ISSN:0975-9646

HLA Based Third Party Auditing For Secure Cloud Storage

Chandinee Saraswathy K. , Keerthi D. , Padma G.

Student , Department of Computer Science and Engineering
Velammal Institute of Technology, Panchetti

Abstract – **Using Cloud Storage, users can remotely store their data and enjoy high quality applications and services from a shared pool of resources. However, users, when they don't have the physical possession of the data, chances of raising data integrity issue are possible. Thus, verification of data integrity is the at most important for a user, who has outsourced data to the cloud. To make the integrity check, a public auditing must be made possible. For it, we resort to a Third Party Auditor (TPA). Also, the auditing process should not bring in further more burdens to the user. In this paper, we propose a secure cloud storage for which we integrate the technique of Homomorphic linear authenticator with random masking. Thereby, we can assure integrity of the user's outsourced data in the cloud.**

Keywords* – *Data integrity, Third Party Audting, Public Auditing, Data storage, Cloud computing, 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 transference 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 [14], [15].

Simply exploiting data encryption before outsourcing [15], [11] 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. Unauthorized data leakage still remains possible due to the potential exposure of decryption keys.

Therefore, how to enable a privacy-preserving third-party auditing protocol, independent to data encryption, is the problem we are going to tackle in this paper. As the individual auditing of these growing tasks can be tedious and cumbersome, a natural demand is then how to enable the TPA to efficiently perform multiple auditing tasks in a batch manner, i.e., simultaneously.

To address these problems, our work utilizes the technique of public key-based homomorphic linear authenticator (or HLA for short) [9], [13], [8], which enables TPA to perform the auditing without demanding the local copy of data and thus drastically reduces the communication and computation overhead as compared to the straightforward data auditing approaches. By integrating the HLA with random masking, our protocol guarantees that the TPA could not learn any knowledge about the data content stored in the cloud server (CS) during the efficient auditing process.

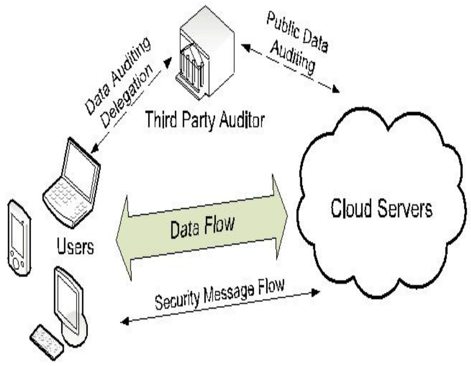
The aggregation and algebraic properties of the authenticator further benefit our design for the batch auditing. Specifically, we concentrate on the following aspects:

1. Our scheme enables an external auditor to audit user's cloud data without learning the data content.
2. Our scheme achieves batch auditing where multiple delegated auditing tasks from different users can be performed simultaneously by the TPA in a privacy-preserving manner.
3. We prove the security and justify the performance of our proposed schemes through concrete experiments and comparisons with the state of the art.

II. Problem Statement

1. *The System and Threat Model*

We consider a cloud data storage service involving three different entities, as illustrated in the Figure: 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 space and computation resources; 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. 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 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. 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.

Note that in our model, beyond users' reluctance to leak data to TPA, we also assume that cloud servers have no incentives to reveal their hosted data to external parties. Therefore, we assume that neither CS nor TPA has motivations to collude with each other during the auditing process.

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.

2. *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:

a) 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.

b) Storage correctness: To ensure that there exists no cheating cloud server that can pass the TPA's audit without indeed storing users' data intact.

c) Privacy preserving: To ensure that the TPA cannot derive users' data content from the information collected during the auditing process.

d) 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.

e) Lightweight: To allow TPA to perform auditing with minimum communication and computation overhead.

III. The Proposed Schemes

1. *Notation and Preliminaries*

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.

- F: the data file to be outsourced, denoted as a sequence of n blocks m1, m2, m3,..,mi,.., mn € Zp for some large prime p.
- MAC(.) (.) : message authentication code (MAC) function, defined as: K X {0, 1}* => {0, 1}l where K denotes Key space.
- H (.), h (.) : cryptographic hash functions

We now introduce some necessary cryptographic background for our proposed scheme.

Bilinear Map: Let G1, G2, and GT be multiplicative cyclic groups of prime order p. Let g1 and g2 be generators of G1 and G2, respectively. A bilinear map e is a map:

G1 X G2 -> GT such that for all u € G1G2 and

a; b €Zp, $e(u^a , v^b) = e(u,v)^{ab}$. This bilinearity implies that for any u1, u2 €G1, v€G2, e(u1.u2, v)= e(u1,v).e(u2; v). Of course, there exists an efficiently computable algorithm for computing e and the map should be nontrivial, i.e., e is non degenerate: e (g1; g2) ≠ 1.

