Hindawi
Security and Communication Networks
Volume 2017, Article ID 4603237, 17 pages
https://doi.org/10.1155/2017/4603237

*Research Article*

# Privacy-Preserving Outsourced Auditing Scheme for Dynamic Data Storage in Cloud

**Tengfei Tu, Lu Rao, Hua Zhang, Qiaoyan Wen, and Jia Xiao**

*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China*

Correspondence should be addressed to Hua Zhang; zhanghua_288@bupt.edu.cn

Received 3 August 2017; Accepted 10 October 2017; Published 27 November 2017

Academic Editor: Xuyun Zhang

As information technology develops, cloud storage has been widely accepted for keeping volumes of data. Remote data auditing scheme enables cloud user to confirm the integrity of her outsourced file via the auditing against cloud storage, without downloading the file from cloud. In view of the significant computational cost caused by the auditing process, outsourced auditing model is proposed to make user outsource the heavy auditing task to third party auditor (TPA). Although the first outsourced auditing scheme can protect against the malicious TPA, this scheme enables TPA to have read access right over user's outsourced data, which is a potential risk for user data privacy. In this paper, we introduce the notion of User Focus for outsourced auditing, which emphasizes the idea that lets user dominate her own data. Based on User Focus, our proposed scheme not only can prevent user's data from leaking to TPA without depending on data encryption but also can avoid the use of additional independent random source that is very difficult to meet in practice. We also describe how to make our scheme support dynamic updates. According to the security analysis and experimental evaluations, our proposed scheme is provably secure and significantly efficient.

## 1. Introduction

In recent years, cloud computing has triggered profound technology changes in the field of information industry, promoting the rapid development of IoT (Internet of things) and big data that have gained so much attention in our daily social and economic activities [1]. As one of the vital services of cloud computing, cloud storage offers many attractive advantages, including the location-independent resources, ubiquitous network access, and on-demand storage space [2], motivating more and more enterprises and individuals to outsource their own data to cloud. Benefiting from the big data that is gathered together into the cloud, all kinds of data-driven techniques, such as data mining [3, 4] and data signal processing [5, 6], can be deployed upon the cloud storage environment to play their effective roles for creating more information wealth.

Despite that fact that many potential gains can be achieved based on the cloud storage, there also exists new threats from the cloud user's point of view. After user uploads all of her own data to cloud, one of the most pressing issues for user is how to verify the integrity of outsourced data stored at remote cloud side. Note that user loses the physical possession over her data in the context of data outsourcing, so it is clearly not feasible to directly apply traditional local data verification techniques that require access to the entire data, since both user and cloud servers cannot afford the heavy communication cost of frequently transferring all the outsourced data across a network to perform the data integrity verification. In this case, a variety of remote data auditing schemes [7–23] are designed, which can support the periodic integrity verifications upon outsourced data and simultaneously avoid transferring all these data for the minimum communication overhead. In addition, as an important feature to further reduce the burden on the user, public auditing is first proposed by Ateniese et al. [7] and has been adopted extensively by the subsequent improved schemes [13–22], which enables a third party auditor (TPA) to audit cloud servers on behalf of user for ensuring the outsourced data integrity. Nonetheless, happiness will not come so easily. When TPA is introduced, the following security risk arises.

*Malicious TPA*. TPA is considered as a trusted (or semitrusted, i.e., honest but curious) entity who cannot violate the auditing protocols in existing public auditing schemes [13–22]. But actually TPA might be untrusted [23]. Obviously, if the irresponsible TPA is lazy and does nothing, there is no difference between entrusting a malicious TPA and casting away all prior public auditing schemes for user.

In order to protect against the above malicious TPA, Armknecht et al. [23] first presented the outsourced auditing scheme Fortress to achieve this goal. Meanwhile, Fortress can protect the honest TPA from a malicious user, which is also a potential security issue that has not been considered in existing public auditing models. However, during the data preprocessing step, Fortress enables TPA to have read access right over the whole user's outsourced data in cloud, which is a significant limitation for practical applications. On the one hand, since Fortress exposes all outsourced data to TPA, in Fortress the only way for data privacy protection against curious TPA is to encrypt user's files before outsourcing. Nevertheless, as shown in [13, 14], although data encryption alone is an approach to relieve the privacy concern in cloud storage, encryption itself is often not enough to prevent user's data from leaking to TPA during the auditing process. On the other hand, in the era of big data, user's outsourced data is one kind of core business assets of CSP [15], which means the wealth and the future for CSP. Thus, CSP is selfish and has no incentives to reveal user's outsourced data to TPA under any circumstance. Besides, user is also often reluctant to expose her data to a third party [24]. In this case, for the various online cloud storage applications (e.g., online videos) where user cannot encrypt her data prior to outsourcing and only resorts to CSP to protect against outsourced data leakage, it is clearly that the direct extension of Fortress upon these online applications is impractical, since the design of revealing outsourced data to TPA is inevitable in Fortress. Therefore, it is necessary for an outsourced auditing scheme to include the privacy-preserving mechanism that is independent of data encryption to defend against curious TPA.

Furthermore, Fortress argues that the challenges for auditing cannot depend on any of the involved three entities since they might be malicious. So Fortress requires the aid of additional independent random source to produce the secure challenges for protecting against any malicious entity. However, as shown in [25], under the environment of cloud storage, the requirement of additional independent servers is already a strong assumption that is very difficult to meet in commercial contexts, and thus the similar assumption of requiring additional independent random source in Fortress is the same situation.

To address the above problems, in this paper, we introduce for outsourced auditing model a novel notion User Focus, which emphasizes the idea of restoring user's data autonomy lost in cloud storage setting. As shown in Sections 2.2 and 2.3, User Focus means to let user control all challenges throughout the process of outsourced auditing, avoiding the limitation of introducing the additional bitcoin pseudorandom source for generating challenges as in existing Fortress scheme. Furthermore, the user's autonomy enabled by User Focus is also reflected in that the data only needs to be preprocessed by user herself, avoiding the unfavorable situation in Fortress that TPA must fetch all user's data from cloud for initialization. With introducing User Focus, we propose an efficient and secure outsourced auditing scheme, which not only can defend against any malicious entity but also can protect user's outsourced data from curious TPA without depending on data encryption. In general, the contributions of this paper can be summarized as follows.

(1) By empowering user to play the leading role, we propose a formal User Focus outsourced auditing model along with the security definitions, which do not depend on any additional pseudo-random source. Our model extends the model of Fortress and takes into account the problem of preserving user's data privacy when introducing TPA, which is not covered in Fortress.

(2) Based on our proposed model, we design a concrete User Focus outsourced auditing scheme, the security of which is analyzed. Although the notion of User Focus empowers user to generate the challenges, it does not mean that a malicious user can do whatever she wants to do, since our scheme can also defend against the malicious user. In addition, under the environment of outsourced auditing, our scheme can enable user to predefine enough challenges for avoiding keeping user online all the time and also support the dynamic data updates by relying on the MHT authenticated data structure.

(3) Our scheme applies the RSA public key cryptography rather than the symmetric cryptography technology as utilized in Fortress and thus enables TPA to complete his preparatory work for auditing without requiring access to user's outsourced data at cloud side, which solves the significant performance problem faced by Fortress. We evaluate the run time of our scheme through concrete implementation when compared to Fortress. The evaluation results show that our solution is promising according to the improved performance.

The rest of this paper is organized as follows: Section 2 introduces the notion of User Focus and extends the outsourced auditing model along with the security definitions. In Section 3, we propose the User Focus outsourced auditing scheme, followed by the security analysis. In Section 4, the concrete algorithms for supporting dynamic updates are described. Section 5 gives the implementation results with the performance evaluation. Section 6 overviews the related work. Finally, Section 7 gives the concluding remark of this paper.

## 2. Problem Statement

In this section, we introduce the notion of User Focus, which should be an important requirement for user in the setting of storage outsourcing. Then we propose a formal User Focus outsourced auditing model and the corresponding security definitions.

### *2.1. Outsourced Auditing for Cloud Storage.*

Various remote data auditing schemes [7–23] provide a cloud user the ability of confirming that her outsourced data is intact at the cloud, with the advantage that it is no need to fetch the data from cloud. The private auditing schemes [8, 12] only include two entities: a user and CSP, where user has to audit CSP regularly by herself to ascertain that CSP holds the stored data all the time. In view of user's limited resources and the expensive computation cost incurred by the frequent audits, the public auditing schemes are proposed [13–22], which introduce a trusted TPA to perform the above auditing task. By employing TPA, user is alleviated from the auditing burden. However, trusted TPA is just an ideal hypothesis in real world.

Based on the prior auditing solutions, the first outsourced auditing scheme [23] is proposed to defend against the malicious TPA. Compared with the public auditing model, although there are also three entities included in outsourced auditing setting, the major difference is that anyone of the three entities might be dishonest, as described as follows:

(i) User might be a dishonest entity, who uploads her data to the cloud servers. User needs to remotely update outsourced data as necessary. And user might maliciously deny the fact that the auditing work performed by TPA against CSP is correct for claiming compensations from TPA.

(ii) CSP might be a dishonest entity, who is the owner of cloud servers (so CSP and the cloud servers are not distinguished in our paper), holding a large amount of resources to store and maintain outsourced data. CSP might try to cheat on auditor when data loss or data corruption occurs in cloud.

(iii) TPA might be a dishonest entity, who has capabilities and expertise, on behalf of user, to regularly audit CSP for confirming the intactness of user's data in cloud. But TPA might be lazy and fail to perform the auditing task required by user. In addition, TPA might be curious and try to deduce user's outsourced data during performing his auditing task against CSP.

*2.2. User Focus.* "Customer Focus" is a marketing term that means keeping customer in mind and bringing customer the good experience of services. Clearly, "The customer is a God" is not only the truth in business, but also a similar situation in our cloud storage environment, where the user is the targeted customer of CSP and all kinds of auditing solutions. User experience is a determining factor signifying whether an auditing scheme is accepted or not in practice. If user experience of a scheme is poor, no matter whatever sophisticated technology is adopted, it is impossible for this scheme to get a practical application widely.

In spite of various auditing schemes that are proposed to cover many critical issues, but user experience is ignored. On the one hand, whichever of the private auditing schemes is based on the design that the auditing protocol needs to be frequently executed by user, resulting in the nonnegligible computation overhead at user side. Obviously, this is a terrible experience for user who just holds limited resources, such as smartphone. On the other hand, within public auditing schemes, the assumption of a "trusted" TPA is also a bad experience for user, since it is impractical for every ordinary user to find an idealistic "trusted" TPA.

Note that the purpose of remote data auditing is to provide user with a mechanism for confirming the security of her outsourced data. Here, user is regarded as the demander. Therefore, user ought to be put at the center when designing the auditing scheme, and her experience should not be ignored. For this reason, we introduce the notion of *User Focus*, which is defined as follows:

> *User is the initiator of the auditing protocols and controls all the challenges, who can timely receive the exception message of her outsourced data without frequently working, since both CSP and TPA have to frequently provide all proofs around user's needs.*

We stress that User Focus does not mean a malicious user can do whatever she wants. As shown in Section 3.2.5, although user controls all the challenges, our scheme can still ensure the security for honest TPA to defend against the malicious user.

Actually, the essential concept of cloud services is the centralized management of user's data in cloud, which casts a psychological shadow on user. To save the storage space or availably access data without restriction of time and location, user is required to outsource her data to cloud, which means that user is no longer able to physically possess all her data. In other words, data outsourcing makes user lose the physical ownership and autonomy of her data, which is one of the main obstacles for the application and promotion of cloud storage.

