IJRAR.ORG E-ISSN: 2348-1269, P-ISSN: 2349-5138

INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS (IJRAR) | IJRAR.ORG

An International Open Access, Peer-reviewed, Refereed Journal

Preserving the Privacy of Remote Data in Cloud using Bilinear Pairing Auditing Mechanism

Kiruthika M

Department of Computer Science and Engineering

Dr. Mahalingam College of Engineering and Technology, Pollachi, TamilNadu, India

Abstract- Cloud computing is a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications. In cloud computing, data owners host their data on cloud servers and users can access the data from cloud servers. But hosting of data services introduces new security challenges. Owners would worry that the data could be lost in the cloud. Some existing system, remote integrity checking methods can only serve for static data thus cannot be applied to the auditing service since the data in the cloud can be dynamically updated. The Trusted Third Party (TTP) is introduced to audit the data. In this paper, we design an auditing framework for cloud storage systems and propose an efficient and privacy-preserving auditing protocol. Our proposed model provides an independent auditing service to check the integrity of the remote data. Our auditing protocol handles static data as well as dynamic data. The analysis and simulation results show that our proposed auditing protocols are secure and it reduce the computation cost of the auditor.

Keywords – Storage Auditing, Privacy- Preserving Auditing, Bilinear Pairing, Cloud Storage, Cloud Computing

I. Introduction

Cloud storage is an important service of cloud computing [1], which allows data owners (owners) to move data from their local computing systems to the cloud. More and more owners start to store the data in the cloud [2]. However, this new paradigm of data hosting service also introduces new security challenges [3]. Owners would worry that the data could be lost in the cloud. This is because data loss could happen in any infrastructure [4]. Sometimes, cloud service providers might be dishonest. They could discard the data which has not been accessed or rarely accessed to save the storage space and claim that the data are still correctly stored in the cloud. Therefore, owners need to be convinced that the data are correctly stored in the cloud.

Traditionally, owners can check the data integrity based on two-party storage auditing protocols [9]. In cloud storage system, however, it is inappropriate to let either side of cloud service providers or owners conduct such auditing, because none of them could be guaranteed to provide unbiased auditing result. In this situation, third party auditing is a natural choice for the storage auditing in cloud computing. A third party auditor (auditor) that has expertise and capabilities can do a more efficient work and convince both cloud service providers and owners.

II. EXISTING SYSTEM

Recently, several remote integrity checking protocols were proposed to allow the auditor to check the data integrity on the remote server [17]. Table 1 gives the comparisons among some existing remote integrity checking schemes in terms of type of data, the privacy protection, the support of dynamic operations and the batch auditing for multiple owners and multiple clouds.

Table -1 Comparison of Different Integrity Check Techniques

METHOD	TECHNIQUE USED	TYPE OF DATA	PRIVACY	DYNAMIC OPERATI-ONS	BATCH OPERATION	
					MULTI- OWNER	MULTI-USER
PDP [1]	Homomorphic Verifiable tag	static data	Yes	Not allowed	Not Supported	Not Supported
POR [8]	Sentinels	static data	Yes	Not allowed	Not Supported	Not Supported
Public Auditability [12]	Merkle Hash trees and Bilinear aggregation function	Dynamic data	Yes	Allowed	Supported	Not Supported
POR scheme with full proof mechanism [5]	Homomorphic authentication technique with BLS signature	Dynamic data	Yes	Allowed	Not Supported	Not Supported
Cooperative PDP model [16]	Homomorphic Verifiable Response and Hash Index Hierarchy	Dynamic data	No	Not allowed	Not Supported	Not Supported
Dynamic PDP model [7]	Rank based authentication	Dynamic data	No	Allowed but partially	Not Supported	Not Supported
Our Scheme	Bilinear Pairing	Dynamic Data	Yes	Allowed	Supported	Supported

III. SYSTEM MODEL

The auditing system for cloud storage as shown in Figure 1, which involves data owners (owners), the cloud server (server) and the third party auditor (auditor).

