International Journal of Innovative Research in Computer Science and Engineering (IJIRCSE)
www.ioirp.com ISSN: 2394-6364, Volume – 1, Issue – 6. May 2015

# ENABLING PUBLIC AUDITABILITY AND DATA DYNAMICS FOR STORAGE SECURITY IN CLOUD COMPUTING

[1]P.ILAMATHY
M.TECH-INFORMATION TECHNOLOGY
AMS ENGINEERING COLLEGE
NAMAKKAL, INDIA
ilamathyponnuvel@gmail.com

[2]V.ANITHA,M.E.
ASST PROFESSOR
M.TECH-INFORMATION TECHNOLOGY
AMS ENGINEERING COLLEGE
NAMAKKAL, INDIA

***Abstract*— Using cloud storage, users can remotely store their data and enjoy the on-demand high-quality applications and services from a shared pool of configurable computing resources, without the burden of local data storage and maintenance. However, the fact that users no longer have physical possession of the outsourced data makes the data integrity protection in cloud computing a formidable task, especially for users with constrained computing resources. Moreover, users should be able to just use the cloud storage as if it is local, without worrying about the need to verify its integrity. Thus, enabling public audit ability for cloud storage is of critical importance so that users can resort to a third-party auditor (TPA) to check the integrity of outsourced data and be worry free. To securely introduce an effective TPA, the auditing process should bring in no new vulnerabilities toward user data privacy, and introduce no additional online burden to user. In this paper, we propose a secure cloud storage system supporting privacy-preserving public auditing. We further extend our result to enable the TPA to perform audits for multiple users simultaneously and efficiently. Extensive security and performance analysis show the proposed schemes are provably secure and highly efficient.**

***Keywords*— Data storage, privacy preserving, public audit ability, cloud computing, delegation, batch verification, zero knowledge.**

## I. INTRODUCTION

Cloud computing has been envisioned as the next-generation information technology (IT) architecture for enterprises, due to its long list of unprecedented advantages in the IT history: on-demand self-service, ubiquitous network access, location independent resource pooling, rapid resource elasticity, usage-based pricing and transfer-ence of risk [2]. As a disruptive technology with profound implications, cloud computing is transforming the very nature of how businesses use information technology. One fundamental aspect of this paradigm shifting is that data are being centralized or outsourced to the cloud. From users' perspective, including both individuals and IT enterprises, storing data remotely to the cloud in a flexible on-demand manner brings appealing benefits: relief of the burden for storage management, universal data access with location independence, and avoidance of capital expenditure on hardware, software, and personnel maintenances, etc., [3].

While cloud computing makes these advantages more appealing than ever, it also brings new and challenging security threats toward users' outsourced data. Since cloud service providers (CSP) are separate administrative entities, data outsourcing is actually relinquishing user's ultimate control over the fate of their data. As a result, the correctness of the data in the cloud is being put at risk due to the following reasons. First of all, although the infrastructures under the cloud are much more powerful and reliable than personal computing devices, they are still facing the broad range of both internal and external threats for data integrity [4]. Examples of outages and security breaches of noteworthy cloud services appear from time to time [5], [6], [7]. Second, there do exist various motivations for CSP to behave unfaithfully toward the cloud users regarding their outsourced data status. For examples, CSP might reclaim storage for monetary reasons by discarding data that have not been or are rarely accessed, or even hide data loss incidents to maintain a reputation [8], [9], [10]. In short, although outsourcing data to the cloud is economic-ally attractive for long-term large-scale storage, it does not immediately offer any guarantee on data integrity and availability. This problem, if not properly addressed, may impede the success of cloud architecture.

As users no longer physically possess the storage of their data, traditional cryptographic primitives for the purpose of data security protection cannot be directly adopted [11]. In particular, simply downloading all the data for its integrity verification is not a practical solution due to the expensiveness in I/O and transmission cost across the network. Besides, it is often insufficient

to detect the data corruption only when accessing the data, as it does not give users correctness assurance for those unaccessed data and might be too late to recover the data loss or damage.

