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], and [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]. Moreover, there are legal regulations, such as the US HLAlth Insurance Portability and Accountability Act (HIPAA) [16], further demanding the outsourced data not to be leaked to external parties [10]. 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. 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 domain. 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. Our work is among the first few ones to support privacy-preserving trusted public auditing in cloud computing, with a focus on data storage. Besides, with the prevalence of cloud computing, a predictable increase of auditing tasks from different users may be deputed to TPA. 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 Homogenous elongate 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 collecting and algebraic properties of the authenticator further benefit our design for the batch auditing. Specifically, our contribution can be summarized as the following three aspects:

1. We motivate the Trusted public auditing system of data storage security in cloud computing and provide a privacy-preserving auditing protocol. Our scheme enables an external auditor to audit user's cloud data without learning the data content.
2. To the best of our knowledge, our scheme is the first to support scalable and efficient privacy-preserving public storage auditing in cloud. Specifically, our scheme achieves batch auditing where multiple deputed 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.

The rest of the paper is organized as follows: Section 2 introduces the system and threat model, and our design goals. Then, we provide the detailed description of our scheme in Section 3. Section 4 gives the security analysis and performance evaluation. Section 5 presents further discussions on a zero-knowledge auditing protocol, followed by Section 6 that overviews the related work. Finally, Section 7 gives the concluding remark of the whole paper.

II. PROBLEM STATEMENT

2.1 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 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.

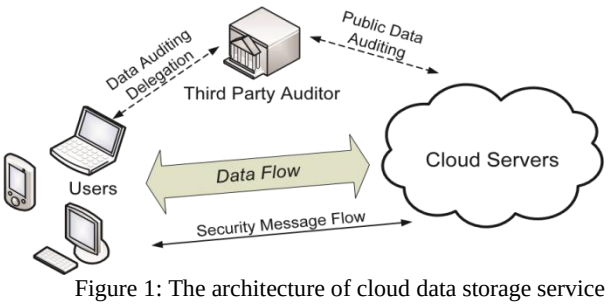


Figure 1: The architecture of cloud data storage service

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.

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. On the one hand, there are regulations, e.g., HIPAA [16], requesting CS to maintain users' data privacy. On the other hand, as users' data belong to their business asset [10], there also exist financial incentives for CS to protect it from any external parties. Therefore, we assume that neither CS nor TPA has motivations to collude with each other during the auditing process. In other words, neither entity will deviate from the prescribed protocol execution in the following presentation.

To authorize the CS to respond to the audit deputed 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.

2.2 Design Goals

To enable privacy-preserving trusted public auditing for cloud data storage under the aforenamed 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 cHLAting 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 deputations 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 supports the trusted public auditing in cloud computing, with a focus on data storage. Besides, with the prevalence of cloud computing, a predictable increase of auditing tasks from different users may be deputed to TPA. Our work also utilizes the technique of public key-based homogenous linear authenticator or HLA 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 collecting and algebraic properties of the authenticator further benefit our design for the batch auditing. Finally, we discuss how to generalize our trusted public auditing scheme and its support of data dynamics.

3.1 Notation and Preludes

- F—the data file to be outsourced, denoted as a sequence of n blocks m1;...;mi;...;mn2 ZZpfor some large prime p.
- MACðÞðÞ—message authentication code (MAC) function, defined as: Kf0;1g!f0;1gl where K denotes the key space.
- HðÞ, hðÞ—cryptographic hash functions.

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

Bilinear Map.Let GG_1, GG_2, and GG_T be multiplicative cyclic groups of prime order p. Let g_1 and g_2 be generators of GG_1 and GG_2, respectively. A bilinear map is a map e : GG_1 GG_2 !GG_T such that for all u 2 GG_1, v 2 GG_2 and a;b 2 ZZ_p, eðu^a;v^{b}Þ¼ eðu;vÞ^{ab}. This bilinearity implies that for any u_1, u_2 2 GG_1, v 2 GG_2, eðu_1 u_2;vÞ¼ eðu_1;vÞ eðu_2;vÞ. Of course, there exists an efficiently computable algorithm for computing e and the map should be nontrivial, i.e., e is nondegenerate: eðg_1;g_2Þ6¼1.