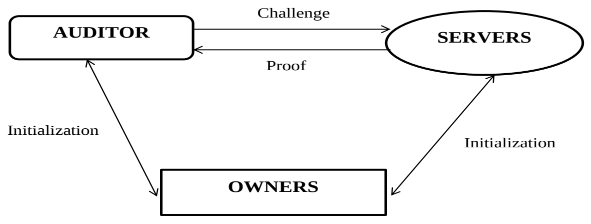

Figure 1. System Model of the Data Storage Auditing

The Owners create the data and host their data in the cloud. The cloud server stores the owner's data and provides the data access to user (data consumer). The auditor is a trusted third party that has expertise and capabilities to provide data storage auditing service for both owners and servers. The auditor can be trusted organization managed by the government, which can provide unbiased auditing result for both owners and servers.

Before describing the auditing protocol definition, we first define some notations as listed in Table 2.

Table -2 Notations

Symbol	Physical Meaning
stk	secret tag key
ptk	public tag key
shk	secret hash key
M	data component
T	set of data tags
n	number of blocks in each component
s	number of sectors in each data block
M_{info}	abstract information of *M*
C	challenge generated by the auditor
P	proof generated by the server

IV. PRIVACY- PRESERVING AUDITING PROTOCOL

In this section, we first present some techniques we applied in the design of our efficient and privacy-preserving auditing protocol. Then, we describe the algorithms and the detailed construction of our auditing protocol for cloud storage systems.

4.1 Problem Definition–

The work provides a solution for the data privacy problem (i.e., the auditing protocol should protect the data privacy against the auditor) in the cloud environment, by introducing a third party auditor to audit and preserves the integrity of the data.

To achieve the data privacy, an encrypted proof is generated with the challenge stamp. The stamp is created using the bilinear property of the bilinear pairing, such that the auditor cannot decrypt it. But the auditor can verify the correctness of the proof without decrypting it.

A file F has m data components as $F = (F_1, \cdot\cdot\cdot, F_m)$. Each data component has its physical meanings and can be updated dynamically by the data owners. For public data components, the data owner does not need to encrypt it, but for private data component, the data owner needs to encrypt it with its corresponding key.

Each data component F_k is divided into n_k data blocks denoted as $F_k = (m_{k1}, m_{k2}, \cdot\cdot\cdot, m_{knk})$. Due to the security reason, the data block size should is restricted by the security parameter. For example, suppose the security level is set to be 160-bit (20Byte), the data block size should be 20-Byte. A 50-KByte data component will be divided into 2500 data blocks and generates 2500 data tags, which incurs 50-KByte storage overhead.

Bilinear Pairing: Let G1, G2 and GT be three multiplicative groups with the same prime order p.
A bilinear map is a map e: G1 × G2 → GT. Let g1 and g2 be the generators of G1 and G2 respectively.
A bilinear map has the following properties:

1) Bilinearity: $e(ua, vb) = e(u, v)^{ab}$ for all u ∈ G1, v ∈ G2 and a, b ∈ Zp.

2) Non-degeneracy: There exist u ∈ G1, v ∈ G2 such that e (u, v) = I, where I is the identity element of GT.

3) Computability: e can be computed in an efficient way.

Such a bilinear map is called a bilinear pairing.

4.2. Algorithms for Auditing Protocol–

The privacy preserving auditing protocol consists of the following algorithms:
Key Generation Algorithm: The key generation algorithm takes no input other than the implicit security parameter p. It chooses two random number stk, shk∈ Zp as the secret tag key and the secret hash key. It outputs the public tag key as ptk = g2stk∈ G2, the secret tag key and the secret hash key.