When outsourcing data to cloud is inevitable, the notion of User Focus can make user step out of the psychological shadow brought by cloud storage, enabling user to enjoy cloud services more confidently. User Focus expresses the idea that let user dominate her own data, which is realized in our model by the way that "the gain offsets the loss." By empowering user to gain the control right of challenges, user gets back the autonomy that is lost after data outsourcing and is able to proactively check the intactness of the specific data just by adjusting challenges, which can bring user the feeling that there is really no difference for intactness assurance between her data stored in local disks and outsourced in cloud, since everything is under control from the user's perspective. Apparently, User Focus will be an attractive property for user. Especially when our proposed scheme is implemented as a cloud service and CSP hopes that this service can be broadly accepted by potential customers, User Focus will be a fascinating feature to persuade every customer to try this cloud service.

*2.3. User Focus Outsourced Auditing Model.* Now, we begin with the description of User Focus outsourced auditing model, as shown in Figure 1. To avoid the considerable user online direct interactions during the frequent TPA's auditing against CSP, user will pregenerate enough challenges which can support running the auditing protocol for ages. Since the size of a challenge can be very small (e.g., only 88 bytes for a challenge as shown in Section 5), all these pregenerated challenges can be stored in user's email box (e.g., only 8.5 MB email box memory is required for storing

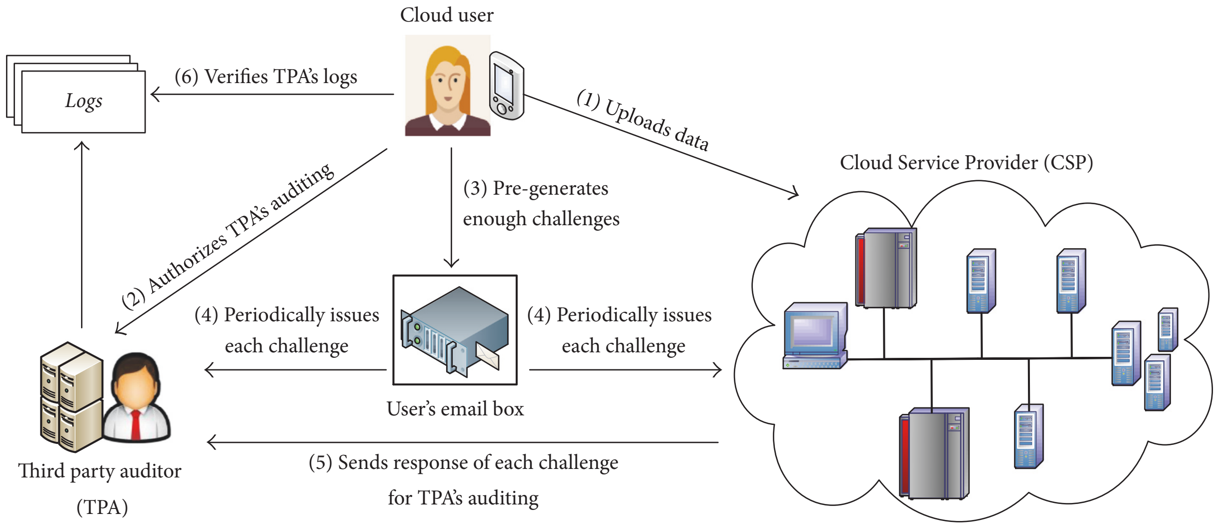

Cloud user
Logs
(6) Verifies TPA's logs
(1) Uploads data
Cloud Service Provider (CSP)
(3) Pre-generates enough challenges
(2) Authorizes TPA's auditing
(4) Periodically issues each challenge
(4) Periodically issues each challenge
User's email box
Third party auditor (TPA)
(5) Sends response of each challenge for TPA's auditing

Figure 1: User Focus outsourced auditing architecture.

100,000 pregenerated challenges). In this case, after user uploads her data to cloud and delegates the auditing work to TPA, based on the built-in timer of email box, each challenge will be periodically issued from user's email box to automatically trigger TPA's auditing against CSP without involving user herself. Furthermore, TPA must produce the corresponding log when he finishes each auditing against CSP. Based on the contract established by three entities, TPA has to immediately inform user (e.g., gives user a phone call) as soon as any exceptional situation about user's outsourced data is detected. If TPA is lazy and hence does not find out the data corruption happening upon the challenged data blocks, once user launches her auditing to TPA by checking TPA's logs, the lazy TPA will be identified with deterministic evidence. Finally, when all the pregenerated challenges are exhausted, user will add the new challenges to her email box. However, note that such operation for adding challenges and the auditing against TPA's logs are only rarely executed by user, so user can go offline most of the time throughout our model.

In contrast to existing outsourced auditing model of [23], one major difference in our model is that the notion of User Focus is introduced, enabling user to play the leading role on her outsourced data with minimal effort. So user is the only one who can possess the additional secret key, besides a signing key pair, to preprocess the data. Based on the modern legal society with the spirit of contract, our model can achieve that not any honest entity will be wronged and that any malicious entity can be captured. More specifically, User Focus outsourced auditing model consists of five protocols *Setup*, *Preproccess*, *AuditCSP*, *AuditTPA*, and *IdentifyMalice*, which are described as follows.

*(1) The Setup Protocol.* For each involved entity, this randomized protocol produces a public-private key pair used for signing subsequently. For simplicity, we always imply that every entity uses as inputs its own private key and the public keys of the other entities. Moreover, user randomly generates the secret key $SK_{Tag}$, which will be used to preprocess the data before uploading it to cloud servers.

*(2) The Preproccess Protocol.* This randomized protocol, launched by user, takes as input user's secret key $SK_{Tag}$ and a file $\mathbf{F}$ owned by user. The output $\widetilde{\mathbf{F}} := \{\mathbf{F}, (\Psi, \tau)\}$ marks the outsourced data that will be uploaded to CSP. Observe that $\widetilde{\mathbf{F}}$ should include not only the file $\mathbf{F}$, but also some metadata $(\Psi, \tau)$. The role of metadata is reflected in three aspects which (i) enables TPA to frequently verify the response from CSP, (ii) prevents TPA from deducing any data information of the file $\mathbf{F}$ for privacy-preserving, and (iii) enables user to effectively audit the log files produced by TPA about his past auditing work. Formally, the following holds:

$$\mathit{Preproccess}\colon \left[\mathit{User} : SK_{Tag}, \mathbf{F}\right] \longrightarrow \left[\mathit{User} : \widetilde{\mathbf{F}}\right]. \quad (1)$$

Furthermore, three entities need to agree on the financial contract that defines their respective liabilities and liquidated damage, as well as the system parameters set $\Gamma$ that will be used throughout the outsourced auditing scheme. Let $\mathit{Agree}[\mathbb{E}_1, \mathbb{E}_2, [D]]$ denote that both entities $\mathbb{E}_1$ and $\mathbb{E}_2$ establish agreement on the data $D$. Then the contract consists of three agreements as follows. Formally,

$$\begin{aligned} &\mathit{Preproccess}\colon \mathit{Contract}\,[\mathit{User},\ \mathit{CSP},\ \mathit{TPA}] \\ &\quad\longrightarrow \Big[\mathit{Agree}\left[\mathit{User}, \mathit{CSP}, \left[\widetilde{\mathbf{F}}, \Gamma\right]\right]; \\ &\quad \mathit{Agree}\,[\mathit{CSP}, \mathit{TPA}, [\Psi, \Gamma]]\,; \\ &\quad \mathit{Agree}\,[\mathit{User}, \mathit{TPA}, [\Psi, \Gamma]]\Big]\,. \end{aligned} \quad (2)$$

If all the above agreements succeed, the *Preproccess* protocol runs to completion, meaning that $\widetilde{\mathbf{F}}$ is outsourced and the contract is established by three entities. At last, user deletes $\widetilde{\mathbf{F}}$ from local.

*(3) The AuditCSP Protocol.* This protocol is launched by a challenge $\mathbb{C}$, which is generated from user and sent to both CSP and TPA with user's signature. After receiving $\mathbb{C}$, CSP takes $\widetilde{\mathbf{F}}$ as input and TPA takes $(\Psi, \Gamma)$ as input, running this protocol, respectively. At the end, TPA must produce a binary value $\mathbb{D}_{TPA}$ that expresses whether CSP's running results pass his audit or not. If $\mathbb{D}_{TPA} = NO$, TPA must immediately report this exception to user. In addition, the auditing log $\Theta$ corresponding to $\mathbb{D}_{TPA}$ must be generated by TPA for each challenge. Formally, the following holds:

$$\begin{aligned} &AuditCSP\colon \left[User : \mathbb{C}; CSP : \widetilde{\mathbf{F}}; TPA{:}\Psi, \Gamma\right] \\ &\quad \longrightarrow \left[TPA : \mathbb{D}_{TPA}, \Theta\right]. \end{aligned} \tag{3}$$

If $\mathbb{D}_{TPA} = YES$, indicating that CSP passes TPA's audit for the current challenge, so user will not receive the emergency report from TPA. Then after a certain time interval, this *AuditCSP* protocol will be repeated again by a new challenge.

*(4) The AuditTPA Protocol.* As long as user wants to check if TPA correctly executes *AuditCSP* protocol for all past challenges, the *AuditTPA* protocol can be launched by user. This protocol is a deterministic algorithm with one-time process, which takes as input user's secret key $SK_{Tag}$, the metadata $\Psi$, and the log files $\Theta$ stored by TPA. The output for user is a binary value $\mathbb{D}_{User}$ that is either *YES* or *NO*. $\mathbb{D}_{User} = YES$ indicates that *TPA* is honest. Otherwise, *TPA* is malicious. Formally,

$$AuditTPA\colon \left[User : SK_{Tag}; TPA : \Psi, \Theta\right] \longrightarrow \mathbb{D}_{User}. \tag{4}$$

In practice, *AuditTPA* will be executed much less frequently. If TPA is lazy when data loss occurs, once user executes *AuditTPA*, the output $\mathbb{D}_{User}$ must be *NO*, which can be taken as evidence for user to claim compensations from TPA based on their financial contract. Since *AuditTPA* can be launched without requiring the real-time output, user has enough time to execute this protocol. To avoid the breach of contract and financial penalties, TPA has to be honest all the time.

*(5) The IdentifyMalice Protocol.* If all entities are honest, this protocol is not necessary. This protocol will not be invoked until any dishonest entity exists. *IdentifyMalice* is a deterministic algorithm that models the scenario of "forensic debate." Various *proofs* about all the malicious activities can be produced during the operations of previous protocols in our model. As a way for the honest entity to claim compensations or prove its innocence, all the *proofs* will be presented with nonrepudiation and taken as input by the last *IdentifyMalice*.

$$IdentifyMalice\colon \left[entity : proofs\right] \longrightarrow \mathbb{D}_{ent}. \tag{5}$$

The output $\mathbb{D}_{ent}$ is a binary value either *YES* or *NO*, indicating whether the entity is honest or malicious. Obviously, the existing of *IdentifyMalice* will provide deterrence against all the possible misconducts. Since any malicious activity is bound to be identified, to avoid the financial penalties or even the legal liability, every involved entity has to be honest.

*2.4. Security Definitions.* For security, a User Focus outsourced auditing scheme should be correct and sound, simultaneously, supporting privacy protection against a curious TPA. The definitions are explained as below.

