Collaborative Approach for Data Integrity Verification in Cloud Computing

Rajat Saxena and Somnath Dey

Department of Computer Science and Engineering,
Indian Institute of Technology Indore, India
{rajat.saxena,somnathd}@iiti.ac.in

Abstract. High security is one of leading restriction for shining up bright eras and vision of Cloud computing. In latest trend of Cloud, all the sensitive applications and data are moved towards cloud infrastructure and data center which run on virtual computing resources in the form of virtual machine. The large scale usage of virtualization to achieve cloud infrastructure brings additional security burden for tenants of a public cloud service. In this paper, we primarily aim to achieve better data integrity verification technique and help users to utilize Data as a Service (Daas) in Cloud computing. The experimental results are included in order to show the effectiveness of the proposed method for data integrity verification.

Keywords: Proof of Retrievability (PoR), Provable Data Possession (PDP), Third Party Auditing, Algebraic Signature, Homomorphic TAG.

1 Introduction

Cloud computing is defined as services and applications that are enforced on a distributed network using virtual resources and accessed by common networking standards and Internet protocols. It is distinguished from the traditional system in this manner that resources are virtual and limitless and implementation details of the physical systems on which software runs are abstracted from the user.

In Cloud, the complexity of security is greatly increased in comparison with traditional systems. The reason for this is that data is stored and operated in multi-tenant systems which are distributed over a wider area and shared by unrelated users. In addition, maintenance of security audit logs may be difficult or impossible for a user that has limited resources. Thus, the role of cloud service providers is important that it must devote proper security measures and resources to maintain privacy preservation and data integrity. It is possible that cloud provider may delete or sell some non operational data for its greed or profit that is not used for a long time. It is also possible that an adversary may exploit this data by performing various attacks. The customer also must ensure that the provider had taken the proper security measures to protect their information.

There are a number of security threats associates with utility of DaaS in cloud computing. New security challenges introduced by storing data in the cloud are following.

G. Martínez Pérez et al. (Eds.): SNDS 2014, CCIS 420, pp. 1–15, 2014.

1. **Data integrity:** when data stores on cloud storage servers, anyone from any location can access this data. Cloud is unable to differentiate between sensitive data from common data thus it enables anyone to access sensitive data. Tampering the sensitive data causes the data integrity issue. Thus, there is lack of data integrity in cloud computing.
2. **Data theft or loss:** The cloud servers are distrusted in terms of both security and reliability, which means that data may lost or modified maliciously or accidentally. Administrative errors may cause data loss (e.g. backup and restore, data migration, and changing memberships in point to point systems). Additionally, adversaries may initiate attacks for taking advantage of control loss over data by data owners.
3. **Privacy issues:** As most of the servers are external; the vendor should make sure who is accessing the data and who is maintaining the server. The cloud vendor also make sure that the customer personal information is well secured from other operators. This enables vendor to protect the personal information of customers.
4. **Infected application:** Vendor should have the complete access to the server for monitoring and maintenance. This prevents any malicious user from uploading any infected application on Cloud which will severely affect the customer.
5. **Loss of physical control:** Cloud customers have their data and program outsourced to cloud servers. As a result, owners lose direct control on the data sets and programs. Loss of physical control means that customers are unable to resist certain attacks and accidents.
6. **Data location:** In cloud environment data location are not transparent from customers. The customer doesn't know where his data is located and the Vendor does not reveal where all data is stored. The data won't even be in the same country of the customer, it might be located anywhere in the world. It might raise SLA and legal issue.
7. **Cross-VM attack via side channels:** Cross-VM attack exploits multi-tenant nature of cloud. In Multi-tenent environment, VMs belonging to different customers may co-reside on the same physical machine. Cross-VM attack may corrupt whole file structure and leak information from one VM to another VM.

We concentrate on the data integrity verification issue. It is one of the biggest concerns with cloud data storage at untrusted servers because it may be possible that cloud user or /and cloud provider may be malicious. It is also an interesting problem that how cloud users and cloud providers have trusted to each other for storing the data and how privacy of the cloud users should be maintained. One solution of this problem is to perform encryption and decryption operations but it involves with computational and operational overheads. Another solution of this problem is to perform data auditing.

Organization. The rest of the paper is organized as follows: In the section 2, we provide literature survey on data auditing till current state-of-the-art work. In section 3, we describe our system model and data integrity verification scheme.