$$\mathbf{KeyGen\ (p)} \rightarrow \mathbf{(shk, stk, ptk)}$$

Tag Generation Algorithm: The tag generation algorithm takes each data component M, the secret tag key and the secret hash key as inputs. It first chooses s random values x1, x2, · · ·, xs∈ Zp and computes $u_j = g1^{x_j} \in G1$. For each data block m_i ($i \in [1, n]$), it computes a data tag t_i as

$$t_i = (h\,(shk, w_i)\,.\,\prod_{j=1}^{s} u_j^{m_{ij}})^{stk}$$

Where W_i = FID||i (the "||" denotes the concatenation operation), in which FID is the identifier of the data and i represents the block number of mi. It outputs the set of data tags $T = \{t_i\}_{i\in[1,n]}$.

$$\mathbf{TagGen\ (M, stk, shk)} \rightarrow \mathbf{T}$$

Challenge Algorithm: The challenge algorithm takes the abstract information of the data M_{info} as the input. It selects some data blocks to construct the Challenge Set Q and generates a random number $v_i \in Z^*_p$ for each chosen data block m_i ($i \in Q$). Then, it computes the challenge stamp $R = (ptk)^r$ by randomly choosing a number $r \in Z^*_p$. It outputs the challenge as $C = (\{i, v_i\}_{i\in Q}, R)$.

$$\mathbf{Chall\ (M_{info})} \rightarrow \boldsymbol{\rho}$$

Prove Algorithm: The prove algorithm takes as inputs the data M and the received challenge $C = (\{i, v_i\}_{i\in Q}, R)$. The proof consists of the tag proof TP and the data proof DP. The tag proof is generated as

$$TP = \prod_{i\in Q} t_i^{v_i}$$

To generate the data proof, it first computes the sector linear combination of all the challenged data blocks MP_j for each $j \in [1, s]$ as

$$MP_j = \sum_{i\in Q} v_i .\ m_{ij}$$

Then, it generates the data proof DP as

$$DP = \prod_{j=1}^{s} e(u_j, R)^{MP_j}$$

It outputs the proof P = (TP, DP).

$$\textbf{Prove (M, T, ρ)} \rightarrow P$$

Verification Algorithm: The verification algorithm takes as inputs the challenge C, the proof P, the secret hash key, the public tag key and the abstract information of the data component. It first computes the hash values h (shk, W_i) of all the challenged data blocks and computes the challenge hash H_{chal} as

$$H_{chal} = \prod_{i \in Q} h(shk, w_i)^{rv_i}$$

It then verifies the proof from the server by the following verification equation:

$$DP.e\,(H_{chal},\ ptk) == e\,(TP\ ,\ g_2^{\ r})$$

If the above verification equation holds, it outputs 1.Otherwise, it outputs 0.

$$\textbf{Verify (ρ, P, shk, ptk, } M_{info}\textbf{)} \rightarrow 0/1$$

4.3. Construction of our Privacy-Preserving Auditing Protocol Protocol–

As illustrated in Figure 2, our storage auditing protocol consists of three phases: Owner Initialization, Confirmation Auditing and Sampling Auditing. During the system initialization, the owner generates the keys and the tags for the data. After storing the data on the server, the owner asks the auditor to conduct the confirmation auditing to make sure that their data is correctly stored on the server. Once confirmed, the owner can choose to delete the local copy of the data. Then, the auditor conducts the sampling auditing periodically to check the data integrity.