*Definition 1.* The correctness of User Focus outsourced auditing scheme requires that if all the involved entities are honest, for all keys output by *Setup* protocol and for all files $\mathbf{F} \in \{0,1\}^*$ and the corresponding $\widetilde{\mathbf{F}}$ output by *Preproccess* protocol, TPA always accepts probability 1 at the end of each *AuditCSP* protocol run and likewise the user at the end of each *AuditTPA* protocol. In addition, the correctness can be defined from another aspect that *IdentifyMalice* protocol will never (i.e., with probability 0) be invoked, since if everything goes well then none of the entities will abort for invoking the last protocol.

To define the soundness of our model, we start by the notion of liability in [23]. In contrast to the traditional public auditing security models, providing security to TPA is also an important aspect that should not be ignored in outsourced auditing setting. Especially in situations where some problems occur (e.g., the outsourced data has been lost in CSP), TPA must not be blamed as long as he can prove that he has fulfilled the auditing obligation according to the protocols. Namely, by providing his log files, if TPA convinces user with a high probability that he is taking over guarantees about the service quality of user's outsourced data, then TPA is actually honest to finish his auditing task for each past challenge against CSP. This is the definition of liability which is formalized in [23]. Although the security model of [23] splits the definition of soundness into two parts, extractability and liability, we stress that the soundness of our User Focus outsourced auditing model can be defined as a whole, since liability becomes an intrinsic property and implicitly exists in our model without having to be proved for each instantiation.

More precisely, as shown in Section 2.3, throughout the *AuditCSP* protocol, TPA does not use any secret key when he performs his auditing work, which means that all operations conducted by TPA are deterministic in *AuditCSP* protocol. Firstly, the metadata $\Psi$ and the public parameters set $\Gamma$ are confirmed by the contract after *Preproccess* protocol is over. Secondly, the challenge $\mathbb{C}$ is individually generated by user, which is incontestable for user because of her signature. Therefore, TPA has no influence on all the involved parameters that are applied to complete his auditing work. The *IdentifyMalice* protocol can reconstruct the results that should have occurred for TPA during each execution of *AuditCSP* protocol and compare these results with TPA's logs to judge if TPA is lazy or not. Clearly, these results are destined to be the objective and incontestable proofs once the *AuditCSP* protocol is executed. For this reason, no matter whether any malicious entity exists, a User Focus outsourced auditing scheme is bound to produce the objective proofs to protect a honest TPA who correctly performs his work, while guaranteeing that any lazy TPA can be identified as long as TPA's logs do not conform to the objective proofs as above.

In conclusion, our model is naturally sound for TPA in the case that TPA is honest, since honest TPA has been implicitly protected from the malicious user. So in the following we can define the soundness for User Focus outsourced auditing scheme by excluding the situation that user is corrupted.

Now, we start by using the similar game employed in [23]. Firstly, given an adversary $\mathscr{A}$ who corrupts any entity $\mathbb{E} \in \{CSP, TPA\}$. Subsequently, the adversary $\mathscr{A}$ takes over the role of corrupted entity $\mathbb{E}$ and plays the training game with an environment as follows:

(1) By running the *Setup* protocol, the environment generates all the public-private keys for the involved entities and the secret key for user. The adversary $\mathscr{A}$ obtains all the keys of the corrupted entity $\mathbb{E}$.

(2) For learning the knowledge of various outputs provided to the corrupted entity $\mathbb{E}$, adversary $\mathscr{A}$ adaptively interacts with the environment, which plays the role of the honest entity. Given any file $\mathbf{F}$, $\mathscr{A}$ can request the execution of *Preproccess* protocol which outputs the outsourced data $\widetilde{\mathbf{F}}$. Afterwards, $\mathscr{A}$ can request the executions of *AuditCSP* and *AuditTPA* protocols upon any above $\widetilde{\mathbf{F}}$ stored in cloud. All these protocol executions can be arbitrarily interleaved with each other.

(3) Finally, after finishing all the learning, adversary $\mathscr{A}$ outputs a challenge file $\mathbf{F}$ that should be stored and the description of a cheating prover $\mathscr{CP}$ corresponding to this file $F$.

The cheating prover $\mathscr{CP}$ is *ε-admissible* if the probability that the honest entity interacts with $\mathscr{CP}$ without aborting the protocols is at least $\varepsilon$. Here, the probability is over the coins of the honest and malicious entities. To formalize the definition of soundness, we adopt the notion of *extractor* algorithm $\mathbf{EA}(\cdot)$, which takes as inputs all the information provided to the honest entity, and the description of the cheating prover $\mathscr{CP}$. The output of $\mathbf{EA}(\cdot)$ is the file $\mathbf{F}$. In particular, $\mathbf{EA}(\cdot)$ is given non-black-box access to the cheating prover $\mathscr{CP}$ and can rewind it. Formally we have the following.

*Definition 2.* A User Focus outsourced auditing scheme is *ε-sound* with respect to any corrupted entity $\mathbb{E} \in \{CSP, TPA\}$ if there exists an extractor algorithm $\mathbf{EA}(\cdot)$ such that, for any honest entity establishing the contract on a file $\widetilde{\mathbf{F}} = \{\mathbf{F}, (\Psi, \tau)\}$ and interacting with the *ε-admissible* cheating prover $\mathscr{CP}$, if the whole system remains running without being aborted by honest entity, the extractor can recover $\mathbf{F}$ from $\mathscr{CP}$ with overwhelming probability.

Based on the above definition, we can informally say that if the *ε-admissible* prover $\mathscr{CP}$ convinces the honest entity with a sufficient level of probability, then the challenged data is actually intact and extractable.

Moreover, since the user is often reluctant to reveal her outsourced data to TPA, privacy protection is necessary, as defined in the following.

*Definition 3.* A User Focus outsourced auditing scheme with privacy-preserving requires that no matter what running results are obtained by TPA during the operation of this scheme, TPA cannot deduce any privacy information of user's outsourced file $\mathbf{F}$—except possibly with negligible probability.

## 3. The Proposed Scheme

In this section, based on the proposed model, we present a concrete User Focus outsourced auditing scheme, the security of which is analyzed according to our security definitions.

*3.1. Preliminaries.* Given that the user wants to outsource her file $\mathbf{F}$ to CSP. The file $\mathbf{F}$ can be seen as a set of $\mathbf{n}$ blocks: $\mathbf{F} = \{\mathbf{b_1},\mathbf{b_2},\ldots,\mathbf{b_n}\}$. We first introduce several necessary techniques, which are important under the environment of remote data auditing.

*Blockless Verification and Homomorphic Tags [7].* Blockless verification technique enables verifier to audit if the cloud servers possess certain file blocks, without having to retrieve these actual file blocks from the cloud. Blockless verification is essential since it is expensive and impracticable to retrieve all specified file blocks for frequent audits. Homomorphic tags can meet the requirement of blockless verification. Given two file blocks $b_i$ and $b_j$, along with their corresponding homomorphic tags $\tau_i$ and $\tau_j$, then the combination of $\tau_i$ and $\tau_j$ into a value $\tau_{i+j}$ will correspond to the tag of the sum of blocks $b_i$ and $b_j$.

*Merkle Hash Tree (MHT) [26].* It is a kind of authenticated data structure, which can be used to efficiently and securely prove that, in a given set of elements, the value of each element and the order of all elements are both undamaged. Based on a collision-resistant cryptographic hash function $\mathbf{H}(\cdot)$, MHT can be constructed as a binary tree, by the rule that the value of each parent node is defined as $\mathbf{H}(left\_child\ value \parallel right\_child\ value)$, where the leaf nodes are the hash values of authentic file blocks. In order to realize the authentication of each leaf node in MHT, *Leaves Auxiliary Authentication Information (LAAI)* is defined as the siblings of the nodes on the path from the leaf nodes to the MHT *root*. An example of MHT is shown in Figure 2. Assume that auditor possesses the *root* and wants to authenticate the appointed leaf nodes $\mathscr{LN} = \{h_3, h_6\}$ provided by the adversary. According to the given $LAAI_{\mathscr{LN}}$ from adversary, auditor can compute $h_b, h_c, h_e, h_f$ in order and finally computes $root^* = \mathbf{H}(h_e \parallel h_f)$. If $root^* = root$, auditor accepts all the leaf nodes of $\mathscr{LN}$; otherwise, it rejects them. In our paper, the order of leaf nodes in MHT is treated from left to right. By following the given order, any leaf node can be located and authenticated by the *root* with its corresponding *LAAI*. Obviously, all the leaves of *MHT* can be authenticated just by the *root*.

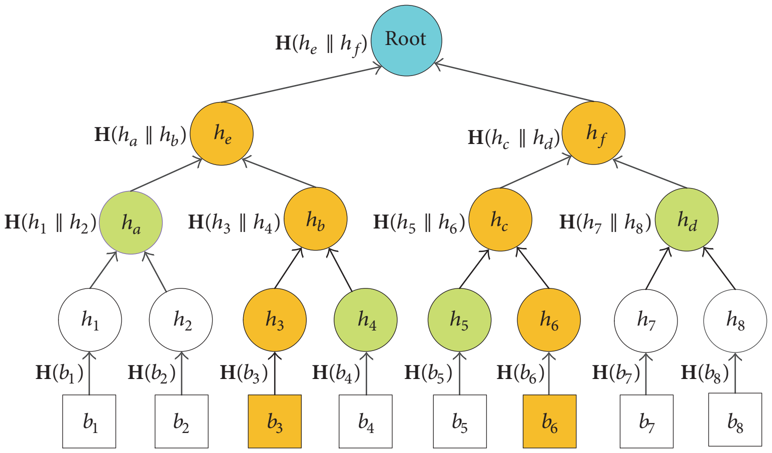


FIGURE 2: The MHT based on 8 file blocks. For the appointed leaf nodes set $\mathscr{L}\mathscr{N} = \{h_3, h_6\}$, the corresponding $LAAI_{\mathscr{L}\mathscr{N}} = \{h_4, h_5, h_a, h_d\}$.

*3.2. Scheme Construction.* Now, we detail the specifications of User Focus outsourced auditing scheme, comprising five protocols *Setup*, *Preproccess*, *AuditCSP*, *AuditTPA*, and *IdentifyMalice*.

*3.2.1. Design of Setup.* For each entity $\mathbb{E} \in \{User, CSP, TPA\}$, a corresponding public-private key pair $(pk_{\mathbb{E}}, sk_{\mathbb{E}})$ is generated for signature by executing the signature key generation algorithm $\mathscr{Sign\_Key}(1^k)$. Based on these key pairs, the secure communication links between any two entities can be established and authenticated in terms of the Transport Layer Security protocols.

In order to construct the homomorphic tags, the user relies on the RSA algorithm to output the public key $PK_{Tag} = (N, e)$, where $N = pq$ is the product of two large primes $p$ and $q$; $e$ is a random large prime chosen by user. Let $d = e^{-1} \bmod \phi(N)$; then user obtains the private key $SK_{Tag} = (N, d)$. The $PK_{Tag}$ will be sent to TPA, and $SK_{Tag}$ is kept by user as her secret.

*3.2.2. Design of Preproccess.* This protocol, comprising the following four phases, is initiated and dominated by user who holds the file $\mathbf{F}$.