Then, we provide security and performance analysis in section 4. Finally, we conclude in section 5.

2 Literature Survey

Data auditing is a periodic event to assess quality or utility of data for a specific purpose like to evaluate security, data integrity, privacy preservation and computational accuracy. Data auditing could be a primary source for shielding corporate data assets against potential risk and loss. Data auditing relies on a registry that could be a storage space for information and data assets. during data auditing, the creation, origin or format of data may be reviewed to assess its utility and value.

There are two type of approach for data integrity auditing: Probabilistic auditing in which the blocks are randomly selected by using the probabilistic checking method and Deterministic auditing in which auditors checks the integrity of all data blocks. Traditional systems for data auditing are PDP and PoR schemes. Both of these schemes are based on the facts that client directly communicates with the data storage to produce proof of access, retrieve and possession of the data. The difference between PDP and POR techniques is that PDP techniques only produce a proof for recoverable data possession but POR schemes checks the possession of data and it can recover data in case of data access failure or data loss. usually a PDP scheme can be transformed into POR scheme by adding erasure or error correcting codes.

The PDP techniques [1], [2], [3], [4], [5] generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. PDP techniques have two parts of action : First, the client (verifier) allows to preprocesses the data, keep a small amount of metadata and then sends whole data to an untrusted data storage server (prover) for storing. later, Client (verifier) allows to verify with the help of metadata that the data storage server still possesses the clients original data and stored data has not been tampered or deleted. In PDP techniques, the client maintains a constant amount of metadata to verify the proof. The challenge/response protocol transmits a low, constant amount of data that minimizes network communication. Thus, the PDP schemes for remote data checking support large data sets in widely distributed storage systems. Table 1 shows the comparative analysis of the different PDP schemes.

PoR schemes [6], [7], [8], [9], [10] have two parts of action : First, the client (verifier) allows to store a file on an untrusted data storage server or prover. later, the client run data audit proof algorithm. This proof help provers to ensure that it still possesses the clients data file and client can recover the entire file. In this schemes, an encrypted file randomly embeds a set of randomly-valued check blocks or Sentinels. The use of sentinel for data auditing minimizes the client and server storage. It also minimizes the communication complexity of the audit and the number of file-blocks accessed by server during audit. An auspiciously executed POR scheme encourages verifiers that the provers presents a protocol

Table 1. Comparison of different PDP Schemes

Properties	**PDP [1]**	**S-PDP [2]**	**E-PDP [3]**	**D-PDP[4]**	**C-DPDP[5]**
Primitives	Homomorphic Verifiable Tags (HVTs)	Symmetric key Cryptography	Asymmetric key , cryptography (RSA Modules)	Rank-based Authenticated Dictionary and Skip List, RSA Tree	Algebraic Signature
Type of guarantee	Probabilistic	Probabilistic	Probabilistic	Probabilistic	Probabilistic
Public Verifiability	Yes	No	No	No	Yes
With the help of TPA	No	No	No	No	No
Data dynamics	Append only(Static)	Yes	No	Yes	Yes
Privacy preserving	No	No	Not Supported	Not Supported	No
Support for sampling	Yes	Yes	No	Yes	Yes
Probability of detection	$[1\text{-}(1-p)^c]$	$[1\text{-}(1-p)^c]$	$[1\text{-}(1-p)^{c*s}]$	$[1\text{-}(1-p)^c]$	$[1\text{-}(1-p)^{c*s}]$

Table 2. Comparison of different PoR Schemes

Properties	**PoR [6]**	**C-PoR [7]**	**PoR-HA [8]**	**PoR-TI [9]**	**HAIL [10]**
Primitives	Error Correcting Code,Symmetric Key Cryptography, Sentinel Creation and Permutation	BLS Signature, Pseudorandom Functions	Error Correcting Codes,Reed -Solomon Codes Hitting Sampler	Adversarial Error Correcting Codes	Integrity Protected Error Correcting Universal Hash Function, MAC
Type of guarantee	Probabilistic	Probabilistic	Probabilistic	Probabilistic / Deterministic	Probabilistic / Deterministic
Public Verifiability	No	Yes	Yes	Yes	Yes
With the help of TPA	No	No	No	No	No
Data dynamics	No	No	Append only	Append only	Yes
Privacy preserving	No	No	No	No	Yes
Support for sampling	Yes	Yes	Yes	Yes	Yes
Probability of detection	$[1\text{-}(1-p)^c]$	$[1\text{-}(1-p)^{c*s}]$	$[1\text{-}(1-p)^c]$	$[1\text{-}(1-p)^c]$	$[1\text{-}(1-p)^{c*s}]$