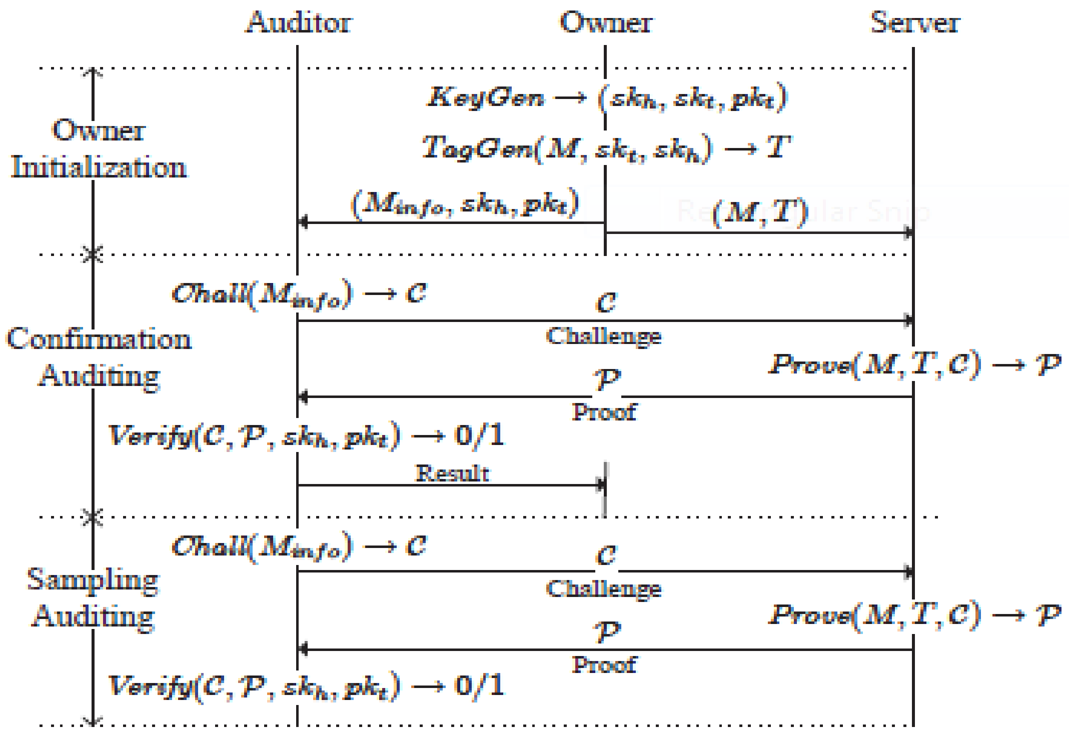

Figure 2. Framework for Privacy-Preserving Auditing Protocol

Phase 1: Owner Initialization

The owner runs the key generation algorithm KeyGen to generate the secret hash key shk, the pair of secret-public tag key (stk, ptk). Then, it runs the tag generation algorithm TagGen to compute the data tags. After all the data tags are generated, the owner sends each data component $M = \{m_i\}_{i \in [1,n]}$ and its corresponding data tags $T = \{t_i\}_{i \in [1,n]}$ to the server together with the set of parameters $\{u_j\}_{j \in [1,s]}$. The owner then sends the public tag key ptk, the secret hash key shk and the abstract information of the data M_{info} to the auditor, which includes the data identifier FID, the total number of data blocks n.

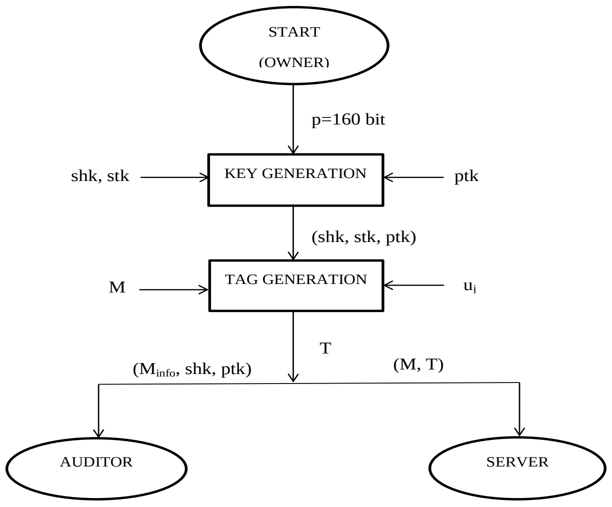

Figure 3. Flow Diagram for Owner Initialization

Phase 2: Confirmation Auditing

In our auditing construction, the auditing protocol only involves two-way communication: Challenge and Proof. During the confirmation auditing phase, the owner requires the auditor to check whether the owner's data is correctly stored on the server. The auditor conducts the confirmation auditing phase as

1) The auditor runs the challenge algorithm Chall to generate the challenge C for all the data blocks in the data component and sends the C = $(\{i, v_i\}_{i \in Q}, R)$ to the server.