3.2 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 Trusted public auditing system.

This Trusted 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. Gen Proof is run by the cloud server to generate a proof of data storage correctness, while Verify Proof is run by the TPA to audit the proof.

Running a Trusted 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 Verify Proof.

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 Trusted 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.[1]

3.3 The Basic Schemes

Before giving our main result, we study two classes of schemes. 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 trusted 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 Homogenous elongate 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 skto 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. 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 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 deputations 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.

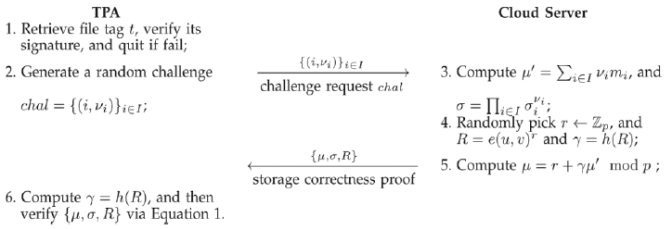
HLA-based solution. To effectively support public auditability without having to retrieve the data blocks themselves, the HLA technique [9], [13], [8] can be used. HLAs, like MACs, are also some unforgeable verification metadata that authenticate the integrity of a data block. The difference is that HLAs can be aggregated. It is possible to compute an aggregated HLA which authenticates a linear combination of the individual data blocks.

The idea is as follows: Before data outsourcing, the cloud user chooses s random message authentication code keys $fskg_{1s}$, recomputes (deterministic) MACs, $fMAC_{sk}ðFÞg_{1s}$ 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 skto 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 $MAC_{sk}ðFÞ$and sk. However, it suffers from the following severe drawbacks: 1) the number of times a particular data file can

At a high level, an HLA-based proof of storage system works as follow. The user still authenticates each element of F ¼fm_ig by a set of HLAs . The TPA verifies the cloud storage by sending a random set of challenge f_ig. The cloud server then returns ¼$_{ii}$ m_i and its aggregated authenticator computed from .

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, ¼P_{ii} 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 $m_1;m_2;...;m_c$ using c different sets of random coefficients f_ig, TPA can accumulate c different linear combinations $_1;...;_c$. With f_igand fig, TPA can derive the user's data m1, m2; . . .;mcby simply solving a system of linear equations 1. We refer readers to [17], [18] for the details on integration of error correcting codes and remote data integrity checking.

TABLE 1: Trusted public auditing for secured cloud storage



3.4 Trusted public auditing Scheme

Overview. To achieve privacy-preserving Trusted public auditing, we propose to uniquely integrate the Homogenous elongate authenticator with random masking technique. In our protocol, the linear combination of sampled blocks in the server's response is disguised 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 GG_1, GG_2, and GG_T be multiplicative cyclic groups of prime order p, and e : GG_1 GG_2 ! GG_T be a bilinear map as introduced in preliminaries. Let g be a generator of GG_2. HðÞis a secure map-to-point hash function: f0;1g! GG_1, which maps strings uniformly to GG_1. Another hash function hðÞ: GG_T ! ZZ_pmaps group element of GG_T uniformly to ZZ_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 ðspk;sskÞ, a random x ZZ_p, a random element u GG_1, and computes v g^x. The secret parameter is sk ¼ðx;sskÞand the public parameters are pk ¼ðspk;v;g;u;eðu;vÞÞ.

Given a data file F ¼fm_ig, the user runs SigGen to compute authenticator $_i$ðHðW$_i$Þu^{mi}Þ^{x}2 GG_1 for each i. Here, W$_i$¼ namekiand name is chosen by the user uniformly at random from ZZ_pas the identifier of file F.
Denote the set of authenticators by ¼f$_i$g$_{1in}$.

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 ¼ namekSSig$_{ssk}$ðnameÞas the file tag for F, where SSig$_{ssk}$ðnameÞis 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 ð;tÞto 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 SSig$_{ssk}$ðnameÞvia 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 ¼fs$_1$;...;s$_c$gof set ½1;n. For each element i 2 I, the TPA also chooses a random value $_i$(of bit length that can be shorter than jpj, as explained in [13]). The message "chal" specifies the positions of the blocks required to be checked. The TPA sends chal ¼fði;$_i$Þg$_{i2I}$to the server.