Table 3. Comparison of different Data Auditing Techniques with TPA

Properties	**Wang et al [11]**	**Wang et al [12]**	**Hao et al [13]**	**Co-PDP[14]**
Primitives	Bilinear Map, MAC, Homomorphic Authenticator	Merkle Hash Tree, Aggregate Signature	RSA based Bilinear Homomorphic Verifiable Tags	Homomorphic Verifiable Hash Index Hierarchy
Type of guarantee	Probabilistic	Probabilistic	Deterministic	Probabilistic
Public Verifiability	Yes	Yes	Yes	Yes
With the help of TPA	Yes	Yes	Yes	Yes
Data dynamics	Yes	Yes	Yes	Yes
Privacy preserving	Yes	Yes	Yes	Yes
Support for sampling	Yes	Yes	No	Yes
Probability of detection	$[1\text{-}(1-p)^c]$	$[1\text{-}(1-p)^{c*s}]$	$[1\text{-}(1-p)^{c*s}]$	Z^*

1. n is the block number, c is the sampling block number and s is the numbers of sectors in blocks. p is the probability of block corruption in a cloud server and P_k is the probability of k^{th} cloud server in a multi-cloud.
2.

$$Z^* = [1 - \prod p_k \epsilon p(1-p_k)^{r_k * c * s}] \tag{1}$$

interface through which the verifiers can collectively retrieve the file. Table 2 shows the comparative analysis of different PoR schemes.

The issues with PoR and PDP schemes are: These schemes focus on only static data. These schemes apply for only encrypt files that allows a limited number of queries. There is a tradeoff between privacy preservation and dynamic data operations thus some schemes do not preserve privacy. They are complex and computation intensive and have to be done at the user end. None of this scheme consider batch auditing process. The effectiveness of these schemes primarily rests on the preprocessing steps that the user conducts before out-source the data file. This introduces significant computational and communication complexity overhead. These techniques provide tradeoff between storage overhead and cost of communication thus some of this techniques store less storage with high cost.

In cloud scenario, the users might have limited CPU, battery power and communication resource constraints. so, they are not capable to perform data audits. Instead of them, Third Party Auditors (TPA) are responsible for data audits. A trusted TPA has certain special expertise and technical capabilities, which the clients do not have.

The schemes [11], [12], [13], [14] assigns auditing work to single TPA. Trusted Third Party (TTP) involves an independent outside trusted and authenticate entity to conduct data audit. External trusted third-party audit mechanism is important and indispensable for the protection of data security and the reliability of services in cloud environment. TPA should be able to efficiently audit the cloud data storage without demanding the local copy of data and introduce no additional on-line burden to the cloud user. The third party auditing process should bring in no new vulnerability towards user data privacy.

Table 3 shows comparative analysis of data auditing schemes that has single TPA. In these schemes, single TPA cannot handle SLA and legal issues for data possession and prone to single-point failure. For these schemes, error localization is very difficult to find. All the above schemes provide only binary results about the storage status for identifying misbehaving server(s). none of these scheme support multiple TPAs for cross checks and cross authenticate the data integrity verification, privacy preservation and computation accuracy. There is a tradeoff between data dynamics, privacy preservation and public verifiability in these schemes. TPA may simultaneously handle various audit sessions from different users for their outsourced data files by multi-user setting during efficient auditing process.

To address the above problems, we propose multiple TPA system in which each TPA may simultaneously handle various audit sessions from different users for their outsourced data files. our work utilizes the algebraic signature and homomorphic tag for auditing. Algebraic signature use symmetric key techniques to enhance efficiency. The running of algebraic signature can achieve high speed from tens to hundreds of megabytes per second. An algebraic signature allows challenger to verify data integrity by comparing only the responds returned by the storage server. for this challenger does not need whole original data for verification. Algebraic signature use only small challenges and responses. TPA group need

to store only two secret keys and several random numbers. This makes task of TPA group easy and computation intensive. The efficiency of algebraic schemes permits the construction of large-scale distributed storage systems in which large amounts of storage can be verified with maximal efficiency and minimal overhead. The aggregation and algebraic properties of the algebraic signature provide extra benefit for batch auditing in our design.