2. *Definitions and Framework*

We follow a similar definition of previously proposed schemes in the context of remote data integrity checking [9], [11], [13] 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 signa-tures. 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 delete its local copy. As part of pre processing, 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 [13]. 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.

3. *The Basic Scheme*

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 much 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.

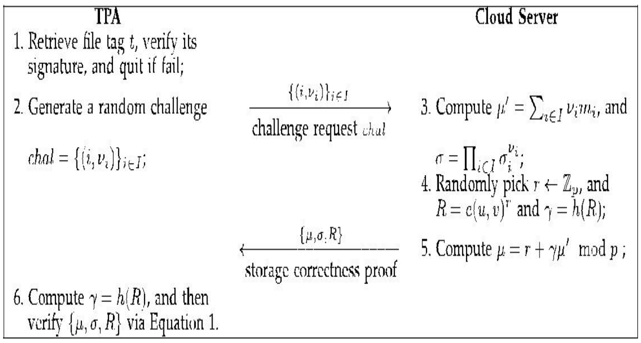
MAC-based solution: There are two possible ways to make use of MAC to authenticate the data. A trivial way is just uploading the data blocks with their MACs to the server, and sends the corresponding secret key sk to the TPA. Later, the TPA can randomly retrieve blocks with their MACs and check the correctness 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 $\{sk_r\}_{1<=r<=s}$, precomputes MACs for the whole file F and publishes these verification metadata (the keys and the MACs) to TPA. The TPA can reveal a secret key sk_r 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_r(F) and sk_r. However, it suffers from the following severe drawbacks:

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 prior. Once all possible secret keys are exhausted, the user then has to retrieve data in full to recompute 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.

HLA-based solution: To effectively support public auditability without having to retrieve the data blocks themselves, the HLA technique [9], [13], [8] can be used. HLAs, like MACs, are also some unforgeable verification metadata that authenticate the integrity of a data block. The difference is that HLAs can be aggregated. It is possible to compute an aggregated HLA which authenticates a linear combination of the individual data blocks. At a high level, an HLA-based proof of storage system works as follow. The user still authenticates each element of F = $\{m_i\}$ by a set of HLAs ϕ. The TPA verifies the cloud storage by sending a random set of challenge $\{v_i\}$. The cloud server then returns $\mu=\sum_i v_i.\ m_i$ and its aggregated authenticator σ computed from ϕ.

Though allowing efficient data auditing and consuming only constant bandwidth, the direct adoption of these HLA based techniques is still not suitable for our purposes. This is because the linear combination of blocks, $\mu=\sum_i v_i.\ m_i$, may potentially reveal user data information to TPA, and violates the privacy-preserving guarantee. Specifically, by challenging the same set of c block m1, m2 . . ., mc using c different sets of random coefficients $\{v_i\}$, TPA can accumulate c different linear ombinations $\mu_1, \ldots \mu_c$. With $\{\mu_i\}$ and $\{v_i\}$, TPA can derive the user’s data $m_1, m_2, \ldots, m_c$ by simply solving a system of linear equations.


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

4. *Privacy Preserving Public Auditing Scheme*

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 [19].

Scheme details: Let G1, G2, and G_T be multiplicative cyclic groups of prime order p, and e: G1 X G2 => G_T be a bilinear map as introduced in preliminaries. Let g be a generator of G2. H(.) is a secure map-to-point hash function: {0, 1}*=>G1, which maps strings uniformly to G1. Another hash function h(.) : G_T => Zp maps group element of G_T uniformly to Zp. Our scheme is asfollows:

Setup Phase: The cloud user runs KeyGen to generate the public and secret parameters. Specifically, the user chooses a random signing key pair (spk, ssk), a random x <=Zp, a random element u <= G1, and computes v <= g^x. The secret parameter is sk = (x, ssk) and the public parameters are pk = (spk, v, g, u; e(u, v)).

Given a data file F= $\{m_i\}$, the user runs SigGen to compute authenticator σ_i <= H((W_i) . u^{mi} $)^x$ € G1 for each i. Here, W_i = name || i and name is chosen by the user uniformly at random from Zp as the identifier of file F. Denote the set of authenticators by $\phi = \{\sigma_i\}_{1<=i<=n}$. The last part of SigGen is for ensuring the integrity of the unique file identifier name. One simple way to do this is to compute t=name || SSigssk(name) as the file tag for F, where SSigssk(name) is the signature on name under the private key ssk. For simplicity, we assume the TPA knows the number of blocks n. The user then sends F along with the verification metadata (ϕ, t) to the server and deletes them from local storage.

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 SSigssk(name) via spk, and quits by emitting FALSE if the verification fails. Otherwise, the TPA recovers name.

