International Journal of Science Engineering and Advance Technology

Confidentiality-Conserving Community Auditing For Protected Haze Storing

Dr.D. Srujan Chandra Reddy#1, Kareti Venkaiah#2
#1 Department Of Cse, Pbr Visvodaya Institute Of Technology & Science,Kavali.
#2student Of M.Tech(C.S) And Department Of Cse, Pbr Visvodaya Institute Of Technology & Science,Kavali.

Abstract— Using Haze Storing, 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 Storing and maintenance. However, the fact that users no longer have physical possession of the outsourced data makes the data integrity protection in Haze Computing a formidable task, especially for users with constrained computing resources. Moreover, users should be able to just use the Haze Storing as if it is local, without worrying about the need to verify its integrity. Thus, enabling Community auditability for Haze Storing 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 Protectedly introduce an effective TPA, the auditing process should bring in no new vulnerabilities towards user data Confidentiality, and introduce no additional online burden to user. In this paper, we propose a Protected Haze Storing system supporting Confidentiality-Conserving Community 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 Protected and highly efficient.

Index Terms—Data Storing, Confidentiality-Conserving, Community auditability, cryptographic protocols, Haze computing ,independent resource pooling, rapid resource

1 INTRODUCTION

CLOUD Computing has been envisioned as the implications, Haze Computing is transforming the next generation information technology (IT) very nature of how businesses use information architecture for technology. One fundamental aspect of this enterprises, due to its long list of unprecedented paradigm shifting is that data is being centralized risk [1]. As a disruptive technology with profound advantages in the IT history: on-demand selfor outsourced to the Haze. From users' service, ubiquitous network access, location perspective, including both individuals and IT enterprises, storing data remotely to the Haze in a flexible on demand manner brings appealing benefits: relief of the burden for Storing management, universal data access with independent geographical locations, and avoidance of capital expenditure on hardware, software, and personnel maintenances, etc [2].

While Haze Computing makes these advantages more appealing than ever, it also brings new and challenging security threats towards users' outsourced data. Since Haze 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 Haze is being put at risk due to the following reasons. First of all, although the infrastructures under the Haze 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. Examples of outages and security breaches of noteworthy Haze services appear from time to time [3]–[7]. Secondly, there do exist various motivations for CSP to behave unfaithfully towards the Haze users regarding the status of their outsourced data. For examples, CSP might reclaim Storing for monetary reasons by discarding data that has not been or is rarely accessed, or even hide data loss incidents so as to maintain a reputation [8]–[10]. In short, although outsourcing data to the Haze is economically attractive for long-term largescale data Storing, it does not immediately offer any guarantee on data integrity and availability. This problem, if not properly addressed, may impede the successful deployment of the Haze architecture.

As users no longer physically possess the Storing 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 un accessed 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 Haze environment can be formidable and expensive for the Haze users
[10], [12]. Moreover, the overhead of using Haze Storing should be minimized as much as possible, such that user does not need to perform too many operations to use the data (in additional to retrieving the data). For example, it is desirable that users do not need to worry about the need to verify the integrity of the data before or after the data retrieval. Besides, there may be more than one user accesses the same Haze Storing, say in an enterprise setting. For easier management, it is desirable that the Haze server only entertains verification request from a single designated party.
To address these problems, our work utilizes the technique of Community key based homomorphic linear authenticator (or HLA for short) [8], [9], [10], 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 Haze server during the efficient auditing process. The aggregation 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 Community auditing system of data Storing security in Haze Computing and provide a Confidentiality-Conserving auditing protocol, i.e., our scheme enables an external auditor to audit user's outsourced data in the Haze without learning the data content.
2) To the best of our knowledge, our scheme is the first to support scalable and efficient Community auditing in the Haze Computing. Specifically, our scheme achieves batch auditing where multiple delegated auditing tasks from different users can be performed simultaneously by the TPA.
3) We prove the security and justify the performance of our proposed schemes through concrete experiments and comparisons with the state-oftheart.
.

