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 towards 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. Our preliminary experiment conducted on Amazon EC2 instance further demonstrates the fast performance of the design.

Keyword: TPA, cloud computing, cryptographic protocols, data storage

I. Introduction

The problem of assessing security is usually approached theoretically. A gap between the theoretical methods proposed and the real life expectations exists, making academic research’s value disputable. Even research which has a more practical perspective can sometimes be insufficient for producing useful results.

There is some property of the population we are interested in. That property is called a parameter. In the case of election auditing, the parameter is the net inflation of the apparent margin compared to the margin a full hand count would show. In the food examples, one parameter is the total number of coconut jelly beans; the other is the total amount of salt in the stock. We want to learn about the parameter without examining every member of the population. Instead, we will look at a subset of the population, called a sample. Samples can be drawn in countless ways. We will consider two: simple random samples and random cluster samples.

A simple random sample is one in which every subset (of a predetermined size) of the population is equally likely to be drawn. Simple random samples are drawn "without replacement." For instance, a simple random sample of 4oz of jelly beans is one in which every 4oz is equally likely to be drawn. Such a sample can be drawn by mixing the jelly beans together really well, then reaching in without looking and scooping out 4oz. A simple random sample of one tablespoon (0.5 ounces) of soup stock can be drawn by putting all the stock in a big cauldron, stirring it well, then dipping in a tablespoon. A simple random sample of 500 ballots can be drawn from a set of 50,000 ballots putting the ballots in a huge basket, stirring them really well, and drawing 500 without looking. (That turns out to be a terrible way to try to draw a simple random sample, because it’s really hard to stir ballots. A better way is to put the ballots in some order, make a list of 50,000 random numbers, and take the sample to be the ballots corresponding to the 500 largest random numbers. For instance, if the 17th random number is the biggest, the 17th ballot would be in the sample.)

A cluster sample is one in which the population is partitioned into non-overlapping groups, called clusters; then a cluster (or a predetermined number of clusters) is drawn at random. A cluster sample of 4oz of jelly beans can be drawn by dividing the beans into 100 4oz bags, then picking one or more of those bags at random. A cluster sample of 12oz of soup stock can be drawn by dividing the soup into 100 12oz cans, then picking one or more cans at random. A cluster sample of 500 ballots could be drawn by picking one of the 100 precincts at random, and taking the sample to be the 500 ballots cast in that precinct. A simple random sample is the same as a cluster sample using clusters of size one. If we estimate the total salt by stirring all the stock together, drawing a tablespoon at random, and multiplying the salt in the tablespoon by 2,400, that estimate is likely to be off by some amount. If we estimate the total salt in the stock by selecting a can at random and multiplying the amount of salt in the can by 100, the estimate is also likely to be off by some amount. The amount by which the estimate is off is called sampling error. The sampling error will tend to be much smaller in the first case, where the cans
are mixed together before the sample is drawn, even though the sample is much smaller (there is 1/24 as much stock in a 0.5oz tablespoon than in a 12oz can). Similarly, if we estimate the net inflation of the margin in the contest to be 100 times the net inflation of the margin in a sample of 500 ballots, that estimate is likely to be off by some amount, the sampling error, owing to the luck of the draw. The sampling error will tend to be smaller if the 500 ballots are a simple random sample than if they are a cluster sample consisting of all the ballots in a single precinct selected at random.

Sampling error tends to be smaller on average for simple random samples than for cluster samples. (There are exceptions, depending on how the clusters are formed. If the clusters are themselves random samples from the population and a single cluster is drawn, there’s no difference between a cluster sample and a simple random sample. If the clusters are constructed so that they exactly match the population, the sampling error will be smaller for a cluster sample than for a simple random sample.) So, for instance, 100 times the net inflation of the margin in a simple random sample of 500 ballots typically tends to be closer to the total net inflation of the margin for the contest than 100 times the net inflation in a cluster sample of 500 ballots will tend to be.

Problem formulation

Having in mind the difficulty of fully describing and performing security assessment, an initial set of problems that require resolution has to be identified.

- Comprehension of the existing methods in the area of security assessment.
- Identification of the design requirements for building assessment methods.
- Ability to assess security in different levels of detail.

Purpose

As mentioned before, there is not much research conducted on the technical aspects of providing a way to measure and compare IT systems’ security. And that is exactly what the goal of this thesis is; to add to the accumulated knowledge in the domain of security assessment from a technical perspective.

First of all, there needs to be a comparison of the available methods, in order to track the advantages and disadvantages of these methods and to be able to decide what needs to be done. This comparison will not be restricted to the technical methods, but also includes methods of a more theoretical character so as to harvest any good aspects presented in them.

By comparing the existing methods, the next goal is to achieve a better understanding of the rules that need to be set when designing assessment methods. The different aspects that can be combined when approaching this issue need to be discussed in order to come up with clear and distinctive design requirements.

II. Proposed System

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 independent arbitration purposes. 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.

We motivate the 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. 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 delegated auditing tasks from different users can be performed simultaneously by the TPA in a privacy-preserving manner. We prove the security and justify the performance of our proposed schemes through concrete experiments and comparisons with the state-of-the-art.

![Fig. 1: The architecture of cloud data storage service](image-url)
III. Related Work

Ateniese et al. [8] are the first to consider public auditability in their defined “provable data possession” (PDP) model for ensuring possession of data files on untrusted storages. Their scheme utilizes the RSA-based homomorphic linear authenticators for auditing outsourced data and suggests randomly sampling a few blocks of the file. However, the public auditability in their scheme demands the linear combination of sampled blocks exposed to external auditor. When used directly, their protocol is not provably privacy preserving, and thus may leak user data information to the auditor. Juels et al. 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. Dodis et al. [25] give a study on different variants of PoR with private auditability. Shacham et al. [13] design an improved PoR scheme built from BLS signatures [17] with full proofs of security in the security model defined in [11]. Similar to the construction in [8], 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 does not support privacy-preserving auditing for the same reason as [8]. Shah et al. [9], [14] propose allowing a TPA to keep online storage honest.

IV. Conclusion

In this paper, we propose a privacy-preserving public auditing system for data storage security in Cloud Computing. 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 multi-user 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.

References