Upon receiving challenge chal ¼fði; $_i$Þg$_{i2I}$, the server runs GenProof to generate a response proof of data storage correctness. Specifically, the server chooses a random element r ZZ_p, and calculates R ¼ eðu;vÞ^{r}2 GG_T . Let ^{0}denote the linear combination of sampled blocks specified in chal: ^{0} ¼P$_{i2I}$ $_i$m$_i$. To blind ^{0} with r, the server computes: ¼ r þ ^{0}mod p, where ¼ hðRÞ2 ZZ_p. ¼Q $_{i2I}$ ^{i}2 GG_1. It then sends f;;Rgas the response proof of storage Meanwhile, the server also calculates an aggregated authenticator correctness to the TPA. With the response, the TPA runs Verify Proof to validate it by first computing ¼hðRÞand then checking the verification equation

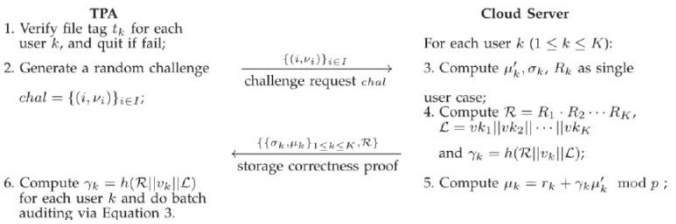
Properties of our protocol. It is easy to see that our protocol achieves public auditability. There is no secret keying material or states for the TPA to keep or maintain between audits, and the auditing protocol does not pose any potential online burden on users. This approach ensures the privacy of 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 hðRÞand 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 over HLA for server's response during the audit: the size of f;;Rgis independent of the number of sampled blocks c.

Previous work [9], [8] 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.

For some cloud storage providers, it is possible that certain information dispersal algorithms (IDA) may be used to fragment and geographically distribute the user's outsourced data for increased availability. We note that these cloud side operations would not affect the behavior of our proposed mechanism, as long as the IDA is systematic, i.e., it preserves user's data in its original form after encoding with redundancy. This is because from user's perspective, as long as there is a complete yet unchanged copy of his outsourced data in cloud, the precomputed verification metadata ð;tÞwill remain valid. As a result, those metadata can still be utilized in our auditing mechanism to guarantee the correctness of user's outsourced cloud data.

Storage and communication tradeoff. As described above, each block is accompanied by an authenticator of equal size of jpjbits. This gives about 2storage over HLA on server. However, as noted in [13], we can introduce a parameter s in the authenticator construction to adjust this storage over HLA, in the cost of communication over HLA in the auditing protocol between TPA and cloud server. In particular, we assume each block m_i consists of s sectors fm_{ij}gwith 1 j s,

TABLE 2 : The Batch Auditing Protocol



where m$_{ij}$2 ZZ_p. The public parameter pkis now ðspk;v;g;fu$_j$g;feðu$_j$;vÞgÞ, 1 j s, where u$_1$;u$_2$;...;u$_s$ are randomly

To respond to the auditing challenge chal ¼fði;iÞgi2I, for 1 j s, the cloud server chooses a random elements r_jZZ$_p$, and calculates R_j¼ eðu;vÞ^{rj}2 GG$_T$. Then, the server blinds each ^0_j ¼P$_{i2I\ i}$m$_{ij}$with r_j, and derives the blinded $_j$¼ r_jþ ^0_j mod p, where ¼ hðR$_1$kR$_2$kkR$_s$Þ2 ZZ$_p$. The aggregated authenticator is still computed as before. It then sends f;f$_j$;R$_j$g$_{1js}$g as the proof response to TPA. With the proof, TPA first computes ¼ hðR$_1$kR$_2$kkR$_s$Þ,

The correctness elaboration is similar to (1) and thus omitted. The overall storage over HLA is reduced to ð1 þ 1=sÞ, but the proof size now increases roughly s due to the additional s element pairs f$_j$;R$_j$g$_{1js}$that the cloud server has to return. For presentation simplicity, we continue to choose s ¼ 1 in our following scheme description. We will present some experiment results with larger choice of s in Section 4.