2 PROBLEM STATEMENT

2.1 The System and Threat Model We consider a Haze data Storing service involving three different entities, as illustrated in Fig. 1: the Haze user (U), who has large amount of data files to be stored in the Haze; the Haze server (CS), which is managed by the Haze service provider (CSP) to provide data Storing service and has significant Storing space and computation resources (we will not differentiate CS and CSP hereafter); the third party auditor (TPA), who has expertise and capabilities that Haze users do not have and is trusted to assess the Haze Storing service reliability on behalf of the user upon request. Users rely on the CS for Haze data Storing and maintenance. They may also dynamically interact with the CS to access and update their stored data for various application purposes. To save the computation resource as well as the online burden, Haze users may resort to TPA for ensuring the Storing integrity of their outsourced data, while hoping to keep their data private from TPA.
We consider the existence of a semi-trusted CS as [11] does. Namely, in most of time it behaves properly and does not deviate from the prescribed protocol execution. However, for their own benefits the CS might neglect to keep or deliberately delete rarely accessed data files which belong to ordinary Haze users. Moreover, the CS may decide to hide the data corruptions caused by server hacks or Byzantine failures to maintain reputation. We assume the TPA, who is in the business of auditing, is reliable and independent, and thus has no incentive to collude with either the CS or the users during the auditing process. However, it harms the user if the TPA could learn the outsourced data after the audit.
To authorize the CS to respond to the audit delegated to TPA's, the user can sign a certificate granting audit rights to the TPA's Community 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 Confidentiality-Conserving Community auditing for Haze data Storing under the aforementioned model, our protocol design should achieve the following security and performance guarantees.

1) Community auditability: to allow TPA to verify the correctness of the Haze data on demand without retrieving a copy of the whole data or introducing additional online burden to the Haze users.

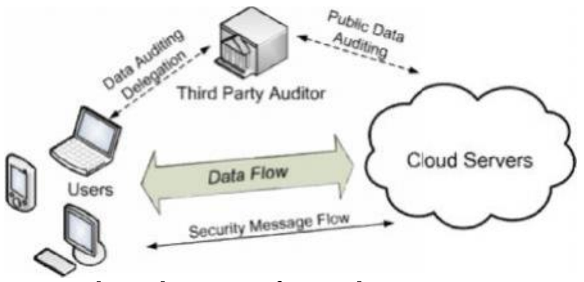


Fig. 1: The architecture of Haze data Storing service

2) Storing correctness: to ensure that there exists no cheating Haze server that can pass the TPA's audit without indeed storing users' data intact.

3) Confidentiality-Conserving: 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 Protected and efficient auditing capability to cope with multiple auditing delegations from possibly large number of different users simultaneously.

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

3 THE PROPOSED SCHEMES

This section presents our Community auditing scheme which provides a complete outsourcing solution of data – not only the data itself, but also its integrity checking. We start from an overview of our Community 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 Confidentiality-Conserving Community auditing scheme and its support of data dynamics.

3.1 DEFINITIONS AND FRAMEWORK

We follow a similar definition of previously proposed schemes in the context of remote data integrity checking [8], [11] and adapt the framework for our Confidentiality-Conserving Community auditing system.

A Community 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 MAC, signatures, or other related information that will be used for auditing. GenProof is run by the Haze server to generate a proof of data Storing correctness, while VerifyProof is run by the TPA to audit the proof from the Haze server.

Running a Community auditing system consists of two phases, Setup and Audit:

• Setup: The user initializes the Community 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 Haze server, and deletes 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 Haze server to make sure that the Haze server has retained the data file F properly at the time of the audit. The Haze server will derive a response message from a function of the stored data file F and its verification metadata by executing GenProof. The TPA then verifies the response via VerifyProof.

Our framework assumes the TPA is stateless, which is a desirable property achieved by our proposed solution. It is easy to extend the framework above to capture a stateful auditing system, essentially by spliting the verification metadata into two parts which are stored by the TPA and the Haze server respectively.