2) Upon receiving the challenge C from the auditor, the server runs the prove algorithm Prove to generate the proof P = (TP, DP) and sends it back to the auditor.

3) When the auditor receives the proof P from the server, it runs the verification algorithm Verify to check the correctness of P and extract the auditing result.

The auditor then sends the auditing result to the owner. If the result is true, the owner is convinced that its data is correctly stored on the server and it may choose to delete the local version of the data.

Phase 3: Sampling Auditing

The auditor will carry out the sampling auditing periodically by challenging a sample set of data blocks. The frequency of taking auditing operation depends on the service agreement between the data owner and the auditor (and also depends on how much trust the data owner has over the server).

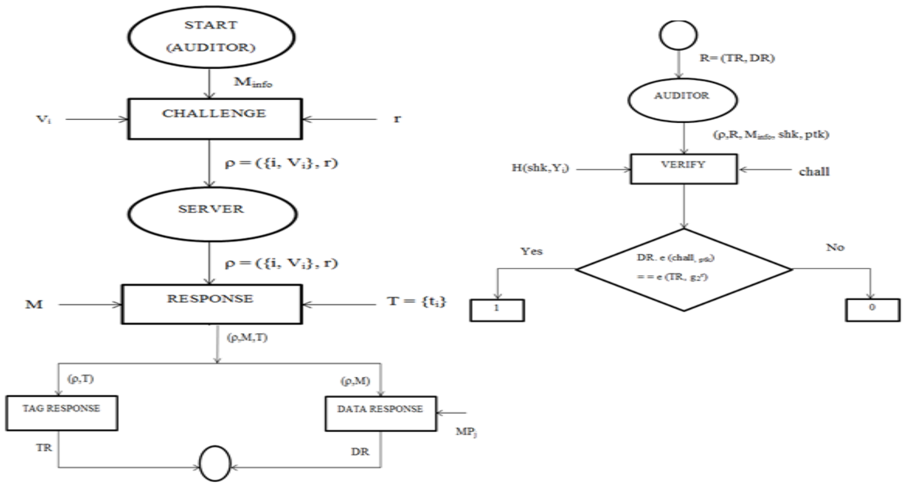

Figure 4. Flow Diagram for Confirmation and Sampling Auditing

Suppose each sector will be corrupted with a probability of ρ on the server. For a sampling auditing involved with t challenged data blocks, the probability of detection can be calculated as

$$Pr (t, s) = 1 - (1 - \rho)\ t \cdot s$$

That is, this t-block sampling auditing can detect any data corruption with a probability of Pr (t, s).

V. RESULT ANALYSIS

The simulation is done on a windows system with an i5 CPU at 2.00 GHz and 4GB RAM. The implementation uses PBC library version 0.5.14. We compare the computation load of the auditor, on the basis of file size in kb and time in minutes. In our proposed model the auditing time remains constant for varying file sizes whereas the auditing time varies with the file size for the existing system.

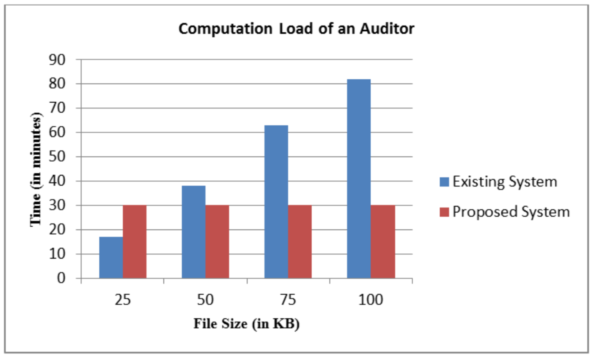

Figure 5. Computation Load of an Auditor

VI. CONCLUSION

Our auditing protocol ensures the data privacy by using the cryptography method and the bilinear property of the bilinear pairing. In our method, the server computes the proof as an intermediate value of the verification, such that the auditor can directly use this intermediate value to verify the correctness of the data. Therefore, the computation load of an auditor was greatly reduced which intern improves the auditing performance.