Considering the large size of the outsourced data and the user's constrained resource capability, the tasks of auditing the data correctness in a cloud environment can be formidable and expensive for the cloud users [12], [8]. Moreover, the overhead of using cloud storage should be minimized as much as possible, such that a user does not need to perform too many operations to use the data (in additional to retrieving the data). In particular, users may not want to go through the complexity in verifying the data integrity. Besides, there may be more than one user accesses the same cloud storage, say in an enterprise setting. For easier management, it is desirable that cloud only entertains verification request from a single designated party.

To fully ensure the data integrity and save the cloud users' computation resources as well as online burden, it is of critical importance to enable public auditing service for cloud data storage, so that users may resort to an independent third-party auditor (TPA) to audit the outsourced data when needed. The TPA, who has expertise and capabilities that users do not, can periodically check the integrity of all the data stored in the cloud on behalf of the users, which provides a much more easier and affordable way for the users to ensure their storage correctness in the cloud. Moreover, in addition to help users to evaluate the risk of their subscribed cloud data services, the audit result from TPA would also be beneficial for the cloud service providers to improve their cloud-based service platform, and even serve for indepen-dent arbitration purposes [10]. In a word, enabling public auditing services will play an important role for this nascent cloud economy to become fully established, where users will need ways to assess risk and gain trust in the cloud.

Recently, the notion of public auditability has been proposed in the context of ensuring remotely stored data integrity under different system and security models [9], [13], [11], [8]. Public auditability allows an external party, in addition to the user himself, to verify the correctness of remotely stored data. However, most of these schemes [9], [13], [8] do not consider the privacy protection of users' data against external auditors. Indeed, they may potentially reveal user's data to auditors, as will be discussed in Section 3.4. This severe drawback greatly affects the security of these protocols in cloud computing. From the perspective of protecting data privacy, the users, who own the data and rely on TPA just for the storage security of their data, do not want this auditing process introducing new vulnerabilities of unauthorized information leakage toward their data security. Moreover, there are legal regulations, such as the US Health Insurance Port-ability and Accountability Act (HIPAA) , further demanding the outsourced data not to be leaked to external parties [10]. Simply exploiting data encryption before outsourcing could be one way to mitigate this privacy concern of data auditing, but it could also be an overkill when employed in the case of unencrypted/public cloud data (e.g., outsourced libraries and scientific data sets), due to the unnecessary processing burden for cloud users. Besides, encryption does not completely solve the problem of protecting data privacy against third-party auditing but just reduces it to the complex key management.

## II. PROBLEM STATEMENT

### *A. The System and Threat Model*

We consider a cloud data storage service involving three different entities, as illustrated in Fig. 1: the cloud user, who has large amount of data files to be stored in the cloud; the cloud server, which is managed by the cloud service provider to provide data storage service and has significant storage

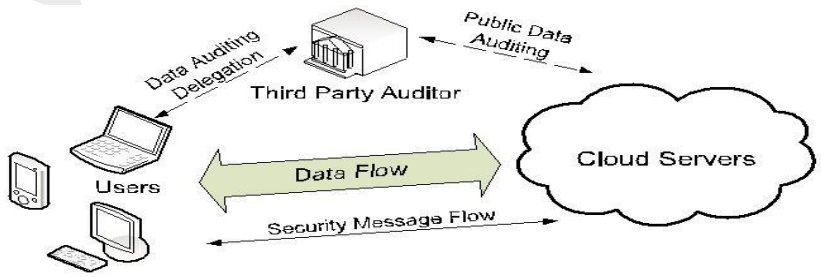



Fig. 1. The architecture of cloud data storage service.

space and computation resources (we will not differentiate CS and CSP hereafter); the third-party auditor, who has expertise and capabilities that cloud users do not have and is trusted to assess the cloud storage service reliability on behalf of the user

upon request. Users rely on the CS for cloud data storage and maintenance. They may also dynamically interact with the CS to access and update their stored data for various application purposes. As users no longer possess their data locally, it is of critical importance for users to ensure that their data are being correctly stored and maintained. To save the computation resource as well as the online burden potentially brought by the periodic storage correctness verification, cloud users may resort to TPA for ensuring the storage integrity of their outsourced data, while hoping to keep their data private from TPA.