Now it comes to the "core" part of the auditing process. To generate the challenge message for the audit "chal," the TPA picks a random c- element subset I = {s1, s2, s3,..., sc} of set [1, n]. For each element i € I, the TPA also chooses a random value v_i. The message "chal" specifies the positions of the blocks required to be checked. The TPA sends chal = {(i, v_i)} i€I to the server.

Upon receiving challenge chal = {(i, v_i)} i€I , the server runs GenProof to generate a response proof of data storage correctness. Specifically, the server chooses a random element r <= Zp, and calculates R = $e(u, v)^r$ € GT . Let μ' denote the linear combination of sampled blocks specified in chal:

μ' = $\sum_{i€I} v_i m_i$. To blind μ' with r, the server computes: μ = r +ϒ μ' mod p, where ϒ= h(R) € Zp. Meanwhile, the server also calculates an aggregated authenticator σ = $Л_{i€I}\ \sigma_i^{vi}$ € G1. It then sends {μ, σ, R}, as the proof of storage correctness to the TPA. With the response, the TPA runs VerifyProof to validate it by first computing ϒ = h(R) and then checking the verification equations.

5. *Support for Data Dynamics*

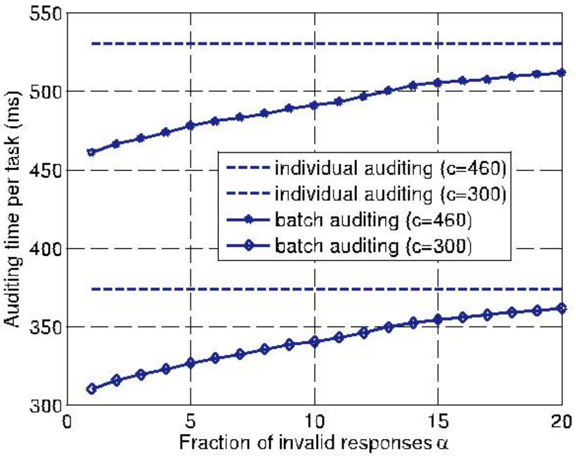
In cloud computing, outsourced data might not only be accessed but also updated frequently by users for various application purposes [21], [8], [22], [23]. Hence, supporting data dynamics for privacy-preserving public auditing is also of paramount importance. Now, we show how to build upon the existing 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 mi 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 σi = $(H(mi) . u^{mi})^x$. We can adopt this technique in our design to achieve privacy preserving public auditing with support of data dynamics.

6. *Generalisation*

As mentioned before, our protocol is based on the HLA in [13]. It has been shown in [25] that HLA can be constructed by homomorphic identification protocols. One may apply the random masking technique we used to construct the corresponding zero knowledge proof for different homomorphic identification protocols. Therefore, our privacy preserving public auditing system for secure cloud storage can be generalized based on other complexity assumptions, such as factoring [25].

IV. Zero Knowledge Public Auditing



Comparison on auditing time between batch and individual auditing, when α-fraction of 256 responses are invalid: Per task auditing time denotes the total auditing time divided by the number of tasks.

The above auditing protocol achieves zero knowledge information leakage to the TPA, and it also ensures the storage correctness guarantee.

Proof:

Zero-knowledge is easy to see. Randomly pick ϒ; μ; & ζ from Zp and ∑ from G1, set R<= $e((Л_{i=s1}^{sc} H(W_i)^{vi})^{ϒ} . u^{Л}$, v) . $e(g1, g)^{ζ}/e(∑^{ϒ}$; g) and backpatch ϒ = h(R). For proof of storage correctness, we can extract ῤ similar to the extraction of μ'. Likewise, σ can be recovered from ∑. To conclude, a valid pair of σ and μ' can be extracted.

V. Related Work

Ateniese et al. [9] are the first to consider public auditability in their "provable data possession" (PDP) model for ensuring possession of data files on untrusted storages. They utilize the RSA-based homomorphic linear authenticators for auditing outsourced data and suggest randomly sampling a few blocks of the file. However, among their two proposed schemes, the one with public auditability exposes the linear combination of sampled blocks to external auditor. When used directly, their protocol is not provably privacy preserving, and thus may leak user data information to the external auditor. Juels et al. [11] describe a "proof of retrievability" (PoR) model, where spot-checking and error-correcting codes are used to ensure both "possession" and "retrievability" of data files on remote archive service systems. However, the number of audit challenges a user can perform is fixed a prior, and public auditability is not supported in their main scheme. Although they describe a straightforward Merkle-tree construction for public PoRs, this approach only works with encrypted data. Later, Bowers et al. [18] propose an improved framework for POR protocols that generalizes Juels' work. Dodis et al. [29] also give a study on different variants of PoR with private auditability. Shacham and Waters [13] design an improved PoR scheme built from BLS signatures [19] with proofs of security in the security model defined in [11]. Similar to the construction in [9], they use publicly verifiable homomorphic linear authenticators that are built from provably secure BLS signatures.