*(1) Generating Metadata* $(\Psi, \tau)$. Let $\mathbf{H} : \{0,1\}^* \rightarrow \mathbb{Z}_N^*$ be a full-domain hash. For each block $\{b_i\}_{1\le i\le n}$, user calculates $h_i \leftarrow \mathbf{H}(b_i)$ as the corresponding leaf node of MHT. When the leaf nodes of all the blocks are generated, user can compute the *root* of MHT $\Psi$ by iteratively hashing as shown in Section 3.1. Note that user just needs to compute the *root*, so it is not necessary for user to construct and store the whole MHT $\Psi$.

Then user randomly chooses a public element $u \overset{R}{\leftarrow} \mathbb{Z}_N^*$ and a secret element $x \overset{R}{\leftarrow} \mathbb{Z}_{\phi(N)}^*$ and computes $\omega \leftarrow u^x \bmod N$. In terms of her secret key $SK_{Tag} = (N, d)$, for each block $b_i$, user computes the corresponding tags $\tau_i \leftarrow (u^{h_i} \cdot \omega^{b_i})^d \bmod N$. Let $\widetilde{\mathbf{F}}$ denote the processed file, described as follows:

$$\widetilde{\mathbf{F}} \coloneqq \{\mathbf{F} = \{b_i\}, leaves = \{h_i\}, \tau = \{\tau_i\} \mid 1 \le i \le n\}. \quad (6)$$

*(2) Uploading* $\widetilde{\mathbf{F}}$ *to CSP.* After receiving $\widetilde{\mathbf{F}}$ from user, for each $b_i \in \mathbf{F}$, CSP recomputes $h_i^* \leftarrow \mathbf{H}(b_i)$ and compares $h_i^*$ with the corresponding $h_i \in leaves$. If $h_i{}^* \neq h_i$, which signifies that there exists inconsistent data within $\widetilde{\mathbf{F}}$, thus $\widetilde{\mathbf{F}}$ will be rejected by CSP at once. If $\widetilde{\mathbf{F}}$ passes CSP's verification, based on all the $h_i \in leaves$, CSP can reconstruct the whole MHT, denoted by $\Psi_{CSP}$ with $root_{CSP}$. After this, CSP uses its signature $sign_{CSP}$ to respond to user as follows:

$$response_{CSP} \coloneqq \{(root_{CSP}, n) \parallel sign_{CSP}\}. \quad (7)$$

Upon receiving $response_{CSP}$, user verifies CSP's signature and then checks whether $root_{CSP}$ is equal to $root$ stored in her local. If $root_{CSP} = root$, indicating that the same MHT has been constructed at CSP side, then user stores $response_{CSP}$ and sends out her acceptance to CSP. Otherwise, user aborts the protocol.

*(3) Authorizing TPA's Auditing.* When receiving the acceptance from user, CSP immediately sends all leaf nodes $\{h_i\}_{1\le i\le n}$ to TPA. Similarly, TPA also reconstructs the whole MHT, denoted by $\Psi_{TPA}$ with the $root_{TPA}$ and sends his response to user:

$$response_{TPA} \coloneqq \{(root_{TPA}, n) \parallel sign_{TPA}\}. \quad (8)$$

User executes the same verification process for $response_{TPA}$ as the way for $response_{CSP}$. If $root_{TPA} = root$, $response_{TPA}$ is accepted and stored in local by user. At last, user sends the public key $PK_{Tag} = (N, e)$ to TPA, meaning that the authorization of auditing CSP is provided to TPA.

It is worth mentioning that we make CSP, on behalf of user, to transfer all leaf nodes of MHT to TPA in above process. This is in order to reduce bandwidth cost of user, who might have the limited bandwidth resource. We stress that user does not have to be worried about any security issue by this way. If malicious CSP tries to provide any false leaf node to TPA, as long as the hash function $\mathbf{H}(\cdot)$ preserves the collision-resistant property, then $root_{TPA}$ will not be equal to the $root$ at user side with overwhelming probability.

Obviously, such unusual condition will be detected by user, who will terminate the auditing authorization immediately.

*(4) Agreeing on the Contract.* A constant number $l_{\mathbb{C}}$ needs to be assigned by user for denoting the amount of challenged blocks during each TPA's auditing to CSP. Let $\Gamma := \{root, l_{\mathbb{C}}, n, u, \omega\}$ denote the set of system parameters, which is signed by user and sent to both CSP and TPA. Afterwards, CSP and TPA must respond to user with their own signatures upon $\Gamma$, respectively, which means that all entities sign the contract for reaching an agreement. Note that CSP and TPA will store user's signature $sign_{User}(\Gamma)$ in their locals, respectively, to protect themselves from a dishonest user.

At last $\widetilde{\mathbf{F}}$ will be deleted from user side to complete this protocol. Different from the corresponding *Store* protocol of [23] that enables TPA to access the whole user's outsourced data in cloud, throughout our *Preproccess* protocol TPA can only obtain from CSP the hash values of file blocks, which contributes to protecting user's data privacy from the curious TPA in the context of no encryption upon the outsourced data.

*3.2.3. Design of AuditCSP.* When each pregenerated challenge is released from user's email box, this protocol will be launched, described as follows.

*(1) User's Challenge.* Given a specific point-in-time $t$, the form of corresponding challenge $\mathbb{C}_t$ is defined as $\mathbb{C}_t := t \parallel (PRF_t, PRP_t, sign_{user})$, where $PRF_t$ is the key for public pseudo-random function $\mathbb{F}$, $PRP_t$ is the key for public pseudo-random permutation $\mathbb{P}$, and $sign_{User}$ is user's signature.

(i) $\mathbb{F}$: $PRF_t \times \{0,1\}^{\log_2(n)} \longrightarrow \mathbb{Z}_N^*$.

(ii) $\mathbb{P}$: $PRP_t \times \{0,1\}^{\log_2(n)} \longrightarrow \{0,1\}^{\log_2(n)}$.

Both $PRF_t$ and $PRP_t$ can be randomly or selectively generated by user. The meaning of selectively lies in that user can cover any specific block in a challenge by selectively generating the particular $PRP_t$.

Let $\mathbb{F}(PRF_t, j)$ denote $\mathbb{F}$ keyed with key $PRF_t$ applied on the input $j$. According to public $\mathbb{F}$ and $\mathbb{P}$, after receiving the challenge $\mathbb{C}_t$ and checking user's signature, for each $j \in [1, l_{\mathbb{C}}]$, CSP can calculate the locations $i_{j,t}$ of challenged blocks by $i_{j,t} \longleftarrow \mathbb{P}(PRP_t, j) \in [1, n]$ and the associated values $v_{j,t}$ such as $v_{j,t} \longleftarrow \mathbb{F}(PRF_t, j) \in \mathbb{Z}_N^*$. Here, we use $\mathscr{S}_t$ to denote the $l_{\mathbb{C}}$-elements set $\{(i_{j,t}, v_{j,t})\}_{1\le j\le l_{\mathbb{C}}}$, which can be generated based on $\mathbb{C}_t$ at time $t$.

*(2) CSP's Proof.* After computing $\mathscr{S}_t$, by randomly choosing a secret interfering element $\xi_t \overset{R}{\longleftarrow} \mathbb{Z}_N^*$ corresponding to the challenge $\mathbb{C}_t$, CSP computes and signs its proof $\mathscr{P}_t := (\mathscr{B}_t, \tau_t, \omega_t, u_t) \parallel sign_{CSP}$ as below, where both $u$ and $\omega$ are the public parameters:

$$\begin{aligned}\mathscr{B}_t &\longleftarrow \left(\sum_{(i_{j,t},v_{j,t})\in\mathscr{S}_t} v_{j,t}\cdot b_{i_{j,t}}\right)+\xi_t \in \mathbb{Z},\\ \tau_t &\longleftarrow \prod_{(i_{j,t},v_{j,t})\in\mathscr{S}_t} {\tau_{i_{j,t}}}^{v_{j,t}} \bmod N,\\ \omega_t &\longleftarrow \omega^{\xi_t} \bmod N,\\ u_t &\longleftarrow u^{\xi_t} \bmod N.\end{aligned} \tag{9}$$

Then CSP transfers $\mathscr{P}_t$ to TPA. Since the size of the combined block $\mathscr{B}_t$ within $\mathscr{P}_t$ is roughly the size of a single file block, so CSP has a much smaller communication cost than for transferring $l_{\mathbb{C}}$ challenged blocks. The key attribute of $\mathscr{P}_t$ is that there is no data privacy that can be deduced from $\mathscr{P}_t$ and leaked to TPA, as will be analyzed in Section 3.3.

*(3) TPA's Verification.* Upon receiving challenge $\mathbb{C}_t$ from user's email box, TPA can compute $\mathscr{S}_t$ by the same way as CSP. Then TPA calculates his auditing parameter $\eta_t$ as below, where all the hash values $h_{i_{j,t}}$ can be found from the leaf nodes of the MHT $\Psi_{TPA}$ stored at TPA side.

$$\eta_t \longleftarrow \prod_{(i_{j,t},v_{j,t})\in\mathscr{S}_t} u^{v_{j,t}\cdot h_{i_{j,t}}} \bmod N. \tag{10}$$

Subsequently, TPA parses CSP's proof $\mathscr{P}_t$ to obtain $\mathscr{B}_t, \tau_t$, and $\omega_t$. Now TPA can use user's public key $PK_{Tag} = (N, e)$ to check whether

$$\frac{\eta_t\cdot\omega^{\mathscr{B}_t}}{\omega_t} \overset{?}{=} {\tau_t}^{e} \bmod N. \tag{11}$$

If so, TPA outputs *YES*. Otherwise, according to the signed contract, TPA must report to user that CSP does not pass the integrity auditing upon the challenged blocks. It is clear that the interfering element $\xi_t$, generated and kept secretly by CSP, has no influence on the correctness of TPA's verification. But $\xi_t$ plays an important effect on the privacy protection of outsourced data, as will be discussed in Section 3.3.

In addition, to prove that the auditing work related to the challenge $\mathbb{C}_t$ has been performed correctly, TPA must generate the corresponding $\log\Theta_t := (\mathbb{C}_t, \mathscr{P}_t, \eta_t)$, which records the necessary information for the auditing against TPA launched by user.

*3.2.4. Design of AuditTPA.* In our scheme, we stress that TPA has to inform user as soon as anything has gone wrong during his auditing. If TPA is lazy and does nothing, then he cannot detect the outsourced data corruption in time, and user will find out such TPA's misbehavior when executing the *AuditTPA* protocol upon a batch of TPA's past logs.

Specifically, user randomly selects a $k$-elements point-in-time set $\mathscr{T} = \{t_i\}_{1\le i\le k}$, where each $t_i$ indicates the time when a past challenge is issued from user's email box. User sends $\mathscr{T}$ to TPA. According to $\mathscr{T}$, TPA parses his local logs to obtain the $\{\mathscr{P}_t = (\mathscr{B}_t, \tau_t, \omega_t, u_t), \eta_t\}_{t\in\mathscr{T}}$ and accumulates these values into the corresponding single value, respectively:

$$\mathscr{B}_{\mathscr{T}} \longleftarrow \sum_{t\in\mathscr{T}} \mathscr{B}_t,$$

$$
\begin{aligned}
\tau_{\mathscr{T}} &\longleftarrow \prod_{t\in\mathscr{T}} \tau_t \bmod N,\\
\omega_{\mathscr{T}} &\longleftarrow \prod_{t\in\mathscr{T}} \omega_t \bmod N,\\
u_{\mathscr{T}} &\longleftarrow \prod_{t\in\mathscr{T}} u_t \bmod N,\\
\eta_{\mathscr{T}} &\leftarrow \prod_{t\in\mathscr{T}} \eta_t \bmod N.
\end{aligned}
\tag{12}
$$

Note that, for each $t \in \mathscr{T}$, TPA can also read the key $PRP_t$ from $\mathbb{C}_t$ within local log $\Theta_t$ and then compute the locations of all challenged blocks related to the time set $\mathscr{T}$ by using the pseudo-random permutation $\mathbb{P}(PRP_t, j), 1 \leq j \leq l_{\mathbb{C}}$. Let $locat := \{i_{j,t}\}_{t\in\mathscr{T}}$ denote the set of all specified locations. Since some blocks might be challenged more than once, the repetitive locations will be taken away to ensure that every location in *locat* is unique.