By integrating the homomorphic tag with random masking, our protocol guarantees that TPA could not learn any knowledge about the data content stored in the cloud server during the efficient auditing process. Specifically, our contribution in this work can be summarized as the following three aspects:

1. We motivate the public auditing system of data storage security in Cloud Computing and provide a privacy-preserving auditing protocol with multiple TPA.
2. To the best of our knowledge, our scheme is the first to support scalable and efficient public auditing with multiple TPA in the Cloud Computing. In particular, our scheme achieves batch auditing in which each TPA may simultaneously handle various audit sessions from different users for their outsourced data files.
3. We prove the security and justify the performance of our proposed schemes through concrete experiments and comparisons with the state-of-the-art.

3 The Proposed Scheme

In this section, we present our security protocols for cloud data storage service with the aforementioned research goals in mind. first we establish notation related our scheme, then we explain details about algebraic signature. Thereafter, we discuss our system model that subsequently represent our scheme.

3.1 Notation and Preliminaries

1. AS_g (•): denote the Algebraic signature.
2. f(•) is a pseudo-random function (PRF) which maps as follow $\mathrm{f} : \{0,1\}^k \times \{0,1\}^l \longrightarrow \{0,1\}^l$.
3. σ(•) is a pseudo-random permutation (PRP) which maps as follow $\sigma : \{0,1\}^k \times \{0,1.....n\} \longrightarrow \{0,1.....n\}$.
4. $E_{k_t}(\bullet)$ and $D_{k_t}(\bullet)$: denote the encryption and decryption algorithms.
5. L: the length of a bit string, with typical values L = 16 bits. Each data block will be divided into equal bit strings on which the algebraic signature is computed.
6. t : the number of verification.
7. R: the number of blocks required for each challenge.
8. k: the master key, which is used to compute the locations of data blocks to compute verifiable tags.

9. k_t: the tag encryption key, which is used to encrypt the verifiable tags.
10. r_1, r_2: random numbers chosen from the Galois field.
11. F = F [1], F [2] . . . F [n]: F denotes a file, and F [i] denote a data block of the file F.
12. T = $T_1, T_2 ... T_t : T$ denotes all block tags and T_i denotes one tag of T.

3.2 Algebraic Signature

Algebraic signature [15] of the file blocks which has composition of strings $s_0, s_1, ..., s_{n-1}$ is a kind of hash functions simply defined as follows

$$AS_g(s_0, s_1, ..., s_{n-1}) = \sum_{s=0}^{n-1} s_i.g^i \tag{2}$$

Algebraic signature [16] itself is a single string . The data compression rate of algebraic signature is the n $= \frac{F[i]}{L}$, where F [i] is size of a file block and L is length of string. For example, if the size of a file block F [i] is 1 KB and L = 64 bits, then corresponding algebraic signature is 64 bits and n = $\frac{F[i]}{L} = 16$, so the data compression rate of algebraic signature is 16.

Property: Taking the sum of signatures of some file blocks provides the equal result as taking the signature of sum of corresponding blocks.

$$AS_g(X) + AS_g(Y) = AS_g(X + Y) \tag{3}$$

Proof: The property of algebraic signature can be verified as follows.

$\Rightarrow$ $AS_g(X) + AS_g(Y)$.
$\Rightarrow$ $AS_g(x_0, x_1, ...x_{n-1}) + AS_g(y_0, y_1, ...y_{n-1})$.
$\Rightarrow$ $\sum_{i=0}^{n-1} x_i.g^i + \sum_{i=0}^{n-1} y_i.g^i$.
$\Rightarrow$ $\sum_{i=0}^{n-1} (x_i + y_i).g^i$.
$\Rightarrow$ $AS_g(X + Y)$.

We use property of a algebraic signature for tag generation (an algebraic signature of the sum of some file blocks) which can be calculated solely using the signatures of the file blocks.