Our design does not assume any additional property on the data file. If the user wants to have more errorresiliency, he/she can always first redundantly encodes the data file and then uses our system with the data file that has errorcorrecting codes integrated.

3.2 CONFIDENTIALITY-CONSERVING

COMMUNITY AUDITING SCHEME

Overview. To achieve Confidentiality-Conserving Community 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 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 Community key based HLA, to equip the auditing protocol with Community auditability. Specifically, we use the HLA proposed in [11], which is based on the short signature scheme proposed by Boneh, Lynn and Shacham (hereinafter referred as BLS signature) [12].

Efficiency Improvement. As shown in Equation 2, 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 + 1. Thus, a considerable amount of auditing time is expected to be saved.

Identification of Invalid Responses. The verification equation (Equation 2) only holds when all the responses are valid, and fails with high probability when there is even one single invalid response in the batch auditing, as we will show in Section 4. In many situations, a response collection may contain invalid responses, especially {µk}1≤k≤K, caused by accidental data corruption, or possibly malicious activity by a Haze server. The ratio of invalid responses to the valid could be quite small, and yet a standard batch auditor will reject the entire collection. To further sort out these invalid responses in the batch auditing, we can utilize a recursive divideand-conquer approach (binary search), as suggested by . Specifically, if the batch auditing fails, we can simply divide the collection of responses into two halves, and recurse the auditing on halves via Equation 2. TPA may now require the server to send back all the {Rk}1≤k≤K, as in individual auditing. we show through carefully designed experiment that using this recursive binary search approach, even if up to 18% of responses are invalid, batch auditing still performs faster than individual verification.

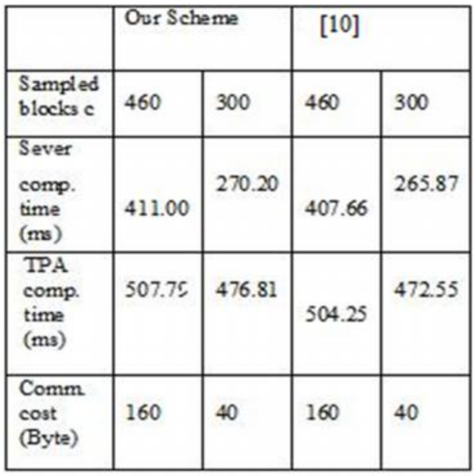
3.3 SUPPORT FOR DATA DYNAMICS

In Haze Computing, outsourced data might not only be accessed but also updated frequently by users for various application purposes [10], [11]. Hence, supporting data dynamics for Confidentiality Conserving Community auditing is also of paramount importance. Now we show how to build upon the existing work [10] and adapt our main scheme to support data dynamics, including block level operations of modification, deletion and insertion.
In [10], 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) [12] 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 Confidentiality Conserving Community risk auditing with support of data dynamics. Specifically, in the Setup phase, the user has to generate and send the tree root TRMHT 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(mi)} iIand their corresponding auxiliary authentication information aux in the MHT. Upon receiving the response, TPA should first use TRMHT and aux to authenticate {H(mi)} iIcomputed by the server. Once {H(mi)} iIare authenticated, TPA can then perform the auditing on {µ,σ,R,{H(mi)}i I} via Equation 1, where Qs1≤i≤sc H(Wi) viis now replaced by Qs1≤i≤sc H(mi) vi. Data Confidentiality is still preserved due to the random mask. The details of handling dynamic operations are similar to [10] and thus omitted.

3.4 LEARNING µ ′ FROM Σ

Though our scheme prevents the TPA from directly deriving µ′ from µ, it does not rule out the possibility of offline guessing attack from the TPA using valid σ from the response. Specifically, the TPA can always guess whether the stored data contains certain message m˜ , by checking 4.1 Security Guarantee for Batch Auditing Now we show that our way of extending our result to a multi-user setting will not affect the Aforementioned.

TABLE 1: Performance under different number of sampled auditing.



