Cloud Auditing: Privacy Preserving using Fully Homomorphic Encryption in TPA

Shivani Gambhir
University of Petroleum and Energy Studies, India

Ajay Rawat
University of Petroleum and Energy Studies, India

Rama Sushil
DIT University, India

ABSTRACT
Cloud computing is the future of the next generation architecture of IT solutions. Cloud provides computing resources on subscription basis over the internet. The Cloud data storage network includes a Third Party Auditor which has the power and capabilities that a client does not have. It is a trusted entity that has the access to, other than cloud and check on the exposed risk involved in cloud storage data on behalf of the client. In this paper, the problem of data security and integrity has been presented. Also, a scheme to provide maximum data integrity. In proposed scheme, existing fully homomorphic encryption is integrated with TPA auditing system is proposed. This scheme can audit data integrity without decrypting it.

Keywords
TPA, Cloud Auditing, Data security and Integrity, Fully Homomorphic Encryption

1. INTRODUCTION
Cloud is an emerging trend and envisioned as a next generation of IT. It defines a framework to deliver IT as a service in most efficient and the fastest way possible, without a need to actually own the resource. It provides level of transparency and monitoring which was not possible in earlier computing paradigms. According to the definition provided by the NIST, "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort and service provider interaction"[9]. The services of Cloud Computing are broadly divided into three categories: Infrastructure-as-a-Service (IaaS) provides infrastructure services to the end user on a subscription basis), Platform-as-a-Service (PaaS) provides platform services to the end user on a subscription basis and Software-as-a-Service (SaaS) provides services to the end user on a subscription basis). Cloud computing is thus being used significantly in IT industry [11]. However, data security is a major concern in adaption of cloud. Client customer does not have full access to outsourced data; it have major concern of ensuring that provider has taken security measures to protect and store data safely. Auditing is thus, employed as a verification tool. It is a process where tracing and logging of data is significant and is used for analyzing and validating security measures to achieve security in all area. Even after applying auditing process there are some loopholes like, data can still be revealed to outsider via auditor for their own benefits [8]. User does not want to add more vulnerability to data leakage and stored data by giving authority to auditor. Many schemes have been introduced to address this problem. We are providing a scheme which could audit data integrity without decrypting it.

The rest of the paper is organized as follows: In section 2 Cloud system model is described. Section 3 gives an overview of TPA role with some basic definition and background. Section 4 presents the basic scheme in use. In section 5 and 6 we described existing work and its problem statement respectively. Then there is preliminaries for the proposed scheme in section 7 and in section 8 our proposed scheme. Finally the future scope and conclusion of the paper is given in section 9 and 10 respectively.

2. CLOUD SYSTEM MODEL
Whole system of cloud architecture can be divided into three significant components:

1) Client: An entity, which contains large data and data files that are to be stored in the cloud for the computation and monitoring purpose. Client totally relies on the cloud provider for security of their data and they can be either individual consumers or organization.

2) Cloud Services Provider (CSP): It is an entity, which stores and manages all the data stored by the client. The Cloud storage service provider makes all the computation resources available to manage the data files.

3) Third Party Auditor (TPA): It is an entity, which has power and capabilities that a client does not have and a trusted entity that has the access to other than cloud to check on the exposed risk of cloud storage data on behalf of the client [7,3]. In the following section we present the details of TPA and other related preliminaries and background.

3. THIRD PARTY AUDITOR
As Cloud Service Provider belongs to separate entity, the stored data are at stake. The data outsourcing takes away client's full control over data. As a result the data integrity is hard to check. First, the correctness of the data is being put at a risk since cloud has internal and external threats. Second, there may be some motivations or personal benefits which can make CSP to behave unfaithful e.g. Byzantine failure where CSP can hide errors from the client for own benefits [5]. Another and more serious issues can be deliberately deleting or neglecting data which are rarely accessed by clients, just to save money and storage. Thus, TPA was introduced to cloud system architecture. TPA reduces the burden of Client for managing their data and ensures that the data in cloud is intact and data integrity is maintained. TPA not only helps clients for securing their data, but is also beneficial for the cloud service provider to maintain and increase standards of their platforms and to gain trust over the cloud by consumers [2].

In other words, providing public auditing services plays a substantial role for this aborning cloud economy to become fully established; users will be needing ways to assess risk and gain trust in the cloud [1].

Public auditability allows TPA to check on correctness of the data stored in cloud on client's behalf. But this scheme does

not provide privacy and protection to data. TPA has potential to reveal client's vital data to outsider. Thus, the clients don't want to give authority to TPA for creating new issues of vulnerability, related to data leakage and data storage security [7,8,1]. Some important programs used in TPA are discussed below [10, 6].