3.3 System Model

We propose a distributed multiple third party data auditing technique.In this technique, Multiple TPA have shared the huge load responsibility of single TPA by load balancing. Figure 1, illustrates network architecture for cloud data storage, which incorporates our distributed multiple third party data auditing technique. This figure divides proposed scheme into three parts:

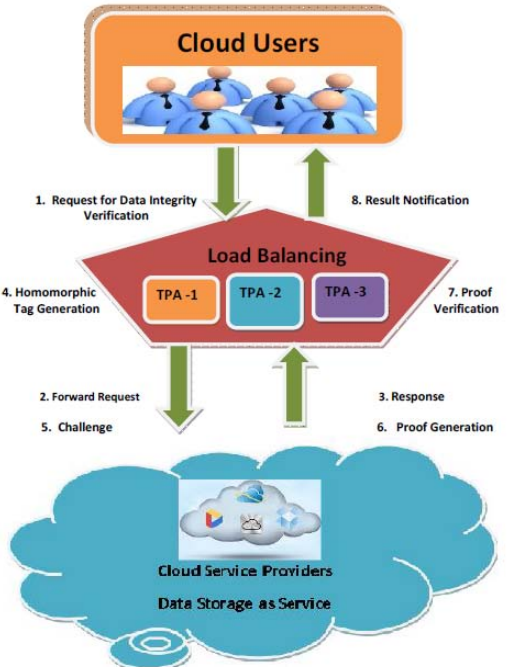

Fig. 1. System Model

1. Cloud Users: Any end user may interpreted as a cloud user. We assume that these cloud users have limited resources. Thus, Cloud user is not capable to perform computation intensive tasks such as data integrity and privacy preserving audits.

2. Multiple TPA: In this region, TPA is an authorized and authentic entity that is responsible for data integrity verification and privacy preservation. It also takes care for the SLA and legal issue related to data migration. We consider multiple TPA to achieving load balance and batch audits.

3. Cloud Service Provider: In this group, cloud service providers have established enough infrastructure and resources to provide data storage as a service for cloud customers. These resources may be distributed across the world.

3.4 Proposed Data Integrity Verification Scheme

We divide whole scheme in 8 parts and working of this parts among system model provides in Figure 1.

1. **Request for Data Integrity Verification:** During data audits, cloud users send request to the TPA group for verification of the data of file F. In TPA group one TPA can share and distribute this load with other TPA by load balancing and batch auditing.

2. **Forward Request:** TPA group use setup operation for generating some initialization parameters such as the master key k, Homomorphic Tag encryption key k_t and random numbers r_1, r_2. This initial parameters are common to all TPA. TPA group forwards request to the cloud service provider on the sample blocks of physical data centers for checking data integrity.

Algorithm 1. Setup operation

Input: $\{0,1\}^k$.
Output: Master key k , Homomorphic Tag Encryption Key k_t , Random numbers r_1, r_2 .

1: Master key k generates by $k \xleftarrow{R} \{0,1\}^k$.
2: Homomorphic Tag Encryption Key k_t generates by $k_t \xleftarrow{R} \{0,1\}^k$.
3: Random numbers r_1, r_2 generates by $r_1 \xleftarrow{R} \{0,1\}^k$ and $r_2 \xleftarrow{R} \{0,1\}^k$.

3. **Response:** Cloud service provider chooses some random sample blocks from the whole data base related to file F and responses to the TPA group with c blocks.
4. **Homomorphic Tag Generation:** Now, TPA group use homomorphic tag generation algorithm. Each TPA individually chooses random sample blocks $c_1, c_2, ...c_n$ from responded c blocks and compute algebraic signature sum for these blocks as the $AS_g(s_1)$, $AS_g(s_2)$, ... $AS_g(s_n)$. Through load balancing homomorphic tag generation process distribute among all TPA. TPA group sends entry (F,T) to the cloud service provider for storage.

Algorithm 2. Homomorphic TAG Generation

Input: Random sample blocks $c_1, c_2, ...c_n$ from responded c blocks .
Output: Entry {F , T}.

1: **if** the number of verification is t **then**
2: TPA x has compute t tags with this procedure.
for $0 < i \leqslant t$
$k_i = f_k(r_1 + i)$
$s_x = 0$
for $0 < j \leq c_x$
$l_j = \sigma_{k_i}(r_2 + j)$
$s_x = s_x + F[l_j]$
3: Compute $AS_g(s_x)$
4: $AS_g(S) = \sum_{x=1}^{n} AS_g(s_x)$, where n is the number of TPA's.
5: Homomorphic Verifiable Tag = $\partial_i = AS_g(S)$.
6: $T_i = E_{k_t}(\partial_i)$.
7: **end if**
8: TPA group send entry {F , T}, which corresponds to file F and all block tags T, to the cloud service provider for store.

5. **Challenge:** TPA group computes k_i by challenge operation for the i^{th} verification using the master key k, then sends the (r_2, k_i) to storage server.