	Our Scheme		[10]	
Sampled blocks c	460	300	460	300
Sever comp. time (ms)	411.00	270.20	407.66	265.87
TPA comp. time (ms)	507.79	476.81	504.25	472.55
Comm. cost (Byte)	160	40	160	40

A. 59 bits and the embedding degree 6. The security
Our Scheme [9]

4 EVALUATION

4.1 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 Storing correctness and Confidentiality Conserving property. We start and |p|= message during the Audit phase. Under this setting, we quantify the cost introduced of the Confidentiality Conserving auditing in terms of server computation, auditor computation as well as communication overhead.

The experiment is conducted using C on a Linux system with an Intel Core 2 processor running at 1.86 GHz, 2048 MB of RAM, and a 7200 RPM Western Digital 250 GB Serial ATA drive with an 8 MB buffer. Our code uses the Pairing-Based Cryptography (PBC) library version 0.4.18. The elliptic curve utilized in the experiment is a MNT curve, with base field size of considering only the total number of pairing operations. However, on the practical side, there are additional less expensive operations required for batching, such as modular exponentiations and multiplications. Meanwhile, the different sampling strategies, i.e., different number of sampled blocks c, is also a variable factor that affects the batching efficiency. Thus, whether the benefits of removing pairings significantly outweighs these additional operations is remained to be verified. To get a complete view of batching efficiency, we conduct 160. All experimental results represent the mean of 20 trials.

level is chosen to be 80 bit, which means |νi| = 80 uµ˜′,ν), where µ˜′ is constructed from random coefficients chosen by the TPA in the challenge and the guessed message m˜ . Thus, our main scheme is not semantically Protected yet. However, we must note that µ˜′ is chosen from Zpand |p| is usually larger than 160 bits in practical security settings (see Section 4.2). Therefore, the TPA has to test basically all the values of Zpin order to make a successful guess.

4.2PERFORMANCE ANALYSIS

completeness, we will give a provably zero- We now assess the performance of the proposed knowledge based Community auditing scheme, Confidentiality-Conserving Community auditing which further eliminates the possibilities of above schemes to show that they are indeed offline guessing attack.

4.2.1 BATCH AUDITING EFFICIENCY

Discussion in Section 3.5 gives an asymptotic batch auditing for the TPA in multi-user setting. efficiency analysis on the batch auditing, by a timed batch auditing test, where the number of auditing tasks is increased from 1 to approximately 200 with intervals of 8. The performance of the corresponding non batched (individual) auditing is provided as a baseline for the measurement. Following the same experimental settings c = 300 and c = 460, the average per task auditing time, which is computed by dividing total auditing time by the number of tasks, is given in Fig. 4 for both batch and individual auditing. It can be shown that compared to individual auditing, batch auditing indeed helps reducing the TPA's computation cost, as more than 11% and 14% of per-task auditing time is saved, when c is set to be 460 and 300, respectively.

4.2.3SORTING OUT INVALID RESPONSES

Now we use experiment to justify the efficiency of our recursive binary search approach for the TPA to sort out the invalid responses when batch auditing fails, as discussed in Section 3.5. This experiment is

tightly pertained to the work in, which evaluates the batch verification efficiency of various short signatures.
To evaluate the feasibility of the recursive approach, we first generate a collection of 256 valid responses, which implies the TPA may concurrently handle 256 different auditing delegations. We then conduct the tests repeatedly while randomly corrupting an αfraction, ranging from 0 to 18%, by replacing them with random values. The average auditing time per task against the individual auditing approach is presented . The result shows that even the number of invalid responses exceeds 15% of the total batch size, the performance of batch auditing can still be safely concluded as more preferable than the straightforward individual auditing. Note that the random distribution of invalid responses within the collection is nearly the worst-case for batch auditing. If invalid responses are grouped together, it is possible to achieve even better results.