We assume the data integrity threats toward users' data can come from both internal and external attacks at CS. These may include: software bugs, hardware failures, bugs in the network path, economically motivated hackers, malicious or accidental management errors, etc. Besides, CS can be self-interested. For their own benefits, such as to maintain reputation, CS might even decide to hide these data corruption incidents to users. Using third-party auditing service provides a cost-effective method for users to gain trust in cloud. We assume the TPA, who is in the business of auditing, is reliable and independent. However, it may harm the user if the TPA could learn the outsourced data after the audit.

To authorize the CS to respond to the audit delegated to TPA's, the user can issue a certificate on TPA's public key, and all audits from the TPA are authenticated against such a certificate. These authentication handshakes are omitted in the following presentation.

### *B. Design Goals*

To enable privacy-preserving public auditing for cloud data storage under the aforementioned model, our protocol design should achieve the following security and performance guarantees:

1. Public auditability: to allow TPA to verify the correctness of the cloud data on demand without retrieving a copy of the whole data or introducing additional online burden to the cloud users.
2. Storage correctness: to ensure that there exists no cheating cloud server that can pass the TPA's audit without indeed storing users' data intact.
3. Privacy preserving: to ensure that the TPA cannot derive users' data content from the information collected during the auditing process.
4. Batch auditing: to enable TPA with secure and efficient auditing capability to cope with multiple auditing delegations from possibly large number of different users simultaneously.
5. Lightweight: to allow TPA to perform auditing with minimum communication and computation overhead.

## III. THE PROPOSED SCHEMES

This section presents our public auditing scheme which provides a complete outsourcing solution of data—not only the data itself, but also its integrity checking. After introducing notations and brief preliminaries, we start from an overview of our public auditing system and discuss two straightforward schemes and their demerits. Then, we present our main scheme and show how to extent our main scheme to support batch auditing for the TPA upon delegations from multiple users. Finally, we discuss how to generalize our privacy-preserving public auditing scheme and its support of data dynamics.

### *A. Notation and Preliminaries*

- F —the data file to be outsourced, denoted as a sequence of n blocks $m_1, \ldots m_i, \ldots, m_n \in Z_p$ for some large prime p.
- $MAC_{(.)}(.)$—message authentication code (MAC) function, defined as: K*{0,1}*→{0,1}where K denotes the key space.
- H(.), h(.)—cryptographic hash functions.

We now introduce some necessary cryptographic back-ground for our proposed scheme.

Bilinear Map. Let $G_1$, $G_2$, and $G_T$ be multiplicative cyclic groups of prime order p. Let $g_1$ and $g_2$ be generators of $G_1$ and $G_2$, respectively. A bilinear map is a map e :$G_1$ * $G_2$ →$G_T$ such that for all u € $G_1$, v €$G_2$ and a; b € $Z_p$, e($u^a$, $v^b$) This bilinearity implies that for any $u_1$, $u_2$ € $G_1$, v € $G_2$, e($u_1$ . $u_{2)}$ .

*B. Definitions and Framework*

We follow a similar definition of previously proposed schemes in the context of remote data integrity checking and adapt the framework for our privacy-preserving public auditing system.

A public auditing scheme consists of four algorithms (KeyGen, SigGen, GenProof, VerifyProof). KeyGen is a key generation algorithm that is run by the user to setup the scheme. SigGen is used by the user to generate verification metadata, which may consist of digital signatures. GenProof is run by the cloud server to generate a proof of data storage correctness, while VerifyProof is run by the TPA to audit the proof .Running a public auditing system consists of two phases, Setup and Audit:

- **Setup:** The user initializes the public and secret parameters of the system by executing KeyGen, and pre-processes the data file F by using SigGen to generate the verification metadata. The user then stores the data file F and the verification metadata at the cloud server, and deletes its local copy. As part of preprocessing, the user may alter the data file F by expanding it or including additional metadata to be stored at server.