Algorithm 3. Challenge operation

Input: Master Key k, Encryption function f_k and random number r_1.
Output: Entry {r_2, k_i } to the storage server.

1: **for** i^{th} verification, TPA group calculates **do**
2: $k_i = f_k(r_1 + i)$
3: **end for**
4: TPA group sends {r_2, k_i } to cloud service provider.

6. **Proof Generation:** In Proof Generation storage server computes the locations of the requested blocks using k_i, computes their sum F_i' and then returns to the TPA group (F_i' , T_i'), where T_i' is the homomorphic verifiable tag stored on cloud service provider corresponding to F_i'.

Algorithm 4. Proof Generation

Input: $F_i' = 0$, random number r_2.
Output: Cloud Service Provider return {F_i', T_i' } to TPA group.

1: $F_i' = 0$
2: **for** $0 < j \leq R$ **do**
3: $l_j = \sigma_{k_i}(r_2 + j)$
4: $F_i' = F_i' + F[l_j]$
5: **end for**
6: Cloud Service Provider return {F_i', T_i' } to TPA group.

7. **Proof Verification:** The TPA group decrypts T_i' using the tag decryption key k_t ,computes the algebraic signature of F_i' and then verify whether they are equal. If yes, it indicates that the integrity of file is maintained else the integrity of file is corrupted.

Algorithm 5. Proof Verification

Input: Decryption key k_t, Decryption function D_{k_t}
Output: Verification Result $AS_g(F_i') \stackrel{?}{=} \rho_i$

1: $\rho_i = D_{k_t}(T_i')$.
2: Verifies $AS_g(F_i') \stackrel{?}{=} \rho_i$.

8. **Result Notification:** TPA group notify result to the cloud user.

4 Analysis

In this section, we analyze the security strength of our scheme against server misbehavior and explain why challenge the random blocks can improve the security strength of our proposed scheme. We also provide performance analysis based on the obtained result of experiments.

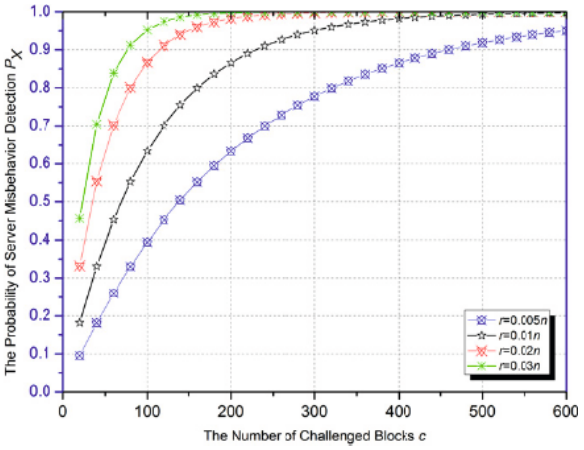

Fig. 2. The Probability of Server misbehavior detection

4.1 Security Analysis

Algebraic signature is ideally suited for use in verifying large amounts of cloud data stores in remote data centers because of their minimal network impact,reasonable computation loads and resistance to malicious modification. The use of algebraic signatures compresses file blocks into a very small entity that can change with little bit change in the block. for large bit string, Algebraic signature is cryptographically secure in comparison with hash functions such as MD5 and SHA1.

TPA group chooses c file blocks randomly from n blocks each time. This sampling incredibly reduces workload on the server, while still achieving server misbehavior detection with high provability. We assume that out of n blocks, the server deletes r blocks. X is a discrete random variable which is defined to the number of blocks chosen by TPA group that match the blocks deleted by the server. We compute P_X, the probability that at least one of the blocks picked by TPA group matches one of the blocks deleted by server, with following equation.

$$P_X = P\{X \geqslant 1\} = 1 - P\{X = 0\} = 1 - \{\frac{n-r}{n}.\frac{n-1-r}{n-1}.\frac{n-2-r}{n-2}....\frac{n-c+1-r}{n-c+1}\} \tag{4}$$

P_X indicates the probability of detection of server misbehaviour that depends on total number of file blocks n, deleted r blocks and challenged c blocks. if storage server deletes r blocks of the file, then the TPA group will detect server misbehavior after a challenge for c blocks. Figure 2 exhibit P_X for different values of r, c.