5. RELATED WORK

Ateniese et al. [8] are the first to consider Community auditability in their defined "provable data possession" (PDP) model for ensuring possession of data files on untrusted Storings. Their scheme utilizes the RSAbasedhomomorphic linear authenticators for auditing outsourced data and suggests randomly sampling a few blocks of the file. However, the Community auditability in their scheme demands the linear combination of sampled blocks exposed to external auditor. When used directly, their protocol is not provably Confidentiality Conserving, and thus may leak user data information to the 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 Community auditability is not supported in their main scheme. Although they describe a straightforward Merkle-tree construction for Community PoRs, this approach only works with encrypted data. Dodis give a study on different variants of PoR with private auditability. Shacham et al. [3] design an improved PoR scheme built from BLS signatures [7] with full proofs of security in the security model defined in [11]. Similar to the construction in [8], they use Communityly verifiable homomorphic linear authenticators that are built from provably Protected BLS signatures. Based on the elegant BLS construction, a compact and Community verifiable scheme is obtained.

In other related work, Ateniese et al. [9] propose a partially dynamic version of the prior PDP scheme, using only symmetric key cryptography but with a bounded number of audits. In [10], Wang et al. consider a similar support for partial dynamic data Storing in a distributed scenario with additional feature of data error localization. In a subsequent work, Wang et al. [10] propose to combine BLS-based HLA with MHT to support both Community auditability and full data dynamics. Almost simultaneously, Erway et al. [11] developed a skip lists based scheme to enable provable data possession with full dynamics support.

6. CONCLUSION

In this paper, we propose a Confidentiality Conserving Community auditing system for data Storing security in Haze 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 Haze server during the efficient auditing process, which not only eliminates the burden of Haze 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 Confidentiality-Conserving Community uditing 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 Protected and highly efficient.

REFERENCES

[1] P. Mell and T. Grance, "Draft NIST working definition of Haze computing," Referenced on June. 3rd, 2009 Online at http://csrc.nist.gov/groups/SNS/Hazecomputing/i ndex. html, 2009.

[2] 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 Hazes: A berkeley view of Haze computing," University of California, Berkeley, Tech. Rep.

[3]M. Arrington, "Gmail disaster: Reports of mass email deletions," Online at http://www.techcrunch.com/2006/ 12/28/gmail

disaster reports-of-massemaildeletions/, December 2006.

[4]J. Kincaid, "MediaMax/ TheLinkup Closes Its Doors," Online at http://www.techcrunch.com /2008/07/10/ mediamaxthelinkupcloses-its-doors/, July 2008.

[5] Amazon.com, "Amazon s3 availability event: July 20, 2008," Online at http://status.aws.amazon.com/s320080720.ht ml, 2008.

[6] S. Wilson, "Appengine outage," Online at http://www. cioweblog.com/50226711/ appengineoutage.php, June 2008.

[7] B. Krebs, "Payment Processor Breach May Be Largest Ever," Online at http://voices. washingtonpost.com/securityfix/ 2009/01/payment processor breach may b.html, Jan. 2009.

[8] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Proc. of CCS'07, Alexandria, VA, October 2007, pp. 598–609.

[9] M. A. Shah, R. Swaminathan, and M. Baker, "Confidentiality Conserving audit and extraction of digital contents," Cryptology ePrint Archive, Report 2008/186, 2008.

[10] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling Community verifiability and data dynamics for Storing security in Haze computing," in Proc. of ESORICS'09, volume 5789 of LNCS. SpringerVerlag, Sep. 2009, pp. 355–370.

[11] A. Juels and J. Burton S. Kaliski, "Pors: Proofs of retrievability for large files," in Proc. of CCS'07, Alexandria, VA, October 2007, pp. 584–597.

[12] Haze Security Alliance, "Security guidance for critical areas of focus in Haze computing," 2009, http://www. Hazesecurityalliance.org.

Dr.D.Srujan Chandra Reddy, Assoc.Professor ,Department Of Cse, PBR Visvodaya Institute of Technology & Science, Kavali.

Kareti Venkaiah, M.Tech (C.S) Student, Department Of Cse, PBR Visvodaya Institute of Technology & Science, Kavali.