- **Audit:** The TPA issues an audit message or challenge to the cloud server to make sure that the cloud server has retained the data file F properly at the time of the audit. The cloud server will derive a response message by executing GenProof using F and its verification metadata as inputs. The TPA then verifies the response via VerifyProof.

Our framework assumes that the TPA is stateless, i.e., TPA does not need to maintain and update state between audits, which is a desirable property especially in the public auditing system . Note that it is easy to extend the framework above to capture a stateful auditing system, essentially by splitting the verification metadata into two parts which are stored by the TPA and the cloud server, respectively. Our design does not assume any additional property on the data file. If the user wants to have more error resilience, he can first redundantly encodes the data file and then uses our system with the data that has error-correcting codes integrated.[1]

*C. The Basic Schemes*

Before giving our main result, we study two classes of schemes as a warm-up. The first one is a MAC-based solution which suffers from undesirable systematic demerits— bounded usage and stateful verification, which may pose additional online burden to users, in a public auditing setting. This also shows that the auditing problem is still not easy to solve even if we have introduced a TPA. The second one is a system based on homomorphic linear authenticators, which covers many recent proof of storage systems. We will pinpoint the reason why all existing HLA-based systems are not privacy preserving. The analysis of these basic schemes leads to our main result, which overcomes all these drawbacks. Our main scheme to be presented is based on a specific HLA scheme.

MAC-based solution. There are two possible ways to make use of MAC to authenticate the data. A trivial way is just uploading the data blocks with their MACs to the server, and sends the corresponding secret key sk to the

1. We refer readers for the details on integration of error-correcting codes and remote data integrity checking.

TPA. Later, the TPA can randomly retrieve blocks with their MACs and check the correctness via sk. Apart from the high (linear in the sampled data size) communication and computation complexities, the TPA requires the knowledge of the data blocks for verification.

To circumvent the requirement of the data in TPA verification, one may restrict the verification to just consist of equality checking. The idea is as follows: Before data outsourcing, the cloud user chooses s random message authentication code keys fsk_ $g_{1\_\_s}$, precomputes s (deter-ministic) MACs, $fMAC_{sk\_}$ ðF Þ$g_{1__s}$ for the whole data file F , and publishes these verification

metadata (the keys and the MACs) to TPA. The TPA can reveal a secret key sk_ to the cloud server and ask for a fresh keyed MAC for comparison in each audit. This is privacy preserving as long as it is impossible to recover F in full given $MAC_{sk\_}$ ðF Þ and sk_ . However, it suffers from the following severe draw-backs: 1) the number of times a particular data file can be audited is limited by the number of secret keys that must be fixed a priori. Once all possible secret keys are exhausted, the user then has to retrieve data in full to recomputed and republish new MACs to TPA; 2) The TPA also has to maintain and update state between audits, i.e., keep track on the revealed MAC keys. Considering the potentially large number of audit delegations from multiple users, maintaining such states for TPA can be difficult and error prone; 3) it can only support static data, and cannot efficiently deal with dynamic data at all. However, supporting data dynamics is also of critical importance for cloud storage systems. For the reason of brevity and clarity, our main protocol will be presented based on static data. Section 3.6 will describe how to adapt our protocol for dynamic data.

TABLE 1
The Privacy-Preserving Public Auditing Protocol

| TPA | | Cloud Server |
|---|---|---|
| 1. Retrieve file tag $t$, verify its signature, and quit if fail; | | |
| 2. Generate a random challenge $chal = \{(i, \nu_i)\}_{i \in I}$; | $\xrightarrow[\text{challenge request } chal]{\{(i,\nu_i)\}_{i\in I}}$ | 3. Compute $\mu' = \sum_{i \in I} \nu_i m_i$, and $\sigma = \prod_{i \subset I} \sigma_i^{\nu_i}$; |
| | | 4. Randomly pick $r \leftarrow \mathbb{Z}_p$, and $R = e(u, v)^r$ and $\gamma = h(R)$; |
| | $\xleftarrow[\text{storage correctness proof}]{\{\mu,\sigma,R\}}$ | 5. Compute $\mu = r + \gamma\mu' \mod p$ ; |
| 6. Compute $\gamma = h(R)$, and then verify $\{\mu, \sigma, R\}$ via Equation 1. | | |