For each $i_{j,t} \in locat$, TPA reads from his local MHT all the corresponding leaf nodes $h_{i_{j,t}}$ that constitute the set $\mathscr{L}\mathscr{N}_{\mathscr{T}} := \{h_{i_{j,t}}\}$. In addition, according to $\mathscr{L}\mathscr{N}_{\mathscr{T}}$, TPA generates the corresponding Leaves Auxiliary Authentication Information, $LAAI_{\mathscr{L}\mathscr{N}_{\mathscr{T}}}$ (as shown in Section 3.1). Now, TPA constructs and signs his proof $\beta_{\mathscr{T}} := (\mathscr{B}_{\mathscr{T}}, \tau_{\mathscr{T}}, \omega_{\mathscr{T}}, u_{\mathscr{T}}, \eta_{\mathscr{T}}, \mathscr{L}\mathscr{N}_{\mathscr{T}}, LAAI_{\mathscr{L}\mathscr{N}_{\mathscr{T}}})$, which is sent to user as TPA's response.

Upon receiving $\beta_{\mathscr{T}}$ and checking TPA's signature, user first uses $\mathscr{L}\mathscr{N}_{\mathscr{T}}$ and $LAAI_{\mathscr{L}\mathscr{N}_{\mathscr{T}}}$ to compute the new $root^*$ of MHT and then compares $root^*$ with the *root* held by herself. User accepts $\mathscr{L}\mathscr{N}_{\mathscr{T}}$ only when $root^* = root$, which indicates that all values of $\mathscr{L}\mathscr{N}_{\mathscr{T}}$ pass user's verification. If $root^* \neq root$, a malicious TPA can be identified immediately.

For each $t \in \mathscr{T}$, by accessing all the challenges $\mathbb{C}_t$ stored in her email box, user can use the same approach as CSP/TPA to regenerate each $\mathscr{S}_t = \{(i_{j,t}, v_{j,t})\}_{1\leq j\leq l_{\mathbb{C}}}$ by herself. Then user computes the following parameter $\zeta_{\mathscr{T}}$ in terms of all hash values $h_{i_{j,t}}$ within the accepted $\mathscr{L}\mathscr{N}_{\mathscr{T}}$:

$$\zeta_{\mathscr{T}} \longleftarrow \prod_{t\in\mathscr{T}} \prod_{(i_{j,t}, v_{j,t})\in\mathscr{S}_t} u^{v_{j,t}\cdot h_{i_{j,t}}} \bmod N. \tag{13}$$

By comparing $\zeta_{\mathscr{T}}$ with $\eta_{\mathscr{T}}$ given by TPA within the proof $\beta_{\mathscr{T}}$, if $\zeta_{\mathscr{T}} \neq \eta_{\mathscr{T}}$, there is no doubt that TPA is indolent and thus user outputs *NO*. If $\zeta_{\mathscr{T}} = \eta_{\mathscr{T}}$, with her secret element $x$, user further checks whether

$$(u_{\mathscr{T}})^x \overset{?}{=} \omega_{\mathscr{T}} \bmod N. \tag{14}$$

If this check fails, user outputs *NO*, confirming that CSP tries to misconduct by submitting the incompatible proof to TPA. Otherwise, based on her secret key $SK_{Tag} = (N, d)$, user finally checks whether

$$(\omega_{\mathscr{T}})^d \cdot \tau_{\mathscr{T}} \overset{?}{=} \left(\zeta_{\mathscr{T}} \cdot \omega^{\mathscr{B}_{\mathscr{T}}}\right)^d \bmod N. \tag{15}$$

If the last check fails, user will learn that (i) TPA is feckless for his past auditing work and (ii) the outsourced data has been damaged at cloud side. Note that if there is something wrong with the challenged file blocks stored in CSP, the corresponding TPA's auditing is bound to fail and therefore TPA should report to user immediately according to the *AuditCSP* protocol. So the execution of *AuditTPA* protocol launched by user means that TPA implicitly makes a commitment to user about his conscientiousness for all the past auditing against CSP. In this context, once user's last check fails, TPA is malicious without any doubt. So user can take actions such as claiming compensations from TPA and CSP in terms of their signed contract.

In practice, as long as user's check upon (15) passes, user can delete all past challenges $\mathbb{C}_t$ from her email box. Meanwhile, the authorization of allowing TPA to delete all corresponding logs $\Theta_t$, signed by user herself, is generated and sent to TPA. In this case, both user and TPA just only need a constant storage space for storing the pregenerated challenges and the corresponding logs, respectively. And by this way the *AuditTPA* protocol will gain a significant performance improvement since the total number of point-in-time within the set $\mathscr{T}$ can be controlled under a reasonable upper limit.

*3.2.5. Design of IdentifyMalice.* Obviously, the process of the *AuditCSP* protocol has produced enough proofs to identify the malicious CSP who tries to conceal the fact of outsourced data corruption, and the process of the *AuditTPA* protocol also produces the undeniable proofs to identify the malicious TPA who tries to be lazy or does not perform the required auditing tasks correctly. Now, we show that how our scheme can defend against the malicious user, who wants to put the honest TPA in the wrong by purposely denying TPA's correct auditing work.

As shown in the *AuditCSP* protocol, firstly, the validity of TPA's auditing parameter $\eta_t$ is undeniable for the malicious user. Since all the elements used for computing $\eta_t$ are incontestable from the user's perspective, such as all the $(i_{j,t}, v_{j,t})$ derived from the challenge $\mathbb{C}_t$ signed by user herself, all the hash values $\{h_{i_{j,t}}\}$ can be authenticated based on the public MHT *root*, and the element $u$ is public. Secondly, the required auditing work for TPA is to use $\eta_t$ and CSP's proof $\mathscr{P}_t$ to check (11), where the authenticity of $\mathscr{P}_t$ can be verified by verifying CSP's signature, and the other two elements $\omega$ and $e$ within (11) are also public. Therefore, in case of dispute or litigation, honest TPA can reveal his own MHT $\Psi_{TPA}$ and the logs $\Theta_t = (\mathbb{C}_t, \mathscr{P}_t, \eta_t)$ (or the corresponding authorization of allowing TPA to delete the past logs, signed by user, as described in the *AuditTPA* protocol). In this case, provided that TPA always performs his auditing work correctly, then the malicious user has no chance to frame TPA up, since after authenticating $\Psi_{TPA}$ by the public *root*, anyone can use the open logs $\Theta_t$ to reconstruct $\eta_t$ and check (11) again to prove that honest TPA has actually fulfilled his auditing responsibility.

*3.3. Security Analysis.* According to the definitions in Section 2.4, we analyze the security of the proposed scheme in this section. The correctness of (11) within the *AuditCSP* protocol can be elaborated as follows:

$$\begin{aligned}
{\tau_t}^{e} &= \left(\prod_{(i_{j,t},v_{j,t})\in\mathcal{S}_t} {\tau_{i_{j,t}}}^{v_{j,t}}\right)^{e} \bmod N \\
&= \left(\prod_{(i_{j,t},v_{j,t})\in\mathcal{S}_t} u^{v_{j,t}\cdot h_{i_{j,t}}}\cdot \omega^{v_{j,t}\cdot b_{i_{j,t}}}\right)^{ed} \bmod N \\
&= \eta_t \cdot \prod_{(i_{j,t},v_{j,t})\in\mathcal{S}_t} \omega^{v_{j,t}\cdot b_{i_{j,t}}} \bmod N \\
&= \eta_t \cdot \omega^{\mathscr{B}_t-\xi_t} \bmod N = \frac{\eta_t\cdot\omega^{\mathscr{B}_t}}{\omega_t} \bmod N.
\end{aligned} \tag{16}$$

Furthermore, the correctness of (15) within the *AuditTPA* protocol is elaborated as follows:

$$\begin{aligned}
&(\omega_{\mathscr{T}})^{d}\cdot\tau_{\mathscr{T}} \\
&= \prod_{t\in\mathscr{T}}\left(\omega^{\xi_t}\cdot\prod_{(i_{j,t},v_{j,t})\in\mathcal{S}_t} u^{v_{j,t}\cdot h_{i_{j,t}}}\cdot\omega^{v_{j,t}\cdot b_{i_{j,t}}}\right)^{d} \bmod N \\
&= (\zeta_{\mathscr{T}})^{d}\cdot\prod_{t\in\mathscr{T}}\left(\omega^{\xi_t}\cdot\prod_{(i_{j,t},v_{j,t})\in\mathcal{S}_t} \omega^{v_{j,t}\cdot b_{i_{j,t}}}\right)^{d} \bmod N \\
&= (\zeta_{\mathscr{T}})^{d}\cdot\left(\prod_{t\in\mathscr{T}}\omega^{\mathscr{B}_t}\right)^{d} \bmod N \\
&= (\zeta_{\mathscr{T}})^{d}\cdot\left(\omega^{\mathscr{B}_{\mathscr{T}}}\right)^{d} \bmod N = \left(\zeta_{\mathscr{T}}\cdot\omega^{\mathscr{B}_{\mathscr{T}}}\right)^{d} \bmod N.
\end{aligned} \tag{17}$$

*ε-Sound*. As shown in Section 2.4, our scheme is naturally sound for honest TPA because of the implicit protection. Now, we prove the soundness for the other two cases where either user or CSP is honest. The soundness of our scheme for honest user is based on the Knowledge of Exponent Assumption (KEA), which was introduced by Damgård [27], formalized by Bellare and Palacio [28], and proved to be secure in the generic group model by Abe and Fehr [29]. Formally, KEA is described with an *extractor* as follows.

*KEA*. For any adversary $\mathscr{A}$ that takes input $N$, $u$, $u^x$ and returns $(P, Q)$ with $P = Q^x$, there exists an "extractor" $\mathbf{EA}(\cdot)$, given the same inputs as $\mathscr{A}$ returns $\widetilde{\mathscr{B}}$ such that $Q = u^{\widetilde{\mathscr{B}}}$.