4. THE BASIC SCHEME

Before going to proposed schemes we study two classes of algorithm that is being used in Third Party Auditing. The first one is a MAC-based solution which goes through undesirable systematic demerits– bounded usage and state full verification, which poses additional online burden to client, in a public auditing setting. It shows that the auditing problem is not easy to solve even with TPA. The second system based on RSA algorithm, which covers many recent proofs of storage systems [1]

4.1 MAC Algorithm

There are many possible ways to use Message Authentication Code [13].One of them have been discussed here in this section. Data blocks mi (i € 1…n) are being uploaded with their specific per-computed (MACs) i= MACsk (i||mi) to the server and thus, correspondingly generate secret key sk, that are send to TPA. Later, TPA can check correctness via retrieving random blocks and verifying it by sk. There are certain drawbacks to this system. The TPA has to maintain knowledge about MACs key generated by every data block. The computation complexity and high communication makes it difficult to execute. It is difficult for cloud server to reveal a fresh MAC key for every comparison when TPA reveals its secret key. Once all possible secret key are exhausted, cloud server need to retrieve data from server to re- compute and re-publish mac keys again.

4.2 RSA Algorithm

RSA is a Partial homomorphic encryption, with multiplicative encryption technique.

In this algorithm, TPA selects two prime no's p1 and q1 thus values are computed [12]

$n1 = p1 * q1$

$fn1 = (p1-1) * (q1-1)$

Then, the public key $\alpha 1$ is selected. So, the Private Key of the TPA is:

$\beta 1 = (1/\alpha 1) \% fn1$

Similarly, the client selects prime numbers p2 and q2 with these the private key and the public key of the client is calculated as:

$n2 = p2 * q2$

$fn2 = (p2-1) * (q2-1)$

The public key of the client is declared as $\alpha 2$. So, the Private Key of the client: $\beta 2 = (1/\alpha 2) \% fn2$

Now, Key set of TPA is: $\{\alpha 1, n1\}, \{\beta 1, n1\}$

Key set of client is: $\{\alpha 2, n2\}, \{\beta 2, n2\}$

Thus, with the help of keys, data is being encrypted. But this algorithm still has some security problems; if attackers could interrupt two ciphers which are encrypted by same private key, it is then become easy for hacker to decrypt all messages exchanged between the server and it's client and it is only because the fully homomorphic encryption is multiplicative, i.e. the product of the ciphers equal the ciphers of the product [4].This scheme could not solve all the problems related to the unauthorised data leakage. It just reduces complex key management domain.

5. EXISTING WORK

Privacy-Preserving Public Auditing for Secure Cloud Storage:

It motivates the correctness of the cloud data can be verified by the TPA without retrieving the copy of the data. To achieve this, homomorphic linear authenticator with random masking technique is used. This technique, does not allow the TPA to have all the necessary information to build upon correct group of linear equation and therefore TPA cannot derive the user's data[1]

The linear combination of sampled blocks in the server's response is masked with randomness generated by server known as challenges. The user can ask the server to compute challenge for various inputs, and the server uses secret key to derive a short and efficiently verifiable proof that certifies correctness of the computation [9].

In this model, the client first needs to send some challenges c=chall(x) to the server and proof Ψ is computed in response to c.
Consider outsourcing t CD(.) that has the data D, gets input a program P, and outputs CD(P) = P(D). Then, we can think of the pre-processing of CD as creating an authentication tag σ for the data D.

Later, the user can take a program P, create a challenge c = chall (P), and get back a short tag Ψ that authenticates y = CD(P) = P(D)[6].

6. THE PROBLEM STATEMENT

Some identified problems in existing auditing systems are discussed below.

Existing systems require round of interaction; User first has to create challenge and only then server can authenticate output with respect to challenge [6].
The above delegation allow anyone to evaluate chosen encrypted data and non-interactively authenticate the output user needs to outsource all of data in one shot and stores some small secret state associated with the data to verify computation. Thus data can be easily predicted if decrypted by other than TPA

The schemes require a prior bounded computation by some fixed polynomial chosen during authentication. Furthermore, the complexity of authentication is proportional to the polynomial used. Thus, it increases complexity of algorithm.

Here we propose new concept of integration of fully homomorphic encryption [10] with TPA, which resolves the above mentioned problem.