*D. Privacy-Preserving Public Auditing Scheme*

Overview. To achieve privacy-preserving public auditing, we propose to uniquely integrate the homomorphic linear authenticator with random masking technique. In our protocol, the linear combination of sampled blocks in the server's response is masked with randomness generated by the server. With random masking, the TPA no longer has all the necessary information to build up a correct group of linear equations and therefore cannot derive the user's data content, no matter how many linear combinations of the same set of file blocks can be collected. On the other hand, the correctness validation of the block-authenticator pairs can still be carried out in a new way which will be shown shortly, even with the presence of the randomness. Our design makes use of a public key-based HLA, to equip the auditing protocol with public auditability. Specifically, we use the HLA proposed in [13], which is based on the short signature scheme proposed by Boneh, Lynn, and Shacham (hereinafter referred as BLS signature) [19].

Scheme details. Let $G_1$, $G_2$, and $G_T$ be multiplicative cyclic groups of prime order p, and e : $G_1$ _ $G_2$ ! $G_T$ be a bilinear map as introduced in preliminaries. Let g be a generator of $G_2$. Hð_Þ is a secure map-to-point hash function: f0; 1g- ! $G_1$, which maps strings uniformly to $G_1$. Another hash function hð_Þ : $G_T$ ! $Z_p$ maps group element of $G_T$ uniformly to $Z_p$. Our scheme is as follows:

- **Setup Phase:** The cloud user runs KeyGen to generate the public and secret parameters. Specifically, the user chooses a random signing key pair.
- **Audit Phase:** The TPA first retrieves the file tag t. With respect to the mechanism we describe in the Setup phase, the TPA verifies the signature, and quits by emitting FALSE if the verification fails. Otherwise, the TPA recovers name.

Properties of our protocol. It is easy to see that our protocol achieves public auditability. There is no secret keying material or states for the TPA to keep or maintain between audits, and the auditing protocol does not pose any potential online burden on users. This approach ensures the privacy of user data content during the auditing process by employing a random masking r to hide _, a linear combination of the data blocks. Note that the value R in our protocol, which enables the privacy-preserving

guarantee, will not affect the validity of the equation, due to the circular relationship between R and _ in _ ¼ hðRÞ and the verification equation. Storage correctness thus follows from that of the underlying protocol [13]. The security of this protocol will be formally proven in Section 4. Besides, the HLA helps achieve the constant communication overhead for server's response during the audit: the size of f_; _; Rg is independent of the number of sampled blocks c.

Previous work showed that if the server is missing a fraction of the data, then the number of blocks that needs to be checked in order to detect server misbehavior with high probability is in the order of Oð1Þ. In particular, if t fraction of data is corrupted, then random sampling c blocks would reach the detection probability P ¼ 1 _ ð1 _ tÞ$^{c}$. Here, every block is chosen uniformly at random. When t ¼ 1% of the data F , the TPA only needs to audit for c ¼ 300 or 460 randomly chosen blocks of F to detect this misbehavior with probability larger than 95 and 99 percent, respectively. Given the huge volume of data outsourced in the cloud, checking a portion of the data file is more affordable and practical for both the TPA and the cloud server than checking all the data, as long as the sampling strategies provides high-probability assurance. In Section 4, we will present the experiment result based on these sampling strategies.