*Case 1*. Honest user interacts with adversary $\mathscr{A}$ without aborting the protocol run. In this case, note that $u$ and $\omega = u^x$ are the public parameters taken as input by adversary $\mathscr{A}$, who cannot deduce any information about user's secret element $x$ since the discrete logarithm problem is hard. We define $\widetilde{\mathscr{B}}$ as $\widetilde{\mathscr{B}} := \sum_{t\in\mathscr{T}}(\sum_{j=1}^{l_{\mathbb{C}}} v_{j,t}\cdot b_{i_{j,t}})$. As described in the *AuditTPA* protocol, user will receive the returned $\mathscr{B}_{\mathscr{T}}, \omega_{\mathscr{T}}, u_{\mathscr{T}}$ from adversary $\mathscr{A}$. So the following two elements $P$ and $Q$ satisfying that $P = Q^x$ are the implicit output of adversary $\mathscr{A}$:

$$\begin{aligned}
P &= \frac{\omega^{\mathscr{B}_{\mathscr{T}}}}{\omega_{\mathscr{T}}} = \omega^{\widetilde{\mathscr{B}}} = \left(u^{\widetilde{\mathscr{B}}}\right)^{x}; \\
Q &= \frac{u^{\mathscr{B}_{\mathscr{T}}}}{u_{\mathscr{T}}} = u^{\widetilde{\mathscr{B}}}.
\end{aligned} \tag{18}$$

Based on the *KEA*, the *extractor* $\mathbf{EA}(\cdot)$ is able to extract $\widetilde{\mathscr{B}}$. Assume that the time set $\mathscr{T}$ has $k$ elements; then $\widetilde{\mathscr{B}} = \sum_{t\in\mathscr{T}}(\sum_{j=1}^{l_{\mathbb{C}}} v_{j,t}\cdot b_{i_{j,t}})$ is a linear equation built upon $k \times l_{\mathbb{C}}$ blocks set $\mathscr{M} := \{b_{i_{j,t}}\}_{1\le j\le l_{\mathbb{C}}, t\in\mathscr{T}}$, since user can control the challenge $\mathbb{C}_t$, which means that $\{(i_{j,t}, v_{j,t})\}_{1\le j\le l_{\mathbb{C}}, t\in\mathscr{T}}$ can be selectively determined by user. By depending on the honest user to generate independent coefficients $\{v_{j,t}\}_{1\le j\le l_{\mathbb{C}}, t\in\mathscr{T}}$ for the repetitive executions of *AuditTPA* protocol upon the same file blocks set $\mathscr{M}$, *extractor* $\mathbf{EA}(\cdot)$ can obtain a system of $k\times l_{\mathbb{C}}$ independent linear equations for $\mathscr{M}$. At last, *extractor* $\mathbf{EA}(\cdot)$ can recover each file block of $\mathscr{M}$ just by solving the system of these linear equations. In terms of Definition 2 in Section 2.4, our scheme is *ε-sound*.

*Case 2*. Honest CSP interacts with adversary $\mathscr{A}$. At this point, the same security argument as in [23] can be adopted here. Throughout our scheme, as long as user's outsourced data $\widetilde{\mathbf{F}} = \{\mathbf{F}, leaves, \tau\}$ can pass CSP's verification during the execution of the *Preproccess* protocol, then honest CSP must correctly possess $\widetilde{\mathbf{F}}$ and follow the subsequent protocols. So it is clear that the *extractor* $\mathbf{EA}(\cdot)$ can always rely on the honest CSP to extract the file $\mathbf{F}$ from $\widetilde{\mathbf{F}}$ for defending against any adversary $\mathscr{A}$. This completes the proof for soundness.

*Privacy-Preserving against Curious TPA*. Throughout our scheme, all the running results obtained by TPA are MHT $\Psi$ and CSP's proof $\mathscr{P}_t = (\mathscr{B}_t, \tau_t, \omega_t, u_t)$, which are derived from the executions of *Preproccess* and *AuditCSP* protocols, respectively, with respect to MHT $\Psi$, where every leaf node is hash value such as $h_i \leftarrow \mathbf{H}(b_i)$. So based on the preimage resistance attribute of the cryptographic hash function $\mathbf{H}(\cdot)$, none of privacy about the outsourced data block $b_i$ can be deduced from the leaf nodes of MHT $\Psi$.

Now, we argue that TPA cannot deduce user's data information from the proof $\mathscr{P}_t$ output by CSP. Due to the intractability of discrete logarithm, for each $t \in \mathscr{T}$, all the secret random interfering elements $\xi_t$, generated and stored individually by CSP for the corresponding challenges $\mathbb{C}_t$, cannot be derived by TPA from $\omega_t = \omega^{\xi_t}$ and $u_t = u^{\xi_t}$ even though $\omega$ and $u$ are the public parameters. As a consequence, for any specified blocks set $\{b_{i_{j,t}}\}_{1\le j\le l_{\mathbb{C}}}$, although TPA, via a number of challenges, might gather a system of linear equations such as $\mathscr{B}_t = \xi_t + \sum_{j=1}^{l_{\mathbb{C}}} v_{j,t}\cdot b_{i_{j,t}}$ with the different values of $\mathscr{B}_t$ and the independent coefficient $\{v_{j,t}\}_{1\le j\le l_{\mathbb{C}}}$, it is impossible for TPA to solve the system and deduce any data block $b_{i_{j,t}}$ since $\xi_t$ is unknown and will randomly change within each linear equation of the system.

With regard to $\tau_{\ell} \in \mathscr{P}_{\ell}$, which can be expressed as

$$\tau_{\ell} = \left(u^{\sum_{j=1}^{l_{\mathbb{C}}} v_{j,\ell} \cdot h_{i_{j,\ell}}}\right)^{d} \cdot \left(\omega^{\sum_{j=1}^{l_{\mathbb{C}}} v_{j,\ell} \cdot b_{i_{j,\ell}}}\right)^{d}, \qquad (19)$$

obviously, to deduce the privacy of data block $b_{i_{j,\ell}}$, TPA has to compute the exponentiation value $\sum_{j=1}^{l_{\mathbb{C}}} v_{j,\ell} \cdot b_{i_{j,\ell}}$ from $\tau_{\ell}$, which is the same as the intractability of discrete logarithm, let alone $d$ which is the secret key stored by user and is unknown for TPA. Summing up, no matter how many running results collected by TPA, our scheme can always protect user's data privacy from the curious TPA.

## 4. Dynamic Updates

Different from the outsourced auditing scheme of [23], our scheme constructs for each block $b_i$ the corresponding homomorphic tag $\tau_i$ without involving the block location $i$. In this way, the dynamic operations such as modification, insertion, and deletion can be realized by merely updating the targeted file block without affecting any others. For ease of description, we define the function of $CalRoot(node, LAAI_{node})$ that is to calculate the hash value of MHT root by applying the *node* and its corresponding $LAAI_{node}$, the example of which is described as shown in Section 3.1. As for the insertion operation, by default the new block $b_i^*$ will be inserted behind the appointed block location $i$.

Now we can present Algorithm 1 for modifying or inserting a new block $b_i^*$, and Algorithm 2 for deleting an appointed block at cloud server side. The main idea of these two algorithms is to let user control the update of MHT root throughout the process of data dynamics, since both CSP and TPA cannot output the expected root hash value unless they have correctly performed the dynamic operations as required by user. Both of these two algorithms are launched by user, who first transmits to CSP the necessary update command including the appointed location $i$. After receiving user's command, CSP must respond with the corresponding $(h_i, LAAI_{h_i})$. Note that user holds the original MHT *root*, which enables user to verify the authenticity of $(h_i, LAAI_{h_i})$ from CSP by checking if the *root* hash value is equal to the output of running $CalRoot(h_i, LAAI_{h_i})$. Once this user's verification passes, by calling $CalRoot(\cdot)$ again based on the $LAAI_{h_i}$ and above update command, user can compute by herself what the new $root^*$ would be after performing her update command upon the MHT stored at CSP side. In this case, to obtain user's authorization for actually executing the dynamic operations upon user's outsourced file $\widetilde{\mathbf{F}}$, CSP has to update its MHT in terms of user's command and output to user the same new $root^*$. Similarly, user will send the update command to TPA, and the same situation of updating MHT applies to the TPA side. Obviously, both CSP and TPA have to be well-behaving all the time; otherwise, their misconducts will be detected when the *AuditCSP* and *AuditTPA* protocols of our proposed outsourced auditing scheme are launched. Finally, user will update her local MHT *root* with the new $root^*$, meaning that the dynamic operations are executed successfully.

## 5. Experimental Evaluation

In this section, we simulate the computations of our proposed User Focus scheme and the Fortress scheme of [23] on the Inspur NF5270M4 servers with Intel Xeon CPU E5-2620 at 2.10 GHz, 16 GB RAM, and 7200 RPM 1 TB Serial ATA drive with a 32 MB buffer. Our experiments are implemented by using python language, and all the cryptographic functions are derived from the python cryptography toolkit [30]. We employ SHA1 to produce the 160 bit hash value, and the size of RSA module $N$ is 1024 bit for security. As for the Fortress scheme, we also utilize the tools of bitcoin block explorer [31] to access bitcoin resource for obtaining the random challenges, and we typically set the sector size to be 1 KB (e.g., each 64 KB file block consists of 64 sectors in Fortress).

Note that the typical block size for cloud storage is 64 KB–256 KB, as shown in [32]. Since outsourced auditing scheme runs above the cloud storage, the reasonable lower limit of block size should be no less than 64 KB. In our evaluation, user's outsourced file is chosen to 1 GB. We do not measure the time of uploading the outsourced file from user to CSP, since this overhead is common to the two investigated schemes. Our statistical results are an average of 20 rounds.

Firstly, user's file must be preprocessed before outsourcing. Figure 3 shows the required total time for the corresponding preprocessing phases of the two schemes. Furthermore, we also evaluate the computing time consumed by user for our scheme and Fortress, respectively. It turned out that for both schemes the computational overhead incurred at user side accounts for most of thetotal time when preprocessing the outsourced data, and the preprocessing performance of our scheme is orders of magnitude faster than of Fortress. As shown in [23], TPA needs to download the whole user's file $\mathbf{F}$ from cloud and convince the user that he correctly preprocessed $\mathbf{F}$. In this case Fortress requires user to carry out a time-consuming zero-knowledge-proof (ZKP) with TPA, resulting in the heavy computational overhead for user. Compared to Fortress, our User Focus scheme can effectively avoid such ZKP operation since TPA is not involved in preprocessing $\mathbf{F}$ and thus gain the performance enhancement.

Secondly, in Figure 4, with respect to the different block sizes, we measure the latency incurred by performing TPA's auditing against CSPonce in our scheme and in Fortress, respectively. In this experiment, we employ the same parameters as in [23]. During each outsourced auditing executed by TPA, the number of challenged blocks, denoted by $l_{\mathbb{C}}$, is set to 10% of the whole file blocks for both schemes. In addition, since Fortress involves the condition of parallel user-challenge, we also set the number of user-challenged blocks to be 10% of the TPA-challenged blocks as in [23] (i.e., 1% of the whole file blocks). The results show that during TPA's auditing phase the performance of our scheme is slightly better than Fortress at 64 KB block level, and the latency of our scheme declines faster than that of Fortress as the block size increases.