Surprisingly, the TPA group can detect server misbehaviour with a certain probability by challenge a constant amount of blocks, independently of the total number of file blocks: e.g., if r = 1% of n, then the client asks for 460 blocks and 300 blocks in order to achieve P_X of at least 99% and 95%, respectively. Thus, our scheme is probabilistic secure approach. Moreover, we can improve detection probability by performing the detection process more frequently and requesting more blocks for each challenge. Large enough bit string provides resistance from accidental collision of similar signatures. For an example, if we choose a 64 bits signature then collision probability will be 2^{-64}.

We choose 256 bits algebraic signature in our work with minimal collision probability of 2^{-256}. If the size of a file 1 GB, size of file block F [i] is 8 KB and length of bit string L = 256 bits, then corresponding algebraic signature is 256 bits and n = $\frac{F[i]}{L}$ = 256. Thus, our scheme provides data compression rate 256 for 1GB size file which is huge in cloud environment. This makes prediction task complex for a site that does not knows some secrets to generate a coherent set of signatures. The additional storage cost for 1 GB size file is = $\frac{size-of-file}{size-of-algebraic-signature}$ = 4 MB. Thus, additional storage overhead is only 4 MB for 1 GB size file.

4.2 Performance Analysis

From the performance point of view, we focuses on how frequently and efficiently user verify that storage server can faithfully store his data without retrieving it. In our scheme, the number of verification and the number of blocks required for each challenge can be set flexible according to user's requirement. If data will not be stored for a long time, user can be set small number of verification and blocks to further reduce the overload.

The experiment has run on two PCs configured with an Intel core i3-2330M 2.20 GHz and 2 GB RAM. We have configured Citrix Xen Server 6.1.0 on one PC that is used for file storage. The second PC used as a TPA group that audits the stored files on the behalf of cloud users. We observe that :

- In setup operation, TPA group generates some secret keys and random numbers.
- For Homomorphic TAG Generation, each TPA needs to perform t times PRF, c times PRP operations, c times sum, t times algebraic signature and symmetric encryption operations. If number of TPA is n, then TPA group needs to perform t*n times PRF and c*n times PRP operations, c*n times sum, t*n times algebraic signature and symmetric encryption operations.
- In challenge operation, TPA group only needs to transfer about 512 bits information for 256 bit secret keys.

– For proof generation, Cloud service provider needs to perform c times PRP and sum operations, and then it needs to transfer about 8 KB responding results (for the size block is 8KB).
– For proof verification, TPA group only needs one time decryption and comparison operations.

There only symmetric key encryption and decryption, sum, PRF and PRP operations are used. All operations are simple and efficient in computation. For our approach number of verification are infinite and server computation complexity, client computation complexity, communication complexity and client storage complexity are O(1). Thus, our approach can be utilized in cloud storage for very large data sets.

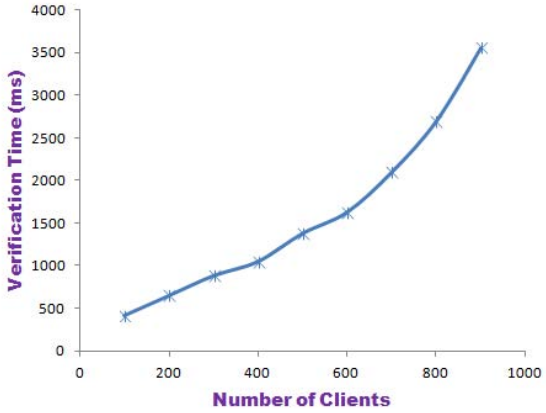

Fig. 3. The Verification delay for multiple clients

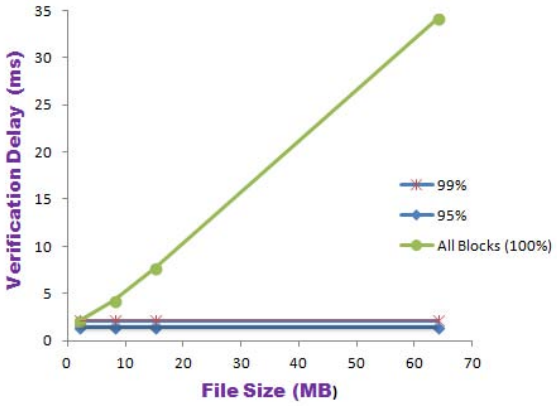

Fig. 4. The Verification overhead at multiple confidence level

We evaluate the verification delay for multiple clients in Figure 3. For 100 clients, the verification delay is about 180 ms. The delay increases quickly, when we increase the number of clients. When it reaches up to 1000 clients, the verification delay is about 3560 ms.