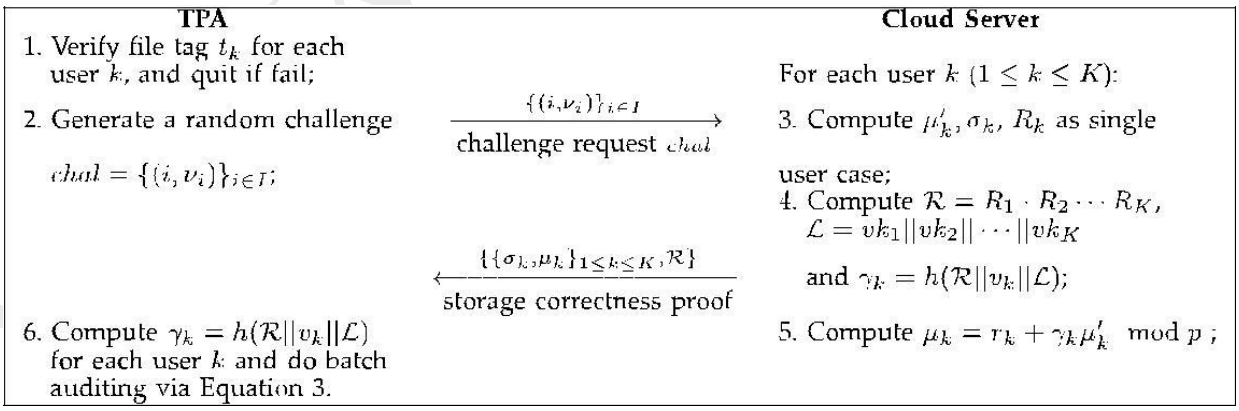
### *E. Support for Batch Auditing*

With the establishment of privacy-preserving public auditing, the TPA may concurrently handle multiple auditing upon different users' delegation. The individual auditing of these tasks for the TPA can be tedious and very inefficient. Given K auditing delegations on K distinct data files from K different users, it is more advantageous for the TPA to batch these multiple tasks together and audit at one time. Keeping this natural demand in mind, we slightly modify the protocol in a single user case, and achieves the aggregation of K verification equations (for K auditing tasks) into a single one, as shown in (3). As a result, a secure batch auditing protocol for simultaneous auditing of multiple tasks is obtained. The details are described as follows:

- **Setup phase:** Basically, the users just perform Setup independently. Suppose there are K users in the system, and each user k has a data file $F_k$ ¼ ðm$_{k;1}$; . . . ; m$_{k;n}$Þ to be outsourced to the cloud server, where k 2 f1; . . . ; Kg. For simplicity, we assume each file $F_k$ has the same number of n blocks. For a particular user k, denote his/her secret key as ðx$_k$; ssk$_k$Þ, and the corresponding public parameter as ðspk$_k$; v$_k$; g; u$_k$; eðu$_k$; v$_k$ÞÞ where v$_k$ ¼ g$^{xk}$ . Similar to the single user case, each user k has already randomly chosen a different (with overwhelming probability) name name$_k$ 2 $Z_p$ for his/her file $F_k$, and has correctly generated the corresponding file tag t$_k$ ¼ name$_k$kSSig$_{sskk}$ ðname$_k$Þ. Then, each user k runs SigGen and computes _$_{k;i}$ for block.

- **Audit phase:** TPA first retrieves and verifies file tag t$_k$ for each user k for later auditing. If the verification fails, TPA quits by emitting FALSE. Otherwise, TPA recovers

TABLE 2 The Batch Auditing Protocol




Efficiency improvement. As shown in (3), batch audit-ing not only allows TPA to perform the multiple auditing tasks simultaneously, but also greatly reduces the computation cost on the TPA side. This is because aggregating K verification equations into one helps reduce the number of relatively expensive pairing operations from 2K, as required in the individual auditing, to K þ 1, which saves a considerable amount of auditing time.