Finally, we focus on evaluating the time incurred on the user when she audits a batch of TPA's logs all at once. For a fair comparison, in this experiment we set the number of

```
Algorithm: $root^* = Modify\_or\_Insert\_Block\ (i, b_i^*)$.
Input: the appointed block location $i$, and the new block $b_i^*$.
Output: the new $root^*$ of the updated MHT.
User: compute $h_i^* \leftarrow \mathbf{H}(b_i^*)$; $\tau_i^* \leftarrow (u^{h_i^*} \cdot \omega^{b_i^*})^d \bmod N$;
    transmit $(mark, i, b_i^*, h_i^*, \tau_i^*)$ to CSP;
CSP: compute $\widetilde{h_i^*} \leftarrow \mathbf{H}(b_i^*)$
    if $\widetilde{h_i^*} = h_i^*$ then
        if $mark$ = modify then
            $S \longleftarrow h_i^*$;
        else {$mark = insert$}
            $S \longleftarrow \mathbf{H}(h_i \parallel h_i^*)$;
        end if
        compute $\widetilde{root}^* \longleftarrow CalRoot(S, LAAI_{h_i})$;
        send $\{(h_i, LAAI_{h_i}), \widetilde{root}^*\}$ to user;
    end if
User: compute $root^T \leftarrow CalRoot(h_i, LAAI_{h_i})$;
    if $root^T = root$ then
        compute $root^* \leftarrow CalRoot(S, LAAI_{h_i})$;
        if $root^* = \widetilde{root}^*$ then
            authorize CSP to execute update operation;
        end if
    end if
CSP: if $mark$ = modify then {note that $root^* = \widetilde{root}^*$}
            $b_i, h_i, \tau_i$ are replaced by $b_i^*, h_i^*, \tau_i^*$ in the outsourced file $\widetilde{\mathbf{F}}$;
            update the MHT $\Psi$ stored in cloud by recalculating all the nodes on the path from
             the $i$th leaf node $h_i^*$ to the $root^*$;
    else {$mark$ = insert, note that $root^* = \widetilde{root}^*$}
            insert $b_i^*, \tau_i^*$ into $\widetilde{\mathbf{F}}$ after $b_i, \tau_i$, respectively;
            update the MHT $\Psi$ by replacing with $\mathbf{H}(h_i \parallel h_i^*)$ the $i$th primordial leaf node that is transformed into
             a parent node having the left-child leaf $h_i$ and the right-child leaf $h_i^*$, and then recalculating all the
             nodes on the path from $\mathbf{H}(h_i \parallel h_i^*)$ to the $root^*$;
    end if
User: transmit $(mark, i, h_i^*)$ to TPA;
TPA: according to the $mark$, update the MHT $\Psi$ stored at TPA side by the same way as CSP;
        send to user the root of the updated MHT $\Psi$, denoted by $\overline{\overline{root}}^*$;
User: if $root^* = \overline{\overline{root}}^*$ then
            update the $root$, stored in her local, with $root^*$;
    end if
    return $root^*$;
```

Algorithm 1: Algorithm for modifying or inserting a block.

user-challenged blocks in Fortress to be the same as our User Focus scheme (i.e., each log refers to 10% of the whole blocks for user). As shown in Figure 5, although the latency increases linearly in two schemes, the performance of our User Focus scheme is almost 5 times faster than that of Fortress. Recall that Fortress relies on bitcoin hash values to determine the challenges, so for reproducing each past challenge, user has to repeatedly interact with bitcoin random resource by sending the HTTP requests and receiving the responses to obtain all past bitcoin hash values, which incurs the noticeable delay as the number of accumulated logs increases since a TPA's log is related to a past challenge. In our scheme, based on the property of User Focus, user can directly retrieve past challenges from her email box, which saves considerable time for reconstructing challenges when compared to Fortress. Note that each challenge of our scheme is only 88 bytes (8 bytes for the point-in-time $t$, 40 bytes for the two keys $PRF_t$ and $PRP_t$, and 40 bytes for user's signature), and user can delete all past accumulated challenges once her auditing against TPA is passed (as shown in Section 3.2.4); thus the storage and I/O costs of past challenges are extremely low for user in practice.

Summing up, for the phases of preprocessingand user's auditing against TPA that involve the user, the experimental results present that our scheme greatly improves the performance of user side, so our scheme will provide a

```
Algorithm: $root^* = Delete\_Block\ (i)$.
Input: the location $i$ of the appointed block that will be deleted.
Output: the new $root^*$ of the updated MHT.
User: transmit (delete, $i$) to CSP;
CSP: based on $i$, read $h_i$ and $LAAI_{h_i}$ from its MHT $\Psi$;
    send $(h_i, LAAI_{h_i})$ to user;
User: compute $root^T \leftarrow CalRoot(h_i, LAAI_{h_i})$;
    if $root^T = root$ then
        authorize CSP to execute delete operation;
    end if
CSP: if the sibling node of $h_i$, denoted by $\mathfrak{S}$, is a leaf node then
        replace with $\mathfrak{S}$ the parent node of $h_i$ which is transformed into a leaf node, and then delete $h_i$ from MHT $\Psi$;
    else {$\mathfrak{S}$ is a parent node}
        {let $\Psi_{\mathfrak{S}}$ denote the subtree with the root is $\mathfrak{S}$}
        replace with $\Psi_{\mathfrak{S}}$ the parent node of $h_i$, and delete $h_i$ from MHT $\Psi$;
    end if
    update the MHT $\Psi$ stored in cloud by recalculating all the nodes on the path from
        $\mathfrak{S}$ to the new root of MHT $\Psi$, denoted by $\widetilde{root}^*$;
    delete $b_i$, $\tau_i$ from the outsourced file $\widetilde{\mathbf{F}}$, and send $\widetilde{root}^*$ to user;
User: set $LAAI_{\mathfrak{S}} \leftarrow LAAI_{h_i} - \{\mathfrak{S}\}$; {notice $\mathfrak{S} \in LAAI_{h_i}$}
    compute $root^* \leftarrow CalRoot(\mathfrak{S}, LAAI_{\mathfrak{S}})$;
    if $root^* = \widetilde{root}^*$ then
        transmit (delete, $i$) to TPA;
    end if
TPA: update the MHT $\Psi$ stored at TPA side by the same way as CSP;
    send to user the root of the updated MHT $\Psi$, denoted by $\overline{\overline{root}}^*$;
User: if $root^* = \overline{\overline{root}}^*$ then
        update the $root$, stored in her local, with $root^*$;
    end if
    return $root^*$;
```

Algorithm 2: Algorithm for deleting a block.

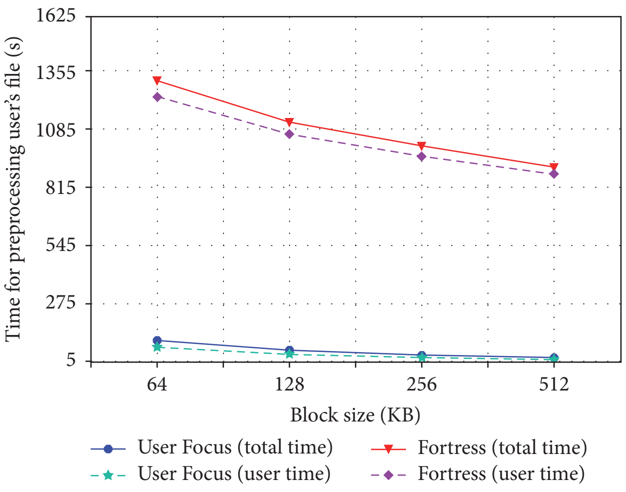


Figure 3: Comparison of time spent for preprocessing user's outsourced file.

more pleasant user experience when compared to Fortress. In addition, our User Focus scheme achieves the same performance as Fortress for TPA's auditing phases at the level of 64 KB block, or even better performance advantage at the bigger block level.

## 6. Related Work

With the popularization of storage outsourcing, the problem of remote data integrity auditing has attracted increasing attentions. All kinds of provable data possession (PDP) and

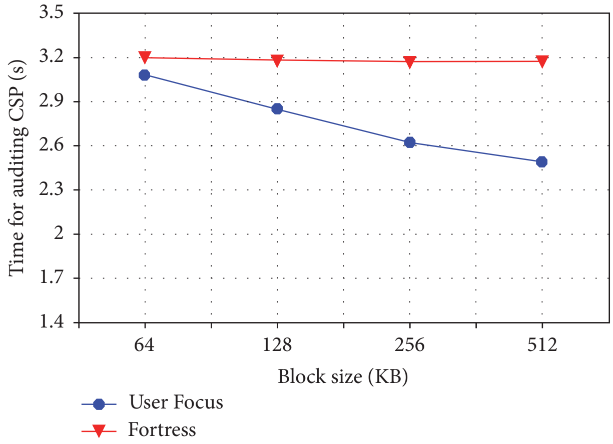


FIGURE 4: Latency for executing outsourced TPA's auditing once with respect to the block size.

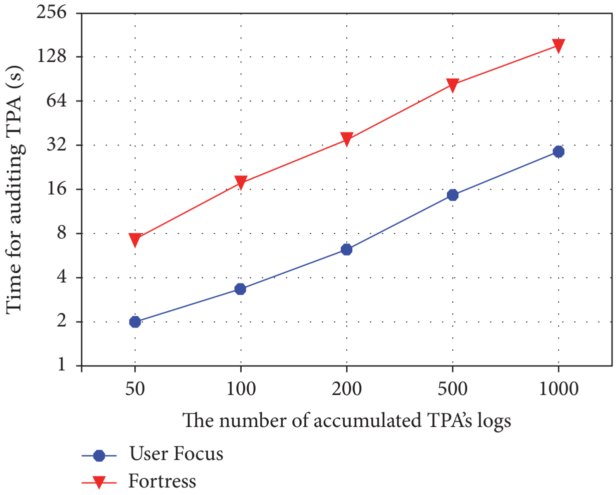


FIGURE 5: Time spent by user for auditing TPA once with respect to the number of accumulated TPA's logs. Here, block size is set to 64 KB that yields the most balanced performance over the two schemes when generating each log, as shown in Figure 4.

proof of retrievability (POR) schemes [7–23] are proposed to defend against the untrusted remote server. Ateniese et al. provided a series of PDP schemes for the storage security of outsourced data. In [7], they first described the formalized definition of PDP and proposed the original PDP schemes by utilizing the homomorphic verifiable tags that are constructed based on the public key cryptological technique. Simultaneously, to allow anyone, not just the data owner, to audit the untrusted server for data possession, the concept of public auditing is first introduced in [7]. Whereafter, in terms of the symmetric key cryptological technique, Ateniese et al. [10] proposed another provably secure PDP scheme for considering the problems of scalability and data dynamics that are not covered in the original PDP method. Besides, in [11], they also presented two more efficient PDP schemes that go one step further than the original schemes of [7]. To support the fully dynamic operations in the context of remote auditing, Erway et al. [12] extended the PDP model of [7] and presented the first dynamic PDP scheme by using the rank-based authenticated skip list. In addition, Wang et al. [16] and Zhu et al. [19] also proposed other efficient dynamic schemes for public auditing, which are based on the data structures of Merkle Hash Tree (MHT) and Index-Hash Table (IHT), respectively.

Juels and Kaliski Jr. [8] first proposed a formal POR model along with the corresponding security definitions. According to the model of [8], Shacham and Waters [9] constructed two POR schemes upon the static data storage but supporting the unlimited number of challenges. The first scheme is built from the pseudorandom functions to enable private auditing, and the second scheme with public auditing is built from the BLS signature [33]. Given that TPA might be curious during the process of public auditing, Wang et al. [13] integrated the random mask technique with the BLS-based public auditing scheme to prevent user's outsourced data from leaking to TPA, and the scheme of [13] has been further improved to support data dynamics in [15]. Moreover, under the environment of public auditing, many other schemes are also designed to meet the demands of different scenarios, such as fast data error localization [17], the auditing against shared data [18], batch auditing for multiple clouds [20], fine-grained data updates [21], and the lightweight computations for low performance end devices [22].