3.5 Support for Batch Auditing

With the establishment of privacy-preserving Trusted public auditing, the TPA may concurrently handle multiple auditing upon different users' deputation. The individual auditing of these tasks for the TPA can be tedious and very inefficient. Given K auditing deputations 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 collecting of K verification equations (forK 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$_k$¼ðm$_{k;1}$;...;m$_{k;n}$Þto be outsourced to the cloud server, where k 2f1;...;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$_k$ÞÞwhere 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$2 ZZ$_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 mk;i: k;iHðnamekkiÞumkk;ixk

$$¼HðW_{k;i}Þu^{m_{k;ixk}}_k2\ GG_1ði\ 2f1;...;ngÞ;$$

whereW$_{k;i}$¼ name$_k$ki. Finally, each user k sends file F$_k$, set of authenticators $_k$, and tag t$_k$to the server and deletes them from local storage.

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 name$_k$and sends the audit challenge chal ¼fði;iÞgi2Ito the server for auditing data files of all K users.
Upon receiving chal, for each user k 2f1;...;Kg, the server randomly picks r$_k$2 ZZ$_p$and computes R$_k$¼ eðu$_k$;v$_k$Þ^{rk}. Denote R¼ R$_1$ R$_2$R$_K$, and L¼ vk$_1$kvk$_2$k kvk$_K$, our protocol further requires the server to compute $_k$ ¼ hðRkv$_k$kLÞ. Then, the randomly disguised responses can be generated.

Efficiency improvement. As shown in (3), batch auditing 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 collecting may contain invalid responses, especially f$_k$g$_{1kK}$, 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 collecting. 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 collecting of responses into two halves, and repeat the auditing on halves via (3). TPA may now require the server to send back all the fR$_k$g$_{1kK}$, 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.

3.6 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 Trusted 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 iwith 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 $_{σi}$= (H(mi).umi)x. We can adopt this technique in our design to achieve privacy-preserving Trusted 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 H(mi). In the Audit phase, besides {μ,σ,R}, the server's response should also include {H(m$_i$)}$_{i€I}$and their corresponding auxiliary authentication information aux in the MHT. Upon receiving the response, TPA should first use TR$_{MHT}$ and aux to authenticate {H(m$_i$)}$_{i€I}$computed by the server. Once fHðm$_i$Þg$_{i2I}$ are authenticated, TPA can then perform theQs1auditingconif; ;R;fHðmiÞgi2QIg1 viacH(1)ðm, iÞwherei. All is πs1<i<scH(Wi)viis now replaced by

πsi<i<scH(mi)vi these changes does not interfere with the proposed random masking technique, so data privacy is still preserved. To support data dynamics, each data update would require the user to generate a new tree root TR_{MHT} , which is later sent to TPA as the new metadata for storage auditing task. The details of handling dynamic operations are similar to [8] and thus omitted.

Application to version control system. The above scheme allows TPA to always keep the new tree root for auditing the updated data file. But it is worth noting that our mechanism can be easily extended to work with version control system, where both current and previous versions of the data file F and the corresponding authenticators are stored and need to be audited on demand. One possible way is to require TPA to keep tracks of both the current and previous tree roots generated by the user, denoted as fTR^{1}_{MHT} ;TR^{2}_{MHT};...;TR^{V}_{MHT} g. Here, V is the number of file versions and TR^{V}_{MHT} is the root related to the most current version of the data file F. Then, whenever a designated version v (1 vV) of data file is to be audited, the TPA just uses the corresponding $TR^{v}_{MHT}$to perform the auditing. The cloud server should also keep track of all the versions of data file F and their authenticators, in order to correctly answer the auditing request from TPA. Note that cloud server does not need to replicate every block of data file in every version, as many of them are the same after updates. However, how to efficiently manage such block storage in cloud is not within the scope of our paper.

3.7 Generalization

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

IV. EVALUATION

Security Analysis We evaluate the security of the proposed scheme by analyzing its fulfillment of the security guarantee described in Section 2.2, namely, the storage correctness and privacypreserving property. We start from the single user case, where our main result is originated. Then, we show the security guarantee of batch auditing for the TPA in multiuser setting.