Identification of invalid responses. The verification equation (3) only holds when all the responses are valid, and fails with high probability when there is even one single invalid response in the batch auditing, as we will show in Section 4. In many situations, a response collection may contain invalid responses, especially f_kg1_k_K , caused by accidental data corruption, or possibly malicious activity by a cloud server. The ratio of invalid responses to the valid could be quite small, and yet a standard batch auditor will reject the entire collection. To further sort out these invalid responses in the batch auditing, we can utilize a recursive divide-and-conquer approach (binary search), as suggested by Ferrara et al. [20]. Specifically, if the batch auditing fails, we can simply divide the collection of responses into two halves, and repeat the auditing on halves via (3). TPA may now require the server to send back all the fRkg1_k_K , as in individual auditing. In Section 4.2.2, we show through carefully designed experiment that using this recursive binary search approach, even if up to 20 percent of responses are invalid, batch auditing still performs faster than individual verification.

### *F. Support for Data Dynamics*

In cloud computing, outsourced data might not only be accessed but also updated frequently by users for various application purposes]. Hence, supporting data dynamics for privacy-preserving public auditing is also of paramount importance. Now, we show how to build upon the existing work [8] and adapt our main scheme to support data dynamics, including block level operations of modification, deletion, and insertion.

In [8], data dynamics support is achieved by replacing the index information i with $m_i$ in the computation of block authenticators and using the classic data structure— Merkle hash tree (MHT) [24] for the underlying block sequence enforcement. As a result, the authenticator for each block is changed to _i ¼ ðHðmiÞ _ u mi Þx. We can adopt this technique in our design to achieve privacy-preserving public auditing with support of data dynamics. Specifically, in the Setup phase, the user has to generate and send the tree root $TR_{MHT}$ to TPA as additional metadata, where the leaf nodes of MHT are values of HðmiÞ. In the Audit phase, besides f_; _; Rg, the server's response should also include fHðmiÞgi2I and their corresponding auxiliary authentication information aux in the MHT. Upon receiving the response, TPA should first use $TR_{MHT}$ and aux to authenticate fHðmiÞgi2I computed by the server.

## IV. EVALUATION

### *A. Performance Analysis*

We now report some performance results of our experiments. We consider our auditing mechanism happens between a dedicated TPA and some cloud storage node, where user's data are outsourced to. In our experiment, the TPA/user side process is implemented on a workstation with an Intel Core 2 processor running at 1.86 GHz, 2,048 MB of RAM, and a 7,200 RPM Western Digital 250 GB Serial ATA drive. The cloud server side process is implemented on Amazon Elastic Computing Cloud (EC2) with a large instance type [27], which has 4 EC2 Compute Units, 7.5 GB memory, and 850 GB instance storage. The randomly generated test data is of 1 GB size. All algorithms are implemented using C language. Our code uses the Pairing-Based Cryptography (PBC) library version 0.4.21. The elliptic curve utilized in the experiment is an MNT curve, with base field size of 159 bits and the embedding degree 6. The security level is chosen to be 80 bit, which means j_j ¼ 80 and jpj ¼ 160. All experimental results represent the mean of 20 trials.

Because the cloud is a pay-per-use model, users have to pay both the storage cost and the bandwidth cost (for data transfer) when using the cloud storage auditing. Thus, when implementing our mechanism, we have to take into consideration both factors. In particular, we conduct the experiment with two different sets of storage/communications tradeoff parameter s as introduced in Section 3.4. When s ¼ 1, the mechanism incurs extra storage cost as large as

TABLE 3 Notation of Cryptographic Operations

| | |
|---|---|
| $Hash^t_{\mathbb{G}_1}$ | hash $t$ values into the group $\mathbb{G}_1$. |
| $Mult^t_{\mathbb{G}}$ | $t$ multiplications in group $\mathbb{G}$. |
| $Exp^t_{\mathbb{G}}(\ell)$ | $t$ exponentiations $g^{a_i}$, for $g \in \mathbb{G}$, $\lvert a_i\rvert = \ell$. |
| $m\text{-}MultExp^t_{\mathbb{G}}(\ell)$ | $t$ $m$-term exponentiations $\prod_{i=1}^{m} g^{a_i}$. |
| $Pair^t_{\mathbb{G}_1,\mathbb{G}_2}$ | $t$ pairings $e(u_i, g_i)$, where $u_i \in \mathbb{G}_1$, $g_i \in \mathbb{G}_2$. |
| $m\text{-}MultPair^t_{\mathbb{G}_1,\mathbb{G}_2}$ | $t$ $m$-term pairings $\prod_{i=1}^{m} e(u_i, g_i)$. |