Recently a variety of cloud storage application schemes have been proposed, such as keyword-based data retrieval and image copy detection at cloud side. Xia et al. [34] constructed a special tree-based index structure and proposed a secure multikeyword ranked search scheme enabling dynamic updates upon outsourced encrypted data. Fu et al. [35] designed the parallel search algorithm and proposed another flexible searchable encryption scheme supporting both multikeyword ranked search and parallel search. In the setting of multikeyword fuzzy search, to solve the out-of-order problems during the ranking process, Fu et al. [36] also developed a new keyword transformation method and presented the corresponding efficient search scheme. In view of that traditional keyword-based search schemes that cannot completely match users' search intention, the innovative semantic search scheme based on the concept hierarchy is proposed in [37], making the personalized search more effective and context-aware. And the content-based search scheme of [38] has further solved the problems of semantic search by utilizing the conceptual graphs and the efficient measure of "sentence scoring." On the other hand, to protect the images stored in cloud, Xia et al. [39] proposed a privacy-preserving and copy-deterrence CBIR scheme using encryption and watermarking techniques, which can prevent the image user from illegally distributing the retrieved images. Li et al. [40] presented a solution to detect the copy-move forgery in an image, by first segmenting the targeted image into semantically independent patches prior to keypoint extraction and comparison. For detecting the image copies of a given original image generated by arbitrary rotation, Zhou et al. [41] proposed a novel copy detection method based on two global features extracted from rotation invariant partitions. In addition, Zhou et al. [42] designed a global context verification scheme to filter false matches for copy detection, which further addresses the problems of limited discriminability and quantization errors that exist in the bag-of-visual-words (BOW) model adopted by previous detection methods. However, since all application schemes mentioned above are designed upon the outsourced data of cloud storage, so it is obviously important for us to firstly focus on how to audit and confirm the integrity of remote outsourced data. But the existing public auditing schemes cannot protect against the malicious TPA. As shown in [23], malicious TPA is a potential security risk for outsourced data integrity and thus should not be ignored, which is the motivation of this paper.

## 7. Conclusion

Any public auditing/verification scheme can be transformed into a private scheme, just by making user perform the auditing work that should be delegated to TPA. Clearly, public auditing schemes might be more easily large-scale adopted by cloud users in practice, since user's heavy burden incurred by frequently auditing can be transferred to TPA. Nevertheless, how to protect user from a malicious TPA is a key problem that is never considered by various existing public auditing schemes. The first outsourced auditing scheme Fortress is proposed to defend against the malicious TPA, but Fortress enables TPA to download all outsourced data and thus only relies on data encryption to protect user's data privacy. A secure outsourced auditing scheme against malicious TPA should be designed to deprive TPA of the access rights over user's outsourced data in cloud, which is achieved in this paper. Although our proposed scheme is designed without relying on additional independent random source, it also achieves the security of protecting against any malicious entity. In addition, based on the MHT data structure, we extend the outsourced auditing scheme to support dynamic updates. With the analysis and evaluations, our scheme is provably secure and significantly efficient.

## Conflicts of Interest

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

## Acknowledgments

The authors would like to thank NSFC (Grant no. 61502044).

## References

[1] J. Xiao, X. Li, S. Chen, X. Zhao, and M. Xu, "An inside look into the complexity of box-office revenue prediction in China," *International Journal of Distributed Sensor Networks*, vol. 13, no. 1, 2017.

[2] M. Sookhak, H. Talebian, E. Ahmed, A. Gani, and M. K. Khan, "A review on remote data auditing in single cloud server: taxonomy and open issues," *Journal of Network and Computer Applications*, vol. 43, pp. 121–141, 2014.

[3] S. Fong, R. Wong, and A. Vasilakos, "Accelerated pso swarm search feature selection for data stream mining big data," *IEEE Transactions on Services Computing*, 2015.

[4] F. Tian, T. Lan, K.-M. Chao et al., "Mining suspicious tax evasion groups in big data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 10, pp. 2651–2664, 2016.

[5] X. Hu, S. Peng, and W.-L. Hwang, "EMD revisited: a new understanding of the envelope and resolving the mode-mixing problem in AM-FM signals," *IEEE Transactions on Signal Processing*, vol. 60, no. 3, pp. 1075–1086, 2012.

[6] X. Hu, S. Peng, and W.-L. Hwang, "Adaptive integral operators for signal separation," *IEEE Signal Processing Letters*, vol. 22, no. 9, pp. 1383–1387, 2015.

[7] 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 (CCS '07)*, pp. 598–609, Virginia, Va, USA, November 2007.

[8] A. Juels and B. S. Kaliski Jr., "Pors: proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 584–597, ACM, Alexandria, VA, USA, November 2007.

[9] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Advances in Cryptology—ASIACRYPT 2008: Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia,*

*December 2008*, vol. 5350 of *Lecture Notes in Computer Science*, pp. 90–107, Springer, Berlin, Germany, 2008.

[10] 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 (SecureComm '08)*, pp. 1–10, ACM, Istanbul, Turkey, September 2008.

[11] G. Ateniese, R. Burns, R. Curtmola et al., "Remote data checking using provable data possession," *ACM Transactions on Information and System Security*, vol. 14, no. 1, article 12, 2011.

[12] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of the Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*, pp. 213–222, ACM, Chicago, Ill, USA, November 2009.

[13] 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 2010*, usa, March 2010.

[14] H. Tian, Y. Chen, C.-C. Chang et al., "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Transactions on Services Computing*, vol. PP, no. 99, 2015.

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

[16] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011.

[17] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 220–232, 2012.

[18] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Transactions on Services Computing*, vol. 8, no. 1, pp. 92–106, 2015.

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

[20] 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, 2012.

[21] C. Liu, J. Chen, L. T. Yang et al., "Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2234–2244, 2014.

[22] J. Li, L. Zhang, J. K. Liu, H. Qian, and Z. Dong, "Privacy-Preserving Public Auditing Protocol for Low-Performance End Devices in Cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2572–2583, 2016.

[23] F. Armknecht, J.-M. Bohli, G. O. Karame, Z. Liu, and C. A. Reuter, "Outsourced proofs of retrievability," in *Proceedings of the 21st ACM Conference on Computer and Communications Security, CCS 2014*, pp. 831–843, usa, November 2014.

[24] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in *Proceedings of the 11th USENIX workshop on Hot topics in operating systems*, vol. 7, 2007.

[25] J. Liu, N. Asokan, and B. Pinkas, "Secure deduplication of encrypted data without additional independent servers," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS 2015*, pp. 874–885, usa, October 2015.

[26] R. C. Merkle, "A Certified Digital Signature," in *Proceedings of the Theory and Application of Cryptology, Lecture Notes in Computer Science*, vol. 435, pp. 218–238, 1989.

[27] I. Damgård, "Towards practical public key systems secure against chosen ciphertext attacks," in *Proceedings of the Annual International Cryptology Conference*, pp. 445–456, 1992.

[28] M. Bellare and A. Palacio, "The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols," in *Advances in Cryptology*, vol. 3152 of *Lecture Notes in Comput. Sci.*, pp. 273–289, Springer, Berlin, 2004.

[29] M. Abe and S. Fehr, "Perfect NIZK with adaptive soundness," in *Theory of cryptography*, pp. 118–136, 2007.

[30] D. C. Litzenberger, "Python Cryptography Toolkit," https://pypi.python.org/pypi/pycrypto, 2017.

[31] Bitcoin real-time status and tools, https://block-explorer.com/api-ref, 2017.

[32] E. Stefanov, M. V. Dijk, E. Shi et al., "Path ORAM: an extremely simple oblivious RAM protocol," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 299–310, 2013.

[33] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *International Conference on the Theory and Application of Cryptology and Information Security, Advances Cryptology*, vol. 2248, pp. 514–532, 2001.

[34] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[35] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Transactions on Communications*, vol. E98B, no. 1, pp. 190–200, 2015.

[36] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2706–2716, 2016.

[37] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2546–2559, 2015.

[38] Z. Fu, F. Huang, X. Sun, A. V. Vasilakos, and C. N. Yang, "Enabling semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Transactions on Services Computing*, 2016.

[39] Z. Xia, X. Wang, L. Zhang, Z. Qin, X. Sun, and K. Ren, "A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2594–2608, 2016.

[40] J. Li, X. Li, B. Yang, and X. Sun, "Segmentation-based image copy-move forgery detection scheme," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 507–518, 2015.

[41] Z. Zhou, C. Yang, B. Chen, X. Sun, Q. Liu, and Q. J. Wu, "Effective and efficient image copy detection with resistance

to arbitrary rotation," *IEICE Transactions on Information and Systems D*, vol. 99, no. 6, pp. 1531–1540, 2016.

[42] Z. Zhou, Y. Wang, Q. J. Wu, C. N. Yang, and X. Sun, "Effective and effcient global context verifcation for image copy detection," *IEEE Transactions on Information Forensicsand Security*, vol. 12, no. 1, pp. 48–63, 2016.

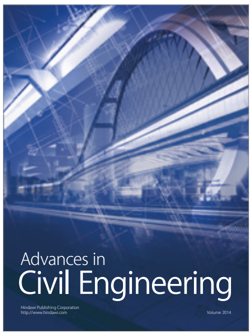
Journal of
Engineering

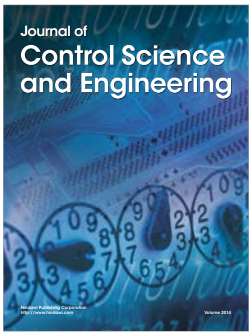
The Scientific World Journal

International Journal of
Rotating Machinery

Journal of
Sensors

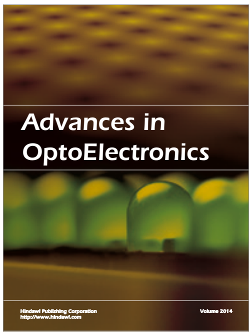
International Journal of
Distributed Sensor Networks


Advances in
Civil Engineering


Journal of
Control Science and Engineering

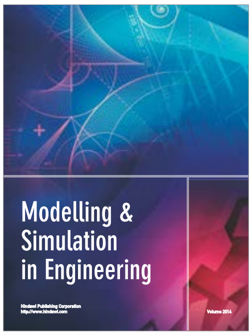
Journal of
Robotics

Journal of
Electrical and Computer Engineering

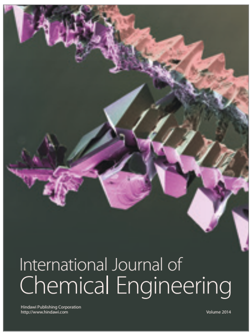

Advances in
OptoElectronics

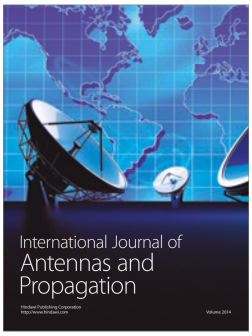
VLSI Design

International Journal of
Navigation and Observation

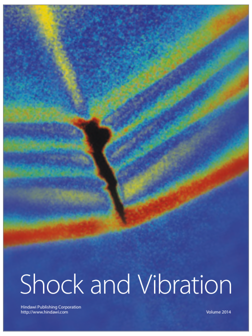

Modelling & Simulation in Engineering

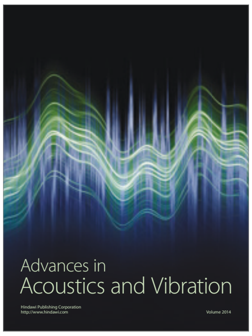
International Journal of
Aerospace Engineering


International Journal of
Chemical Engineering


International Journal of
Antennas and Propagation

Active and Passive Electronic Components


Shock and Vibration


Advances in
Acoustics and Vibration