VII. FUTURE ENHANCEMENTS

To extend our auditing protocol to support the data dynamic operation and batch auditing for multiple owners as well as multiple clouds. Our multi-cloud batch auditing does not require any additional trusted organizer to send the commitment to the auditor. The previous work doesn’t support the batch auditing for multiple owners. Since the parameters for generating the data tags used by each owner are different and thus auditor cannot combine the data tags from the multiple owners to conduct the batch auditing. Our protocol could combine these auditing requests together and only conduct the batch auditing for multiple owners simultaneously. So that the computation cost of the auditor is reduced.

REFERENCES

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z.Peterson, and D.Song, Provable Data Possession at Untrusted Stores,*Proc. 14th ACM Conference on Computer and Communications Security (CCS’07)*, 598-609 (2007).

[2] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, Scalable and Efficient Provable Data Possession, *Proc.IEEE INFOCOM*, 954-962 (2009).

[3] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, Aggregate and Verifiably Encrypted Signatures from Bilinear Maps, *Proc. 22nd Int’l Conf. Theory and Applications of Cryptographic techniques (Eurocrypt ’03)*, 416-432 (2003).

[4] K.D. Bowers, A. Juels, and A. Oprea, Proofs of Retrievability: Theory and Implementation, *Report 2008/175, Cryptology ePrint Archive*, (2008).

[5] E.C. Chang and J. Xu, Remote Integrity Check with Dishonest Storage Server, *Proc. 13th European Symp. Research in Computer Security (ESORICS’08)*, 223-237 (2008).

[6] Y. Deswarte, J. Quisquater, and A. Saidane, Remote integrity checking, *The Sixth Working Conference on Integrity and Internal Control in Information Systems (IICIS).Springer Netherlands*, (2004).

[7] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, Dynamic Provable Data Possession, *Proc. 16th ACM Conference on Computer and Communications Security (CCS’09)*, (2009).

[8] A. Juels and B.S. Kaliski Jr., Pors: Proofs of Retrievability for Large Files, *Proc. 14th ACM Conference on Computer and Communications Security (CCS’07)*, 584-597 (2007).

[9] R.C. Merkle, Protocols for Public Key Cryptosystems, *Proc. IEEE Symp. Security and Privacy*, 122-133 (1980).

[10] A. Oprea, M.K. Reiter, and K. Yang, Space-Efficient Block Storage Integrity, *Proc. 12th Ann. Network and Distributed System Security Symp. (NDSS ’05),* (2005).

[11] H. Shacham and B. Waters, Compact Proofs of Retrievability, *Proc. 14th Int’l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT ’08)*, 90-107 (2008).

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

[13] Q. Wang, K. Ren, W. Lou, and Y. Zhang, Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance, *Proc.IEEE INFOCOM*, 954-962 (2009).

[14] C. Wang, Q. Wang, K. Ren, and W. Lou, Ensuring Data Storage Security in Cloud Computing, *Proc. 17th Int’l Workshop Quality of Service (IWQoS ’09),* (2009).

[15] XiaohuaJia and Kan Yang, An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing, *IEEE Transactions on Parallel and Distributed Systems*, 1–11 (2012).

[16] Y. Zhu, H. Hu, G. Ahn, and M. Yu, Cooperative provable data possession for integrity verification in multi-cloud storage, *IEEE Transactions on Parallel and Distributed Systems*, 2231-2244 (2012).

[17] Y. Zhu, H. Wang, Z. Hu, G.J. Ahn, H. Hu, and S. S. Yau, Dynamic audit services for outsourced storages in clouds, *IEEE Transactions on Services Computing*, 227-237 (2013).