We measured the verification overhead for detecting 1% missing or faulty data at 100%, 99% and 95% confidence in Figure 4. Examining time for all blocks have linear scaling relationship with the file size. Sampling breaks this relationship between verification time and file size. At 99% confidence the verification overhead of our scheme is about 2.15 ms for any file. At 95% confidence the verification overhead of our scheme is about 1.4 ms for any file.

5 Conclusions and Future Work

In this paper, we propose a collective approach for data integrity verification with multiple third party auditors. This approach uses algebraic signature and homomorphic verification tag for data integrity verification. Benefits of algebraic signature and efficiency of homomorphic tag makes it ideally suited for cloud storage. Experiments shows that the performance bottleneck is bounded by disk I/O and not by our approach. Fortunately, when the server deletes a fraction of file, the client can detect server misbehavior with high probability by challenge a constant amount of blocks.

In near future, we planned to design protocols that supports dynamic data updating operations with less overhead.

References

1. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable Data Possession at Untrusted Stores. In: Proc. 14th ACM Conf. Computer and Comm. Security, pp. 598–609. ACM (2007)
2. Ateniese, G., Pietro, R.D., Mancini, L.V., Tsudik, G.: Scalable and efficient provable data possession. In: ACM SecureCom (2008)
3. Sebe, F., Domingo-Ferrer, J., Martinez-Balleste, A., Deswarte, Y., Quisquater, J.-J.: Efficient Remote Data Possession Checking in Critical Information Infrastructures. IEEE Trans. Knowledge and Data Eng. 20(8), 1034–1038 (2008)
4. Erway, C., Kupcu, A., Papamanthou, C., Tamassia, R.: Dynamic Provable Data Possession. In: Proc. 16th ACM Conf. Computer and Communication Security (CCS 2009), pp. 213–222 (2009)
5. Chen, L.: Using algebraic signatures to check data possession in cloud storage. Future Generation Computer Systems (December 2012)
6. Juels, A., Kaliski, B.S.: PORs: Proofs of retrievability for large files. In: ACM Conf. Computer and Comm. Security (2007)
7. Shacham, H., Waters, B.: Compact Proofs of Retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008)
8. Dodis, Y., Vadhan, S.P., Wichs, D.: Proofs of retrievability via hardness amplification. In: ACM TCC-2009, pp. 109–127 (2009)
9. Bowers, K.D., Juels, A., Oprea, A.: Proofs of retrievability: Theory and implementation. In: ACM Workshop on Cloud Computing Security, pp. 43–45 (2009)

10. Bowers, K.D., Juels, A., Oprea, A.: HAIL: A high-availability and integrity layer for cloud storage. In: Proc. 16th ACM Conference on Computer and Communications Security, pp. 187–198 (2009)
11. Wang, C., Wang, Q., Ren, K., Lou, W.: Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. In: Proc. IEEE INFOCOM. IEEE (2010)
12. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing. IEEE Transactions on Parallel and Distributed Systems 22(5), 847–859 (2011)
13. Hao, Z., Zhong, S., Yu, N.: A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability. IEEE Transactions on Knowledge and Data Engineering 23(9) (September 2011)
14. Zhu, Y., Wang, H., Hu, Z., Ahn, G.-J., Hu, H., Yau, S.S.: Cooperative Provable Data Possession. In: Cryptology ePrint Archive, Report 2012/234, pp. 257–265 (2012)
15. Schwarz, T.J.E., Miller, E.L.: Store, forget, and check: using algebraic signatures to check remotely administered storage. In: Proc. of ICDCS 2006, p. 12 (2006)
16. Litwin, W., Schwarz, T.J.E.: Algebraic signatures for scalable, distributed data structures. In: ICDE 2004, Boston, MA, pp. 412–423 (2004)
17. Chen, L., Guo, G.: An efficient remote data possession checking in cloud storage. JDCTA: International Journal of Digital Content Technology and its Applications 5(4), 43–50 (2011)
18. Saxena, R., Ruj, S., Sarma, M.: Collaborative Model for Privacy Preservation and Data Integrity Verification in Cloud Computing. In: Proceedings of the Security and Privacy Symposium, IIT Kanpur, Kanpur, India, February 28-March 2 (2013)