Based on the elegant BLS construction, a compact and public verifiable scheme is obtained. Again, their approach is not privacy preserving due to the same reason as [9]. Shah et al. [15], [10] propose introducing a TPA to keep online storage honest by first encrypting the data then sending a number of pre-computed symmetric-keyed hashes over the encrypted data to the auditor. The auditor verifies the integrity of the data file and the server's possession of a previously committed decryption key. This scheme only works for encrypted files, requires the auditor to maintain state, and suffers from bounded usage, which potentially brings in online burden to users when the keyed hashes are used up.

Dynamic data have also attracted attentions in the recent literature on efficiently providing the integrity guarantee of remotely stored data. Ateniese et al. [21] is the first to propose a partially dynamic version of the prior PDP scheme, using only symmetric key cryptography but with a bounded number of audits. In [22], Wang et al. consider a similar support for partially dynamic data storage in a distributed scenario with additional feature of data error localization. In a subsequent work, Wang et al. [8] propose to combine BLS-based HLA with MHT to support fully data dynamics. Concurently, Erway et al. [23] develop a skip list-based scheme to also enable provable data possession with full dynamics support. However, the verification in both protocols requires the linear combination of sampled blocks as an input, like the designs in [9], [13], and thus does not support privacy-preserving auditing.

In other related work, Sebe et al. [30] thoroughly study a set of requirements which ought to be satisfied for a remote data possession checking protocol to be of practical use. Their proposed protocol supports unlimited times of file integrity verifications and allows preset tradeoff between the protocol running time and the local storage burden at the user. Schwarz and Miller [31] propose the first study of checking the integrity of the remotely stored data across multiple distributed servers. Their approach is based on erasure-correcting code and efficient algebraic signatures, which also have the similar aggregation property as the homomorphic authenticator utilized in our approach. Curtmola et al. [32] aim to ensure data possession of multiple replicas across the distributed storage system. They extend the PDP scheme in [9] to cover multiple replicas without encoding each replica separately, providing guarantee that multiple copies of data are actually maintained. In [33], Bowers et al. utilize a two-layer erasure-correcting code structure on the remotely archived data and extend their POR model [18] to distributed scenario with high-data availability assurance. While all the above schemes provide methods for efficient auditing and provable assurance on the correctness of remotely stored data, almost none of them necessarily meet all the requirements for privacy-preserving public auditing of storage. Moreover, none of these schemes consider batch auditing, while our scheme can greatly reduce the computation cost on the TPA when coping with a large number of audit delegations.

Portions of the work presented in this paper have previously appeared as an extended abstract in [1]. We have revised the paper a lot and improved many technical details as compared to [1]. The primary improvements are as follows: First, we provide a new privacy-preserving public auditing protocol with enhanced security strength in a previous section. Second, based on the enhanced main auditing scheme, we provide a new provably secure batch auditing protocol. All the experiments in our performance evaluation for the newly designed protocol are completely redone. Finally, we provide formal analysis of privacy-preserving guarantee and storage correctness, while only heuristic arguments are sketched in [1].

VI. 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] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Storage Security in Cloud Computing," Proc. IEEE INFOCOM '10, Mar. 2010.

[2] P. Mell and T. Grance, "Draft NIST Working Definition of Cloud Computing," http://csrc.nist.gov/groups/SNS/cloud-computing/index.html, June 2009.

[3] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," Technical Report UCB-EECS-2009-28, Univ. of California, Berkeley, Feb. 2009.

[4] Cloud Security Alliance , "Top threats to Cloud Computing [29].

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

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

[7] Amazon.com, "Amazon s3 Availability Event: July 20, 2008,"

http://status.aws.amazon.com/s3-20080720.html, July 2008.

[8] 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.

[9] 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.

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

[11] 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.

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

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

[14] 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.

[15] 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.

[16] 104th United States Congress, "Health Insurance Portability and Accountability Act of 1996 (HIPPA)," http://aspe.hhs.gov/ admnsimp/pl104191.htm, 1996.

[17] R. Curtmola, O. Khan, and R. Burns, "Robust Remote Data Checking," Proc. Fourth ACM Int'l Workshop Storage Security and Survivability (StorageSS '08), pp. 63-68, 2008.

[18] F. Sebe, J. Domingo-Ferrer, A. Martı´nez-Balleste, Y. Deswarte, and J.-J. Quisquater, "Efficient Remote Data Possession Checking in Critical Information Infrastructures," IEEE Trans. Knowledge and Data Eng., vol. 20, no. 8, pp. 1034-1038, Aug. 2008.