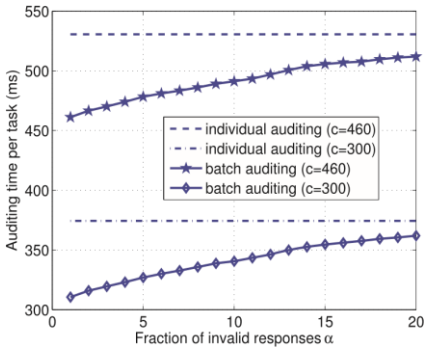


Figure 2. Comparison on auditing time between batch and individual auditing, when -fraction of 256 responses are invalid: Per task auditing time denotes the total auditing time divided by the number of tasks.

Zero Knowledge

Though our scheme prevents the TPA from directly deriving 0 from , it does not rule out the possibility of offline guessing threat by TPA using valid from the response. Specifically, the TPA can always guess whether 0 ¼? ~0, by checking eð;gÞ¼? eððQsi¼c s1 HðWiÞiÞ u~0 ;vÞ, where ~0 is constructed from random coefficients chosen by the TPA in the challenge and the guessed message fm~igs1isc . However, we must note that ~0 is chosen from ZZpand jpjis usually larger than 160 bits in practical security settings (see Section 4.2). Given no background information, the success of this all-or-nothing guess on 0 launched by TPA over such a large space ZZpcan be very difficult. Besides, because TPA must at least make c successful guesses on the same set of blocks to derive fmigs1isc from the system of c linear equations, we can specify c to be large enough in the protocol (e.g., as discussed in Section 3.4, a strict choice of c should be at least larger than 460), which can significantly decrease the TPA's successful guessing probability. In addition, we can also restrict the number of re-auditing on exactly the same set of blocks (e.g., to limit the repeated auditing times on exactly the same set of blocks to be always less than c). In this way, TPA can be kept from accumulating successful guesses on 0 for the same set of blocks, which further diminishes the chance for TPA to solve for fmigs1isc . In short, by appropriate choices of parameter c and group size ZZp, we can effectively defeat such potential offline guessing threat.

Nevertheless, we present a Trusted public auditing scheme with provably zero knowledge leakage. This scheme can completely eliminate the possibilities of above offline guessing attack, but at the cost of a little higher communication and computation over HLA d. The setup phase is similar to our main scheme presented in Section 3.4. The secret parameters are sk ¼ðx;sskÞ and the public parameters are pk ¼ðspk;v;g;u;eðu;vÞ;g1Þ, where g1 2 GG1 is an additional public group element. In the audit phase, upon receiving challenge chal ¼fði;iÞgi2I, the server chooses three random elements rm;r; ZZp, and calculates R ¼ eðg1;gÞreðu;vÞrm2 GGT and ¼ hðRÞ2 ZZp. Let 0 denote the linear combination denote the aggregated of sampled authenticator blocks 0¼¼PQ i2i2II imiii2, GGand1. To ensure the auditing leaks zero knowledge, the server has to blind both 0 and . Specifically, the server computes: ¼ rmþ 0 mod p, and

¼ g1. It then sends f&;;;Rgas the response proof of storage correctness to the TPA, where & ¼ rþ mod p.

With the response from the server, the TPA runs Verify Proof to validate the response by first computing γ= h(R)and then checking the verification equation

Theorem 4.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 ;;& Q from ZZ_p and from GG_1, set R eðð$_i^{s}$ ¼ $^{c}_{s1}$ HðW$_i$Þ^{i}Þ u ;vÞ eðg$_1$;gÞ$^{\&}$=eð;gÞand backpatch ¼ hðRÞ. For proof of storage correctness, we can extract similar to the extraction of ^{0} as in the proof of Theorem 1. Likewise, can be recovered from . To conclude, a valid pair of and^{0} 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 homogenous 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 priori, 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] proposed 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 homogenous linear authenticators that are built from provably secure BLS signatures. Based on 372 IEEE TRANSACTIONS ON COMPUTERS, VOL. 62, NO. 2, FEBRUARY 2013 Fig. 2. Comparison on auditing time between batch and individual auditing, when _-fraction of 256 responses are invalid: Per task auditing time denotes the total auditing time divided by the number of tasks. 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 listbased 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 collecting property as the homogenous linear 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 Trusted 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 deputations.

