]
5. Hsien, W.F.; Yang, C.C.; Hwang, M.S. A Survey of Public Auditing for Secure Data Storage in Cloud Computing. *Int. J. Netw. Secur.* **2016**, *18*, 133–142.

6. Hussien, Z.A.; Jin, H.; Abduljabbar, Z.A.; Yassin, A.A.; Hussain, M.A.; Abbdal, S.H.; Zou, D. Public auditing for secure data storage in cloud through a third-party auditor using modern ciphertext. In Proceedings of the 2015 11th International Conference on Information Assurance and Security (IAS), Marrakech, Morocco, 14–16 December 2015.
7. Razaque, A.; Rizvi, S.S. Privacy preserving model: A new scheme for auditing cloud stakeholders. *J. Cloud Comput.* **2017**, *6*, 7. [CrossRef]
8. Razaque, A.; Rizvi, S.S. Triangular data privacy-preserving model for authenticating all key stakeholders in a cloud environment. *Comput. Secur.* **2016**, *62*, 328–347. [CrossRef]
9. Han, S.; Xing, J.C. Ensuring data storage security through a novel third party auditor scheme in cloud computing. In Proceedings of the 2011 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), Beijing, China, 15–17 September 2011; pp. 264–268.
10. Razaque, A.; Vennapusa, N.R.; Soni, N.; Janapati, G.S.; Vangala, K.R. Task scheduling in cloud computing. In Proceedings of the 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, USA, 29–29 April 2016; pp. 1–5.
11. Ranchal, R.; Bhargava, B.; Othmane, L.B.; Lilien, L.; Kim, A.; Kang, M.; Linderman, M. Protection of identity information in cloud computing without trusted third party. In Proceedings of the 2010 29th IEEE Symposium on Reliable Distributed Systems, New Delhi, India, 31 October–3 November 2010; pp. 368–372.
12. Shimbre, N.; Deshpande, P. Enhancing Distributed Data Storage Security for Cloud Computing Using TPA and AES Algorithm. In Proceedings of the 2015 International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 26–27 February 2015.
13. Rizvi, S.; Razaque, A.; Cover, K. Third-Party Auditor (TPA): A Potential Solution for Securing a Cloud Environment. In Proceedings of the 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 3–5 November 2015.
14. Li, H.; Lin, X.D.; Yang, H.M.; Liang, X.H.; Lu, R.X.; Shen, X.M. EPPDR: An efficient privacy-preserving demand response scheme with adaptive key evolution in smart grid. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 2053–2064. [CrossRef]
15. Shrinivas, D. Privacy-preserving public auditing in cloud storage security. *Int. J. Comput. Sci. Inf. Technol.* **2011**, *2*, 2691–2693.
16. Wang, B.; Li, B.; Li, H. Panda: Public auditing for shared data with efficient user revocation in the cloud. *IEEE Trans. Serv. Comput.* **2015**, *8*, 92–106. [CrossRef]
17. Wang, C.; Wang, Q.; Ren, K.; Lou, W.J. Privacy-preserving public auditing for data storage security in cloud computing. In Proceedings of the INFOCOM, IEEE Computer and Communications Societies, IEEE Annual Joint Conference, San Diego, CA, USA, 14–19 March 2010.
18. Wang, C.; Chow, S.S.M.; Wang, Q.; Ren, K.; Lou, W.J. Privacy-preserving public auditing for secure cloud storage. *IEEE Trans. Comput.* **2013**, *62*, 362–375. [CrossRef]
19. Worku, S.G.; Xu, C.X.; Zhao, J.N.; He, X.H. Secure and efficient privacy-preserving public auditing scheme for cloud storage. *Comput. Electr. Eng.* **2014**, *40*, 1703–1713. [CrossRef]
20. Frej, M.B.H.; Dichter, J.; Gupta, N. Light-weight accountable privacy preserving (LAPP) protocol to determine dishonest role of third-party auditor in cloud auditing. In Proceedings of the 2018 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 12–14 January 2018.