7. DEFINITIONS AND PRELIMINARY BACKGROUND

In this section, we propose highly privacy-preserving public auditing it uses concept of fully homomorphic encryption. This scheme, not only check the data integrity but also keep the data secure even from the third auditor [6]. Following are some important terminology used in the scheme:
Homomorphic Authenticator: It consists of probabilistic-polynomial time algorithms given by Gentry Cargis in his paper[10]. Following functions are integral part of Homomorphic authenticator.

KeyGen(1n) →(evk,sk) : Takes input string and initializing secret key "sk" for it and an evaluation key "evk".
Authsk(b, τ) →σ: It creates a tag σ that authenticates the bit b ∈ {0, 1} of sk under the label τ ∈ {0,1}*.
Evalevk(f, σ) → Ψ: The deterministic evaluation procedure takes a vector of tags σ = (σ1,….., σk) and a circuit f: {0,1}k →{0, 1}. It outputs a tag Ψ. If each σi authenticates a bit bi as the output of some labelled-program Pi (possibly the identity program), then should authenticate b* = f(b,….., bk) as the output of the composed program P* = f(P1,….,Pk).
Versk(e,P,Ψ)→ {accept; reject}: The deterministic verification procedure uses the tag to check that e ∈ {0, 1} is the output of the program P on previously authenticated labeled data.
Fully homomorphic Encryption: A Fully Homomorphic (public-key) Encryption (FHE) scheme is also consist of probabilistic-polynomial time algorithms given by Gentry [9].HE = (HE.KeyGen; HE.Enc; HE.Dec; HE.Eval) defined as follows
.
HE.KeyGen (1n) → (pk, evk, sk): Outputs a public encryption key pk, a public evaluation key evk and a secret decryption key sk.
HE.Encpk (b) → c: Encrypts a bit b ∈ {0,
1} under public key pk. Outputs cipher text c.
HE.Decsk(c) → b: Decrypts ciphertext c using sk to a plaintext bit b ∈ {0, 1}.
HE.Evalevk (g;c1,….,ct) → c*: The deterministic evaluation algorithm takes the evaluation key evk, a boolean circuit g : {0; 1}t → {0; 1}, and a set of t ciphertexts c1,….,ct. It outputs the result cipher text c*.

8. SCHEME DETAILS

Let HE = (HE.KeyGen, HE.Enc, HE.Dec,HE.Eval) be fully homomorphic encryption scheme. Let {fK:{0, 1}*→ {0; 1}r(n)}K∈ {0;1}n be a pseudo-random function PRF family. Let H be a family of collision-resistant hash functions (CRHF) H :{0; 1}*→{0, 1}m(n).

User first chooses a public/secret key for an FHE and a pseudo-random function. To authenticate a bit 'b' under a Labelled Program τ creates ciphertexts. User chooses ciphertexts and encrypt as random encryptions of bit b being authenticated. Given some program P with t inputs and authentication tags, we can homomorphically derive an authentication tag for output.

We derive each cipher text by homomorphically evaluating the program P over t cipher texts that lie in position within the tags. In particular, we set Eval. We assume that evaluation procedure for the FHE scheme is deterministic so that results are reproducible.

User can verify to certify, that y is the output of the labelled program P. For indices, user can re-compute the pseudo-random input cipher texts using the PRF, without knowing actual input bits. User then computes Eval and checks that the ciphertexts in tag were computed correctly. If this is the case, and all of other ciphertexts decrypt to claimed bit y, then user accepts.

We define the authenticator scheme = (KeyGen, Auth, Eval, Ver) as follows:

KeyGen(1n): Choose a PRF key K {0; 1}n and a CRHF H ← H. Choose an encryption key (pk, evk', sk') HE.KeyGen(1n). Select a subset S ⊆ [n] by choosing whether to add each index i ∈ [n] to the set S independently with probability ½.Output evk = (evk',H),sk =(pk,evk',H,sk',K, S).

Authsk(b,τ): Given b ∈{0; 1} and τ∈{0; 1}* do the following:
1. Choose random coins rand1,...,randn by setting randi = fk((τ; i)). Set υ :=fK(τ).
2. Create n ciphertexts c1, . ..,cn as follows. For i ∈ [n] n S, choose ci = HE.Encpk(b; randi) asencryptions of the bit b. For i ∈ S, choose ci = HE.Encpk(0,randi) as encryption of 0.
3. Output _ = (c1,.. .,cn; υ).