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 Trusted public auditing protocol with enhanced security strength in Section 3.4. For completeness, we also include an additional (but slightly less efficient) protocol design for provably secure zero-knowledge leakage Trusted public auditing scheme in Section 5.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. Third, we extend our main scheme to support data dynamics in Section 3.6, and provide discussions on how to generalize our privacy-preserving Trusted public auditing scheme in Section 3.7,

which are lacking in [1]. 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 trusted public auditing process for secure cloud storage.To ensure that the TPA does not learn anything about the user data which is stored on the cloud server during the process of auditing, we use Homogenous elongate authenticator algorithm and random masking technique. Because it reduces the burden of cloud user, from auditing process and also provides relief from leakage of data.The experiment conducted on EC2 instance shows the fast performance of our design. Reports also shows that this method is very efficient and highly secure. We can further improve the trusted public auditing process as a multiuser secure cloud storage by making TPA to handle multiple auditing task in batch manner. This is implemented as a future extension.

VII. ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation (NSF) under grants CNS-1054317, CNS1116939, CNS-1156318, and CNS-1117111, and by Amazon web service research grant. A preliminary version [1] of this paper was presented at the 29th IEEE Conference on Computer Communications (INFOCOM '10).

VIII. REFERENCES

[1] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Trusted 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/cloudcomputing/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," http://www.cloudsecurityalliance.org, 2010.

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

[6] J. Kincaid, "MediaMax/TheLinkup Closes Its Doors," http:// www.techcrunch.com/2008/07/10/mediamaxthelinkup-closesits-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, "HLAlth 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] K.D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," Proc. ACM Workshop Cloud Computing Security (CCSW '09), pp. 43-54, 2009.

[19] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," J. Cryptology, vol. 17, no. 4, pp. 297-319, 2004.

[20] A.L. Ferrara, M. Green, S. Hohenberger, and M. Pedersen, "Practical Short Signature Batch Verification," Proc. Cryptographers' Track at the RSA Conf. 2009 on Topics in Cryptology (CT-RSA), pp. 309-324, 2009.

[21] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. Int'l Conf. Security and Privacy in Comm. Networks (SecureComm '08), pp. 1-10, 2008.

[22] C. Wang, Q. Wang, K. Ren, and W. Lou, "Towards Secure and Dependable Storage Services in Cloud Computing," IEEE Trans. Service Computing, vol. 5, no. 2, 220-232, Apr.-June 2012.

[23] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," Proc. ACM Conf. Computer and Comm. Security (CCS '09), pp. 213-222, 2009.

[24] R.C. Merkle, "Protocols for Public Key Cryptosystems," Proc. IEEE Symp. Security and Privacy, 1980.

[25] G. Ateniese, S. Kamara, and J. Katz, "Proofs of Storage from Homogenous Identification Protocols," Proc. 15th Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT), pp. 319-333, 2009.

[26] M. Bellare and G. Neven, "Multi-Signatures in the Plain PublicKey Model and a General Forking Lemma," Proc. ACM Conf. Computer and Comm. Security (CCS), pp. 390-399, 2006.

[27] Amazon.com, "Amazon Elastic Compute Cloud," http://aws. amazon.com/ec2/, 2009.

[28] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. Yau, "Efficient Provable Data Possession for Hybrid Clouds," Cryptology ePrint Archive, Report 2010/234, 2010.

[29] Y. Dodis, S.P. Vadhan, and D. Wichs, "Proofs of Retrievability via Hardness Amplification," Proc. Theory of Cryptography Conf. Theory of Cryptography (TCC), pp. 109-127, 2009.

[30] 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.

[31] T. Schwarz and E.L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage," Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS '06), 2006.

[32] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-Replica Provable Data Possession," Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS '08), pp. 411-420, 2008.

[33] K.D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Proc. ACM Conf. Computer and Comm. Security (CCS '09), pp. 187-198, 2009.