the data itself, but only takes very small auditing bandwidth cost. Such a mechanism can be adopted when the auditing has to happen very frequently (e.g., checking the storage correctness every few minutes [21]), because the resulting data transfer charge could be dominant in the pay-per-use-model. On the other hand, we also choose a properly larger s ¼ 10, which reduces the extra storage cost to only 10 percent of the original data but increases the auditing bandwidth cost roughly 10 times larger than the choice of s ¼ 1. Such a case is relatively more desirable if the auditing does not need to happen frequently. In short, users can flexibly choose the storage/communication tradeoff parameter s for their different system application scenarios.

## V. CONCLUSION

In this paper, we propose a privacy-preserving public auditing system for data storage security in cloud comput-ing. We utilize the homomorphic linear authenticator and random masking to guarantee that the TPA would not learn any knowledge about the data content stored on the cloud server during the efficient auditing process, which not only eliminates the burden of cloud user from the tedious and possibly expensive auditing task, but also alleviates the users' fear of their outsourced data leakage. Considering TPA may concurrently handle multiple audit sessions from different users for their outsourced data files, we further extend our privacy-preserving public auditing protocol into a multiuser setting, where the TPA can perform multiple auditing tasks in a batch manner for better efficiency. Extensive analysis shows that our schemes are provably secure and highly efficient. Our preliminary experiment conducted on Amazon EC2 instance further demonstrates the fast performance of our design on both the cloud and the auditor side. We leave the full-fledged implementation of the mechanism on commercial public cloud as an important future extension, which is expected to robustly cope with very large scale data and thus encourage users to adopt cloud storage services more confidently.

## *References*

[1] Cloud Security Alliance, "Top Threats to Cloud Computing," [29] Y. Dodis, S.P. Vadhan, and D. Wichs, "Proofs Retrievability via http://www.cloudsecurityalliance.org, 2010.

[2] M. Arrington, "Gmail Disaster: Reports of Mass Email Deletions," http://www.techcrunch.com/2006/12/28/gmail-disasterreports-of-mass-email-deletions/, 2006.

[3] J. Kincaid, "MediaMax/TheLinkup Closes Its Doors," http:// www.techcrunch.com/2008/07/10/mediamaxthelinkup-closes-its-doors/, July 2008.

[4] Amazon.com, "Amazon s3 Availability Event: July 20, 2008," http://status.aws.amazon.com/s3-20080720.html, July 2008.

[5] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," IEEE Trans. Parallel and Distributed Systems, vol. 22, no. 5, pp. 847-859, May 2011.

[6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 598-609, 2007.

[7] M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-Preserving Audit and Extraction of Digital Contents," Cryptology ePrint Archive, Report 2008/186, 2008.

[8] A. Juels and J. Burton, S. Kaliski, "PORs: Proofs of Retrievability for Large Files," Proc. ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, Oct. 2007.

[9] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing," http://www.cloudsecurityalliance. org, 2009.

[10] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (Asia crypt), vol. 5350, pp. 90-107, Dec. 2008.

[11] C. Wang, K. Ren, W. Lou, and J. Li, "Towards Publicly Auditable Secure Cloud Data Storage Services," IEEE Network Magazine, vol. 24, no. 4, pp. 19-24, July/Aug. 2010.

[12] M.A. Shah, M. Baker, J.C. Mogul, and R. Swaminathan, "Auditing to Keep Online Storage Services Honest," Proc. 11th USENIX Workshop Hot Topics in Operating Systems (HotOS '07), pp. 1-6, 2007.

[13] 104th United States Congress, "Health Insurance Portability and Accountability Act of 1996 (HIPPA)," http://aspe.hhs.gov/ admnsimp/pl104191.htm, 1996.