Evalevk(g,σ): Given σ = (σ1,…..,σt), parse each σj = (c1;j ,…..,cn;j; υj).
For each i ∈ [n], compute c*i = HE:Evalevk0(g; ci;1 ,….., ci;t).
Compute υ* = gH(υ1,…,υt) to be the output of the hash-tree of g evaluated at υ1,….,υt.
Output Ψ = (c*1,….., c* n; υ*).

Versk(e;P;Ψ): Parse P = (g; τ1,….,τt) and Ψ = (c*1,…..,c*n,υ*).
1. Compute υ1 := fK(τ1),…..,υt = fK(τt). If υ≠gH (υ1,…..,υt) then output reject.
2. For i ∈ [n], j ∈ [t], compute randi;j := fK((τj,i)) and, for i ∈ S, set ci;j := HE.Encpk(0,randi;j). For each i ∈ S, evaluate c'i := HE.Evalevk'(g; ci;1; : : : ; ci;t) and if c'i≠ c*i, **output reject.**
3. For each i ∈[n] \ S, evaluate c'I = HE. Encpk (e;REvalevk'(g,randi,1,…., randi,t)) and if c'i≠c*i, **output reject.**

If the above doesn't reject, **output accept**. It is also used to hide difference between data-independent pseudo random cipher texts which allow user to check upon the computation and the output is right or not.

Also for any given sk, it shall possible to decrypt the correct bit b from the tag authenticating it, but given only the evaluation key evk, the tags will not reveal any information about the authenticated bits. Thus we can say:

Versk(P;Ψ) →{0,1, reject} as follows: run **Versk(e;P;Ψ)** for both choices of e ∈ {0; 1} and, if exactly one of the runs is accepting, return the corresponding e, else **return reject.**

Even if the attacker gets evk and access to the authentication oracle Authsk(b, τ) , if he later sees a tag σ ←Authsk(b, τ) for a fresh label τ , he cannot distinguish between the cases b = 0 and b = 1.

9. FUTURE SCOPE

In proposed scheme fully homomorphic authenticator is being included in TPA. This scheme is being theoretically approached. Thus in future we intended to add experimental analysis and some derive some more experimental validation and conclusion.

10. CONCLUSION

In this paper we propose fully homomorphic encryption auditing system for the data storage security in the cloud computing. We have integrated the fully homomorphic encryption in TPA auditing system. Proposed scheme of fully homomorphic provides auditing technique which not only preserves privacy but also provide authenticator that allow an unbounded number of verification. In future we intend to verify its approach practically and to add experimental analysis to derive some conclusion from it.

11. REFRENCES

[1] Wang, Cong, et al. "Privacy-preserving public auditing for secure cloud storage." (2013): 1-1.

[2] Xu, Jia. Auditing the auditor: secure delegation of auditing operation over cloud storage.IACR Cryptology ePrint Archive 2011, 2011.

[3] Chuang, I-Hsun, et al. "An effective privacy protection scheme for cloud computing." Advanced Communication Technology (ICACT), 2011 13th International Conference on.IEEE, 2011.

[4] TEBAA, Maha, Saïd EL HAJJI, and Abdellatif EL GHAZI. "Homomorphic Encryption Applied to the Cloud Computing Security." Proceedings of the World Congress on Engineering.Vol. 1. 2012.

[5] Sowparnika, Miss M., and R. Dheenadayalu. "Improving data integrity on cloud storage services."

[6] Gennaro, Rosario, and Daniel Wichs. Fully homomorphic message authenticators. May 2012. Cryptology eprint 290, 2012.

[7] Wang, Qian, et al. "Enabling public auditability and data dynamics for storage security in cloud computing." Parallel and Distributed Systems, IEEE Transactions on 22.5 (2011): 847-859.

[8] Hamlen, Kevin, et al. "Security issues for cloud computing." International Journal of Information Security and Privacy (IJISP) 4.2 (2010): 36-48.

[9] Mell, Peter, and Timothy Grance. "The NIST definition of cloud computing (draft)." NIST special publication 800 (2011): 145.

[10] Gentry, Craig. A fully homomorphic encryption scheme.Diss. Stanford University, 2009.

[11] AS, ANUPRIYA, S. Ananthi, and S. Karthik. "TPA BASED CLOUD STORAGE SECURITY TECHNIQUES." International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 1.8 (2012): pp-194.

[12] Rivest, R.; A. Shamir; L. Adleman (1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM 21 (2): 120126.doi:10.1145/359340.359342

[13] Kaliski, Burt, and Matt Robshaw. "Message authentication with MD5."CryptoBytes (RSA Labs Technical Newsletter) 1.1 (1995).